Effective and annotation efficient deep learning for image understanding

Spyridon Gidaris

École Doctorale Paris-Est
Mathématiques & Sciences et Technologies de l'Information et de la Communication

Thése de doctorat
de l'Université Paris-Est
Domaine : Traitement du Signal et des Images

Présentée par

Spyros GIDARIS

pour obtenir le grade de

Docteur de l'Université Paris-Est

Effective and Annotation Efficient Deep Learning for Image Understanding

---

Soutenue publiquement le devant le jury composé de :

Vincent LEPETIT	Professor, University of Bordeaux	Rapporteur
Patrick PEREZ	Director of Research, valeo.ai	Rapporteur
Camille COUPRIE	Researcher, Facebook (FAIR)	Examinateur
Hugues TALBOT	Professor, CentraleSupelec, Universite Paris-Saclay	Examinateur
Renaud MARLET	Director of Research, École des Ponts ParisTech	Examinateur
Nikos KOMODAKIS	Associate Professor, École des Ponts ParisTech	Directeur de thèse

Abstract

Recent development in deep learning have achieved impressive results on image understanding tasks. However, the success of deep learning based approaches on such tasks heavily depends on employing the appropriate deep neural network architecture for the task of interest and having available a large-size manually labeled dataset for training. In this context, the objective of this dissertation is to propose deep learning techniques and architectures for core image understanding tasks in order to (1) drastically improve the effectiveness (i.e., accuracy) with which those tasks are performed, and (2) make their learning process more annotation efficient, i.e., less dependent on the availability of large amounts of manually labeled training data.

We first focus on improving the state-of-the-art on object detection. More specifically, we attempt to boost the ability of object detection systems to recognize (even difficult) object instances by proposing a multi-region and semantic segmentation-aware ConvNet based representation that captures a diverse set of discriminative appearance factors. Also, we aim to improve the localization accuracy of object detection systems by proposing iterative detection schemes and a novel localization model for estimating the bounding box of the objects. We demonstrate that the proposed technical novelties lead to significant improvements in the object detection performance of PASCAL and MS COCO benchmarks.

Another core image understanding task of which we wish to improve the state-of-the-art is that of pixel-wise image labeling. Here we explore a family of deep neural network architectures that perform structured prediction by learning to (iteratively) improve some initial estimates of the output labels. The goal is to identify which is the optimal architecture for implementing such deep structured prediction models. In this context, we propose to decompose the label improvement task into three steps: detecting which initial labels are incorrect, replacing those erroneous labels with new ones, an finally refining the renewed labels by predicting residual corrections w.r.t. them. We evaluate the explored architectures on the disparity estimation task and we demonstrate that the proposed architecture achieves state-of-the-art results on the KITTI 2015 benchmark.

In order to accomplish our goal for annotation efficient learning, we propose a self-supervised learning approach that learns ConvNet-based image representations by training the ConvNet to recognize the 2d rotation that is applied to the image that it gets as input. We empirically demonstrate that this apparently simple task actually provides a very powerful supervisory signal for semantic feature learning. Specifically, the image features learned from this task exhibit very good results when transferred on the visual tasks of object detection and semantic segmentation, surpassing prior unsupervised learning approaches and thus narrowing the gap with the supervised case.

Finally, also in the direction of annotation efficient learning, we proposed a few-shot object recognition system that after its training is able to dynamically learn novel categories from only a few samples (e.g., only one or five examples) while it does not forget the categories on which it was trained. In order to implement the proposed recognition system we introduce two technical novelties, an attention based few-shot classification weight generator, and implementing the classifier of the ConvNet model as a cosine similarity function between feature representations and classification vectors. We demonstrate that the proposed approach achieves state-of-the-art results on relevant few-shot benchmarks.

Résumé

L'objectif de cette thèse est de faire progresser l'efficacité de l'analyse d'image pour les données et les applications du monde réel. Plus précisément, il se concentre sur les tâches de reconnaissance d'objets, de détection d'objets et d'étiquetage des images à l'échelle du pixel, qu'elles répondent "quoi" et "où" quelque chose est représentée dans une image. Perfectionner ce type de compréhension d'image est une condition préalable à la mise au point des systèmes d'intelligence artificielle, comme les voitures qui circulent dans les rues de la ville, les robots autonomes qui effectuent des tâches d'entretien ménager, ou des dispositifs d'aide aux personnes malvoyantes qui aident leurs utilisateurs à percevoir leur environnement. Afin d'atteindre l'objectif susmentionné, la thèse est divisée en deux parties, intitulées "Méthodes d'apprentissage profond pour l'analyse efficace d'images" (en anglais "Effective Deep Learning for Image Understanding") et "Méthodes d'apprentissage profond pour l'analyse efficace d'images en limitant l'annotation humaine" (en anglais "Annotation Efficient Deep Learning for Image Understanding"), chacun d'entre eux poursuit un sous-objectif différent.

1ère Partie: Méthodes d'apprentissage profond pour l'analayse efficace d'images

L'objectif de la première partie de la thèse est de faire progresser l'état de l'art sur les deux majeures tâches de l'analyse d'image, la détection d'objets et d'étiquetage des images à l'échelle du pixel, en proposant des approches efficaces basées sur l'apprentissage profond.

Représentations discriminantes pour la détection d'objets. Un élément central d'un système de détection d'objets est un modèle qui reconnaît si une région d'image contient ou non un objet d'intérêt. Afin d'améliorer la précision de ce modèle, nous proposons une représentation de région d'image enrichie basée sur ConvNet qui code l'apparence de

plusieurs régions (autour de la région d'image d'entrée) ainsi que des fonctionnalités de segmentation sémantique. Ceci est réalisé en concevant une architecture ConvNet multi-composants où chaque composant du réseau est forcé de se concentrer sur une région différente de l'objet d'intérêt. L'objectif est de rendre la représentation basée sur ConvNet capable de capturer un ensemble diversifié de facteurs d'apparence discriminants, telles que les caractéristiques d'apparence pure d' un objet, l'apparence distincte de ses différentes régions (parties d'objet), l'apparence du contexte, l'apparence des deux côtés des limites de l'objet, ou l'information consciente à la segmentation sémantique. Nous pensons qu'une représentation aussi riche améliorera les capacités de reconnaissance du système de détection, même lorsqu'il est confronté aux instances d'objets difficiles que l'on rencontre souvent dans la tâche de détection d'objets. De plus, la représentation obtenue a une sensibilité de localisation accrue, ce qui est essentiel pour la détection d'objets. Nous exploitons ces propriétés du module de reconnaissance proposé en l'intégrant dans un mécanisme de localisation itératif qui, à partir de certaines régions initiales de l'image, alterne entre la classification de ces régions et l'affinement de leurs emplacements afin de mieux localiser les objets d'intérêt. Grâce à l'utilisation efficace de nos modules, nous détectons les objets avec une très grande précision. Sur les défis de détection de PASCAL VOC2007 et PASCAL VOC2012, nous obtenons un mAP de $78,2\%$ et $73,9\%$ respectivement, surpassant de loin tout ouvrage publié antérieurement.

Localisation précise d'objet dans la détection d'objets. En plus de l'aspect reconnaissance, nous essayons d'améliorer la précision de localisation des systèmes de détection d'objets en concevant un nouveau modèle de localisation qui prend comme entrée une région de recherche dans une image et qui vise à localiser précisément un objet dans cette région. La majorité des approches antérieures, afin de mettre en œuvre de tels modèles de localisation, adoptent le paradigme de régression des boîtes, qui utilise une fonction de régression afin de prédire directement les coordonnées de la boîte qui entoure étroitement l'objet d'intérêt. Cependant, nous croyons qu'essayer de régresser directement les coordonnées de la boîte cible constitue une tâche d'apprentissage difficile qui ne peut donner des résultats suffisamment précis. C'est pour cette raison que nous formulons le problème de localisation avec une dense méthode de classification. Plus précisément, étant donné la

région de recherche, notre modèle attribue des probabilités conditionnelles à chaque ligne et colonne de cette région. Ces probabilités fournissent des informations utiles concernant l'emplacement des limites de l'objet à l'intérieur de la région de recherche et permettent l'inférence précise de la boîte de l'objet dans un cadre probabiliste simple.

Nous implémentons notre modèle de localisation avec une architecture ConvNet correctement adaptée, appelée LocNet, et nous l'intégrons sur une méthodologie de localisation itérative. Nous démontrons expérimentalement que LocNet présente des performances de localisation supérieures aux modèles de régression en boîtes et qu'il permet d'améliorer significativement la métrique mAP lorsque l'on considère des valeurs élevées pour le seuil IoU (intersection-over-union), i.e., une précision de localisation élevée. De plus, il peut être facilement couplé à des systèmes de l'etat de l'art en détection d'objets, ce qui leur permet d'améliorer leurs performances. Enfin, nous adaptons la méthodologie de localisation à la tâche de génération de propositions de localisation d'objets. Le système qui en résulte, appelé "AttractionNet", permet d'obtenir des résultats d'etat de l'art dans cette tâche et, lorsqu'il est couplé à un système de détection basé sur LocNet, d'obtenir une excellente performance de détectiond d'objet.

Prédiction de structure basée sur les réseaux de neurones profonds pour l'étiquetage d'images à l'échelle du pixel L'un des principaux défis de l'étiquetage des images à l'échelle du pixel est d'apprendre l'espace commun des variables d'entrée et de sortie. Une approche fondée sur les données pour l'apprentissage implicite de cet espace commun consiste à entraîner un réseau neuronal profond de sorte que, en donnant en entrée une estimation initiale des étiquettes de sortie et de l'image d'entrée, on puisse prévoir une nouvelle estimation affinée pour les étiquettes. Nous appelons ces méthodes des "modèles d'entrées-sorties conjointes et profondes". Dans ce contexte, la contribution de notre thèse est d'explorer quelle est l'architecture optimale pour réaliser cette tâche d'amélioration du label. Nous soutenons que les approches antérieures qui consistaient soit à prédire directement les nouvelles estimations des étiquettes, soit à prédire les corrections résiduelles par rapport aux étiquettes initiales à l'aide d'architectures de réseaux profonds à propagation avant sont sous-optimales. Nous proposons plutôt une architecture générique qui décompose la tâche d'amélioration des étiquettes en trois étapes : (1) détecter les estimations initiales

incorrectes des étiquettes, (2) remplacer les étiquettes incorrectes par de nouvelles étiquettes, et enfin (3) affiner les étiquettes renouvelées en prédisant les corrections résiduelles. De plus, nous explorons et comparons diverses architectures alternatives pour des “modèles d’entrées-sorties conjointes et profondes” qui se composent des composants “Détecter”, “Remplacer” et “Affiner” mentionnés ci-dessus. Nous évaluons de manière approfondie les architectures explorées dans la tâche difficile de l’estimation de la disparité (en anglais “disparity estimation” or “stereo matching”) et nous présentons les résultats quantitatifs et qualitatifs sur trois ensembles de données différents qui démontrent les avantages de notre approche. Enfin, le réseau neuronal d’estimation des disparités qui met en œuvre l’architecture générique proposée obtient d’excellents résultats sur le benchmark KITTI 2015, dépassant largement les approches précédentes.

2ème Partie: Méthodes d’apprentissage profond pour l’analyse efficace d’images en limitant l’annotation humaine

La deuxième partie de la thèse porte sur l’exploration des techniques qui permettent d’apprendre les modèles d’analyse d’images sans avoir besoin d’une grande quantité de données de formation étiquetées manuellement. Deux approches générales qui tentent de contourner la dépendance des approches fondées en apprentissage profond à l’égard d’ensembles de données de grande taille étiquetées manuellement sont l’apprentissage à l’aide de données non étiquetées (i.e., l’apprentissage non supervisé) ou l’apprentissage fondé sur des données étiquetées de problèmes différents mais semblables pour lesquels les étiquettes sont plus faciles à obtenir ou déjà disponibles (i.e., l’apprentissage par transfert). Dans notre cas, nous proposons deux approches, une approche d’apprentissage par représentation non supervisée, qui fait partie de l’approche plus large d’apprentissage non supervisé, et une approche d’apprentissages peu-instantanés (en anglais “few-shot learning”), qui fait partie de l’approche plus large d’apprentissage par transfert.

Apprentissage non supervisé de la représentation d’images. Les réseaux de neurones convolutifs (ConvNets) se sont avérés extrêmement efficaces pour résoudre les tâches d’analyse d’images grâce à leur capacité inégalée d’apprendre des représentations d’images sémantiques de haut niveau par un apprentissage supervisé. Par exemple, les représentations

d'images obtenues par l'entraînement d'un ConvNet sur les ensembles de données de classification d'images de ImageNet [129] ou Place205 [175], qui contiennent des millions d'images annotées manuellement, ont obtenu des résultats remarquables lors de leur transfert sur des tâches d'analyse d'images en aval, telles que la détection d'objets et la segmentation sémantique d'images. Compte tenu de notre objectif, une question très intéressante est de savoir si l'apprentissage de la représentation sémantique d'images est possible sans supervision humaine, i.e., sans la nécessiter d'effort d'annotation manuelle. Une approche prometteuse pour résoudre le problème posé par cette question est l'apprentissage auto-supervisé. L'apprentissage auto-supervisé est une forme d'apprentissage non supervisé qui définit une tâche de "prétexte" sans annotation en utilisant l'information visuelle présente sur les images afin de fournir un signal de supervision de substitution pour l'apprentissage de la représentation sémantique des images.

Suivant cette approche, notre contribution est de proposer d'apprendre les représentations d'images en entraînant un ConvNet à reconnaître la rotation 2D qui est appliquée à l'image qu'il reçoit en entrée. Nous démontrons qualitativement et quantitativement que cette simple tâche fournit en fait un signal de supervision très puissant pour l'apprentissage de la représentation sémantique d'images. Nous évaluons de manière exhaustive notre méthode à l'aide de différents benchmarks d'apprentissage non supervisés et nous démontrons dans chacun d'eux des performances d'etat de l'art. Plus précisément, nos résultats par rapport à ces repères révèlent des améliorations importantes par rapport aux approches antérieures pour l'apprentissage non supervisé de la représentation, ce qui réduit considérablement l'écart avec l'apprentissage supervisé des caractéristiques. Par exemple, dans la tâche de détection PASCAL VOC 2007, le modèle AlexNet qui est entraîné au préalable avec notre méthode d'apprentissage non supervisé atteint une mAP de $54,4\%$, soit seulement 2,4 points de moins que dans le cas supervisé. Nous obtenons des résultats tous aussi bons lorsque nous transférons les représentations d'images basées sur notre méthode d'apprentissage non supervisée sur diverses autres tâches, telles que la classification ImageNet, la classification PASCAL, la segmentation PASCAL, et la classification CIFAR-10.

L'apprentissage de nouvelles catégories en utilisant peu d'exemples. L'apprentissage peu-instantané (en anglais "few-shot learning") est lié au problème plus large de l'apprentissage

par transfert qui tente de stocker et d'exploiter les connaissances acquises tout en apprenant à résoudre un problème afin d'apprendre plus tard à résoudre plus efficacement un problème nouveau mais lié. Dans le "few-shot learning" en particulier, l'objectif est d'exploiter les connaissances acquises afin de réduire drastiquement le nombre d'exemples d'entraînement requis pour le nouveau problème ou, en d'autres termes, de résoudre plus efficacement le nouveau problème tout en ayant accès à très peu d'exemples d'entraînement pour ce problème. Par exemple, dans la reconnaissance d'objets, les connaissances acquises en apprenant à reconnaître les chats et les lions pourraient être exploitées en apprenant à reconnaître la nouvelle catégorie tigre à partir de seulement quelques exemples d'entraînement, e.g., un seul (1-shot) ou cinq (5-shot). Le système visuel humain fait preuve d'une telle capacité d'apprentissage par transfert ; il apprend sans effort de nouveaux concepts visuels à partir d'un ou de quelques exemples grâce à sa capacité d'exploiter son expérience passée du monde visuel. L'imitation de ce comportement sur les systèmes de vision artificielle est un problème de recherche intéressant et difficile à résoudre.

Dans ce contexte, notre contribution est de proposer un nouveau système de reconnaissance d'objets qui, après son entraînement, est capable d'apprendre dynamiquement de nouvelles catégories à partir de quelques exemples seulement (typiquement, seulement un ou cinq), sans oublier les catégories sur lesquelles il a été formé. Pour réaliser cela, nous proposons (a) d'étendre un système de reconnaissance d'objets avec une composante de réseau neuronal supplémentaire qui, à partir de quelques exemples d'entraînement d'une nouvelle catégorie, génère les poids de classification pour cette catégorie, et (b) de reconcevoir le classificateur d'un modèle ConvNet comme fonction de similarité cosinus entre la représentation d'images et des vecteurs poids de classification. Cette dernière, en plus d'unifier la reconnaissance des catégories nouvelles et initiales, conduit également à des représentations d'images qui donnent de meilleurs résultats sur les nouvelles catégories. Nous évaluons en profondeur notre approche sur MiniImageNet, où nous réussissons à améliorer l'état de l'art antérieur sur la reconnaissance "few-shot" (i.e., que nous obtenons 56.20% et 73.00% respectivement sur les réglages 1-shot et 5-shot) tout en ne sacrifiant aucune précision sur les catégories initiales, une caractéristique que la plupart des approches antérieures manquent. Nous appliquons également notre approche sur le benchmark Ima-

geNet de Hariharan et Girshick [51] où nous obtenons également des résultats de l'etat de l'art.

Contents

List of Figures

List of Tables

5.2 Exploring the quality of the self-supervised learned features w.r.t. the number of recognized rotations. For all the entries we trained a non-linear classifier with 3 fully connected layers (similar to Table 5.1) on top of the feature maps generated by the 2nd conv. block of a RotNet model with 4 conv. blocks in total. The reported results are from CIFAR-10. 181

5.3 Evaluation of unsupervised feature learning methods on CIFAR-10. The *Supervised NIN* and the *(Ours) RotNet + conv* entries have exactly the same architecture but the first was trained fully supervised while on the second the first 2 conv. blocks were trained unsupervised with our rotation prediction task and the 3rd block only was trained in a supervised manner. In the *Random Init. + conv* entry a conv. classifier (similar to that of *(Ours) RotNet + conv*) is trained on top of two NIN conv. blocks that are randomly initialized and stay frozen. Note that each of the prior approaches has a different ConvNet architecture and thus the comparison with them is just indicative. 182

5.4 Per class CIFAR-10 classification accuracy. 182

5.5 **Task Generalization: ImageNet top-1 classification with non-linear layers**. We compare our unsupervised feature learning approach with other unsupervised approaches by training non-linear classifiers on top of the feature maps of each layer to perform the 1000-way ImageNet classification task, as proposed by [111]. For instance, for the conv5 feature map we train the layers that follow the conv5 layer in the AlexNet architecture (i.e., fc6, fc7, and fc8). Similarly for the conv4 feature maps. We implemented those non-linear classifiers with batch normalization units after each linear layer (fully connected or convolutional) and without employing drop out units. All approaches use AlexNet variants and were pre-trained on ImageNet without labels except the ImageNet labels and Random entries. During testing we use a single crop and do not perform flipping augmentation. We report top-1 classification accuracy. 185

Chapter 1

Introduction

1.1 Objective

The objective of this thesis is to propose deep learning based approaches that would advance the effectiveness of machine image understanding to real-world data and applications. But what do we mean by machine image understanding? In general, machine image understanding can be defined as any machine process that, given an image, extracts a description from that image that is useful for the user of the process. Specifically in this thesis we are interested with the following image understanding tasks:

Object recognition and detection: The most simple and commonly studied image understanding task is that on which the machine process gets as input an image that is assumed to be the picture of a single object, and it has to recognize if the object belongs to one of several predefined semantic categories (e.g., dogs, cats, cars, or bikes). This type of image understanding task is typically called object recognition. Due to its assumption about the input image, the usefulness of object recognition is relatively limited in real-world applications where the processed images can depict numerous objects and in various spatial positions in the image. Therefore, a much more interesting image understanding task is that of object detection that given an image requires to find in that image all the object instances of one or more semantic categories in form of bounding boxes that tightly enclose those objects. In Figure 1-1

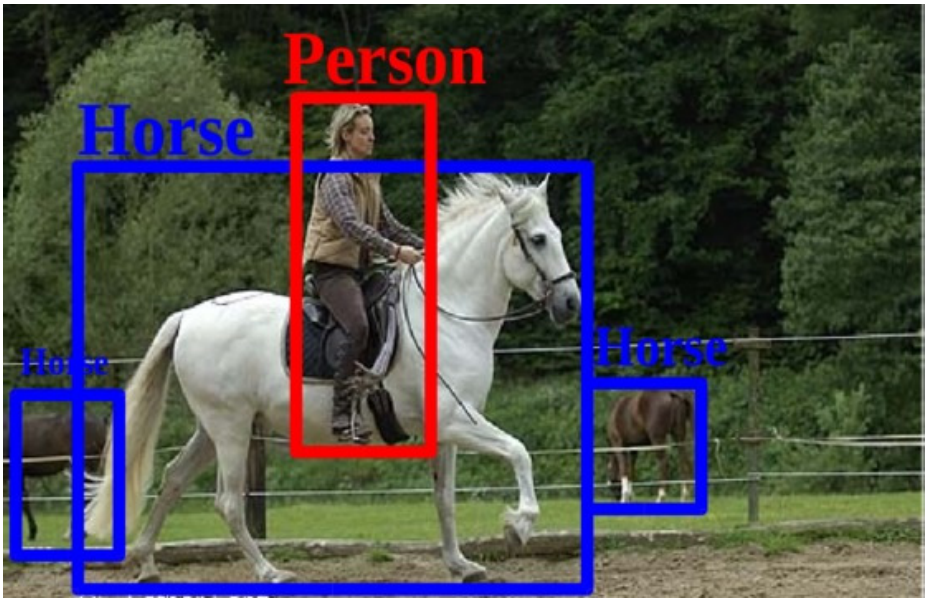


Figure 1-1: Object detection example. We draw the ground truth bounding boxes of the human and horse objects depicted in the image.

we provide an object detection example for the semantic categories person and horse.

Pixel-wise image labeling: Another very important type of image understanding tasks is that of assigning a descriptive label to each pixel of an image. For example, in the semantic segmentation task each pixel of an image is labeled with a semantic category that describes that pixel (e.g., road, person, car, or pavement; see example in Figure 1-2). Another instance of the pixel-wise image labeling task that we are interested with in this thesis is that of depth estimation from a stereo image (also called disparity estimation). In depth estimation each pixel of the left image of the stereo rig is assigned a continuous label that indicates its horizontal displacement in the right image (i.e., disparity). Those disparity maps reveal the depth (from the stereo camera) of the surface of the scene objects depicted in the image (see Figure 1-3).

It is evident that object recognition, object detection, and pixel-wise image labeling tasks are answering "what" and "where" is something depicted in an image. Perfecting such type of machine image understanding is prerequisite for being able to develop artificial intelligence systems, such as self-driving cars that navigate through city streets, autonomous robots that perform household maintenance duties, assistance devices for visually impaired people that describe the environment where their user moves, or augmented reality systems

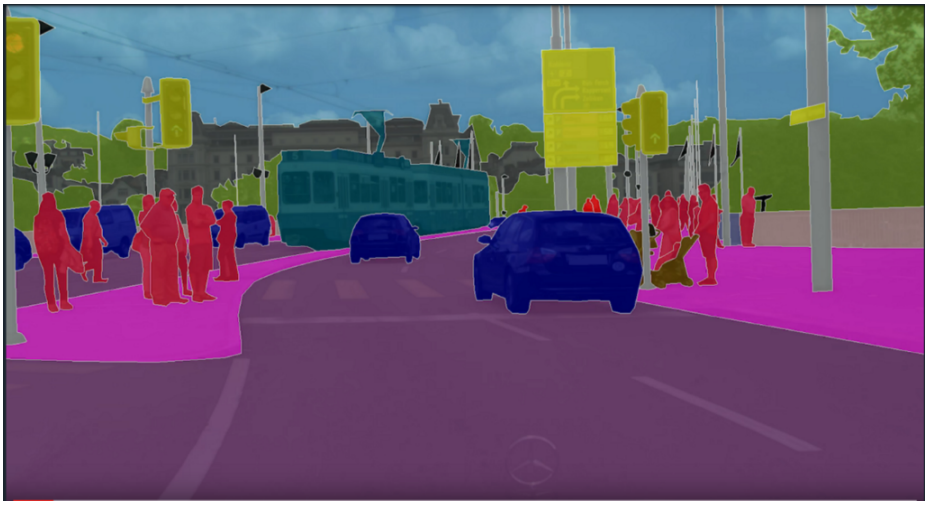


Figure 1-2: Pixel wise labeling - semantic segmentation example. We visualize the ground truth semantic label of each pixel of the image.

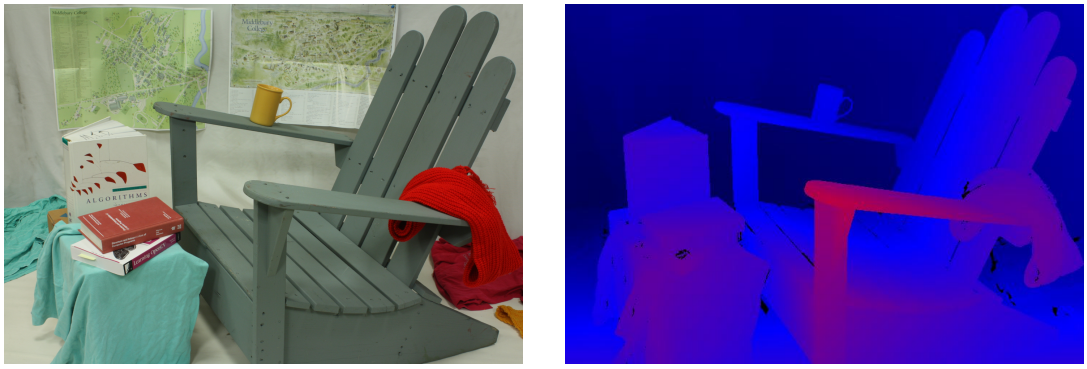


Figure 1-3: Pixel wise labeling - depth estimation example. We visualize the ground truth depth label of each pixel of the image.

that enhance or alter our visual perception with computer generated visual information.

1.2 Deep learning approach

As already mentioned, in order to accomplish our objective we employ deep learning techniques. Deep learning belongs to the broader family of machine learning techniques that learn from data the computational model that performs a certain task (as opposed to explicitly programming it). In deep learning particularly, deep neural networks, which are cascades of non-linear processing units arranged in sequential order, learn to gradually transform the inputs to more abstract and composite representations till they end up with the final outputs. For example, in the object recognition case, a deep neural network might learn a

first representation level that transforms an image to oriented edges, a second representation level that composes and encodes arrangements of oriented edges that form object parts (such as nose, eyes, car wheel, etc), and a third representation level that composes and encodes arrangements of object parts that form objects such as faces, cars, or elephants, and thus recognizing the object depicted in the input image (see Figure 1-4). The most common type of deep neural networks for learning image understanding tasks is that of convolutional neural networks (ConvNets).

The most prominent approach with which deep neural networks learn to perform a task is via supervised learning. In supervised learning, the deep neural network is trained to approximate a function that maps inputs to outputs based on example input-output pairs. This set of input-output pairs (aka training examples) consist the training data for learning and are obtained by manually labeling input data with the desired outputs according to the definition of the task of interest. For instance, in object recognition a training example is an image and its semantic category label, in object detection a training example is an image and a list that includes the bounding box coordinates and the semantic category labels of each object of interest in the image, and in pixel-wise image labeling a training example is an image and the ground truth label values of each pixel. The goal is after training, the deep neural network to be able to generalize well on new input data, for which their outputs are unknown, and provide for them a good estimate of their ground truth outputs.

1.3 Challenges

Recent developments in supervised deep learning have achieved impressive results on learning image understanding tasks. For instance, the (relatively simple) object recognition task can be practically solved now if the proper ConvNet architecture is employed and enough labeled training data are available. However, devising and deploying effective deep learning based approaches for more complicated image understanding problems, such as object detection and pixel-wise image labeling, is far from trivial. Furthermore, even for object recognition, collecting large-size labeled datasets is a very laborious effort that limits the employment of machine image understanding models in real world data and applications.

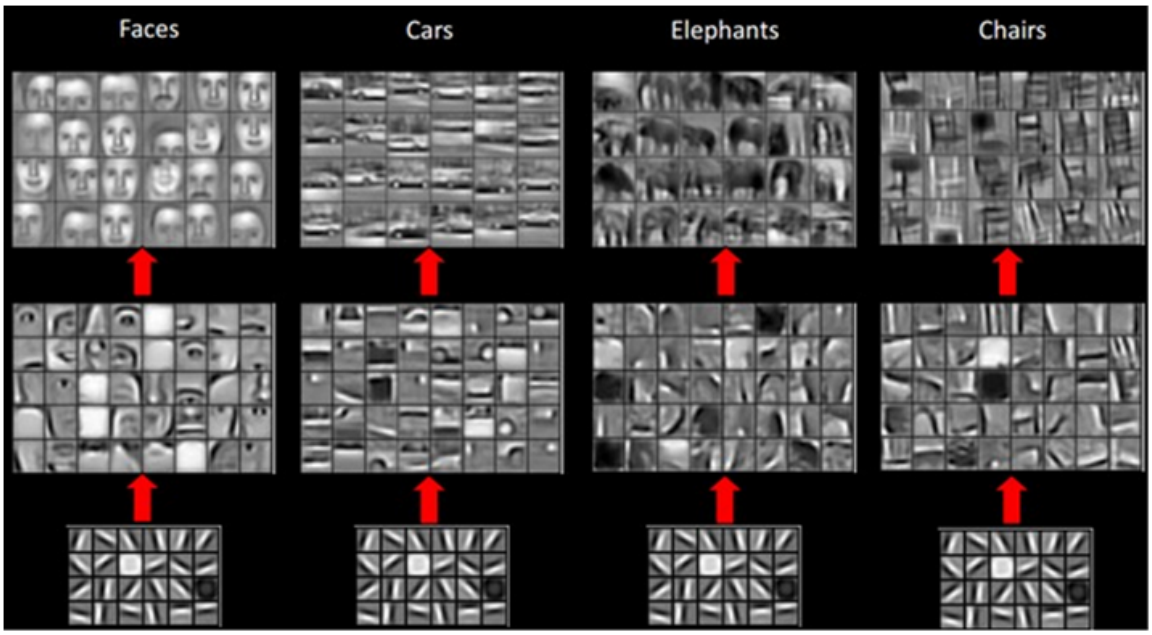


Figure 1-4: Representation hierarchies learned by a deep neural network (source [85]).

Figure 1-5: Instances of the semantic object motorbike (indicated by red bounding boxes). The motorbike in the red box of first (from the left) image is easy to be recognized. In contrast, the motorbikes in the red boxes of the remaining three images are much more difficult to be recognized due to the cluttered environment (second image), heavy occlusions (third image), or being on the background of the image (fourth image).

In the remaining of this section we discuss in more detail those challenges.

1.3.1 Object detection challenges

1.3.1.1 Detecting difficult object instances

Detecting objects in an image requires at minimum a recognition model that given an image region (i.e., a rectangular image patch) predicts whether or not it contains an object of interest. Despite the fact that ConvNet based approaches have achieved remarkable results on object recognition benchmarks, the problem of recognizing objects in real-world object detection applications is much more difficult. Specifically, most object recognition benchmarks include images on which the objects are depicted in iconic-view. That means that the objects

are on the foreground of the image, unobstructed by other items, and in neatly composed scenes (see for example the motorbike instance in the first from the left image of Figure 1-5). Such object instances are very easy to be recognized by ConvNet based recognition models. However, an object detection system should be able to process images from everyday life scenes, which means that its recognition model should be able to recognize object instances that might be in the background of the image, in "tricky" viewpoints, significantly occluded, or in cluttered environments (see for example the motorbike instances in the last three from the left images of Figure 1-5). Recognizing such difficult object instances is a much more challenging task that requires more powerful ConvNet based image representations. Even more, in object detection the recognition model must be localization sensitive, in the sense that if a given image region depicts an object but without localizing it accurately enough in order to be considered that it detects it, then the recognition model should classify that image region as negative (i.e., that it does not detect the object of interest). For instance, in PASCAL VOC [33] detection challenge an image region is considered to detect an object if the Intersection over Union (IoU) between the bounding box of the image region and the ground truth bounding box of the object is greater or equal to 0.5. Making ConvNet based recognition models exhibit such localization sensitivity is a challenging problem due to the built-in localization invariances of ConvNet models, which stem from the use of max-pooling or other similar down-sampling layers.

1.3.1.2 Accurate object localization in object detection

Achieving accurate object localization is the ultimate goal of object detection and a very daunting problem in practice. For instance, addressing the object localization aspect of object detection by naively examining all possible locations (i.e., box sizes, aspect ratios, and 2D positions) with a recognition model is computationally prohibitive and likely to generate many spurious detections. Instead, most prior approaches detect objects by classifying (with the recognition model) and refining (via bounding box regression) a few candidate bounding boxes. Those initial candidate bounding boxes are generated either by sliding window schemes or most commonly by other algorithmic components designed to generate for a given image a set of bounding boxes that cover with high recall all the objects that appear

in the image regardless of their semantic category, i.e., category agnostic box proposal algorithms. However, if the initial candidate bounding boxes miss an object, i.e., if there is no box proposal in the proximity of the object of interest, then the detection system would fail to detect it. Therefore, in order to address this issue, considerable efforts must be given in developing effective box proposal algorithms that rarely miss an object and / or detection systems that are more robust with respect to quality of the initial bounding boxes.

Furthermore, although many detection benchmarks, such as in PASCAL VOC [33], decide whether an object has been successfully detected using loose localization criteria (e.g., in PASCAL VOC, a detection threshold of 0.5 IoU is used for deciding whether an object has been successfully detected), in real life applications a higher localization accuracy (e.g., IoU $\geqslant 0.7$) is normally required. Such a need is also reflected in the recently introduced COCO detection challenge [91], which uses as evaluation metric the traditional average precision (AP) measurement but averaged over multiple IoU thresholds between 0.5 (loosely localized object) and 1.0 (perfectly localized object) so as to reward detectors that exhibit good localization accuracy. Devising ConvNet-based detectors that exhibit such highly accurate (and not loose) localization of ground truth objects makes the localization aspect of object detection even more challenging.

1.3.2 Structured prediction in pixel-wise image labeling

Differently from the object recognition and object detection problems, in pixel-wise image labeling problems there is rich structure not only on the input images but also on the output labels. For example, see in Figure 1-3 how the output disparity labels form continuous surfaces when their corresponding image pixels belong to the same object or how they discontinue across object boundaries. This means that the output variables (pixel labels) interrelate not only with the input variables (image pixels) but also with other (nearby) output variables. Therefore, in order for a pixel-wise image labeling algorithm to be able to achieve accurate and precise labeling results, it has to consider the dependencies that exist in the joint space of both the input and the output variables.

Deep learning approaches that implement the pixel-wise image labeling task by simply

employing independent ConvNet based patch predictors [34, 92, 37, 101, 110], which directly predict each pixel label given as input an image patch centered on it, cannot capture those joint dependencies. In order to model such joint dependencies, several approaches combine independent pixel-wise ConvNet predictors with Conditional Random Fields (CRFs) [80, 73] that refine and disambiguate their predictions [135, 11, 174, 12]. CRFs employ graphical models that encode the known structure of the label / output space with pairwise edge potentials between the graph nodes of output variables, and predict the image labels by performing maximum a posteriori inference in this graphical model. For example, in the case of semantic segmentation, those pairwise potentials enforce label consistency among similar or spatially adjacent pixels. However, a major drawback of most CRF based approaches is that the pairwise potentials have to be carefully hand designed in order to incorporate simple human assumptions about the structure of the output labels and at the same time to allow for tractable inference. Instead, it would be more interesting and practical to be able learn the joint structure of both input and output variables in a data-driven way.

1.3.3 Dependence on large volumes of annotated training data

Deep learning successes on image understanding tasks hugely depend on the availability of massive amounts of manually labeled training data. However, having humans annotate such large set of data is error prone, expensive, and very slow. Furthermore, for some types of visual data, such as medical data, there is lack of qualified human experts that are able to annotate them. In contrast, there might be vast amounts of available unlabeled visual data (e.g., 350 million images are uploaded on Facebook daily and 65 hours of video are uploaded on YouTube per minute) that would remain unexploited if human supervision is prerequisite. Even more, it is impractical to constantly have to annotate big volumes of new visual data whenever the visual environment that an image understanding model perceives change (e.g., in case of autonomous robots) or whenever new visual concepts need to be taken into account (e.g., introducing to a recognition model novel semantic categories that need to be recognized). Therefore, it would be desirable to being able to learn effective image understanding models without requiring massive amount of manually labeled training

data.

1.4 Thesis structure and contributions

Given the nature of the above challenges, we analyze our objective into two sub-objectives, named "effective deep learning for image understanding" and "annotation efficient learning". Therefore, we break our thesis into two parts with the subject of each of them being the pursuit of the corresponding sub-objective. In the remaining of this section, we describe the objective of each part, we introduce our work towards achieving it, and we highlight the contributions.

1.4.1 Part 1: Effective deep learning for image understanding

The objective of the first part of the thesis is to make progress to the state-of-the-art of two core image understanding problems, object detection and pixel-wise image labeling, by proposing effective deep learning based approaches.

1.4.1.1 Discriminative representations for object detection

As already explained, a core component of an object detection system is a recognition model that given an image region recognizes whether or not it tightly encloses an object of interest. In order to improve the accuracy of this model, we propose an enriched ConvNet-based image region representation that encodes the appearance of multiple regions (around the input image region) as well as semantic segmentation aware features. This is achieved by designing a multi-component ConvNet architecture where each network component is forced to focus on a different region of the object of interest. The goal is to make the learned representation to be able to capture a diverse set of discriminative appearance factors, such as its pure appearance characteristics, the distinct appearance of its different regions (object parts), context appearance, the joint appearance on both sides of the object boundaries, or semantic segmentation aware information. We believe that such a rich representation will improve the recognition capabilities of the detection system even when faced with

the difficult object instances that are often encountered in the object detection task (see discussion in 1.3.1.1). Furthermore, the learned representation exhibits increased localization sensitivity, which is essential in object detection. We exploit these properties of the proposed recognition module by integrating it on an iterative localization mechanism that starting from some initial candidate regions in the image alternates between classifying them and refining their locations such that they more tightly enclose the objects of interest. Thanks to the efficient use of our modules, we detect objects with very high accuracy. On the detection challenges of PASCAL VOC2007 and PASCAL VOC2012 we achieve mAP of 78.2% and 73.9% correspondingly, surpassing by a significant margin any prior or contemporaneous published work.

This work was accepted for publication at ICCV 2015. Implementation code and models are published at `https://github.com/gidariss/mrcnn-object-detection`.

1.4.1.2 Accurate object localization in object detection

Apart from the recognition aspect, we attempt to boost the localization accuracy of object detection systems by devising a novel localization model that, given a loosely localized search region inside an image, aims to return the accurate location of an object in this region. Most prior approaches, in order to implement such localization models, adopt the bounding box regression paradigm, which uses a regression function to directly predict the four object bounding box coordinates. However, we believe that trying to directly regress the target bounding box coordinates, constitutes a difficult learning task that cannot yield accurate enough bounding boxes. Instead we formulate the localization problem in a dense classification way. Specifically, given the search region our model assigns conditional probabilities to each row and column of this region, where these probabilities provide useful information regarding the location of the boundaries of the object inside the search region and allow the accurate inference of the object bounding box under a simple probabilistic framework.

We implement our localization model with a properly adapted ConvNet architecture, called LocNet, and we incorporate it on an iterative localization methodology. We show experimentally that LocNet exhibits superior localization performance to bounding box

regression models, achieves a very significant improvement on the mAP for high IoU threshold on PASCAL VOC2007 test set, and that it can be very easily coupled with recent state-of-the-art object detection systems, helping them to boost their performance. We also demonstrate that our detection approach can achieve high detection accuracy even when it is given as input a set of sliding windows, thus proving that it can be independent of box proposal methods. Finally, we adapt the overall localization methodology to the box proposal generation task and the resulting system, called "AttractionNet", achieves state-of-the-art box proposal results that when coupled with a LocNet based detection system achieve excellent detection performance.

Parts of this work were accepted for publication at CVPR 2016 and BMVC 2016. Implementation code and relevant data are published at `https://github.com/gidariss/LocNet` and at `https://github.com/gidariss/AttractioNet`.

1.4.1.3 Deep structure prediction for pixel-wise image labeling

As already explained, one of the main challenges of pixel-wise image labeling is to learn the joint space of both input and output variables. A data-driven approach for implicitly learning this joint space is by training a deep neural network such that, given as input an initial estimate of the output labels and the input image, it will be able to predict a new refined estimate for the labels. We refer to these methods as *deep joint input-output models*. In that context, the contribution of our thesis on the pixel-wise image labeling problem is on exploring what is the optimal architecture for performing the label improvement task. We argue that the prior approaches of either directly predicting new label estimates or predicting residual corrections w.r.t. the initial labels with feed-forward deep network architectures are sub-optimal. Instead, we propose a generic architecture that decomposes the label improvement task to three steps: **(1)** *detecting* the initial label estimates that are incorrect, **(2)** *replacing* the incorrect labels with new ones, and finally **(3)** *refining* the renewed labels by predicting residual corrections w.r.t. them. Furthermore, we explore and compare various other alternative architectures for deep joint input-output models that consist of the aforementioned *Detection*, *Replace*, and *Refine* components. We extensively evaluate the examined architectures in the challenging task of dense disparity estimation (stereo

matching) and we report both quantitative and qualitative results on three different datasets that demonstrate the advantages of our approach. Finally, our dense disparity estimation network that implements the proposed generic architecture, achieves state-of-the-art results on the KITTI 2015 benchmark surpassing prior approaches by a significant margin. We also provide preliminary results of our approach in two semantic segmentation tasks, the Cityscapes and the ECP facade parsing tasks, obtaining very promising experimental results.

This work was accepted for publication at CVPR 2017.

1.4.2 Part 2: Annotation efficient deep learning for image understanding

The second part of the thesis focuses on exploring techniques that will allow to learn image understanding models without requiring extensive amount of manually labeled training data, or as we call it, in an annotation efficient learning way. Two broad approaches that try to circumvent the dependence of deep learning models on large-size manually labeled datasets are learning using unlabeled data (i.e, unsupervised learning) or learning using labeled data of different but similar problems for which labels are easier to obtain or already available (i.e., transfer learning). In our case, we propose two approaches for annotation efficient learning, an unsupervised representation learning approach, which belongs to the broader unsupervised learning approach, and a few-shot learning approach, which belongs to the broader transfer learning approach.

1.4.2.1 Unsupervised visual representation learning

ConvNets have been proven extremely successful at solving image understanding tasks thanks to their unparalleled ability to learn high level semantic image features through supervised learning. For instance, the image features learned by training a ConvNet on the image classification datasets of ImageNet [129] or Place205 [175], which contain millions of manually annotated images, have achieve remarkable results when transferred on downstream image understanding tasks, such as object detection and semantic segmentation. Given our goal for annotation efficient learning, a very interesting question is whether

semantic visual representation learning is possible without human supervision, i.e., without requiring any manual annotation effort. A promising approach for the problem posed by this question is self-supervised learning, which is a form of unsupervised learning that defines an annotation free pretext task, using only the visual information present on the images, in order to provide a surrogate supervision signal for semantic feature learning.

Following this approach, our contribution is to propose to learn image representations by training ConvNets to recognize the 2D rotation that is applied to the image that it gets as input. We demonstrate both qualitatively and quantitatively that this apparently simple task actually provides a very powerful supervisory signal for semantic feature learning. We exhaustively evaluate our method in various unsupervised feature learning benchmarks and we exhibit in all of them state-of-the-art performance. Specifically, our results on those benchmarks demonstrate dramatic improvements w.r.t. prior state-of-the-art approaches in unsupervised representation learning and thus significantly close the gap with supervised feature learning. For instance, in PASCAL VOC 2007 detection task our unsupervised pre-trained AlexNet model achieves the state-of-the-art (among unsupervised methods) mAP of 54.4% that is only 2.4 points lower from the supervised case. We get similarly striking results when we transfer our unsupervised learned features on various other tasks, such as ImageNet classification, PASCAL classification, PASCAL segmentation, and CIFAR-10 classification.

This work was accepted for publication at ICLR 2018. Implementation code and trained models are published at `https://github.com/gidariss/FeatureLearningRotNet`.

1.4.2.2 Few-shot visual learning without forgetting

Few-shot learning is related to the broader transfer learning problem that attempts to store and exploit the knowledge acquired while learning to solve one problem in order later on to more efficiently learn to solve a different / novel but related problem. In the few-shot learning specifically, the goal is the acquired knowledge to be exploited in order to drastically reduce the amount training examples required for the novel problem or in other words to more effectively solve the novel problem while having access to very few training examples for that problem. For example, in the object recognition application, the knowledge acquired

while learning to recognize cats and lions could be exploited when learning to recognize the novel category tiger from only a few training examples of tigers, e.g., only one (1-shot) or five (5-shot) training examples. The human visual system exhibits such transfer learning ability; it effortlessly learns novel visual concepts from only one or a few examples thanks to its ability to exploit its past experience about the visual world. Mimicking that behavior on artificial vision systems is an interesting and challenging research problem. Solving it moves us towards the direction of annotation efficient learning.

In this context, our contribution is to propose a few-shot visual learning system that is capable of dynamically learning novel categories from only a few training data (e.g., 1 or 5 training examples per category) while at the same time is not forgetting the initial categories on which it was trained (here called base categories). In order to achieve that we propose (a) to extend an object recognition system with an attention based few-shot classification weight generator, and (b) to redesign the classifier of a ConvNet model as the cosine similarity function between feature representations and classification weight vectors. The latter, apart from unifying the recognition of both novel and base categories, also leads to feature representations that generalize better on "unseen" categories. We extensively evaluate our approach on Mini-ImageNet where we manage to improve the prior state-of-the-art on few-shot recognition (i.e., we achieve 56.20% and 73.00% accuracy on the 1-shot and 5-shot settings respectively) while at the same time we do not sacrifice any accuracy on the base categories, which is a characteristic that most prior approaches lack. Finally, we apply our approach on the recently introduced few-shot benchmark of Hariharan and Girshick [51] where we also achieve state-of-the-art results.

This work was accepted for publication at CVPR 2018. Implementation code and relevant data are published at `https://github.com/gidariss/FewShotWithoutForgetting`.

1.5 Publications

The work during this PhD lead to the following publications:

- Spyros Gidaris and Nikos Komodakis. "Object detection via a multi-region and semantic segmentation-aware cnn model." Proceedings of the IEEE International

Conference on Computer Vision (ICCV), 2015.

- Spyros Gidaris and Nikos Komodakis. "Locnet: Improving localization accuracy for object detection." Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2016.
- Spyros Gidaris and Nikos Komodakis. "Attend Refine Repeat: Active Box Proposal Generation via In-Out Localization." Proceedings of the British Machine Vision Conference (BMVC), 2016.
- Spyros Gidaris and Nikos Komodakis. "Detect, replace, refine: Deep structured prediction for pixel wise labeling." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations." International Conference on Learning Representations (ICLR), 2018.
- Spyros Gidaris and Nikos Komodakis. "Dynamic Few-Shot Visual Learning without Forgetting." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

Also, under preparation for submission are the following journal papers:

- An extended version of our ICLR 2018 paper, "Unsupervised Representation Learning by Predicting Image Rotations".
- An extended version of our CVPR 2018 paper, "Dynamic Few-Shot Visual Learning without Forgetting".
- Extended versions of our CVPR 2016 and BMVC 2016 papers combined into a single journal paper.

1.6 Outline

As already explained, this thesis is organized into two parts. The first part includes chapters 2, 3, and 4; in chapter 2 we present our work on devising a discriminative image representation for the object detection task, in chapter 3 we focus on the localization aspect of object detection and we propose a novel object localization model capable of boosting the

localization accuracy of object detectors, and in chapter 4 we present our work on exploring and devising deep structured prediction models for pixel-wise image labeling problems. The second part includes chapters 5 and 6; in chapter 5 we present our unsupervised visual representation learning approach, and in chapter 6 we present our few-shot visual learning system. Finally, we conclude our thesis in chapter 7 where we also present possible avenues for future work.

Part 1

Effective Deep Learning for Image Understanding

Chapter 2

Discriminative Representations for Object Detection

2.1 Introduction

In this chapter we deal with the object detection task. Over the past few years, tremendous progress has been achieved on the task of object detection thanks to the recent advances of deep learning community [83, 6, 58, 78, 144]. Among them, most notable is the work of Sermanet et al. [137] with the Overfeat framework and the work of Girshick et al. [43] with the R-CNN framework.

Overfeat [137] uses two CNN models that are applied on a sliding window fashion on multiple scales of an image. The first is used to classify if a window contains an object and the second to predict the true bounding box location of the object. Finally, the dense class and location predictions are merged with a greedy algorithm in order to produce the final set of object detections.

R-CNN [43] uses Alex Krizhevsky's Net [78] to extract features from box proposals provided by selective search [153] and then classifies them with class specific linear SVMs. The authors manage to train networks with millions of parameters by first pre-training on the auxiliary task of classifying the images of ImageNet dataset [129] and then fine-tuning on a small set of images annotated for the detection task. This simple pipeline surpasses by a large margin the detection performance of all the previously published systems, such as

Figure 2-1: Left: detecting the sheep on this scene is very difficult without referring on the context, mountainish landscape. **Center:** In contrast, the context on the right image can only confuse the detection of the boat. The pure object characteristics is what a recognition model should focus on in this case. **Right:** This car instance is occluded on its right part and the recognition model should focus on the left part in order to confidently detect.

deformable parts models [35] or non-linear multi-kernel approaches [154]. Their success comes from the fact that they replaced the hand-engineered features like HOG [21] or SIFT [93] with the high level object representations produced from the last layer of a CNN model. By employing an even deeper CNN model, such as the 16-layers VGG-Net [144], they boosted the performance another 7 points.

In this chapter we aim to further advance the state-of-the-art on object detection by improving on two key aspects that play a critical role in this task: object representation and object localization.

Object representation. One of the lessons learned from the above-mentioned works is that indeed powerful representations are essential on object detection. However, instead of proposing only a network architecture that is deeper, here we also opt for an architecture of greater width, i.e., one whose last hidden layers provide features of increased dimensionality. In doing so, our goal is to build a richer and more discriminative candidate box representation. This goal is accomplished at two levels:

(1). At a first level, we want our object representation to capture several different aspects of an object such as its pure appearance characteristics, the distinct appearance of its different regions (object parts), context appearance, the joint appearance on both sides of the object boundaries, and semantics. We believe that such a rich representation will further facilitate the problem of recognizing (even difficult) object instances under a variety of circumstances (like, e.g., those depicted in Figure 2-1). In order to achieve our goal, we propose a multi-component CNN model, called *multi-region CNN* hereafter, each

component of which is steered to focus on a different region of the object thus enforcing diversification of the discriminative appearance factors captured by it.

Additionally, as we will explain shortly, by properly choosing and arranging some of these regions, we aim also to help our representation in being less invariant to inaccurate localization of an object. Note that this property, which is highly desirable for detection, contradicts with the built-in invariances of CNN models, which stem from the use of max-pooling layers.

(2). At a second level, inspired by the close connection that exists between segmentation and detection, we wish to enrich the above representation so that it also captures semantic segmentation information. To that end, we extend the above CNN model such that it also learns novel CNN-based semantic segmentation-aware features. Importantly, learning these features (i.e., training the extended unified CNN model) do not require having ground truth object segmentations as training data.

Object localization. Besides object representation, our work is also motivated from the observation that, due to the tremendous classification capability of the recent CNN models [78, 167, 144, 66, 54, 147], the bottleneck for good detection performance is now the accurate object localization. Indeed, it was noticed on R-CNN [43] that the most common type of false positives is the mis-localized detections. They fix some of them by employing a post processing step of bounding box regression that they apply on the final list of detections. However, their technique only helps on small localization errors. We believe that there is much more space for improvement on this aspect. In order to prove it, we attempt to build a more powerful localization system that relies on combining our multi-region CNN model with a CNN-model for bounding box regression, which are used within an iterative scheme that alternates between scoring candidate boxes and refining their coordinates.

To summarize, the contributions of our work presented in this chapter, are as follows:

- We develop a multi-region CNN recognition model that yields an enriched object representation capable to capture a diversity of discriminative appearance factors and to exhibit localization sensitivity that is desired for the task of accurate object localization.
- We furthermore extend the above model by proposing a unified neural network

architecture that also learns semantic segmentation-aware CNN features for the task of object detection. These features are jointly learnt in a weakly supervised manner, thus requiring no additional annotation.

- We show how to significantly improve the localization capability by coupling the aforementioned CNN recognition model with a CNN model for bounding box regression, adopting a scheme that alternates between scoring candidate boxes and refining their locations, as well as modifying the post-processing step of non-maximum-suppression.
- Our detection system achieves mAP of 78.2% and 73.9% on VOC 2007 [33] and VOC2012 [32] detection challenges respectively, thus surpassing by a very significant margin the previous state-of-the-art. (at the time of conducting the work presented in this chapter).

The remainder of the chapter is structured as follows: We discuss related work in §2.2. We describe our multi-region CNN model in §2.3. We show how to extend it to also learn semantic segmentation-aware CNN features in §2.4. Our localization scheme is described in §2.5 and implementation details are provided in §2.6. We present experimental results in §2.7, qualitative results in §2.8 and conclude in §2.9.

2.2 Related Work

Apart from Overfeat [137] and R-CNN [43], several other recent papers are dealing with the object detection problem using deep neural networks. One is the work of Zhu et al. [176], which shares some conceptual similarities with ours. Specifically, they extract features from an additional region in order to capture the contextual appearance of a candidate box, they utilize a MRF inference framework to exploit object segmentation proposals (obtained through parametric min-cuts) in order to improve the object detection accuracy, and also use iterative box regression (based on ridge regression). More than them, we use multiple regions designed to diversify the appearance factors captured by our representation and to improve localization, we exploit CNN-based semantic segmentation-aware features (integrated in a unified neural network architecture), and make use of a deep CNN model for bounding box regression, as well as a box-voting scheme after non-max-suppression. Feature

extraction from multiple regions has also been exploited for performing object recognition in videos by Leordeanu et al. [87]. As features they use the outputs of HOG [21]+SVM classifiers trained on each region separately and the 1000-class predictions of a CNN pre-trained on ImageNet. Instead, we fine-tune our deep networks on each region separately in order to accomplish our goal of learning deep features that will adequately capture their discriminative appearance characteristics. Furthermore, our regions exhibit more variety on their shape that, as we will see in section 2.3.1, helps on boosting the detection performance. Szegedy et al. [149] designed a deep CNN model for object proposals generation and use contextual features extracted by applying on large crops of the image a CNN model pre-trained on the ImageNet classification task. Ouyang et al. [113] introduce a deep CNN with a novel deformation constrained pooling layer, a new strategy for pre-training that uses the bounding box annotations provided from ImageNet localization task, and contextual features derived by applying a pre-trained on ImageNet CNN on the whole image and treating the 1000-class probabilities for ImageNet objects as global contextual features. In the SPP-Net detection framework [55], instead of applying their deep CNN on each candidate box separately as R-CNN does, they extract the convolutional feature maps from the whole image, project the candidate boxes on them, and then with an adaptive max-pooling layer, which consists of multiple pooling levels, they produce fixed length feature vectors that they pass through the fully connected layers of the CNN model. Thanks to those modifications, they manage to speed up computation by a considerable factor while maintaining high detection accuracy. Our work adopts this processing paradigm.

Contemporary to our work are the approaches of [128, 42, 127] that are also based on the SPP-Net framework. Ren et al. [128] improve the SPP framework by replacing the sub-network component that is applied on the convolutional features extracted from the whole image with a deeper convolutional network. The Fast R-CNN framework of Girshick [42] simplifies the training phase of SPP-Net and R-CNN and speeds up both the testing and the training phases. Also, by fine-tuning the whole network and adopting a multi-task objective that has both box classification loss and box regression loss, its manages to improve the detection accuracy of the system. Finally, Shaoqing et al. [127] propose the Faster R-CNN framework that extends Fast R-CNN [42] by adding a new sub-network

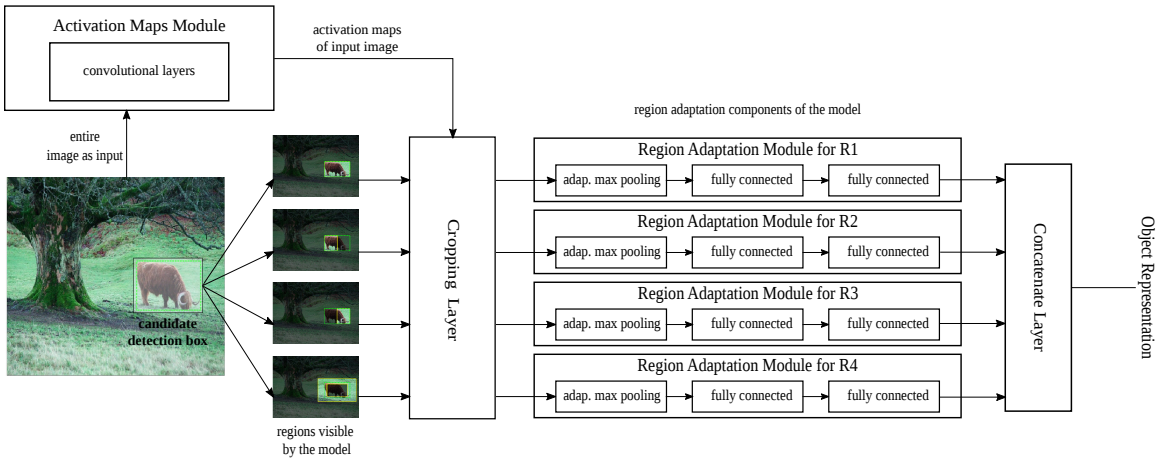


Figure 2-2: Multi Region CNN architecture. For clarity we present only four of the regions that participate in it. An "adaptive max pooling" layer uses spatially adaptive pooling as in [55] (but with a one-level pyramid). The above architecture can be extended to also learn semantic segmentation-aware CNN features (see section §2.4) by including additional 'activation-maps' and 'region-adaptation' modules that are properly adapted for this task.

component for predicting class-independent proposals and thus making the system both faster and independent of object proposal algorithms.

2.3 Multi-Region CNN Model

The recognition model that we propose consists of a multi-component CNN network, each component of which is chosen so as to focus on a different region of an object. We call this a Multi-Region CNN model. We begin by describing first its overall architecture. To that end, in order to facilitate the description of our model we introduce a general CNN architecture abstraction that decomposes the computation into two different modules:

Activation maps module. This part of the network gets as input the entire image and outputs activation maps (feature maps) by forwarding it through a sequence of convolutional layers.

Region adaptation module. Given a region R on the image and the activation maps of the image, this module projects R on the activation maps, crops the activations that lay inside it, pools them with a spatially adaptive (max-)pooling layer [55], and then forwards them through a multi-layer network.

Under this formalism, the architecture of the Multi-Region CNN model can be seen in Figure 2-2. Initially, the entire image is forwarded through the activation maps module. Then, a candidate detection box B is analyzed on a set of (possibly overlapping) regions $\{R_i\}_{i=1}^{k}$ each of which is assigned to a dedicated region adaptation module (note that these regions are always defined relatively to the bounding box B). As mentioned previously, each of these region adaptation modules passes the activations pooled from its assigned region through a multilayer network that produces a high level feature. Finally, the candidate box representation is obtained by concatenating the last hidden layer outputs of all the region adaptation modules.

By steering the focus on different regions of an object, our aim is: (i) to force the network to capture various complementary aspects of the object's appearance (e.g., context, object parts, etc.), thus leading to a much richer and more robust object representation, and (ii) to also make the resulting representation more sensitive to inaccurate localization (e.g., by focusing on the border regions of an object), which is also crucial for object detection.

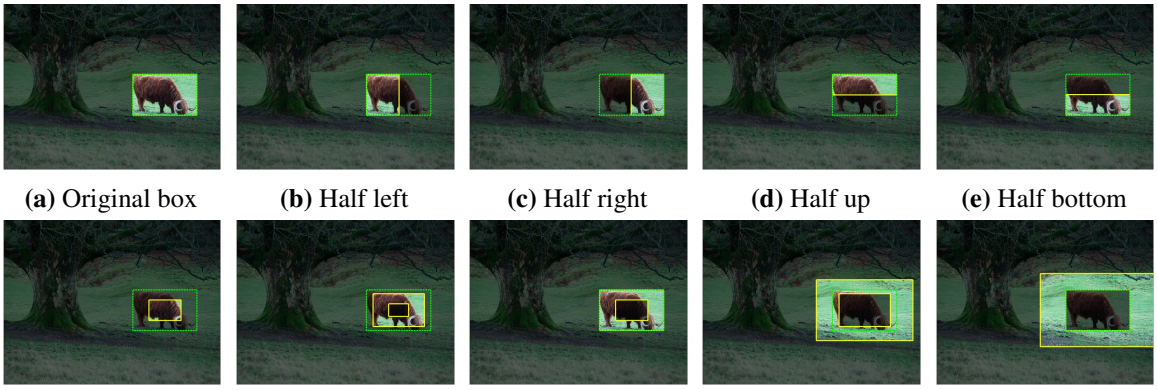
In the next section we describe how we choose the regions $\{R_i\}_{i=1}^{k}$ to achieve the above goals, and also discuss their role on object detection.

2.3.1 Region components and their role on detection

We utilize 2 types of region shapes: rectangles and rectangular rings, where the latter type is defined in terms of an inner and outer rectangle. We describe below all of the regions that we employ, while their specifications are given in the caption of Figure 2-3.

Original candidate box: this is the candidate detection box itself as being used in R-CNN [43] (Figure 2-3a). A network trained on this type of region is guided to capture the appearance information of the entire object. When it is used alone, it consists the baseline of our work.

Half boxes: those are the left/right/up/bottom half parts of a candidate box (Figures 2-3b, 2-3c, 2-3d, and 2-3e). Networks trained on each of them, are guided to learn the appearance characteristics present only on each half part of an object or on each side of the objects borders, aiming also to make the representation more robust with respect to occlusions.



(a) Original box **(b)** Half left **(c)** Half right **(d)** Half up **(e)** Half bottom

(f) Central Region **(g)** Central Region **(h)** Border Region **(i)** Border Region **(j)** Context. Region

Figure 2-3: Illustration of the regions used on the Multi-Region CNN model. With yellow solid lines are the borders of the regions and with green dashed lines are the borders of the candidate detection box. **Region a:** it is the candidate box itself as being used on R-CNN [43]. **Region b, c, d, e:** they are the left/right/up/bottom half parts of the candidate box. **Region f:** it is obtained by scaling the candidate box by a factor of 0.5. **Region g:** the inner box is obtained by scaling the candidate box by a factor of 0.3 and the outer box by a factor of 0.8. **Region h:** we obtain the inner box by scaling the candidate box by a factor of 0.5 and the outer box has the same size as the candidate box. **Region i:** the inner box is obtained by scaling the candidate box by a factor of 0.8 and the outer box by a factor of 1.5. **Region j:** the inner box is the candidate box itself and the outer box is obtained by scaling the candidate box by a factor of 1.8.

Central Regions: there are two type of central regions in our model (Figures 2-3f and 2-3g). The networks trained on them are guided to capture the pure appearance characteristics of the central part of an object that is probably less interfered from other objects next to it or its background.

Border Regions: we include two such regions, with the shape of rectangular rings (Figures 2-3h and 2-3i). We expect that the networks dedicated on them will be guided to focus on the joint appearance characteristics on both sides of the object borders, also aiming to make the representation more sensitive to inaccurate localization.

Contextual Region: there is one region of this type that has rectangular ring shape (Figure 2-3j). Its assigned network is driven to focus on the contextual appearance that surrounds an object such as the appearance of its background or of other objects next to it.

Role on detection. Concerning the general role of the regions on object detection, we briefly focus below on two of the reasons why using these regions helps:

Discriminative feature diversification. Our hypothesis is that having regions that render visible to their network-components only a limited part of the object or only its immediate

surrounding forces each network-component to discriminate image boxes solely based on the visual information that is apparent on them thus diversifying the discriminative factors captured by our overall recognition model. For example, if the border region depicted on Figure 2-3i is replaced with one that includes its whole inner content, then we would expect that the network-component dedicated on it will not pay the desired attention on the visual content that is concentrated around the borders of an object. We tested such a hypothesis by conducting an experiment where we trained and tested two Multi-Region CNN models that consist of two regions each. Model A included the original box region (Figure 2-3a) and the border region of Figure 2-3i that does not contain the central part of the object. On model B, we replaced the latter region (Figure 2-3i), which is a rectangular ring, with a normal box of the same size. Both of them were trained on PASCAL VOC 2007 [33] train+val set and tested on the test set of the same challenge. Model A achieved 64.1% mAP while Model B achieved 62.9% mAP which is 1.2 points lower and validates our assumption.

Localization-aware representation. We argue that our multi-region architecture as well as the type of regions included, address to a certain extent one of the major problems on the detection task, which is the inaccurate object localization. We believe that having multiple regions with network-components dedicated on each of them imposes soft constraints regarding the visual content allowed on each type of region for a given candidate detection box. We experimentally justify this argument in sections 2.7.2 and 2.7.3.

2.4 Semantic Segmentation-Aware CNN Model

To further diversify the features encoded by our representation, we extend the Multi-Region CNN model so that it also learns semantic segmentation-aware CNN features. The motivation for this extension comes from the close connection between segmentation and detection tasks as well as from the fact that segmentation related cues are empirically known to often help object detection [27, 50, 106]. In the context of our multi-region CNN network, the incorporation of the semantic segmentation-aware features is done by adding properly adapted versions of the two main modules of the network, i.e., the 'activation-maps' and 'region-adaptation' modules (see architecture in Figure 2-4). We hereafter refer to the

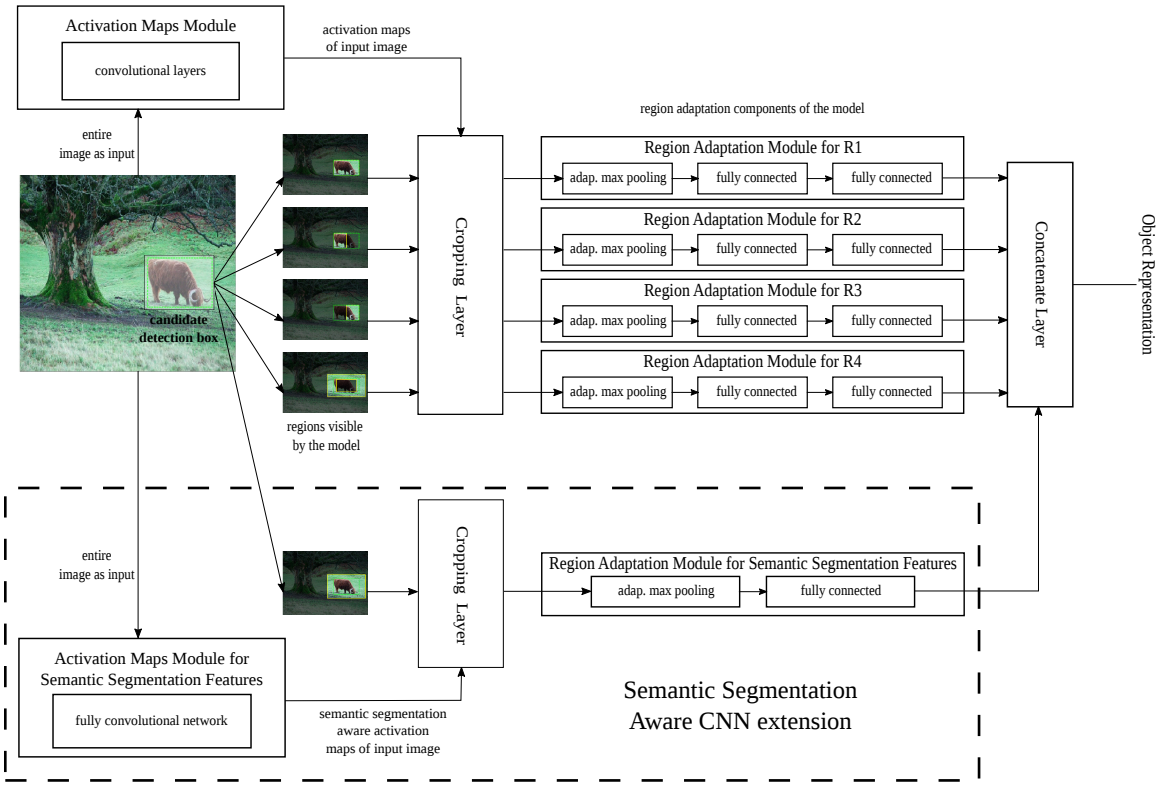


Figure 2-4: Multi Region CNN architecture extended with the semantic segmentation-aware CNN features.

resulting modules as:

- *Activation maps module for semantic segmentation-aware features.*
- *Region adaptation module for semantic segmentation-aware features.*

It is important to note that the modules for the semantic segmentation-aware features are trained *without the use of any additional annotation*. Instead, they are trained in a *weakly supervised manner* using only the provided bounding box annotations for detection.

We combine the Multi-Region CNN features and the semantic segmentation aware CNN features by concatenating them (see Figure 2-4). The resulting network thus jointly learns deep features of both types during training.

2.4.1 Activation maps module for semantic segmentation-aware features

Fully Convolutional Nets. In order to serve the purpose of exploiting semantic segmentation aware features, for this module we adopt a Fully Convolutional Network [92] architecture,

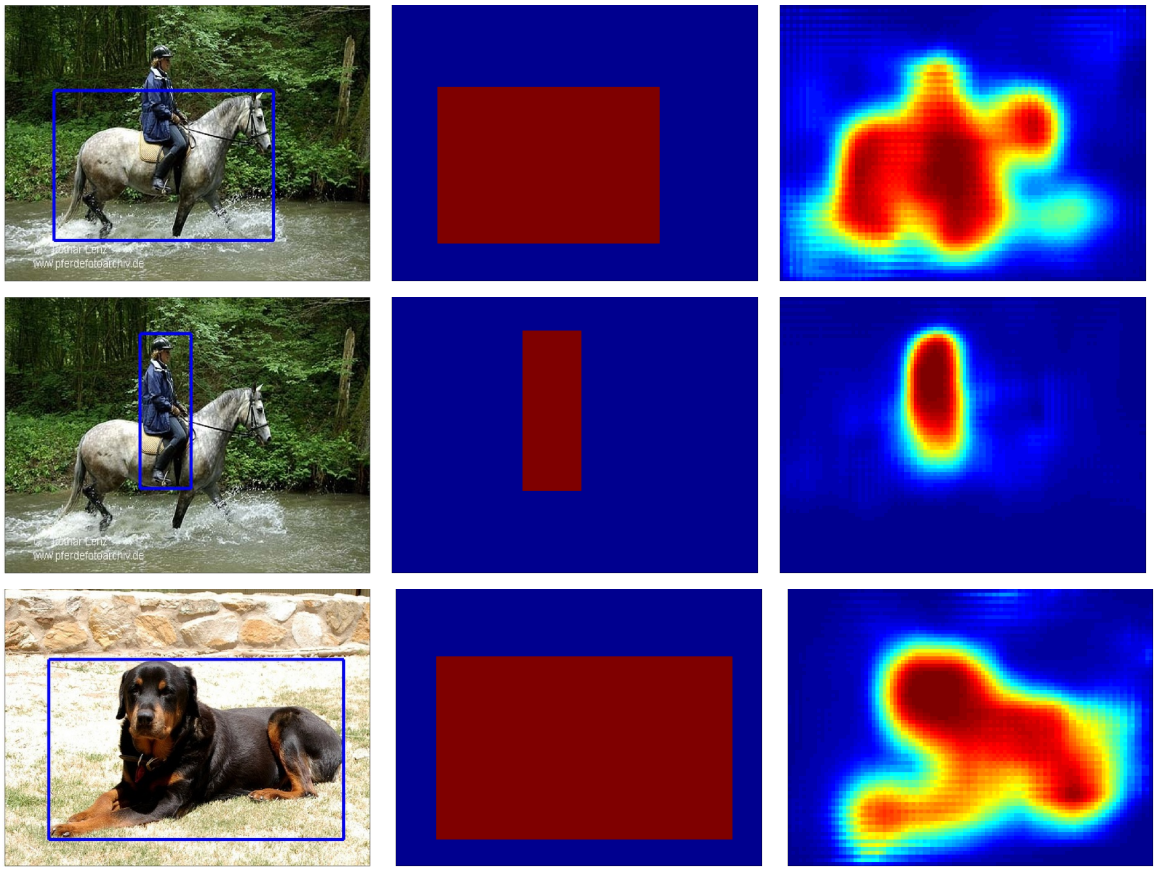


Figure 2-5: Illustration of the weakly supervised training of the FCN [92] used as activation maps module for the semantic segmentation aware CNN features. **Left column:** images with the ground truth bounding boxes drawn on them. The classes depicted from top to down order are horse, human, and dog. **Middle column:** the segmentation target values used during training of the FCN. They are artificially generated from the ground truth bounding box(es) on the left column. We use blue color for the background and red color for the foreground. **Right column:** the foreground masks estimated from our trained FCN model. These clearly verify that, despite the weakly supervised training, our extracted features carry significant semantic segmentation information.

abbreviated hereafter as FCN, trained to predict class specific foreground probabilities (we refer the interested reader to [92] for more details about FCN where it is being used for the task of semantic segmentation).

Weakly Supervised Training. To train the activation maps module for the class-specific foreground segmentation task, we only use the annotations provided on object detection challenges (so as to make the training of our overall system independent of the availability of segmentation annotations). To that end, we follow a weakly supervised training strategy and we create artificial foreground class-specific segmentation masks using bounding box annotations. More specifically, the ground truth bounding boxes of an image are projected

on the spatial domain of the last hidden layer of the FCN, and the ”pixels” that lay inside the projected boxes are labeled as foreground while the rest are labeled as background (see left and middle column in Figure 2-5). The aforementioned process is performed independently for each class and yields as many segmentation target images as the number of our classes. As can be seen in Figure 2-5 right column, despite the weakly supervised way of training, the resulting activations still carry significant semantic segmentation information, enough even to delineate the boundaries of the object and separate the object from its background.

Activation Maps. After the FCN has been trained on the auxiliary task of foreground segmentation, we drop the last classification layer and we use the rest of the FCN network in order to extract from images semantic segmentation aware activation maps.

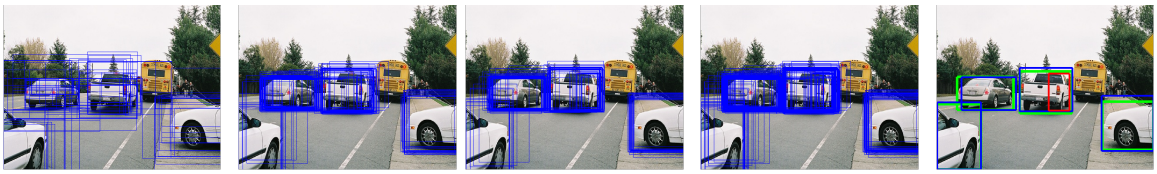
2.4.2 Region adaptation module for semantic segmentation-aware features

We exploit the above activation maps by treating them as mid-level features and adding on top of them a single region adaptation module trained for our primary task of object detection. In this case, we choose to use a single region obtained by enlarging the candidate detection box by a factor of 1.5 (such a region contains semantic information also from the surrounding of a candidate detection box). The reason that we do not repeat the same regions as in the initial Multi-Region CNN architecture is for efficiency as these are already used for capturing the appearance cues of an object.

2.5 Object Localization

As already explained, the proposed Multi-Region CNN recognition model exhibits the localization awareness property that is necessary for accurate object localization. However, by itself it is not enough. In order to make full use of it, our recognition model needs to be presented with well localized candidate boxes that in turn will be scored with high confidence from it. The solution that we adopt consists of 3 main components:

CNN region adaptation module for bounding box regression. We introduce an extra



(a) Step 1 **(b)** Step 2 **(c)** Step 3 **(d)** Step 4 **(e)** Step 5

Figure 2-6: Illustration of the object localization scheme for instances of the class car. **Step 1:** the initial box proposal of the image. For clarity we visualize only the box proposals that are not rejected after the first scoring step. **Step 2:** the new box locations obtained after performing CNN based bounding box regression on the boxes of Step 1. **Step 3:** the boxes obtained after a second step of box scoring and regressing on the boxes of Step 2. **Step 4:** the boxes of Step 2 and Step 3 merged together. **Step 5:** the detected boxes after applying non-maximum-suppression and box voting on the boxes of Step 4. On the final detections we use blue color for the true positives and red color for the false positives. Also, the ground truth bounding boxes are drawn with green color. The false positive that we see after the last step is a duplicate detection that survived from non-maximum-suppression.

region adaptation module that, instead of being used for object recognition, is trained to predict the object bounding box. It is applied on top of the activation maps produced from the Multi-Region CNN model and, instead of a typical one-layer ridge regression model [43], consists of two hidden fully connected layers and one prediction layer that outputs 4 values (i.e., a bounding box) per category. In order to allow it to predict the location of object instances that are not in the close proximity of any of the initial candidate boxes, we use as region a box obtained by enlarging the candidate box by a factor of 1.3. This combination offers a significant boost on the detection performance of our system by allowing it to make more accurate predictions and for more distant objects.

Iterative Localization. Our localization scheme starts from the selective search proposals [153] and works by iteratively scoring them and refining their coordinates. Specifically, let $\mathbf{B}_c^t = \{B_{i,c}^t\}_{i=1}^{N_{c,t}}$ denote the set of $N_{c,t}$ bounding boxes generated on iteration t for class c and image X. For each iteration[1] $t = 1, ..., T$, the boxes from the previous iteration $\mathbf{B}_c^{t-1}$ are scored with $s_{i,c}^t = \mathcal{F}_{rec}(B_{i,c}^{t-1}|c, X)$ by the recognition model $\mathcal{F}_{rec}(.)$ and refined into $B_{i,c}^t = \mathcal{F}_{reg}(B_{i,c}^{t-1}|c, X)$ by the CNN regression model $\mathcal{F}_{reg}(.)$, thus forming the set of candidate detections $\mathbf{D}_c^t = \{(s_{i,c}^t, B_{i,c}^t)\}_{i=1}^{N_{c,t}}$. For the first iteration $t = 1$, the box proposals $\mathbf{B}_c^0$ are coming from selective search [153] and are common between all the classes. Also, those with score $s_{i,c}^0$ below a threshold τ_s are rejected[2] in order to reduce the computational

[1] In practice $T = 2$ iterations were enough for convergence.

[2] We use $\tau_s = -2.1$, which was selected such that the average number of box proposals per image from all

burden of the subsequent iterations. This way, we obtain a sequence of candidate detection sets $\{\mathbf{D}_c^t\}_{t=1}^T$ that all-together both exhibit high recall of the objects on an image and are well localized on them.

Bounding box voting. After the last iteration T, the candidate detections $\{\mathbf{D}_c^t\}_{t=1}^T$ produced on each iteration t are merged together $\mathbf{D}_c = \cup_{t=1}^T \mathbf{D}_c^t$. Because of the multiple regression steps, the generated boxes will be highly concentrated around the actual objects of interest. We exploit this "by-product" of the iterative localization scheme by adding a step of bounding box voting. First, standard non-max suppression [43] is applied on $\mathbf{D}_c$ and produces the detections $\mathbf{Y}_c = \{(s_{i,c}, B_{i,c})\}$ using an IoU overlap threshold of 0.3. Then, the final bounding box coordinates $B_{i,c}$ are further refined by having each neighboring box $B_{j,c} \in \mathcal{N}(B_{i,c})$ to vote for the bounding box location using as weight its score $w_{j,c} = max(0, s_{j,c})$:

$$B'_{i,c} = \frac{\sum_{j:B_{j,c} \in \mathcal{N}(B_{i,c})} w_{j,c} \cdot B_{j,c}}{\sum_{j:B_{j,c} \in \mathcal{N}(B_{i,c})} w_{j,c}}, \tag{2.1}$$

where $\mathcal{N}(B_{i,c})$ is the set of boxes in $\mathbf{D}_c$ that overlap with $B_{i,c}$ by more than 0.5 on IoU metric. The final set of object detections for class c will be $\mathbf{Y}'_c = \{(s_{i,c}, B'_{i,c})\}$.

In Figure 2-6 we provide a visual illustration of the object localization.

2.6 Implementation Details

For all the CNN models involved in our proposed system, we used the publicly available 16-layers VGG model [144] pre-trained on ImageNet [22] for the task of image classification[3]. For simplicity, we fine-tuned only the fully connected layers (fc6 and fc7) of each model while we preserved the pre-trained weights for the convolutional layers (conv1_1 to conv5_3), which are shared among all the models of our system.

Multi-Region CNN model. Its activation maps module consists of the convolutional part (layers conv1_1 to conv5_3) of the 16-layers VGG-Net that outputs 512 feature channels. The max-pooling layer right after the last convolutional layer is omitted on this module.

the classes together is around 250.

[3]https://gist.github.com/ksimonyan/

Each region adaptation module inherits the fully connected layers of the 16-layers VGG-Net and is fine-tuned separately from the others. Regarding the regions that are rectangular rings, both the inner and outer box are projected on the activation maps and then the activations that lay inside the inner box are masked out by setting them to zero (similar to the Convolutional Feature Masking layer proposed on [18]). In order to train the region adaptation modules, we follow the guidelines of R-CNN [43]. As an optimization objective we use the softmax-loss and the minimization is performed with stochastic gradient descent (SGD). The momentum is set to 0.9, the learning rate is initially set to 0.001 and then reduced by a factor of 10 every $30k$ iterations, and the minibatch has 128 samples. The positive samples are defined as the selective search proposals [153] that overlap a ground-truth bounding box by at least 0.5. As negative samples we use the proposals that overlap with a ground-truth bounding box on the range $[0.1, 0.5)$. The labelling of the training samples is relative to the original candidate boxes and is the same across all the different regions.

Activation maps module for semantic segmentation aware features. Its architecture consists of the 16-layers VGG-Net without the last classification layer and transformed to a FCN [92] (by reshaping the fc6 and fc7 fully connected layers to convolutional ones with kernel size of 7×7 and 1×1 correspondingly). For efficiency purposes, we reduce the output channels of the fc7 layer from 4096 to 512. In order to learn the semantic segmentation aware features, we use an auxiliary fc8 convolutional classification layer (of kernel size 1×1) that outputs as many channels as our classes and a binary (foreground vs background) logistic regression loss applied on each spatial cell and for each class independently. Initially, we train the FCN with the 4096 channels on the fc7 layer until convergence. Then, we replace the fc7 layer with another one that has 512 output channels, which is initialized from a Gaussian distribution, and the training of the FCN starts from the beginning and is continued until convergence again. For loss minimization we use SGD with minibatch of size 10. The momentum is set to 0.9 and the learning rate is initialized to 0.01 and decreased by a factor of 10 every 20 epochs. For faster convergence, the learning rate of the randomly initialized fc7 layer with the 512 channels is multiplied by a factor of 10.

Region adaptation module for semantic segmentation aware features. Its architecture consists of a spatially adaptive max-pooling layer [92] that outputs feature maps of 512

channels on a 9×9 grid, and a fully connected layer with 2096 channels. In order to train it, we use the same procedure as for the region components of the Multi-Region CNN model. During training, we only learn the weights of the region adaptation module layers that are randomly initialized from a Gaussian distribution.

Classification SVMs. In order to train the SVMs we follow the same principles as in [43]. As positive samples are considered the ground truth bounding boxes and as negative samples are considered the selective search proposals [153] that overlap with the ground truth boxes by less than 0.3. We use hard negative mining the same way as in [43, 35].

CNN region adaptation module for bounding box regression. The activation maps module used as input in this case is common with the Multi-Region CNN model. The region adaptation module for bounding box regression inherits the fully connected hidden layers of the 16-layers VGG-Net. As a loss function we use the Euclidean distance between the target values and the network predictions. For training samples we use the box proposals [153] that overlap by at least 0.4 with the ground truth bounding boxes. The target values are defined the same way as in R-CNN [43]. The learning rate is initially set to 0.01 and reduced by a factor of 10 every $40k$ iterations. The momentum is set to 0.9 and the minibatch size is 128.

Multi-Scale Implementation. In our system we adopt a similar multi-scale implementation as in SPP-Net [55]. More specifically, we apply the activation maps modules of our models on multiple scales of an image and then a single scale is selected for each region adaptation module independently.

- *Multi-Region CNN model:* The activation maps module is applied on 7 scales of an image with their shorter dimension being in $\{480, 576, 688, 874, 1200, 1600, 2100\}$. For training, the region adaptation modules are applied on a random scale and for testing, a single scale is used such that the area of the scaled region is closest to 224×224 pixels. In the case of rectangular ring regions, the scale is selected based on the area of the scaled outer box of the rectangular ring.
- *Semantic Segmentation-Aware CNN model:* The activation maps module is applied on 3 scales of an image with their shorter dimension being in $\{576, 874, 1200\}$. For training, the region adaptation module is applied on a random scale and for testing, a single scale is selected such that the area of the scaled region is closest to 288×288

pixels.

- *Bounding Box Regression CNN model:* The activation maps module is applied on 7 scales of an image with their shorter dimension being in $\{480, 576, 688, 874, 1200, 1600, 2100\}$. Both during training and testing, a single scale is used such that the area of the scaled region is closest to 224×224 pixels.

Training/Test Time. On a Titan GPU and on PASCAL VOC 2007 train+val dataset, the training time of each region adaptation module is approximately 12 hours, of the activation maps module for the semantic segmentation features is approximately 4 days, and of the linear SVM is approximately 16 hours. In order to speed up the above steps, the activation maps (conv5_3 features and the fc7 semantic segmentation aware features) were pre-cashed on a SSD. Finally, the per image runtime is around 30 seconds.

2.7 Experimental Evaluation

We evaluate our detection system on PASCAL VOC 2007 [33] and on PASCAL VOC2012 [32]. During the presentation of the results, we will use as baseline either the *Original candidate box* region alone (Figure 2-3a) and/or the R-CNN framework with VGG-Net [144]. We note that, when the *Original candidate box* region alone is used then the resulted model is a realization of the SPP-Net [55] object detection framework with the 16-layers VGG-Net [144]. Except if otherwise stated, for all the PASCAL VOC 2007 results, we trained our models on the train+val set and tested them on the test set of the same year.

2.7.1 Results on PASCAL VOC 2007

First, we asses the significance of each of the region adaptation modules alone on the object detection task. Results are reported in Table 2.1. As we expected, the best performing component is the *Original candidate box*. What is surprising is the high detection performance of individual regions like the *Border Region* in Figure 2-3i 54.8% or the *Contextual Region* in Figure 2-3j 47.2%. Despite the fact that the area visible by them includes limited or not at all portion of the object, they outperform previous detection systems that were based on hand crafted features. Also interesting, is the high detection performance of the semantic

Adaptation Modules	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Original Box fig. 2-3a	**0.729**	**0.715**	**0.593**	**0.478**	**0.405**	**0.713**	**0.725**	**0.741**	**0.418**	**0.694**	0.591	**0.713**	0.662	**0.725**	**0.560**	**0.312**	**0.601**	0.565	0.669	**0.731**	**0.617**
Left Half Box fig. 2-3b	0.635	0.659	0.455	0.364	0.322	0.621	0.640	0.589	0.314	0.620	0.463	0.573	0.545	0.641	0.477	0.300	0.532	0.442	0.546	0.621	0.518
Right Half Box fig. 2-3c	0.626	0.605	0.470	0.331	0.314	0.607	0.616	0.641	0.278	0.487	0.513	0.548	0.564	0.585	0.459	0.262	0.469	0.465	0.573	0.620	0.502
Up Half Box fig. 2-3d	0.591	0.651	0.470	0.266	0.361	0.629	0.656	0.641	0.305	0.604	0.511	0.604	0.643	0.588	0.466	0.220	0.545	0.528	0.590	0.570	0.522
Bottom Half Box fig. 2-3e	0.607	0.631	0.406	0.397	0.233	0.594	0.626	0.559	0.285	0.417	0.404	0.520	0.490	0.649	0.387	0.233	0.457	0.344	0.566	0.617	0.471
Central Region fig. 2-3f	0.552	0.622	0.413	0.244	0.283	0.502	0.594	0.603	0.282	0.523	0.424	0.516	0.495	0.584	0.386	0.232	0.527	0.358	0.533	0.587	0.463
Central Region fig. 2-3g	0.674	0.705	0.547	0.367	0.337	0.678	0.698	0.687	0.381	0.630	0.538	0.659	0.667	0.679	0.507	0.309	0.557	0.530	0.611	0.694	0.573
Border Region fig. 2-3h	0.694	0.696	0.552	0.470	0.389	0.687	0.706	0.703	0.398	0.631	0.515	0.660	0.643	0.686	0.539	0.307	0.582	0.537	0.618	0.717	0.586
Border Region fig. 2-3i	0.651	0.649	0.504	0.407	0.333	0.670	0.704	0.624	0.323	0.625	0.533	0.594	0.656	0.627	0.517	0.223	0.533	0.515	0.604	0.663	0.548
Contextual Region fig. 2-3j	0.624	0.568	0.425	0.380	0.255	0.609	0.650	0.545	0.222	0.509	0.522	0.427	0.563	0.541	0.431	0.163	0.482	0.392	0.597	0.532	0.472
Semantic-aware region.	0.652	0.684	0.549	0.407	0.225	0.658	0.676	0.738	0.316	0.596	**0.635**	0.705	**0.670**	0.689	0.545	0.230	0.522	**0.598**	**0.680**	0.548	0.566

Table 2.1: Detection performance of individual regions on VOC 2007 test set. They were trained on VOC 2007 train+val set.

Approach	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN with VGG-Net	0.716	0.735	0.581	0.422	0.394	0.707	0.760	0.745	0.387	0.710	0.569	0.745	0.679	0.696	0.593	0.357	0.621	0.640	0.665	0.712	0.622
R-CNN with VGG-Net & bbox reg.	0.734	0.770	0.634	0.454	0.446	0.751	0.781	0.798	0.405	0.737	0.622	0.794	0.781	0.731	0.642	0.356	0.668	0.672	0.704	0.711	0.660
Best approach of [172]	0.725	0.788	0.67	0.452	0.510	0.738	0.787	0.783	0.467	0.738	0.615	0.771	0.764	0.739	0.665	0.392	0.697	0.594	0.668	0.729	0.665
Best approach of [172] & bbox reg.	0.741	**0.832**	0.670	0.508	0.516	0.762	0.814	0.772	0.481	0.789	0.656	0.773	0.784	0.751	0.701	0.414	0.696	0.608	0.702	0.737	0.685
Original Box fig. 2-3a	0.729	0.715	0.593	0.478	0.405	0.713	0.725	0.741	0.418	0.694	0.591	0.713	0.662	0.725	0.560	0.312	0.601	0.565	0.669	0.731	0.617
MR-CNN	0.749	0.757	0.645	0.549	0.447	0.741	0.755	0.760	0.481	0.724	0.674	0.765	0.724	0.749	0.617	0.348	0.617	0.640	0.735	0.760	0.662
MR-CNN & S-CNN	0.768	0.757	0.676	0.551	0.456	0.776	0.765	0.784	0.467	0.747	0.688	0.793	0.742	0.770	0.625	0.374	0.643	0.638	0.740	0.747	0.675
MR-CNN & S-CNN & Loc.	**0.787**	0.818	**0.767**	**0.666**	**0.618**	**0.817**	**0.853**	**0.827**	**0.570**	**0.819**	**0.732**	**0.846**	**0.860**	**0.805**	**0.749**	**0.449**	**0.717**	**0.697**	**0.787**	**0.799**	**0.749**

Table 2.2: Detection performance of our modules on VOC 2007 test set. Each model was trained on VOC 2007 train+val set.

Approach	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN with VGG-Net from [172]	0.402	0.433	0.234	0.144	0.133	0.482	0.445	0.364	0.171	0.340	0.279	0.363	0.268	0.282	0.212	0.103	0.337	0.366	0.316	0.489	0.308
Best approach of [172]	0.463	0.581	0.311	0.216	0.258	0.571	0.582	0.435	0.230	0.464	0.290	0.407	0.406	0.463	0.334	0.106	0.413	0.409	0.458	0.563	0.398
Best approach of [172] & bbox reg.	0.471	**0.618**	0.352	0.181	0.297	**0.660**	0.647	0.480	**0.253**	0.504	0.349	0.437	0.508	0.494	0.368	0.137	0.447	0.436	0.498	**0.605**	0.437
Original Candidate Box	0.449	0.426	0.237	0.175	0.157	0.441	0.444	0.377	0.182	0.295	0.303	0.312	0.249	0.332	0.187	0.099	0.302	0.286	0.337	0.499	0.305
MR-CNN	0.495	0.505	0.292	0.235	0.179	0.513	0.504	0.481	0.206	0.381	0.375	0.387	0.296	0.403	0.239	0.151	0.341	0.389	0.422	0.521	0.366
MR-CNN & S-CNN	0.507	0.523	0.316	0.266	0.177	0.547	0.513	0.492	0.210	0.450	0.361	0.433	0.309	0.408	0.246	0.151	0.359	0.427	0.438	0.534	0.383
MR-CNN & S-CNN & Loc.	**0.549**	0.613	**0.430**	**0.315**	**0.383**	0.646	**0.650**	**0.512**	**0.253**	**0.544**	**0.505**	**0.521**	**0.591**	**0.540**	**0.393**	**0.159**	**0.485**	**0.468**	**0.553**	0.573	**0.484**

Table 2.3: Detection performance of our modules on VOC 2007 test set. In this table, the IoU overlap threshold for positive detections is 0.7. Each model was trained on VOC 2007 train+val set.

segmentation aware region, 56.6%.

In Table 2.2, we report the detection performance of our proposed modules. The Multi-Region CNN model without the semantic segmentation aware CNN features (*MR-CNN*), achieves 66.2% mAP, which is 4.2 points higher than *R-CNN with VGG-Net* (62.0%) and 4.5 points higher than the *Original candidate box* region alone (61.7%). Moreover, its detection performance slightly exceeds that of *R-CNN with VGG-Net* and bounding box regression (66.0%). Extending the Multi-Region CNN model with the semantic segmentation aware CNN features (*MR-CNN & S-CNN*), boosts the performance of our recognition model another 1.3 points and reaches the total of 67.5% mAP. Comparing to the recently published method of Yuting et al. [172], our *MR-CNN & S-CNN* model scores 1 point higher than their best performing method that includes generation of extra box proposals via Bayesian optimization and structured loss during the fine-tuning of the VGG-Net. Significant is also the improvement that we get when we couple our recognition model with the CNN model

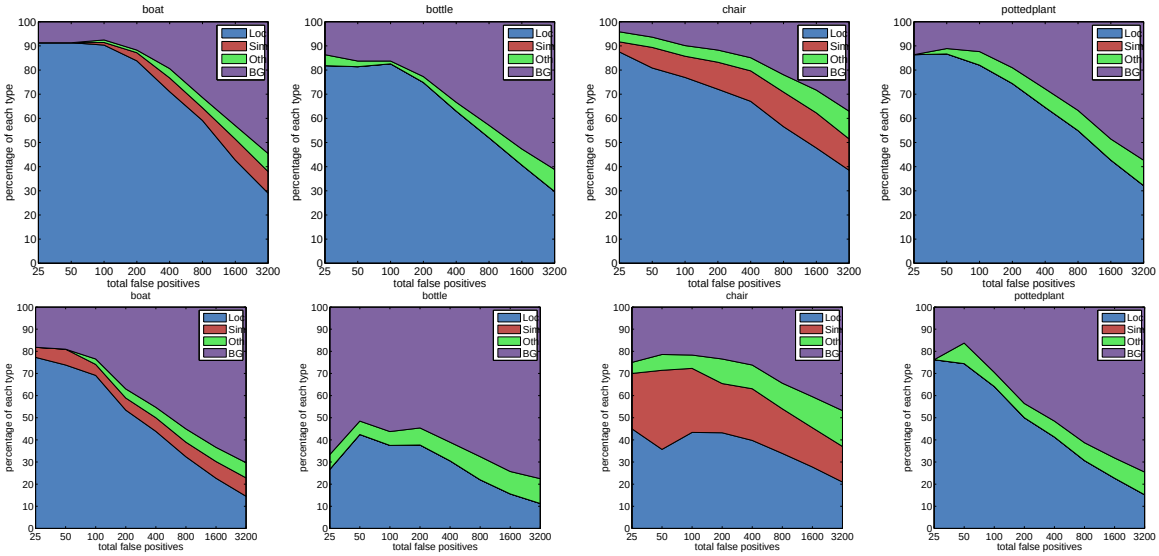


Figure 2-7: Top ranked false positive types. **Top row:** our baseline which is the *original candidate box* only model. **Bottom row:** our overall system. We present only the graphs for the classes boat, bottle, chair, and pottedplant (which are some of the most difficult classes of PASCAL VOC challenge) for space efficiency reasons.

Approach	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Original candidate box-Baseline	0.7543	0.7325	0.6634	0.5816	0.5775	0.7109	0.7390	0.7277	0.5718	0.7112	0.6007	0.7000	0.7039	0.7194	0.6607	0.5339	0.6855	0.6461	0.6903	0.7359
MR-CNN	0.7938	0.7864	0.7180	0.6424	0.6222	0.7609	0.7918	0.7758	0.6186	0.7483	0.6802	0.7448	0.7562	0.7569	0.7166	0.5753	0.7268	0.7148	0.7391	0.7556

Table 2.4: Correlation between the IoU overlap of selective search box proposals [153] (with the closest ground truth bounding box) and the scores assigned to them.

Approach	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Original candidate box-Baseline	0.9327	0.9324	0.9089	0.8594	0.8570	0.9389	0.9455	0.9250	0.8603	0.9237	0.8806	0.9209	0.9263	0.9317	0.9151	0.8415	0.8932	0.9060	0.9241	0.9125
MR-CNN	0.9462	0.9479	0.9282	0.8843	0.8740	0.9498	0.9593	0.9355	0.8790	0.9338	0.9127	0.9358	0.9393	0.9440	0.9341	0.8607	0.9120	0.9314	0.9413	0.9210

Table 2.5: The Area-Under-Curve (AUC) measure for the well-localized box proposals against the mis-localized box proposals.

for bounding box regression under the iterative localization scheme proposed (*MR-CNN & S-CNN & Loc.*). Specifically, the detection performance is raised from 67.5% to 74.9%.

In Table 2.3, we report the detection performance of our system when the overlap threshold for considering a detection positive is set to 0.7. This metric was proposed from [172] in order to reveal the localization capability of their method. From the table we observe that each of our modules exhibits very good localization capability, which was our goal when designing them, and our overall system exceeds in that metric the approach of [172].

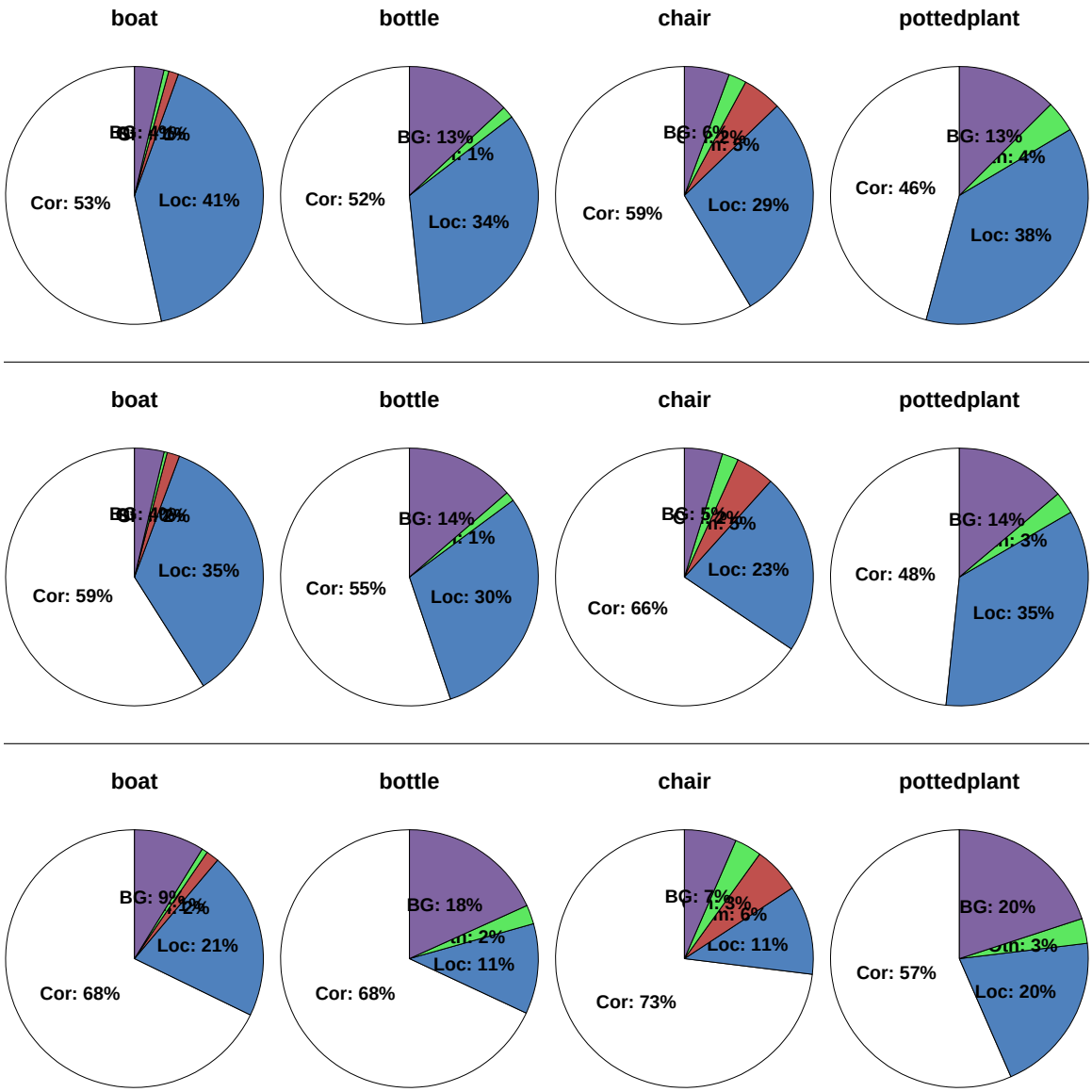


Figure 2-8: Fraction of top N detections (N=num of objs in category) that are correct (Cor; in white color), or false positives due to poor localization (Loc; in blue color), confusion with similar objects (Sim; in red color), confusion with other VOC objects (Oth; in green color), or confusion with background or unlabeled objects (BG; in purple color). **Top row:** our baseline which is the *original candidate box* only model. **Middle row:** Multi-Region CNN model without the semantic segmentation aware CNN features. **Bottom row:** our overall system. We present only the pie charts for the classes boat, bottle, chair, and pottedplant (which are some of the most difficult classes of PASCAL VOC challenge) for space efficiency reasons.

2.7.2 Detection error analysis

We use the tool of Hoiem et al. [61] to analyze the detection errors of our system. In Figure 2-8, we plot pie charts with the percentage of detections that are false positive due to bad localization, confusion with similar category, confusion with other category, and triggered

on the background or an unlabeled object. We observe that, by using the Multi-Region CNN model instead of the *Original Candidate Box* region alone, a considerable reduction in the percentage of false positives due to bad localization is achieved. This validates our argument that focusing on multiple regions of an object increases the localization sensitivity of our model. Furthermore, when our recognition model is integrated on the localization module developed for it, the reduction of false positives due to bad localization is huge. A similar observation can be deducted from Figure 2-7 where we plot the top-ranked false positive types of the baseline and of our overall proposed system.

2.7.3 Localization awareness of Multi-Region CNN model

Two extra experiments are presented here that indicate the localization awareness of our Multi-Region CNN model without the semantic segmentation aware CNN features (*MR-CNN*) against the model that uses only the original candidate box (*Baseline*).

Correlation between the scores and the IoU overlap of box proposals. In this experiment, we estimate the correlation between the IoU overlap of box proposals [153] (with the closest ground truth bounding box) and the score assigned to them from the two examined models. High correlation coefficient means that better localized box proposals will tend to be scored higher than mis-localized ones. We report the correlation coefficients of the aforementioned quantities both for the *Baseline* and *MR-CNN* models in Table 2.4. Because with this experiment we want to emphasize on the localization aspect of the Multi-Region CNN model, we use proposals that overlap with the ground truth bounding boxes by at least 0.1 IoU.

Area-Under-the-Curve of well-localized proposals against mis-localized proposals. The ROC curves are typically used to illustrate the capability of a classifier to distinguish between two classes. This discrimination capability can be measured by computing the Area-Under-the-Curve (AUC) metric. The higher the AUC measure is, the more discriminative is the classifier between the two classes. In our case, the set of well-localized box proposals is the positive class and the set of miss-localized box proposals is the negative class. As well-localized are considered the box proposals that overlap with a ground-truth

Approach	trained on	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN [43] with VGG-Net & bbox reg.	VOC12	0.792	0.723	0.629	0.437	0.451	0.677	0.667	0.830	0.393	0.662	0.517	0.822	0.732	0.765	0.642	0.337	0.667	0.561	0.683	0.610	0.630
Network In Network [90]	VOC12	0.802	0.738	0.619	0.437	0.430	0.703	0.676	0.807	0.419	0.697	0.517	0.782	0.752	0.769	0.651	0.386	0.683	0.580	0.687	0.633	0.638
Best approach of [172] & bbox reg.	VOC12	0.829	0.761	0.641	0.446	0.494	0.703	0.712	**0.846**	0.427	0.686	0.558	0.827	0.771	0.799	0.687	0.414	**0.690**	0.600	0.720	0.662	0.664
MR-CNN & S-CNN & Loc. (Ours)	VOC07	0.829	0.789	0.708	0.528	0.555	0.737	0.738	0.843	0.480	0.702	0.571	0.845	0.769	**0.819**	0.755	**0.426**	0.685	0.599	0.728	0.717	0.691
MR-CNN & S-CNN & Loc. (Ours)	VOC12	**0.850**	**0.796**	**0.715**	**0.553**	**0.577**	**0.760**	**0.739**	**0.846**	**0.505**	**0.743**	**0.617**	**0.855**	**0.799**	0.817	**0.764**	0.410	**0.690**	**0.612**	**0.777**	**0.721**	**0.707**

Table 2.6: Comparative results on VOC 2012 test set.

Approach	trained on	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
MR-CNN & S-CNN & Loc. (Ours)	VOC07+12	**0.803**	**0.841**	**0.785**	**0.708**	**0.685**	**0.880**	**0.859**	**0.878**	**0.603**	**0.852**	**0.737**	**0.872**	**0.865**	**0.850**	0.764	**0.485**	**0.763**	0.755	**0.850**	**0.810**	**0.782**
MR-CNN & S-CNN & Loc. (Ours)	VOC07	0.787	0.818	0.767	0.666	0.618	0.817	0.853	0.827	0.570	0.819	0.732	0.846	0.860	0.805	0.749	0.449	0.717	0.697	0.787	0.799	0.749
Faster R-CNN [127]	VOC07+12	0.765	0.790	0.709	0.655	0.521	0.831	0.847	0.864	0.520	0.819	0.657	0.848	0.846	0.775	**0.767**	0.388	0.736	0.739	0.830	0.726	0.732
NoC [128]	VOC07+12	0.763	0.814	0.744	0.617	0.608	0.847	0.782	0.829	0.530	0.792	0.692	0.832	0.832	0.785	0.680	0.450	0.716	**0.767**	0.822	0.757	0.733
Fast R-CNN [42]	VOC07+12	0.770	0.781	0.693	0.594	0.383	0.816	0.786	0.867	0.428	0.788	0.689	0.847	0.820	0.766	0.699	0.318	0.701	0.748	0.804	0.704	0.700

Table 2.7: Comparative results on VOC 2007 test set for models trained with extra data.

Approach	trained on	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
MR-CNN & S-CNN & Loc. (Ours)	VOC07+12	**0.855**	**0.829**	**0.766**	**0.578**	**0.627**	**0.794**	**0.772**	0.866	**0.550**	**0.791**	**0.622**	0.870	**0.834**	**0.847**	0.789	0.453	**0.734**	0.658	0.803	**0.740**	**0.739**
MR-CNN & S-CNN & Loc. (Ours)	VOC12	0.850	0.796	0.715	0.553	0.577	0.760	0.739	0.846	0.505	0.743	0.617	0.855	0.799	0.817	0.764	0.410	0.690	0.612	0.777	0.721	0.707
Faster R-CNN [127]	VOC07+12	0.849	0.798	0.743	0.539	0.498	0.775	0.759	0.885	0.456	0.771	0.553	0.869	0.817	0.809	**0.796**	0.401	0.726	0.609	0.812	0.615	0.704
Fast R-CNN & YOLO [126]	VOC07+12	0.830	0.785	0.737	0.558	0.431	0.783	0.730	0.892	0.491	0.743	0.566	0.872	0.805	0.805	0.747	0.421	0.708	0.683	**0.815**	0.670	0.704
Deep Ensemble COCO [48]	VOC07+12, COCO [91]	0.840	0.794	0.716	0.519	0.511	0.741	0.721	0.886	0.483	0.734	0.578	0.861	0.800	0.807	0.704	**0.466**	0.696	**0.688**	0.759	0.714	0.701
NoC [128]	VOC07+12	0.828	0.790	0.716	0.523	0.537	0.741	0.690	0.849	0.469	0.743	0.531	0.850	0.813	0.795	0.722	0.389	0.724	0.595	0.767	0.681	0.688
Fast R-CNN [42]	VOC07+12	0.823	0.784	0.708	0.523	0.387	0.778	0.716	**0.893**	0.442	0.730	0.550	**0.875**	0.805	0.808	0.720	0.351	0.683	0.657	0.804	0.642	0.684

Table 2.8: Comparative results on VOC 2012 test set for models trained with extra data.

boxes in the range $[0.5, 1.0]$ and as mis-localized are considered the box proposals that overlap with a ground truth bounding box in the range $[0.1, 0.5)$. In Table 2.5, we report the AUC measure for each class separately and both for the *MR-CNN* and the *Baseline* models.

2.7.4 Results on PASCAL VOC2012

In Table 2.6, we compare our detection system against other published work on the test set of PASCAL VOC 2012 [32]. Our overall system involves the Multi-Region CNN model enriched with the semantic segmentation aware CNN features and coupled with the CNN based bounding box regression under the iterative localization scheme. We tested two instances of our system. Both of them have exactly the same components but they have being trained on different datasets. For the first one, the fine-tuning of the networks as well as the training of the detection SVMs was performed on VOC 2007 train+val dataset that includes 5011 annotated images. For the second one, the fine-tuning of the networks was performed on VOC 2012 train dataset that includes 5717 annotated images and the training of the detection SVMs was performed on VOC 2012 train+val dataset that includes 11540 annotated images. As we observe from Table 2.6, we achieve excellent mAP (69.1% and 70.7% correspondingly) in both cases setting the new state-of-the-art on this test set and for those training sets.

2.7.5 Training with extra data and comparison with contemporaneous work

Approaches contemporary to ours [128, 42, 127, 126], train their models with extra data in order to improve the accuracy of their systems. We follow the same practice and we report results on Tables 2.7 and 2.8. Specifically, we trained our models on VOC 2007 and 2012 train+val datasets using both selective search [153] and EdgeBox [177] proposals. During test time we only use EdgeBox proposals that are faster to be computed. From the tables, it is apparent that our method outperforms the other approaches even when trained with less data. Finally, at the time of completing this work, our entries were ranked ranked 1st and 2nd on the leader board of PASCAL VOC 2012 object detection comp4 benchmark (see Table 2.8) and the difference of our top performing entry from the 3rd was 3.5 points.

2.8 Qualitative Results

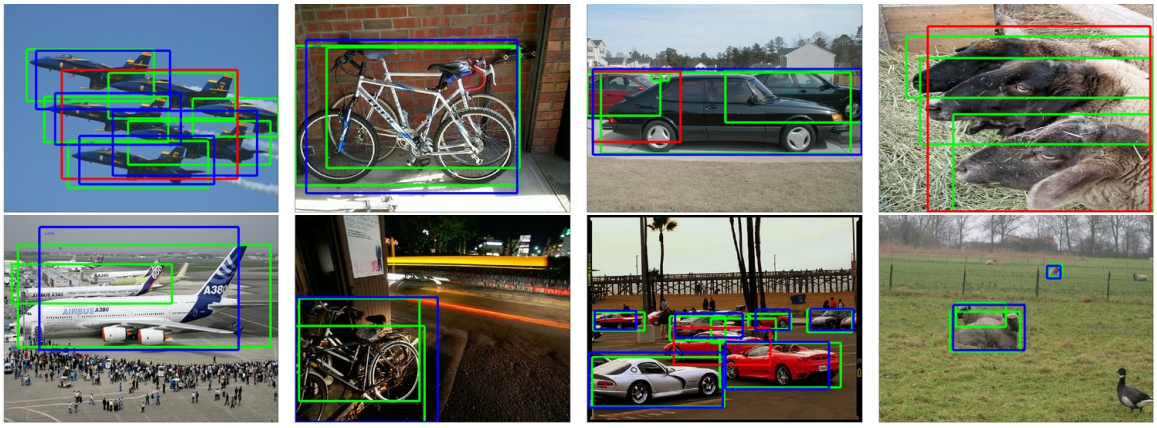
In Figures 2-12, 2-13, and 2-14 we present some object detection examples obtained by our approach. We use blue bounding boxes to mark the true positive detections and red bounding boxes to mark the false positive detections. The ground truth bounding boxes are marked with green color.

Failure cases. Accurately detecting multiple adjacent object instances remains in many cases a difficult problem even for our approach. In Figure 2-9 we present a few difficult examples of this type. In Figure 2-10 we show some other failure cases.

Missing annotations. There were also cases of object instances that were correctly detected by our approach but which were not in the ground truth annotation of PASCAL VOC 2007. Figure 2-11 presents a few such examples of non-annotated object instances.

2.9 Conclusions

In this chapter, we proposed a powerful CNN-based representation for object detection that relies on two key factors: (i) diversification of the discriminative appearance factors



(a) Aeroplane **(b)** Bicycle **(c)** Car **(d)** Sheep

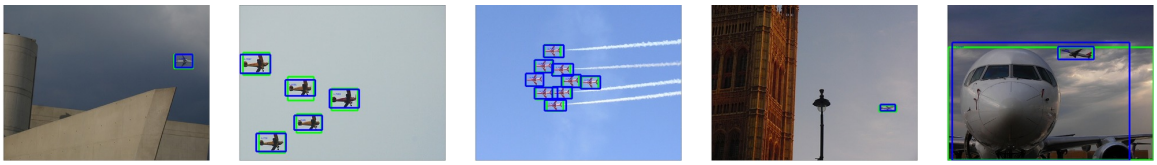
Figure 2-9: Examples of multiple adjacent object instances where our approach fails to detect all of them. We use blue bounding boxes to mark the true positive detections and red bounding boxes to mark the false positive detections. The ground truth bounding boxes are drawn with green color.

Figure 2-10: Examples of false positive detections for the class boat due to the fact that the detected bounding boxes do not include inside their borders the mast of the boat (it is worth noting that on same cases also the annotation provided from PASCAL neglects to include them on its ground truth bounding boxes). The false positive bounding boxes are drawn with red color and the ground truth bounding boxes are drawn with green color.

(a) Bottle **(b)** Chair **(c)** Pottedplant **(d)** TV monitor

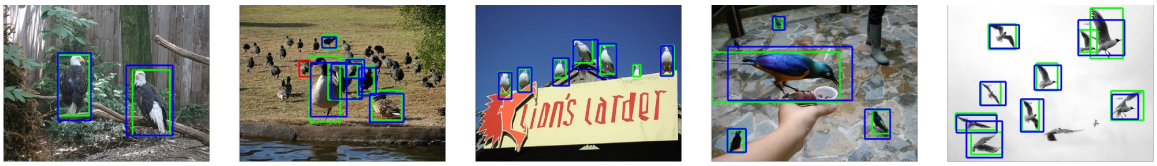
Figure 2-11 – Missing Annotations: Examples where our proposed detection system have truly detected an object instance, but because of missed annotations it is considered false positive. For those detections we used red bounding boxes. For any true positive detection on those images we use blue bounding boxes and the corresponding ground truth bounding boxes are drawn with green color.

captured by it through steering its focus on different regions of the object, and (ii) the encoding of semantic segmentation-aware features. By using it in the context of a CNN-based localization refinement scheme, we show that it achieves excellent results that surpass the state-of-the art by a significant margin.

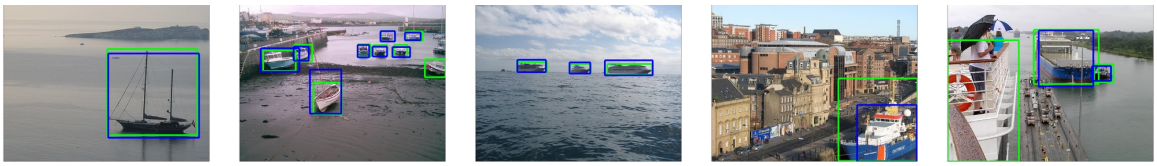


(a) Aeroplane detections.

(b) Bicycle detections.



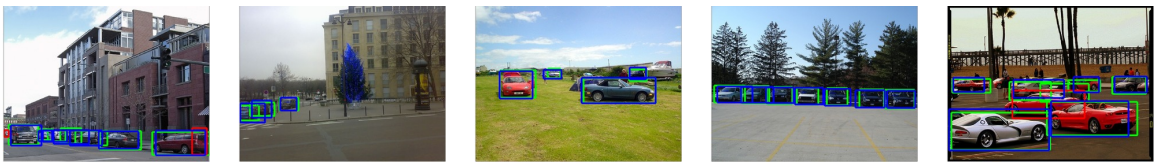
(c) Bird detections.



(d) Boat detections.

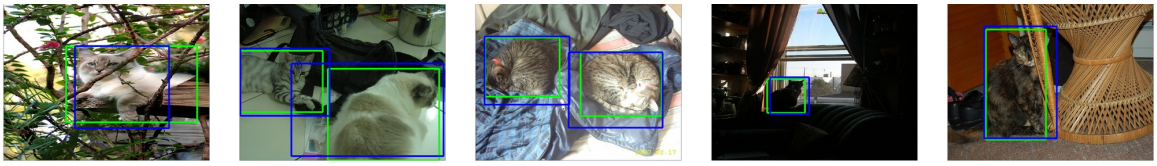
(e) Bottle detections.

(f) Bus detections.



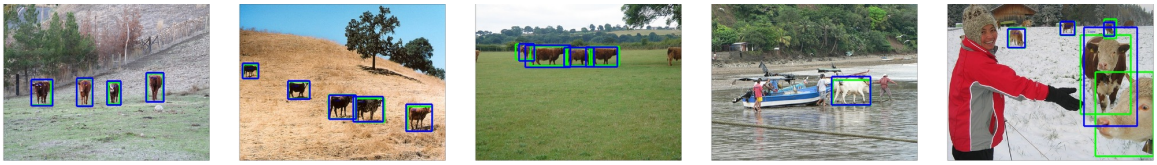
(g) Car detections.

Figure 2-12: We use blue bounding boxes for the true positive detections and red bounding boxes (if any) for the false positive detections. The ground truth bounding boxes are drawn with green color.

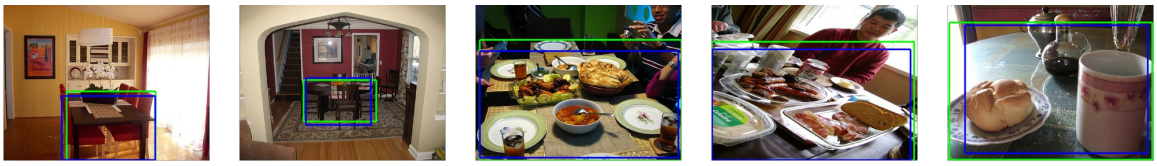


(a) Cat detections.

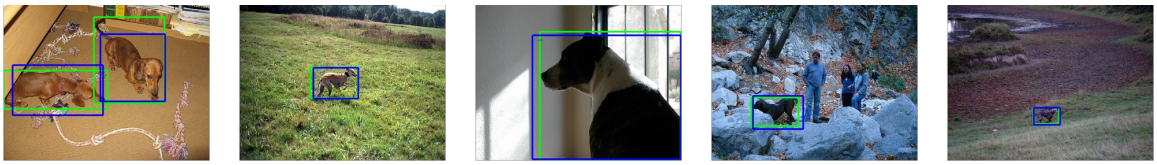
(b) Chair detections.



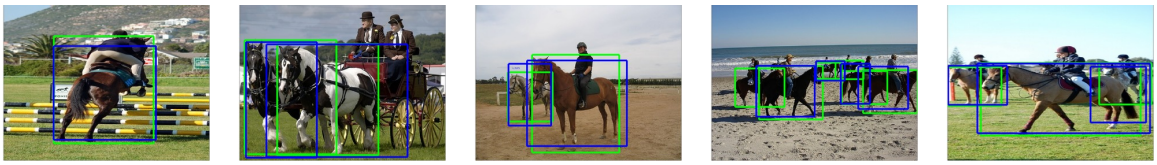
(c) Cow detections.



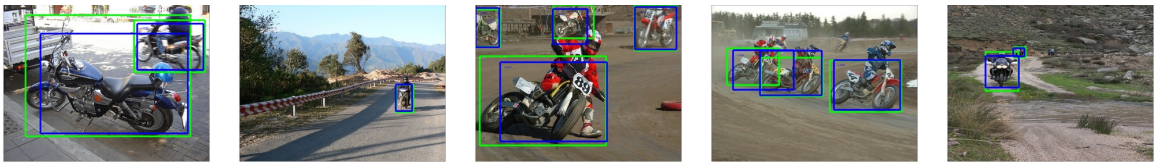
(d) Dinningtable detections.



(e) Dog detections.



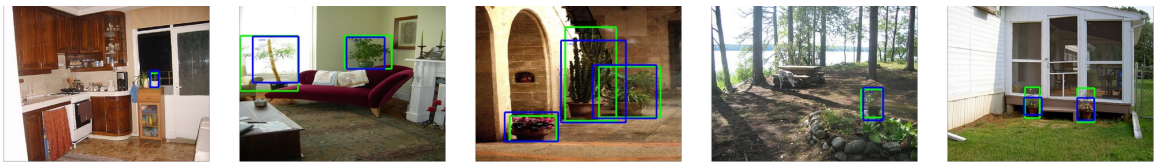
(f) Horse detections.



(g) Motorbike detections.

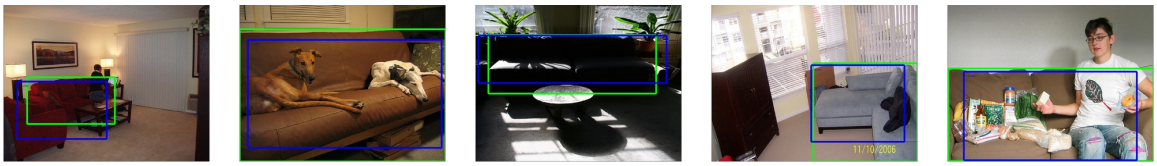
Figure 2-13: We use blue bounding boxes for the true positive detections and red bounding boxes (if any) for the false positive detections. The ground truth bounding boxes are drawn with green color.

(a) Person detections.

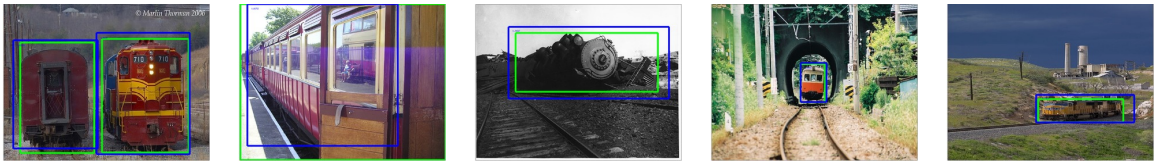


(b) Pottedplant detections.

(c) Sheep detection.



(d) Sofa detections.



(e) Train detections.

(f) TV monitor detections.

Figure 2-14: We use blue bounding boxes for the true positive detections and red bounding boxes (if any) for the false positive detections. The ground truth bounding boxes are drawn with green color.

Chapter 3

Improving Localization Accuracy for Object Detection

3.1 Introduction

In this chapter we focus on the localization aspect of the object detection problem. The localization accuracy by which a detection system is able to predict the bounding boxes of the objects of interest is typically judged based on the Intersection over Union (IoU) between the predicted and the ground truth bounding box. Although in challenges such as PASCAL VOC an IoU detection threshold of 0.5 is used for deciding whether an object has been successfully detected, in real life applications a higher localization accuracy (e.g., IoU $\geqslant 0.7$) is normally required (e.g, consider the task of a robotic arm that must grasp an object). Such a need is also reflected in the very recently introduced *COCO* detection challenge [91], which uses as evaluation metric the traditional average precision (AP) measurement but averaged over multiple IoU thresholds between 0.5 (loosely localized object) and 1.0 (perfectly localized object) so as to reward detectors that exhibit good localization accuracy.

Therefore, proposing detectors that exhibit highly accurate (and not loose) localization of the ground truth objects should be one of the major challenges in object detection. The aim of this work is to take a further step towards addressing this challenge. In practical terms, our goal is to boost the bounding box detection AP performance across a wide range

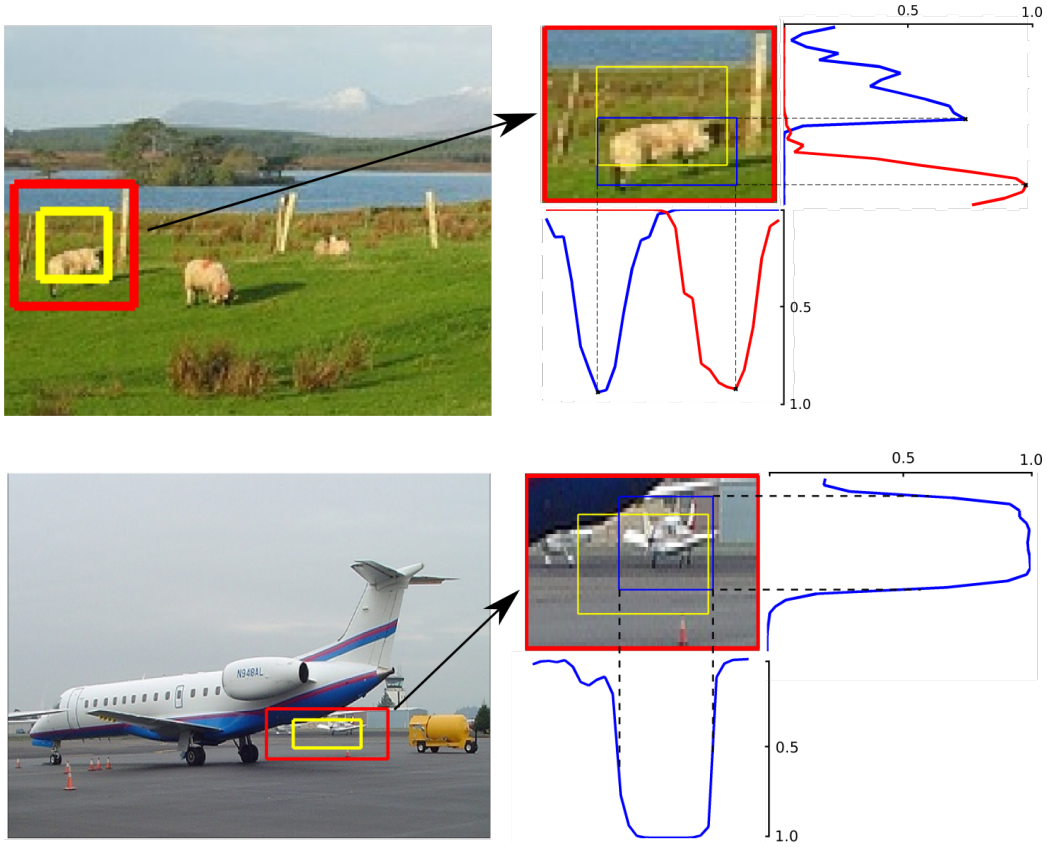


Figure 3-1: Illustration of the basic work-flow of our localization module. ***Left column:*** given a candidate box B (yellow box), our model "looks" on a search region R (red box), which is obtained by enlarging box B by a constant factor, in order to localize the bounding box of an object of interest. ***Right column:*** to localize a bounding box the model assigns one or more probabilities on each row and independently on each column of region R. Those probabilities can be either the probability of an element (row or column) to be one of the four object borders (see top-right image), or the probability for being on the inside of an objects bounding box (see bottom-right image). In either case the predicted bounding box is drawn with blue color.

of IoU thresholds (i.e., not just for IoU threshold of 0.5 but also for values well above that). To that end, a main technical contribution of this work is to propose a novel *object localization model* that, given a loosely localized search region inside an image, aims to return the accurate location of an object in this region (see Figure 3-1).

A crucial component of this new model is that it does not rely on the commonly used bounding box regression paradigm, which uses a regression function to directly predict the object bounding box coordinates. Indeed, the motivation behind our work stems from the belief that trying to directly regress to the target bounding box coordinates, constitutes a difficult learning task that cannot yield accurate enough bounding boxes. We argue that it is

far more effective to attempt to localize a bounding box by first assigning a probability to each row and independently to each column of the search region for being the left, right, top, or bottom borders of the bounding box (see Fig. 3-1 top) or for being on the inside of an object's bounding box (see Fig. 3-1 bottom). In addition, this type of probabilities can provide a measure of confidence for placing the bounding box on each location and they can also handle instances that exhibit multi-modal distributions for the border locations. They thus yield far more detailed and useful information than the regression models that just predict 4 real values that correspond to estimations of the bounding box coordinates. Furthermore, as a result of this, we argue that the task of learning to predict these probabilities is an easier one to accomplish.

To implement the proposed localization model, we rely on a convolutional neural network model, which we call *LocNet*, whose architecture is properly adapted such that the amount of parameters needed on the top fully connected layers is significantly reduced, thus making our LocNet model scalable with respect to the number of object categories.

Importantly, such a localization module can be easily incorporated into many of the current state-of-the-art object detection systems [40, 42, 127], helping them to significantly improve their localization performance. Here we use it in an iterative manner as part of a detection pipeline that utilizes a recognition model for scoring candidate bounding boxes provided by the aforementioned localization module, and show that such an approach significantly boosts AP performance across a broad range of IoU thresholds.

Furthermore, inspired by the high localization accuracy of our methodology in the detection task, we decided to adapt it to the category agnostic object proposal generation task. The definition of this task is that for a given image a small set of bounding boxes must be generated that will cover with high recall all the objects that appear in the image regardless of their semantic category. In object detection, applying the recognition models to such a reduced set of category independent location hypotheses [43] instead of an exhaustive scan of the entire image [35, 137], has the advantage of drastically reducing the amount of recognition model evaluations and thus allowing the use of more sophisticated machinery for that purpose. As a result, proposal based detection systems manage to achieve state-of-the-art results and have become the dominant paradigm in the object detection

literature [43, 55, 42, 40, 41, 127, 164, 5, 142]. However, the usefulness of object proposals (aka box proposals) is not limited only to the object detection task. To the contrary, they are are utilized as a core component in many image understanding tasks such as weakly-supervised object detection [15], exemplar 2D-3D detection [100], visual semantic role labelling [49], caption generation [70], or visual question answering [138]. Due to that, the box proposal generation task has received an increased amount of attention over the last years. In our case, we exploit the two key ingredients of our overall object localization methodology (devised for the detection task), which are iterative localization and the LocNet bounding box localization module, in order to build an active box proposal generation system that starts from a set of seed boxes, which are uniformly distributed on the image, and then progressively (i.e., in iterative manner) moves its attention on the promising image areas where it is more likely to discover well localized bounding box proposals. We call our approach AttractioNet and a core component of it is a category agnostic version of the LocNet module that is capable of yielding accurate and robust bounding box predictions regardless of the object category.

To summarize, our contributions in this chapter are as follows:

- We cast the problem of localizing an object's bounding box as that of assigning probabilities on each row and column of a search region. Those probabilities represent either the likelihood of each element (row or column) to belong on the inside of the bounding box or the likelihood to be one of the four borders of the object. Both cases are studied and compared with the bounding box regression model. To implement the above model, we propose a properly adapted convolutional neural network architecture that has a reduced number of parameters and results in an efficient and accurate object localization network named LocNet.
- We extensively evaluate our approach on VOC2007 [33] and we show that it achieves a very significant improvement over the bounding box regression with respect to the mAP for IoU threshold of 0.7 and the COCO style of measuring the mAP. It also offers an improvement with respect to the traditional way of measuring the mAP (i.e., for IoU $\geqslant 0.5$), achieving in this case 78.4% and 74.78% mAP on VOC2007 [33] and VOC2012 [32] test sets, which were the state-of-the-art when finishing the work of

this chapter. Given those results we believe that our localization approach could very well replace the existing bounding box regression paradigm in future object detection systems.

- Furthermore, we adapted the overall localization methodology of our detection system to the object box proposal generation task. The resulting box proposal system, which we call *AttractioNet*: *(Att)end (R)efine Repeat: (Act)ive Box Proposal Generation via (I)n-(O)ut Localization (Net)work*, is evaluated both on PASCAL and on the more challenging COCO datasets and it demonstrates significant improvement with respect to the state-of-the-art on box proposal generation. Furthermore, we provide strong evidence that our object location refinement module is capable of generalizing to unseen categories by reporting results for the unseen categories of ImageNet detection task and NYU-Depth dataset. Finally, we evaluate our box proposal generation approach in the context of the object detection task using a VGG16-Net based detection system and the achieved average precision performance on the COCO test-dev set manages to significantly surpass all other VGG16-Net based detection systems while even being on par with the ResNet-101 based detection system of He et al. [57].

The remainder of the chapter is structured as follows. We describe related work in §3.2. Then we present our localization methodology for the object detection task with its experimental evaluation in §3.3. Its adaption to the box proposal generation task and the corresponding experimental results are provided in §3.4. Finally, we conclude in §3.5.

3.2 Related work

Here we describe related work in the object detection and category agnostic box proposal generation tasks.

Object detection. Most of the recent literature on object detection, treats the object localization problem at pre-recognition level by incorporating category-agnostic object proposal algorithms [153, 177, 120, 2, 75, 76, 4, 150, 149]. Those proposals are later classified by a category-specific recognition model in order to create the final list of detections [43]. Instead, in our work we focus on boosting the localization accuracy at

post-recognition time, at which the improvements can be complementary to those obtained by improving the pre-recognition localization. Till now, the work on this level has been limited to the bounding box regression paradigm that was first introduced by Felzenszwalb et al. [35] and ever-since it has been used with success on most of the recent detection systems [43, 42, 127, 137, 55, 172, 176, 128, 113]. Given an initial candidate box that is loosely localized around an object, a regression model tries to predict the coordinates of its ground truth bounding box. Lately this model is enhanced by high capacity convolutional neural networks to further improve its localization capability [40, 42, 137, 127]. There are also some approaches that follow an iterative localization methodology for the object detection task [8, 44, 160]. More notably, Caicedo et al. [8] and Yoo et al. [160] attempt to localize an object by sequentially choosing one among a few possible actions that either transform the bounding box or stop the searching procedure. We also follow an iterative localization methodology in our work that is based however on a very accurate bounding box localization module.

Category agnostic box proposal generation. Several approaches have been proposed in the literature for this task [153, 4, 177, 76, 75, 2, 94, 14, 13, 17]. Among them our work is most related to the CNN-based objectness scoring approaches [79, 39, 120] that recently have demonstrated state-of-the-art results [120, 121]. In the objectness scoring paradigm, a large set of image boxes is ranked according to how likely it is for each image box to tightly enclose an object — regardless of its category — and then this set is post-processed with a non-maximum-suppression step and truncated to yield the final set of object proposals. In this context, Kuo et al. [79] with their DeepBox system demonstrated that training a convolutional neural network to perform the task of objectness scoring can yield superior performance over previous methods that were based on low level cues and they provided empirical evidence that it can generalize to unseen categories. In order to avoid evaluating the computationally expensive CNN-based objectness scoring model on hundreds of thousands image boxes, which is necessary for achieving good localization of all the objects in the image, they use it only to re-rank the proposals generated from a faster but less accurate proposal generator thus being limited by its localization performance. Instead, more recent CNN-based approaches apply their models only to ten of thousands

image boxes, uniformly distributed in the image, and jointly with objectness prediction they also infer the bounding box of the closest object to each input image box. Specifically, the Region Proposal Network in Faster-RCNN [127] performs bounding box regression for that purpose while the DeepMask method predicts the foreground mask of the object centred in the image box and then it infers the location of the object's bounding box by extracting the box that tightly encloses the foreground pixels. The latter has demonstrated state-of-the-art results and was recently extended with a top-down foreground mask refinement mechanism that exploits the convolutional feature maps at multiple depths of a neural network [121]. Our work is also based on the paradigm of having a CNN model that, given an image box, jointly predicts its objectness and a new bounding box that is better aligned on the object that it contains. However, we advance the state-of-the-art on box proposal generation by improving the aforementioned paradigm in two ways: **(1)** implementing the object's bounding box prediction step with a category agnostic LocNet model, and **(2)** actively generating the set of image boxes that will be processed by the CNN model.

3.3 Object Localization Methodology

3.3.1 Overview

Algorithm 1: Object detection pipeline

Input : Image I, initial set of candidate boxes $\mathbf{B}^1$
Output : Final list of detections $\mathbf{Y}$
for $t \leftarrow 1$ ***to*** T **do**
  $\mathbf{S}^t \leftarrow \mathit{Recognition}(\mathbf{B}^t|I)$
  if $t < T$ **then**
    $\mathbf{B}^{t+1} \leftarrow \mathit{Localization}(\mathbf{B}^t|I)$
  end
end
$\mathbf{D} \leftarrow \cup_{t=1}^{T}\{\mathbf{S}^t, \mathbf{B}^t\}$
$\mathbf{Y} \leftarrow \mathit{PostProcess}(\mathbf{D})$

Our detection pipeline includes two basic components, the recognition and the localization models, integrated into an iterative scheme (see Algorithm 1). This scheme starts

from an initial set of candidate boxes $\mathbf{B}^1$ (which could be, e.g, either dense sliding windows [137, 116, 126, 86] or category-agnostic bounding box proposals [177, 153, 127]) and on each iteration t it uses the two basic components in the following way:

Recognition model: Given the current set of candidate boxes $\mathbf{B}^t = \{B_i^t\}_{i=1}^{N_t}$, it assigns a confidence score to each of them $\mathbf{S}^t = \{s_i^t\}_{i=1}^{N_t}$ that represents how likely it is for those boxes to be localized on an object of interest.

Localization model: Given the current set of candidate boxes $\mathbf{B}^t = \{B_i^t\}_{i=1}^{N_t}$, it generates a new set of candidate boxes $\mathbf{B}^{t+1} = \{B_i^{t+1}\}_{i=1}^{N_{t+1}}$ such that those boxes will be closer (i.e., better localized) to the objects of interest (so that they are probably scored higher from the recognition model).

In the end, the candidate boxes that were generated on each iteration from the localization model along with the confidence scores that were assigned to them from the recognition model are merged together and a post-processing step of non-max-suppression [35] followed by bounding box voting [40] (which is described in §2.5) applied to them. The output of this post-processing step consists the detections set produced by our pipeline. Both the recognition and the localization models are implemented as convolutional neural networks [83, 144, 78, 54]. More details about our detection pipeline are provided in appendix A.1.

The focus of this section is to improve the localization model of this pipeline. The abstract work-flow that we use for this localization model is that it gets as input a candidate box B in the image, it enlarges it by a factor γ to create a search region R and then it returns a new candidate box that ideally will tightly enclose an object of interest in this region (see right column of Figure 3-1). The crucial question is, of course, what is the most effective approach for constructing a model that is able to generate a good box prediction. One choice could be, for instance, to learn a regression function that directly predicts the 4 bounding box coordinates (see for more details section 3.3.2). However, we argue that this is not the most effective solution. Instead, we opt for a different approach, which is detailed in section 3.3.4.2.

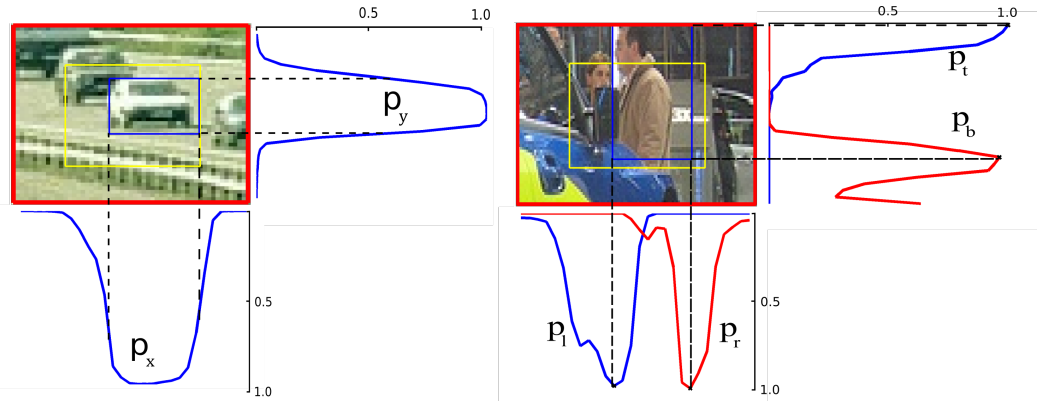


Figure 3-2: The posterior probabilities that our localization model yields given a region R. ***Left Image:*** the in-out conditional probabilities that are assigned on each row (p_y) and column (p_x) of R. They are drawn with the blues curves on the right and on the bottom side of the search region. ***Right Image:*** the conditional probabilities p_l, p_r, p_t, and p_b of each column or row to be the left (l), right (r), top (t) and bottom (b) border of an object's bounding box. They are drawn with blue and red curves on the bottom and on the right side of the search region.

3.3.2 Bounding box regression localization model

Here we describe in more detail the bounding box regression paradigm, which is used as a baseline for our approach. In this case the localization model consists of four (ConvNet-based) scalar regression functions $f_x(R, c)$, $f_y(R, c)$, $f_w(R, c)$, and $f_h(R, c)$ that given a regions R [1] and a category c, they actually predict the coefficients of a geometric transformation that will ideally map the search regions R to a ground truth bounding box of the c object category [43]. Specifically, if $R = (R_x, R_y, R_w, R_h)$ are the coordinates of the search region in form of its top-left corner (R_x, R_y) and its width and height (R_w, R_h), then the predicted candidate box $\hat{B} = (\hat{B}_x, \hat{B}_y, \hat{B}_w, \hat{B}_h)$ is given by the following equations:

$$\hat{B}_x = R_w \cdot f_x(R, c) + R_x \tag{3.1}$$

$$\hat{B}_y = R_h \cdot f_y(R, c) + R_y \tag{3.2}$$

$$\hat{B}_w = R_w \cdot \exp(f_w(R, c)) \tag{3.3}$$

$$\hat{B}_h = R_h \cdot \exp(f_h(R, c)). \tag{3.4}$$

[1] In many bounding box regression implementations the region R is identical to the input candidate box B.

Hence, the four scalar target regression values $T = \{t_x, t_y, t_w, t_h\}$ for the ground truth bounding box $B^{gt} = (B^{gt}_x, B^{gt}_y, B^{gt}_w, B^{gt}_h)$ are defined as:

$$t_x = \frac{B^{gt}_x - R_x}{R_w} \qquad t_y = \frac{B^{gt}_y - R_y}{R_h} \tag{3.5}$$

$$t_w = \log(\frac{B^{gt}_w}{R_w}) \qquad t_h = \log(\frac{B^{gt}_h}{R_h})). \tag{3.6}$$

3.3.3 LocNet: proposed localization model

Instead of the bounding box regression approach, we opt to formulate the localization problem in a dense classification way. More specifically, given a search region R and object category c, our object localization model considers a division of R in M equal horizontal regions (rows) as well as a division of R in M equal vertical regions (columns), and outputs for each of them one or more conditional probabilities. Each of these conditional probabilities is essentially a vector of the form $p^{R,c} = \{p(i|R,c)\}_{i=1}^{M}$ (hereafter we drop the R and c conditioning variables so as to reduce notational clutter). Two types of conditional probabilities are considered here:

In-Out probabilities: These are vectors $p_x = \{p_x(i)\}_{i=1}^{M}$ and $p_y = \{p_y(i)\}_{i=1}^{M}$ that represent respectively the conditional probabilities of each column and row of R to be inside the bounding box of an object of category c (see left part of Figure 3-2). A row or column is considered to be inside a bounding box if at least part of the region corresponding to this row or column is inside this box. For example, if B^{gt} is a ground truth bounding box with top-left coordinates (B^{gt}_l, B^{gt}_t) and bottom-right coordinates (B^{gt}_r, B^{gt}_b),[2] then the In-Out probabilities $p = \{p_x, p_y\}$ from the localization model should ideally be equal to the following target probabilities $T = \{T_x, T_y\}$:

$$\forall i \in \{1, \ldots, M\}, \;\; T_x(i) = \begin{cases} 1, & \text{if } B^{gt}_l \leq i \leq B^{gt}_r \\ 0, & \text{otherwise} \end{cases},$$

[2]We actually assume that the ground truth bounding box is projected on the output domain of our model where the coordinates take integer values in the range $\{1, \ldots, M\}$. This is a necessary step for the definition of the target probabilities

$$\forall i \in \{1,\dots,M\},\ \ T_y(i) = \begin{cases} 1, & \text{if } B_t^{gt} \leq i \leq B_b^{gt} \\ 0, & \text{otherwise} \end{cases}.$$

Border probabilities: These are vectors $p_l = \{p_l(i)\}_{i=1}^{M}, p_r = \{p_r(i)\}_{i=1}^{M}, p_t = \{p_t(i)\}_{i=1}^{M}$ and $p_b = \{p_b(i)\}_{i=1}^{M}$ that represent respectively the conditional probability of each column or row to be the left (l), right (r), top (t) and bottom (b) border of the bounding box of an object of category c (see right part of Figure 3-2). In this case, the target probabilities $T = \{T_l, T_r, T_t, T_b\}$ that should ideally be predicted by the localization model for a ground truth bounding box $B^{gt} = (B_l^{gt}, B_t^{gt}, B_r^{gt}, B_b^{gt})$ are given by

$$\forall i \in \{1,\dots,M\},\ \ T_s(i) = \begin{cases} 1, & \text{if } i = B_s^{gt} \\ 0, & \text{otherwise} \end{cases},$$

where $s \in \{l, r, t, b\}$. Note that we assume that the left and right border probabilities are independent and similarly for the top and bottom cases.

3.3.3.1 Bounding box inference

Given the above output conditional probabilities, we model the inference of the bounding box location $\hat{B} = (\hat{B}_l, \hat{B}_t, \hat{B}_r, \hat{B}_b)$ using one of the following probabilistic models:

In-Out ML: Maximizes the likelihood of the *in-out* elements of B

$$L_{\text{in-out}}(\hat{B}) = \prod_{i \in \{\hat{B}_l,\dots,\hat{B}_r\}} p_x(i) \prod_{i \in \{\hat{B}_t,\dots,\hat{B}_b\}} p_y(i) \prod_{i \notin \{\hat{B}_l,\dots,\hat{B}_r\}} \tilde{p}_x(i) \prod_{i \notin \{\hat{B}_t,\dots,\hat{B}_b\}} \tilde{p}_y(i), \tag{3.7}$$

where $\tilde{p}_x(i) = 1 - p_x(i)$ and $\tilde{p}_y(i) = 1 - p_y(i)$. The first two terms in the right hand of the equation represent the likelihood of the rows and columns of box $\hat{B}$ (*in*-elements) to be inside a ground truth bounding box and the last two terms the likelihood of the rows and columns that are not part of $\hat{B}$ (*out*-elements) to be outside a ground truth bounding box.

Borders ML: Maximizes the likelihood of the borders of box $\hat{B}$:

$$L_{\text{borders}}(\hat{B}) = p_l(\hat{B}_l) \cdot p_t(\hat{B}_t) \cdot p_r(\hat{B}_r) \cdot p_b(\hat{B}_b). \tag{3.8}$$

Combined ML: It uses both types of probability distributions by maximizing the likelihood for both the *borders* and the *in-out* elements of $\hat{B}$:

$$L_{\text{combined}}(\hat{B}) = L_{\text{borders}}(\hat{B}) \cdot L_{\text{in-out}}(\hat{B}). \tag{3.9}$$

3.3.3.2 Discussion

The reason we consider that the proposed formulation of the problem of localizing an object's bounding box is superior is because the In-Out or Border probabilities provide much more detailed and useful information regarding the location of a bounding box compared to the typical bounding box regression paradigm. In particular, in the latter case the model simply directly predicts real values that corresponds to estimated bounding box coordinates but it does not provide, e.g, any confidence measure for these predictions. On the contrary, our model provides a conditional probability for placing the four borders or the inside of an object's bounding box on each column and row of a search region R. As a result, it is perfectly capable of handling also instances that exhibit multi-modal conditional distributions (both during training and testing). During training, we argue that this makes the per row and per column probabilities much easier to be learned from a convolutional neural network that implements the model, than the bounding box regression task (e.g, see Figure 3-3), thus helping the model to converge to a better training solution. Indeed, as we demonstrate, e.g, in Figure 3-4, our CNN-based *In-Out ML* localization model converges faster and on higher localization accuracy (measured with the mAR [64] metric) than a CNN-based bounding box regression model [42, 40]. This behaviour was consistently observed in all of our proposed localization models. Furthermore, during testing, these conditional distributions as we saw can be exploited in order to form probabilistic models for the inference of the bounding box coordinates.

Alternatively to our approach, we could predict the probability of each pixel to belong

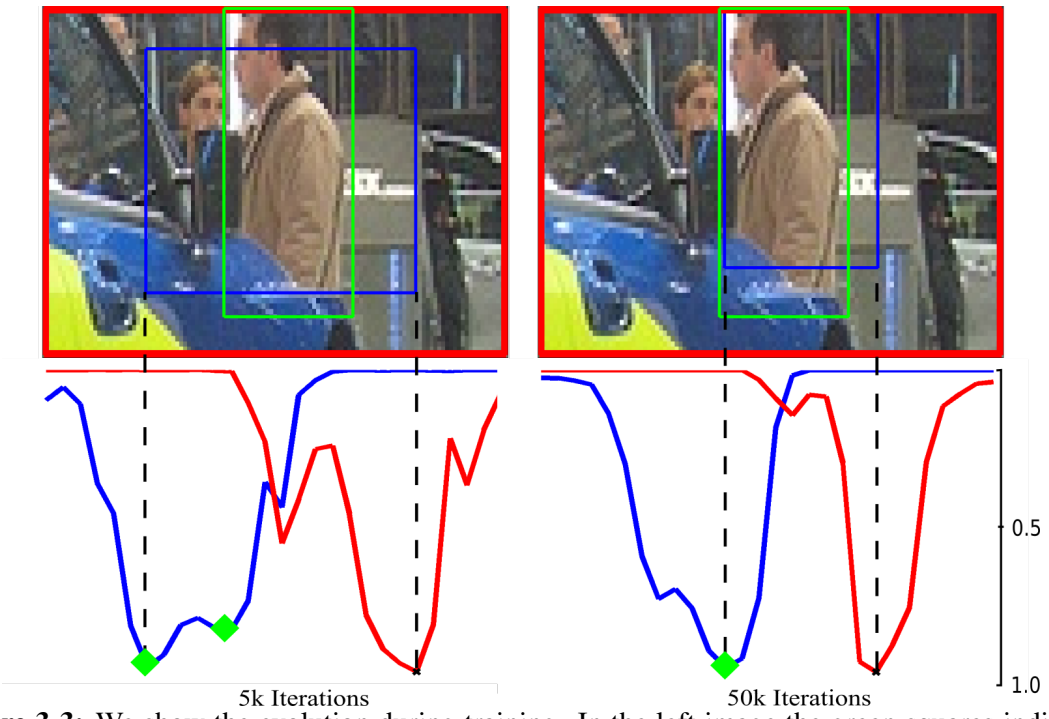


Figure 3-3: We show the evolution during training. In the left image the green squares indicate the two highest modes of the left border probabilities predicted by a network trained only for a few iterations (5k). Despite the fact that the highest one is erroneous, the network also maintains information for the correct mode. As training progresses (50k), this helps the network to correct its mistake and recover a correct left border(right image).

on the foreground of an object, as Pinheiro et al. [120] does. However, in order to learn such a type of model, pixel-wise instance segmentation masks are required during training, which in general is a rather tedious task to collect. In contrary, for our model to learn those per row and per column probabilities, only bounding box annotations are required. Even more, this independence is exploited in the design of the convolutional neural network that implements our model in order to keep the number of parameters of the prediction layers small (see § 3.3.3.3). This is significant for the scalability of our model with respect to the number of object categories since we favour category-specific object localization that has been shown to exhibit better localization accuracy [144].

3.3.3.3 Network architecture

Our localization model is implemented through the convolutional neural network that is visualized in Figure 3-5 and which is called LocNet. The processing starts by forwarding the entire image I (of size $w_I \times h_I$), through a sequence of convolutional layers (conv. layers

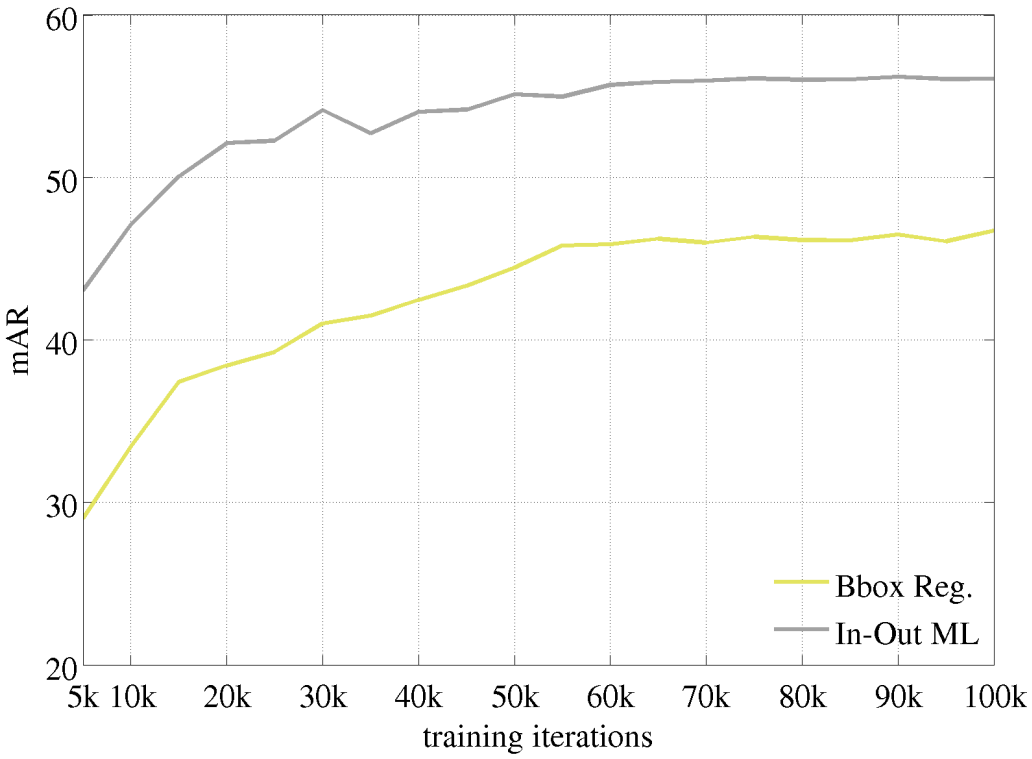


Figure 3-4: mAR as a function of the training iteration for the bounding box regression model (*Bbox reg.*) and the *In-Out ML* localization model. In order to create this plot, we created a small validation set of candidate boxes with a ground truth bounding box assigned on each of them, and during training given those candidates as input to the models we measure the mAR of the predicted boxes. We observe that the *In-Out ML* localization model converges faster and to a higher mAR than the *bounding box regression* localization model.

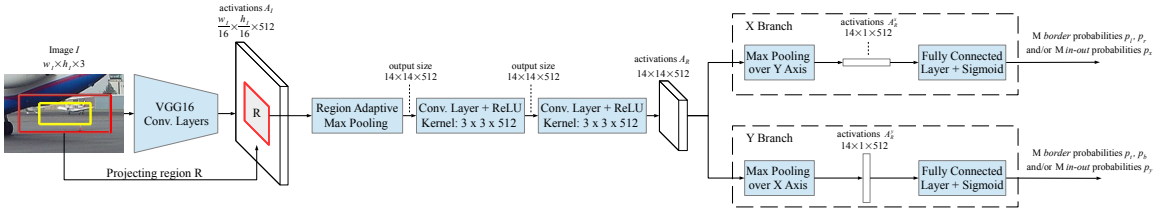


Figure 3-5: Visualization of the LocNet network architecture. In the input image, with yellow is drawn the candidate box B and with red the search region R. In its output, the LocNet network yields probabilities for each of the C object categories. The parameter M that controls the output resolution is set to the value 28 in our experiments. The convolutional layers of the VGG16-Net [144] that are being used in order to extract the image activations A_I are those from conv1_1 till conv5_3. The new layers that are not derived from the VGG16-Net [144], are randomly initialized with a Gaussian distribution with standard deviation of 0.001 for the hidden layers and 0.01 for the final fully connected layers.

of VGG16 [144]) that outputs the A_I activation maps (of size $\frac{w_I}{16} \times \frac{h_I}{16} \times 512$). Then, the region R is projected on A_I and the activations that lay inside it are cropped and pooled with a spatially adaptive max-pooling layer [55]. The resulting fixed size activation maps

($14 \times 14 \times 512$) are forwarded through two convolutional layers (of kernel size $3 \times 3 \times 512$), each followed by ReLU non-linearities, that yield the localization-aware activation maps A_R of region R (with dimensions size $14 \times 14 \times 512$).

At this point, given the activations A_R the network yields the probabilities that were described in section §3.3.3. Specifically, the network is split into two branches, the *X* and *Y*, with each being dedicated for the predictions that correspond to the dimension (x or y respectively) that is assigned to it. Both start with a max-pool layer that aggregates the A_R activation maps across the dimension perpendicular to the one dedicated to them, i.e.,

$$A_R^x(i, f) = \max_j A_R(i, j, f), \tag{3.10}$$

$$A_R^y(j, f) = \max_i A_R(i, j, f), \tag{3.11}$$

where i,j,and f are the indices that span over the width, height, and feature channels of A_R respectively. The resulted activations A_R^x and A_R^y (both of size 14×512) efficiently encode the object location only across the dimension that their branch handles. This aggregation process could also be described as marginalizing-out localization cues irrelevant for the dimension of interest. Finally, each of those aggregated features is fed into the final fully connected layer that is followed by sigmoid units in order to output the conditional probabilities of its assigned dimension. Specifically, the *X* branch outputs the p_x and/or the (p_l, p_r) probability vectors whereas the *Y* branch outputs the p_y and/or the (p_t, p_b) probability vectors. Despite the fact that the last fully connected layers output category-specific predictions, their number of parameters remains relatively small due to the facts that: 1) they are applied on features of which the dimensionality has been previously drastically reduced due to the max-pooling layers of equations 3.10 and 3.11, and 2) that each branch yields predictions only for a single dimension.

3.3.3.4 Training

During training, the localization network learns to map a search regions R (created by enlarging a candidate box B) in an image I to the target probabilities T that are conditioned on the object category c. Given a set of N^L training samples $\{(B_k, T_k, c_k, I_k)\}_{k=1}^{N^L}$ the loss

function that is minimized is

$$\frac{1}{N^L}\sum_{k=1}^{N^L} L_{loc}(\theta|B_k, T_k, c_k, I_k), \tag{3.12}$$

where θ are the network parameters that are learned and $L_{loc}(\theta|B, T, c, I)$ is the loss for one training sample.

Both for the *In-Out* and the *Borders* probabilities we use the sum of binary logistic regression losses per row and column. Specifically, the per sample loss of the *In-Out* case is:

$$\sum_{a\in\{x,y\}}\sum_{i=1}^{M} T_a(i)\log(p_a(i)) + \tilde{T}_a(i)\log(\tilde{p}_a(i))\,, \tag{3.13}$$

and for the *Borders* case is:

$$\sum_{s\in\{l,r,u,b\}}\sum_{i=1}^{M} \lambda^+ T_s(i)\log(p_s(i)) + \lambda^- \tilde{T}_s(i)\log(\tilde{p}_s(i))\,, \tag{3.14}$$

where $\tilde{T} = 1 - T$. In objective function (3.14), λ^+ and λ^- represent the weightings of the losses for misclassifying a border and a non-border element respectively. These are set as

$$\lambda^- = 0.5 \cdot \frac{M}{M-1}\,,\;\; \lambda^+ = (M-1)\cdot\lambda^-\,,$$

so as to balance the contribution on the loss of those two cases (note that $\tilde{T}_s(i)$ will be non-zero $M-1$ times more than $T_s(i)$). We observed that this leads to a model that yields more "confident" probabilities for the borders elements. For the *Borders* case we also tried to use as loss function the Mean Square Error, while modifying the target probabilities to be Gaussian distributions around the border elements, but we did not observe an improvement in performance.

3.3.4 Experimental results

We empirically evaluate our localization models on PASCAL VOC detection challenge [31]. Specifically, we train all the recognition and localization models on VOC2007+2012 trainval

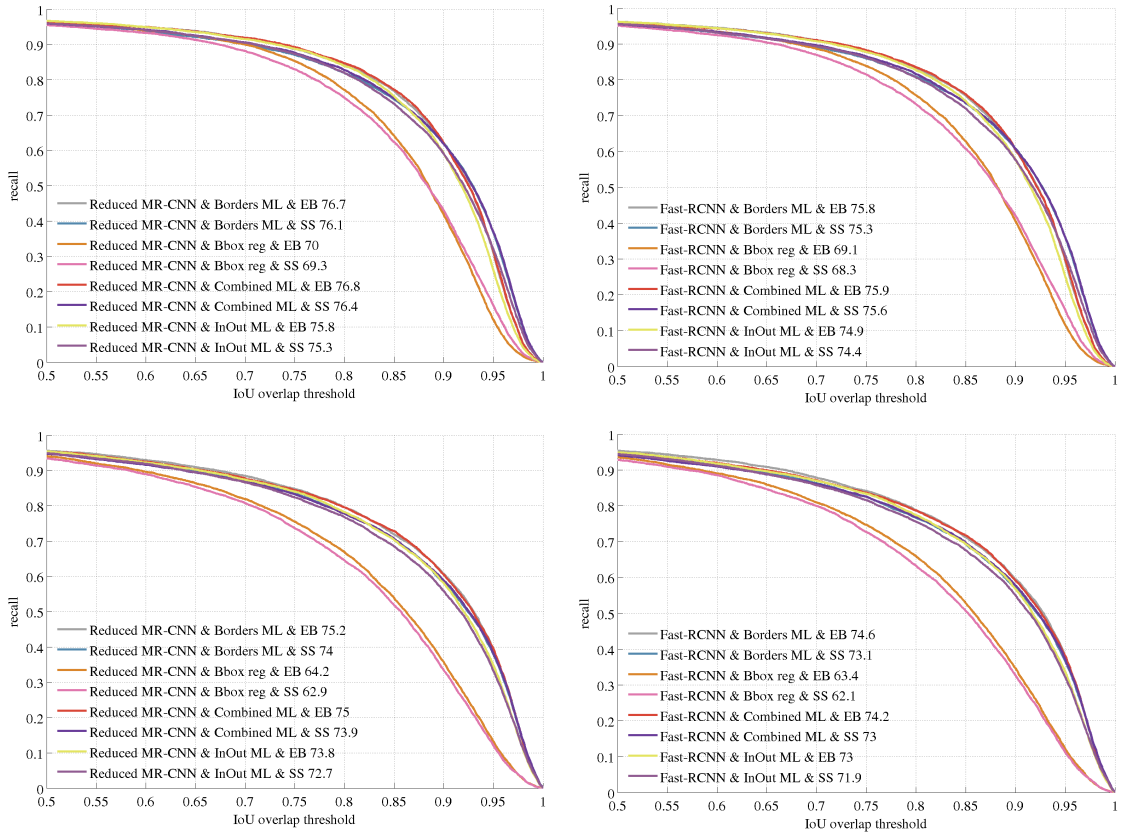


Figure 3-6: Recall of ground truth bounding boxes as a function of the IoU threshold on PASCAL VOC2007 test set. Note that, because we perform class-specific localization the recall that those plots report is obtained after averaging the per class recalls. ***Top-Left:*** Recalls for the *Reduced MR-CNN* model after one iteration of the detection pipeline. ***Bottom-Left:*** Recalls for the *Reduced MR-CNN* model after four iterations of the detection pipeline.***Top-Right:*** Recalls for the *Fast-RCNN* model after one iteration of the detection pipeline. ***Bottom-Right:*** Recalls for the *Fast-RCNN* model after four iterations of the detection pipeline.

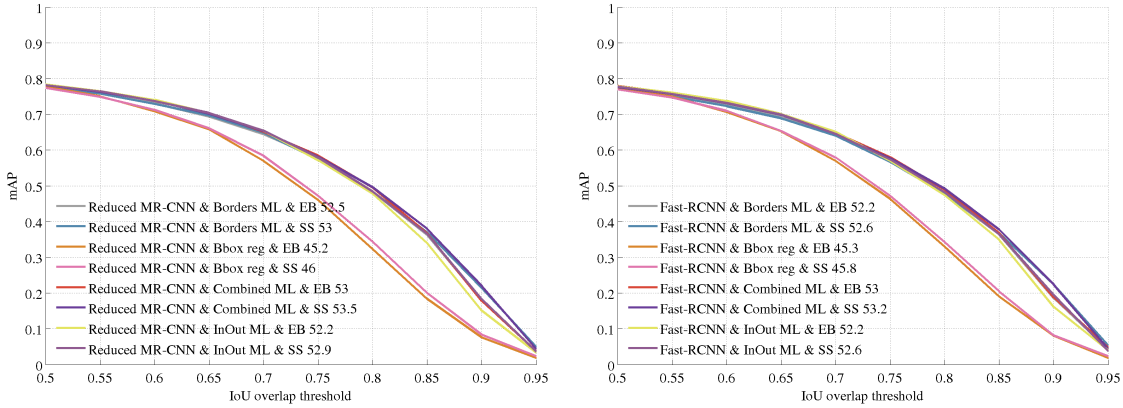


Figure 3-7: mAP as a function of the IoU threshold on PASCAL VOC2007 test set. ***Left plot:*** includes the configurations with the *Reduced-MR-CNN* recognition model. ***Right plot:*** includes the configurations with the *Fast-RCNN* recognition model.

sets and we test them on the VOC2007 test set. As baseline we use a ConvNet-based bounding box regression model [40]. The remaining components of the detection pipeline include:

Initial set of candidate boxes. We examine three alternatives for generating the initial set of candidate boxes: the Edge Box algorithm [177] (*EB*), the Selective Search algorithm (*SS*), and a sliding windows scheme. In Table 3.1 we provide the recall statistics of the those bounding box proposal methods.

Recognition model. For the recognition part of the detection system we use either the *Fast-RCNN [42]* or the *MR-CNN* recognition models that was presented in the previous section. During implementing the latter one, we performed several simplifications on its architecture and thus we call the resulting model *Reduced-MR-CNN* (those modifications are detailed in the subsection that follows).

In the remaining of this subsection, we fist provide implementation details of the localization and recognition models (§3.3.4.1), then we examine the performance of our approach with respect to localization (§3.3.4.2) and detection (§3.3.4.3) accuracy. We also report the detection accuracy of our approach for the sliding windows case (§3.3.4.4) and finally, we provide preliminary results of our approach on COCO detection challenge in §3.3.4.5 and qualitative results in §3.3.4.6.

3.3.4.1 Implementation details

For the implementation code of this section we make use of the Caffe framework [67]. During training of all the models (both the localization and the recognition ones) we fine-tune only from the conv4_1 convolutional layer and above. As training samples we use both selective search [153] and edge box [177] proposals. Finally, both during training and testing we use a single image scale that is obtained after resizing the image such as its smallest dimension to be 600 pixels.

Proposed localization models (*In-Out ML*, *Borders ML*, *Combined ML*): In order to create the training samples we take proposals of which the IoU with a ground truth bounding box is at least 0.4, we enlarge them by a factor of 1.8 in order to obtain the search regions R, and we assign to them the ground truth bounding box with which the original box proposal

has the highest IoU in order to obtain the target bounding boxes and the corresponding target vectors T. This process is performed independently for each category. The parameter M that controls the output resolution of our networks, is set to the value 28. For optimization we use stochastic gradient descend (SGD) with mini-batch size of 128 training candidate boxes. To speed up the training procedure we follow the paradigm of Fast-RCNN [42] and those 128 training candidate boxes are coming from only two images on each mini-batch. The weight decay is set to 0.00005 and the learning rate is set to 0.001 and is reduced by a factor of 10 after each $60k$ iterations. The overall training procedure is continued for up to $150k$ iterations and it takes around 1.5 days in one NVIDIA Titan Black GPU.

Bounding box regression localization model: In our comparative experiments, for the CNN architecture that implements the bounding box regression model we adopt the one described in the previous chapter (i.e., in 2.6). As a loss function we use the sum of sEuclidean distances between the target values and the predicted values of each training sample. The final fully connected layer is initialized from a Gaussian distribution with standard deviation of 0.01. The rest of training details (i.e., SGD, mini-batch, definition of training samples) are similar to those used for the proposed localization models. As proposed in Fast-RCNN [42], when training the bounding box regression model we simultaneously train the Fast-RCNN recognition model with the two models sharing their convolutional layers. In our experiments, this way of training improves the accuracy of both the Fast-RCNN recognition model and the bounding box regression model. On the contrary, (for simplicity) the newly proposed localization models (i.e., *Borders ML*, *In-Out ML*, and *Combined ML*) are not trained simultaneously with the recognition model. We expect that the joint training of these models with the recognition model can help to further improve their overall performance.

Reduced MR-CNN recognition model: We based the implementation of this model on the MR-CNN detection system that was described in the previous chapter. In this implementation however, for efficiency reasons and in order to speed up the experiments, we applied the following reductions: we include only six out of the ten regions proposed, by skipping the half regions; we do not include the semantic segmentation-aware CNN features; and we reduce the total amount of parameters on the region adaptation modules. In

order to achieve the reduction of parameters on the hidden fully connected layers fc6 and fc7 of the region adaptation modules, each of them is decomposed in two fully connected layers without any non-linearities between them. Specifically, the fc6 layer with weight matrix $W_6 : 25088 \times 4096$ is decomposed in the layers fc6_1 and fc6_2 with weight matrices $W_{6.1} : 25088 \times 1024$ and $W_{6.2} : 1024 \times 4096$ correspondingly. The fc7 layer with weight matrix $W_7 : 4096 \times 4096$ is decomposed in the layers fc7_1 and fc7_2 with weight matrices $W_{7.1} : 4096 \times 256$ and $W_{7.2} : 256 \times 4096$ correspondingly. To train the Reduced MR-CNN network, we first train only the original candidate box region of it without reducing the parameters of the fc6 and fc7 layers. Then, we apply the truncated SVD decomposition on the aforementioned layers (for more details see section §3.1 of [42]) that results in the layers fc6_1, fc6_2, fc7_1, and fc7_2. We copy the parameters of the resulting fully connected layers to the corresponding layers of the remaining region adaptation modules of the model and we continue training.

Fast-RCNN recognition model: We re-implemented Fast-RCNN based on the publicly available code provided from Fast-RCNN [42] and Faster-RCNN [127]. Here we will describe only the differences of our implementation with the original Fast-RCNN system [42]. In our implementation, we have different branches for the recognition of a candidate box and for its bounding box regression after the last convlutional layer (conv5_3 for the VGG16-Net [144]) that do not share any weights. In contrary, the original Fast-RCNN model splits to two branches after the last hidden layer. We applied this modification because, in our case, the candidate box that is fed to the regression branch is enlarged by a factor $\alpha = 1.3$ while the candidate box that is fed to recognition branch is not. Also, after the fine-tuning, we remove the softmax layer of the recognition branch and we train linear SVMs on top of the features that the last hidden layer of the recognition branch yields, just as R-CNN [43] does. Finally, we do not reduce the parameters of the fully connected layers by applying the truncated SVD decomposition on them as in the original paper. In our experiments those changes improved the detection performance of the model.

Both the Fast-RCNN and Reduced-MR-CNN models use as top classification layers class-specific linear SVMs [43].

Initial set of candidate boxes	Number	Recall		
		IoU≥0.5	IoU≥0.7	mAR
Sliding Windows	around 10k	0.920	0.389	0.350
Edge Box	around 2k	0.928	0.755	0.517
Sel. Search	around 2k	0.936	0.687	0.528

Table 3.1: Recall statistics on VOC2007 test set of the box proposals methods that we use in our work in order to generate the initial set of candidate boxes.

3.3.4.2 Localization performance

We first evaluate merely the localization performance of our models, thus ignoring in this case the recognition aspect of the detection problem. For that purpose we report the recall that the examined models achieve. Specifically, in Figure 3-6 we provide the recall as a function of the IoU threshold for the candidate boxes generated on the first iteration and the last iteration of our detection pipeline. Also, in the legends of these figures we report the average recall (AR) [64] that each model achieves. Note that, given the set of initial candidate boxes and the recognition model, the input to the iterative localization mechanism is exactly the same and thus any difference on the recall is solely due to the localization capabilities of the models. We observe that for IoU thresholds above 0.65, the proposed models achieve higher recall than bounding box regression and that this improvement is actually increased with more iterations of the localization module. Also, the AR of our proposed models is on average 6 points higher than bounding box regression.

3.3.4.3 Detection performance

Here we evaluate the detection performance of the examined localization models when plugged into the detection pipeline that was described in §3.3.1. In Table 3.2 we report the mAP on VOC2007 test set for IoU thresholds of 0.5 and 0.7 as well as the COCO style of mAP that averages the traditional mAP over various IoU thresholds between 0.5 and 1.0. The results that are reported are obtained after running the detection pipeline for $T = 4$ iterations. We observe that the proposed *InOut ML*, *Borders ML*, and *Combined ML* localization models offer a significant boost on the mAP for IoU ≥ 0.7 and the COCO style mAP, relative to the bounding box regression model (*Bbox reg.*) under all the tested cases. The improvement on both of them is on average 7 points. Our models also improve for the

Detection Pipeline			mAP		
Localization	Recognition	Initial Boxes	IoU ≥ 0.5	IoU ≥ 0.7	*COCO* style
–	*Reduced-MR-CNN*	2*k Edge Box*	0.747	0.434	0.362
InOut ML	*Reduced-MR-CNN*	2*k Edge Box*	0.783	**0.654**	0.522
Borders ML	*Reduced-MR-CNN*	2*k Edge Box*	0.780	0.644	0.525
Combined ML	*Reduced-MR-CNN*	2*k Edge Box*	**0.784**	0.650	0.530
Bbox reg.	*Reduced-MR-CNN*	2*k Edge Box*	0.777	0.570	0.452
–	*Reduced-MR-CNN*	2*k Sel. Search*	0.719	0.456	0.368
InOut ML	*Reduced-MR-CNN*	2*k Sel. Search*	0.782	**0.654**	0.529
Borders ML	*Reduced-MR-CNN*	2*k Sel. Search*	0.777	0.648	0.530
Combined ML	*Reduced-MR-CNN*	2*k Sel. Search*	0.781	0.653	**0.535**
Bbox reg.	*Reduced-MR-CNN*	2*k Sel. Search*	0.774	0.584	0.460
–	*Fast-RCNN*	2*k Edge Box*	0.729	0.427	0.356
InOut ML	*Fast-RCNN*	2*k Edge Box*	0.779	0.651	0.522
Borders ML	*Fast-RCNN*	2*k Edge Box*	0.774	0.641	0.522
Combined ML	*Fast-RCNN*	2*k Edge Box*	0.780	0.648	0.530
Bbox reg.	*Fast-RCNN*	2*k Edge Box*	0.773	0.570	0.453
–	*Fast-RCNN*	2*k Sel. Search*	0.710	0.446	0.362
InOut ML	*Fast-RCNN*	2*k Sel. Search*	0.777	0.645	0.526
Borders ML	*Fast-RCNN*	2*k Sel. Search*	0.772	0.640	0.526
Combined ML	*Fast-RCNN*	2*k Sel. Search*	0.775	0.645	0.532
Bbox reg.	*Fast-RCNN*	2*k Sel. Search*	0.769	0.579	0.458

Table 3.2: mAP results on VOC2007 test set for IoU thresholds of 0.5 and 0.7 as well as the COCO style mAP that averages the traditional AP for various IoU thresholds between 0.5 and 1 (specifically the thresholds 0.5:0.05:95 are being used). The hyphen symbol (–) indicates that the localization model was not used at all and that the pipeline ran only for $T = 1$ iteration. The other entries are obtained after running the detection pipeline for $T = 4$ iterations.

Year	Metric	Approach	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
2007	IoU ≥ 0.5	*Reduced-MR-CNN & Combined ML & EB*	0.804	0.855	0.776	0.729	0.622	0.868	0.875	0.886	0.613	0.860	0.739	0.861	0.870	0.826	0.791	0.517	0.794	0.752	0.866	0.777	0.784
2007	IoU ≥ 0.7	*Reduced-MR-CNN & In Out ML & EB*	0.707	0.742	0.622	0.481	0.452	0.840	0.747	0.786	0.429	0.730	0.670	0.754	0.779	0.669	0.581	0.309	0.655	0.693	0.736	0.690	0.654
2007	COCO style	*Reduced-MR-CNN & Combined ML & SS*	0.580	0.603	0.500	0.413	0.367	0.703	0.631	0.661	0.357	0.581	0.500	0.620	0.625	0.545	0.494	0.269	0.522	0.579	0.602	0.555	0.535
2012	IoU ≥ 0.5	*Reduced-MR-CNN & In Out ML & EB*	0.863	0.830	0.761	0.608	0.546	0.799	0.790	0.906	0.543	0.816	0.620	0.890	0.857	0.855	0.828	0.497	0.766	0.675	0.832	0.674	0.748
2012	IoU ≥ 0.5	*Reduced-MR-CNN & Borders ML & EB*	0.865	0.827	0.755	0.602	0.535	0.791	0.785	0.902	0.533	0.800	0.607	0.886	0.857	0.848	0.826	0.496	0.765	0.673	0.831	0.676	0.743
2012	IoU ≥ 0.5	*Reduced-MR-CNN & Combined ML & EB*	0.866	0.834	0.765	0.604	0.544	0.798	0.786	0.902	0.546	0.810	0.618	0.889	0.857	0.847	0.828	0.498	0.763	0.678	0.830	0.679	0.747

Table 3.3: Per class AP results on VOC2007 and VO2012 test sets.

mAP with IoU≥ 0.5 case but with a smaller amount (around 0.7 points). In Figure 3-7 we plot the mAP as a function of the IoU threshold. We can observe that the improvement on the detection performance thanks to the proposed localization models starts to clearly appear on the 0.65 IoU threshold and then grows wider till the 0.9. In Table 3.3 we provide the per class AP results on VOC2007 for the best approach on each metric. In the same table we also report the AP results on VOC2012 test set but only for the IoU ≥ 0.5 case since this is

Detection Pipeline			mAP		
Localization	Recognition	Initial Boxes	IoU ≥ 0.5	IoU ≥ 0.7	*COCO* style
–	*Reduced-MR-CNN*	10*k Sliding Windows*	0.617	0.174	0.227
InOut ML	*Reduced-MR-CNN*	10*k Sliding Windows*	0.770	0.633	0.513
Borders ML	*Reduced-MR-CNN*	10*k Sliding Windows*	0.764	0.626	0.513
Combined ML	*Reduced-MR-CNN*	10*k Sliding Windows*	0.773	0.639	0.521
Bbox reg.	*Reduced-MR-CNN*	10*k Sliding Windows*	0.761	0.550	0.436

Table 3.4: mAP results on VOC2007 test set when using 10*k sliding windows* as initial set of candidate boxes. In order to generate the sliding windows we use the publicly available code that accompanies the work of Hosang et al. [64] that includes a sliding window implementation inspired by *BING* [14, 173]).

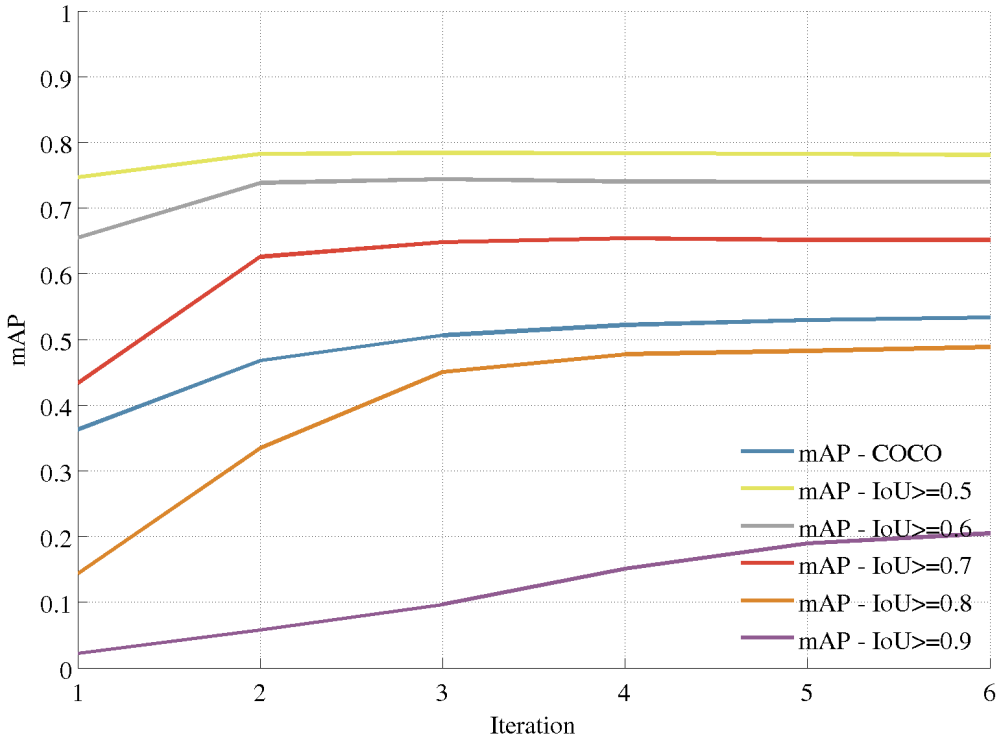


Figure 3-8: Plot of the mAP as a function of the iterations number of our detection pipeline on VOC2007 test set. To generate this plot we used the *Reduced-MR-CNN* recognition model with the *In-Out ML* localization model and Edge Box proposals.

the only metric that the evaluation server provides. In this dataset we achieve mAP of 74.8% which was the state-of-the-art at the time of working this chapter (6/11/2015). Finally, in Figure 3-8 we examine the detection performance behaviour with respect to the number of iterations used by our pipeline. We observe that as we increase the number of iterations, the mAP for high IoU thresholds (e.g., IoU ≥ 0.8) continues to improve while for lower thresholds the improvements stop on the first two iterations.

3.3.4.4 Sliding windows as initial set of candidate boxes

In Table 3.4 we provide the detection accuracy of our pipeline when, for generating the initial set of candidate boxes, we use a simple sliding windows scheme (of $10k$ windows per image). We observe that:

- Even in this case, our pipeline achieves very high mAP results that are close to the ones obtained with selective search or edge box proposals. We emphasize that this is true even for the IoU≥ 0.7 or the COCO style of mAP that favour better localized detections, despite the fact that in the case of sliding windows the initial set of candidate boxes is considerably less accurately localized than in the edge box or in the selective search cases (see Table 3.1).
- In the case of sliding windows, just scoring the candidate boxes with the recognition model (hyphen (–) case) yields much worse mAP results than the selective search or the edge box proposals case. However, when we use the full detection pipeline that includes localization models and re-scoring of the new better localized candidate boxes, then this gap is significantly reduced.
- The difference in the mAP results between the proposed localization models (*In-Out ML*, *Borders ML*, and *Combined ML*) and the *bounding box regression* model (*Bbox reg.*) is even greater in the case of sliding windows.

We note that we had not experimented with increasing the number of sliding windows. Also, the tested recognition model and localization models were not re-trained with sliding windows in the training set. As a result, we foresee that by exploring those two factors one might be able to further boost the detection performance for the sliding windows case.

3.3.4.5 Preliminary results on COCO

To obtain some preliminary results on COCO, we applied our training procedure on COCO train set. The only modification was to use $320k$ iterations (no other parameter was tuned). Therefore, LocNet results can still be significantly improved but the main goal was to show the relative difference in performance between the *Combined ML* localization model and the box regression model. Results are shown in Table 3.5, where it is observed that the proposed

Detection Pipeline			mAP			
Localization	Recognition	Proposals	Dataset	IoU ≥ 0.5	IoU ≥ 0.75	*COCO* style
Combined ML	*Fast R-CNN*	*Sel. Search*	$5K$ mini-val set	**0.424**	**0.282**	**0.264**
Bbox reg.	*Fast R-CNN*	*Sel. Search*	$5K$ mini-val set	0.407	0.202	0.214
Combined ML	*Fast R-CNN*	*Sel. Search*	test-dev set	0.429	0.279	0.263

Table 3.5 – Preliminary results on COCO. In those experiments the *Fast R-CNN* recognition model uses a softmax classifier [42] instead of class-specific linear SVMs [43] that are being used for the PASCAL experiments.

model boosts the mAP by 5 points in the COCO-style evaluation, 8 points in the $\mathrm{IoU} \geq 0.75$ case and 1.4 points in the $\mathrm{IoU} \geq 0.5$ case. More detection results on the COCO dataset are provided in §3.4.2.3.

3.3.4.6 Qualitative results

In Figure 3-9 we provide same qualitative results that compare the proposed localization models (*In-Out ML*, *Borders ML*, and *Combined ML*) with the *bounding box regression* localization model.

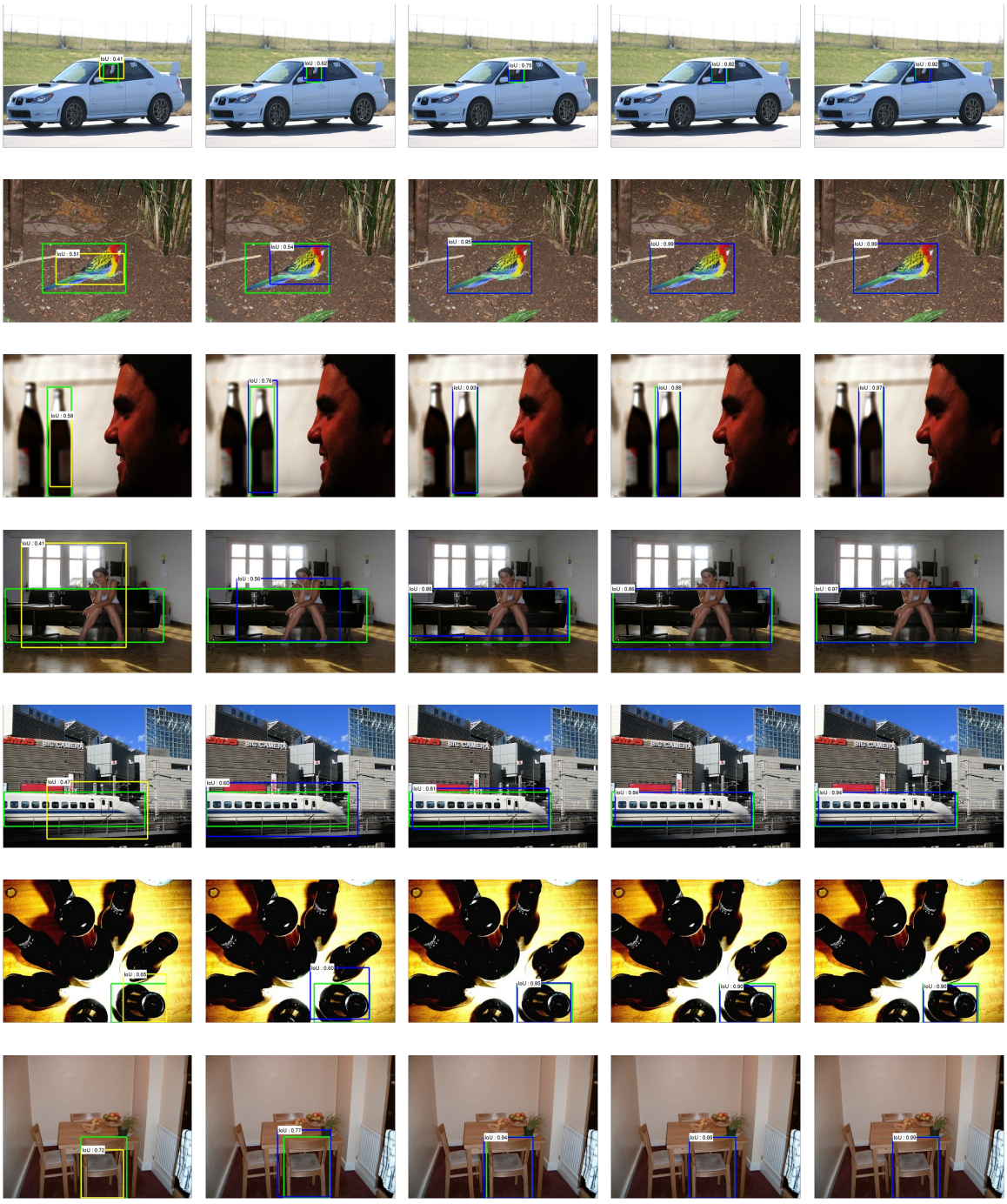
3.4 Adaption to the box proposal generation task

3.4.1 AttractioNet box proposals

Algorithm 2: ***Attend Refine Repeat Box Proposal Generation***

Input : Image I
Output : Bounding box proposals $\mathbf{P}$
$\mathbf{C} \leftarrow \emptyset$, $\mathbf{B}^0 \leftarrow$ seed boxes
for $t \leftarrow 1$ ***to*** T **do**
  `/* Attend & Refine procedure */`
  $\mathbf{O}^t \leftarrow$ *ObjectnessScoring*$(\mathbf{B}^{t-1}|I)$
  $\mathbf{B}^t \leftarrow$ *ObjectLocationRefinement*$(\mathbf{B}^{t-1}|I)$
  $\mathbf{C} \leftarrow \mathbf{C} \cup \{\mathbf{B}^t, \mathbf{O}^t\}$
end
$\mathbf{P} \leftarrow$ *NonMaxSuppression*$(\mathbf{C})$

Here we adapt the object localization methodology presented in the previous section, which was devised for the object detection task, to the box proposal generation task. So,



(a) Initial box **(b)** Regression **(c)** In-Out ML **(d)** Borders ML **(e)** Combined ML

Figure 3-9: Qualitative results of the bounding box localization step given an initial candidate box (column *(a)*) from the bounding box regression model (column *(b)*), the *In-Out ML* localization model (column *(c)*), the *Borders ML* localization model (column *(d)*), and the *Combined ML* localization model (column *(e)*). The candidate box is drawn with yellow color, the predicted boxes are drawn with blue color, and the ground truth bounding box is drawn with green color.

similarly to the object detection case, we employ an active box proposal generation strategy, which we call *Attend Refine Repeat* algorithm, that starts from a set of seed boxes, which only depend on the image size, and it then sequentially produces newer boxes that will better cover the objects of the image while avoiding the "objectless" image areas (see Figure 3-10). At the core of this algorithm lies a CNN-based box proposal network that implements two models:

Category agnostic localization model. This is the LocNet model presented in the previous section (i.e., in §3.3), and specifically its *In-Out ML* version, adapted to the category agnostic case. This means that, given an image I and a candidate box B, this model must estimate the coordinates of a new box $\tilde{B}$ that would be more tightly aligned on the object closest[3] to the input box B *regardless of what its semantic category might be* (as opposed to the LocNet model presented in §3.3 that performs category-specific object location refinement). The form of the In-Out probabilities that this category-agnostic LocNet model must predict are defined in §3.4.1.2.

Category agnostic objectness scoring model. Given a candidate box B and the image I, it scores the box B (with an objectness score O) based on how likely it is to enclose an object, regardless of its semantic category.

The pseudo-code of the *Attend Refine Repeat* algorithm is provided in Algorithm 2. Specifically, it starts by initializing the set of candidate boxes $\mathbf{C}$ to the empty set and then creates a set of seed boxes $\mathbf{B}^0$ by uniformly distributing boxes of various fixed sizes in the image (similar to Cracking Bing [173]). Then on each iteration t it estimates the objectness $\mathbf{O}^t$ of the boxes generated in the previous iteration, $\mathbf{B}^{t-1}$, and it refines their location (resulting in boxes $\mathbf{B}^t$) by attempting to predict the bounding boxes of the objects that are closest to them. The results $\{\mathbf{B}^t, \mathbf{O}^t\}$ of those operations are added to the candidates set $\mathbf{C}$ and the algorithm continues. In the end, non-maximum-suppression [35] is applied to the candidate box proposals $\mathbf{C}$ and the top K box proposals, set $\mathbf{P}$, are returned.

The advantages of having an algorithm that sequentially generates new box locations given the predictions of the previous stage are two-fold:

[3] By closest we mean the object whose bounding box has the highest intersection over union (IoU) overlap with the input box B.

Figure 3-10: Illustration of the image areas being attended by our box proposal generator algorithm at each iteration. In the first iteration the box proposal generator attends the entire image since the seed boxes are created by uniformly distributing boxes across the image. However, as the algorithm progresses its attention is concentrated on the image areas that actually contain objects.

- **Attention mechanism:** First, it behaves as an attention mechanism that, on each iteration, focuses more and more on the promising locations (in terms of box coordinates) of the image (see Figure 3-10). As a result of this, boxes that tightly enclose the image objects are more likely to be generated and to be scored with high objectness confidence.
- **Robustness to initial boxes:** Furthermore, it allows to refine some initially imperfect box predictions or to localize objects that might be far (in terms of center location, scale and/or aspect ratio) from any seed box in the image. This is illustrated via a few characteristic examples in Figure 3-11. As shown in each of these examples, starting from a seed box, the iterative bounding box predictions gradually converge to the closest (in terms of center location, scale and/or aspect ratio) object without actually being affected by any nearby instances.

Figure 3-11: Illustration of the consecutive bounding box predictions made by our category agnostic location refinement module. In each row, from left to right we depict a seed box (iteration 0) and the bounding box predictions in each iteration. Despite the fact that the seed box might be quite far from the object (in terms of center location, scale and/or aspect ratio) the refinement module has no problem in converging to the bounding box closest to the seed box object. This capability is not affected even in the case that the seed box contains also other instances of the same category as in rows 3 and 4.

3.4.1.1 Network architecture

We call the overall network architecture that implements the *Attend Refine Repeat* algorithm with its *In-Out* object location refinement module and its objectness scoring module, *AttractioNet*[4]. Given an image I, our *AttractioNet* model will be required to process multiple image boxes of various sizes, by two different modules and repeat those processing steps for several iterations of the *Attend Refine Repeat* algorithm. So, in order to have an efficient implementation we follow the SPP-Net [55] and Fast-RCNN [42] paradigm and share the operations of the first convolutional layers between all the boxes, as well as across the two modules and all the *Attend Refine Repeat* algorithm repetitions (see Figure 3-12). Specifically, our *AttractioNet* model first forwards the image I through a first sequence of convolutional layers (conv. layers of VGG16-Net [144]) in order to extract convolutional feature maps F_I from the entire image. Then, on each iteration t the box-wise part of the

[4] *AttractioNet : (Att)end (R)efine Repeat: (Act)ive Box Proposal Generation via (I)n-(O)ut Localization (Net)work*

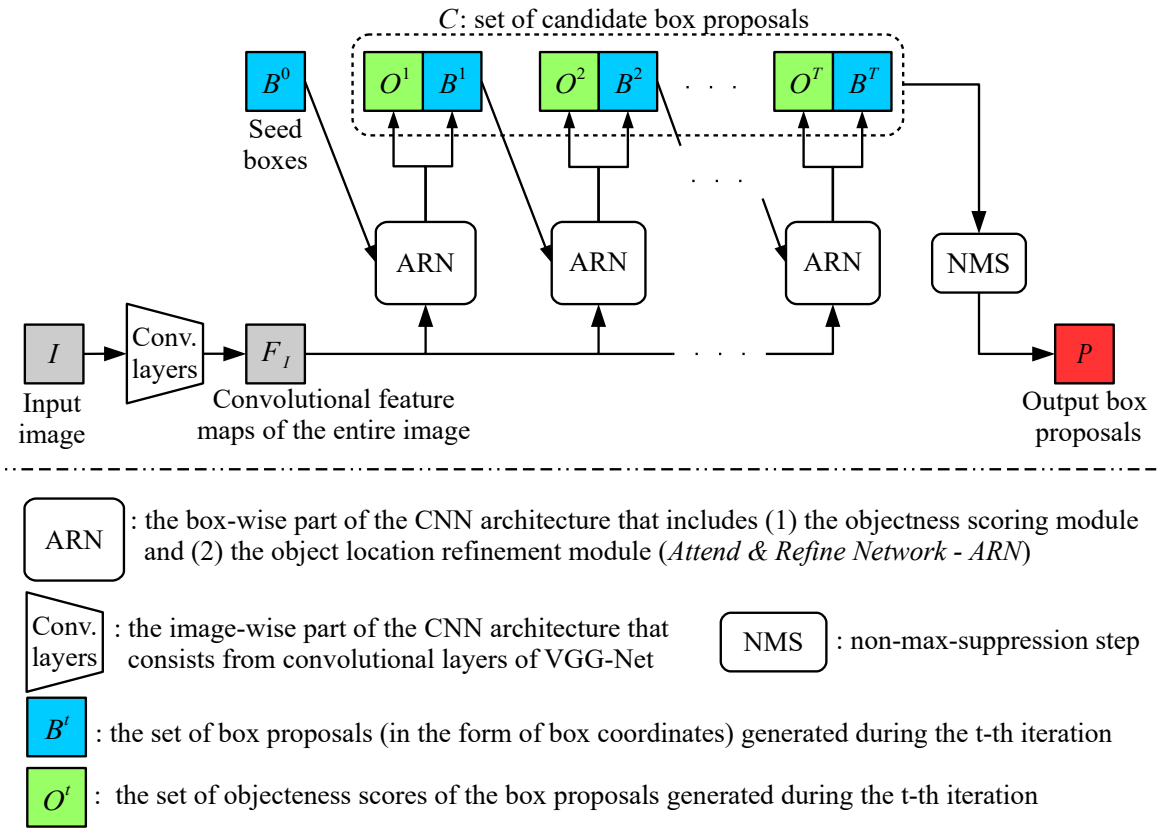


Figure 3-12: ***AttractioNet work-flow.*** The *Attend Refine Repeat* algorithm is implemented through a CNN model, called *AttractioNet*, whose run-time work-flow (when un-rolled over time) is illustrated here. On each iteration t the box-wise part of the architecture (*Attend & Refine Network: ARN*) gets as input the image convolutional feature maps F_I (extracted from the image-wise part of the CNN architecture) as well as a set of box locations $\mathbf{B}^{t-1}$ and yields the refined bounding box locations $\mathbf{B}^t$ and their objectness scores $\mathbf{O}^t$ using its *category agnostic object location refinement* module and its *category agnostic objectness scoring* module respectively. To avoid any confusion, note that our *AttractioNet* model does not include any recurrent connections.

architecture, which we call *Attend & Refine Network*, gets as input the image convolutional feature maps F_I and a set of box locations $\mathbf{B}^{t-1}$ and yields the refined bounding box locations $\mathbf{B}^t$ and their objectness scores $\mathbf{O}^t$ using its object location refinement module sub-network and its objectness scoring module sub-network respectively. In Figure 3-13 we provide the work-flow of the *Attend & Refine Network* when processing a single input box B. The architecture of its two sub-networks is described in more detail in the rest of this section:

Object location refinement module sub-network. Differently from the localization model architecture presented in §3.3.3.3, the convolutional layers of this sub-network output

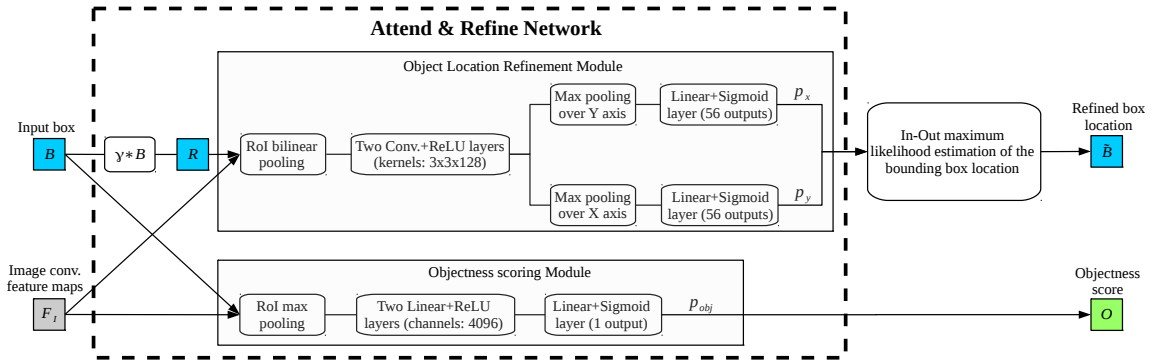


Figure 3-13: ***Attend & Refine Network* architecture.** The *Attend & Refine Network* is the box-wise part of the *AttractioNet* architecture. In this figure we depict the work-flow for a single input box B. Specifically, given an input box B and the image convolutional feature maps F_I, the *Attend & Refine Network* yields **(1)** the in-out location probability vectors, p_x and p_y, (using its object location refinement sub-network) and **(2)** the objectness scalar probability p_{obj} (using its objectness scoring sub-network). Given the *in-out* probabilities, p_x and p_y, the object location inference is formulated as a simple maximum likelihood estimation problem that results in the refined bounding box coordinates $\tilde{B}$.

128 feature channels instead of 512, which speeds up the processing by a factor of 4 without affecting the category-agnostic localization accuracy. Also, in order to yield a fixed size feature for the R region, instead of region adaptive max-pooling this sub-network uses region bilinear pooling [19, 17] that in our initial experiments gave slightly better results. Finally, our version is designed to yield two probability vectors of size M[5], instead of $C \times 2$ vectors of size M (where C is the number of categories), given that in our case we aim for category-agnostic object location refinement.

Objectness scoring module sub-network. Given the image feature maps F_I and the window B it first performs region adaptive max pooling of the features inside B that yields a fixed size feature ($7 \times 7 \times 512$). Then it forwards this feature through two linear+ReLU hidden layers of 4096 channels each (fc_6 and fc_7 layers of VGG16) and a final linear+sigmoid layer with one output that corresponds to the probability p_{obj} of the box B tightly enclosing an object. During training the hidden layers are followed by Dropout units with dropout probability $p = 0.5$.

[5]Here we use $M = 56$.

3.4.1.2 Training

Training loss: During training the following multi-task loss is optimized:

$$\underbrace{\frac{1}{N^L}\sum_{k=1}^{N^L} L_{loc}(\theta|B_k, T_k, I_k)}_{\text{localization task loss}} + \underbrace{\frac{1}{N^O}\sum_{k=1}^{N^O} L_{obj}(\theta|B_k, y_k, I_k)}_{\text{objectness scoring task loss}}, \tag{3.15}$$

where θ are the learnable network parameters, $\{B_k, T_k, I_k\}_{k=1}^{N^L}$ are N^L training triplets for learning the localization task and $\{B_k, y_k, I_k\}_{k=1}^{N^O}$ are N^O training triplets for learning the objectness scoring task. Each training triplet $\{B, T, I\}$ of the localization task includes the image I, the box B and the target localization probability vectors $T = \{T_x, T_y\}$. If (B_l^*, B_t^*) and (B_r^*, B_b^*) are the top-left and bottom-right coordinates of the target box B^* then the target probability vectors $T_x = \{T_{x,i}\}_{i=1}^M$ and $T_y = \{T_{y,i}\}_{i=1}^M$ are defined as:

$$T_{x,i} = \begin{cases} 1, & \text{if } B_l^* \leq i \leq B_r^* \\ 0, & \text{otherwise} \end{cases} \text{ and } T_{y,i} = \begin{cases} 1, & \text{if } B_t^* \leq i \leq B_b^* \\ 0, & \text{otherwise} \end{cases}, \forall i \in \{1, \ldots, M\} \tag{3.16}$$

The loss $L_{loc}(\theta|B, T, I)$ of this triplet is the sum of binary logistic regression losses:

$$\frac{1}{2M}\sum_{a\in\{x,y\}}\sum_{i=1}^{M} T_{a,i}\log(p_{a,i}) + (1 - T_{a,i})\log(1 - p_{a,i}), \tag{3.17}$$

where p_a are the output probability vectors of the localization module for the image I, the box B and the network parameters θ. The training triplet $\{B, y, I\}$ for the objectness scoring task includes the image I, the box B and the target value $y \in \{0, 1\}$ of whether the box B contains an object (positive triplet with $y = 1$) or not (negative triplet with $y = 0$). The loss $L_{obj}(\theta|B, y, I)$ of this triplet is the binary logistic regression loss $y\log(p_{obj}) + (1 - y)\log(1 - p_{obj})$, where p_{obj} is the objectness probability for the image I, the box B and the network parameters θ.

Creating training triplets: In order to create the localization and objectness training triplets of one image we first artificially create a pool of boxes that our algorithm is likely to

see during test time. Hence we start by generating seed boxes (as the test time algorithm) and for each of them we predict the bounding boxes of the ground truth objects that are closest to them using an ideal object location refinement module. This step is repeated one more time using the previous ideal predictions as input. Because of the finite search area of the search region R the predicted boxes will not necessarily coincide with the ground truth bounding boxes. Furthermore, to account for prediction errors during test time, we repeat the above process by jittering this time the output probability vectors of the ideal location refinement module with 20% noise. Finally, we merge all the generated boxes (starting from the seed ones) to a single pool. Given this pool, the positive training boxes in the objectness scoring task are those that their IoU with any ground truth object is at least 0.5 and the negative training boxes are those that their maximum IoU with any ground truth object is less than 0.4. For the localization task we use as training boxes those that their IoU with any ground truth object is at least 0.5.

Optimization: To minimize the objective we use stochastic gradient descent (SGD) optimization with an image-centric strategy for sampling training triplets. Specifically, in each mini-batch we first sample 4 images and then for each image we sample 64 training triplets for the objectness scoring task (50% are positive and 50% are negative) and 32 training triplets for the localization task. The momentum is set to 0.9 and the learning schedule includes training for $320k$ iterations with a learning rate of $l_r = 0.001$ and then for another $260k$ iterations with $l_r = 0.0001$. The training time is around 7 days (although we observed that we could have stopped training on the 5th day with insignificant loss in performance).

Scale and aspect ratio jittering: During test time our model is fed with a single image scaled such that its shortest dimension to be 1000 pixels or its longest dimension to not exceed the 1400 pixels. However, during training each image is randomly resized such that its shortest dimension to be one of the following number of pixels $\{300 : 50 : 1000\}$ (using Matlab notation) taking care, however, the longest dimension to not exceed 1000 pixels. Also, with probability 0.5 we jitter the aspect ratio of the image by altering the image

dimensions from $W \times H$ to $(\alpha W) \times H$ or $W \times (\alpha H)$ where the value of α is uniformly sampled from $2^{\cdot 2:.2:1.0}$ (Matlab notation). We observed that this type of data augmentation gives a slight improvement on the results.

3.4.2 Experimental results

Here we perform an exhaustive evaluation of our box proposal generation approach, which we call *AttractioNet*, under various test scenarios. Specifically, we first evaluate our approach with respect to its object localization performance by comparing it with other competing methods and we also provide an ablation study of its main novel components in §3.4.2.1. Then, we study its ability to generalize to unseen categories in §3.4.2.2, we evaluate it in the context of the object detection task in §3.4.2.3 and finally, we provide qualitative results in §3.4.2.4.

Training set: In order to train our *AttractioNet* model we use the training set of MS COCO [91] detection benchmark dataset that includes $80k$ images and it is labelled with 80 different object categories. Note that the MSCOCO dataset is an ideal candidate for training our box proposal model since: **(1)** it is labelled with a decent number of different object categories and **(2)** it includes images captured from complex real-life scenes containing common objects in their natural context. The aforementioned training set properties are desirable for achieving good performance on difficult test images (a.k.a. images in the wild) and generalizing to unseen during training object categories.

Implementation details: In the active box proposal algorithm we use $10k$ seed boxes generated with a technique similar to Cracking Bing [173][6]. To reduce the computational cost of our algorithm, after the first repetition we only keep the top $2k$ scored boxes and we continue with this number of candidate box proposals for four more extra iterations. In the non-maximum-suppression [35] (NMS) step the optimal IoU threshold (in terms of the achieved AR) depends on the desired number of box-proposals. For example, for

[6]We use seed boxes of 3 aspect ratios, $1:2$, $2:1$ and $1:1$, and 9 different sizes of the smallest seed box dimension $\{16, 32, 50, 72, 96, 128, 192, 256, 384\}$.

10, 100, 1000 and 2000 proposals the optimal IoU thresholds are 0.55, 0.75, 0.90 and 0.95 respectively (note that the aforementioned IoU thresholds were cross validated on a set different from the one used for evaluation). For practical purposes and in order to have a unified NMS process, we first apply NMS with the IoU threshold equal to 0.95 and get the top 2000 box proposals, and then follow a multi-threshold NMS strategy that re-orders this set of 2000 boxes such that for any given number K, the top K box proposals in the set better cover (in terms of achieved AR) the objects in the image (see appendix A.2).

3.4.2.1 Object box proposal generation evaluation

Here we evaluate our *AtractioNet* method in the end task of box proposal generation. For that purpose, we test it on the first $5k$ images of the COCO validation set and the PASCAL [31] VOC2007 test set (that also includes around $5k$ images).

Evaluation Metrics: As evaluation metric we use the average recall (AR) which, for a fixed number of box proposals, averages the recall of the localized ground truth objects for several Intersection over Union (IoU) thresholds in the range .5:.05:.95 (Matlab notation). The average recall metric has been proposed from Hosang et al. [64, 63] where in their work they demonstrated that it correlates well with the average precision performance of box proposal based object detection systems. In our case, in order to evaluate our method we report the AR results for 10, 100 and 1000 box proposals using the notation *AR@10*, *AR@100* and *AR@1000* respectively. Also, in the case of 100 box proposals we also report the AR of the small ($\alpha < 32^2$), medium ($32^2 \leq \alpha \leq 96^2$) and large ($\alpha > 96^2$) sized objects using the notation *AR@100-Small*, *AR@100-Medium* and *AR@100-Large* respectively, where α is the area of the object. For extracting those measurements we use the COCO API (`https://github.com/pdollar/coco`).

Average recall evaluation. In Table 3.6 we report the average recall (AR) metrics of our method as well as of other competing methods in the COCO validation set. We observe that the average recall performance achieved by our method exceeds all the previous work

Method	AR@10	AR@100	AR@1000	AR@100-Small	AR@100-Medium	AR@100-Large
EdgeBoxes [177]	0.074	0.178	0.338	0.015	0.134	0.502
Geodesic [75]	0.040	0.180	0.359	-	-	-
Selective Search [153]	0.052	0.163	0.357	0.012	0.0132	0.466
MCG [4]	0.101	0.246	0.398	0.008	0.119	0.530
DeepMask [120]	0.153	0.313	0.446	-	-	-
DeepMaskZoom [120]	0.150	0.326	0.482	-	-	-
Co-Obj [53]	0.189	0.366	0.492	0.107	0.449	0.686
SharpMask [121]	0.192	0.362	0.483	0.060	0.510	0.665
SharpMaskZoom [121]	0.192	0.390	0.532	0.149	0.507	0.630
SharpMaskZoom^2 [121]	0.178	0.391	0.555	0.221	0.454	0.588
AttractioNet (Ours)	**0.328**	**0.533**	**0.662**	**0.315**	**0.622**	**0.777**

Table 3.6: Average Recall results on the first $5k$ images of COCO validation set.

Method	AR@10	AR@100	AR@1000	AR@100-Small	AR@100-Medium	AR@100-Large
EdgeBoxes [177]	0.203	0.407	0.601	0.035	0.159	0.559
Geodesic [75]	0.121	0.364	0.596	-	-	-
Selective Search [153]	0.085	0.347	0.618	0.017	0.134	0.364
MCG [4]	0.232	0.462	0.634	0.073	0.228	0.618
DeepMask [120]	0.337	0.561	0.690	-	-	-
Best of Co-Obj [53]	0.430	0.602	0.745	0.453	0.517	0.654
AttractioNet (Ours)	**0.554**	**0.744**	**0.859**	**0.562**	**0.670**	**0.794**

Table 3.7: Average Recall results on the PASCAL VOC2007 test set.

in all the AR metrics by a significant margin (around 10 absolute points in the percentage scale). Similar gains are also observed in Table 3.7 where we report the average recall results of our methods in the PASCAL VOC2007 test set. Furthermore, in Figure 3-14 we provide for our method the recall as a function of the IoU overlap of the localized ground truth objects. We see that the recall decreases relatively slowly as we increase the IoU from 0.5 to 0.75 while for IoU above 0.85 the decrease is faster. In Figure 3-15 we compare the box proposals generated from our *AttractioNet* model (*Ours* entry) against those generated from the previous state-of-the-art [121] (entries *SharpMask*, *SharpMaskZoom* and *SharpMaskZoom*^{2}) w.r.t. the recall versus IoU trade-off and average recall versus proposals number trade-off that they achieve. Also, in Table 3.6 we report the AR results both for our method and for the SharpMask entries. We observe that the model proposed in our work has clearly superior performance over the SharpMask entries under all test cases.

Ablation study. We perform an ablation study of the two key ideas for improving the state-of-the-art on the bounding box proposal generation task, the location refinement module and the active box generation strategy. In order to assess the importance of our

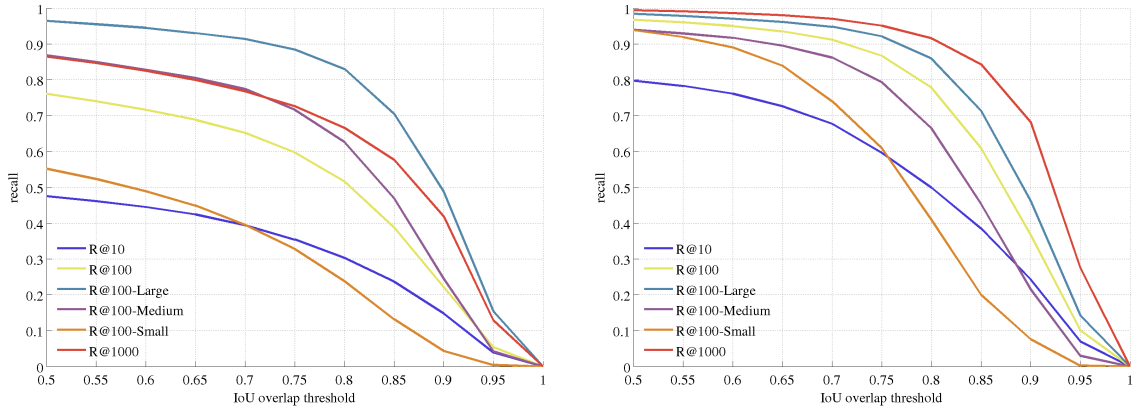


Figure 3-14: Recall versus IoU overlap plots of our *AttractioNet* approach under different test cases: 10 proposals (*R@10*), 100 proposals (*R@100*), 1000 proposals (*R@1000*), 100 proposals and small sized objects (*R@100-Small*), 100 proposals and medium sized objects (*R@100-Medium*) and 100 proposals and large sized objects (*R@100-Large*). **(Left)** Results in the first $5k$ images of COCO validation set. **(Right)** Results in the PASCAL VOC2007 test set.

Box refinement	Active box generation	# attended boxes	AR@10	AR@100	AR@1000	AR@100-Small	AR@100-Medium	AR@100-Large
		$18k$	0.147	0.260	0.326	0.122	0.317	0.412
✓		$18k$	0.298	0.491	0.622	0.281	0.583	0.717
✓	✓	$18k$	**0.328**	**0.533**	**0.662**	**0.315**	**0.622**	**0.777**

Table 3.8: **Ablation study of our *AttractioNet* box proposal system.** In the first row we simply apply the objectness scoring module on a set of $18k$ seed boxes. In the second row we apply on the same set of $18k$ seed boxes both the objectness scoring module and the box refinement module. In the last row we utilize our full active box generation strategy that in total attends $18k$ boxes of which $10k$ are seed boxes and the rest $8k$ boxes are actively generated. The reported results are from the first $5k$ images of COCO validation set.

object location refinement module we evaluated two test cases for generating box proposals: **(1)** simply applying the objectness scoring module on a set of $18k$ seed boxes (first row of Table 3.8) and **(2)** applying both the objectness scoring module and the object location refinement module on the same set of $18k$ seed boxes (second row of Table 3.8). Note that in none of them is the active box generation strategy being used. The average recall results of those two test cases are reported in the first two rows of Table 3.8. We observe that without the object location refinement module the average recall performance of the box proposal system is very poor. In contrast, the average recall performance of the test case that involves the object location refinement module but not the active box generation strategy is already better than the previous state-of-the-art as reported in Table 3.6, which demonstrates the very good localization accuracy of our category agnostic location refinement module. The active box generation strategy, which we call *Attend Refine Repeat* algorithm, attends in

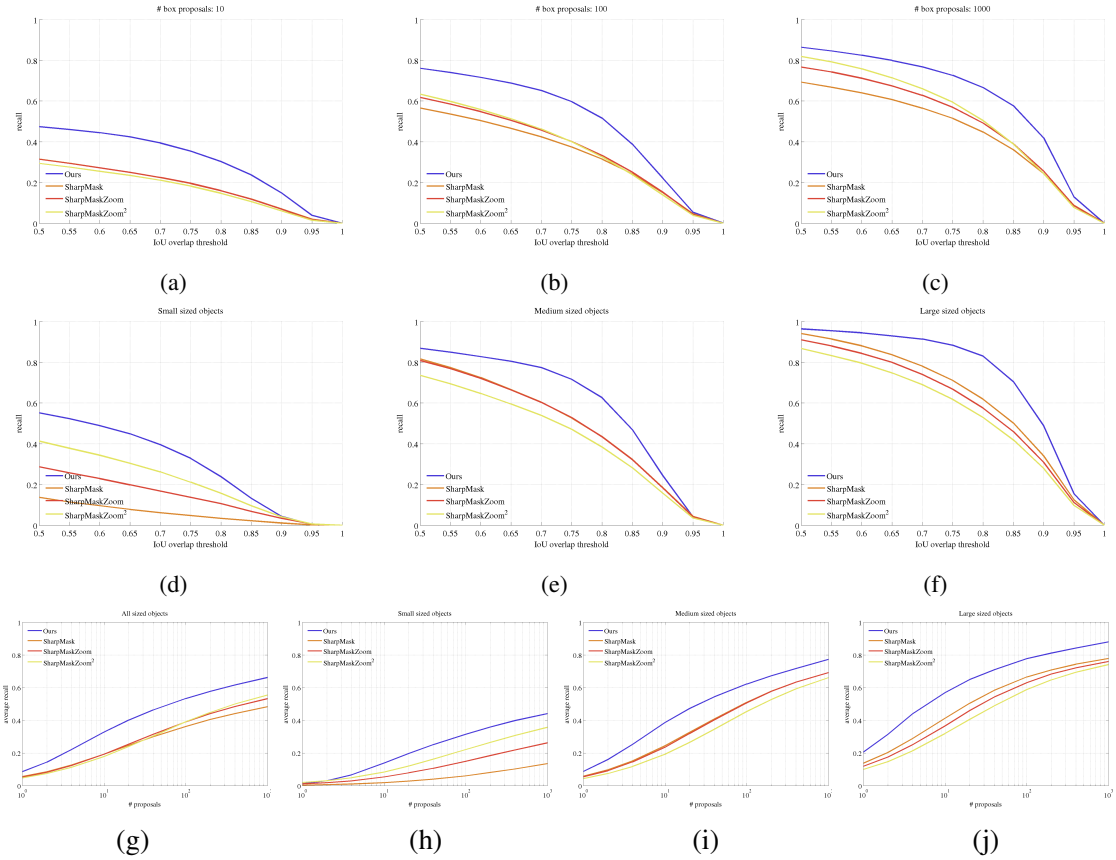


Figure 3-15: **Comparison with previous state-of-the-art.** Comparison of our *AttractioNet* box proposal model (*Ours* entry) against the previous state-of-the-art [121] (*SharpMask*, *SharpMaskZoom* and *SharpMaskZoom*^{2} entries) w.r.t. the recall versus IoU trade-off and average recall versus proposals number trade-off that they achieve under various test scenarios. Specifically, the sub-figures (a), (b) and (c) plot the recall as a function of the IoU threshold for 10, 100 and 1000 box proposals respectively and the sub-figures (d), (e) and (f) plot the recall as a function of the IoU threshold for 100 box proposals and with respect to the small, medium and large sized objects correspondingly. Also, the sub-figures (g), (h), (i) and (j) plot the average recall as a function of the proposals number for all the objects regardless of their size as well as for the small, medium and large sized objects respectively. The reported results are from the first $5k$ images of the COCO validation set.

total $18k$ boxes before it outputs the final list of box proposals. Specifically, it attends $10k$ seed boxes in the first repetition of the algorithm and $2k$ actively generated boxes in each of the following four repetitions. A crucial question is whether actively generating those extra $8k$ boxes is really essential in the task or we could achieve the same average recall performance by directly attending $18k$ seed boxes and without continuing on the active box generation stage. We evaluated such test case and we report the average recall results in

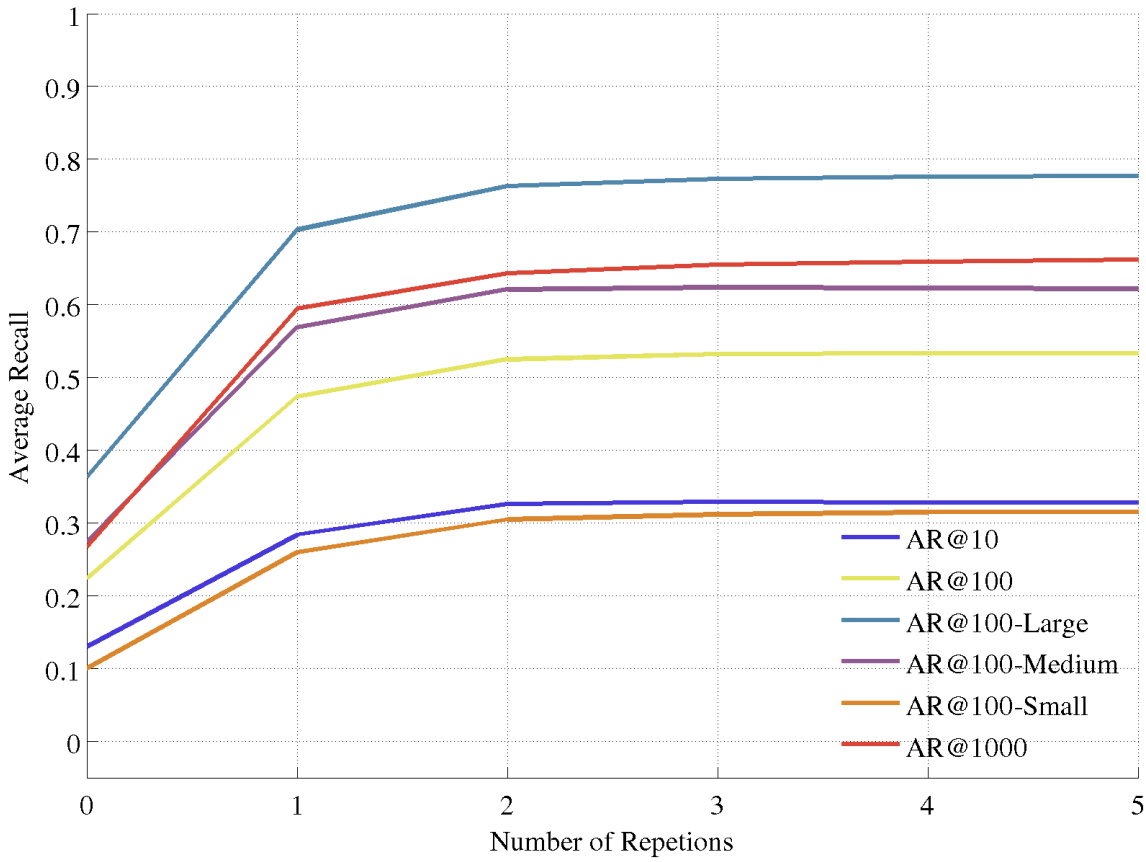


Figure 3-16: Average recall versus the repetitions number of the active box proposal generation algorithm in the COCO validation set. Note that 0 repetitions is the scenario of simply applying the objectness module on the seed boxes.

Table 3.8 (see rows 2 and 3). We observe that employing the active box generation strategy (3rd row in Table 3.8) offers a significant boost in the average recall performance (between 3 and 6 absolute points in the percentage scale) thus proving its importance on yielding well localized bounding box proposals. Also, in the right side of Figure 3-16 we plot the average recall metrics as a function of the repetitions number of our active box generation strategy. We observe that the average recall measurements are increased as we increase the repetitions number and that the increase is steeper on the first repetitions of the algorithm while it starts to converge after the 4th repetition.

Run time: In the current work we did not focus on providing an optimized implementation of our approach. There is room for significantly improving computational efficiency. For instance, just by using SVD decomposition on the fully connected layers of the objectness

Method	Run time	AR@10	AR@100	AR@1000	AR@100-Small	AR@100-Medium	AR@100-Large
COCO validation set							
AttractioNet (Ours)	4.00 sec	**0.328**	**0.533**	**0.662**	0.315	**0.622**	**0.777**
AttractioNet (Ours, fast version)	1.63 sec	0.326	0.532	0.660	**0.317**	0.621	0.771
VOC2007 test set							
AttractioNet (Ours)	4.00 sec	**0.554**	**0.744**	**0.859**	0.562	**0.670**	**0.794**
AttractioNet (Ours, fast version)	1.63 sec	0.547	0.740	0.848	**0.575**	0.666	0.788

Table 3.9: **Run time of our approach on a GTX Titan X GPU.** The reported results are from the first $5k$ images of COCO validation set and the PASCAL VOC2007 test set.

module at post-training time (similar to Fast-RCNN [42]) and early stopping a sequence of bounding box location refinements in the case it has already converged[7], the runtime drops from 4.0 seconds to 1.63 seconds with losing almost no accuracy (see Table 3.9). There are also several other possibilities that we have not yet explored such as tuning the number of feature channels and/or network layers of the CNN architecture (similar to the DeepBox [79] and the SharpMask [121] approaches). In the remainder of this section we will use the fast version of our *AttractioNet* approach in order to provide experimental results.

3.4.2.2 Generalization to unseen categories

So far we have evaluated our *AttractioNet* approach — in the end task of object box proposal generation — on the COCO validation set and the PASCAL VOC2007 test set that are labelled with the same or a subset of the object categories seen in the training set. In order to assess the *AttractioNet*'s capability to generalize to unseen categories, as it is suggested by Chavali et al. [10], we evaluate our *AttractioNet* model on two extra datasets that are labelled with object categories that are not present in its training set (unseen object categories).

From COCO to ImageNet [129]. Here we evaluate our COCO trained *AttractioNet* box proposal model on the ImageNet [129] ILSVRC2013 detection task validation set that is labelled with 200 different object categories and we report average recall results in Table 3.10. Note that among the 200 categories of ImageNet detection task, 60 of them, as we identified, are also present in the *AttractioNet*'s training set (see Appendix A.3). Thus, for a better insight on the generalization capabilities of *AttractioNet*, we divided the ImageNet detection task categories on two groups, the categories seen by *AttractioNet* and

[7] A sequence of bounding box refinements is considered that it has converged when the IoU between the two lastly predicted boxes in the sequence is greater than 0.9.

Method	All categories			Seen categories			Unseen categories		
	AR@10	AR@100	AR@1000	AR@10	AR@100	AR@1000	AR@10	AR@100	AR@1000
AttractioNet (Ours)	**0.412**	**0.618**	**0.748**	**0.474**	**0.671**	**0.789**	**0.299**	**0.521**	**0.673**
EdgeBoxes [177]	0.182	0.377	0.550	0.194	0.396	0.566	0.160	0.344	0.519
Selective Search [153]	0.132	0.358	0.562	0.143	0.372	0.568	0.111	0.332	0.551
MCG [4]	0.219	0.428	0.603	0.228	0.447	0.623	0.205	0.395	0.568

Table 3.10: **Generalization to unseen categories: from COCO to ImageNet.** In this table we report average recall results on the ImageNet [129] ILSVRC2013 detection task validation set that includes around $20k$ images and it is labelled with 200 object categories. ***Seen categories*** are the set of object categories that our COCO trained *AttractioNet* model "saw" during training. In contrast, ***unseen categories*** is the set of object categories that were not present in the training set of our *AttractioNet* model.

Method	AR@10	AR@100	AR@1000	AR@100-Small	AR@100-Medium	AR@100-Large
AttractioNet (Ours)	**0.159**	**0.389**	**0.579**	**0.205**	**0.419**	**0.498**
EdgeBoxes [177]	0.049	0.160	0.362	0.020	0.131	0.332
Selective Search [153]	0.024	0.143	0.422	0.008	0.085	0.362
MCG [4]	0.078	0.237	0.441	0.045	0.195	0.476

Table 3.11: **Generalization to unseen categories: from COCO to NYU-Depth V2 dataset.** In this table we report average recall results on the 1449 labelled images of the NYU-Depth V2 dataset [143]. Note that the NYU-Depth V2 dataset is densely labelled with more than 800 different categories.

the unseen categories, and we report the average recall results separately for those two groups of object categories in Table 3.10. For comparison purposes we also report the average recall performance of a few indicative other box proposal methods whose code is publicly available. We observe that, despite the performance difference of our approach between the seen and the unseen object categories (which is to be expected), its average recall performance on the unseen categories is still quite high and significantly better than the other box proposal methods. Note that even the non-learning based approaches of Selective Search and EdgeBoxes exhibit a performance drop on the unseen by *AttractioNet* group of object categories, which we assume is because this group contains more intrinsically difficult to discover objects.

From COCO to NYU Depth dataset [143]. The NYU Depth V2 dataset [143] provides 1449 images (recorded from indoor scenes) that are densely pixel-wise annotated with 864 different categories. We used the available instance-wise segmentations to create ground truth bounding boxes and we tested our COCO trained *AttractioNet* model on them (see Table 3.11). Note that among the 864 available pixel categories, a few of them are "stuff"

categories (e.g., wall, floor, ceiling or stairs) or in general non-object pixel categories that our object box proposal method should by definition not recall. Thus, during the process of creating the ground truth bounding boxes, those non-object pixel segmentation annotations were excluded (see Appendix A.4). In Table 3.11 we report the average recall results of our *AttractioNet* method as well as of a few other indicative methods whose code is publicly available. We again observe that our method surpasses all other approaches by a significant margin. Furthermore, in this case the superiority of our approach is more evident on the average recall of the small and medium sized objects.

To conclude we argue that our learning based *AttractioNet* approach exhibits good generalization behaviour. Specifically, its average recall performance on the unseen object categories remains very high and is also much better than other competing approaches, including both learning-based approaches such as the MSG and hand-engineered ones such as the Selective Search or the EdgeBoxes methods. A performance drop is still observed while going from seen to unseen categories, but this is something to be expected given that any machine learning algorithm will always exhibit a certain performance drop while going from seen to unseen data (i.e., training set accuracy versus test set accuracy).

3.4.2.3 Evaluation in the context of the object detection task

Here we evaluate our *AttractioNet* box proposals in the context of the object detection task by training and testing a box proposal based object detection system on them (specifically we use the fast version of *AttractioNet*).

Detection system. Our box proposal based object detection network consists of a Fast-RCNN [42] category-specific recognition module and a LocNet Combined ML category-specific bounding box refinement module that share the same image-wise convolutional layers (conv1_1 till conv5_3 layers of VGG16-Net). The detection network is trained on the union of the COCO train set that includes around $80k$ images and on a subset of the COCO validation set that includes around $35k$ images (the remaining $5k$ images of COCO validation set are being used for evaluation). For training we use our *AttractioNet* box

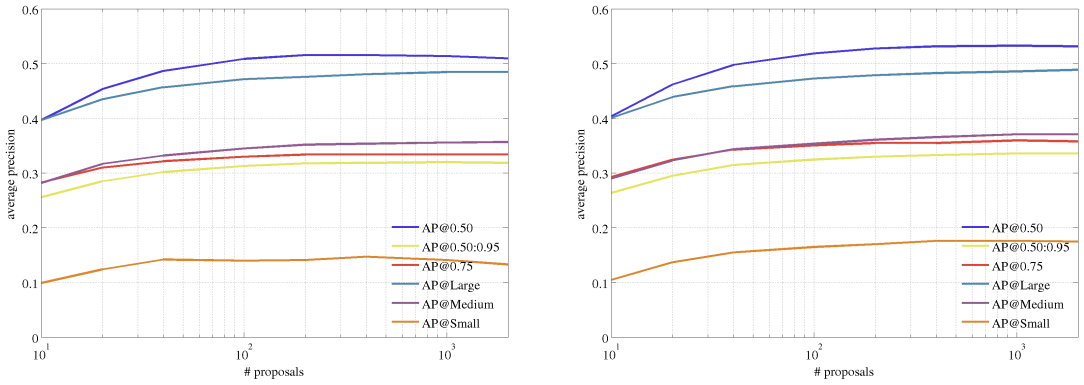
proposals and we define as positives those that have IoU overlap with any ground truth bounding box at least 0.5 and as negatives the remaining proposals. For training we use SGD where each mini-batch consists of 4 images with 64 box proposals each (256 boxes per mini-batch in total) and the ratio of negative-positive boxes is 3:1. We train the detection network for $500k$ SGD iterations starting with a learning rate of 0.001 and dropping it to 0.0001 after $320k$ iterations. We use the same scale and aspect ratio jittering technique that is used on *AttractioNet* and is described in section 3.4.1.2. During test time, as post-processing we use a non-max-suppression step (with IoU threshold of 0.35) that is enhanced with the box voting technique described in §2.5 with IoU threshold of 0.75. Note that we did not include iterative object localization since the bounding box proposals are already very well localized and we did not get any significant improvement from running the detection system for extra iterations. Using the same trained model we provide results for two test cases: **(1)** using a single scale of 600 pixels during test time and **(2)** using two scales of 500 and 1000 pixels during test time.

Detection evaluation setting. The detection evaluation metrics that we use are the average precision (AP) for the IoU thresholds of 0.50 (AP@0.50), 0.75 (AP@0.75) and the COCO style of average precision (AP@$0.50:0.95$) that averages the traditional AP over several IoU thresholds between 0.50 and 0.95. Also, we report the COCO style of average precision with respect to the small (AP@Small), medium (AP@Medium) and large (AP@Large) sized objects. We perform the evaluation on $5k$ images of COCO 2014 validation set and we provide final results on the COCO 2015 test-dev set.

Detection results. In Figure 3-17 we provide plots of the achieved average precision (AP) as a function of the used box proposals number and in Table 3.12 we provide the average precision results for 10, 100, 1000 and 2000 box proposals. We observe that in all cases, the average precision performance of the detection system seems to converge after the 200 box proposals. Furthermore, for single scale test case our best COCO-style average precision is 0.320 and for the two scales test case our best COCO-style average precision is 0.337. By including horizontal image flipping augmentation during test time

our COCO-style average precision performance is increased to 0.343. Finally, in Table 3.13 we provide the average precision performance in the COCO test-dev 2015 set where we achieve a COCO-style AP of 0.341. By comparing with the average precision performance of the other competing methods, we observe that:

- Comparing with the other VGG16-Net based object detection systems (ION [5] and MultiPath [164] systems), our detection system achieves the highest COCO-style average precision with its main novelties w.r.t. the Fast R-CNN [42] baseline being **(1)** the use of the *AttractioNet* box proposals that are introduced in this chapter and **(2)** the LocNet category specific object location refinement technique that replaces the bounding box regression step.
- Comparing with the ION [5] detection system, which is also VGG16-Net based, our approach is better on the COCO-style AP metric (that favours good object localization) while theirs is better on the typical AP@0.50 metric. We hypothesize that this is due to the fact that our approach targets to mainly improve the localization aspect of object detection by improving the box proposal generation step while theirs targets to improve the recognition aspect of object detection. The above observation suggests that many of the novelties introduced on the ION [5] and MultiPath [164] systems w.r.t. object detection could be orthogonal to our box proposal generation work.
- The achieved average precision performance of our VGG16-Net based detection system is close to the state-of-the-art *ResNet-101 based* Faster R-CNN+++ detection system [57] that exploits the (more) recent successes in deep representation learning introduced — under the name Deep Residual Networks — in the same work by He et al. [57]. Presumably, our overall detection system could also benefit from being based on the Deep Residual Networks [57] or the more recent wider variant called Wide Residual Networks [163].
- Finally, our detection system has the highest average precision performance w.r.t. the small sized objects, which is a challenging problem, surpassing by a healthy margin even the ResNet-101 based Faster R-CNN+++ detection system [57]. This is thanks to the high average recall performance of our box proposal method on the small sized objects.



(a) **(b)**

Figure 3-17: **Detection results: Average precision versus *AttractioNet* box proposals number.** **(a)** During test time a single scale of 600 pixels is being used. **(b)** During test time two scales of 500 and 1000 pixels are being used. The reported results are from $5k$ images of COCO validation set.

Test scale(s)	# proposals	AP@0.50	AP@0.75	AP@0.50:0.95	AP@Small	AP@Medium	AP@Large
$600px$	10	0.397	0.283	0.256	0.099	0.282	0.397
$600px$	100	0.509	0.330	0.313	0.140	0.345	0.472
$600px$	1000	0.514	0.334	0.320	0.141	0.356	0.485
$600px$	2000	0.510	0.334	0.319	0.133	0.357	0.485
$500px, 1000px$	10	0.404	0.293	0.264	0.105	0.290	0.401
$500px, 1000px$	100	0.519	0.351	0.325	0.165	0.354	0.473
$500px, 1000px$	1000	0.533	0.360	0.336	0.176	0.371	0.486
$500px, 1000px$	2000	0.532	0.358	0.336	0.175	0.371	0.489
$500px, 1000px$ ★	2000	**0.540**	**0.364**	**0.343**	**0.184**	**0.382**	**0.491**

Table 3.12: **Detection results: Average precision performance using *AttractioNet* box proposals.** The reported results are from $5k$ images of COCO validation set. The last entry with the ★ symbol uses horizontal image flipping augmentation during test time.

3.4.2.4 Qualitative results

In Figure 3-18 we provide qualitative results of our *AttractioNet* box proposal approach on images coming from the COCO validation set. Note that our approach manages to recall most of the objects in an image, even in the case that the depicted scene is crowded with multiple objects that heavily overlap with each other.

3.5 Conclusions

We proposed a novel object localization methodology that is based on assigning probabilities related to the localization task on each row and column of the region in which it searches

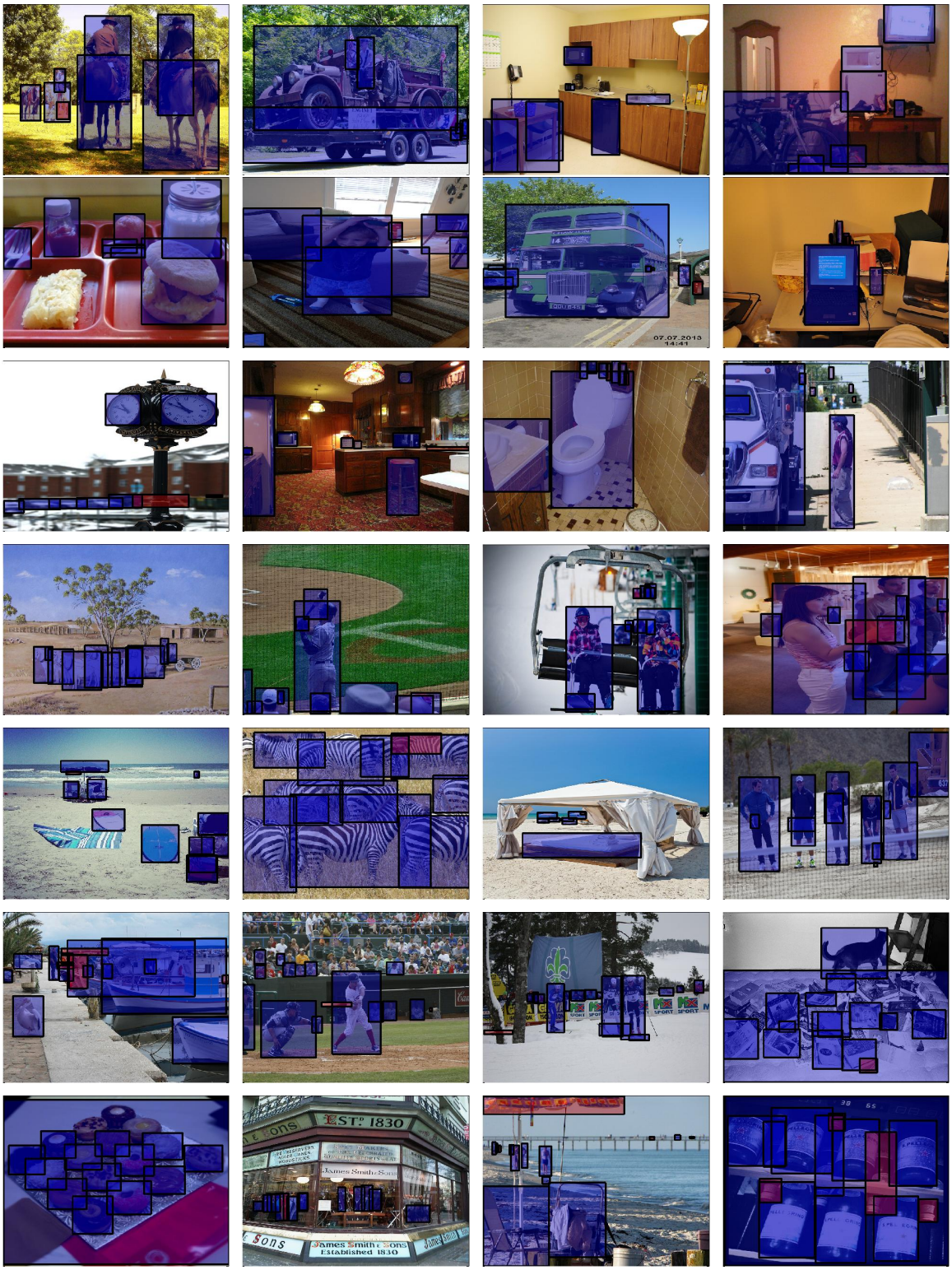


Figure 3-18: **Qualitative results in COCO.** The blue rectangles are the box proposals generated by our approach that best localize (in terms of IoU) the ground truth boxes. The red rectangles are the ground truth bounding boxes that were not discovered by our box proposal approach (their IoU with any box proposal is less than 0.5). Note that not all the object instances on the images are annotated.

Method	Base CNN	AP@0.50	AP@0.75	AP@0.50:0.95	AP@Small	AP@Medium	AP@Large
AttractioNet based detection system (Ours)	VGG16-Net [144]	0.537	0.363	0.341	0.175	0.365	0.469
ION [5]	VGG16-Net [144]	0.557	0.346	0.331	0.145	0.352	0.472
MultiPath [164]	VGG16-Net [144]	-	-	0.315	-	-	-
Faster R-CNN+++ [57]	ResNet-101 [57]	0.557	-	0.349	0.156	0.387	0.509

Table 3.13: **Detection results in COCO test-dev 2015 set.** In this table we report the average precision performance of our *AttractioNet* box proposals based detection system that uses 2000 proposals and two test scales of 500 and 1000 pixels. Note that: (1) all methods in this table (including ours) use horizontal image flipping augmentation during test time, (2) the ION [5] and MultiPath [164] detection systems use a single test scale of 600 and 800 pixels respectively while the Faster R-CNN+++ entry uses the scales $\{200, 400, 600, 800, 1000\}$, (3) apart from the ResNet-101 based Faster R-CNN+++ [57] entry, all the other methods are based on the VGG16-Network [144], (4) the reported results of all the competing methods are from the single model versions of their systems (and not the model ensemble versions) and (5) the reported results of the MultPath system are coming from $5k$ images of the COCO validation set (however, we expect the AR metrics on the test-dev set to be roughly similar).

the object. Those probabilities provide useful information regarding the location of the object inside the search region and they can be exploited in order to infer its boundaries with high accuracy. We implemented our model via using a convolutional neural network architecture properly adapted for this task, called LocNet, and we extensively evaluated it on PASCAL VOC2007 test set. We demonstrate that it outperforms CNN-based bounding box regression on all the evaluation metrics and it leads to a significant improvement on those metrics that reward good localization. Importantly, LocNet can be easily plugged into existing state-of-the-art object detection methods, in which case we show that it contributes to significantly boosting their performance. Also, we demonstrate that our object detection methodology can achieve very high mAP results even when the initial set of candidate boxes is generated by a simple sliding windows scheme.

Furthermore, we adapted the object localization methodology (devised for the detection task) to the box proposal generation task and built a novel box proposal generation system called *AttractioNet*. We extensively evaluate our system on several image datasets (i.e., COCO, PASCAL, ImageNet detection and NYU-Depth V2 datasets) demonstrating in all cases average recall results that surpass the previous state-of-the-art by a significant margin while also providing strong empirical evidence about the generalization ability of our approach w.r.t. unseen categories. Even more, we show the significance of our *AttractioNet* approach in the object detection task by coupling it with a VGG16-Net based detector

and thus managing to surpass the detection performance of all other VGG16-Net based detectors while even being on par with a heavily tuned ResNet-101 based detector. We note that, apart from object detection, there exist several other vision tasks, such as exemplar 2D-3D detection [100], visual semantic role labelling [49], caption generation [70] or visual question answering [138], for which a box proposal generation step can be employed. We are thus confident that our *AttractioNet* approach could have a significant value with respect to many other important applications as well.

Chapter 4

Deep structured prediction for pixel-wise image labeling

4.1 Introduction

While the previous two chapters focused on the object detection problem, in this one we deal with the pixel-wise image labeling problem (also called dense image labeling). Dense image labeling is a problem of paramount importance in the computer vision community as it encompasses many low or high level vision tasks including stereo matching [165], optical flow [62], surface normals estimation [29], and semantic segmentation [92], to mention a few characteristic examples. As already explained, the goal is to assign a discrete or continuous value for each pixel in the image. Due to its importance, there is a vast amount of work on this problem. Recent methods can be roughly divided into three main classes of approaches.

The first class focuses on developing independent patch classifiers/regressors [141, 139, 140, 34, 92, 37, 101, 110] that would directly predict the pixel label given as input an image patch centered on it or, in cases like stereo matching and optical flow, would be used for comparing patches between different images in order to pick pairs of best matching pixels [96, 162, 165, 166]. Deep convolutional neural networks (DCNNs) [84] have demonstrated excellent performance in the aforementioned tasks thanks to their ability to learn complex image representations by harnessing vast amount of training data [78, 144,

57]. However, despite their great representational power, just applying DCNNs on image patches, does not capture the structure of output labels, which is an important aspect of dense image labeling tasks. For instance, independent feed-forward DCNN patch predictors do not take into consideration the correlations that exist between nearby pixel labels. In addition, feed-forward DCNNs have the extra disadvantages that they usually involve multiple consecutive down-sampling operations (i.e., max-pooling or strided convolutions) and that the top most convolutional layers do not capture factors such as image edges or other fine image structures. Both of the above properties may prevent such methods from achieving precise and accurate results in dense image labeling tasks.

Another class of methods tries to model the joint dependencies of both the input and output variables by use of probabilistic graphical models such as Conditional Random Fields (CRFs) [80]. In CRFs, the dense image labeling task is performed through maximum a posteriori (MAP) inference in a graphical model that incorporates prior knowledge about the nature of the task in hand with pairwise edge potential between the graph nodes of the label variables. For example, in the case of semantic segmentation, those pairwise potentials enforce label consistency among similar or spatially adjacent pixels. Thanks to their ability to jointly model the input-output variables, CRFs have been extensively used in pixel-wise image labeling tasks [73, 130]. Recently, a number of methods has attempted to combine them with the representational power of DCNNs by getting the former (CRFs) to refine and disambiguate the predictions of the later one [135, 11, 174, 12]. Particularly, in semantic segmentation, DeepLab [11] uses a fully connected CRF to post-process the pixel-wise predictions of a convolutional neural network while in CRF-RNN [174], they unify the training of both the DCNN and the CRF by formulating the approximate mean-field inference of fully connected CRFs as Recurrent Neural Networks (RNN). However, a major drawback of most CRF based approaches is that the pairwise potentials have to be carefully hand designed in order to incorporate simple human assumptions about the structure of the output labels Y and at the same time to allow for tractable inference.

A third class of methods relies on a more data-driven approach for learning the joint space of both the input and the output variables. More specifically, in this case a deep neural network gets as input an initial estimate of the output labels and (optionally) the

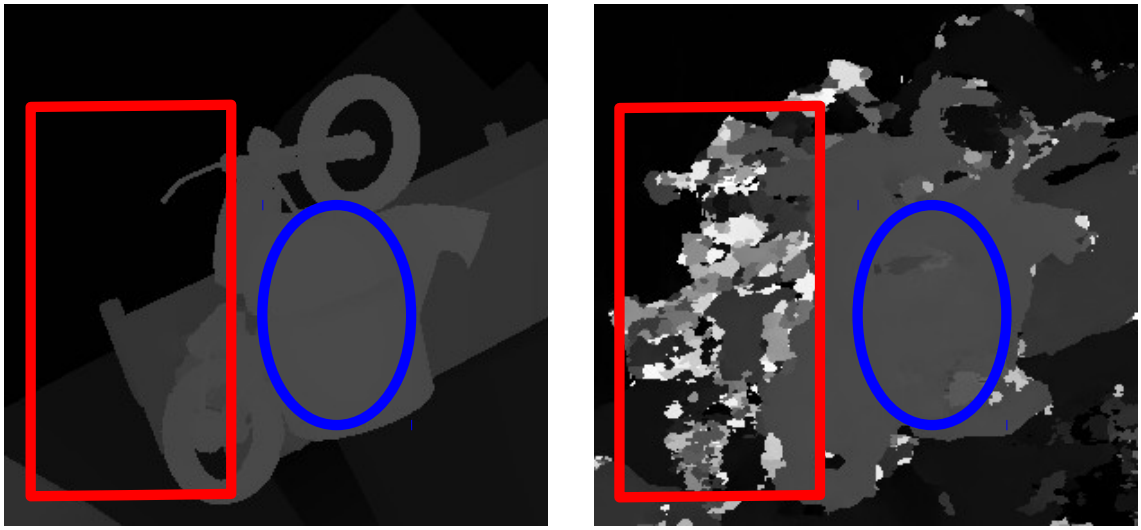


Figure 4-1: In this figure we visualize two different types of erroneously labeled image regions. On the left hand are the ground truth labels and on the right hand are some initial label estimates. With the red rectangle we indicate a dense concentration of "hard" mistakes in the initial labels that it is very difficult to be corrected by a residual refinement component. Instead, the most suitable action for such a region is to replace them by predicting entirely new labels for them. In contrast, the blue eclipse indicates an image region with "soft" label mistakes. Those image regions are easier to be handled by a residual refinement components.

input image and it is trained to predict a new refined estimate for the labels, thus being implicitly enforced to learn the joint space of both the input and the output variables. The network can learn either to predict new estimates for all pixel labels (transform-based approaches) [161, 52, 88], or alternatively, to predict residual corrections w.r.t. the initial label estimates (residual-based approaches) [9]. We will hereafter refer to these methods as *deep joint input-output models*. These are, loosely speaking, related to the CRF models in the sense that the deep neural network is enforced to learn the joint dependencies of both the input image and output labels, but with the advantage of being less constrained about the complexity of the input-output dependencies that it can capture.

Our work belongs to this last category of dense image labeling approaches, thus it is not constrained on the complexity of the input-output dependencies that it can capture. However, here we argue that prior approaches in this category use a sub-optimal strategy. For instance, the transform-based approaches (that always learn to predict new label estimates) often have to learn something more difficult than necessary since they must often simply learn to operate as identity transforms in case of correct initial labels, yielding the same label in their

output. On the other hand, for the residual based approaches it is easier to learn to predict zero residuals in the case of correct initial labels, but it is more difficult for them to refine "hard" mistakes that deviate a lot from the initial labels (see figure 4-1). Due to the above reasons, in our work we propose a deep joint input-output model that decomposes the label estimation/refinement process as a sequence of the following easier to execute operations: (1) *detection* of errors in the input labels, (2) *replacement* of the erroneous labels with new ones (i.e., directly predicting new pixel labels for the pixels detected as erroneously labeled), and finally (3) an overall *refinement* of all output labels by predicting residual corrections w.r.t. the labels generated by step (2). Each of the described operations in our framework is executed by a different component implemented with a deep neural network. Even more, those components are embedded in a unified architecture that is fully differentiable thus allowing for an end-to-end learning of the dense image labeling task by only applying the objective function on the final output. As a result of this, we are also able to explore a variety of novel deep network architectures by considering different ways of combining the above components, including the possibility of performing the above operations iteratively, as it is done in [88], thus enabling our model to correct even large, in area, regions of incorrect labels. It is also worth noting that the error detection component in the proposed architecture, by being forced to detect the erroneous pixel labels (given both the input and the initial estimates of the output labels), implicitly learns the joint structure of the input-output space, which is an important requirement for a successful application of any type of structured prediction model.

To summarize, the contribution of the work presented in this chapter are as follows:

- We propose a deep structured prediction framework for the dense image labeling task, which we call *Detect, Replace, Refine*, that relies on three main building blocks: (1) recognizing errors in the input label maps, (2) replacing the erroneous labels, and (3) performing a final refinement of the output label map. We show that all of the aforementioned steps can be embedded in a unified deep neural network architecture that is end-to-end trainable.
- In the context of the above framework, we also explore a variety of other network architectures for deep joint input-output models that result from utilizing different

combinations of the above building blocks.

- We implemented and evaluated our framework on the disparity prediction (stereo matching) and semantic segmentation tasks and we provide both qualitative and quantitative evidence about the advantages of the proposed approach.
- We show that our disparity estimation model that implements the proposed *Detect, Replace, Refine* architecture achieved state of the art results in the KITTI 2015 test set outperforming (at the time of completing this work) all prior published work by a significant margin.

The remainder of the chapter is structured as follows: We first describe our structured dense label prediction framework in §4.2 and its implementation w.r.t. the dense disparity estimation task (stereo matching) in §4.3. Then, we provide experimental results for the disparity estimation and semantic segmentation tasks in §4.4 and §4.5 respectively and we finally conclude the paper in §4.6.

4.2 Methodology

Let $X = \{x_i\}_{i=1}^{H \times W}$ be the input image[1] of size $H \times W$, where x_i are the image pixels, and $Y = \{y_i\}_{i=1}^{H \times W}$ be some initial label estimates for this image, where y_i is the label for the i-th pixel. Our dense image labeling methodology belongs on the broader category of approaches that consist of a deep joint input-output model model $F(.)$ that given as input the image X and the initial labels Y, learns to predict new, more accurate labels $Y' = F(X, Y)$. Note that in this setting the initial labels Y could come from another model $F_0(.)$ that depends only on the image X. Also, in the general case, the pixel labels Y can be of either discrete or continuous nature. In this work, however, we focus on the continuous case where greater variety of architectures can be explored. Note that in the discrete case (e.g., in the semantic segmentation task), in label map $Y = \{y_i\}_{i=1}^{H \times W}$ the label y_i of the i-th pixel, instead of being a continuous value as in the continues case, is defined as a probability vector with the probability distribution of the possible discrete values. For example, in the semantic

[1]Here, for simplicity, we consider images defined on a 2D domain, but our framework can be readily applied to images defined on any domain.

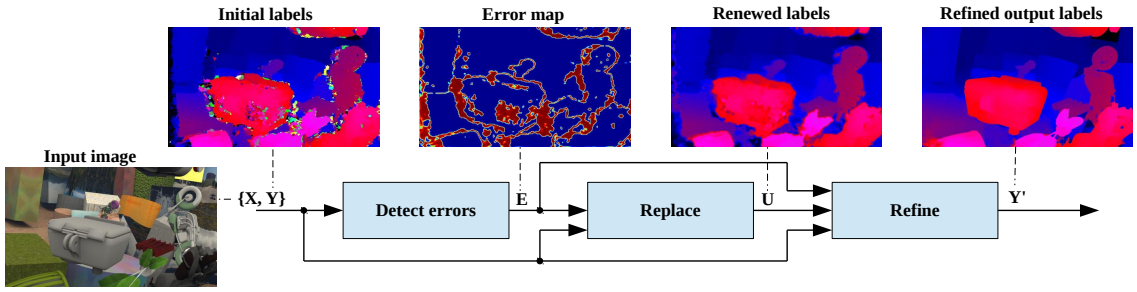


Figure 4-2: In this figure we demonstrate the generic architecture that we propose for the dense image labeling task. In this architecture the task of the deep joint input-output model is decomposed into three different sub-tasks that are: (1) detection of the erroneous initial labels, (2) replacement of the erroneous labels with new ones (leading to a renewed label map U), and then (3) refinement Y' of the renewed label map. The illustrated example is coming from the dense disparity labeling task (stereo matching).

segmentation task, y_i is the probability distribution over the available semantic categories for the i-th pixel.

The crucial question is what is the most effective way of implementing the deep joint input-output model $F(.)$. The two most common approaches in the literature involve a feed-forward deep convolutional neural network, $F_{DCNN}(.)$, that either directly predicts new labels $Y' = F_{DCNN}(X, Y)$ or it predicts the residual correction w.r.t. the input labels: $Y' = Y + F_{DCNN}(X, Y)$. We argue that both of them are sub-optimal solutions for implementing the $F(.)$ model. Instead, in our work we opt for a decomposition of the task of model $F(.)$ (i.e., predicting new, more accurate labels Y') in three different sub-tasks that are executed in sequence.

In the remainder of this section, we first describe the proposed architecture in §4.2.1, then we discuss the intuition behind it and its advantages in §4.2.2, and finally we describe other alternative architectures that we explored in §4.2.3.

4.2.1 Detect, Replace, Refine architecture

The generic dense image labeling architecture that we propose decomposes task of the deep joint input-output model in three sub-tasks each of them handled by a different learnable network component (see Figure 4-2). Those network components are: the error detection component $F_e(.)$, the label replacement component $F_u(.)$, and the label refinement

component $F_r(.)$. The sub-tasks that they perform, are:

Detect: The first sub-task in our generic pipeline is to detect the erroneously labeled pixels of Y by discovering which pixel labels are inconsistent with the remaining labels of Y and the input image X. This sub-task is performed by the error detection component $F_e(.)$ that basically needs to yield a probability map $E = F_e(X, Y)$ of the same size as the input labels Y that will have high probabilities for the "hard" mistakes in Y. These mistakes should ideally be forgotten and replaced with entirely new label values in the processing step that follows (see Figures 4-3a, 4-3b, and 4-3c). As we will see below, the topology of our generic architecture allows the error detection component $F_e(.)$ to learn its assigned task (i.e., detecting the incorrect pixel labels) without explicitly being trained for this, e.g., through the use of an auxiliary loss. The error detection function $F_e(.)$ can be implemented with any deep (or shallow) neural network with the only constraint being that its output map E must take values in the range $[0, 1]$.

Replace: In the second sub-task, a new label field U is produced by the convex combination of the initial label field Y and the output of the label replacement component $F_u(.)$: $U = E \odot F_u(X, Y, E) + (1 - E) \odot Y$ (see Figures 4-3e and 4-3f). We observe that the error probabilities generated by the error detection component $F_e(.)$ now act as gates that control which pixel labels of Y will be forgotten and replaced by the outputs of $F_u(.)$, which will be all pixel labels that are assigned high probability of being incorrect. In this context, the task of the Replace component $F_u(.)$ is to replace the erroneous pixel labels with new ones that will be in accordance both w.r.t. the input image X and w.r.t. the non-erroneous labels of Y. Note that for this task the Replace component $F_u(.)$ gets as input also the error probability map E. The reason for doing this is to help the Replace component to focus its attention only on those image regions that their labels need to be replaced. The component $F_u(.)$ can be implemented by any neural network whose output has the same size as the input labels Y.

Refine: The purpose of the erroneous label detection and label replacement steps so far was to perform a crude "fix" of the "hard" mistakes in the label map Y. In contrast,

the purpose of the current step is to do a final refinement of the entire output label map U, which is produced by the previous steps, in the form of residual corrections: $Y' = U + F_r(X, Y, E, U)$ (see Figures 4-3g and 4-3h). Intuitively, the purpose of this step is to correct the "soft" mistakes of the label map U and to better align the output labels Y' with the fine structures in the image X. The Refine component $F_r(.)$ can be implemented by any neural network whose output has the same size as the input labels U.

The above three steps can be applied for more than one iterations which, as we will see later, allows our generic framework to recover a good estimate of the ground truth labels or, in worst case, to yield more plausible results even when the initial labels Y are severely corrupted (see Figure 4-10 in the experiments section §4.4.3.5).

To summarize, the workings of our dense labeling generic architecture can be concisely described by the iterative application of the following three equations:

$$E = F_e(X, Y), \tag{4.1}$$

$$U = E \odot F_u(X, Y, E) + (1 - E) \odot Y, \tag{4.2}$$

$$Y' = U + F_r(X, Y, E, U). \tag{4.3}$$

We observe that the above generic architecture is fully differentiable as long as the function components $F_e(.)$, $F_u(.)$, and $F_r(.)$ are also differentiable. Due to this fact, the overall proposed architecture is end-to-end learnable by directly applying an objective function (e.g., Absolute Difference or Mean Square Error loss functions) on the final output label maps Y'.

4.2.2 Discussion

Role of the Detection component $F_e(.)$ and its synergy with the Replace component $F_u(.)$: The error detection component $F_e(.)$ is a key element in our generic architecture and its purpose is to indicate which are the image regions whose labels are incorrect. This type of information is exploited in the next step of label replacement in two ways. Firstly, the

Replace component $F_u(.)$ that gets as input the error map E, which is generated by $F_e(.)$, is able to know which are the image regions whose labels need to be replaced and thus it is able to focus its attention only on those image regions. At this point note that, in equation 4.2, the error maps E, apart from being given as input attention maps to the Replace component $F_u(.)$, also act as gates that control which way the information will flow both during the forward propagation and during the backward propagation. Specifically, during the forward propagation case, in the cases that the error map probabilities are either 0 or 1, it holds that:

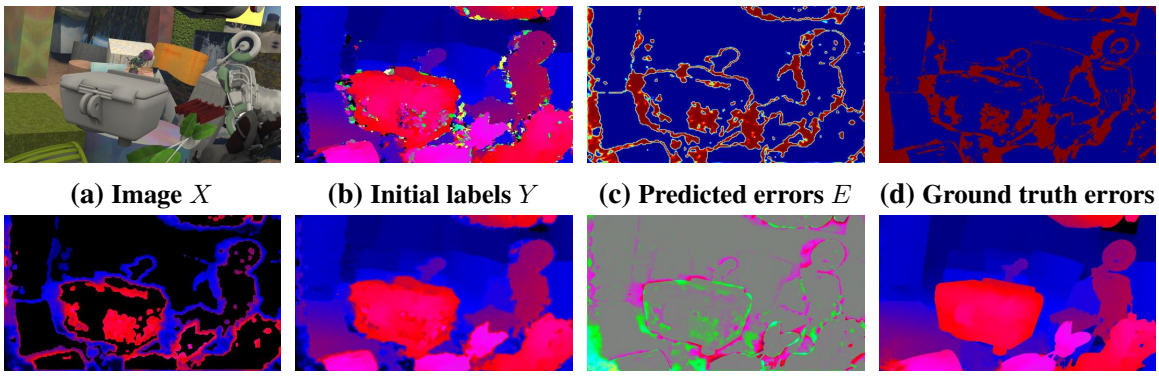
$$U = \begin{cases} Y, & \text{if } F_e(X,Y) = \mathbf{0}, \\ F_u(X,Y,E), & \text{if } F_e(X,Y) = \mathbf{1}, \end{cases} \tag{4.4}$$

which basically means that the Replace component $F_u(.)$ is being utilized mainly for the erroneously labeled image regions. Also, during the backward propagation, it is easy to see that the gradients of the replace function w.r.t. the loss L (in the cases that the error probabilities are either 0 or 1) are:

$$\frac{dL}{dF_u(.)} = \begin{cases} \mathbf{0}, & \text{if } F_e(X,Y) = \mathbf{0}, \\ \frac{dL}{dU}, & \text{if } F_e(X,Y) = \mathbf{1}, \end{cases} \tag{4.5}$$

which means that gradients are back-propagated through the Replace component $F_u(.)$ only for the erroneously labeled image regions. So, in a nutshell, during the learning procedure the Replace component $F_u(.)$ is explicitly trained to predict new values mainly for the erroneously labeled image regions. The second advantage of giving the error maps E as input to the Replace component $F_u(.)$, is that this allows the Replace component to know which image regions contain "trusted" labels that can be used for providing information on how to fill the erroneously labeled regions.

Estimated error probability maps by the Detection component $F_e(.)$: Thanks to the topology of our generic architecture, by optimizing the reconstruction of the ground truth labels $\hat{Y}$, the error detection component $F_e(.)$ implicitly learns to act as a joint probability model for patches of X and Y centered on each pixel of the input image, assigning a high



(a) Image X **(b) Initial labels Y** **(c) Predicted errors E** **(d) Ground truth errors**

(e) $F_u(.)$ predictions **(f) Renewed labels U** **(g) $F_r(.)$ residuals** **(h) Final labels Y'**

Figure 4-3: Here we provide an example that illustrates the functions performed by the Detect, Replace, and Refine steps in our proposed architecture. The example is coming from the dense disparity labeling task (stereo matching). Specifically, subfigures **(a)**, **(b)**, and **(c)** depict respectively the input image X, the initial disparity label estimates Y, and the error probability map E that the detection component $F_e(.)$ yields for the initial labels Y. Notice the high similarity of map E with the ground truth error map of the initial labels Y depicted in subfigure **(d)**, where the ground truth error map has been computed by thresholding the absolute difference of the initial labels Y from the ground truth labels with a threshold of 3 pixels (red are the erroneous pixel labels). In subfigure **(e)** we depict the label predictions of the Replace component $F_u(.)$. For visualization purposes we only depict the $F_u(.)$ pixel predictions that will replace the initial labels that are incorrect (according to the detection component) by drawing the remaining ones (i.e., those whose error probability is less than 0.5) with black color. In subfigure **(f)** we depict the renewed labels $U = E \odot F_u(X, Y, E) + (1 - E) \odot Y$. In subfigure **(g)** we depict the residual corrections that the Refine component $F_r(.)$ yields for the renewed labels U. Finally, in the last subfigure **(h)** we depict the final label estimates $Y' = U + F_r(X, Y, E, U)$ that the Refine step yields.

probability of error for patches that do not appear to belong to the joint input-output space (X, Y). In Figures 4-3c and 4-3d we visualize the estimated by the Detection component $F_e(.)$ error maps and the ground truth error maps in the context of the disparity estimation task (more visualizations are provided in Figure 4-6). It is interesting to note that the estimated error probability maps are very similar to the ground truth error maps despite the fact that we are not explicitly enforcing this behaviour, e.g., through the use of an auxiliary loss.

Error detection component and Highway Networks: Note that the way the Detection component $F_e(.)$ and Replace component $F_u(.)$ interact bears some resemblance to the basic building blocks of the Highway Networks [146] that are being utilized for training extremely deep neural network architectures. Briefly, each highway building block gets as input some hidden feature maps and then predicts transform gates that control which feature values will

be carried on the next layer as is and which will be transformed by a non-linear function. There are however some important differences. For instance, in our case the error gate prediction and the label replacement steps are executed in sequence with the latter one getting as input the output of the former one. Instead, in Highway Networks the gate prediction and the non-linear transform of the input feature maps are performed in parallel. Furthermore, in Highway Networks the components of each building block are implemented by simple affine transforms followed by non-linearities and the purpose is to have multiple building blocks stacked one on top of the other in order to learn extremely deep image representations. In contrast, the components of our generic architecture are themselves deep neural networks and the purpose is to learn to reconstruct the input labels Y.

Two stage refinement approach: Another key element in our architecture is that the step of predicting new, more accurate labels Y', given the initial labels Y, is broken in two stages. The first stage is handled by the error detection component $F_e(.)$ and the label replacement component $F_u(.)$. Their job is to correct only the "hard" mistakes of the input labels Y. They are not meant to correct "soft" mistakes (i.e., errors in the label values of small magnitude). In order to learn to correct those "soft" mistakes, it is more appropriate to use a component that yields residual corrections w.r.t. its input. This is the purpose of our Refine component $F_r(.)$, in the second stage of our architecture, from which we expect to improve the "details" of the output labels U by better aligning them with the fine structures of the input images. This separation of roles between the first and the second refinement stages (i.e., coarse refinement and then fine-detail refinement) has the potential advantage, which is exploited in our work, to perform the actions of the first stage in lower resolution thus speeding up the processing and reducing the memory footprint of the network. Also, the end-to-end training procedure allows the components in the first stage (i.e., $F_e(.)$ and $F_u(.)$) to make mistakes as long as those are corrected by the second stage. This aspect of our architecture has the advantage that each component can more efficiently exploit its available capacity.

4.2.3 Explored architectures

In order to evaluate the proposed architecture we also devised and tested various other architectures that consist of the same core components as those that we propose. In total, the architectures that are explored in our work are:

***Detect + Replace + Refine* architecture:** This is the architecture that we proposed in section 4.2.1.

***Replace* baseline architecture:** In this case the model directly replaces the old labels with new ones: $Y' = F_u(X, Y)$.

***Refine* baseline architecture:** In this case the model predicts residual corrections w.r.t. the input labels: $Y' = Y + F_r(X, Y)$.

***Replace + Refine* architecture:** Here the model first replaces the entire label map Y with new values $U = F_u(X, Y)$ and then residual corrections are predicted w.r.t. the updated values U, $Y' = U + F_r(X, Y, U)$.

***Detect + Replace* architecture:** Here the model first detects errors on the input label maps $E = F_e(X, Y)$ and then replace those erroneous pixel labels $Y' = E \odot F_u(X, Y, E) + (1 - E) \odot Y$.

***Detect + Refine* architecture:** In this case, after the detection of the errors $E = F_e(X, Y)$, the erroneous pixel labels are masked out by setting them to the mean label value l_{mu}, $U = E \odot l_{mu} + (1 - E) \odot Y$. Then the masked label maps are given as input to a residual refinement model $Y' = U + F_r(X, Y, E, U)$. Note that this architecture can also be considered as a specific instance of the general Detect + Replace + Refine architecture where the Replace component $F_u(.)$ does not have any learnable parameters and constantly returns the mean label value, i.e., $F_u(.) = l_{mu}$.

***Parallel* architecture:** Here, after the detection of the errors, the erroneous labels are replaced by the Replace component $F_u(.)$ while the rest labels are refined by the Refine component $F_r(.)$. More specifically, the operations performed by this architecture are described by the following equations:

$$E = F_e(X, Y), \tag{4.6}$$

$$U_1 = F_u(X, Y, E), U_2 = Y + F_r(X, Y, E), \tag{4.7}$$

$$Y' = E \odot U_1 + (1 - E) \odot U_2. \tag{4.8}$$

Basically, in this architecture the components $F_u(.)$ and $F_r(.)$ are applied in parallel instead of the sequential topology that is chosen in the Detect + Replace + Refine architecture.

Detect + Replace + Refine $\times T$**:** This is basically the Detect + Replace + Refine architecture but applied iteratively for T iterations. Note that the model implementing this architecture is trained in a multi-iteration manner (i.e., by feeding the output labels generated at one iteration as input to the network at the next iteration).

X-Blind Detect + Replace + Refine **architecture:** This is a "blind" w.r.t. the image X version of the *Detect + Replace + Refine* architecture. Specifically, the "X-Blind" architecture is exactly the same as the proposed *Detect + Replace + Refine* architecture with the only difference being that it gets as input only the initial labels Y and not the image X (i.e., none of the $F_e(.)$, $F_u(.)$, and $F_r(.)$ components depends on the image X). Hence, the model implemented by the "X-Blind" architecture must learn to reconstruct the ground truth labels by only "seeing" a corrupted version of them.

4.3 Detect, Replace, Refine for disparity estimation

In order to evaluate the proposed dense image labeling architecture, as well as the other alternative architectures that are explored in our work, we use the dense disparity estimation (stereo matching) task, according to which, given a left and right image, one needs to assign to each pixel of the left image a continuous label that indicates its horizontal displacement in the right image (disparity). Such a task forms a very interesting and challenging testbed for the evaluation of dense labeling algorithms since it requires dealing with several challenges such as accurately preserving disparity discontinuities across object boundaries, dealing with occlusions, as well as recovering the fine details of disparity maps. At the same time it has many practical applications on various autonomous driving and robot navigation or grasping tasks.

4.3.1 Initial disparities

Generating initial disparity field: In all the examined architectures, in order to generate the initial disparity labels Y we used the deep patch matching approach that was proposed by W. Luo et al. [96] and specifically their architecture with id 37. We then train our models to reconstruct the ground truth labels given as input only the left image X and the initial disparity labels Y. We would like to stress out that the right image of the stereo pair is not provided to our models. This practically means that the trained models cannot rely only on the image evidence for performing the dense disparity labeling task – since disparity prediction from a single image is an ill-posed problem – but they have to learn the joint space of both input X and output labels Y in order to perform the task.

Image & disparity field normalization: Before we feed an image and its initial disparity field to any of our examined architectures, we normalize them to zero mean and unit variance (i.e., mean subtraction and division by the standard deviation). The mean and standard deviation values of the RGB colors and disparity labels are computed on the entire training set. The disparity target labels are also normalized with the same mean and standard deviation values and during inference the normalization effect is inverted on the disparity fields predicted by the examined architectures.

4.3.2 Deep neural network architectures

Each component of our generic architecture can be implemented by a deep neural network. For our disparity estimation experiments we chose the following implementations:

Error detection component: It is implemented by 5 convolutional layers of which the last one yields the error probability map E. All the convolutional layers, apart from the last one, are followed by batch normalization [66] plus ReLU [97] units. Instead, the last convolutional layer is followed by a sigmoid unit. The first two convolutions are followed by max-pooling layers of kernel size 2 that in total reduce the input resolution by a factor of 4. To compensate, a bi-linear up-sampling layer is placed on top of the last convolution layer in order the output probability map E to have the same resolution as the input image. The number of output feature planes of each of the 5 convolutional layers is 32, 64, 128,

256, and 1 correspondingly.

Replace component: It is implemented with a convolutional architecture that first "compress" the resolution of the feature maps to $\frac{1}{64}$ of the input resolution and then "decompress" the resolution to $\frac{1}{4}$ of the input resolution. For its implementation we follow the guidelines of A. Newel et al. [109] which are to use residual blocks [57] on each layer and parametrized (by residual blocks) skip connection between the symmetric layers in the "compressing" and the "decompressing" parts of the architecture. The "compressing" part of the architecture uses max-pooling layers with kernel size 2 to down-sample the resolution while the "decompressing" part uses nearest-neighbor up-sampling (by a factor of 2). We refer for more details to A. Newel et al. [109]. In our case, during the "compression" part there are in total 6 down-sampling convolutional blocks and during the "decompression" part 4 up-sampling convolutional blocks. The number of output feature planes in the first layer is 32 and each time the resolution is down-sampled the number of feature planes is increased by a factor of 2. For GPU memory efficiency reasons, we do not allow the number of output feature planes of any layer to exceed that of 512. During the "decompression" part, each time we up-sample the resolution we also decrease by a factor of 2 the number of feature planes. The last convolution layer yields a single feature plane with the new disparity labels (without any non-linearity). As already explained, during the "decompressing" part the resolution is increased till that of $\frac{1}{4}$ of the input resolution. The reason for early-stopping the "decompression" is that the Replace component is needed to only perform crude "fixes" of the initial labels and thus further "decompression" steps are not necessary. Before the disparity labels are fed to the next processing steps, bi-linear up-sampling by a factor of 4 (without any learn-able parameter) is being used in order to restore the resolution to that of the input resolution.

Refine component: It follows the same architecture as the replace component with the exception that during the "compressing" part the resolution of the feature maps is reduced till $\frac{1}{16}$ of the input resolution and then during the "decompressing" part the resolution is restored to that of the input resolution.

Alternative architectures: In case the alternative architectures have missing components, then the number of layers and/or the number of feature planes per layer of the

remaining components is being increased such that the total capacity (i.e., number of learnable parameters) remains the same. For the architectures that include only the Replace or Refine components (i.e., *Replace*, *Refine*, *Detect+Replace*, and *Detect+Refine* architectures) the "compression" - "decompression" architecture of this component "compresses" the resolution till $\frac{1}{64}$ of the input resolution and then "decompresses" it to the same resolution as the input image.

Weight initialization: In order to initialize the weights of each convolutional layer we use the initialization scheme proposed by K. He et al. [54].

4.3.3 Training details

We used the $L1$ loss as objective function and the networks were optimized using the Adam [71] method with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The learning rate lr was set to 10^{-3} and was decreased after 20 epochs to 10^{-4} and then after 15 epochs to 10^{-5}. We then continued optimizing for another 5 epochs. Each epoch lasted approximately 2000 batch iterations where each batch consisted of 24 training samples. Each training sample consists of patches with spatial size 256×256 and 4 channels (3 RGB color channels + 1 initial disparity label channel). The patches are generated by randomly cropping with uniform distribution an image and its corresponding initial disparity labels.

Augmentation: During training we used horizontal flip augmentation and chromatic transformations such as color, contrast, and brightness transformations.

4.4 Experimental results

In this section we present an exhaustive experimental evaluation of the proposed architecture as well as of the other explored architectures in the task of dense disparity estimation. Specifically, we first describe the evaluation settings used in our experiments (section 4.4.1), then we report detailed quantitative results w.r.t. the examined architectures (section 4.4.2), and finally we provide qualitative results of the proposed *Detect, Replace, Refine* architecture and all of its components, trying in this way to more clearly illustrate their role (section 4.4.3).

4.4.1 Experimental settings

Training set: In order to train the explored architectures we used the large scale synthetic dataset for disparity estimation that was recently introduced by N. Mayer et al. [101]. We call this dataset the Synthetic dataset. It consists of three different type of synthetic image sequences and includes around $34k$ stereo images. Also, we enriched this training set with 160 images from the training set of the KITTI 2015 dataset [103, 104][2].

Evaluation sets: We evaluated our architectures on three different datasets. On 2000 images from the test split of the Synthetic dataset, on 40 validation images coming from KITTI 2015 training dataset, and on 15 images from the training set of the Middlebury dataset [133]. Prior to evaluating the explored architectures in the KITTI 2015 validation set, we fine-tuned the models that implement them only on the 160 image of the KITTI 2015 training split. In this case, we start training for 20 epochs with a learning rate of 10^{-4}, we then reduce the learning rate to 10^{-5} and continue training for 15 epochs, and then reduce again the learning rate to 10^{-6} and continue training for 5 more epochs (in total 40 epochs). The epoch size is set to 400 batch iterations.

Evaluation metrics: For evaluation we used the end-point-error (EPE), which is the averaged Euclidean distance from the ground truth disparity, and the percentage of disparity estimates whose absolute difference from the ground truth disparity is more than t pixels ($>$ t pixel). Those metrics are reported for the non-occluded pixels (Non-Occ), all the pixels (All), and only the occluded pixels (Occ).

4.4.2 Quantitative results

4.4.2.1 Disparity estimation performance

In Tables 4.1, 4.2, and 4.3 we report the stereo matching performance of the examined architectures in the Synthetic, Middlebury, and KITTI 2015 evaluation sets correspondingly.

Single-iteration results: We first evaluate all the examined architectures when they are applied for a single iteration. We observe that all of them are able to improve the initial

[2]The entire training set of KITTI 2015 includes 200 images. In our case we split those 200 images in 160 images that were used for training purposes and 40 images that were used for validation purposes

	> 2 pixel	> 3 pixel	> 4 pixel	> 5 pixel	EPE
Architectures	All	All	All	All	All
Initial labels Y	24.3175	22.9004	21.9140	21.1680	12.0218
Single-iteration results					
Replace (baseline)	12.8007	10.4512	8.8966	7.7467	2.4456
Refine (baseline)	14.5996	12.2246	10.3046	8.7873	2.1235
Replace + Refine	11.1152	9.1821	7.8430	6.8550	2.2356
Detect + Replace	11.6970	9.2419	7.6812	6.6018	2.1504
Detect + Refine	10.5309	8.5565	7.2154	6.2186	1.8210
Parallel	11.0146	8.9261	7.5029	6.4742	2.0241
Detect + Replace + Refine	9.5981	7.9764	6.7895	5.9074	1.8569
Multi-iteration results					
Detect + Replace + Refine x2	**8.8411**	**7.2187**	**6.0987**	**5.2853**	**1.6899**

Table 4.1: Stereo matching results on the Synthetic dataset.

	> 2 pixel			> 3 pixel			> 4 pixel			> 5 pixel			EPE		
Architectures	Non-Occ	All	Occ	Non-Occ	All	Occ	Non-Occ	All	Occ	Non-Occ	All	Occ	Non-Occ	All	Occ
Initial labels Y	18.243	26.714	86.125	15.664	23.986	82.330	14.208	22.282	78.758	13.237	21.044	75.579	6.058	8.709	25.598
Single-iteration results															
Replace (baseline)	15.767	21.089	57.197	12.323	16.793	46.303	10.312	14.020	37.922	9.032	12.147	31.770	2.731	3.221	5.818
Refine (baseline)	13.981	19.742	58.039	11.110	16.042	47.732	9.266	13.406	39.218	7.889	11.392	32.467	1.953	2.551	5.665
Replace + Refine	14.262	19.257	52.036	11.297	15.701	43.905	9.552	13.459	37.910	8.408	11.891	33.125	2.292	2.908	6.216
Detect + Replace	15.368	20.984	58.745	11.243	16.169	48.568	8.957	13.176	40.663	7.571	11.179	34.482	2.013	2.676	6.462
Detect + Refine	13.732	19.375	56.383	10.718	15.552	46.281	8.893	12.975	38.197	7.600	11.012	31.478	2.105	2.626	5.389
Parallel	14.917	20.345	57.459	11.363	15.907	46.221	9.234	12.941	37.218	7.840	10.940	30.854	2.012	2.552	5.607
Detect + Replace + Refine	12.845	17.825	50.407	10.096	14.379	41.704	8.285	11.957	34.801	7.057	10.253	29.560	**1.774**	2.368	5.457
Multi-iteration results															
Detect + Replace + Refine x2	**11.529**	**16.414**	**47.922**	**8.757**	**12.874**	**37.977**	**6.997**	**10.482**	**30.634**	**5.911**	**8.916**	**25.514**	1.789	**2.321**	**4.971**

Table 4.2: Stereo matching results on Middlebury.

	> 2 pixel			> 3 pixel			> 4 pixel			> 5 pixel			EPE		
Architectures	Non-Occ	All	Occ	Non-Occ	All	Occ	Non-Occ	All	Occ	Non-Occ	All	Occ	Non-Occ	All	Occ
Initial labels Y	8.831	10.649	98.098	6.412	8.253	96.559	5.222	7.059	94.742	4.514	6.339	93.139	1.700	2.457	31.214
Single-iteration results															
Replace (Baseline)	4.997	5.668	37.327	3.329	3.888	27.890	2.452	2.892	19.643	1.924	2.292	15.226	0.858	0.923	3.165
Refine (Baseline)	4.429	5.165	33.028	3.075	3.714	25.107	2.370	2.924	19.610	1.933	2.404	15.978	0.867	0.953	3.384
Replace + Refine	3.963	4.529	**27.411**	2.712	3.209	21.465	2.082	2.507	16.481	1.735	2.098	13.611	0.802	0.865	2.859
Detect + Replace	5.126	5.751	35.554	3.469	4.005	27.656	2.517	2.953	20.519	1.911	2.269	15.947	0.886	0.943	3.108
Detect + Refine	4.482	5.169	34.992	3.054	3.634	26.453	2.328	2.799	19.004	1.865	2.258	14.686	0.863	0.926	2.952
Parallel	5.239	5.952	38.392	3.530	4.139	29.436	2.522	3.017	21.208	1.943	2.338	15.748	0.904	0.962	3.095
Detect + Replace + Refine	3.919	4.610	33.947	2.708	3.294	25.697	2.082	2.570	19.123	1.699	2.112	15.140	0.790	0.858	3.056
Multi-iteration results															
Detect + Replace + Refine x2	**3.685**	**4.277**	28.164	**2.577**	**3.075**	**20.762**	**2.001**	**2.424**	**16.086**	**1.652**	**2.004**	**13.056**	**0.779**	**0.835**	**2.723**

Table 4.3: Stereo matching results on KITTI 2015 validation set.

label estimates Y. However, they do not all of them achieve it with the same success. For instance, the baseline models *Replace* and *Refine* tend to be less accurate than the other models. Compared to them, the *Detect + Replace* and the *Detect + Refine* architectures perform considerably better in two out of three datasets, the Synthetic and the Middlebury datasets. This improvement can only be attributed to the error detection step, which is what distinguishes them from the baselines, and indicates the importance of having an error detection component in the dense labeling task. Overall, the best single-iteration performance is achieved by the *Detect + Replace + Refine* architecture that we propose here and which combines both the merits of the error detection component and the two stage refinement strategy. Compared to it, the *Parallel* architecture has considerably worse

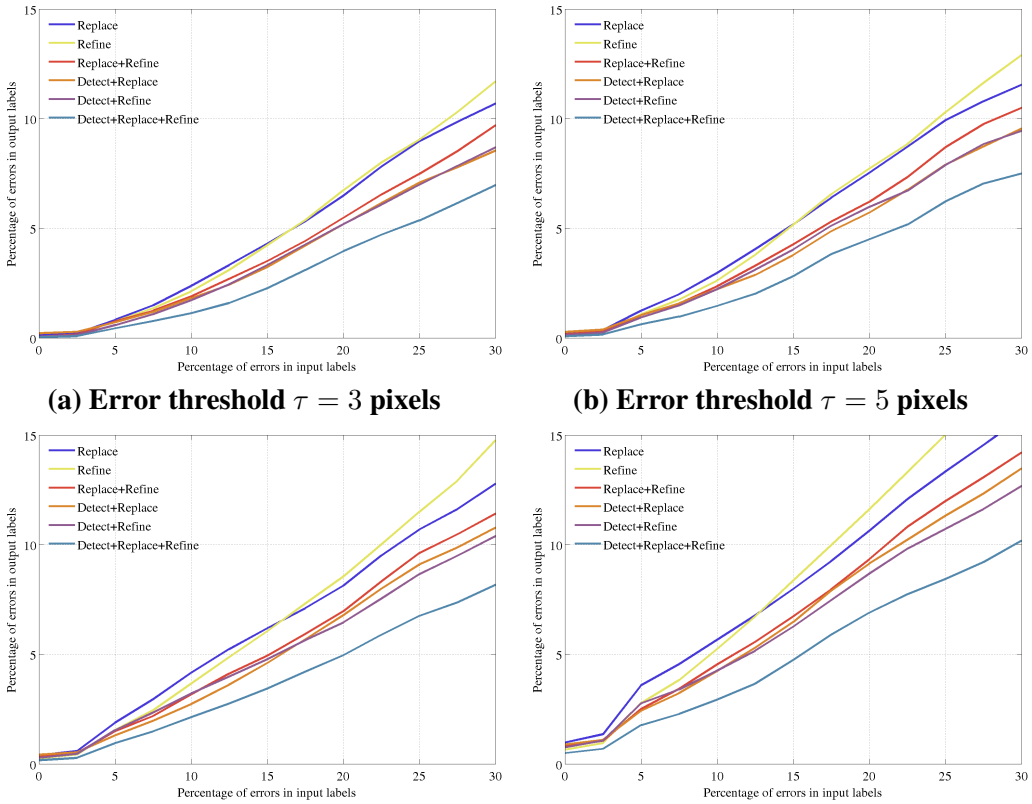
performance, which indicates that the sequential order in the proposed architecture is important for achieving accurate results.

Multi-iteration results: We also evaluated our best performing architecture, which is the *Detect* + *Replace* + *Refine* architecture that we propose, in the multiple iteration case. Specifically, the last entry *Detect* + *Replace* + *Refine x2* in Tables 4.1, 4.2, and 4.3 indicates the results of the proposed architecture for 2 iterations and we observe that it further improves the performance w.r.t. the single iteration case. For more than 2 iterations we did not see any further improvement and for this reason we chose not to include those results. Note that in order to train this two iterations model, we first pre-train the single iteration version and then fine-tune the two iterations version by adding the generated disparity labels from the first iteration in the training set.

4.4.2.2 Label prediction accuracy Vs initial labels quality

In Figure 4-4 we evaluate the ability of each architecture to predict the correct disparity label for each pixel x as a function of the "quality" of the initial disparity labels in a $w \times w$ neighborhood of that pixel. To that end, we plot for each architecture the percentage of erroneously estimated disparity labels as a function of the percentage of erroneous initial disparity labels that exist in the patch of size $w \times w$ centered on the pixel of interest x. In our case, the size of the neighborhood w is set to 65. An estimated pixel label y' for the pixel x is considered erroneous if its absolute difference from the ground truth label is more than $\tau_0 = 3$ pixels. For the initial disparity labels in the patch centered on x, the threshold τ of considering them incorrect is set to $\tau = 3$ (Fig. 4-4.a), $\tau = 5$ (Fig. 4-4.b), $\tau = 8$ (Fig. 4-4.c), or $\tau = 15$ (Fig. 4-4.d). We make the following observations (that are more clearly illustrated from sub-figures 4-4.c and 4-4.d):

- In the case of the *Replace* and *Refine* architectures, when the percentage of erroneous initial labels is low (e.g., less than 10%) then the *Refine* architecture (which predicts residual corrections) is considerably more accurate than the *Replace* architecture (which directly predicts new label values). However, when the percentage of erroneous initial labels is high (e.g., more than 20%) then the *Replace* architecture is more accurate than the *Refine* one. This observation supports our argument that residual



(a) Error threshold $\tau = 3$ pixels **(b) Error threshold $\tau = 5$ pixels**

(c) Error threshold $\tau = 8$ pixels **(d) Error threshold $\tau = 15$ pixels**

Figure 4-4: Percentage of erroneously estimated disparity labels for a pixel x as a function of the percentage of erroneous initial disparity labels in the patch of size $w \times w$ centered on the pixel of interest x. The patch size w is set to 65. An estimated pixel label y' is considered erroneous if its absolute difference from the ground truth label is more than $\tau_0 = 3$ pixels. For the initial disparity labels in each patch, the threshold τ of considering them incorrect is set to **(a)** 3 pixels, **(b)** 5 pixels, **(c)** 8 pixels, and **(d)** 15 pixels. The evaluation is performed on 50 images of the *Synthetic* test set.

corrections are more suitable for "soft" mistakes in the initial labels while predicting an entirely new label value is a better choice for the "hard" mistakes.

- By introducing the error detection component, both the *Refine* and the *Replace* architectures manage to significantly improve their predictions. In the *Detect+Refine* case, the improvement is due to the fact that the error detection component sets the "hard" mistakes to the mean label values (see the description of the *Detect+Refine* architecture) thus allowing the *Refine* component to ignore the values of the "hard" mistakes of the initial labels and instead make residual predictions w.r.t. the mean label values (these mean values are fixed and known in advance and thus it is easier for the network to learn to make residual predictions w.r.t. them). In the case of the

	All / All			All / Est			Noc / All			Noc / Est			Runtime
Architectures	D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all	(secs)
Ours	**2.58**	6.04	**3.16**	**2.58**	6.04	**3.16**	2.34	4.87	**2.76**	2.34	4.87	**2.76**	0.4
DispNetC [101]	4.32	**4.41**	4.34	4.32	**4.41**	4.34	4.11	**3.72**	4.05	4.11	**3.72**	4.05	0.06
PBCB [136]	**2.58**	8.74	3.61	**2.58**	8.74	3.6	**2.27**	7.71	3.17	**2.27**	7.71	3.17	68
Displets v2 [47]	3.00	5.56	3.43	3.00	5.56	3.43	2.73	4.95	3.09	2.73	4.95	3.09	265
MC-CNN [166]	2.89	8.88	3.89	2.89	8.88	3.88	2.48	7.64	3.33	2.48	7.64	3.33	67
SPS-St [158]	3.84	12.67	5.31	3.84	12.67	5.31	3.50	11.61	4.84	3.50	11.61	4.84	2
MBM [30]	4.69	13.05	6.08	4.69	13.05	6.08	4.33	12.12	5.61	4.33	12.12	5.61	0.13

Table 4.4: Stereo matching results on KITTI 2015 test set.

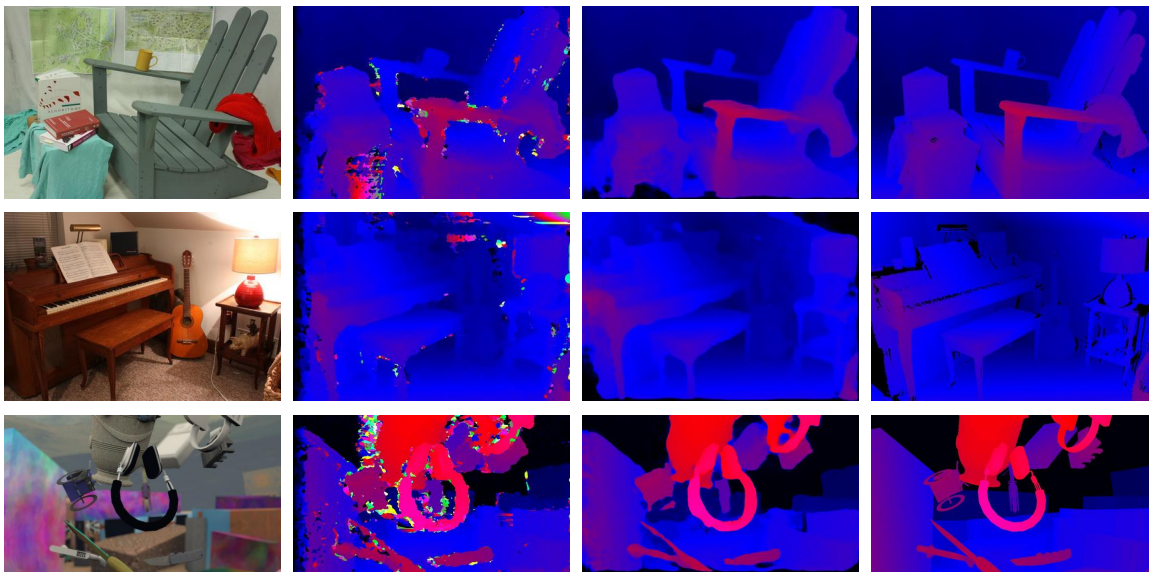
Detect+Replace architecture, the error detection component "dictates" the *Replace* component to predict new label values for the incorrect initial labels while allowing the propagation of the correct ones in the output.

- Finally, the best *"label prediction accuracy Vs initial labels quality"* behavior is achieved by the proposed *Detect + Replace + Refine* architecture, which efficiently combines the error detection component with the two-stage label improvement approach. Interestingly, the improvement margins w.r.t. the rest architectures is increased as the quality of the initial labels is decreased.

4.4.2.3 KITTI 2015 test set results

We submitted our best solution, which is the proposed *Detect + Replace + Refine* architecture applied for two iterations, on the KITTI 2015 test set evaluation server and we achieved state-of-the-art results in the main evaluation metric, D1-all, surpassing at the time of submission all prior work by a significant margin. The results of our submission, as well as of other competing methods, are reported in Table 4.4[3]. Note that our improvement w.r.t. the best prior approach corresponds to a more than 10% relative reduction of the error rate. Our total execution time is 0.4 secs, of which around 0.37 secs is used by the patch matching algorithm for generating the initial disparity labels and the rest 0.03 by our *Detect + Replace + Refine x2* architecture (measured in a Titan X GPU). For this submission, after having train the *Detect + Replace + Refine x2* model on the training split (160 images), we further fine-tuned it on both the training and the validation splits (in which we divided the 200

[3]The link to our KITTI 2015 submission that contains more thorough test set results – both qualitative and quantitative – is:
`http://www.cvlibs.net/datasets/kitti/eval_scene_flow_detail.php?benchmark=stereo&result=365eacbf1effa761ed07aaa674a9b61c60fe9300`



(a) Image X **(b) Initial labels Y** **(c) Final labels Y'** **(d) Target labels**

Figure 4-5: Here we illustrate some examples of the disparity predictions that the "X-Blind" architecture performs. The illustrated examples are from the Synthetic and the Middlebury datasets.

	> 2 pixel			> 3 pixel			> 4 pixel			> 5 pixel			EPE		
Architectures	Non-Occ	All	Occ	Non-Occ	All	Occ	Non-Occ	All	Occ	Non-Occ	All	Occ	Non-Occ	All	Occ
Synthetic dataset															
Initial labels Y		24.3175			22.9004			21.9140			21.1680			12.0218	
Detect + Replace + Refine		9.5981			7.9764			6.7895			5.9074			1.8569	
"X-Blind"		16.0014			14.0196			12.5170			11.3758			3.8810	
Middlebury dataset															
Initial labels Y	18.243	26.714	86.125	15.664	23.986	82.330	14.208	22.282	78.758	13.237	21.044	75.579	6.058	8.709	25.598
Detect + Replace + Refine	12.845	17.825	50.407	10.096	14.379	41.704	8.285	11.957	34.801	7.057	10.253	29.560	1.774	2.368	5.457
"X-Blind"	16.845	22.037	57.324	14.038	18.562	48.356	12.212	16.217	41.941	10.914	14.509	37.022	2.878	3.656	7.945
KITTI 2015 dataset															
Initial labels Y	8.831	10.649	98.098	6.412	8.253	96.559	5.222	7.059	94.742	4.514	6.339	93.139	1.700	2.457	31.214
Detect + Replace + Refine	3.919	4.610	33.947	2.708	3.294	25.697	2.082	2.570	19.123	1.699	2.112	15.140	0.790	0.858	3.056
"X-Blind"	5.040	5.602	32.575	3.671	4.135	24.566	2.722	3.099	18.069	2.191	2.505	14.359	0.910	0.966	2.997

Table 4.5: Stereo matching results for the "X-Blind" architecture. We also include the corresponding results of the proposed *Detect + Replace + Refine* architecture to facilitate their comparison.

images of KITTI 2015 training dataset).

4.4.2.4 "X-Blind" Detect + Replace + Refine architecture

Here we evaluate the "X-Blind" architecture that, as already explained, it is exactly the same as the proposed *Detect + Replace + Refine* architecture with the only difference being that as input it gets only the initial labels Y and not the image X. The purpose of evaluating such an architecture is not to examine a competitive variant of the main *Detect + Replace + Refine* architecture, but rather to explore the capabilities of the latter one in such a scenario. In Table 4.5 we provide the stereo matching results of the "X-Blind" architecture. We observe that it might not be able to compete the original *Detect + Replace + Refine* architecture

but it still can significantly improve the initial disparity label estimates. In Figure 4-5 we illustrate some disparity prediction examples generated by the "X-Blind" architecture. We observe that the "X-Blind" architecture manages to considerably improve the quality of the initial disparity label estimates, however, since it does not have the image X to guide it, it is not able to accurately reconstruct the disparity field on the borders of the objects.

4.4.3 Qualitative results

This section includes qualitative examples that help illustrating the role of the various components of our proposed architecture.

4.4.3.1 Error Detection step

In Figure 4-6 we provide additional examples of error probability maps E (that the error detection component $F_e(X, Y)$ generated w.r.t. the initial labels Y) and compare them with the ground truth error maps of the initial labels. The ground truth error maps are computed by thresholding the absolute difference of the initial labels Y from the ground truth labels with a threshold of 3 pixels (red are the erroneous pixel labels in the figure). Note that this is the logic that is usually followed in the disparity task for considering a pixel label erroneous. We observe that, despite the fact the error detection component $F_e(.)$ is not explicitly trained to produce such ground truth error maps, its predictions still highly correlate with them. This implies that the error detection component $F_e(.)$ seems to have learnt to recognize the areas that look abnormal/atypical with respect to the joint input-output space $\{X, Y\}$ (i.e., it has learnt the "structure" of that space).

4.4.3.2 Replace step

In Figure 4-7 we provide several examples that more clearly illustrate the function performed by the Replace step in our proposed architecture. Specifically, in sub-figures 4-7a, 4-7b, and 4-7c we depict the input image X, the initial disparity label estimates Y, and the error probability map E that the detection component $F_e(.)$ yields for the initial labels Y. In sub-figure 4-7d we depict the label predictions of the replace component $F_u(.)$. For visualization

purposes we only depict the $F_u(.)$ pixel predictions that will replace the initial labels that are incorrect (according to the detection component) by drawing the remaining ones (i.e., those whose error probability is less than 0.5) with black color. Finally, in the last sub-figure 4-7e we depict the renewed labels $U = E \odot F_u(X, Y, E) + (1 - E) \odot Y$. We can readily observe that most of the "hard" mistakes of the initial labels Y have now been crudely "fixed" by the Replace component.

4.4.3.3 Refine step

In Figure 4-8 we provide several examples that more clearly illustrate the function performed by the Refine step in our proposed architecture. Specifically, in sub-figures 4-8a, 4-8b, and 4-8c we depict the input image X, the initial disparity label estimates Y, and the renewed labels U that the Replace step yields. In sub-figure 4-8d we depict the residual corrections that the Refine component $F_r(.)$ yields for the renewed labels U. Finally, in last sub-figure 4-8e we depict the final label estimates $Y' = U + F_r(X, Y, E, U)$ that the Refine step yields. We observe that most of residual corrections that the Refine component $F_r(.)$ yields are concentrated on the borders of the objects. Furthermore, by adding those residuals on the renewed labels U, the Refine step manages to refine the renewed labels U and align the estimated labels Y' with the fine image structures in X.

4.4.3.4 Detect, Replace, Refine pipeline

In Figure 4-9 we illustrate the entire work-flow of the *Detect + Replace + Refine* architecture that we propose and we compare its predictions Y' with the ground truth disparity labels.

4.4.3.5 Multi-iteration architecture

In Figure 4-10, we illustrate the estimated disparity labels after each iteration of our multi-iteration architecture *Detect + Replace + Refine x2* that in our experiments achieved the most accurate results. We observe that the 2^{nd} iteration further improves the fine details of the estimated disparity labels delivering a higher fidelity disparity field. Furthermore, applying the model for a 2^{nd} iteration results in a disparity field that looks more "natural",

i.e., visually plausible.

4.4.3.6 KITTI 2015 qualitative results

We provide qualitative results from KITTI 2015 validation set in Figure 4-11. In order to generate them we used the *Detect + Replace + Refine x2* architecture that gave the best quantitative results. We observe that our model is able to recover a good estimate of the actual disparity map even when the initial label estimates are severely corrupted.

4.5 Experiments on semantic segmentation

In this section we provide some preliminary results obtained by applying the proposed dense image labeling architecture to two semantic segmentation tasks. Note that in semantic segmentation, each pixel of an image must be labeled with a semantic category (e.g., road, building, window, door, fence, etc.).

4.5.1 Implementation details for the semantic segmentation case

In order to generate the initial labels Y in the semantic segmentation case we used an FCN like architecture [92] based on the ResNet50 [57] network backbone. The proposed deep joint input-output model, apart from the image X and the initial labels Y, also takes as input feature maps generated by the FCN model during the label initialization step. We found that this modification improves the quality of the generated labels. We also found advantageous to apply a binary cross entropy loss on the error detection outputs using ground truth error maps (defined from the initial label maps and the ground truth label maps) in order to better force the network to learn the error detection step. Finally, in order to speed-up inference time, the Detect, Replace, Refine steps are implemented with a single network that predicts all those three outputs simultaneously.

4.5.2 Cityscape results

We applied the proposed dense image labeling algorithm in the Cityscapes dataset [16] and our algorithm manages to improve the segmentation accuracy (measured with the mean Intersection-over-Union metric) from 70.09% (the *Initial labels Y* case) to 73.23% (the *Detect + Replace + Refine* case). In Figure 4-12 we visualize the initial labels and the labels estimated by our *Detect + Replace + Refine* architecture. We observe that the proposed dense labeling algorithm has managed to improve the labeling accuracy on the borders of the objects and also to recover objects with thin elongated structures (e.g., poles) that were lost in the initial labels.

4.5.3 Facade Parsing results

We applied the proposed *Detect + Replace + Refine* labeling algorithm on the facade parsing ECP dataset [151] and we provide visualizations in Figure 4-13. We observe again that our dense labeling algorithms manages to significantly improve the labeling accuracy on the borders of the objects.

4.6 Conclusions

In this chapter we explored a family of architectures that performs the structured prediction problem of dense image labeling by learning a deep joint input-output model that (iteratively) improves some initial estimates of the output labels. In this context our main focus was on what is the optimal architecture for implementing this deep model. We argued that the prior approaches of directly predicting the new labels with a feed-forward deep neural networks are sub-optimal and we proposed to decompose the label improvement step in three sub-tasks: (1) detection of the incorrect input labels, 2) their replacement with new labels, and 3) the overall refinement of the output labels in the form of residual corrections. All three steps are embedded in a unified architecture, which we call *Detect + Replace + Refine*, that is end-to-end trainable. We evaluated our architecture in the disparity estimation (stereo matching) task and we report state-of-the-art results in the KITTI 2015 test set. We

also performed preliminary experiments in the semantic segmentation tasks and we report some very promising results.

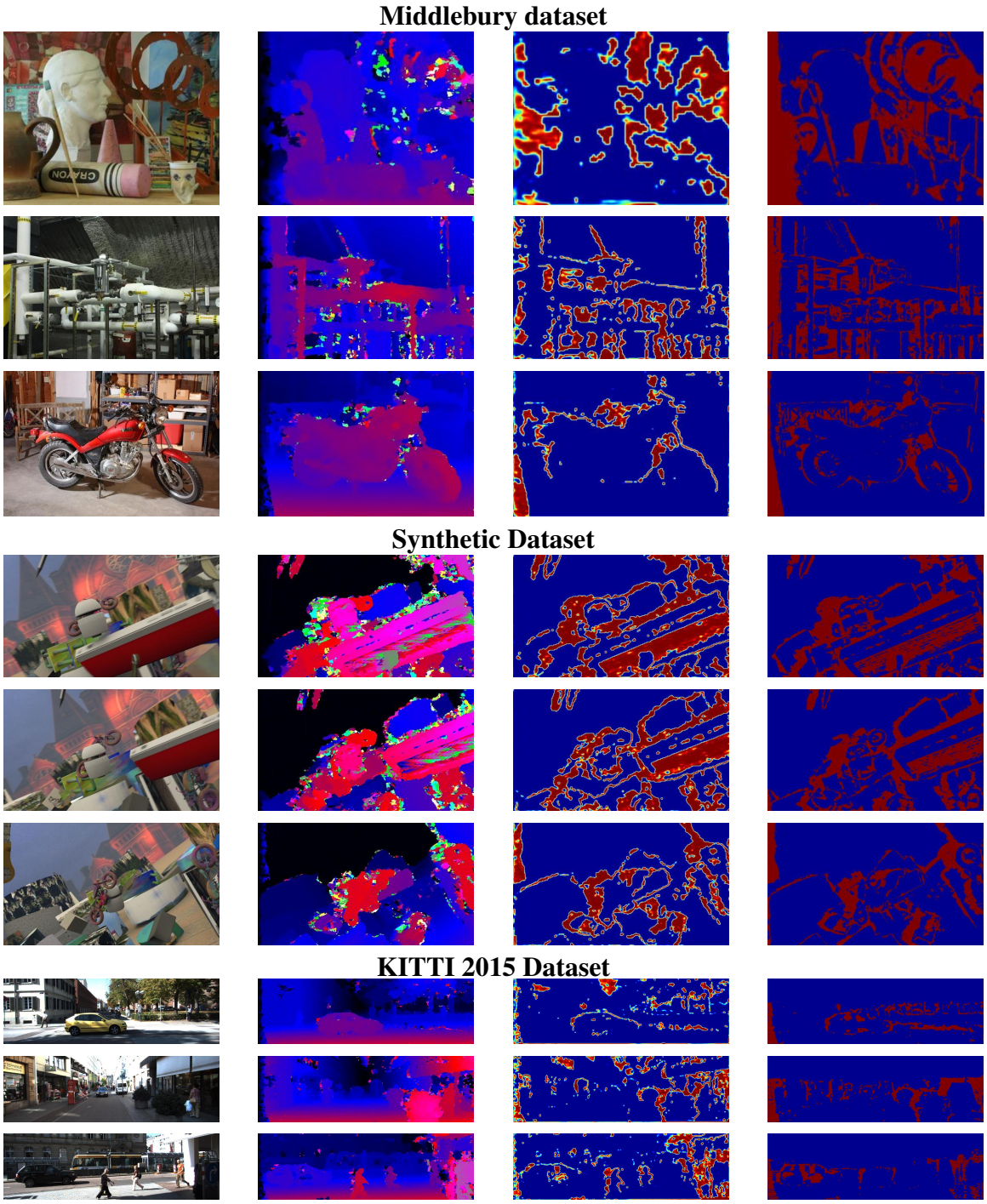
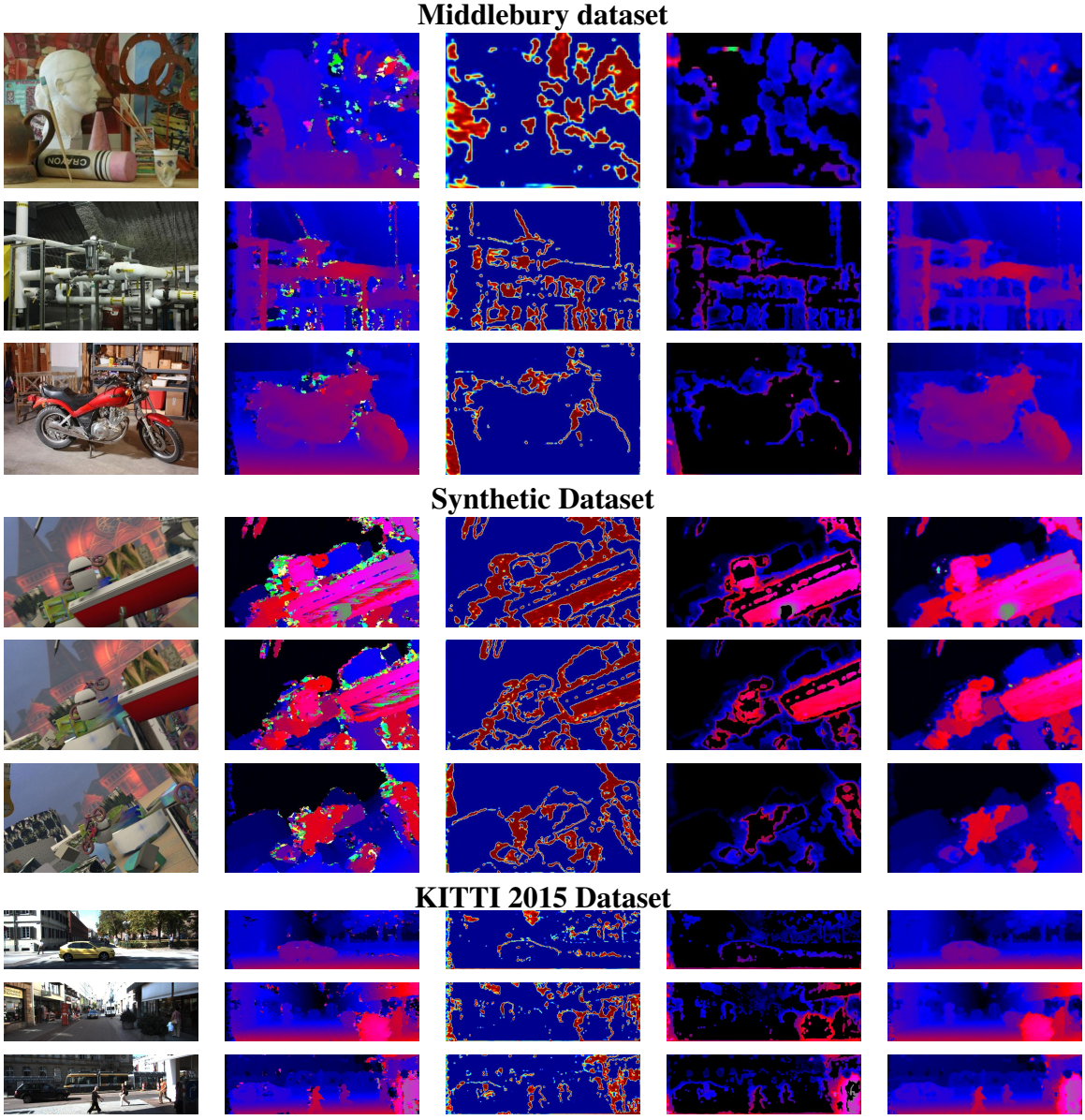
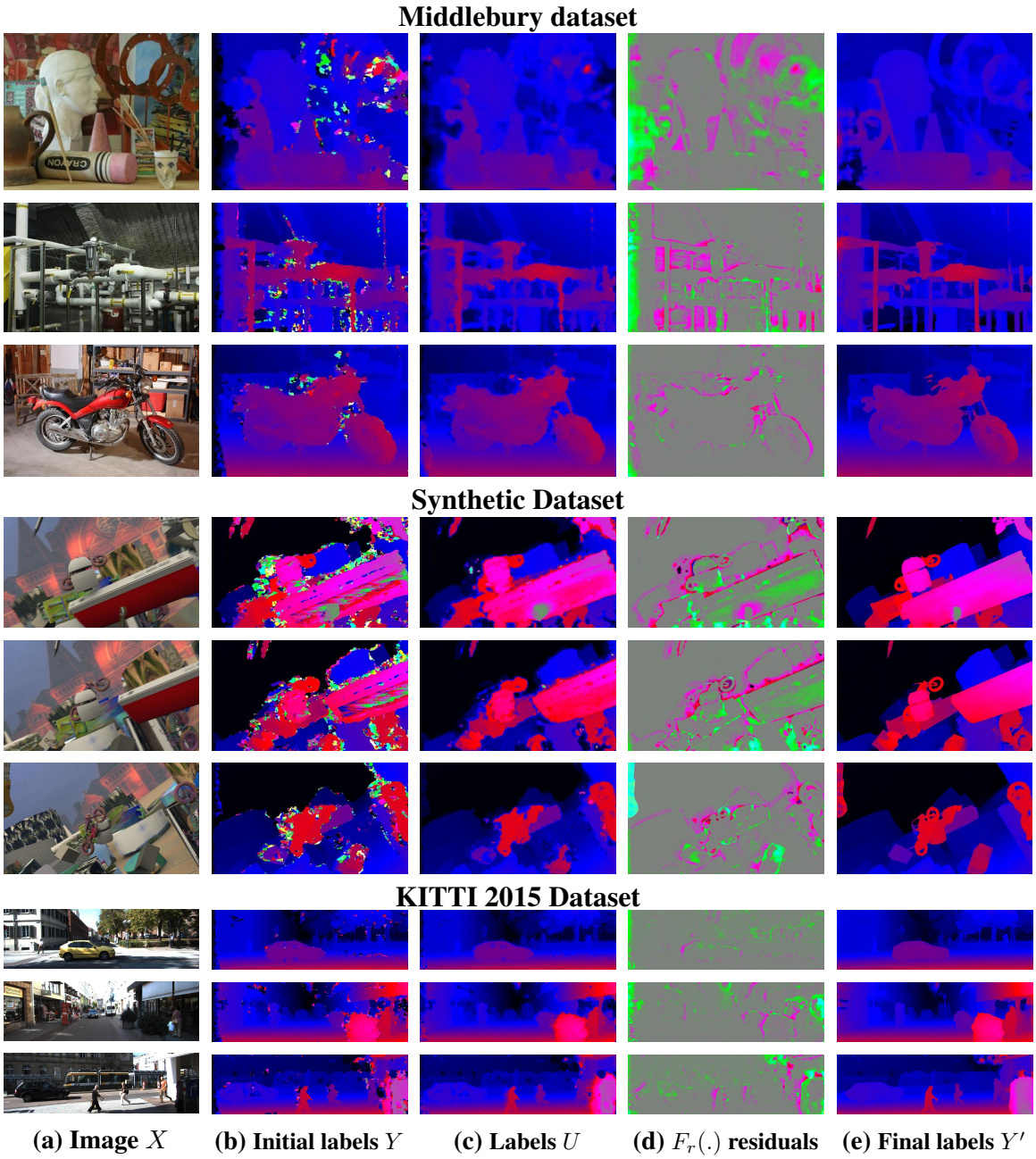


Figure 4-6: Illustration of the error probability maps E that the error detection component $F_e(X, Y)$ yields. The ground truth error maps are computed by thresholding the absolute difference of the initial labels Y from the ground truth labels with a threshold of 3 pixels (red are the erroneous pixel labels). Note that in the case of the KITTI 2015 dataset, the available ground truth labels are sparse and do not cover the entire image (e.g., usually there is no annotation for the sky), which is why some obviously erroneous initial label estimates are not coloured as incorrect (with red color) in the ground truth error maps.



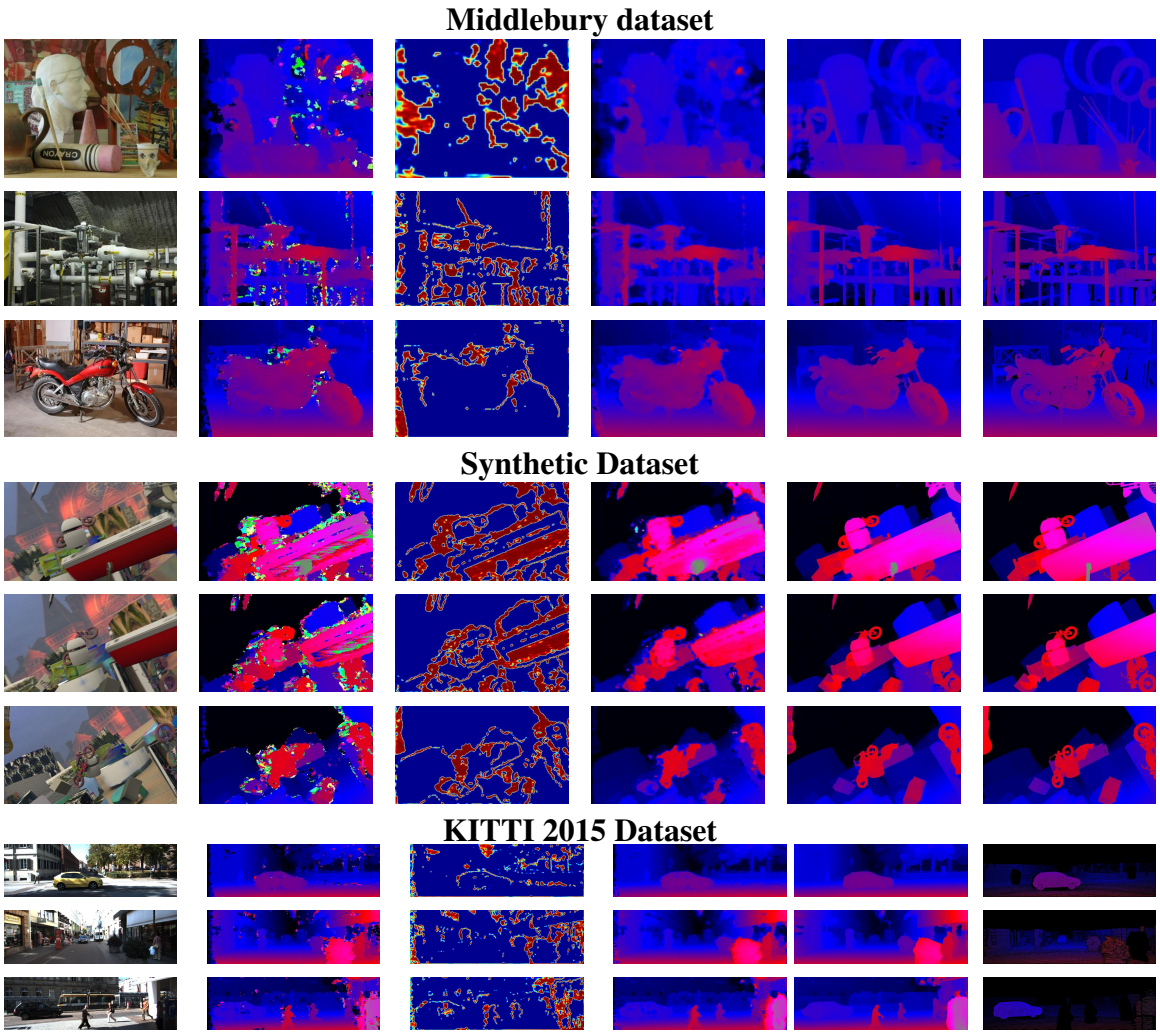
(a) Image X **(b) Initial labels** Y **(c) Error map** E **(d)** $F_u(.)$ **outputs** **(e) New labels** U

Figure 4-7: Here we provide more examples that illustrate the function performed by the Replace step in our proposed architecture. Specifically, sub-figures **(a)**, **(b)**, and **(c)** depict the input image X, the initial disparity label estimates Y, and the error probability map E that the detection component $F_e(.)$ yields for the initial labels Y. In sub-figure **(d)** we depict the label predictions of the replace component $F_u(.)$. For visualization purposes we only depict the $F_u(.)$ pixel predictions that will replace the initial labels that are incorrect (according to the detection component) by drawing the remaining ones (i.e., those whose error probability is less than 0.5) with black color. Finally, in the last sub-figure **(e)** we depict the renewed labels $U = E \odot F_u(X, Y, E) + (1 - E) \odot Y$. We can readily observe that most of the "hard" mistakes of the initial labels Y have now been crudely "fixed" by the Replace component.



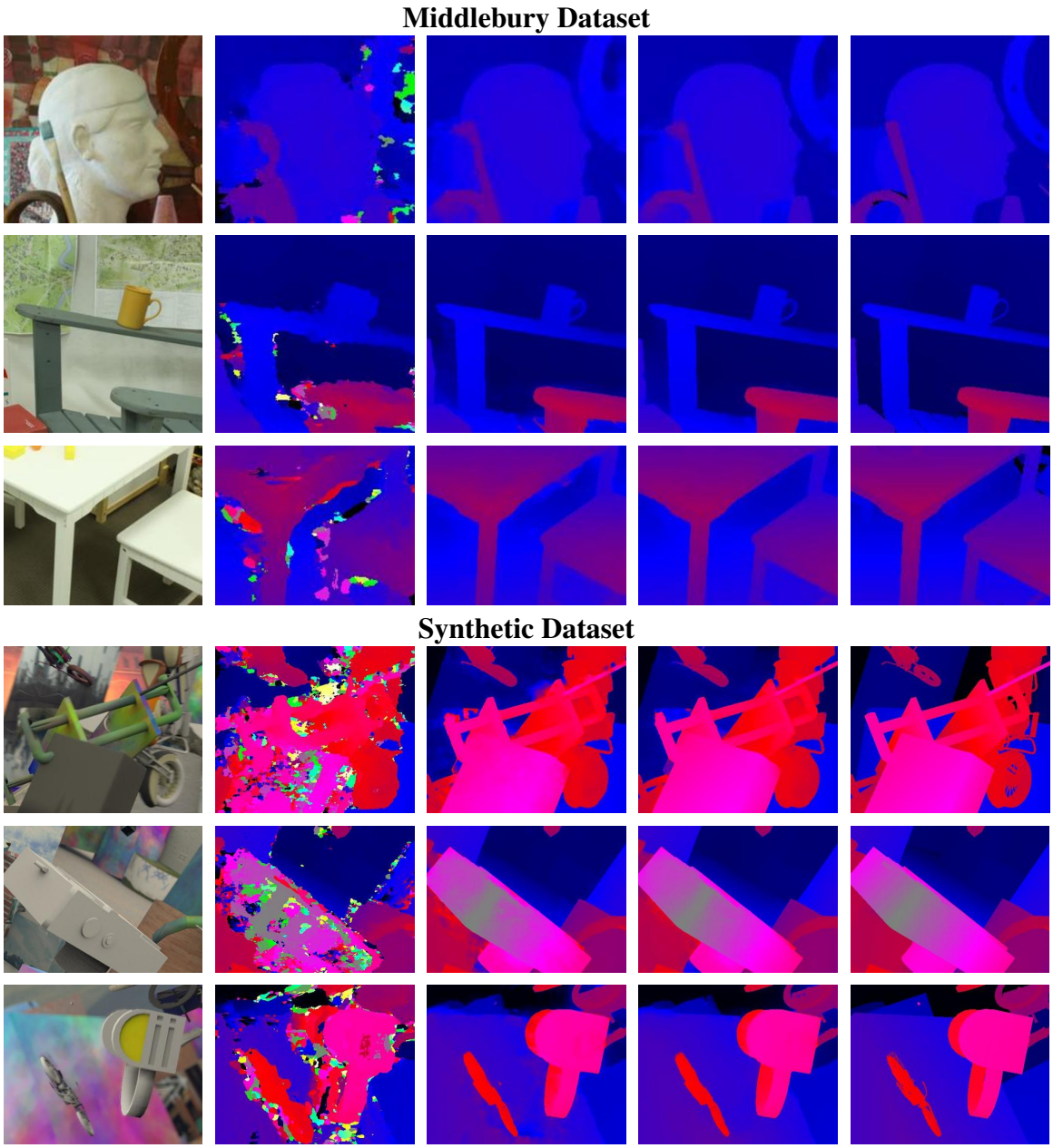
(a) Image X **(b) Initial labels** Y **(c) Labels** U **(d)** $F_r(.)$ **residuals** **(e) Final labels** Y'

Figure 4-8: Here we provide more examples that illustrate the function performed by the Refine step in our proposed architecture. Specifically, in sub-figures **(a)**, **(b)**, and **(c)** we depict the input image X, the initial disparity label estimates Y, and the renewed labels U that the Replace step yields. In sub-figure **(d)** we depict the residual corrections that the Refine component $F_r(.)$ yields for the renewed labels U. Finally, in the last sub-figure **(e)** we depict the final label estimates $Y' = U + F_r(X, Y, E, U)$ that the Refine step yields.



(a) Image X **(b)** Init. labels Y **(c)** Errors E **(d)** Labels U **(e)** Final labels Y' **(f)** Targets

Figure 4-9: Illustration of the intermediate steps of the *Detect + Replace + Refine* work-flow. We observe that the final Refine component $F_r(.)$, by predicting residual corrections, manages to refine the renewed labels U and align the output labels Y' with the fine image structures in image X. Note that in the case of the KITTI 2015 dataset, the available ground truth labels are sparse and do not cover the entire image.



(a) Image X **(b) Initial labels Y** **(c) 1st iter. labels** **(d) 2nd iter. labels** **(e) Target labels**

Figure 4-10: Illustration of the estimated labels on each iteration of the *Detect, Replace, Refine x2* multi-iteration architecture. The visualised examples are from zoomed-in patches from the Middlebury and the Synthetic datasets.

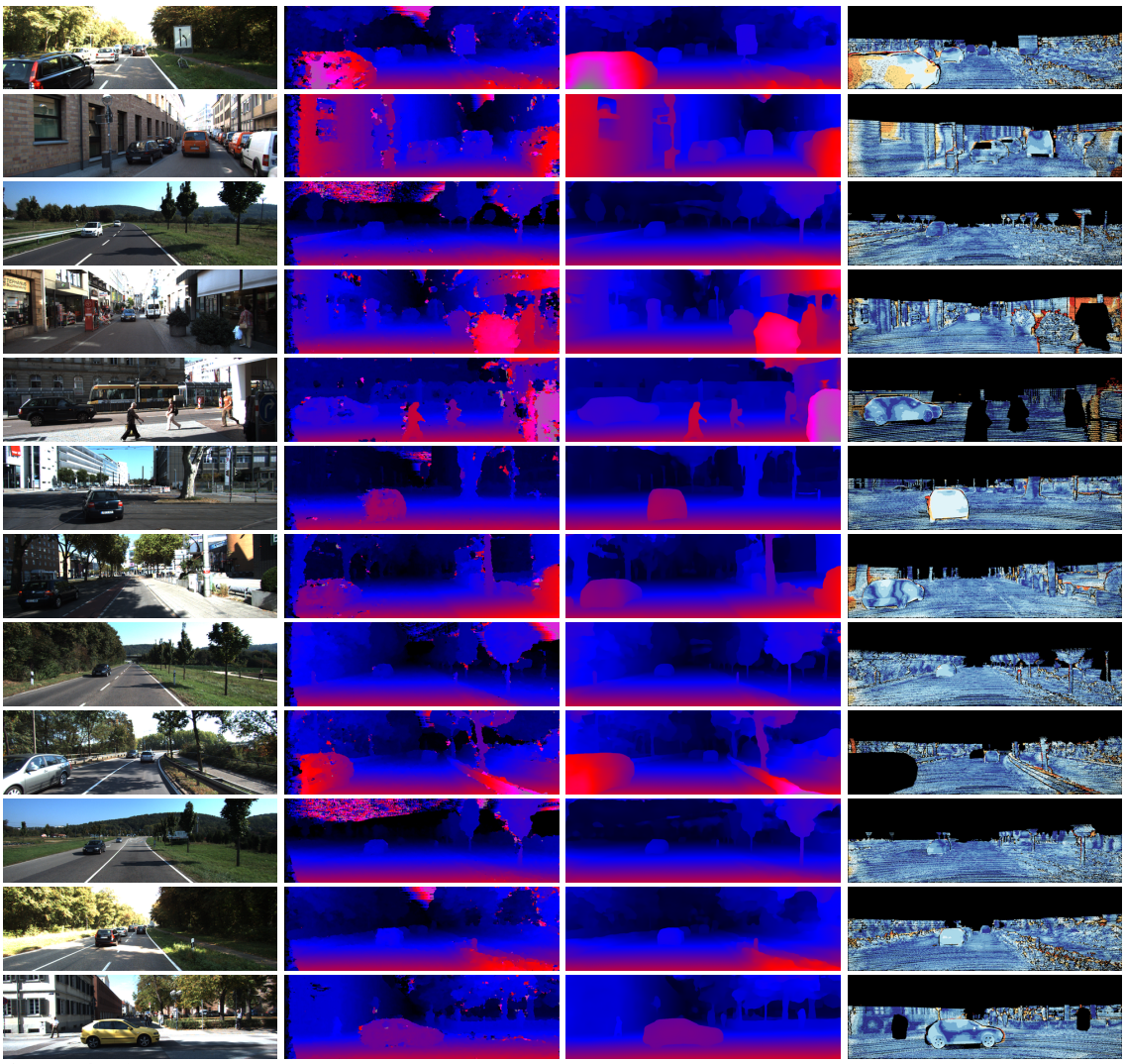


Figure 4-11: Qualitative results in the validation set of KITTI 2015. From left to right, we depict the left image X, the initial labels Y, the labels Y' that our model estimates, and finally the errors of our estimates w.r.t. ground truth.

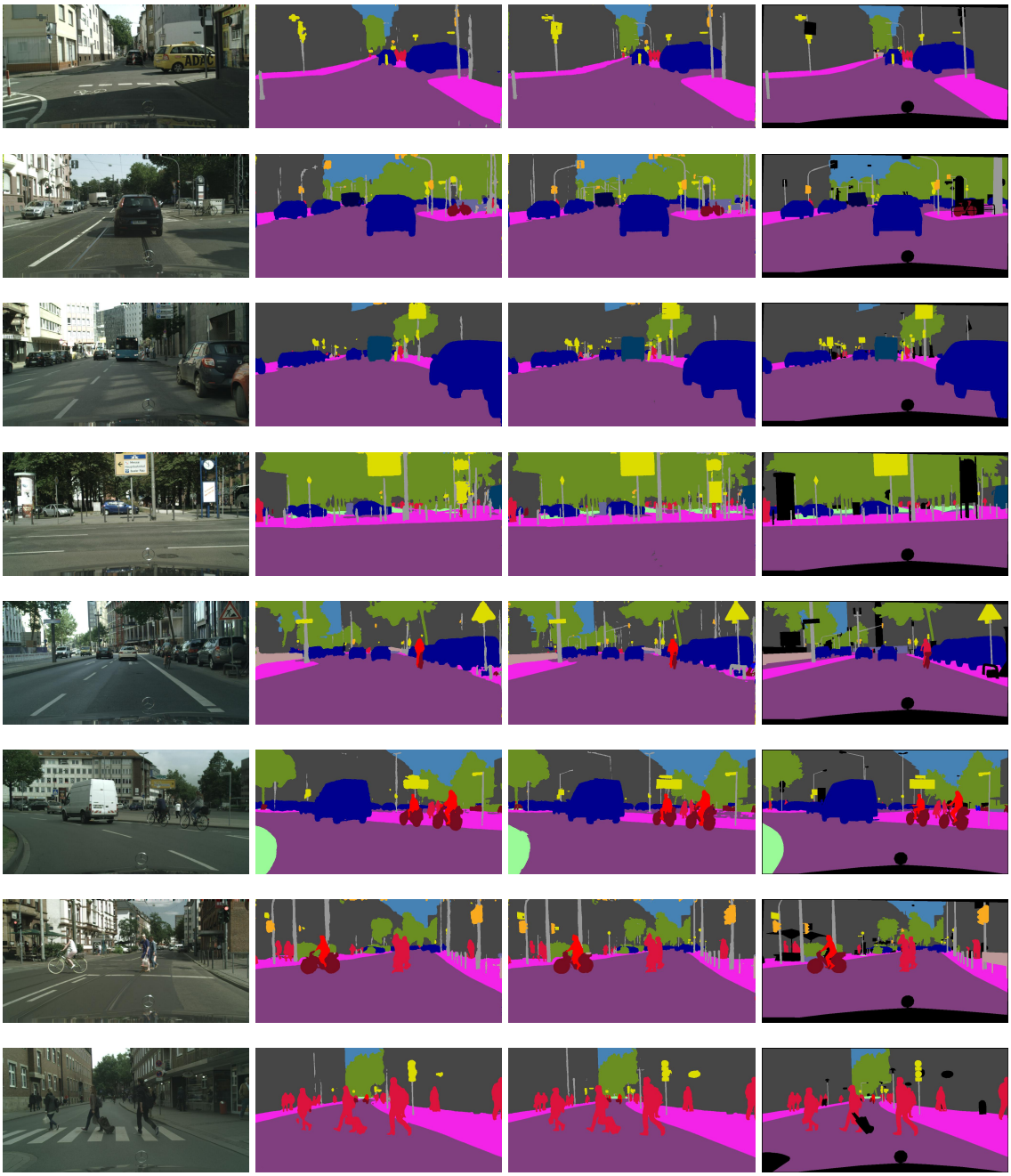


Figure 4-12: Qualitative results in the validation set of Cityscapes dataset. From left to right, we depict the input image X, the initial labels Y, the refined labels Y' that our model estimates, and finally the ground truth labels. Note that the black image regions in the ground truth labels correspond to the unknown category. Those "unknown" image regions are ignored during the evaluation of the segmentation performance.

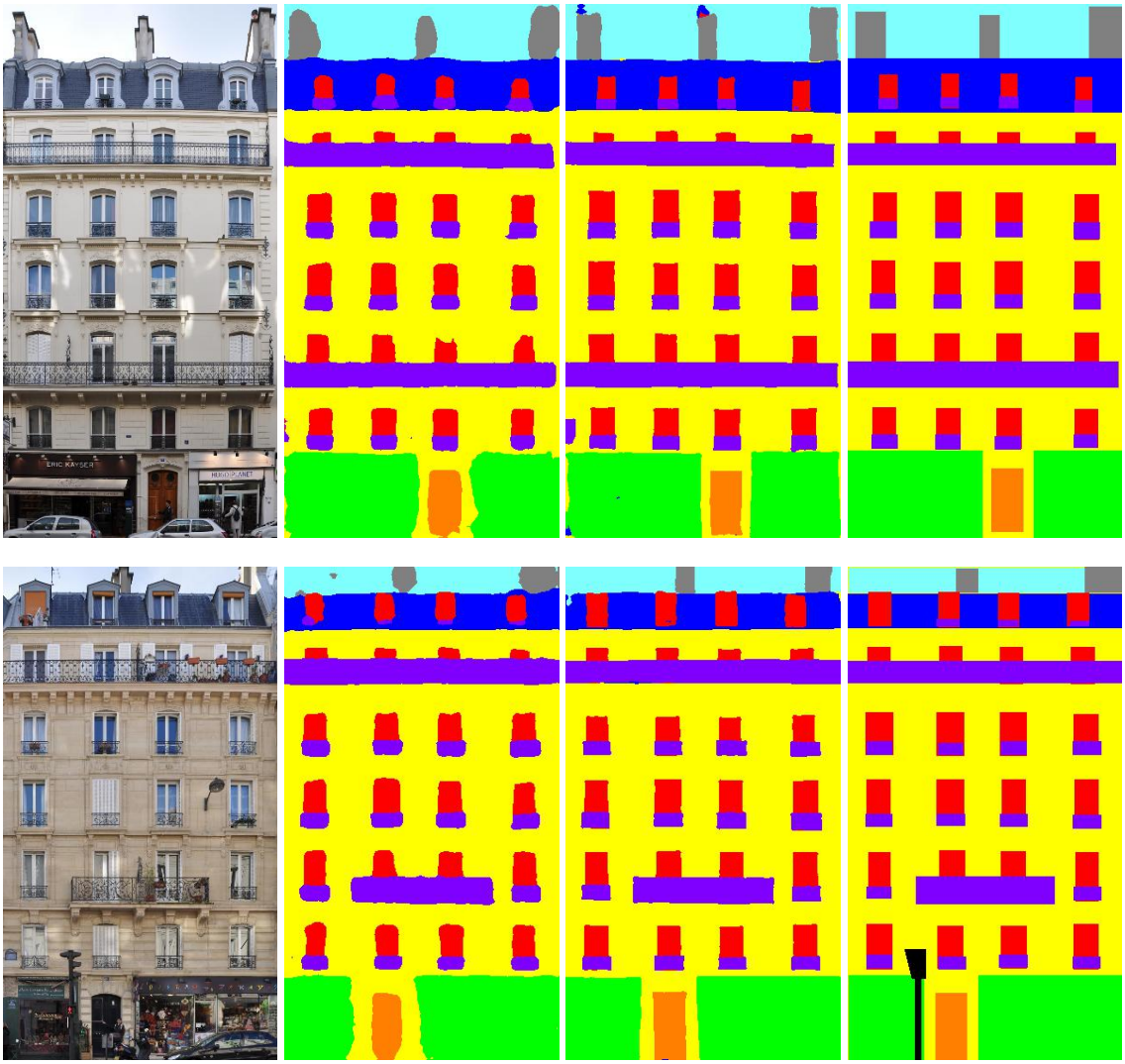


Figure 4-13: Qualitative results in the Facade parsing dataset. From left to right, we depict the input image X, the initial labels Y, the refined labels Y' that our model estimates, and finally the ground truth labels.

Part 2

Annotation Deep Learning for Image Understanding

Chapter 5

Unsupervised Visual Representation Learning

5.1 Introduction

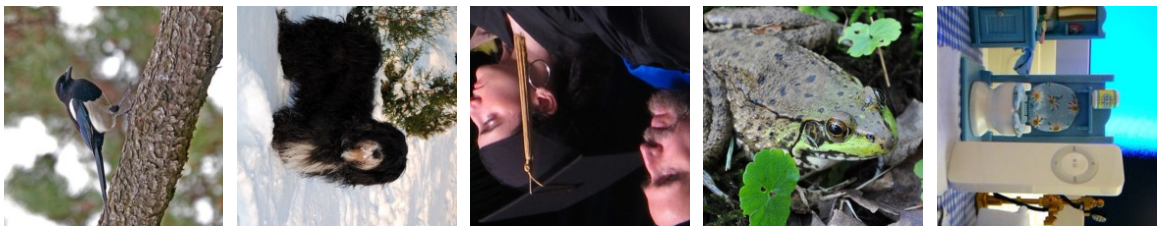
The subject of this chapter is unsupervised visual representation learning, i.e., learning high level ConvNet based representations in an unsupervised manner that avoids manual annotation of visual data. Lately, there is an increased interest for this problem due to the desire for a more annotation efficient learning of ConvNet based image understanding models, which is also one of the main goals of this thesis.

Among the various approaches for unsupervised feature learning, a prominent paradigm is the so-called *self-supervised learning* that defines an annotation free pretext task, using only the visual information present on the images or videos, in order to provide a surrogate supervision signal for feature learning. For example, in order to learn features, [170] and [81] train ConvNets to colorize gray scale images, [24] and [111] predict the relative position of image patches, and [1] predict the egomotion (i.e., self-motion) of a moving vehicle between two consecutive frames. The rationale behind such self-supervised tasks is that solving them will force the ConvNet to learn semantic image features that can be useful for other vision tasks. In fact, image representations learned with the above self-supervised tasks, although they have not managed to match the performance of supervised-learned representations, they have proved to be good alternatives for transferring on other vision

tasks, such as object recognition, object detection, and semantic segmentation [170, 81, 171, 82, 24, 111, 112, 117, 25]. Other successful cases of unsupervised feature learning are clustering based methods [28, 89, 159], reconstruction based methods [6, 65, 99], and methods that involve learning generative probabilistic models [45, 26, 123].

Our work follows the self-supervised paradigm and proposes to learn image representations by training ConvNets to recognize the geometric transformation that is applied to the image that it gets as input. More specifically, we first define a small set of discrete geometric transformations, then each of those geometric transformations is applied to each image in the dataset and the produced transformed images are fed to the ConvNet model that is trained to recognize the transformation of each image. In this formulation, it is the set of geometric transformations that actually defines the classification pretext task that the ConvNet model has to learn. Therefore, in order to achieve unsupervised semantic feature learning, it is of crucial importance to properly choose those geometric transformations (we further discuss this aspect of our methodology in section 5.2.2). What we propose is to define the geometric transformations as the image rotations by 0, 90, 180, and 270 degrees. Thus, the ConvNet model is trained on the 4-way image classification task of recognizing one of the four image rotations (see Figure 5-2). We argue that in order for a ConvNet model to be able recognize the rotation transformation that was applied to an image it will require to understand the concept of the objects depicted in the image (see Figure 5-1), such as their location in the image, their type, and their pose. Throughout this chapter we support that argument both qualitatively and quantitatively. Furthermore we demonstrate in the experimental section of this chapter that despite the simplicity of our self-supervised approach, the task of predicting rotation transformations provides a powerful surrogate supervision signal for feature learning and leads to significant improvements on the relevant benchmarks.

Note that our self-supervised task is different from the work of Dosovitskiy et al. [28] and Agrawal et al. [1] that also involve geometric transformations. Dosovitskiy et al. [28] train a ConvNet model to yield representations that are discriminative between images and at the same time invariant on geometric and chromatic transformations. In contrast, we train a ConvNet model to recognize the geometric transformation applied to an image. It is also



90° rotation 270° rotation 180° rotation 0° rotation 270° rotation

Figure 5-1: Images rotated by random multiples of 90 degrees (e.g., 0, 90, 180, or 270 degrees). The core intuition of our self-supervised feature learning approach is that if someone is not aware of the concepts of the objects depicted in the images, he cannot recognize the rotation that was applied to them.

fundamentally different from the egomotion method of Agrawal et al. [1], which employs a ConvNet model with siamese like architecture that takes as input two consecutive video frames and is trained to predict (through regression) their camera transformation. Instead, in our approach, the ConvNet takes as input a single image to which we have applied a random geometric transformation (i.e., rotation) and is trained to recognize (through classification) this geometric transformation without having access to the initial image.

To summarize, the contribution of the work presented in this chapter are as follows:

- We propose a new self-supervised task that is very simple and at the same time, as we demonstrate throughout this chapter, offers a powerful supervisory signal for semantic feature learning.
- We exhaustively evaluate our self-supervised method under various settings (e.g. semi-supervised or transfer learning settings) and in various vision tasks (i.e., CIFAR-10, ImageNet, Places, and PASCAL classification, detection, or segmentation tasks).
- In all of them, our novel self-supervised formulation demonstrates state-of-the-art results with dramatic improvements w.r.t. prior unsupervised approaches.
- As a consequence we show that for several important vision tasks, our self-supervised learning approach significantly narrows the gap between unsupervised and supervised feature learning.

In the following sections, we describe our self-supervised methodology in §2, we provide experimental results in §3, and finally we conclude in §4.

5.2 Methodology

5.2.1 Overview

The goal of our work is to learn ConvNet based semantic features in an unsupervised manner. To achieve that goal we propose to train a ConvNet model $F(.)$ to estimate the geometric transformation applied to an image that is given to it as input. Specifically, we define a set of K discrete geometric transformations $G = \{g(.|y)\}_{y=1}^{K}$, where $g(.|y)$ is the operator that applies to image X the geometric transformation with label y that yields the transformed image $X^y = g(X|y)$. The ConvNet model $F(.)$ gets as input an image X^{y^*} (where the label y^* is unknown to model $F(.)$) and yields as output a probability distribution over all possible geometric transformations:

$$F(X^{y*}|\theta) = \{F^y(X^{y*}|\theta)\}_{y=1}^{K}, \tag{5.1}$$

where $F^y(X^{y*}|\theta)$ is the predicted probability for the geometric transformation with label y and θ are the learnable parameters of model $F(.)$.

Therefore, given a set of N training images $D = \{X_i\}_{i=1}^{N}$, the self-supervised training objective that the ConvNet model must learn to solve is:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} loss(X_i, \theta), \tag{5.2}$$

where the loss function $loss(.)$ is defined as:

$$loss(X_i, \theta) = -\frac{1}{K} \sum_{y=1}^{K} \log(F^y(g(X_i|y)|\theta)). \tag{5.3}$$

In the following subsection we describe the type of geometric transformations that we propose in our work.

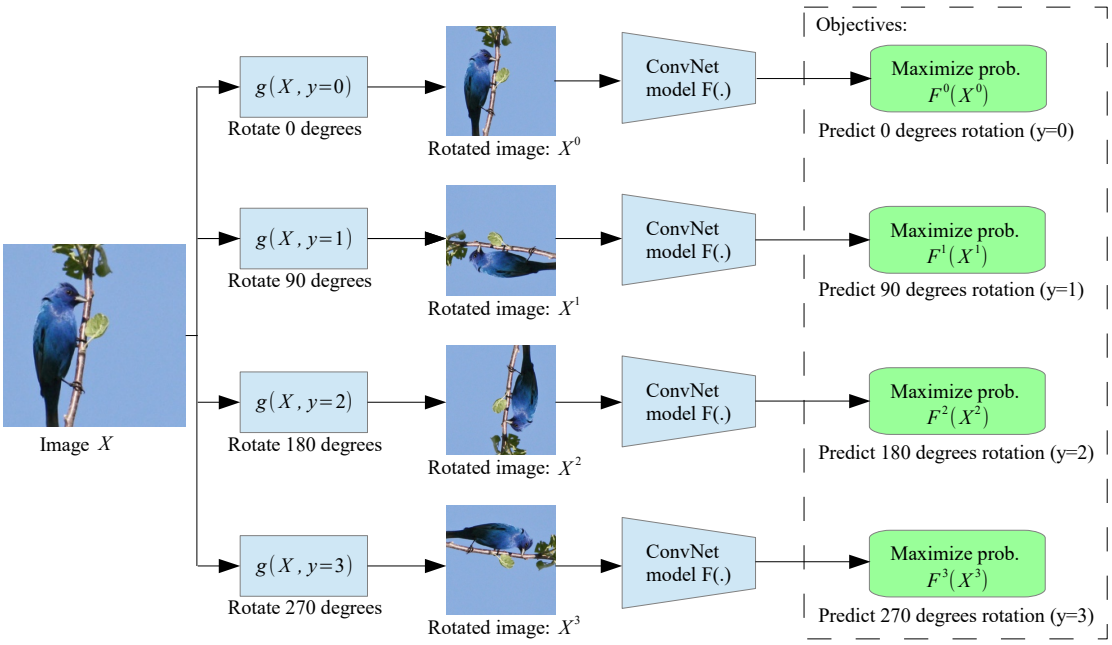


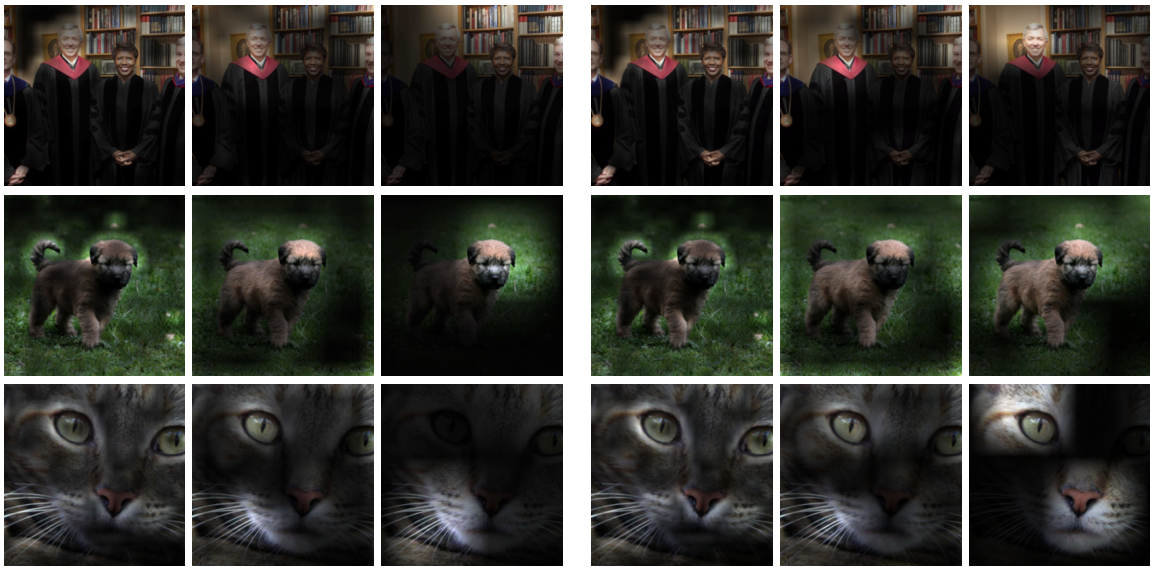
Figure 5-2: Illustration of the self-supervised task that we propose for semantic feature learning. Given four possible geometric transformations, the 0, 90, 180, and 270 degrees rotations, we train a ConvNet model $F(.)$ to recognize the rotation that is applied to the image that it gets as input. $F^y(X^{y^*})$ is the probability of rotation transformation y predicted by model $F(.)$ when it gets as input an image that has been transformed by the rotation transformation y^*.

5.2.2 Choosing geometric transformations: image rotations

In the above formulation, the geometric transformations G must define a classification task that should force the ConvNet model to learn semantic features useful for visual perception tasks (e.g., object detection or image classification). In our work we propose to define the set of geometric transformations G as all the image rotations by multiples of 90 degrees, i.e., 2d image rotations by 0, 90, 180, and 270 degrees (see Figure 5-2). More formally, if $Rot(X, \phi)$ is an operator that rotates image X by ϕ degrees, then our set of geometric transformations consists of the $K = 4$ image rotations $G = \{g(X|y)\}_{y=1}^{4}$, where $g(X|y) = Rot(X, (y-1)90)$.

Forcing the learning of semantic features: The core intuition behind using these image rotations as the set of geometric transformations relates to the simple fact that it is essentially impossible for a ConvNet model to effectively perform the above rotation recognition task unless it has first learnt to recognize and detect classes of objects as well

Input images on the models

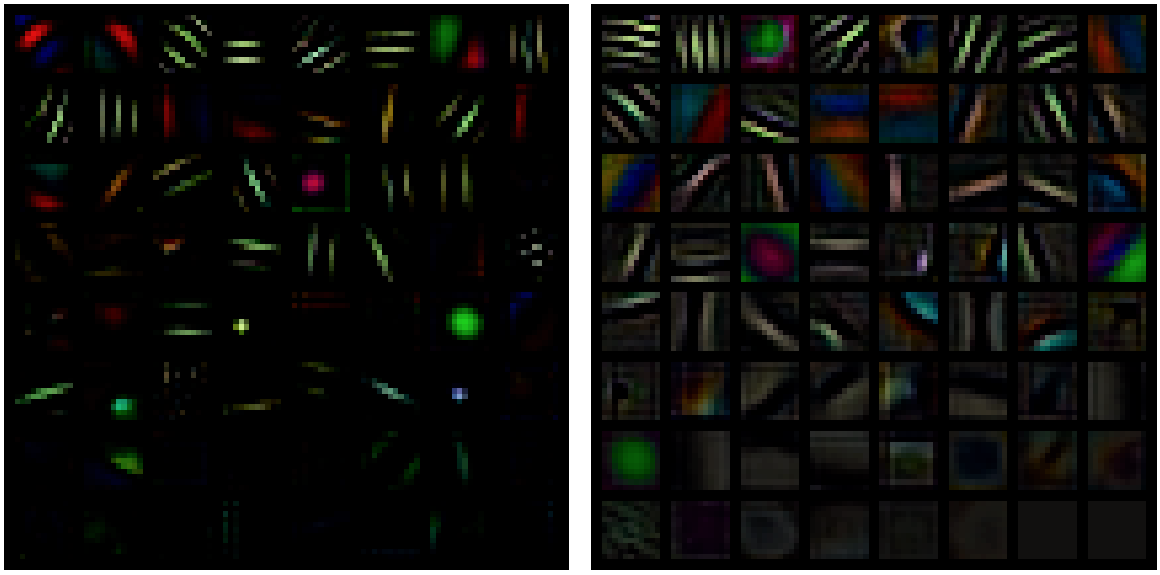


Conv1 27×27 Conv3 13×13 Conv5 6×6 Conv1 27×27 Conv3 13×13 Conv5 6×6

(a) Attention maps of supervised model **(b) Attention maps of self-supervised model**

Figure 5-3: Attention maps generated by an AlexNet model trained **(a)** to recognize objects (supervised), and **(b)** to recognize image rotations (self-supervised). In order to generate the attention map of a conv. layer we first compute the feature maps of this layer, then we raise each feature activation on the power p, and finally we sum the activations at each location of the feature map. For the conv. layers 1, 2, and 3 we used the powers $p = 1$, $p = 2$, and $p = 4$ respectively. For visualization of our self-supervised model's attention maps for all the rotated versions of the images see Figure 5-6.

as their semantic parts in images. More specifically, to successfully predict the rotation of an image the ConvNet model must necessarily learn to localize salient objects in the image, recognize their orientation and object type, and then relate the object orientation with the dominant orientation that each type of object tends to be depicted within the available images. In Figure 5-3b we visualize some attention maps generated by a model trained on the rotation recognition task. These attention maps are computed based on the magnitude of activations at each spatial cell of a convolutional layer and essentially reflect where the network puts most of its focus in order to classify an input image. We observe, indeed, that in order for the model to accomplish the rotation prediction task it learns to focus on high



(a) Supervised model **(b) Our self-supervised model**

Figure 5-4: First layer filters learned by a AlexNet model trained on **(a)** the supervised object recognition task and **(b)** the self-supervised task of recognizing rotated images. We observe that the filters learned by the self-supervised task are mostly oriented edge filters on various frequencies and, remarkably, they seem to have more variety than those learned on the supervised task.

level object parts in the image, such as eyes, nose, tails, and heads. By comparing them with the attention maps generated by a model trained on the object recognition task in a supervised way (see Figure 5-3a) we observe that both models seem to focus on roughly the same image regions.

In Figure 5-6 we visualize the attention maps for all the rotated copies of the images. We observe that the attention maps of all the rotated copies of an image are roughly the same, i.e., the attention maps are equivariant w.r.t. the image rotations. This practically means that in order to accomplish the rotation prediction task the network focuses on the same object parts regardless of the image rotation. Furthermore, in Figure 5-4 we visualize the first layer filters that were learnt by an AlexNet model trained on the proposed rotation recognition task. As can be seen, they appear to have a big variety of edge filters on multiple orientations and multiple frequencies. Remarkably, these filters seem to have a greater amount of variety even than the filters learnt by the supervised object recognition task.

Absence of low-level visual artifacts: An additional important advantage of using image rotations by multiples of 90 degrees over other geometric transformations, is that they

can be implemented by flip and transpose operations (as we will see below) that do not leave any easily detectable low-level visual artifacts that will lead the ConvNet to learn trivial features with no practical value for the vision perception tasks. In contrast, had we decided to use as geometric transformations, e.g., scale and aspect ratio image transformations, in order to implement them we would need to use image resizing routines that leave easily detectable image artifacts.

Well-posedness: Furthermore, human captured images tend to depict objects in an "up-standing" position, thus making the rotation recognition task well defined, i.e., given an image rotated by 0, 90, 180, or 270 degrees, there is usually no ambiguity of what is the rotation transformation (with the exception of images that only depict round objects). In contrast, that is not the case for the object scale that varies significantly on human captured images.

Implementing image rotations: In order to implement the image rotations by 90, 180, and 270 degrees (the 0 degrees case is the image itself), we use flip and transpose operations. Specifically, for 90 degrees rotation we first transpose the image and then flip it vertically (upside-down flip), for 180 degrees rotation we flip the image first vertically and then horizontally (left-right flip), and finally for 270 degrees rotation we first flip vertically the image and then we transpose it.

5.2.3 Discussion

The simple formulation of our self-supervised task has several advantages. It has the same computational cost as supervised learning, similar training convergence speed (that is significantly faster than image reconstruction based approaches; our AlexNet model trains in around 2 days using a single Titan X GPU), and can trivially adopt the efficient parallelization schemes devised for supervised learning [46], making it an ideal candidate for unsupervised learning on internet-scale data (i.e., billions of images). Furthermore, our approach does not require any special image pre-processing routine in order to avoid learning trivial features, as many other unsupervised or self-supervised approaches do. Despite the simplicity of our self-supervised formulation, as we will see in the experimental section of

this chapter, the features learned by our approach achieve dramatic improvements on the unsupervised feature learning benchmarks.

5.3 Experimental Results

In this section we conduct an extensive evaluation of our approach on the most commonly used image datasets, such as CIFAR-10 [77], ImageNet [129], PASCAL [31], and Places205 [175], as well as on various vision tasks, such as object detection, object segmentation, and image classification. We also consider several learning scenarios, including transfer learning and semi-supervised learning. In all cases, we compare our approach with corresponding state-of-the-art methods.

5.3.1 CIFAR experiments

We start by evaluating on the object recognition task of CIFAR-10 the ConvNet based features learned by the proposed self-supervised task of rotation recognition. We will here after call a ConvNet model that is trained on the self-supervised task of rotation recognition *RotNet* model.

Implementation details: In our CIFAR-10 experiments we implement the *RotNet* models with Network-In-Network (NIN) architectures [90]. In order to train them on the rotation prediction task, we use SGD with batch size 128, momentum 0.9, weight decay $5e-4$ and lr of 0.1. We drop the learning rates by a factor of 5 after epochs 30, 60, and 80. We train in total for 100 epochs. In our preliminary experiments we found that we get significant improvement when during training we train the network by feeding it all the four rotated copies of an image simultaneously instead of each time randomly sampling a single rotation transformation. Therefore, at each training batch the network sees 4 times more images than the batch size.

Evaluation of the learned feature hierarchies: First, we explore how the quality of the learned features depends on their depth (i.e., the depth of the layer that they come from) as well as from the total depth of the *RotNet* model. For that purpose, we first train using the CIFAR-10 training images three *RotNet* models which have 3, 4, and 5 convolutional blocks

Model	ConvB1	ConvB2	ConvB3	ConvB4	ConvB5
RotNet with 3 conv. blocks	85.45	88.26	62.09	-	-
RotNet with 4 conv. blocks	85.07	89.06	86.21	61.73	-
RotNet with 5 conv. blocks	85.04	**89.76**	86.82	74.50	50.37

Table 5.1: Evaluation of the unsupervised learned features by measuring the classification accuracy that they achieve when we train a non-linear object classifier on top of them. The reported results are from CIFAR-10. The size of the ConvB1 feature maps is $96 \times 16 \times 16$ and the size of the other feature maps is $192 \times 8 \times 8$.

respectively (note that each conv. block in the NIN architectures that implement our *RotNet* models have 3 conv. layers; therefore, the total number of conv. layers of the examined *RotNet* models is 9, 12, and 15 for 3, 4, and 5 conv. blocks respectively). Afterwards, we learn classifiers on top of the feature maps generated by each conv. block of each *RotNet* model. Those classifiers are trained in a supervised way on the object recognition task of CIFAR-10. They consist of 3 fully connected layers; the 2 hidden layers have 200 feature channels each and are followed by batch-norm and relu units. We report the accuracy results of CIFAR-10 test set in Table 5.1. We observe that in all cases the feature maps generated by the 2nd conv. block (that actually has depth 6 in terms of the total number of conv. layer till that point) achieve the highest accuracy, i.e., between 88.26% and 89.06%. The features of the conv. blocks that follow the 2nd one gradually degrade the object recognition accuracy, which we assume is because they start becoming more and more specific on the self-supervised task of rotation prediction. Also, we observe that increasing the total depth of the RotNet models leads to increased object recognition performance by the feature maps generated by earlier layers (and after the 1st conv. block). We assume that this is because increasing the depth of the model and thus the complexity of its head (i.e., top ConvNet layers) allows the features of earlier layers to be less specific to the rotation prediction task.

Exploring the quality of the learned features w.r.t. the number of recognized rotations: In Table 5.2 we explore how the quality of the self-supervised features depends on the number of discrete rotations used in the rotation prediction task. For that purpose we defined three extra rotation recognition tasks: (a) one with 8 rotations that includes all the multiples of 45 degrees, (b) one with only the 0° and 180° rotations, and (c) one with only the 90° and 270° rotations. In order to implement the rotation transformation of the

# Rotations	Rotations	Classification Accuracy
4	$0°$, $90°$, $180°$, $270°$	**89.06**
8	$0°$, $45°$, $90°$, $135°$, $180°$, $225°$, $270°$, $315°$	88.51
2	$0°$, $180°$	87.46
2	$90°$, $270°$	85.52

Table 5.2: Exploring the quality of the self-supervised learned features w.r.t. the number of recognized rotations. For all the entries we trained a non-linear classifier with 3 fully connected layers (similar to Table 5.1) on top of the feature maps generated by the 2nd conv. block of a RotNet model with 4 conv. blocks in total. The reported results are from CIFAR-10.

$45°$, $135°$, $225°$, $270°$, and $315°$ rotations (in the 8 discrete rotations case) we used an image wrapping routine and then we took care to crop only the central square image regions that do not include any of the empty image areas introduced by the rotation transformations (and which can easily indicate the image rotation). We observe that indeed for 4 discrete rotations (as we proposed) we achieve better object recognition performance than the 8 or 2 cases. We believe that this is because the 2 orientations case offers too few classes for recognition (i.e., less supervisory information is provided) while in the 8 orientations case the geometric transformations are not distinguishable enough and furthermore the 4 extra rotations introduced may lead to visual artifacts on the rotated images. Moreover, we observe that among the RotNet models trained with 2 discrete rotations, the RotNet model trained with $90°$ and $270°$ rotations achieves worse object recognition performance than the model trained with the $0°$ and $180°$ rotations, which is probably due to the fact that the former model does not "see" during the unsupervised phase the $0°$ rotation that is typically used during the object recognition training phase.

Comparison against supervised and other unsupervised methods: In Table 5.3 we compare our unsupervised learned features against other unsupervised (or hand-crafted) features on CIFAR-10. For our entries we use the feature maps generated by the 2nd conv. block of a RotNet model with 4 conv. blocks in total. On top of those RotNet features we train 2 different classifiers: (a) a non-linear classifier with 3 fully connected layers as before (entry *(Ours) RotNet + non-linear*), and (b) three conv. layers plus a linear prediction layer (entry *(Ours) RotNet +conv.*; note that this entry is basically a 3 blocks NIN model with the first 2 blocks coming from a RotNet model and the 3rd being randomly initialized and

Method	Classification Accuracy
Supervised NIN	92.80
Random Init. + conv	72.50
(Ours) RotNet + non-linear	89.06
(Ours) RotNet + conv	**91.16**
(Ours) RotNet + non-linear (fine-tuned)	91.73
(Ours) RotNet + conv (fine-tuned)	92.17
Roto-Scat + SVM [115]	82.3
ExemplarCNN [28]	84.3
DCGAN [123]	82.8
Scattering [114]	84.7

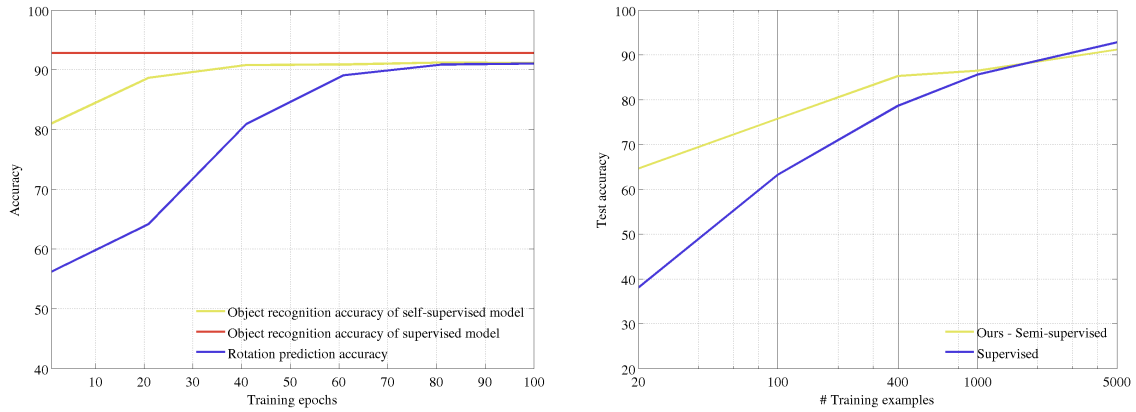
Table 5.3: Evaluation of unsupervised feature learning methods on CIFAR-10. The *Supervised NIN* and the *(Ours) RotNet + conv* entries have exactly the same architecture but the first was trained fully supervised while on the second the first 2 conv. blocks were trained unsupervised with our rotation prediction task and the 3rd block only was trained in a supervised manner. In the *Random Init. + conv* entry a conv. classifier (similar to that of *(Ours) RotNet + conv*) is trained on top of two NIN conv. blocks that are randomly initialized and stay frozen. Note that each of the prior approaches has a different ConvNet architecture and thus the comparison with them is just indicative.

Classes	aero	car	bird	cat	deer	dog	frog	horse	ship	truck
Supervised NIN	**93.7**	**96.3**	**89.4**	82.4	**93.6**	**89.7**	**95.0**	**94.3**	**95.7**	**95.2**
(Ours) RotNet + conv	91.7	95.8	87.1	**83.5**	91.5	85.3	94.2	91.9	**95.7**	94.2

Table 5.4: Per class CIFAR-10 classification accuracy.

trained on the recognition task). We observe that we improve over the prior unsupervised approaches and we achieve state-of-the-art results in CIFAR-10 (note that each of the prior approaches has a different ConvNet architecture thus the comparison with them is just indicative). More notably, *the accuracy gap between the RotNet based model and the fully supervised NIN model is very small*, only 1.64 percentage points (92.80% vs 91.16%). We provide per class breakdown of the classification accuracy of our unsupervised model as well as the supervised one in Table 5.4. In Table 5.3 we also report the performance of the RotNet features when, instead of being kept frozen, they are fine-tuned during the object recognition training phase. We observe that fine-tuning the unsupervised learned features further improves the classification performance, thus reducing even more the gap with the supervised case.

Correlation between object classification task and rotation prediction task: In Fig-



(a) **(b)**

Figure 5-5: **(a)** Plot with the rotation prediction accuracy and object recognition accuracy as a function of the training epochs used for solving the rotation prediction task. The red curve is the object recognition accuracy of a fully supervised model (a NIN model), which is independent from the training epochs on the rotation prediction task. The yellow curve is the object recognition accuracy of an object classifier trained on top of feature maps learned by a *RotNet* model at different snapshots of the training procedure. **(b)** Accuracy as a function of the number of training examples per category in CIFAR-10. *Ours semi-supervised* is a NIN model whose first 2 conv. blocks are *RotNet* model that was trained in a self-supervised way on the entire training set of CIFAR-10 and the 3rd conv. block along with a prediction linear layer that was trained with the object recognition task only on the available set of labeled images.

ure 5-5a, we plot the object classification accuracy as a function of the training epochs used for solving the self-supervised task of recognizing rotations, which learns the features used by the object classifier. More specifically, in order to create the object recognition accuracy curve, in each training snapshot of RotNet (i.e., every 20 epochs), we pause its training procedure and we train from scratch (until convergence) a non-linear object classifier on top of the so far learnt RotNet features. Therefore, the object recognition accuracy curve depicts the accuracy of those non-linear object classifiers after the end of their training while the rotation prediction accuracy curve depicts the accuracy of the RotNet at those snapshots. We observe that, as the ability of the RotNet features for solving the rotation prediction task improves (i.e., as the rotation prediction accuracy increases), their ability to help solving the object recognition task improves as well (i.e., the object recognition accuracy also increases). Furthermore, we observe that the object recognition accuracy converges fast w.r.t. the number of training epochs used for solving the pretext task of rotation prediction.

Semi-supervised setting: Motivated by the very high performance of our unsupervised feature learning method, we also evaluate it on a semi-supervised setting. More specifically, we first train a 4 block *RotNet* model on the rotation prediction task using the entire image dataset of CIFAR-10 and then we train on top of its feature maps object classifiers using only a subset of the available images and their corresponding labels. As feature maps we use those generated by the 2nd conv. block of the *RotNet* model. As a classifier we use a set of convolutional layers that actually has the same architecture as the 3rd conv. block of a NIN model plus a linear classifier, all randomly initialized. For training the object classifier we use for each category 20, 100, 400, 1000, or 5000 image examples. Note that 5000 image examples is the extreme case of using the entire CIFAR-10 training dataset. Also, we compare our method with a supervised model that is trained only on the available examples each time. In Figure 5-5b we plot the accuracy of the examined models as a function of the available training examples. We observe that our unsupervised trained model exceeds in this semi-supervised setting the supervised model when the number of examples per category drops below 1000. Furthermore, as the number of examples decreases, the performance gap in favor of our method is increased. This empirical evidence demonstrates the usefulness of our method on semi-supervised settings.

5.3.2 Evaluation of self-supervised features trained in ImageNet

Here we evaluate the performance of our self-supervised ConvNet models on the ImageNet, Places, and PASCAL VOC datasets. Specifically, we first train a *RotNet* model on the training images of the ImageNet dataset and then we evaluate the performance of the self-supervised features on the image classification tasks of ImageNet, Places, and PASCAL VOC datasets and on the object detection and object segmentation tasks of PASCAL VOC.

Implementation details: For those experiments we implemented our *RotNet* model with an AlexNet architecture. Our implementation of the AlexNet model does not have local response normalization units, dropout units, or groups in the colvolutional layers while it includes batch normalization units after each linear layer (either convolutional or fully connected)[1]. In order to train the AlexNet based *RotNet* model, we use SGD with batch

[1]For the definition of the AlexNet model that we used see:

Method	Conv4	Conv5
ImageNet labels from [7]	59.7	59.7
Random from [111]	27.1	12.0
Tracking [156]	38.8	29.8
Context [24]	45.6	30.4
Colorization [170]	40.7	35.2
Jigsaw Puzzles [111]	45.3	34.6
BIGAN [26]	41.9	32.2
NAT [7]	-	36.0
(Ours) RotNet	50.0	43.8

Table 5.5: Task Generalization: ImageNet top-1 classification with non-linear layers. We compare our unsupervised feature learning approach with other unsupervised approaches by training non-linear classifiers on top of the feature maps of each layer to perform the 1000-way ImageNet classification task, as proposed by [111]. For instance, for the conv5 feature map we train the layers that follow the conv5 layer in the AlexNet architecture (i.e., fc6, fc7, and fc8). Similarly for the conv4 feature maps. We implemented those non-linear classifiers with batch normalization units after each linear layer (fully connected or convolutional) and without employing drop out units. All approaches use AlexNet variants and were pre-trained on ImageNet without labels except the ImageNet labels and Random entries. During testing we use a single crop and do not perform flipping augmentation. We report top-1 classification accuracy.

size 192, momentum 0.9, weight decay $5e-4$ and lr of 0.01. We drop the learning rates by a factor of 10 after epochs 10, and 20 epochs. We train in total for 30 epochs. As in the CIFAR experiments, during training we feed the *RotNet* model all four rotated copies of an image simultaneously (in the same mini-batch).

ImageNet classification task: We evaluate the task generalization of our self-supervised learned features by training on top of them non-linear object classifiers for the ImageNet classification task (following the evaluation scheme of [111]). In Table 5.5 we report the classification performance of our self-supervised features and we compare it with the other unsupervised approaches. *We observe that our approach surpasses all the other methods by a significant margin*. For the feature maps generated by the Conv4 layer, our improvement is more than 4 percentage points and for the feature maps generated by the Conv5 layer, our improvement is even bigger, around 8 percentage points. Furthermore, our approach significantly narrows the performance gap between unsupervised features and supervised

`https://github.com/gidariss/FeatureLearningRotNet/blob/master/architectures/AlexNet.py`

Method	Conv1	Conv2	Conv3	Conv4	Conv5
ImageNet labels	19.3	36.3	44.2	48.3	50.5
Random	11.6	17.1	16.9	16.3	14.1
Random rescaled [74]	17.5	23.0	24.5	23.2	20.6
Context [24]	16.2	23.3	30.2	31.7	29.6
Context Encoders [118]	14.1	20.7	21.0	19.8	15.5
Colorization [170]	12.5	24.5	30.4	31.5	30.3
Jigsaw Puzzles [111] (from [112])	18.2	28.8	34.0	33.9	27.1
BIGAN [26]	17.7	24.5	31.0	29.9	28.0
Split-Brain [171]	17.7	29.3	35.4	35.2	32.8
Counting [112]	18.0	30.6	34.3	32.5	25.7
(Ours) RotNet	**18.8**	**31.7**	**38.7**	**38.2**	**36.5**

Table 5.6: Task Generalization: ImageNet top-1 classification with linear layers. We compare our unsupervised feature learning approach with other unsupervised approaches by training logistic regression classifiers on top of the feature maps of each layer to perform the 1000-way ImageNet classification task, as proposed by [170]. All weights are frozen and feature maps are spatially resized (with adaptive max pooling) so as to have around 9000 elements (as proposed by [170]). All approaches use AlexNet variants and were pre-trained on ImageNet without labels except the ImageNet labels and Random entries.

features. In Table 5.6 we report similar results but for linear (logistic regression) classifiers (following the evaluation scheme of [170]). Again, our unsupervised method demonstrates significant improvements over prior unsupervised methods.

Transfer learning evaluation on PASCAL VOC: In Table 5.8 we evaluate the task and dataset generalization of our unsupervised learned features by fine-tuning them on the PASCAL VOC classification, detection, and segmentation tasks. As with the ImageNet classification task, we outperform by significant margin all the competing unsupervised methods in all tested tasks, significantly narrowing the gap with the supervised case. Notably, the PASCAL VOC 2007 object detection performance that our self-supervised model achieves is 54.4% mAP, which is only 2.4 points lower than the supervised case. We provide the per class detection performance of our method in Table 5.9).

Places classification task: In Table 5.7 we evaluate the task and dataset generalization of our approach by training linear (logistic regression) classifiers on top of the learned features in order to perform the 205-way Places classification task. Note that in this case the learnt features are evaluated w.r.t. their generalization on classes that were "unseen" during

Method	Conv1	Conv2	Conv3	Conv4	Conv5
Places labels [175]	22.1	35.1	40.2	43.3	44.6
ImageNet labels	22.7	34.8	38.4	39.4	38.7
Random	15.7	20.3	19.8	19.1	17.5
Random rescaled [74]	21.4	26.2	27.1	26.1	24.0
Context [24]	19.7	26.7	31.9	32.7	30.9
Context Encoders [118]	18.2	23.2	23.4	21.9	18.4
Colorization [170]	16.0	25.7	29.6	30.3	29.7
Jigsaw Puzzles [111] (from [112])	23.0	31.9	35.0	34.2	29.3
BIGAN [26]	22.0	28.7	31.8	31.3	29.7
Split-Brain [171]	21.3	30.7	34.0	34.1	32.5
Counting [112]	**23.3**	**33.9**	**36.3**	**34.7**	29.6
(Ours) RotNet	21.5	31.0	35.1	34.6	**33.7**

Table 5.7: Task & Dataset Generalization: Places top-1 classification with linear layers. We compare our unsupervised feature learning approach with other unsupervised approaches by training logistic regression classifiers on top of the feature maps of each layer to perform the 205-way scene classification task of Places205 dataset [175]. All unsupervised methods are pre-trained (in an unsupervised way) on ImageNet. All weights are frozen and feature maps are spatially resized (with adaptive max pooling) so as to have around 9000 elements. All approaches use AlexNet variants and were pre-trained on ImageNet without labels except the Place labels, ImageNet labels, and Random entries.

the unsupervised training phase. As can be seen, even in this case our method manages to either surpass or achieve comparable results w.r.t. prior state-of-the-art unsupervised learning approaches.

5.4 Conclusions

In this chapter we proposed a novel formulation for self-supervised feature learning that trains a ConvNet model to be able to recognize the image rotation that has been applied to its input images. Despite the simplicity of our self-supervised task, we demonstrated that it successfully forces the ConvNet model trained on it to learn semantic features that are useful for a variety of visual perception tasks, such as object recognition, object detection, and object segmentation. We exhaustively evaluated our method in various unsupervised and semi-supervised benchmarks and we achieved in all of them state-of-the-art performance. Specifically, our self-supervised approach managed to drastically improve the state-of-the-art

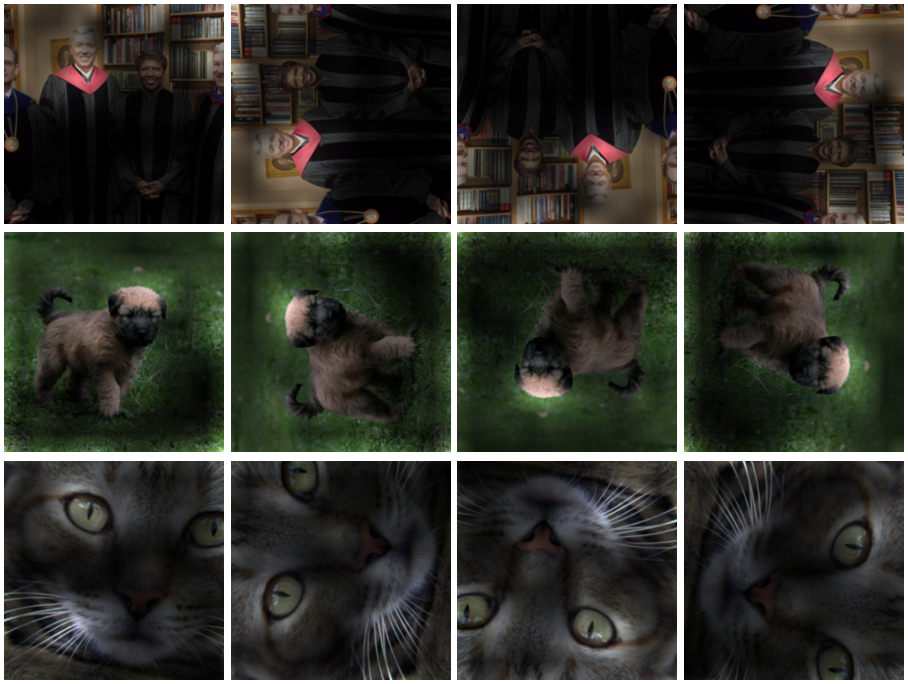
	Classification (%mAP)		Detection (%mAP)	Segmentation (%mIoU)
Trained layers	fc6-8	all	all	all
ImageNet labels	78.9	79.9	56.8	48.0
Random		53.3	43.4	19.8
Random rescaled [74]	39.2	56.6	45.6	32.6
Egomotion [1]	31.0	54.2	43.9	
Context Encoders [118]	34.6	56.5	44.5	29.7
Tracking [156]	55.6	63.1	47.4	
Context [24]	55.1	65.3	51.1	
Colorization [170]	61.5	65.6	46.9	35.6
BIGAN [26]	52.3	60.1	46.9	34.9
Jigsaw Puzzles [111]	-	67.6	53.2	37.6
NAT [7]	56.7	65.3	49.4	
Split-Brain [171]	63.0	67.1	46.7	36.0
ColorProxy [82]		65.9		38.4
Counting [112]	-	67.7	51.4	36.6
(Ours) RotNet	**70.9**	**73.0**	**54.4**	**39.1**

Table 5.8: Task & Dataset Generalization: PASCAL VOC 2007 classification and detection results, and PASCAL VOC 2012 segmentation results. We used the publicly available testing frameworks of [74] for classification, of [42] for detection, and of [92] for segmentation. For classification, we either fix the features before conv5 (column *fc6-8*) or we fine-tune the whole model (column *all*). For detection we use multi-scale training and single scale testing. All approaches use AlexNet variants and were pre-trained on ImageNet without labels except the ImageNet labels and Random entries. After unsupervised training, we absorb the batch normalization units in the linear layers and we use the weight rescaling technique proposed by [74] (which is common among the unsupervised methods). As customary, we report the mean average precision (mAP) on the classification and detection tasks, and the mean intersection over union (mIoU) on the segmentation task.

Classes	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
ImageNet labels	64.0	**69.6**	**53.2**	**44.4**	**24.9**	**65.7**	**69.6**	**69.2**	28.9	**63.6**	**62.8**	**63.9**	**73.3**	64.6	55.8	**25.7**	**50.5**	55.4	69.3	**56.4**
(Ours) RotNet	**65.5**	65.3	43.8	39.8	20.2	65.4	69.2	63.9	**30.2**	56.3	62.3	56.8	71.6	**67.2**	**56.3**	22.7	45.6	**59.5**	**71.6**	55.3

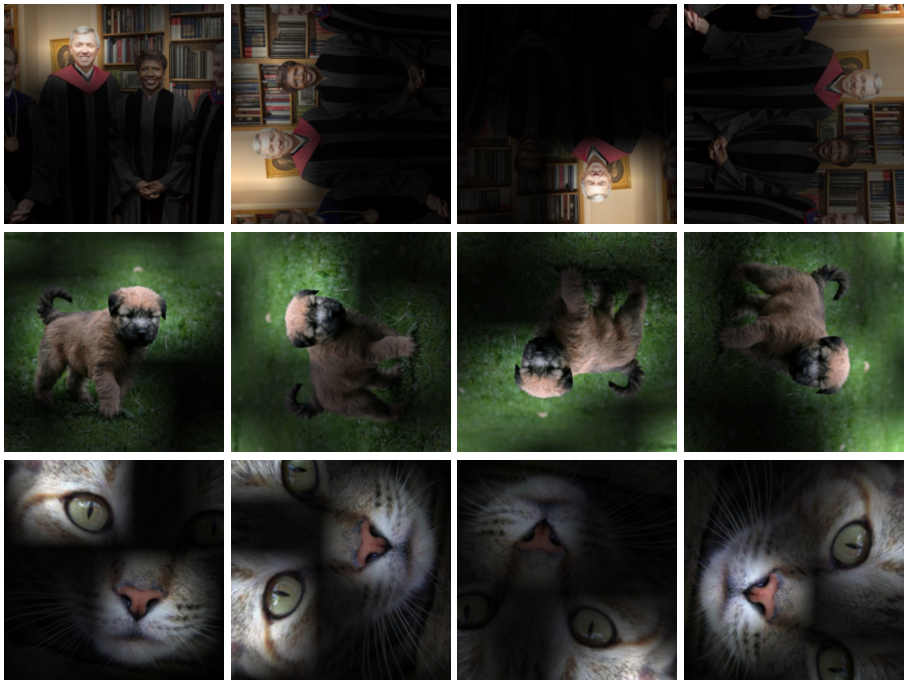
Table 5.9: Per class PASCAL VOC 2007 detection performance. As usual, we report the average precision metric. The results of the supervised model (i.e., ImageNet labels entry) come from [24].

results on unsupervised feature learning for ImageNet classification, PASCAL classification, PASCAL detection, PASCAL segmentation, and CIFAR-10 classification, surpassing prior approaches by a significant margin and thus drastically reducing the gap between unsupervised and supervised feature learning.



0° rotation 90° rotation 180° rotation 270° rotation

(a) Attention maps of Conv3 feature maps (size: 13×13**)**



0° rotation 90° rotation 180° rotation 270° rotation

(b) Attention maps of Conv5 feature maps (size: 6×6**)**

Figure 5-6: Attention maps of the Conv3 and Conv5 feature maps generated by an AlexNet model trained on the self-supervised task of recognizing image rotations. Here we present the attention maps generated for all the 4 rotated copies of an image.

Chapter 6

Few-Shot Visual Learning without Forgetting

6.1 Introduction

Over the last few years, deep convolutional neural networks [78, 144, 148, 57] (ConvNets) have achieved impressive results on image classification tasks, such as object recognition [129] or scene classification [175]. However, in order for a ConvNet to successfully learn to recognize a set of visual categories (e.g., object categories or scene types), it requires to manually collect and label thousands of training examples per target category and to apply on them an iterative gradient based optimization routine [84] that is extremely computationally expensive, e.g., it can consume hundreds or even thousands of GPU hours. Moreover, the set of categories that the ConvNet model can recognize remains fixed after training. In case we would like to expand the set of categories that the ConvNet can recognize, then we need to collect training data for the novel categories (i.e., those that they were not in the initial training set) and restart the aforementioned computationally costly training procedure this time on the enhanced training set such that we will avoid catastrophic interference. Even more, it is of crucial importance to have enough training data for the novel categories (e.g., thousands of examples per category) otherwise we risk overfitting on them.

In contrast, the human visual system exhibits the remarkably ability to be able to effortlessly learn novel concepts from only one or a few examples and reliably recognize

them later on. It is assumed that the reason the human visual system is so efficient when learning novel concepts is that it exploits its past experiences about the (visual) world. For example, a child, having accumulated enough knowledge about mammal animals and in general the visual world, can easily learn and generalize the visual concept of "rhinoceros" from only a single image. Mimicking that behavior on artificial vision systems is an interesting and very challenging research problem with many practical advantages, such as developing real-time interactive vision applications for portable devices (e.g., cell-phones), and aligned with our goal of annotation efficient learning.

Research on this subject is usually termed *few-shot learning*. However, most prior methods neglect to fulfill two very important requirements for a good few-shot learning system: **(a)** the learning of the novel categories needs to be fast, and **(b)** to not sacrifice any recognition accuracy on the initial categories that the ConvNet was trained on, i.e., to not "forget" (from now on we will refer to those initial categories by calling them base categories). Motivated by this observation, in this work we propose to tackle the problem of few-shot learning under a more realistic setting, where a large set of training data is assumed to exist for a set of base categories and, using these data as the sole input, we want to develop an object recognition learning system that, not only is able to recognize these base categories, but also learns to dynamically recognize novel categories from only a few training examples (provided only at test time) while also not forgetting the base ones or requiring to be re-trained on them (*dynamic few-shot learning without forgetting*). Compared to prior approaches, we believe that this setting more closely resembles the human visual system behavior (w.r.t. how it learns novel concepts). In order to achieve our goal, we propose two technical novelties.

Few-shot classification-weight generator based on attention. A typical ConvNet based recognition model, in order to classify an image, first extracts a high level feature representation from it and then computes per category classification scores by applying a set of classification weight vectors (one per category) to the feature. Therefore, in order to be able to recognize novel categories we must be able to generate classification weight vectors for them. In this context, the first technical novelty of our work is that we enhance a typical object recognition system with an extra component, called *few-shot classification weight*

generator that accepts as input a few training examples of a novel category (e.g., no more than five examples) and, based on them, generates a classification weight vector for that novel category. Its key characteristic is that in order to compose novel classification weight vectors, it explicitly exploits the acquired past knowledge about the visual world by incorporating an attention mechanism over the classification weight vectors of the base categories. This attention mechanism offers a significant boost on the recognition performance of novel categories, especially when there is only a single training example available for learning them.

Cosine-similarity based ConvNet recognition model. In order for the *few-shot classification weight generator* to be successfully incorporated into the rest of the recognition system, it is essential the ConvNet model to be able to simultaneously handle the classification weight vectors of both base and novel categories. However, as we will explain in the methodology, this is not feasible with the typical dot-product based classifier (i.e., the last linear layer of a classification neural network). Therefore, in order to overcome this serious issue, our second technical novelty is to implement the classifier as a cosine similarity function between the feature representations and the classification weight vectors. Apart from unifying the recognition of both base and novel categories, features learned with the cosine-similarity based classifier turn out to generalize significantly better on novel categories than those learned with a dot-product based classifier. Moreover, we demonstrate in the experimental section that, by simply training a cosine-similarity based ConvNet recognition model, we are able to learn feature extractors that, when used for image matching, they surpass prior state-of-the-art approaches on the few-shot recognition task.

To summarize, the contribution of the work presented in this chapter are as follows:

- We propose a few-shot object recognition system that is capable of dynamically learning novel categories from only a few training data while at the same time is not forgetting the base categories on which it was trained.
- In order to achieve that we introduced two technical novelties, an attention based few-shot classification weight generator, and the implementation of the classifier of a ConvNet model as a cosine similarity function between feature representations and classification vectors.

- We extensively evaluate our object recognition system on Mini-ImageNet, both w.r.t. its few-shot object recognition performance and its ability to not forget the base categories, and we report state-of-the-art results that surpass prior approaches by a very significant margin.
- Finally, we apply our approach on the recently introduced fews-shot benchmark of Hariharan and Girshick [51] where we achieve state-of-the-art results.

In the following sections, we provide related work in §6.2, we describe our few-shot object learning methodology in §6.3, we provide experimental results in §6.4, and finally we conclude in §6.5.

6.2 Related work

Recently, there is resurgence of interest on the few-shot learning problem. In the following we briefly discuss the most relevant approaches to our work.

Meta-learning based approaches. Meta-learning approaches typical involve a meta-learner model that, given a few training examples of a new task, tries to quickly learn a learner model that "solves" this new task [134, 152, 3, 107, 132]. Specifically, Ravi and Larochelle [124] propose an LSTM [59] based meta-learner that is trained given as input a few training examples of a new classification task to sequentially generate parameter updates that will optimize the classification performance of a learner model on that task. Their LSTM also learns the parameter initialization of the learner model. Finn et al. [36] simplified the above meta-learner model and only learn the initial learner parameters such that only a few gradient descent steps w.r.t. those initial parameters will achieve the maximal possible performance on the new task. Mishra et al. [105] instead propose a generic temporal convolutional network that, given as input a sequence of a few labeled training examples and then an unlabeled test example, predicts the label of that test example. Our system also includes a meta-learner network component, the few-shot classification weight generator.

Metric-learning based approaches. In general, metric learning approaches attempt to learn feature representations that preserve the class neighborhood structure (i.e., features of the same object are closer than features of different objects). Specifically, Koch et

al. [72] formulated the one-shot object recognition task as image matching and train Siamese neural networks to compute the similarity between a training example of a novel category and a test example. Vinyals et al. [155] proposed Matching Networks that, in order to classify a test example, employs a differentiable nearest neighbor classifier implemented with an attention mechanism over the learned representations of the training examples. Prototypical Networks [145] learn to classify test examples by computing distances to prototype feature vectors of the novel categories. They propose to learn the prototype feature vector of a novel category as the average of the feature vectors extracted by the training examples of that category. A similar approach was proposed before by Mensink et al. [102] and Prototypical Networks can be viewed as an adaption of that work for ConvNets. Despite their simplicity, Prototypical Networks demonstrated state-of-the-art performance. Our few-shot classification weight generator also includes a feature averaging mechanism. However, more than that, it also explicitly exploits past knowledge about the visual world with an attention based mechanism and the overall framework allows us to perform unified recognition of both base and novel categories without altering the way base categories are learned and recognized.

In a different line of work, Hariharan and Girshick [51] propose to use during training a l_2 regularization loss on the feature representations that forces them to better generalize on "unseen" categories. In our case, the cosine-similarity based classifier, apart from unifying the recognition of both base and novel categories, also leads to feature representations that are able to better generalize on "unseen" categories. Also, their framework is able to recognize both base and novel categories as ours. However, to achieve that goal they re-train the classifier on both the base categories (with a large set of training data) and the novel categories (with few training data), which is in general slow and requires constantly maintaining in disc a large set of training data.

6.3 Methodology

As an input to our object recognition learning system we assume that there exists a dataset of K_{base} base categories:

$$D_{train} = \bigcup_{b=1}^{K_{base}} \{x_{b,i}\}_{i=1}^{N_b} , \tag{6.1}$$

where N_b is the number of training examples of the b-th category and $x_{b,i}$ is its i-th training example. Using this as the only input, the goal of our work is to be able to both learn to accurately recognize base categories and to learn to perform few-shot learning of novel categories in a dynamic manner and without forgetting the base ones. An overview of our framework is provided in Figure 6-1. It consists of two main components, a *ConvNet-based recognition model* that is able to recognize both base and novel categories and a *few-shot classification weight generator* that dynamically generates classification weight vectors for the novel categories at test time:

ConvNet-based recognition model. It consists of **(a)** a feature extractor $F(.|\theta)$ (with learnable parameters θ) that extracts a d-dimensional feature vector $z = F(x|\theta) \in \mathbb{R}^d$ from an input image x, and **(b)** a classifier $C(.|W^*)$, where $W^* = \{w_k^* \in \mathbb{R}^d\}_{k=1}^{K^*}$ are a set of K^* classification weight vectors - one per object category, that takes as input the feature representation z and returns a K^*-dimensional vector with the probability classification scores $p = C(z|W^*)$ of the K^* categories. Note that in a typical convolutional neural network the feature extractor is the part of the network that starts from the first layer and ends at the last hidden layer while the classifier is the last classification layer. During the single training phase of our algorithm, we learn the θ parameters and the classification weight vectors of the base categories $W_{base} = \{w_k\}_{k=1}^{K_{base}}$ such that by setting $W^* = W_{base}$ the ConvNet model will be able to recognize the base object categories.

Few-shot classification weight generator. This comprises a meta-learning mechanism that, during test time, takes as input a set of K_{novel} novel categories with few training examples per category

$$D_{novel} = \bigcup_{n=1}^{K_{novel}} \{x'_{n,i}\}_{i=1}^{N'_n} , \tag{6.2}$$

where N'_n is the number of training examples of the n-th novel category and $x'_{n,i}$ is its

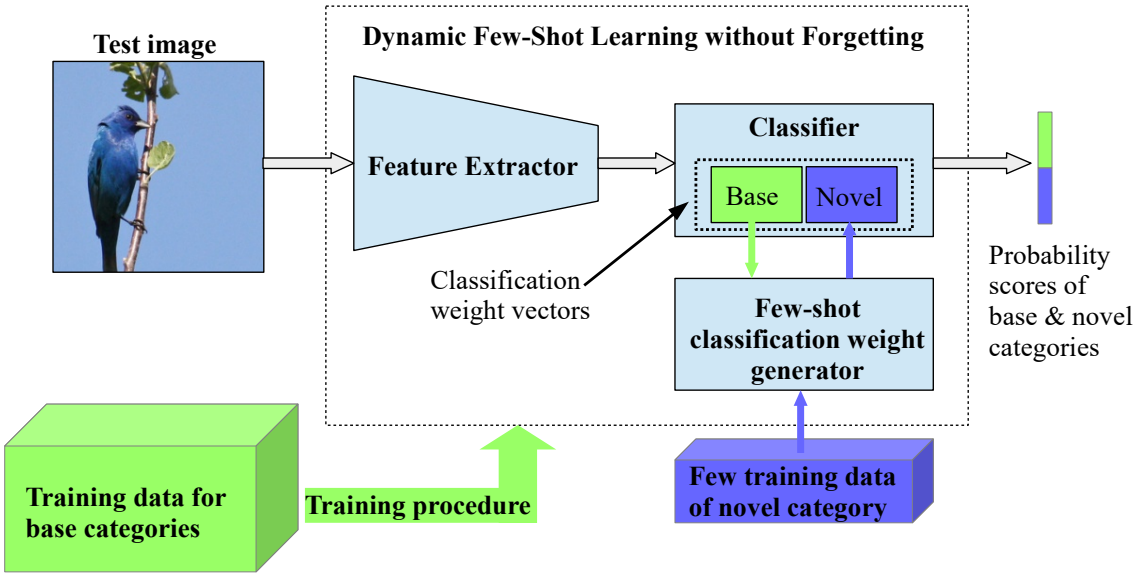


Figure 6-1: Overview of our system. It consists of: (a) a *ConvNet based recognition model* (that includes a feature extractor and a classifier) and (b) a *few-shot classification weight generator*. Both are trained on a set of base categories for which we have available a large set of training data. During test time, the weight generator gets as input a few training data of a novel category and the classification weight vectors of base categories (green rectangle inside the classifier box) and generates a classification weight vector for this novel category (blue rectangle inside the classifier box). This allows the ConvNet to recognize both base and novel categories.

i-th training example, and is able to dynamically assimilate the novel categories on the repertoire of the above ConvNet model. More specifically, for each novel category $n \in [1, N_{novel}]$, the few-shot classification weight generator $G(.,.|\phi)$ gets as input the feature vectors $Z'_n = \{z'_{n,i}\}_{i=1}^{N'_n}$ of its N'_n training examples, where $z'_{n,i} = F(x'_{n,i}|\theta)$, and the classification weight vectors of the base categories W_{base} and generates a classification weight vector $w'_n = G(Z'_n, W_{base}|\phi)$ for that novel category. Note that ϕ are the learnable parameters of the few-shot weight generator, which are learned during the single training phase of our framework. Therefore, if $W_{novel} = \{w'_n\}_{n=1}^{K_{novel}}$ are the classification weight vectors of the novel categories inferred by the few-shot weight generator, then by setting $W^* = W_{base} \cup W_{novel}$ on the classifier $C(.|W^*)$ we enable the ConvNet model to recognize both base and novel categories.

A key characteristic of our framework is that it is able to effortlessly (i.e., quickly during test time) learn novel categories and at the same time recognize both base and novel

categories in a unified manner. In the following subsections, we will describe in more detail the ConvNet-based recognition model in §6.3.1 and the few-shot weight generator in §6.3.2. Finally, we will explain the training procedure in §6.3.3.

6.3.1 Cosine-similarity based recognition model

A crucial difference of our ConvNet based recognition model compared to a standard one is that it should be able to dynamically incorporate at test time a variable number of novel categories (through the few-shot weight generator).

The standard setting for classification neural networks is, after having extracted the feature vector z, to estimate the classification probability vector $p = C(z|W^*)$ by first computing the raw classification score s_k of each category $k \in [1, K^*]$ using the dot-product operator:

$$s_k = z^\intercal w_k^* \,, \tag{6.3}$$

where w_k is the k-th classification weight vector in W^*, and then applying the softmax operator across all the K^* classification scores, i.e., $p_k = softmax(s_j)$, where p_k is the k-th classification probability of p. In our case the classification weight vectors w_k^* could come both from the base categories, i.e., $w_k^* \in W_{base}$, and the novel categories, i.e., $w_k^* \in W_{novel}$. However, the mechanisms involved during learning those classification weights are very different. The base classification weights, starting from their initial state, are slowly modified (i.e., slowly learned) with small SGD steps and thus their magnitude changes slowly over the course of their training. In contrast, the novel classification weights are dynamically predicted (i.e., quickly learned) by the weight generator based on the input training feature vectors and thus their magnitude depends on those input features. Due to those differences, the weight values in those two cases (i.e., base and novel classification weights) can be completely different, and so the same applies to the raw classification scores computed with the dot-product operation, which can thus have totally different magnitudes depending on whether they come from the base or the novel categories. This can severely impede the training process and, in general, does not allow to have a unified recognition of both type of categories. In order to overcome this critical issue, we propose to modify the classifier

(a) Cosine-similarity based features **(b) Dot-product based features**

Figure 6-2: Here we visualize the t-SNE [98] scatter plots of the feature representations learned with **(a)** the cosine-similarity based ConvNet recognition model, and **(b)** the dot-product based ConvNet recognition model. Note that in the case of the cosine-similarity based ConvNet recognition model, we visualize the l_2-normalized features. The visualized feature data points are from the "unseen" during training validation categories of Mini-ImageNet (i.e., novel categories). Each data point in the t-SNE scatter plots is colored according to its category.

$C(.|W^*)$ and compute the raw classification scores using the cosine similarity operator:

$$s_k = \tau \cdot \cos(z, w_k^*) = \tau \cdot \overline{z}^\intercal \overline{w}_k^* \,, \tag{6.4}$$

where $\overline{z} = \frac{z}{\|z\|}$ and $\overline{w}_k^* = \frac{w_k^*}{\|w_k^*\|}$ are the l_2-normalized vectors (from now on we will use the overline symbol $\overline{z}$ to indicate that a vector z is l_2-normalized), and τ is a learnable scalar value[1]. Since the cosine similarity can be implemented by first l_2-normalizing the feature vector z and the classification weight vector w_k^* and then applying the dot-product operator, the absolute magnitudes of the classification weight vectors can no longer affect the value of the raw classification score (as a result of the l_2 normalization that took place).

In addition to the above modification, we also choose to remove the ReLU non-linearity [108] after the last hidden layer of the feature extractor, which allows the feature

[1] The scalar parameter τ is introduced in order to control the peakiness of the probability distribution generated by the softmax operator since the range of the cosine similarity is fixed to $[-1, 1]$. In all of our experiments τ is initialized to 10.

vector z to take both positive and negative values, similar to the classification weight vectors. Note that the removal of the ReLU non-linearity does not make the composition of the last hidden layer with the classification layer a linear operation, since we l_2-normalize the feature vectors, which is a non-linear operation. In our initial experiments with the cosine similarity based classifier we found that such a modification can significantly improve the recognition performance of novel categories.

We note that, although cosine similarity is already well established as an effective similarity function for classifying a test feature by comparing it with the available training features vectors [155, 102, 125], in this work we use it for a different purpose, i.e., to replace the dot-product operation of the last linear layer of classification ConvNets used for applying the learnable weights of that layer to the test feature vectors. The proposed modification in the architecture of a classification ConvNet allows us to unify the recognition of base and novel categories without significantly altering the classification pipeline for the recognition of base categories (in contrast to [102, 125]). To the best of our knowledge, employing the cosine similarity operation in such a way is novel in the context of few shot learning. Interestingly, concurrently to us, Qi et al. [122] also propose to use the cosine similarity function in a similar way for the few-shot learning task. In a different line of work, very recently Chunjie et al. [95] also explored cosine similarity for the typical supervised classification task.

Advantages of cosine-similarity based classifier. Apart from making possible the unified recognition of both base and novel categories, *the cosine-similarity based classifier leads the feature extractor to learn features that generalize significantly better on novel categories than features learned with the dot-product based classifier*. A possible explanation for this is that, in order to minimize the classification loss of a cosine-similarity based ConvNet model, the l_2-normalized feature vector of an image must be very closely matched with the l_2-normalized classification weight vector of its ground truth category. As a consequence, the feature extractor is forced to (a) learn to encode on its feature activations exactly those discriminative visual cues that also the classification weight vectors of the ground truth categories learn to look for, and (b) learn to generate l_2-normalized feature vectors with low intraclass variance, since all the feature vectors that belong to the same

category must be very closely matched with the single classification weight vector of that category. This is visually illustrated in Figure 6-2, where we visualize t-SNE scatter plots of cosine-similarity-based and dot-product-based features related to categories "unseen" during training. As can be clearly observed, the features generated from the cosine-similarity-based ConvNet form more compact and distinctive category-specific clusters (i.e., they provide more discriminative features). Moreover, our cosine-similarity based classification objective resembles the training objectives typically used by metric learning approaches [60]. In fact, it turns out that *our feature extractor trained solely on cosine-similarity based classification of base categories, when used for image matching, manages to surpass all prior state-of-the-art approaches on the few-shot object recognition task*.

6.3.2 Few-shot classification weight generator

The few-shot classification weight generator $G(.,.|\phi)$ gets as input the feature vectors $Z' = \{z'_i\}_{i=1}^{N'}$ of the N' training examples of a novel category (typically $N' \leq 5$) and (optionally) the classification weight vectors of the base categories W_{base}. Based on them, it infers a classification weight vector $w' = G(Z', W_{base}|\phi)$ for that novel category. Here we explain how the above few-shot classification weight generator is constructed.

Feature averaging based weight inference. Since, as we explained in section § 6.3.1, the cosine similarity based classifier of the ConvNet model forces the feature extractor to learn feature vectors that form compact category-wise clusters and the classification weight vectors to learn to be representative feature vectors of those clusters, an obvious choice is to infer the classification weight vector w' by averaging the feature vectors of the training examples (after they have been l_2-normalized):

$$w'_{avg} = \frac{1}{N'} \sum_{i=1}^{N'} \overline{z}'_i \, . \tag{6.5}$$

The final classification weight vector in case we only use the feature averaging mechanism is:

$$w' = \phi_{avg} \odot w'_{avg} \, , \tag{6.6}$$

where $\odot$ is the Hadamard product, and $\phi_{avg} \in \mathbb{R}^d$ is a learnable weight vector. Similar strategy has been previously proposed by Snell et al. [145] and has demonstrated very good results. However, it does not fully exploit the knowledge about the visual world that the ConvNet model acquires during its training phase. Furthermore, in case there is only a single training example for the novel category, the averaging cannot infer an accurate classification weight vector.

Attention-based weight inference. We enhance the above feature averaging mechanism with an attention based mechanism that composes novel classification weight vectors by "looking" at a memory that contains the base classification weight vectors $W_{base} = \{w_b\}_{b=1}^{K_{base}}$. More specifically, an extra attention-based classification weight vector w'_{att} is computed as:

$$w'_{att} = \frac{1}{N'} \sum_{i=1}^{N'} \sum_{b=1}^{K_{base}} Att(\phi_q \overline{z}'_i, k_b) \cdot \overline{w}_b \,, \tag{6.7}$$

where $\phi_q \in \mathbb{R}^{d \times d}$ is a learnable weight matrix that transforms the feature vector $\overline{z}'_i$ to query vector used for querying the memory, $\{k_b \in \mathbb{R}^d\}_b^{K_{base}}$ is a set of K_{base} learnable keys (one per base category) used for indexing the memory, and $Att(.,.)$ is an attention kernel implemented as a cosine similarity function[2] followed by a softmax operation over the K_{base} base categories. The final classification weight vector is computed as a weighted sum of the average based classification vector w'_{avg} and the attention based classification vector w'_{att}:

$$w' = \phi_{avg} \odot w'_{avg} + \phi_{att} \odot w'_{att} \,, \tag{6.8}$$

where $\odot$ is the Hadamard product, and ϕ_{avg}, $\phi_{att} \in \mathbb{R}^d$ are learnable weight vectors.

Why using an attention-based weight composition? Thanks to the cosine-similarity based classifier, the base classification weight vectors learn to be representative feature vectors of their categories. Thus, the base classification weight vectors also encode visual similarity, e.g., the classification vector of a mammal animal should be closer to the classification vector of another mammal animal rather than the classification vector of a

[2]The cosine similarity scores are also scaled by a learnable scalar parameter γ in order to increase the peakiness of the softmax distribution. This scalar learnable parameter is initialized to 10 on the experiments of section 6.4.1 and to 30 on the experiments of section 6.4.2.

vehicle. Therefore, the classification weight vector of a novel category can be composed as a linear combination of those base classification weight vectors that are most similar to the few training examples of that category. This allows our few-shot weight generator to explicitly exploit the acquired knowledge about the visual world (here represented by the base classification weight vectors) in order to improve the few-shot recognition performance. This improvement is very significant especially in the one-shot recognition setting where averaging cannot provide an accurate classification weight vector.

6.3.3 Training procedure

In order to learn the ConvNet-based recognition model (i.e. the feature extractor $F(.|\theta)$ as well as the classifier $C(.|W^*)$) and the few-shot classification weight generator $G(.,.|\phi)$, we use as the sole input a training set $D_{train} = \bigcup_{b=1}^{K_{base}} \{x_{b,i}\}_{i=1}^{N_b}$ of K_{base} base categories. We split the training procedure into 2 stages and at each stage we minimize a different cross-entropy loss of the following form:

$$\frac{1}{K_{base}} \sum_{b=1}^{K_{base}} \frac{1}{N_b} \sum_{i=1}^{N_b} loss(x_{b,i}, b), \tag{6.9}$$

where $loss(x, y)$ is the negative log-probability $-\log(p_y)$ of the y-th category in the probability vector:

$$p = C(F(x|\theta)|W^*). \tag{6.10}$$

The meaning of W^* is different on each of the training stages, as we explain below.

1st training stage: During this stage we only learn the ConvNet recognition model without the few-shot classification weight generator. Specifically, at this stage we learn the parameters θ of the feature extractor $F(.|\theta)$ and the base classification weight vectors $W_{base} = \{w_b\}_{b=1}^{K_{base}}$. This is done in exactly the same way as for any other standard recognition model. Thus, during the 1st training stage the set of classification weight vectors W^* in equation 6.10 is equal to the base classification weight vectors W_{base}.

2nd training stage: During this stage we train the learnable parameters ϕ of the few-shot classification weight generator while we continue training the base classification weight

vectors W_{base} (in our experiments during that training stage we froze the feature extractor). In order to train the few-show classification weight generator, during each training iteration we form the following two-step "training episodes"[3]:

- First, we randomly pick K_{novel} "fake" novel categories from the existing base categories and we treat them in the same way as we will treat the actual novel categories after training. Specifically, instead of using the classification weight vectors in W_{base} (which are learned with stochastic gradient descent) for those "fake" novel categories, we sample N' training examples (typically $N' \leq 5$) for each of them, compute their feature vectors $Z' = \{z'_i\}_{i=1}^{N'}$, and give those feature vectors to the few-shot classification weight generator $G(.,.|\phi)$ in order to compute novel classification weight vectors of those "fake" novel categories Note that during this process we take care to exclude from the base classification weight vectors that are given as a second argument to the few-shot weight generator $G(.,.|\phi)$ those classification vectors that correspond to the "fake" novel categories. The inferred classification weight vectors are used for recognizing the "fake" novel categories. Thus, during this 2nd training stage the set of classification weight vectors W^* in equation 6.10 is the union of the "fake" novel classification weight vectors generated by $G(.,.|\phi)$ and the classification weight vectors of the remaining base categories.
- The second step of the "training episode" is to sample T_{novel} test image examples from the "fake" novel categories, and T_{base} test image examples from the remaining base categories and then classify them using the set of classification weight vectors W^* that was formed from the 1st step of the training episode.

Both the step of generating classification weights vectors from training examples and the step of applying the classification weight vectors to test examples are end-to-end differentiable. Thus, by applying the cross entropy loss (equation 6.9) on the classification probabilities of the $T = T_{novel} + T_{base}$ test examples we are able to train both the learnable parameters of the few-shot classification weight generator and the learnable parameters of the recognition model (i.e., the base classification weight vectors W_{base}).

[3]Note that during each training iteration of stochastic gradient descent the mini-batch that is formed could include multiple different instances of such "training episodes".

Models	5-Shot learning – K_{novel}=5			1-Shot learning – K_{novel}=5		
	Novel	Base	Both	Novel	Base	Both
Matching-Nets [155]	68.87 ± 0.38%	-	-	55.53 ± 0.48%	-	-
Prototypical-Nets [145]	72.67 ± 0.37%	62.10%	32.70%	54.44 ± 0.48%	52.35%	26.68%
Ours						
Cosine Classifier	72.83 ± 0.35%	70.68%	51.89%	54.55 ± 0.44%	70.68%	39.17%
Cosine Classifier & Avg. Weight Gen	74.66 ± 0.35%	70.92%	60.26%	55.33 ± 0.46%	70.45%	48.56%
Cosine Classifier & Att. Weight Gen	**74.92 ± 0.36%**	70.88%	60.50%	**58.55 ± 0.50%**	70.73%	50.50%
Ablations						
Dot Product	64.58 ± 0.38%	63.59%	31.80%	46.09 ± 0.40%	63.59%	24.76%
Dot Product & Avg. Weight Gen	60.30 ± 0.39%	62.15%	46.41%	44.31 ± 0.40%	61.99%	39.05%
Dot Product & Att. Weight Gen	67.81 ± 0.37%	62.11%	48.70%	53.88 ± 0.48%	62.28%	42.41%
Ablations						
Cosine w/ ReLU.	71.04 ± 0.36%	**72.51%**	58.16%	52.91 ± 0.45%	**72.51%**	43.17%
Cosine w/ ReLU. & Avg. Weight Gen	71.30 ± 0.38%	72.47%	59.33%	53.19 ± 0.45%	71.70%	49.53%
Cosine w/ ReLU. & Att. Weight Gen	73.03 ± 0.38%	72.26%	**61.05%**	56.09 ± 0.54%	72.34%	**51.25%**

Table 6.1: Average classification accuracies on the validation set of Mini-ImageNet. The Novel columns report the average 5-way and 1-shot or 5-shot classification accuracies of novel categories (with 95% confidence intervals), the Base and Both columns report the classification accuracies of base categories and of both type of categories respectively. In order to report those results we sampled 2000 tasks each with 15×5 test examples of novel categories and 15×5 test examples of base categories.

6.4 Experimental results

We extensively evaluate the proposed few-shot recognition system w.r.t. both its few-shot recognition performance of novel categories and its ability to not "forget" the base categories on which it was trained.

6.4.1 Mini-ImageNet experiments

Evaluation setting for recognition of novel categories. We evaluate our few-shot object recognition system on the Mini-ImageNet dataset [155] that includes 100 different categories with 600 images per category, each of size 84×84. For our experiments we used the splits by Ravi and Laroche [124] that include 64 categories for training, 16 categories for validation, and 20 categories for testing. The typical evaluation setting on this dataset is first to train a few-shot model on the training categories and then during test time to use the validation (or the test) categories in order to form few-shot tasks on which the trained model is evaluated.

Models	Feature Extractor	5-Shot learning – K_{novel}=5			1-Shot learning – K_{novel}=5		
		Novel	Base	Both	Novel	Base	Both
Matching-Nets [155]	C64F	55.30%	-	-	43.60%	-	-
Ravi and Laroche [124]	C32F	$60.20 \pm 0.71\%$	-	-	$43.40 \pm 0.77\%$	-	-
Finn et al. [36]	C64F	$63.10 \pm 0.92\%$	-	-	$48.70 \pm 1.84\%$	-	-
Prototypical-Nets [145]	C64F	$68.20 \pm 0.66\%$	-	-	$49.42 \pm 0.78\%$	-	-
Mishra et al. [105]	RESNET	$68.88 \pm 0.92\%$	-	-	$55.71 \pm 0.99\%$	-	-
Ours	C32F	$70.27 \pm 0.64\%$	61.08%	52.45%	$54.33 \pm 0.81\%$	61.09%	43.05%
Ours	C64F	$72.81 \pm 0.62\%$	68.13%	57.72%	**$56.20 \pm 0.86\%$**	68.08%	48.09%
Ours	C128F	**$73.00 \pm 0.64\%$**	70.90%	**59.35%**	$55.95 \pm 0.84\%$	70.72%	49.08%
Ours	RESNET	$70.13 \pm 0.68\%$	**80.16%**	56.04%	$55.45 \pm 0.89\%$	**80.24%**	**51.23%**

Table 6.2: Average classification accuracies on the test set of Mini-ImageNet. In order to report those results we sampled 600 tasks in a similar fashion as for the validation set of Mini-ImageNet.

Those few-shot tasks are formed by first sampling K_{novel} categories and one or five training example per category (1-shot and 5-shot settings respectively), which the trained model uses for meta-learning those categories, and then evaluating it on some test examples that come from the same novel categories but do not overlap with the training examples.

Evaluation setting for the recognition of the base categories. When we evaluate our model w.r.t. few-shot recognition task on the validation / test categories, we consider as base categories the 64 training categories on which we trained the model. Since the proposed few-shot object recognition system has the ability to not forget the base categories, we would like to also evaluate the recognition performance of our model on those base categories. Therefore, we sampled 300 extra images for each training category that we use as validation image set for the evaluation of the recognition performance of the base categories and also another 300 extra images that are used for the same reason as test image set. Therefore, when we evaluate our model w.r.t. the few-shot learning task on the validation / test categories we also evaluate w.r.t. recognition performance of the base categories on the validation / test image set of the training categories.

Implementation details of training procedure of the proposed approach. During the 1st training stage, the recognition model was trained for 60 epochs using a stochastic gradient descent optimizer with momentum 0.9 and weight decay $5e-4$. The learning rate was set to 0.1 for the first 20 epochs, then dropped to 0.006 for the next 20 epochs, then again dropped to 0.0012 for the next 10 epochs, and finally again dropped to 0.00024

for the remaining 10 epochs. Each training epoch lasted for 1000 training batches of size 256. During the 2nd training stage, our model was trained for 60 epochs using a stochastic gradient descent optimizer with momentum 0.9 and weight decay $5e-4$. The learning rate was set to 0.1 for the first 20 epochs, then dropped to 0.006 for the next 20 epochs, then again dropped to 0.0012 for the next 10 epochs, and finally again dropped to 0.00024 for the remaining 10 epochs. Each training epoch lasted for 1000 training batches. Each training batch included 8 "training episodes". For each "training episode" 5 "fake" novel categories were sampled from the 64 base categories, $T_{novel} = 15$ test images were sampled from the "fake" novel categories, and $T_{base} = 15$ test images were sampled from the remaining base categories. Note that despite the fact that both training stages last for 60 training epochs, because the training split of Mini-Imagenet is relatively small and there is a danger of overfitting, the training snapshot (i.e., the model parameters at the end of each training epoch) that "survives" from each training stage is that that achieves the highest accuracy on the novel categories of the validation split of Mini-Imagenet. For more implementation details we refer to the implementation code of this chapter.

6.4.1.1 Ablation study

In Table 6.1 we provide an ablation study of the proposed object recognition framework on the validation set of mini-ImageNet. We also compare with two prior state-of-the-art approaches, Prototypical Networks [145] and Matching Nets [155], that we re-implemented ourselves in order to ensure a fair comparison. The feature extractor used in all cases is a ConvNet model that has 4 convolutional modules, with 3×3 convolutions, followed by batch normalization, ReLU nonlinearity[4], and 2×2 max-pooling. Given as input images of size 84×84 it yields feature maps with spatial size 5×5. The first two convolutional layers have 64 feature channels and the latter two have 128 feature channels.

Cosine-similarity based ConvNet model. First we examine the performance of the cosine-similarity based ConvNet recognition model (entry Cosine Classifier) without training the few-shot classification weight generator (i.e., we only perform the 1st training stage as

[4]Unless otherwise stated, our cosine-similarity based models as well as the re-implementation of Matching-Nets do not have a ReLU nonlinearity after the last convolutional layer, since in both cases this modification improved the recognition performance on the few-shot recognition task

was described in section 6.3.3). In order to test its performance on the novel categories, during test time we estimate classification weight vectors using feature averaging. *We want to stress out that in this case there are no learnable parameters involved in the generation of the novel classification weight vectors and also the ConvNet model it was never trained on the few-shot recognition task. Despite that, the features learned by the cosine-similarity based ConvNet model match or even surpass the performance of the Matching-Nets and Prototypical Networks, which are explicitly trained on the few-shot object recognition task.* By comparing the cosine-similarity based ConvNet models (Cosine Classifier entries) with the dot-product based models (Dot Product entries) we observe that the former drastically improve the few-shot object recognition performance, which means that the feature extractor that is learned with the cosine-similarity classifier generalizes significantly better on "unseen" categories than the feature extractor learned with the dot-product classifier. Notably, the cosine-similarity classifier significantly improves also the recognition performance on the base categories.

Removing the last ReLU unit. In our work we propose to remove the last ReLU non-linearity from the feature extractor when using a cosine classifier. Instead, keeping the ReLU units (Cosine w/ ReLU entries) decreases the accuracy on novel categories while increasing it on base categories.

Few-shot classification weight generator. Here we examine the performance of our system when we also incorporate on it the proposed few-shot classification weight generator. In Table 6.1 we provide two solutions for the few-shot weight generator: the entry Cosine Classifier & Avg. Weight Gen that uses only the feature averaging mechanism described in section 6.3.2 and the entry Cosine Classifier & Att. Weight Gen that uses both the feature averaging and the attention based mechanism. Both types of few-shot weight generators are trained during the 2nd training stage that is described in section 6.3.3. We observe that both of them offer a very significant boost on the few-shot recognition performance of the cosine similarity based model (entry Cosine Classifier). Among the two, the attention based solution exhibits better few-shot recognition behavior, especially in the 1-shot setting where it has more than 3 percentage points higher performance. Also, it is easy to see that the few-shot classification weight generator does not affect the recognition performance

Approach	Novel					All					All with prior				
	N'=1	2	5	10	20	N'=1	2	5	10	20	N'=1	2	5	10	20
Prior work															
Prototypical-Nets [145] (from [157])	39.3	54.4	66.3	71.2	73.9	49.5	61.0	69.7	72.9	74.6	53.6	61.4	68.8	72.0	73.8
Matching Networks [155] (from [157])	43.6	54.0	66.0	72.5	76.9	54.4	61.0	69.0	73.7	76.5	54.5	60.7	68.2	72.6	75.6
Logistic regression (from [157])	38.4	51.1	64.8	71.6	76.6	40.8	49.9	64.2	71.9	76.9	52.9	60.4	68.6	72.9	76.3
Logistic regression w/ H [51] (from [157])	40.7	50.8	62.0	69.3	76.5	52.2	59.4	67.6	72.8	76.9	53.2	59.1	66.8	71.7	76.3
SGM w/ H [51]	-	-	-	-	-	54.3	62.1	71.3	75.8	78.1	-	-	-	-	-
Batch SGM [51]	-	-	-	-	-	49.3	60.5	71.4	75.8	78.5	-	-	-	-	-
Concurrent work															
Prototype Matching Nets w/ H [157]	45.8	**57.8**	69.0	74.3	77.4	57.6	64.7	71.9	75.2	77.5	56.4	63.3	70.6	74.0	76.2
Prototype Matching Nets [157]	43.3	55.7	68.4	74.0	77.0	55.8	63.1	71.1	75.0	77.1	54.7	62.0	70.2	73.9	75.9
Ours															
Cosine Classifier & Avg. Weight Gen	45.23	56.90	68.68	74.36	77.69	57.65	64.69	72.35	76.18	78.46	56.43	63.41	70.95	74.75	77.00
	± .25	± .16	± .09	± .06	± .06	± .15	± .10	± .06	± .04	± .04	± .15	± .10	± .06	± .04	± .03
Cosine Classifier & Att. Weight Gen	**46.02**	57.51	**69.16**	**74.83**	**78.11**	**58.16**	**65.21**	**72.72**	**76.50**	**78.74**	**56.76**	**63.80**	**72.72**	**75.02**	**77.25**
	± .25	± .15	**± .09**	**± .06**	**± .05**	**± .15**	**± .09**	**± .06**	**± .04**	**± .03**	**± .15**	**± .10**	**± .06**	**± .04**	**± .04**

Table 6.3: Top-5 accuracy on the novel categories and on all categories (with and without priors) fot the ImageNet based few-shot benchmark proposed in [51] (for more details about the evaluation metrics we refer to [157]). For each novel category we use $N' = 1$, 2, 5, 10 or 20 training examples. Methods with "w/ H" use mechanisms that hallucinate extra training examples for the novel categories. The second rows in our entries report the 95% confidence intervals.

of the base categories, which is around 70.50% in all the cosine-similarity based models. Moreover, by introducing the few-shot weight generator, the recognition performance in both type of categories (columns Both) increases significantly, which means that the ConvNet model achieves better behavior w.r.t. our goal of unified recognition of both base and novel categories. The few-shot recognition performance of our full system, which is the one that includes the attention based few-shot weight generator (entry Cosine classifier & Att. Weight Gen), offers a very significant improvement w.r.t. the prior state-of-the-art approaches on the few-shot object recognition task, i.e., from 72.67% to 74.92% in the 5-shot setting and from 55.53% to 58.55% in the 1-shot setting. Also, our system achieves significantly higher performance on the recognition of base categories compared to Prototypical Networks[5].

6.4.1.2 Comparison with state-of-the-art

Here we compare the proposed few-shot object recognition system with other state-of-the-art approaches on the Mini-ImageNet test set.

Explored feature extractor architectures. Because prior approaches use several different network architectures for implementing the feature extractor of the ConvNet model,

[5]In order to recognize base categories with Prototypical Networks, the prototypes for the base categories are computed by averaging all the available training features vectors

we evaluate our model with each of those architectures. Specifically the architectures that we evaluated are: C32F is a 4 module ConvNet network (which was described in § 6.4.1.1) with 32 feature channels on each convolutional layer, C64F has 64 feature channels on each layer, and in C128F the first two layers have 64 channels and the latter two have 128 channels (exactly the same as the model that was used in § 6.4.1.1). With RESNET we refer to the ResNet [57] like network that was used from Mishra et al. [105] (for more details we refer to [105]).

In Table 6.2 we provide the experimental results. In all cases, our models (that include the cosine-similarity based ConvNet model and the attention-based few-shot weight generator) achieve better few-shot object recognition performance than prior approaches. *Moreover, it is very important to note that our approach is capable to achieve such excellent accuracy on the novel categories while at the same time not sacrificing the recognition performance of the base categories, which is an ability that prior methods lack.*

6.4.1.3 Qualitative evaluation with t-SNE scatter plots

Here we compare qualitatively the feature representations learned by the proposed cosine-similarity based ConvNet recognition model with those learned by the typical dot-product based ConvNet recognition model. For that purpose in Figure 6-2 we provide the t-SNE [98] scatter plots that visualize the local-structures of the feature representations learned in those two cases. Note that the visualized features are from the validation categories of the Mini-ImageNet dataset that are "unseen" during training. Also, in the case of the cosine-similarity based ConvNet recognition model, we visualize the l_2-normalized features, which are the features that are actually learned by the feature extractor.

We observe that the feature extractor learned with the cosine-similarity based ConvNet recognition model, when applied on the images of "unseen" categories (in this case the validation categories of Mini-ImageNet), generates features that form more compact and distinctive category-specific clusters (i.e., more discriminative features). Due to that, as it was argued in section §6.3.1, the features learned with the proposed cosine-similarity based recognition model generalize better on the "unseen" categories than the features learned with the typical dot-product based recognition model.

6.4.2 Few-shot benchmark of Hariharan & Girshick [51]

Here we evaluate our approach on the ImageNet based few-shot benchmark proposed by Hariharan and Girshick [51] using the improved evaluation metrics proposed by Wang et al. [157]. Briefly, this benchmark splits the ImageNet categories into 389 base categories and 611 novel categories; 193 of the base categories and 300 of the novel categories are used for cross validation and the remaining 196 base categories and 311 novel categories are used for the final evaluation (for more details we refer to [51]). We use the same categories split as they did. However, because it was not possible to use the same training images that they did for the novel categories[6], we sample ourselves N' training images per novel category and, similar to them, evaluate using the images in the validation set of ImageNet. We repeat the above experiment 100 times (sampling each time a different set of training images for the novel categories) and report in Table 6.3 the mean accuracies and the 95% confidence intervals for the recognition accuracy metrics proposed in [157].

Implementation details of training procedure of the proposed approach. During the 1st training stage, the recognition model was trained for 100 epochs using a stochastic gradient descent optimizer with momentum 0.9 and weight decay $5e-4$. The learning rate was set to 10^{-1} for the first 30 epochs, then dropped to 10^{-2} for the next 30 epochs, then again dropped to 10^{-3} for the next 30 epochs, and finally again dropped to 10^{-4} for the remaining 10 epochs. Each training epoch lasted for 4000 training batches of size 400. During the 2nd training stage, our model was trained for 6 epochs using a stochastic gradient descent optimizer with momentum 0.9 and weight decay $5e-4$. The learning rate was set to 10^{-2} for the first 4 epochs and then dropped to 10^{-3} for the final 2 epochs. Each training epoch lasted for 4000 training batches. Each training batch included a single "training episode" during which, 250 "fake" novel categories were sampled from the 389 base categories, $T_{novel} = 1500$ test images were sampled from the "fake" novel categories, and $T_{base} = 750$ test images were sampled from the remaining base categories. Since the feature extractor remains "frozen" during the 2nd training stage, we speed-up this training stage by pre-computing and cashing to the hard disk the feature vectors of all ImageNet

[6]It was not possible to establish a correspondence between the index files that they provide and the ImageNet images

images. For more implementation details we refer to the implementation code of this chapter.

Comparison to prior and concurrent work. We compare our full system (Cosine Classifier & Att. Weight Gen entry) against prior work, such as Prototypical-Nets [145], Matching Networks [155], and the work of Hariharan and Girshick [51]. We also compare against the work of Wang et al. [157], which is concurrent to ours. We observe that in all cases our approach achieves superior performance than prior approaches and even exceeds (in all but one cases) the Prototype Matching Net [157] based approaches that are concurrent to our work.

Feature extractor: The feature extractor of all approaches is implemented with a *ResNet-10* [57] network architecture[7] that gets as input images of 224×224 size. Also, when training the attention based few-shot classification weight generator component of our model (2nd training stage) we found helpful to apply dropout with 0.5 probability on the feature vectors generated by the feature extractor.

6.5 Conclusions

In this chapter we proposed a dynamic few-shot object recognition system that is able to quickly learn novel categories without forgetting the base categories on which it was trained, a property that most prior approaches on the few-shot learning task neglect to fulfill. To achieve that goal we proposed a novel attention based few-shot classification weight generator as well as a cosine-similarity based ConvNet classifier. This allows our system to recognize in a unified way both novel and base categories and also leads to learn feature representations with better generalization capabilities. We evaluated our framework on Mini-ImageNet and the recently introduced fews-shot benchmark of Hariharan and Girshick [51] where we demonstrate that our approach is capable of both maintaining high recognition accuracy on base categories and to achieve excellent few-shot recognition accuracy on novel categories that surpasses prior state-of-the-art approaches by a significant margin.

[7] Similar to what it is already explained, our model does not include the last ReLU non-linearity of the *ResNet-10* feature extractor

Chapter 7

Conclusions

7.1 Contributions

In this thesis we developed a series of deep learning based approaches that aimed on improving the effectiveness of image understanding tasks (such as object recognition, object detection, and pixel-wise image labeling) as well as making them less dependent on the availability of large-size manually labeled training datasets.

More specifically, in order to boost the recognition aspect of object detection systems we proposed a multi-region and semantic segmentation aware ConvNet-based image representation. We integrated this recognition module on an iterative localization mechanism and the resulting detection system achieved state-of-the-art results on PASCAL detection challenge, surpassing prior approaches by a significant margin. The proposed idea of using multiple regions in order to enhance the recognition performance of object detectors, has inspired subsequent work in object detection [164, 169] as well as in action recognition [119] and weakly supervised object localization [69]. Furthermore, key ingredients of the localization methodology that was proposed in this work has been used from several object detectors [5, 168, 56, 20] in order to boost their detection performance when participating in the COCO detection challenge [91, 23].

We improved the localization accuracy of object detection systems by proposing a novel localization model, called *LocNet*, that formulates the bounding box localization problem as a dense classification task (instead of a bounding box regression one). The

proposed *LocNet* based detection systems managed to achieve significant improvements on the mAP for high IoU thresholds while also being more robust w.r.t. the quality of the box proposals used as input to the detection system. We also adapted the above two localization techniques (i.e, iterative localization and the LocNet module) to the category agnostic box proposal generation task and the resulting system, called *AttractioNet*, achieved excellent box proposal results. By feeding those *AttractioNet* box proposals to one of our *LocNet* based detection systems we achieved state-of-the-art result among the VGG16-Net based methods in the COCO detection challenge. In the COCO 2016 detection challenge [23] our submission based on *AttractioNet* box proposals and the *LocNet* localization model achieved state-of-the-art results among single model submissions (i.e., submissions that do not use ensemble of models), while some other top-ranked entries also used *AttractionNet* proposals [23].

In the pixel-wise labeling problem, we explored several deep neural network architectures that perform structured prediction by learning to (iteratively) improve some initial (but inaccurate) estimates of the output labels. The goal is to identify what is the optimal architecture for implementing such deep structured prediction models. In this context, we propose to decompose the label improvement task into three steps: 1) detecting the initial label estimates that are incorrect, 2) replacing the incorrect labels with new ones, and finally 3) refining the renewed labels by predicting residual corrections w.r.t. them. We extensively evaluated those architectures to the disparity estimation task (stereo matching) and the proposed one, called *Detect, Replace, Refine*, achieved state-of-the-art results on the KITTI 2015 benchmark.

In order to reduce the dependence of deep learning based approaches on large-size manually labeled training dataset, we proposed a self-supervised representation learning approach that learns semantic ConvNet-based image features by training the ConvNet to recognize the 2d rotation applied to an image. Despite the simplicity of the proposed approach, the learned features exhibit very good results when transferred on the vision tasks of object detection and semantic segmentation, surpassing prior unsupervised learning approaches and thus narrowing the gap with the supervised case.

Finally, in order to achieve our goal of annotation efficient learning we also worked

on the subject of few-shot object recognition. Specifically, we proposed a few-shot object recognition system that learns to dynamically generate classification weights of visual categories given as input a few examples of them. Thanks to this meta-learning mechanism our system is able after its training to learn novel (unseen during training) categories from only a few training data. Furthermore, unlike most previous approaches, our system has the ability to not forget the categories on which it was trained when learning novel ones. Experimental results on few-shot benchmarks showed that our approach outperforms previous state-of-the-art methods.

7.2 Future work

In this thesis we focused our efforts on advancing the state-of-the-art on the image understanding tasks of object detection and pixel-wise image labeling w.r.t. the accuracy with which those tasks are performed. However, in order to effectively employ deep learning based solutions to mobile devices or other embedded systems (e.g., in self-driving cars, or autonomous robots) it is also very crucial to decrease the computational and memory complexity of such approaches without significantly hurting their accuracy. For example, in case such image understanding models are going to be applied to videos, a promising research avenue is to explore how to exploit the temporal dimension of the data in order to decrease the per-frame computational cost of the models while at the same time their accuracy remains the same. Apart from their speed, another practical limitation of deep neural networks is that the are highly specialized to a single task and visual domain. Instead, it would be desirable to have image understanding models that perform well on multiple tasks and domains simultaneously without any performance loss compared to image understanding models dedicated to a specific task and domain.

Many opportunities for improvement and alternatives to be explored exist also in the annotation efficient objective of our work. For instance, more research should be conducted on the problem of unsupervised image representation learning using still images. Also, very interesting and promising research directions for unsupervised image representation learning are those of exploiting the structure present in the temporal dimension of videos, exploiting

extra sensor modalities, such as audio and / or textual inputs, together with the visual modality, or by employing autonomous agents that learn to interact with their environment (instead of only passively observing it). In the topic of few-shot learning, further research is required on how to effectively store and exploit past visual experiences in memory augmented neural networks. Related work has mostly focused on storing and exploiting short-term memories [155, 105, 38, 68, 131], i.e., storing the few training examples of a novel category in order to facilitate its recognition when a test example is presented. However, the visual experiences seen during the training phase of the network (i.e., the training examples of training categories) usually do not participate in the learning process of a novel category. In our work we attempted to exploit such past visual experiences by proposing to infer the classification weights of a novel category via an attention mechanism over the corresponding weights of the training categories. Despite the success of our approach we believe that there is still large room for improvement at this subject. Another interesting research direction is on devising learning mechanisms that would adapt also the feature representations when learning to recognize novel categories without over-fitting on its few-training examples and without forgetting the categories already learned by the model. Although, there is already important work in this subject [124, 36], we believe that it necessitates further exploration. Finally, it is important to extend such few-shot object recognition approaches to other (more complex) image understanding tasks, such as object detection and semantic segmentation.

Appendix A

Improving object localization accuracy in object detection

A.1 Object detection pipeline

In Algorithm 1 of section §3.3 we provide the pseudo-code of the object detection pipeline that we adopt. For clarity purposes, the pseudo-code that is given corresponds to the single object category detection case and not the multiple object categories case that we are dealing with. Moreover, the actual detection algorithm, after scoring the candidate boxes for the first time $t = 1$, prunes the candidate boxes with low confidence in order to reduce the computational burden of the subsequent iterations. For that purpose, we threshold the candidate boxes of each category such that their average number per image and per category is around 18 boxes. Also, during this step, non-max-suppression with IoU of 0.95 is applied in order to remove near duplicate candidate boxes (in the case of using sliding windows to generate the initial set of candidate boxes this IoU threshold is set to 0.85). A more detailed algorithm of our detection pipeline is presented in Algorithm 2. Note that, since the initial candidate boxes $\{\mathbf{B}_c^1\}_{c=1}^C$ are coming from a category-agnostic bounding box proposal algorithm, those boxes are the same for all the categories and when applying on them (during $t = 1$ iteration) the recognition module, the computation between all the categories can be shared.

Algorithm 2: Object detection pipeline

Input : Image $\mathbf{I}$, initial set of candidate boxes $\{\mathbf{B}_c^1\}_{c=1}^C$
Output : Final list of per category detections $\{\mathbf{Y}_c\}_{c=1}^C$

```
for t ← 1 to T do
    for c ← 1 to C do
        S_c^t ← Recognition(B_c^t|I, c)
        if t == 1 then
            {S_c^t, B_c^t} ← PruneCandidateBoxes({S_c^t, B_c^t})
        end
    end
    if t < T then
        for c ← 1 to C do
            B_c^{t+1} ← Localization(B_c^t|I, c)
        end
    end
end
for c ← 1 to C do
    D_c ← ∪_{t=1}^T {S_c^t, B_c^t}
    Y_c ← PostProcess(D_c)
end
```

A.2 Multi-threshold non-max-suppresion re-ordering

As already described in section 3.4.1, at the end of our active box proposal generation strategy we include a non-maximum-suppression [35] (NMS) step that is applied on the set $\mathbf{C}$ of scored candidate box proposals in order then to take the final top K output box proposals (see algorithm 2). However, the optimal IoU threshold (in terms of the achieved AR) for the NMS step depends on the desired number K of output box-proposals. For example, for 10, 100, 1000 and 2000 proposals the optimal IoU thresholds are 0.55, 0.75, 0.90 and 0.95 respectively. Since our plan is to make our box proposal system publicly available, we would like to make its use easier for the end user. For that purpose, we first apply on the set $\mathbf{C}$ of scored candidate box proposals a simple NMS step with IoU threshold equal to 0.95 in order to then get the top 2000 box proposals and then we follow a multi-threshold non-max-suppression technique that re-orders this set of 2000 box proposals such that for any given number K the top K box proposals in the set better cover (in terms of achieved AR) the objects in the image.

Specifically, assume that $\{t_i\}_{i=1}^{N_k}$ are the optimal IoU thresholds for N_k different desired numbers of output box proposals $\{K_i\}_{i=1}^{N_k}$, where both the thresholds and the desired number of box proposals are in ascending order[1]. Our multi-threshold NMS strategy starts by applying on the aforementioned set $\mathbf{L}$ of 2000 box proposals simple single-threshold NMS steps with IoU thresholds $\{t_i\}_{i=1}^{N_k}$ that results on N_k different lists of box proposals $\{\mathbf{L}(t_i)\}_{i=1}^{N_k}$ (note that all the NMS steps are applied on the same list $\mathbf{L}$ and not in consecutive order). Then, starting from the lowest threshold t_1 (which also is the more restrictive one) we take from the list $\mathbf{L}(t_1)$ the top K_1 box proposals and we add them to the set of output box proposals $\mathbf{P}$. For the next threshold t_2 we get the top $K_2 - |\mathbf{P}|$ box proposals from the set $\{\mathbf{L}(t_2) \setminus \mathbf{P}\}$ and again add them on the set $\mathbf{P}$. This process continues till the last threshold t_{N_k} at which point the size of the output box proposals set $\mathbf{P}$ is $K_{N_k} = 2000$. Each time $i = 1, ..., N_k$ we add box proposals on the set $\mathbf{P}$, their objectness scores are altered according to the formula $\tilde{o} = o + (N_k - i)$ (where o and $\tilde{o}$ are the initial and after re-ordering objectness scores correspondingly) such that their new objectness scores to correspond to the order at which they are placed in the set $\mathbf{P}$. Note that this technique does not guarantee an optimal re-ordering of the boxes (in terms of AR), however it works sufficiently well in practice.

A.3 Common categories between ImageNet and COCO

In this section we list the ImageNet detection task object categories that we identified to be present also in the COCO dataset. Those are:

airplane, apple, backpack, baseball, banana, bear, bench, bicycle, bird, bowl, bus, car, chair, cattle, computer keyboard, computer mouse, cup or mug, dog, domestic cat, digital clock, elephant, horse, hotdog, laptop, microwave, motorcycle, orange, person, pizza, refrigerator, sheep, ski, tie, toaster, traffic light, train, zebra, racket, remote control, sofa, tv or monitor, table, watercraft, washer, water bottle, wine bottle, ladle, flower pot, purse, stove, koala bear, volleyball, hair dryer, soccer ball,

[1]We used the IoU thresholds of $\{0.55, 0.60, 0.65, 0.75, 0.80, 0.85, 0.90, 0.95\}$ for the desired numbers of output box proposals $\{10, 20, 40, 100, 200, 400, 1000, 2000\}$. Those IoU thresholds were cross-validated on a validation set different than this used for the evaluation of our approach.

rugby ball, croquet ball, basketball, golf ball, ping-pong ball, tennis ball.

A.4 Ignored NUY Depth dataset categories

In this section we list the 12 most frequent non-object categories that we identified on the NUY Depth V2 dataset:
curtain, cabinet, wall, floor, ceiling, room divider, window shelf, stair, counter, window, pipe and column.

Bibliography

[1] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015.

[2] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, 2012.

[3] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[4] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 328–335, 2014.

[5] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2874–2883, 2016.

[6] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, pages 153–160, 2007.

[7] Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, pages 517–526, 2017.

[8] Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2488–2496, 2015.

[9] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4733–4742, 2016.

[10] Neelima Chavali, Harsh Agrawal, Aroma Mahendru, and Dhruv Batra. Object-proposal evaluation protocol is' gameable'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 835–844, 2016.

[11] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *International Conference on Learning Representations*, 2015.

[12] Liang-Chieh Chen, Alexander G Schwing, Alan L Yuille, and Raquel Urtasun. Learning deep structured models. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, pages 1785—-1794, 2015.

[13] Xiaozhi Chen, Huimin Ma, Xiang Wang, and Zhichen Zhao. Improving object proposals with multi-thresholding straddling expansion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2587–2595, 2015.

[14] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3286–3293, 2014.

[15] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Weakly supervised object localization with multi-fold multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):189–203, 2017.

[16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[17] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. Instance-sensitive fully convolutional networks. In *European Conference on Computer Vision*, pages 534–549. Springer, 2016.

[18] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3992–4000, 2015.

[19] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

[20] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 764–773, 2017.

[21] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[23] MS-COCO detection challenge leaderboard. `http://cocodataset.org/#detection-leaderboard`.

[24] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[25] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060, 2017.

[26] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *International Conference on Learning Representations*, 2017.

[27] Jian Dong, Qiang Chen, Shuicheng Yan, and Alan Yuille. Towards unified object detection and semantic segmentation. In *European Conference on Computer Vision*, pages 299–314. Springer, 2014.

[28] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774, 2014.

[29] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[30] Nils Einecke and Julian Eggert. A multi-block-matching approach for stereo. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 585–592. IEEE, 2015.

[31] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[32] M Everingham, L Van Gool, Chris Williams, J Winn, and A Zisserman. The pascal visual object classes challenge 2012, 2012.

[33] M Everingham, L Van Gool, CKI Williams, J Winn, and A Zisserman. The pascal visual object classes challenge 2007 (voc 2007) results (2007), 2008.

[34] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013.

[35] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[36] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, pages 1126–1135, 2017.

[37] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[38] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *International Conference on Learning Representations*, 2018.

[39] Amir Ghodrati, Ali Diba, Marco Pedersoli, Tinne Tuytelaars, and Luc Van Gool. Deepproposal: Hunting objects by cascading deep convolutional layers. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2578–2586, 2015.

[40] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region & semantic segmentation-aware cnn model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1134–1142, 2015.

[41] Spyros Gidaris and Nikos Komodakis. Locnet: Improving localization accuracy for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 789–798, 2016.

[42] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[43] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[44] Abel Gonzalez-Garcia, Alexander Vezhnevets, and Vittorio Ferrari. An active search strategy for efficient object class detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3022–3031, 2015.

[45] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[46] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[47] Fatma Guney and Andreas Geiger. Displets: Resolving stereo ambiguities using object knowledge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4165–4175, 2015.

[48] Jian Guo and Stephen Gould. Deep cnn ensemble with data augmentation for object detection. *arXiv preprint arXiv:1506.07224*, 2015.

[49] Saurabh Gupta and Jitendra Malik. Visual semantic role labeling. *arXiv preprint arXiv:1505.04474*, 2015.

[50] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[51] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3037–3046, 2017.

[52] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 2016.

[53] Zeeshan Hayder, Xuming He, and Mathieu Salzmann. Learning to co-generate object proposals with a deep structured network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2565–2573, 2016.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[56] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Tech report: Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[58] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.

[59] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[60] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[61] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *European Conference on Computer Vision*, pages 340–353. Springer, 2012.

[62] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[63] J. Hosang, R. Benenson, and B. Schiele. How good are detection proposals, really? In *Proceedings of the British Machine Vision Conference*, 2014.

[64] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):814–830, 2016.

[65] Fu Jie Huang, Y-Lan Boureau, Yann LeCun, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[66] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, pages 448–456, 2015.

[67] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014.

[68] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *International Conference on Learning Representations*, 2017.

[69] Vadim Kantorov, Maxime Oquab, Minsu Cho, and Ivan Laptev. Contextlocnet: Context-aware deep network models for weakly supervised localization. In *European Conference on Computer Vision*, pages 350–365. Springer, 2016.

[70] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[71] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[72] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[73] Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in Neural Information Processing Systems*, pages 109–117, 2011.

[74] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. *International Conference on Learning Representations*, 2016.

[75] Philipp Krähenbühl and Vladlen Koltun. Geodesic object proposals. In *European Conference on Computer Vision*, pages 725–739. Springer, 2014.

[76] Philipp Krähenbühl and Vladlen Koltun. Learning to propose objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1574–1582, 2015.

[77] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[78] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[79] Weicheng Kuo, Bharath Hariharan, and Jitendra Malik. Deepbox: Learning objectness with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2479–2487, 2015.

[80] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning, ICML*, pages 282–289, 2001.

[81] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision*, pages 577–593. Springer, 2016.

[82] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6874–6883, 2017.

[83] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[84] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[85] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th International Conference on Machine Learning, ICML*, pages 609–616, 2009.

[86] K. Lenc and A. Vedaldi. R-cnn minus r. In *Proceedings of the British Machine Vision Conference*, 2015.

[87] Marius Leordeanu, Alexandra Radu, and Rahul Sukthankar. Features in concert: Discriminative feature selection meets unsupervised clustering. *arXiv preprint arXiv:1411.7714*, 2014.

[88] Ke Li, Bharath Hariharan, and Jitendra Malik. Iterative instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3659–3667, 2016.

[89] Renjie Liao, Alex Schwing, Richard Zemel, and Raquel Urtasun. Learning deep parsimonious representations. In *Advances in Neural Information Processing Systems*, pages 5076–5084, 2016.

[90] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[91] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[92] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[93] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[94] Yongxi Lu, Tara Javidi, and Svetlana Lazebnik. Adaptive object detection using adjacency and zoom prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2351–2359, 2016.

[95] Chunjie Luo, Jianfeng Zhan, Xiaohe Xue, Lei Wang, Rui Ren, and Qiang Yang. Cosine normalization: Using cosine similarity instead of dot product in neural networks. *International Conference on Artificial Neural Networks*, pages 382–391, 2018.

[96] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016.

[97] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning, ICML*, 2013.

[98] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[99] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59, 2011.

[100] Francisco Massa, Bryan C Russell, and Mathieu Aubry. Deep exemplar 2d-3d detection by adapting from real to rendered views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6024–6033, 2016.

[101] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.

[102] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *European Conference on Computer Vision*, pages 488–501. Springer, 2012.

[103] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015.

[104] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[105] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*, 2017.

[106] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 891–898, 2014.

[107] Tsendsuren Munkhdalai and Hong Yu. Meta networks. *arXiv preprint arXiv:1703.00837*, 2017.

[108] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning, ICML*, pages 807–814, 2010.

[109] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483—-499. Springer, 2016.

[110] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[111] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[112] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5898–5906, 2017.

[113] Wanli Ouyang, Ping Luo, Xingyu Zeng, Shi Qiu, Yonglong Tian, Hongsheng Li, Shuo Yang, Zhe Wang, Yuanjun Xiong, Chen Qian, et al. Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection. *arXiv preprint arXiv:1409.3505*, 2014.

[114] Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5618–5627, 2017.

[115] Edouard Oyallon and Stéphane Mallat. Deep roto-translation scattering for object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2865–2873, 2015.

[116] George Papandreou, Iasonas Kokkinos, and Pierre-André Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 390–399, 2015.

[117] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2701–2710, 2017.

[118] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[119] Xiaojiang Peng and Cordelia Schmid. Multi-region two-stream r-cnn for action detection. In *European Conference on Computer Vision*, pages 744–759. Springer, 2016.

[120] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollar. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015.

[121] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016.

[122] Hang Qi, Matthew Brown, and David G Lowe. Learning with imprinted weights. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5822–5830, 2018.

[123] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *International Conference on Learning Representations*, 2016.

[124] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2017.

[125] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[126] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[127] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[128] Shaoqing Ren, Kaiming He, Ross Girshick, Xiangyu Zhang, and Jian Sun. Object detection networks on convolutional feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1476–1481, 2017.

[129] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[130] Chris Russell, Pushmeet Kohli, Philip HS Torr, et al. Associative hierarchical crfs for object class image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 739–746, 2009.

[131] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *Proceedings of the 33rd International Conference on Machine Learning, ICML*, pages 1842–1850, 2016.

[132] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016.

[133] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition*, pages 31–42. Springer, 2014.

[134] Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, 1997.

[135] Alexander G Schwing and Raquel Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.

[136] Akihito Seki and Marc Pollefeys. Patch based confidence prediction for dense disparity map. In *Proceedings of the British Machine Vision Conference*, 2016.

[137] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *International Conference on Learning Representations*, 2014.

[138] Kevin J Shih, Saurabh Singh, and Derek Hoiem. Where to look: Focus regions for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4613–4621, 2016.

[139] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[140] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56:116–124, 2013.

[141] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009.

[142] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016.

[143] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[144] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

[145] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077—-4087, 2017.

[146] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[147] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[148] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[149] Christian Szegedy, Scott Reed, Dumitru Erhan, and Dragomir Anguelov. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*, 2014.

[150] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561, 2013.

[151] Olivier Teboul, Iasonas Kokkinos, Loic Simon, Panagiotis Koutsourakis, and Nikos Paragios. Shape grammar parsing via reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2273–2280, 2011.

[152] Sebastian Thrun. Lifelong learning algorithms. *Learning to learn*, 8:181–209, 1998.

[153] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. Segmentation as selective search for object recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1879—-1886, 2011.

[154] Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, 2009.

[155] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[156] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

[157] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7278–7286, 2018.

[158] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *European Conference on Computer Vision*, pages 756–771. Springer, 2014.

[159] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

[160] Donggeun Yoo, Sunggyun Park, Joon-Young Lee, Anthony S Paek, and In So Kweon. Attentionnet: Aggregating weak directions for accurate object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2659–2667, 2015.

[161] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *International Conference on Learning Representations*, 2016.

[162] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4353–4361, 2015.

[163] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[164] Sergey Zagoruyko, Adam Lerer, Tsung-Yi Lin, Pedro O Pinheiro, Sam Gross, Soumith Chintala, and Piotr Dollár. A multipath network for object detection. In *Proceedings of the British Machine Vision Conference*, 2016.

[165] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1592–1599, 2015.

[166] Jure Žbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1):2287–2318, 2016.

[167] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[168] Xingyu Zeng, Wanli Ouyang, Junjie Yan, Hongsheng Li, Tong Xiao, Kun Wang, Yu Liu, Yucong Zhou, Bin Yang, Zhe Wang, et al. Crafting gbd-net for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(9):2109–2123, 2018.

[169] Xingyu Zeng, Wanli Ouyang, Bin Yang, Junjie Yan, and Xiaogang Wang. Gated bi-directional cnn for object detection. In *European Conference on Computer Vision*, pages 354–369. Springer, 2016.

[170] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

[171] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067, 2017.

[172] Yuting Zhang, Kihyuk Sohn, Ruben Villegas, Gang Pan, and Honglak Lee. Improving object detection with deep convolutional networks via bayesian optimization and structured prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 249–258, 2015.

[173] Qiyang Zhao, Zhibin Liu, and Baolin Yin. Cracking bing and beyond. In *Proceedings of the British Machine Vision Conference*, 2014.

[174] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

[175] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495, 2014.

[176] Yukun Zhu, Raquel Urtasun, Ruslan Salakhutdinov, and Sanja Fidler. segdeepm: Exploiting segmentation and context in deep neural networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4703–4711, 2015.

[177] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer, 2014.