Learning with Limited Annotated Data for Visual Understanding

Mikita Dvornik

Communauté UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Mikita Dvornik

Thèse dirigée par **Cordelia Schmid**, Directeur de Recherche , INRIA, et
codirigée par **Julien Mairal,** Chargé de Recherche, INRIA

préparée au sein du **Laboratoire Jean Kuntzmann**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Learning with Limited Annotated Data for Visual Understanding

Apprentissage avec des données annotées limitées pour la vision par ordinateur

Thèse soutenue publiquement le **26 novembre 2019**,
devant le jury composé de :

Monsieur Martial Hebert
Professeur, The Robotics Institute School of Computer Science Carnegie Mellon University, Rapporteur
Monsieur, Andrea Vedaldi
Associate Professor, Oxford University , Rapporteur
Madame, Naila Murray
Directeur scientifique, Naver Labs Europe, Examinateur
Monsieur, Vincent Lepetit
Professeur, Université de Bordeaux, Président
Madame Cordelia Schmid
Directrice de Recherche, INRIA Centre de Grenoble Rhône-Alpes, Directeur de thèse
Monsieur Julien Mairal
Chargé de Recherche, INRIA Centre de Grenoble Rhône-Alpes, Co-Directeur de thèse

Abstract

The ability of deep-learning methods to excel in computer vision highly depends on the amount of annotated data available for training. For some tasks, annotation may be too costly and labor intensive, thus becoming the main obstacle to better accuracy. Algorithms that learn from data automatically, without human supervision, perform substantially worse than their fully-supervised counterparts. Thus, there is a strong motivation to work on effective methods for learning with limited annotations. This thesis proposes to exploit prior knowledge about the task and develops more effective solutions for scene understanding and few-shot image classification.

Main challenges of scene understanding include object detection, semantic and instance segmentation. Similarly, all these tasks aim at recognizing and localizing objects, at region- or more precise pixel-level, which makes the annotation process difficult. The first contribution of this manuscript is a Convolutional Neural Network (CNN) that performs both object detection and semantic segmentation. We design a specialized network architecture, that is trained to solve both problems in one forward pass, and operates in real-time. Thanks to the multi-task training procedure, both tasks benefit from each other in terms of accuracy, with no extra labeled data.

The second contribution introduces a new technique for data augmentation, i.e., artificially increasing the amount of training data. It aims at creating new scenes by copy-pasting objects from one image to another, within a given dataset. Placing an object in a right context was found to be crucial in order to improve scene understanding performance. We propose to model visual context explicitly using a CNN that discovers correlations between object categories and their typical neighborhood, and then proposes realistic locations for augmentation. Overall, pasting objects in "right" locations allows to improve object detection and segmentation performance, with higher gains in limited annotation scenarios.

For some problems, the data is extremely scarce, and an algorithm has to learn new concepts from a handful of examples. Few-shot classification consists of learning a predictive model that is able to effectively adapt to a new class, given only a few annotated samples. While most current methods concentrate on the adaptation mechanism, few works have tackled the problem of scarce training data explicitly. In our third contribution, we show that by addressing the fundamental high-variance issue of few-shot learning classifiers, it is possible to significantly outperform more sophisticated existing techniques. Our approach consists of designing an ensemble of deep networks to leverage the variance of the classifiers, and introducing new strategies to encourage the networks to cooperate, while encouraging prediction diversity. By matching different networks outputs on similar input images, we improve model accuracy and robustness, comparing to classical ensemble training. Moreover, a single network obtained by distillation shows similar to the full ensemble performance and yields state-of-the-art results with no computational overhead at test time.

Keywords: few-shot learning, data augmentation, visual context, object detection, semantic segmentation, ensemble methods, convolutional neural networks, computer vision, machine learning.

Résumé

Les capacités des méthodes d'apprentissage profond pour la vision par ordinateur dépendent de la quantité de données annotées disponibles. Or, pour certaines tâches d'apprentissage, l'annotation manuelle de ces données peut se révéler particulièrement chronophage. De plus, les algorithmes apprenant de manière automatisée, sans intervention/supervision humaine possèdent des performances en deçà de ceux entièrement supervisés. C'est pourquoi il est important de travailler sur des méthodes d'apprentissage efficaces, et requérant peu d'annotations. Cette thèse propose d'exploiter les connaissances à priori sur la tâche d'apprentissage et développe des solutions efficaces pour la compréhension de scènes et la classification d'images lorsque peu d'exemples sont disponibles. Les principaux défis de la compréhension de la scène incluent la détection d'objets, leurs sémantiques et leurs segmentations (découpages).

La première contribution de ce manuscrit consiste en un réseau de neurones convolutif (CNN) qui effectue la détection d'objets et la segmentation sémantique. Ces réseaux sont conçus à l'aide d'une architecture de réseau spécialisée, formée pour résoudre les deux objectifs lors d'une unique passe, en temps réel. Grâce à la procédure d'entrainment multi-tâches, les deux tâches d'apprentissage bénéficient les unes des autres en termes de précision, sans recourir à de nouvelles données étiquetées.

La deuxième contribution introduit une nouvelle technique d'augmentation de données, c'est à dire, augmenter artificiellement la quantité de données d'entraînement des réseaux. Cette augmentation vise à créer de nouvelles scènes dans un ensemble de données en déplaçant des objets d'une image à une autre. Placer un objet dans un contexte approprié s'est avéré être une amélioration majeure des performances des techniques de compréhension de la scène. Globalement, le collage d'objets aux bons emplacements permet d'améliorer les performances de détection et de segmentation d'objets, avec des gains plus élevés dans les scénarios avec annotations limitées.

La troisième contribution revoit un problème majeur des classificateurs apprenant à partir de peu d'exemples, celui du compromis biais-variance. Notre approche consiste à concevoir un ensemble de réseaux profonds afin de tirer parti de la variance des classificateurs et y introduire de nouvelles stratégies pour encourager les réseaux à coopérer, tout en assurant une certaine diversité de prédiction : en faisant correspondre des sorties de réseaux différents sur des entrées similaires, nous améliorons la précision et la robustesse des modèles. Notre approche améliore de manière significative les performances par rapport aux techniques existantes, tout en se relevant beaucoup moins complexe à mettre en œuvre.

Everything is interesting if you go into it deeply enough

Richard Feynman

Acknowledgements

I would like to thank my defense jury members and manuscript reviewers for their time and interesting scientific discussions. I would like to thank my supervisors that I admire and from whom I have learned so much: Cordelia for teaching me how to manage my research, always be curious and never give up; Julien for showing me the big and small picture of the field, for guiding me in the right direction and being there in difficult times. I would also like to thank my collaborator Konstantin for great team work and for teaching me so much. I want to thank my friends for their help and this amazing journey that we have walked together. Finally, I would love to express infinite gratitude to my family: to my parents for all their love and support and to my dear fiance for being the best teammate I could have ever had.

Contents

Chapter 1

Introduction

The world's most valuable
resource is no longer oil,
but data

— The Economist*, 2017*

We live in the era of big data. Every 60 seconds, on average, people send 40 million messages on Facebook, watch 4.5 million of videos on YouTube and scroll over 350K photos on Instagram [1]. Such statistics is quite recent; only in the last two years the world has generated more data than in the whole previous history of humanity. At this scale, people can no longer analyze the information manually and resort to algorithmic data processing. Machine learning and, in particularly, deep learning-based methods are able to make sense of raw data automatically. Such methods are based on training Deep Neural Networks (DNNs) and by now can fully or partially automate certain tasks in vision [92], natural language [36] and audio processing [175].

We owe this success not only to the learning algorithms, but also to human supervision they received during training. For example, in order to distinguish between cats and dogs, a neural network needs to see numerous examples of each category, with a corresponding label. Not surprisingly, a key component of deep learning revolution [92] was the availability of large annotated datasets [145]. Despite the fact that most computer vision datasets are labeled using crowd-

1. https://www.visualcapitalist.com/what-happens-in-an-internet-minute-in-2019

sourcing [21], the process is still costly and time-consuming, which is becoming a bottle-neck in deploying deep learning systems.

The central problem of computer vision is finding a “good” image representation. An image representation, also known as a descriptor or features, is a vector, where each entry indicates presence of a certain geometrical or semantic feature. That could be anything from average value of all pixels to the number of people present in a picture. Hand-crafted image representations [35, 112] where found to be sub-optimal [92] and were replaced by deep image representations provided by Convolutional Neural Networks (CNNs) and optimized from data. They achieve state-of-the-art accuracy on many vision tasks, such as image classification [76, 92, 156], object detection [75, 110, 142], instance and semantic segmentation [25, 75, 111]. A common point between these tasks is, again, availability of large labeled datasets.

Deep convolutional neural networks have higher model capacity than standard tools for image processing [26]; they can extract more complex patterns from data and store them in larger quantities. As a result, training them on even bigger datasets is more rewarding [115, 162]. The problem is that annotating more data is not always feasible. One way to get around this limitation is to use unsupervised learning [23, 86], which aims at discovering patterns in the data automatically, without human supervision. Another option is self-supervised learning [40, 60, 129] where the algorithms exploits structure of input data in order to generate learning signal. These algorithms are able to use millions of unlabeled images freely available on the web in order to learn image representations. Unfortunately, neither of the methods currently works better than training with full supervision even on smaller datasets [23]. Moreover, there is already a lot of labeled data for canonical tasks in computer vision, and we should take advantage of that. Therefore, one may instead concentrate the attention on how to make better use of limited annotated data.

One possible direction is to design less general systems dedicated to more specific problems. This allows to incorporate prior knowledge about a task or its input into the pipeline, which makes the final problem easier to solve. More importantly, prior knowledge does not require learning; it makes the algorithm more data efficient and potentially more accurate. For example, commonly used CNNs have translation invariance built in by default; all computations are performed locally in a sliding-window fashion. Thus, when working with images, CNNs have an advantage over simple feed-forward networks [98], where each neuron in one layer is connected to all neurons in the next layer with different weight. More subtle observations about a task and its input may allow for even more efficient use of available resources, as demonstrated by the modern scene understanding systems [106, 110, 111, 142]. Another way of using existing data wiser is to perform artificial data augmentation; a process of synthesizing new training examples by making little modifications to the original data. Even though we do not mine new information per se, generated images will be perceived as novel training samples by a CNN, and thus increase effective size of the training set. If right image transformations are selected, data augmentation is another way to teach a system invariance properties. For example, varying brightness of training images will make the system robust to illumination conditions. As it is the case for learning algorithms, more effective data augmentation techniques heavily rely on prior knowledge about a task and input structure [45, 58].

Availability of large labeled datasets cannot always be taken for granted. When the data is insufficient, the model is likely to overfit the training set, and perform poorly on test images. Overfitting happens when an algorithm memorizes training examples without learning smart relationships between the data and the labels. As a result, the algorithm learns little about the task and has troubles generalizing to new samples. To prevent overfitting and improve generalization, one needs to regularize model training. That could be done by penalizing model complexity [69], inserting random noise into training [160], or stopping the training procedure early [133]. Using data augmentation or incorporating prior knowledge is another powerful form of regularization [33,99], as it has proven useful not only for problems with little data. These techniques may be as effective in large-scale scenarios [78], mainly because they are designed agnostic to the problem size and instead aim at incorporating human expertise in algorithmic solutions. Thus, regardless the task and amount of available data, delving deep into the problem and revealing the insights is always rewarding.

1.1 Goals and Challenges

In this dissertation, we seek for more data-efficient methods and address two related but independent topics: scene understanding on object level and few-shot image classification. By addressing scene understanding on object level we mean solving object detection and segmentation, that aim at localizing and recognizing objects within an image, as well as classifying each pixel in it. Few-shot classification studies the problem of learning classifiers with only a handful of annotated samples (1 or 5 per category). We now briefly present these two tasks, as well as the challenges involved in each of them.

1.1.1 Unsolved Problems in Object-level Scene Understanding

Scene understanding is a central task of computer vision. By understanding a scene we mean to recognize objects comprising it, find their location, and estimate their shape. Obtaining this information is vital in autonomous driving, where a vehicle must be up to date with the situation on the road, anticipate potential

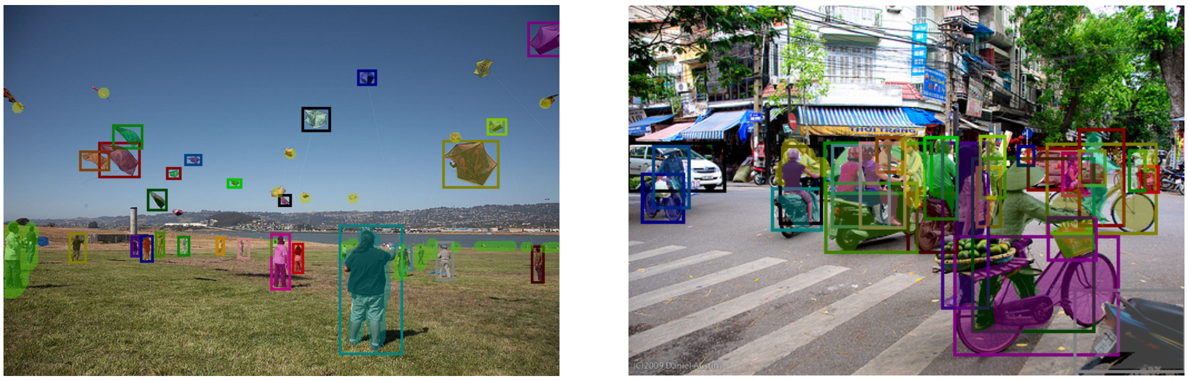


Figure 1.1 – Examples of images from the COCO dataset annotated for object detection and segmentation. This demonstrates how tedious the annotation process for scene understanding could be.

danger, and react promptly. Another application of scene understanding is in robotics, when performing object grasping. An agent needs to know object's location and shape in order to grasp properly.

To extract this information from an image, one needs to solve the tasks of object detection, semantic and/or instance segmentation. To detect an object means to localize an object with a tight bounding box and recognize its category. Segmentation is about classifying each pixel in an image as belonging to a particular class (semantic segmentation) or a particular object (instance segmentation). These are fundamental vision problems that have been studied for decades and are still an active research direction. Here, we elaborate on the main challenges in scene understanding and possible ways of approaching them.

Throughout the history of scene understanding, the community has created a number of extensively annotated datasets that helped advancing the field. Figure 1.1 gives an example of images from the COCO [108] dataset, annotated for detection and segmentation. Just one glance is enough to understand how *tedious and time-consuming* the *annotation process* may get. If drawing bounding boxes around objects does not seem too costly, labeling each pixel of thereof is a tremendous amount of work. The COCO dataset has 2,500,000 segmented objects, and according to [108], annotating 1000 segmentations required 22 worker hours. After making some trivial computations, a natural question arises by itself. Is it worth spending more time annotating even more images? Or should we discover new ways of using existing data more effectively?

In most object detection datasets, even densely annotated, background regions (negatives) still dominate the ones containing objects of interest (positives). This phenomena is called *positive/negative imbalance* and is illustrated in Figure 1.1. For object detection, it results in imbalanced classification problem on rectangular image

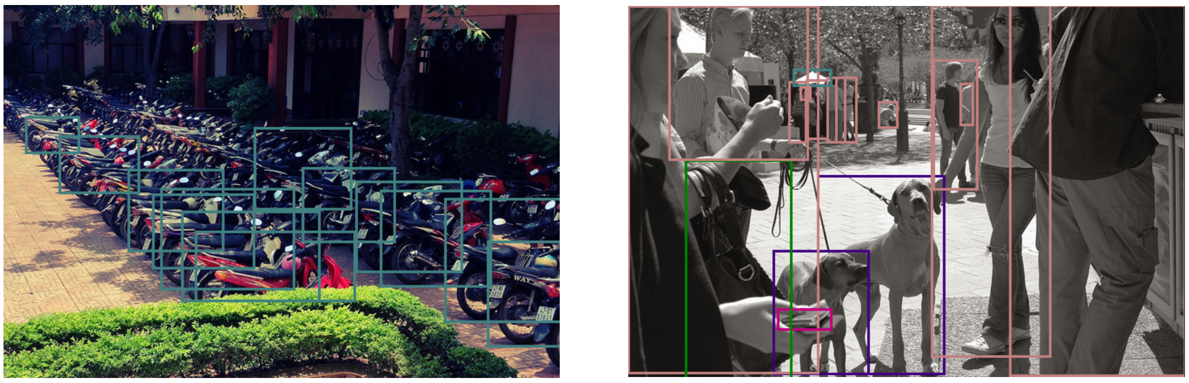


Figure 1.2 – Examples of complicated scenes from the COCO dataset. On the left image, the objects are truncated and occluded, in some areas overlapping to an extent that it is impossible to cover just one object with a single bounding box. On the right image, the humans in front have to be detected by a system as well as the humans on the background, despite the dramatic difference in their scale.

regions, and on pixels in case of segmentation. Moreover, since positive samples are scarce, a system overfits appearance of objects faster than that of background. One way to mitigate this problem is to re-sample positive and negative examples in a more balanced way [61], to re-weight loss corresponding to negative/positive samples [107], or to use data augmentation [110] to reduce overfitting and alleviate the imbalance.

More realistic datasets include numerous examples of *heavy object occlusion and truncation*. This is an old and ever persistent problem that arises in object detection and instance segmentation. On one hand, a detector must be able to identify an object by only its discriminative parts, if the rest of the object is covered. On the other hand, systematically detecting discriminative parts as a independent objects is undesirable when the full object is visible. Hence, to avoid this issue, a good detector must have a large field of view [72, 106] on the input image and have the notion of compositionality [50].

Another problem common for detection and segmentation is *scale variations*, i.e., when within one dataset, objects of the same category appear at completely different scales, from close-ups to bird's-eye view (see Figure 1.2). To successfully detect objects across all scales, the system must obtain descriptors that are invariant or equivariant to object's scale [157]. Moreover, small objects present an additional challenge for CNN-based methods. This is because precise local information tends to saturate deeper in a network due to downsampling operations. Hence, solving this challenge requires designing network architectures that pay equal attention to information at all scales [25, 72, 144].

Last but not least, the *role of visual context* is still unclear for the tasks of scene understanding. It is well known that humans heavily rely on visual context in parsing complicated scenes [10]. This alone is a solid motivation for using this clue in computer vision too. Modern vision systems, based on neural networks, implicitly model visual context by design since the receptive field of "artificial neurons" grows with the network's depth. This allows to include some context information into object descriptors, however the context range seems to be quite limited [113]. The attempts to model context on top of CNN's features explicitly did not contribute much to final detection accuracy [29, 30]. The reasons for such behavior are unclear, which motivates studying visual context in isolation in order to improve scene understanding.

1.1.2 Challenges in Few-shot Learning

Having a large annotated datasets in some scenarios is simply impossible. One of the reasons may be the cost of label acquisition as in drug discovery, or medical imaging. In other cases, data itself can be scarce as when classifying rare animal species. Here, the number of examples is limited naturally. The task where only a few labeled examples is provided for training is called few-shot learning.

In this manuscript, the problem is formulated as learning a classifier from only a few annotated samples (typically 1 to 5) per category, while having access to a bigger annotated dataset with related visual concepts. The goal of such formulation is to leverage available annotated data, even though not directly related to the target task, in the most effective way. Typically few-shot classification is approached in two stages. First, an algorithm has access only to an abundant annotated dataset of base categories and has to extract useful and transferable knowledge from it. At test time, the tasks is to leverage the knowledge acquired during the fist stage to build a classifier for new categories, given only a few samples of each class. The two main problems of few-shot learning are (1) small data support of the target classes and (2) different categories at training and testing. Both problems have several important implications that we now discuss.

Firstly, training machine learning systems on small datasets leads to *overfitting*. The model tends to memorize the training set and fails to do accurate predictions at test time. Addressing this issue by model regularization [19] can alleviate the problem, when more data is not available. Artificial data augmentation may help the situation as well [73]. However, there is a more fundamental issue with using little data for statistical reasoning; the variance of estimators increases as the number of samples goes down [54]. From which it follows that few-shot classifiers have inevitably *high variance*. As far as we know, using diversity in the data and estimated models is the only way to address this problem [37].

Secondly, classification problems dealing with different classes during training and testing are notoriously difficult to approach. An obvious issue is that we *can* *not apply a standard classification framework* here, i.e., simply build a classifier on training classes and apply it to test samples. Instead, we need to build a new classifier at test time, which is a much harder problem. The difficulty is caused by the *domain gap between training and testing data*. This means that the two data sets have different statistics, and features tailored to one set are not necessarily useful for the other one [15]. It is possible to narrow the gap between the two by using domain adaptation [56] or training more flexible features in the first place [138].

Apart from the issues related to small dataset size or to train-test data difference, there are difficulties caused by the combinations of these two factors and are specific to few-shot learning. That is, after training a rich visual model, one will not be able adapt it to new categories using few samples, because *traditional deep learning techniques fail* [52] here. For example, finetuning a network in such scenario results in "forgetting" the previous knowledge and overfitting the new data. To this end, a natural question arises: what kind of learning paradigms should we use to tackle few shot classification? This is an active research topic and this manuscript makes a step towards answering this question.

1.2 Contributions

In this thesis, we bring tree main contributions towards object-level scene understanding and few-shot learning. The first contribution introduces a new real-time pipeline that jointly performs object detection and semantic segmentation. This work has been published at ICCV'17 [44], and the pipeline was made available as an open-source package [121]. The second contribution, published in ECCV'18 [41], develops a new data augmentation method for scene understanding tasks, that is based on explicit context modeling by a CNN. The extension of this work was submitted to PAMI [43] and is under minor revision now. Both the context model and the augmentation pipeline are available online [122]. The third contribution proposes to solve few-shot learning problems by using ensemble methods, and develops more effective strategies for ensemble training. The work was accepted to ICCV'19 [42], and the software code was released [123]. In the following section, we describe each contribution in more detail.

Object detection and semantic segmentation with BlitzNet

The problem of scene understanding often requires to solve object detection and semantic segmentation together. As a result, most of the images in popular scene understanding datasets are annotated for both tasks. In this manuscript, we take advantage of this fact and turn the link between these annotations into an extra

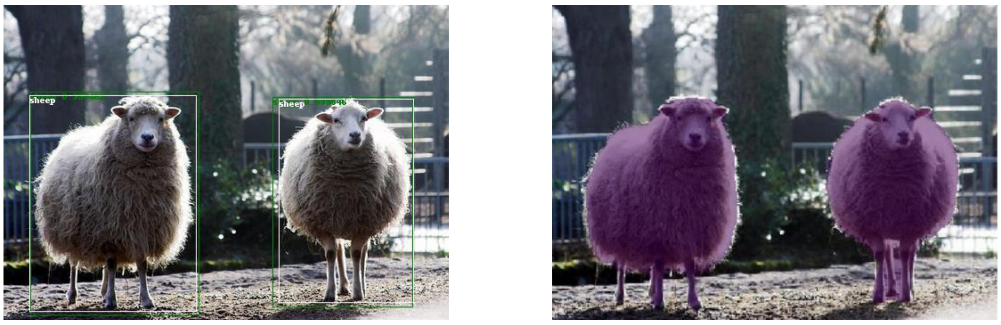


Figure 1.3 – An example of an image annotated for object detection (left) and semantic segmentation (right). In this simple case solving semantic segmentation automatically leads to solving object detection. On the other hand, providing bounding boxes gives a very strong clue for semantic segmentation.

source of information, as illustrated in Figure 1.3. The rest of the section describes current methods for detection and segmentation and our solution, combining them into a single pipeline.

Initial convolutional neural networks architectures for semantic segmentation and object detection were fairly different. First methods for object detection [61, 62, 142] operate in two stages. They first propose plausible object locations and compute their region features, and second, classify and refine each proposal independently using fully-connected networks. This two-staged pipeline is tailored specifically to instance-centric tasks and lacks flexibility. In contrast, semantic segmentation was initially solved using a fully-convolutional network [111] (without fully-connected layers), typically used for dense classification. Not only this framework naturally fits the task of semantic segmentation, its design does not introduce any constraints on the network's architecture and could be used to solve a wider range of problems. Different approaches for the two tasks find their common ground as new feature extractors are being developed. It turns out that both segmentation [111, 120, 124] and detection [14, 72, 106] benefit from multi-scale features with larger field of view, which further highlights similarity between the tasks. One-stage object detectors [110, 139] propose to tackle the problem with a fully-convolutional network which not only allows for simpler and faster detection but also opens up perspectives for a natural multi-task solution.

In this manuscript, we propose a CNN-based pipeline, called BlitzNet, that performs object detection and semantic segmentation simultaneously in one forward pass. We build our approach on top of the SSD detector [110] and extend the fully-convolutional feature extractor by adding deeper deconvolutional layers that combine high- and low- level information. This architecture not only provides better features and allows for real-time computations but naturally incorporates object

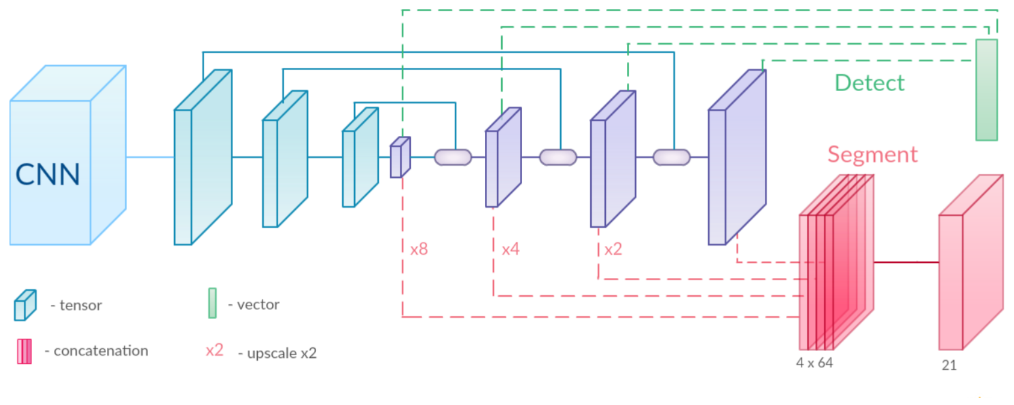


Figure 1.4 – The BlitzNet architecture, which performs object detection and segmentation with one fully convolutional network. On the left, CNN denotes a feature extractor, here ResNet-50 [76]; it is followed by the downscale-stream (in blue) and the last part of the net is the upscale-stream (in purple), which consists of a sequence of deconvolution layers interleaved with ResSkip blocks (see Figure 3.3). The localization and classification of bounding boxes (top) and pixelwise segmentation (bottom) are performed in a multiscale fashion by single convolutional layers operating on the output of deconvolution layers.

detection and semantic segmentation into one pipeline, as depicted in Figure 1.4. Besides the computational gain of having a single network to perform several tasks, we show that object detection and semantic segmentation benefit from each other in terms of accuracy. This is achieved by implicitly learning the correlation between task's annotations and allows to improve accuracy on one task (segmentation) by annotating only for the other one (detection). Experimental results for VOC and COCO datasets show state-of-the-art performance for object detection and segmentation among real time systems. The method is presented in Chapter 3.

The role of visual context for scene understanding

In the light of severe positive/negative region imbalance in scene understanding tasks, the problem of overfitting foreground region appearance needs to be addressed in a principal way. Since collecting and annotating new data may be costly or simply infeasible, an effective way to artificially increase the number of positive samples is of great importance. In this section, we summarize techniques used for this purpose so far and then present our approach for resolving this issue.

One possible way to tackle this problem is to create new images, either by using generative models [53, 130] or rendering purely synthetic scenes [68, 83, 116, 118, 161]. The major difficulty with such methods is that generated images may

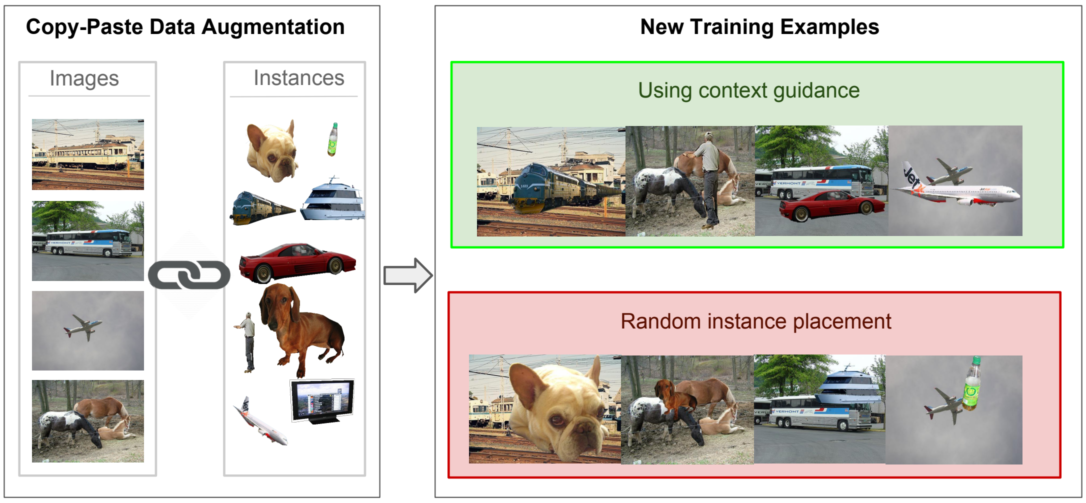


Figure 1.5 – Examples of new scenes synthesized using copy-paste data augmentation with context guidance. We compare images synthesized using our context model to the ones obtained by the random placements strategy. Even though, in either case the results are not perfectly photo-realistic and display blending artifacts, the visual context surrounding objects is more plausible with the explicit context model. Images and objects are taken from the VOC'12 dataset [47] using provided instance masks.

differ substantially from the real ones. Consequently, a system trained on synthetic data will not generalize in the real world. Therefore, a simpler and more reliable approach to obtain new "interesting" training examples is to modify existing images a.k.a. artificial data augmentation. Most of detection [110, 139, 142] and segmentation [25, 75] pipelines already include basic data augmentation such as random horizontal flipping, region cropping and color jittering. The authors of [66] go further and paste artificial text into natural images while accounting for local geometry. Creating new scenes this way reduces the presence of rendering artifacts, and allows to improves accuracy on text detection. A plausible way to augment instance detection datasets is to copy-paste an object into different images [45], using instance segmentation masks. This method places objects at random positions but handles blending artifacts carefully using boundary smoothing, which makes the training step robust to boundary artifacts at pasted locations.

In this thesis, we follow the copy-paste augmentation paradigm and generalize it to the tasks of object-level scene understanding, namely object detection, semantic and instance segmentation. In Chapter 4, we show that randomly pasting objects on images hurts the performance, unless the object is placed in the right context, as

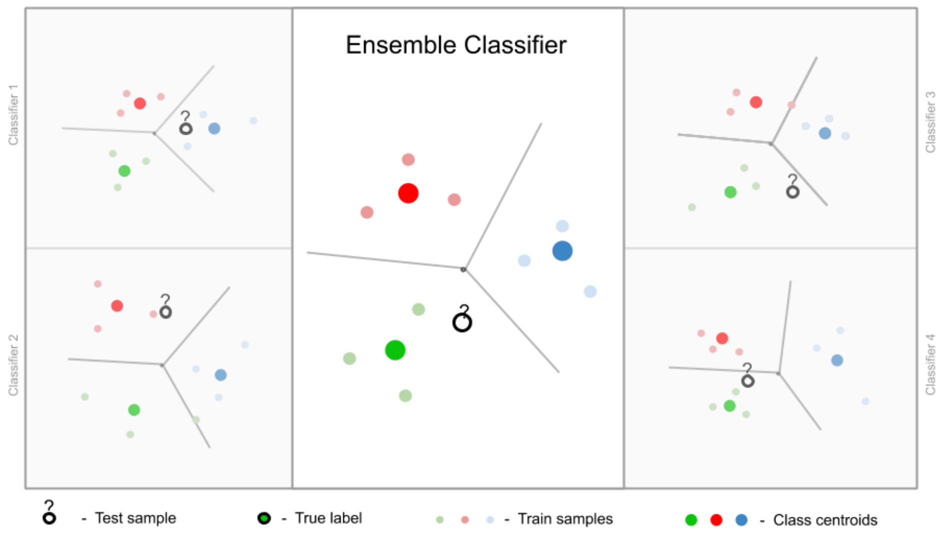


Figure 1.6 – A high-accuracy mean-centroid classifier (middle) obtained by ensembling diverse individual classifiers (left and right). When averaging class probabilities estimated by independent models, individual prediction errors "cancel out", which results in a more accurate final classifier.

illustrated in Figure 1.5. To resolve this issue, we propose an explicit context model by using a convolutional neural network, which predicts whether an image region is suitable for placing an object or not. In our experiments, we show that our approach is able to improve object detection, semantic and instance segmentation accuracy. The results are consistent across different models on PASCAL VOC12 and COCO datasets, with significant gains when few labeled examples are available. We also show that the method is not limited to datasets that come with expensive pixel-wise instance annotations and can be used when only bounding boxes are available, by employing weakly-supervised learning for instance masks approximation.

Few-shot learning with ensembles

As we have observed in the previous section, data augmentation methods are more helpful in scenarios when fewer annotated examples is available. However, when the number of training samples is very low, simple data augmentation is not enough to achieve good performance. This became the main motivation for us to study a problem with scarce data in isolation. The task of few-shot learning was initially considered in [12, 49] and recently experienced its renaissance thanks to the works of [96, 137, 172]. The authors pose a problem as few-shot classification with access to a large dataset of related visual concepts (base categories). Thus, the task is to effectively use the knowledge from extensively annotated concepts for a few-shot classification problem on new classes. The works of [52, 172] demonstrate that neural networks overfit base categories if standard training techniques are used. To tackle this problem, they propose using meta-learning [152, 164] as a method for regularization. Meta-learning extracts "transferable" knowledge from the base categories and adapts it to new classes. Current meta-learning based methods propose to learn a "good" network initialization [52], that adapts to new tasks in just a couple of gradient update steps. It is possible to learn the update rule itself [137] or to predict parameters of a classifier for new categories [18, 59, 135]. Another line of work proposes to learn a metric space suitable for comparison of both base and novel classes. It amounts to training a CNN embedding in a meta-learning fashion and then using it to build a distance-based classifier [159, 172].

A recent work [27] shows that state-of-the-art results could be achieved without using meta-learning or metric learning. They train a CNN classifier on base categories, remove the last classification layer and use the remaining CNN as a fixed feature extractor. When a few-shot problem is presented, a simple linear or cosine classifier, trained on top of the CNN's features, achieves performance on par with the state-of-the-art. This raises a question on whether the community needs to concentrate on algorithms that allow for more transferable knowledge or to investigate other issues related to the few-shot learning problem.

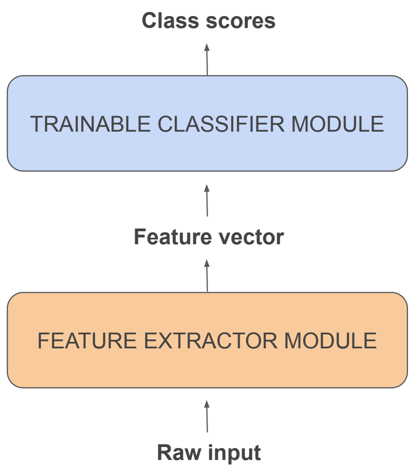
In Chapter 5 we propose to address the fundamental problem of high variance, that inevitably arises when training classifiers on little data. Our approach consists of designing an ensemble of deep networks to leverage the variance of individual predictors (see Figure 1.6) and obtain a final classifier with lower variance and higher accuracy. To achieve this goal we build a mean-centroid classifier on top of each CNN and combine their predictions by averaging soft-max activations. We go beyond independent training of networks and introduce new strategies to encourage the networks to cooperate, while maintaining predictions' diversity. This is achieved by penalizing the difference between output probabilities during ensemble training. Finally, we evaluate our approach on the mini-ImageNet, tiered-ImageNet and CUB datasets, where we show that even a single network obtained by distillation yields state-of-the-art results surpassing all other existing method by a considerable margin.

Chapter 2

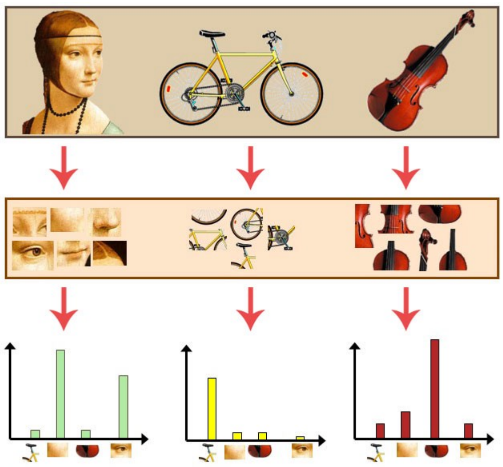
Background

Traditional computer vision algorithms consist of two independent stages illustrated in Figure 2.1. In the first stage, the goal is to extract visual features from data, i.e., to map raw pixels into a meaningful semantic representation. In the second stage, the goal is to build a machine learning model on top of the image representations in order to solve the task, i.e., to map feature vectors into labels. While the second step is common to any reasoning problem, the first stage is specific only to computer vision and had been an active research area for the last several decades. Classical feature descriptors aim at characterizing image patches in a way that is robust to shift, scale and illumination change. They are entirely hand-crafted and rely only on human's prior knowledge about the nature of visual data and downstream applications. The most popular examples of such features are SIFT [112] and HOG [35] descriptors. These are local features that describe image regions independently and need to be aggregated into a single vector to describe an image as a whole. One of the simplest ways to aggregate local feature into a global descriptor is Bag Of Words (BOW) [32]. It performs clustering on local descriptors and uses cluster sizes to describe the whole image (see Figure 2.1). Other methods build on top of this basic idea and propose more sophisticated ways of summarizing information about discovered clusters [81, 149].

The second stage aims at discovering statistical correlations between image descriptors and task labels. This is done by using machine learning methods to estimate a function that maps visual features to desired outputs. With the use of traditional learning techniques, such as SVM [31], kernel methods [153] or boosting [151], the community built successful vision systems for image and video classification [131], object detection [50] and image retrieval [131].



(a)



(b)

Figure 2.1 – (a) General computer vision pipeline for image classification. Firstly, the "feature extractor module" maps images to vectorized feature representations that are later used to train a classifier. (b) Visual explanation of the Bag-Of-Words aggregation strategy [32]. During the first stage, local visual descriptors are built and organized into clusters. The global descriptor is obtained by counting the number of visual words in each bin. It serves as an example of a feature extractor module in the general computer vision pipeline. Image source [1].

The work of [92] is considered to be a turning point in computer vision research. It proposes to use convolutional neural networks for the task of large-scale image classification [145] and outperforms all classical approaches by a big margin. CNNs predict an image class directly from raw pixels, thus merge the two steps of the classical pipelines into one. The whole system is optimized for the final task jointly, using backpropogation [98]. Not only the feature extractor is now learned from data, it extracts representations optimal for the classifier and makes its task easier. This is in contrast to classification of hand-crafted features in two-staged pipelines. Eventually, neural networks led to new state-of-the-art results on all standard vision tasks [25, 75, 76]. However, the price to pay for such success is enormous amount of annotated data required for learning [108, 119, 145], training time and computational resources.

Outline. In Section 2.1 we consider image classification with classical and CNN-base methods. Then, we discuss semantic segmentation, a task of dense classification, and present current approaches for solving in Section 2.2. Section 2.3 gives an overview of object detection and instance segmentation and discusses in details the main CNN-based pipelines for these problems. We cover modern data augmentation techniques in Section 2.4. Finally, Section 2.5 is dedicated to few-shot learning and existing frameworks to tackle this problem.

2.1 Image Classification with CNNs

Image classification is a problem of predicting an object's category, given it's image, where the category is chosen from a predefined set of labels. Image recognition is a central problem in computer vision that has driven progress in the field for decades. Most of the standard modern approaches for classification are built upon parametric statistical models [54]. Their goal is to explain the training data, i.e., given a training image, output a corresponding ground-truth label. To achieve this goal, suitable model's parameters are estimated using optimization procedure. For example, given a dataset of images and their labels $D_{train} = \{x_i, y_i\}_{i=1}^{N}$ we can find model's parameters θ by maximizing log-likelihood of the data under the model. To prevent overfitting the training set, the model should be regularized, for example by penalizing the L^2-norm of the parameters. This can be formalized as follows:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, L(D_{train}, \theta) + \lambda\|\theta\|_2^2 \tag{2.1}$$

$$L(D_{train}, \theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(x_i, y_i, \theta) = -\frac{1}{N}\sum_{i=1}^{N} log(p_\theta(x_i)). \tag{2.2}$$

Here, $p_\theta(x_i)$ is a predicted probability of image x_i belonging to a class y_i given the content of x_i. In classical computer vision, we would model $p_\theta(\cdot)$ in two steps, i.e., first by building a feature descriptor $d(x_i)$ for an image x_i and then fitting a machine learning model $g_\theta(d(x_i))$ on top. Usually, in such approaches, model $g_\theta(\cdot)$ doesn't have a lot of parameters, i.e., θ is a vector of moderate dimentionality. Moreover, the feature extractor $d(\cdot)$ does not have learnable parameters at all as it is hand-crafted. Hence, such pipelines do not require massive annotated datasets for training and reach their top performance with a fair amount of data.

An alternative way to tackle image recognition is by using a neural network $f_\theta(x)$ that directly maps inputs to predicted probabilities. Neural networks are typically composed of multiple layers, each performing a linear transformation of a previous layer's output, followed by a point-wise non-linearity. Such block structure allows to represent a neural network as a composition of simpler functions, where each function is parametrized independently: $f_\theta(x) = \sigma(f^n_{\theta_n}(\sigma(f^{n-1}_{\theta_{n-1}}(...f^1_{\theta_1}(x)...))))$ with $\theta^T = [\theta_1^T, ..., \theta_n^T]$. The work of [98] proposes to optimize the parameters θ directly for the final task, using backpropagation. This technique is called end-to-end training; it allows for simultaneous updates for all θ_i and makes training neural networks practical. The work of [99] established a new framework for image

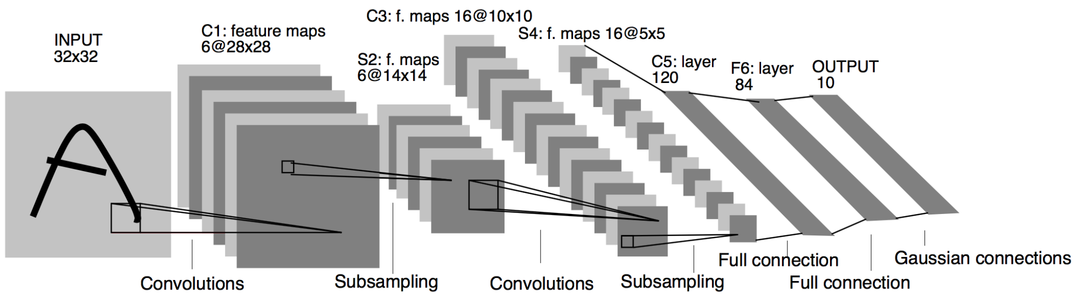


Figure 2.2 – The architecture of LeNet [99]. The network accept input images of size 32x32 and then applies a first convolution with a filter size 5x5 and 6 output channels followed by a 2x2 average pooling. This building block is repeated once and then followed by a series of fully-connected layers, predicting the class scores. Image source [99].

classification, called Convolutional Neural Networks (CNN). Figure 2.2 presents LeNet - the first CNN architecture. As the figure suggests, it is possible to split the network in two homogeneous parts: the first one consisting of convolutional, sub-sampling and non-linear layers interlaced with each other, and the second one being a sequence of fully-connected (linear) layers with non-linearities in between. This highlights the connection with the classical computer vision pipelines. Here, the first part of the network serves as a feature extractor while the second part implements a classifier. More formally:

$$\begin{aligned} d(x) &= \sigma(f^{k}_{\theta_n}(\sigma(f^{k-1}_{\theta_{n-1}}(...f^{1}_{\theta_1}(x)...)))) \\ g(x) &= \sigma(f^{n}_{\theta_n}(\sigma(f^{n-1}_{\theta_{n-1}}(...f^{k+1}_{\theta_{k+1}}(d(x))...)))), \quad \text{where } 1 < k < n \end{aligned}$$

On one hand, convolutional layers learn local correlation patterns and provide filters invariant to spatial locations. This design takes into account the nature of images and hence result in a powerful feature extractor. On the other hand, a multi-layer perceptron (multiple fully-connected layers stacked together) is a flexible, high-capacity model suitable for classification. Combined together, they represent a perspective pipeline for image recognition. An obvious drawback is a large number of parameters to be optimized, in both the feature extractor and the classifier.

The seminal work of [92] has shown that scaling up CNNs for image classification results in significant accuracy improvements over the previous state-of-the-art. That became possible thanks to advanced computational resources and availability of massive annotated datasets [145]. AlexNet architecture resembles LeNet [99], but it is deeper as it stacks more convolutional and pooling layers. The authors showed that using ReLU [67] as a nonlinearity speeds up training and inference.

For optimizing the parameters, [92] employs batch Stochastic Gradient Descent (SGD) with momentum and decreases the learning rate to improve convergence. To regularize the model, the authors used weight decay, dropout [160], and various data augmentation techniques, such as image translations, horizontal reflections, and random image crops. All computations were conducted in a parallel fashion to overcome memory limitations of existing GPUs. Nevertheless, it took almost a week for the model to fully converge. Combining all these ideas together revealed the true potential of neural networks and formed a basis for CNN design and training in the future works.

VGG network [156] is an important step in development of deep convolutional architectures. First of all, it outperforms AlexNet and sets a new state of the art on ImageNet [145], the main large scale image recognition benchmark. More importantly, this work proposes a systematic and more intuitive approach to network design, that results in a simpler and more accurate architecture (see Figure 2.3). The paper manifests that CNNs have to be deep (up to 19 layers) in order better capture hierarchical structure of visual information. All convolutions have kernel of size 3x3; this makes the computations efficient. Stacking these layers together with nonlinearities and max-pooling increases the neuron's Field of Veiw (FoV) - the area on the input image that affects the current neuron's output. Thus, it can approximate more complex operations with larger kernel at lower cost. When layers resolution is halved due to pooling, the number of feature maps is usually doubled. This reduces the total number neurons deeper in the network and forces information compression. The main computational bottleneck is however not the convolutional backbone but the last fully-connected layers, serving as a powerful feature classifier. Overall, the VGG16 served as a main feature extractor in many downstream vision applications for several years, even though it has 140M parameters and could be rather slow in training and inference.

After the success of VGG, the positive influence of networks' depth on the performance became apparent. It seemed like building even deeper networks would solve all the problems, if only one had enough computational resources. However, that was only half true as in some cases stacking more layers could hurt the performance instead [76]. Such behavior was advocated to imperfect initialization strategies and difficulty in optimization, i.e., saturated gradients. ResNet [76] addresses the depth issue by introducing a new computational block that learns residual functions with reference to the layer inputs, instead of learning unreferenced functions. Figure 2.3 gives a visual explanation to this key computational strategy used in the ResNet architecture. This seemingly simple trick allows to avoid the problem of saturated gradients, i.e., there is a path from the loss to the input, where the gradient undergoes little transformations. This in turn makes it possible to build very deep networks (up to 1000 layers) that keep improving accuracy with

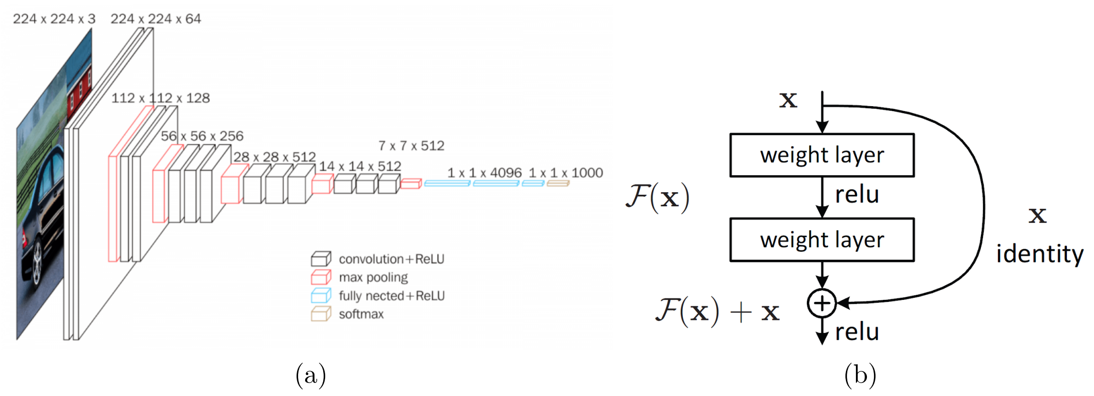


Figure 2.3 – (a) VGG network architecture [156]. Image source [156]. (b) Residual computation block of the ResNet architecture [76]. The block takes an input and computes a residual that is added to the input and passed down the network stream. This is in contrast to standard computational scheme where only residual is computed and propagated. Image source [76].

growing depth. Moreover, the residual block allows to easily learn the identity transformation by a layer (by setting up residual transformation to zero), if further increasing the network's depth is harmful for the performance. Importantly, ResNet demonstrates much better computational efficiency as it is a fully-convolutional architecture; it replaces the fully-connected layers by average-pooling and a linear map. Overall, ResNet [76] is a simpler, faster and more powerful feature extractor, comparing to VGG [156]. We use ResNet through the manuscript and build upon this model in Sections 3, 4, 5.

In most applications, convolutional neural networks surpass classical vision approaches by a big margin. Yet, the large number of parameters inevitably requires massive annotated datasets and a lot of computations. For example, the model introduced in [92] has 62 million parameters and was trained for 6 days on a the ImageNet dataset [145] (over a million annotated images). The need for large labeled datasets is impractical as human annotation becomes a bottleneck. The authors of [92] hypothesized that the CNN feature extractor trained on ImageNet must learn rather general visual representations, useful for other visual tasks. That is, AlexNet with fixed weights could provide image embeddings that capture high-level semantics, useful for image retrieval. This observation is of great importance for other computer vision tasks as it enables transferring the knowledge, learned by the network, from ImageNet to these tasks implicitly. Alternatively, initializing a CNN with ImageNet weights and training for other tasks has a similar effect. Following such initialization strategy greatly improves accuracy for object detection [61] and semantic segmentation [111] comparing to training the network from scratch. This

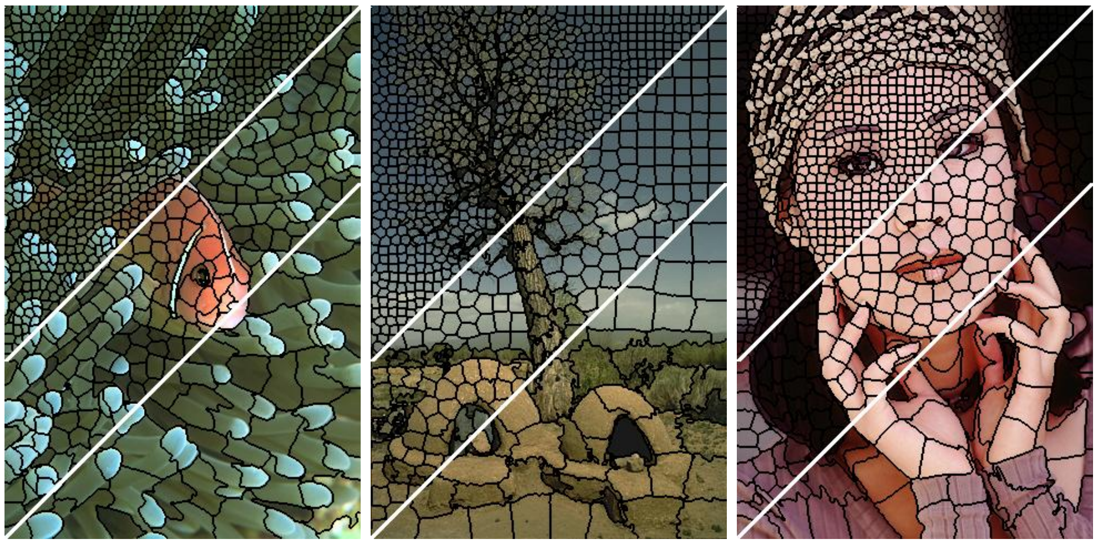


Figure 2.4 – Superpixels are groups of neighboring pixels organized together at several scales based on their color and gradient similarity. Image source [3].

serves as a first important example of judicious data usage in the era of deep learning.

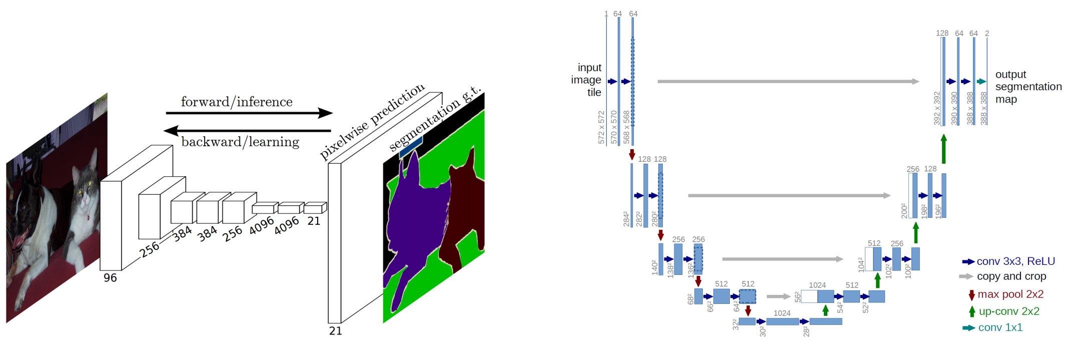
2.2 Semantic Segmentation

Image segmentation is a task of breaking an image into meaningful and coherent segments. There are many ways one can define "meaningful" and hence many types of segmentation problems. Semantic segmentation aims at producing coherent regions of pixels belonging to the same class of objects. In other words, given an image x, each pixel $x_{i,j}$ has to be assigned a class label $y_{i,j}$ from a predefined label set. Viewed this way, semantic segmentation is an example of dense classification problem and could potentially enjoy the advancements in standard image classification, discussed in Section 2.1. By labeling pixels at different spatial location, solving localization is implicitly assumed. As a result, the need to pursue two fairly different goals (classification and localization) simultaneously poses its own challenges. In this section, we discuss these challenges and the ways to address them.

A classical computer vision approach to semantic segmentation consists of two steps. First step is dedicated to finding plausible image segments that could be responsible for objects, while the second one is to classify these segments. It is possible to group pixels into connected regions, based on their color or gradient similarity. Those regions are called superpixels [143] (see Figure 2.4). Working with pixel groups instead of single pixels is advantageous as it brings region statistics into consideration. This allows to compute superpixel's semantic descriptors and classify them into categories. It could even happen, that region size is as small as one pixel [101], the descriptors would still be computed based on the local neighborhood. For example, the work of [101] represents each single pixel with a vector containing outputs of texture detectors. Such descriptors carry semantic information and could be used for classification directly.

When the pixels are classified independently, the relationship between output class variables is not taken into account, and thus, an important source of information is ignored. Probabilistic graphical models [89] allow to model such relationships by specifying a set of conditional independence assumptions. Conditional Random Fields (CRFs) [95] discriminatively model a latent set of variables Y, given an observed set of variables X. They enable reasoning about class labels y_i for all pixels, while taking into account the whole image X and using the prior knowledge induced by the independence assumptions. The work of [93] was the first to apply CRF for binary semantic segmentation. They used two types of pairwise potentials: the first one encouraging smoothness of predicted labels and the second one that adapts to discontinuities in neighboring brightness values. Later many other works applied CRF to refine their initial predictions defined on pixels or superpixels [91, 155, 170].

Performing dense pixel classification is actually a task that naturally suits convolutional neural networks. An early work in this area [48] demonstrates how to apply CNNs to segment image patches. Nowadays, all state-of-the art segmentation methods are based on the fully-convolutional approach [111] and are trained end-to-end. Doing so is more efficient because it eliminates the need to perform redundant computations on overlapping patches. More importantly, [111] introduces a number of ideas essential for obtaining good quality segmentations with CNNs. *Adapting an ImageNet pre-trained network* for dense predictions is one of the keys to good performance. The authors recast the last fully-connected layer of VGG as a convolutional one (with filter size 1×1) and slide it over the feature map. This results in a segmentation map of low resolution. The full pipeline is illustrated in Figure 2.5. The resulting architecture has a sub-sampling ratio 32, which means that the output mask is 32 smaller than the input. Thus the mask is very coarse and missing important local information. *Multi-scale architecture* is proposed to refine coarse predictions obtained from the last layer. Intermediate featuremaps of higher resolution are used for predicting their masks too; skip-connections are employed to make parallel predictions corresponding to different levels of features. As opposed to semantically reach but spatially coarse deeper layers, preceeding featuremaps of higher resolution provide the final masks with finer local details. Predictions from all the levels are then up-scaled to the same resolution and aggregated by summation. *Learnable upscaling* with deconvolutions



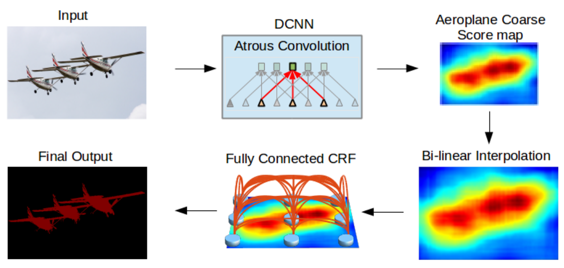
(a) (b)

Figure 2.5 – (a) The first fully convolutional framework of [111] for semantic segmentation built on AlexNet. Image source [111]. (b) Architecture of U-Net. Fusion of layers is achieved by adding up featuremaps. Image source [144].

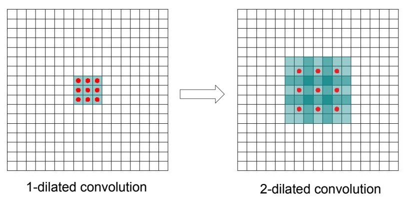
is proposed to process the coarse masks. Initialized as simple bi-linear up-sampling, deconvolutional layers learn a more optimal up-sampling strategy, guided by the final loss. *Shift-and-stitch* mechanism was proposed to enable truly dense prediction with 1-to-1 pixel correspondence between an input image and the predicted mask. The algorithm amounts to shifting an input image by one pixel S (where S is the final network's stride) times in each direction, computing slightly different masks at each location and interlacing them together to obtain the final mask.

The U-Net architecture [144] is an important step in CNN design for semantic segmentation. It proposes to reverse the scale pyramid of a base CNN feature extractor (contractive path) and up-scale image representations deeper in the network (expansive path) to obtain fine-grained predictions. Figure 2.5 demonstrates the network's design. The up-sampled layers in the expansive path are lacking precise local information, lost due to pooling operations. To restore it, the authors propose to transfer the information from the earlier layer on the contractive path. This design provides better localization and allows to incorporate global image information into final pixel predictions.

A fully-convolutional architecture is able to provide high quality labeling of individual pixels, however, as in earlier works on image segmentation, modeling relationships between label predictions explicitly is not implemented. DeepLab [25] bridges this gap by applying a fully-connected CRF on top of predicted masks in order to refine obtained predictions. This step is performed instead of transferring local details from early layers and shows competitive or better performance. The network's high sub-sampling ratio is addressed by dilated (atrous) convolutions [184]. This is an operation conceptually identical to "shift-and-stitch" mechanism but more computationally efficient. A convolutional filter is sparsified (dilated with zeros) and



(a)



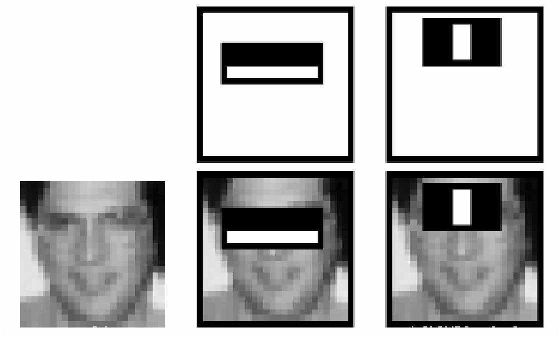
(b)

Figure 2.6 – (a) The overview of the DeepLab v1 pipeline. An input image is feed to a CNN to obtain a coarse activations map which is then up-scaled. The probability map is fed to a fully-connected CRF to produce the final output. Image source [25]. (b) Dilated convolution weights. If a dilation rate is $d > 1$, each original filter weight now has $d - 1$ nearest neighbors in each direction equal to zero (in green). This effectively performs a convolution of the original filter with each d^{th} input location, as illustrated on the right figure. Image source [184].

thus it is increased in size, as elaborated in Figure 2.6. Performing a convolution with a d-dilated filter effectively increases the input's spatial resolution and consequently the output neurons' field of view in d times. The work of [184] proposes to use atrou convolutions instead of pooling layers in the original ImageNet architecture. This prevents resolution degradation and avoids redundant computations. The main goal of increasing the field of view is to account for image global structure and to use visual context, which was shown to improve segmentation quality [25, 184]. This is further demonstrated in [26] by employing atrous spatial pyramid pooling (ASPP) to robustly segment objects at multiple scales. ASPP probes an incoming convolutional feature layer with filters at multiple sampling rates and effective fields-of-views, thus capturing objects as well as image context at multiple scales. Including this operation in the pipeline is enough to achieve state-of-the-art results without CRF post-processing. This suggests that if an output neuron has large enough FoV, the relations between predicted pixels is already learned by the network and does not need to be modeled explicitly.

2.3 Object Detection and Instance Segmentation

If semantic segmentation provides detailed semantic description of the scene, it does not distinguish between different object instances of the same class. The task of instance segmentation addresses this shortcoming; it assign each pixel to an object of predefined categories or labels it as background. This problem formulation is strictly harder than semantic segmentation. The task of object detection is a


(a)

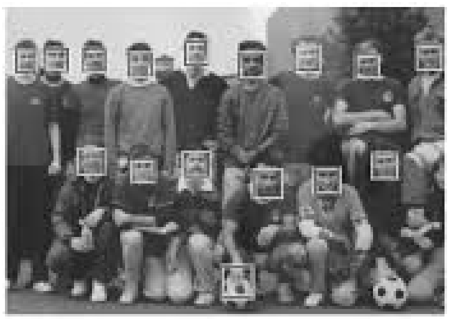

(b)

Figure 2.7 – (a) Examples of Harr filters applied to face detection. (b) : Output of Viola Jones detector. Images source [173].

proxy for instance segmentation. Its goal is to recognize and localize objects via bounding boxes. The most efficient instance segmentation methods to date rely on object detection to find boxes containing instances, and then segment their interior. Hence, in this section we first concentrate on methods for object detection and then describe instance segmentation approaches.

The basic principles for solving object detection changed very little in the last two decades. The classical works [126, 173] on face detection would serve as a simple example to demonstrate these principles in action. A typical object detection pipeline as well consists of two parts: first defining region descriptors suitable for the problem, then building a machine learning model on top. The works [126, 173] proposed to use Harr-like filters for face detection. Given an image region, such filters compute difference between average intensities in adjacent region rectangles. These differences are later used to categorize the region. For example, it is well known that for human face, average intensity around an eye is considerably lower than in the region of cheeks (see Figure 2.7). Therefore, a common Haar feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheeks. A database of positive examples is obtained by extracting faces within annotated bounding boxes. The negative examples correspond to random image patches that have no faces on them. This database is then used to find appropriate Harr filters and to train a classifier on their responses. After this is done, a set of selected filters with a trained classifier constitutes a face detector. To actually detect faces on arbitrary images, this detector is applied to numerous image locations at different scales, in a sliding window fashion. The regions classified positively correspond to detected faces, as illustrated in Figure 2.7. Depending on the difficulty of a detection task, one may use more sophisticated feature descriptors [22, 35, 112, 182] or build more powerful classifiers [31, 97, 151].

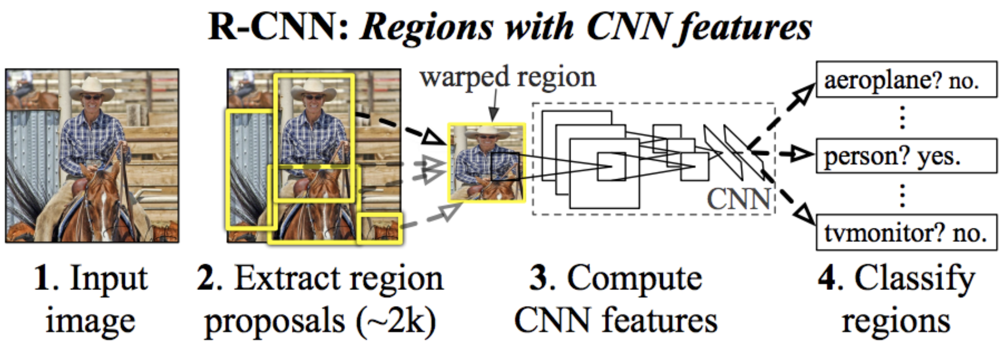


Figure 2.8 – Oevrview of R-CNN system. (1) It takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large (CNN), and then (4) classifies each region using class-specific linear SVMs. Image source [62].

Nevertheless, the core ideas of mining training samples, learning a detector and applying it across various image patches are highly relevant to date.

Detection benchmark datasets become more complicated with time and force the methods to evolve as well. For example, Pascal VOC'07 [47] dataset includes 20 categories of common objects that are present under scale and appearance variation. The work [50] addresses both inter- and intra-class diversity problems in their DPM method. They first compute Histogram of Gradient (HOG) [35] features on a dense grid of image locations and build a part-based model on top. The part based model assumes that objects of the same category are composed of the same parts, and models part's appearence independetly. Once the part detectors are learned, the object model assembles their outputs and robustly detect instances under appearance variations and occlusions. Mining hard negative examples for training further reduces the number of false positive detections. At test time, the model is slid over the whole image at multiple scales to assigns a score to each location, based on the learned object model. Highly-scored boxes correspond to successful detections. Since the model is applied on a dense grid, one object is often detected multiple times. In such situations, A standard practice for removing duplicate detections is to use Non Maximum Suppression (NMS) algorithm. This simple heuristic amounts to keeping only boxes with the highest score while other boxes, highly overlapping with the top ones, are discarded. The overlapping criteria is defined via Intersection over Union (IoU) between the bounding boxes.

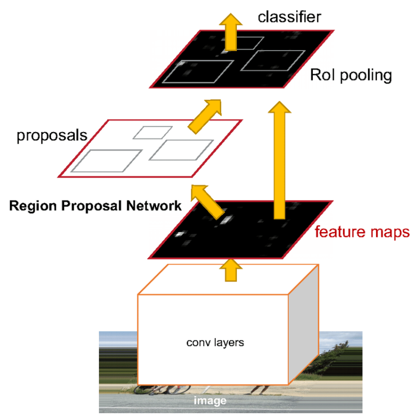
Two-stage CNN detectors

A seminal work of [62] is the first to tackle object detection with convolutional neural networks. In the general detection framework, CNN serves as a region feature extractor. An ImageNet pre-trained network is re-purposed to provide semantic descriptors for squared image patches. One forward pass of a deep CNN is computationally expensive, therefore it is too costly to apply the detector in a sliding window fashion. Instead, the authors use class-generic object proposals [169] that are more likely to contain a foreground object than background. This results in orders of magnitude fewer potential object locations to be evaluated. These regions are then cropped, re-sized to a square, and fed to the neural network. As the CNN is specifically trained for region classification, it leads to highly discriminative region features. The overview of the pipeline is given in Figure 2.8. Once the region features are computed by the network, they are classified using class-specific linear SVMs.

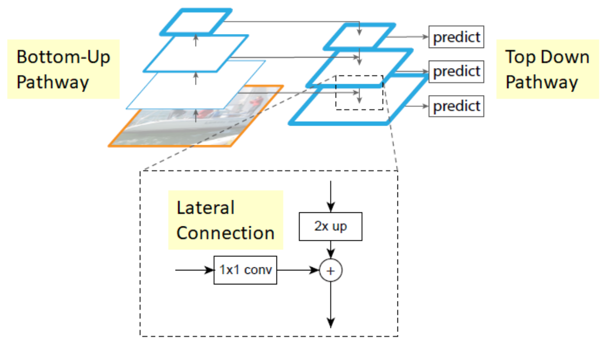
Training R-CNN amounts to fine-tuning AlexNet for region classification, which provides rich semantic features. The training procedure for the SVM classifier is analogous to the one of DPM [50]. Here, however, the hard negative mining also comes from the region proposals. In particular, at each iteration 25% of the region proposals are positive (IoU > 0.5 with a ground-truth bounding box) and the rest are negative (IoU < 0.3 with a ground-truth bounding box). At test time, 2000 proposals are sampled using the Selective Search algorithm [169], re-sized to 227x227, fed to AlexNet [92] and then classified by SVM. Finally, NMS is applied on the scored regions to remove duplicates.

The R-CNN approach establishes basic principles of object detection with CNNs and demonstrates that semantically rich features are the key to high-quality object detection. Overall, R-CNN makes a remarkable step forward in object detection. However, there is a number of drawbacks such as low computational efficiency of the method and separate optimization of feature extractor and classifier. These issues are addressed in the follow-up paper [61]. There, instead of cropping a region from an image and feeding it to a CNN, the whole image is first processed by a network and only then region features are cropped from a deep feature map. Doing so removes redundant computations on overlapping regions and provides object features with neighborhood visual context. Those region features are then re-sampled to a standard (7×7) resolution using RoI-pooling, i.e., an operation that performs max-pooling of non-uniform regions into a fixed sized feature map. Later, the region features are processed by a fully-connected network to predict object's category and bounding box offset. Since the initial object proposals are still computed using traditional methods [169], predicting corrections based on CNN's features substantially improves the localization performance.

Faster-RCNN [142] approach finally abandons standard vision techniques and



(a)



(b)

Figure 2.9 – (a) Overview of the Faster-RCNN system. An input image is first processed by a CNN and mapped to a feature map. From this map, object proposals are predicted. These locations are cropped from the feature map and used for classification and localization. Image source [142]. (b) : The Feature Pyramid Network architecture. Top-down pathway is attached to the base feature extractor (bottom-up pathway). Skip-connection are used to fuse information from earlier layers. Predictions are obtained from each layer in the feature pyramid. Image source [106].

shifts to a pipeline entirely based on deep learning; the system now computes object proposals directly from CNN features. As Fugure 2.9 demonstrates, after processing an image with a series of convolutions, obtained dense descriptors are directly used to predict class agnostic object proposals. This module is called Region Proposal Network (RPN) and is used instead of classical Selective Search [169]. The rest of the pipeline is identicall to Fast-RCNN [61], i.e., predicting category and box offsets from a region feature using a multi-layer perceptron. The experiments show that using RPN instead of shallow image proposals is faster; moreover, the proposals are more accurate and result in better detection performance. Finally, the whole pipeline can now be optimized end-to-end which provides image representations tailored to the final task at all levels. On the PASCAL VOC'07 [47] dataset, the method achieves 73.2% mean Average Precision (mAP) and runs at 5 frames per second (FPS).

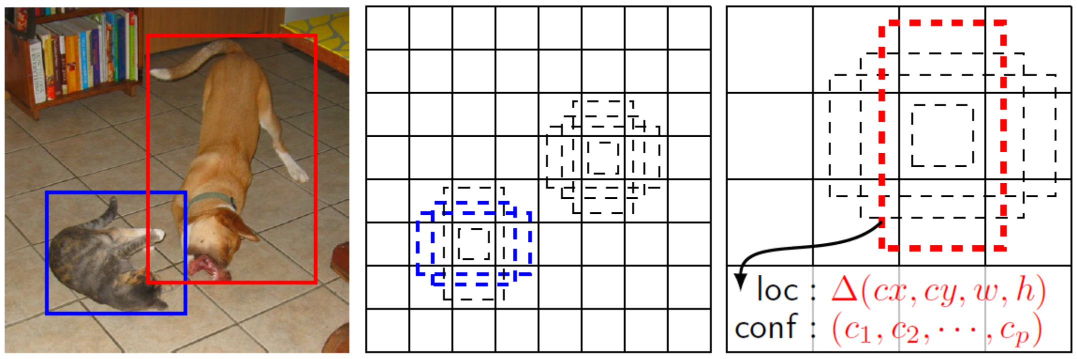
Faster-RCNN establishes an independent two-staged computational framework, called general R-CNN framework, tailored to object-centric tasks, i.e., the tasks concerned with retrieving and processing image regions containing objects of interest. In the general RCNN framework, the first stage computes dense deep image representations and proposes potential object locations, the second stage performs independent per-region computations that are specific to the final task. To advance the framework further, the community concentrated on delivering deep features of better quality. The inside-outside network [14] combines information from several top down layers to aggregate region descriptors across multiple scales. Feature pyramid networks [106] add a deconvolutional head on top of main feature extractor and use lateral connections between layers of the same scale (see Figure 2.9). This helps with detection of small objects and provides better context modeling. It is important to note, that the same tricks led to identical improvements for semantic segmentation, which implies tight connection between the tasks. However, semantic segmentation processes an image as a whole and is not concerned with region-based computations, which renders the general R-CNN framework irrelevant for semantic mask prediction.

One-stage CNN detectors

Another way to perform object detection with CNNs is to abandon region-based framework and share computations across all image locations, as traditionally done in sliding-window approaches. YOLO [139], MutiBox [46] and Single-Shot multibox Detector (SSD) [110] are the first methods to do so. In this section, we focus on SSD, as we build upon this framework in Sections 3 and 4.

SSD relies on a fully convolutional architecture for feature extraction and uses anchor boxes of different scales and aspect ratios, densely covering input image, to form a set of proposals. We call this proposal set default boxes. The authors associate a set of default bounding boxes with each feature map cell, for multiple feature maps at the top of the network. The default boxes tile the feature map in a convolutional manner, so that the position of each box relative to its corresponding cell is fixed. This eliminates the need for any object proposal generator, and thus enables fast computations. Figure 2.10 gives an overview of the pipeline. The network's architecture is composed of a base feature-extractor, that is an ImageNet pre-trained network VGG16, followed by an auxiliary structure composed of convolutional blocks. These blocks decrease in size progressively and allow predictions of detections at multiple scales. Each layer dedicated for prediction is responsible for a specific object scale and produces a fixed set of detections using convolutional filters. The system design is presented in Figure 2.10, bottom. Similar to other frameworks, the model outputs category probabilities and class-specific box offsets, that are measured relative to a default box position.

Prior to training, it has to be determined which default boxes correspond to a ground truth detection. For each ground truth box, all default boxes across different locations, scales, and aspect ratios are considered and matched to the ground truth box only if their IoU is higher than 0.5. Once the target outputs are


loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$

(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

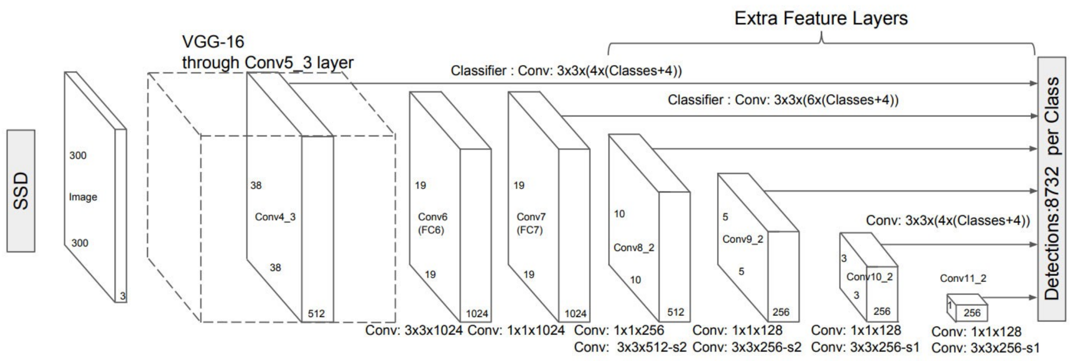


Figure 2.10 – Design principles of Single Shot multibox Detector (SSD). *Top*: Initial image (a) is tiled with candidate default boxes of different aspect ratios at different scales ((b) and (c)). For each default box, object category and coordinate offset are predicted (c). *Bottom*: CNN architecture used for object detection. Here, VGG16 base network is augmented with top-down pathway containing feature maps of decreasing scales. Each feature map is responsible for detecting objects of corresponding scale only. Images source [110].

defined, the image is fed to the network to compute the loss accordingly:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{2.3}$$

The total loss is a weighted combination of classification L_{conf} and localization L_{loc} loss terms. The number of matched boxes N is used to normalize the loss. Let x_{ij}^p be an indicator for matching the i-th default box to the j-th ground truth box of category p. The localization loss is the Smooth L1 loss [61] between the predicted box (l) and the ground truth box (g) parameters. The authors regress to offsets for the center (cx, cy) of the default box (d) and for its width w and height h.

$$
\begin{gathered}
L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} \mathrm{smooth}_{\mathrm{L1}}(l_i^m - \hat{g}_j^m) \\
\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \\
\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)
\end{gathered}
\tag{2.4}
$$

After the matching step, most of the default boxes are negative, i.e., correspond to background. This results in significant imbalance between positive and negative training examples. Naively using all negative examples for computing the loss would bias the training. For that reason, SSD uses hard negative mining; it selects a small fraction of negative samples with the highest loss and adds only those to the total loss computation, along with the positive samples. The authors have found data augmentation to be crucial for training one-stage detectors. They use horizontal flips, random crops, color augmentation and zoom-out procedure. The letter amounts to downsizing an image and placing it on a blank canvas of the original image size. Doing so effectively decreases the size of all objects in the image and creates more examples of small objects in the dataset. During inference, an image is propagated through the network and all default boxes are evaluated. To speed up NMS post-processing, only top 200 candidates are considered for each class, based on the assigned score. When tested, SSD achieves 76.8% mAP on the Pascal VOC'07 detection benchmark, when working with 512×512 images and operating at 19 Frames Per Second (FPS). While being conceptually simpler, it works faster and achieves better results than two-staged Faster-RCNN. Such encouraging results stimulated more thorough investigation of one-stage detectors.

As it is usual for scene understanding with CNNs, further works focused on improving the quality of features, especially on increasing their field of view and enreaching them with global information. This way, the work of [55] proposed Deconvolutional Single Shot Detector which has a better network architecture than that of SSD. Instead of using the layers from top-down stream for detection, the authors attached a deconvolutional head (a cascade of convolutional blocks of increasing resolution) and used it for region classification. Similar to [106, 144], it helps with detecting small objects and takes visual context into account. A more fundamental problem of severe class imbalance between positive and negative proposals in one-stage detection was addressed in [107]. The paper proposed to use all default boxes for loss computation, but to down-weight contribution to the final loss of those examples, that are confidently classified. As shown in the paper, most of the negative proposals are indeed confidently classified as background, so the main contribution to the loss comes from positive and hard negative examples. Overall, one-stage detectors [55, 107] are competitive in terms of accuracy with

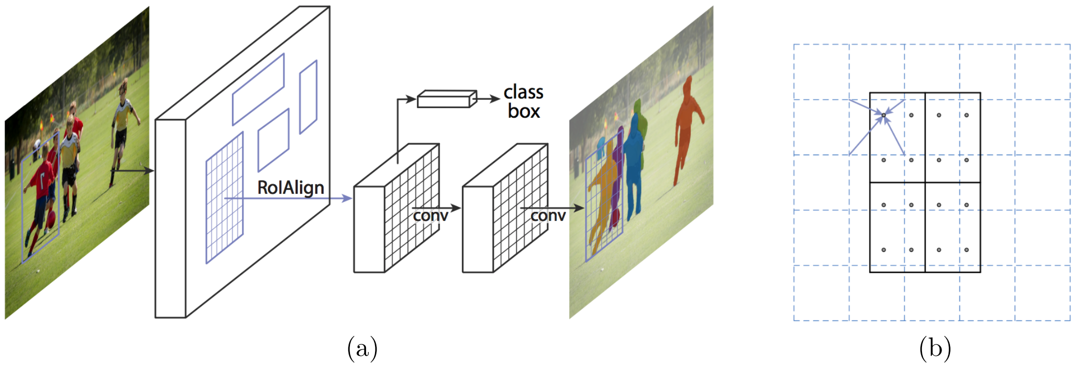


Figure 2.11 – (a) The Mask-RCNN pipeline for instance segmentation. (b) The design of RoIAlign module: the RoI (in bold) entries are composed by bi-linear interpolation of the nearest feature-map (in dashed) points.

the two-staged ones [106, 142], and faster at inference. However, this framework is interesting from a research perspective, as it incorporates less prior knowledge about the task. As other scene understanding methods [25, 28, 111], it uses a fully-convolutional architecture in its core and opens up perspectives for natural multi-task solutions.

Instance segmentation with CNNs

Instance segmentation is a task of classifying each pixel as belonging to a particular instance in a scene. This could be seen as an extension of semantic segmentation, where in addition to a category, each pixel has to be labeled with an object id. An alternative way of looking at instance segmentation is as an extention of object detection. Instead of coarse localization via a bounding box, the method has to find the underlying object boundary too, that is contained inside the bounding box. As a consequence, there are two paradigms of approaching instance segmentation, namely segmentation-based and detection-based. Segmentation based approaches [7,9,90] start with a semantic segmentation map and propose methods to split these segmentations into instances. Detection-based approaches [34,71,75] first perform object detection, then use obtained bounding boxes as instance locations and segment their interior. Recently, the field was dominated by detection-based methods as they are more accurate, thus we concentrate our attention on them. In the following subsection we briefly discuss modern approaches to instance segmentation built on top of object detectors.

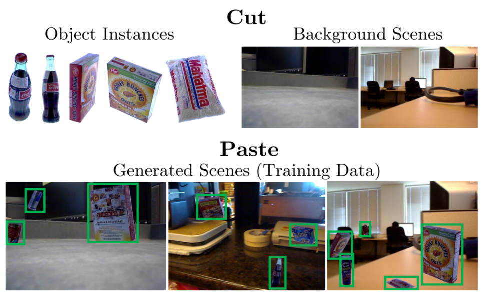
The work of [71] builds on R-CNN framework, but replaces object proposal boxes with region proposal segments, using a bottom-up class agnostic method. For each proposal, features are extracted by a CNN from both foreground region and its bounding box, and classified using SVM. As in classical object-detection pipelines, this step is followed by NMS to remove duplicates. Finally, remaining region proposals are further refined using coarse CNN outputs combined with super-pixels. Next, the follow-up [72] work augments the feature extractor by adding skip-connections and thus improves segmentation quality. Combined with a more advanced feature extractor and improved region refining procedure, the method significantly improves upon previous methods. The work of [34] is the first to propose an end-to-end trainable architecture for instance segmentation. It uses a cascade structure composed of 3 networks sharing convolutional weights. The first network outputs class-agnostic object proposals. The second one uses proposed locations to extract region representations and predict instance masks. The last network performs instance classification. Removing classical region proposal methods and training the whole system end-to-end enabled orders of magnitude faster inference with significantly better accuracy. Finally, Mask-RCNN [75], shown in Figure 2.11, is built on top of the Faster-RCNN [142] framework by adding an extra branch for predicting instance masks. In the Faster-RCNN detector, the regions are mapped to a fixed-sized representation using RoI-pooling, that harshly quantizes the representations spatially. Such effect is harmful when precise local information needs to be preserved for segmentation. Instead, in Mask-RCNN, the authors propose the RoI-align module (see Figure 2.11 (b)), that uses bi-linear interpolation and preserves local information. Doing so results in a significant boost in masks quality and a moderate boost in bounding box localization. The second important element of the pipeline is decoupling classification and segmentation. Segmentation is performed in a class-agnostic way, which removes competition between classes and does not degrade the accuracy. This pipeline is a simple and efficient state-of-the-art instance segmentation method that has high-quality implementation available online. These are the reasons for choosing Mask-RCNN for our experiments in Section 4.

Datasets for scene understanding

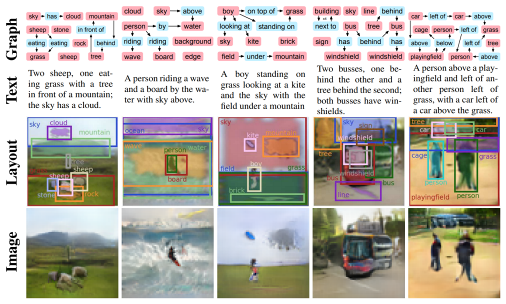
The story of modern general-purpose object detection starts with the PASCAL [47] dataset. The first version contained people and vehicles in realistic scenes and had only 684 images for training and validation. Next versions of the dataset included more common categories and by 2012 the dataset contained 20 classes and 11640 train-val images. The complexity and scale of this dataset were intimidating for existing computer vision methods. Although, for deep learning methods, the challenge seemed more approachable. One problem with PASCAL is that one could not train a deep CNN from scratch, just using this data. The problem however could be alleviated by using external data, i.e., via initializing a network with ImageNet pre-trained weights. Semantic and instance segmentation had fewer labeled images which is not surprising since pixelwise annotation is more difficult. The latest version of the PASCAL dataset included only 2913 images annotated for segmentation, which is 5 times fewer than for detection. Training good quality CNN models for segmentation was unfeasible. To fix this short-coming, the work [70] provides segmentation annotations for the rest of the Pascal VOC images. This enabled more extensive evaluations of CNN methods for segmentation however it was still not possible to train the models from scratch. The COCO [108] dataset was released as a next generation benchmark for scene understanding. It contains images of complicated scenes annotated for object detection and instance segmentation. In total, there is 80 general object categories and around 123K images for training and validation. All objects are annotated with bounding boxes and instance masks which is a tremendous step forward in terms of labeling effort. At this stage, it is possible to train good quality detection models purely on COCO [74]. The next generation of scene understanding datasets [16, 94] contains millions of images with box and mask annotations. Not only one can train better networks from scratch but also improve results on other tasks using pre-training on this data [102]. Surely, increasing the amount of labeled data will keep improving scene understanding performance, but not necessarily our understanding of the problem. An important question to ask is when do we need stop annotating new data and instead to investigate more judicious use of available but limited data?

2.4 Data Augmentation

To make better use of a given dataset, one can perform artificial data augmentation. This process amounts to synthetically expanding a dataset by applying transformations on the available examples. The transformations have to be applied in a controllable way, i.e., after an image is modified, the transformation of the ground-truth label has to be known too. The goal of this procedure is to reduce overfitting and improve generalization of the model. Intuitively, data augmentation is used to teach a model about invariances in the data domain: classifying an object is often insensitive to horizontal flips or translation [99]. Network architectures can also be used to hardcode invariances: convolutional networks bake in translation invariance. However, using data augmentation to incorporate prior knowledge about the problem or domain can be easier than hardcoding them into the model architecture directly. Simple image transformations such as horizontal flipping, cropping, adding random noise or mild color and intensity transformations can already bring a considerable improvement in many vision problems. It is important that augmented samples are faithful with respect to the training data distribution, otherwise test accuracy may suffer. For example in street sign recognition, flipping horizontally “turn left” or “turn right” and leaving the ground-truth label un-



(a)



Two sheep, one eating grass with a tree in front of a mountain; the sky has a cloud.

A person riding a wave and a board by the water with sky above.

A boy standing on grass looking at a kite and the sky with the field under a mountain

Two busses, one behind the other and a tree behind the second; both busses have winshields.

A person above a playingfield and left of another person left of grass, with a car left of a car above the grass.

(b)

Figure 2.12 – Examples of generated images with bounding box annotations. (a) The method extracts objects using segmentation masks and pastes them on random backgrounds at random locations. Image source [45]. (b) : The method uses a GAN to generate new scenes, conditioned on scene graphs. Location of bounding boxes is known in advance. Image source [82]. Modifying natural images, as in (a), produces more photo-realistic scenes, comparing to fully synthetic ones in (b).

changed, will in fact corrupt the labels. On the other hand, changing images' color intensity will just make the system robust to different illumination conditions. Thus, even the simplest augmentation strategy for image classification has to account for the prior knowledge about the data domain and the task.

More ambitious data augmentation strategies rely on human expertise more and may vary from trivial geometrical transformations to synthesizing new training images [53, 130]. Some recent object detectors [44, 110, 139] benefit from standard data augmentation techniques more than others [61, 142]. The performance of Fast- and Faster-RCNN could be for instance boosted by simply corrupting random parts of an image in order to mimic occlusions [188]. The field of semantic segmentation is enjoying a different trend—augmenting a dataset with synthetic images. They could be generated using extra annotations [147], that come from a purely synthetic dataset with dense annotations [68,116] or a simulated environment [136]. For object detection, recent works such as [83, 118, 161] also build and train their models on purely synthetic rendered 2d and 3d scenes. However, a major difficulty for models trained on synthetic images is to guarantee that they will generalize well to real data since the synthesis process introduces significant changes of image statistics [130]. This problem could be alleviated by using transfer-learning techniques such as [150] or by improving photo-realism of synthetic data [13, 158]. To address the same issue, the authors of [66] adopt a different direction by pasting desired targets into natural images, which reduces the presence of rendering artefacts. For object instance detection, the work [58] estimates scene geometry and spatial layout, before synthetically placing objects in the image to create realistic training examples.

In [45], the authors propose an even simpler solution to the same problem by pasting images in random positions but modeling well occluded and truncated objects, and making the training step robust to boundary artifacts at pasted locations. An illustration of this augmentation procedure is given in Figure 2.12. It is although not obvious if placing objects at random would work for traditional scene understanding tasks such as object detection, semantic and instance segmentation, as visual context around an object is sometimes as important for detection as the object itself [29]. In Chapter 4 we investigate this issue in a principal way.

Not all data augmentation techniques are handcrafted. Recently, learning-based data augmentation methods gained popularity. For example, the work of [100] learns how to combine two or more images in order to generate a new training sample. Another possibility is to use a Bayesian approach to generate data based on the distribution learned from the training set [168]. An alternative direction is to use a generative model, such as GAN [64], to create additional samples for training [6]. By conditioning a generator on object category, one obtains an image with a known ground-truth label. In [82], the authors go even further and generate images conditioned on scene graphs. Using more abundant information for conditioining provides object locations as ground-truth labels, and potentially allows to augment for object detection. However, the quality of generated images is still quite low and is not yet suitable for training CNN models [154] (see Figure 2.12). Instead of generating images directly, it is possible to generate a series of image transformations, chosen from a fixed set, that are optimal for data augmentation. Using reinforcement learning, the works of [33] implements this approach for image classification and the follow-up work [190] extends it to object detection. These approaches require a lot of computational resources and are often impractical when working on small scale.

2.5 Few-shot Learning

What works well for big data may be inapplicable when a dataset is small, especially if obtaining more data is not possible. That could happen when the cost of annotation is unaffordable or the data is naturally scarce. Hence, to tackle this problem one needs an algorithm that is able to learn from only a handful of samples per category, i.e., a scenario called few-shot learning. Neural networks are notoriously difficult to train when a large enough dataset is not available [92] and end up overfitting small datasets. However, one can use a richer source of related data to help learning a classifier from few samples. This problem setting is known as few-shot classification.

A typical few-shot classification problem consists of two stages, called meta-training and meta-testing [27]. During the meta-training stage, one is given a large-enough dataset annotated for classification, which is used to train a predictive model. The model is learned from scratch, meaning that it is randomly initialized, and no external data is used. During meta-testing, novel categories are provided with few annotated examples (typically 1 or 5), and we evaluate the capacity of the predictive model to retrain or adapt, and then generalize on these new classes. Unfortunately, simply fine-tuning a convolutional neural network on a new classification task with few samples leads to poor results [52], which has motivated the community to develop new approaches, dedicated to few-shot classification.

Recently, the field of few-shot learning has been dominated by approaches based on meta-learning [152], which is formulated as a principle to learn how to adapt to new learning problems. These approaches typically sample few-shot classification tasks from the meta-training dataset and solve them during training. A few-shot classification task is composed to be agnostic to meta-training categories; a smaller subset of K classes is sampled at random and the class labels are re-assigned to be $\{1, 2, .., K\}$. First, by removing class-specific information the method reduces overfitting to the meta-training categories, as advocated in [172]. Second, since each few-shot classification problem may be considered as an independent task, the method is optimized to perform well across various tasks at the same time. Hence, according to the meta-learning literature [52, 137, 152], the model should generalize to new tasks better.

For instance, in [52], a "good network initialization" is learned such that a small number of gradient update steps on a new problem is sufficient to obtain a good CNN classifier. This is done by defining the network's initial weights as a variable, performing a gradient update on the weights, and optimizing the loss with respect to the initialization. In [137], the authors learn both the network initialization and an update rule (optimization model) represented by a Long-Term-Short-Memory network (LSTM). Instead of using explicit parameter updates at test time, [18] trains a feed-forward network that predicts weights for the feature extractor, based on one image of a new concept. The paper [17] shows that on top of the common feature extractor one may simply use a classifier with a closed-form solution for each few-shot task. This allows to optimize the feature extractor directly, without ever needing to update or predict model's parameters at test time. This methodology is conceptually simpler and results in a speed-up over iterative optimization.

Another approach to few-shot learning problems is based on metric learning. The goal is to learn an embedding to a feature space with only one desirable property — images of the same class have to be close to each other and far from other classes. As in meta-learning pipelines, the information about base categories is discarded during training. This turns out to be sufficient for separating new classes as well [87], resulting in a simple few-shot classifier. Inspired by few-shot learning strategies developed before deep learning approaches became popular [117], distance-based classifiers (based on the distance to a centroid) were also proposed, e.g., prototypical networks [159], or more sophisticated classifiers with attention [172]. All these methods consider a classical backbone network, and learn the weights from scratch using meta-learning training procedure, i.e., by sampling few-shot problems from the meta-training set. This class of methods is easier to train and proposes a clear motivation for using non-parametric distance-based classifiers.

Recently, using meta-learning to train a feature extractor was found to be sub-optimal [59,125,135]. Specifically, better results were obtained by training a CNN on a standard classification task using the meta-training data in the first step, and then only fine-tuning with meta-learning in the second step [125,146,181]. Others, such as [59, 135], simply freeze the network obtained in the first step, and train a single prediction layer with meta-learning, which results in similar performance. Finally, the paper [27] demonstrates that straight-forward baselines without meta-learning (using distance-based classifiers) work equally well. This solution is the simplest conceptually and thus is more appealing from a research point of view. In this manuscript we push these principles even further and shows in Chapter 5 that by appropriate variance-reduction techniques, even the simplest approaches can significantly outperform current state-of-the-art methods, built with meta-learning, or with metric learning.

In fact, there is little difference from a research perspective, if the problem is big or small. Each would have its own specific challenges. They all however have something in common. The solution must originate from the data and be inspired by the constraints. Armed with this simple rule, in this manuscript, we discover how to use limited data in a more effective way by carefully looking at the problem and its input, and transform our observations into a working method.

Chapter 3

Real-time Scene Understanding with BlitzNet

Chapter abstract: Real-time scene understanding has become crucial in many applications such as autonomous driving or robotics, however collecting annotations for these problems is time-consuming. In this chapter, we propose a deep architecture, called BlitzNet, that jointly performs object detection and semantic segmentation in one forward pass, allowing real-time computations. Besides the computational gain of having a single network to perform several tasks, we show that object detection and semantic segmentation benefit from each other in terms of accuracy. Experimental results for VOC and COCO datasets show state-of-the-art performance for object detection and segmentation among real time systems.

The work presented in this chapter was achieved with the collaboration of Konstantin Shmelkov, with equal contribution between Konstantin and myself. The main topic of this chapter is object detection and semantic segmentation, see Sections 2.2 and 2.3 for background and related work. The source code implementing the algorithm is available online [121] and can be found in Appendix B. Additional qualitative results are presented in Appendix C. The material of this chapter is based on the following publication:

Nikita Dvornik*, Konstantin Shmelkov*, Julien Mairal, Cordelia Schmid. BlitzNet: A Real-Time Deep Network for Scene Understanding. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017.

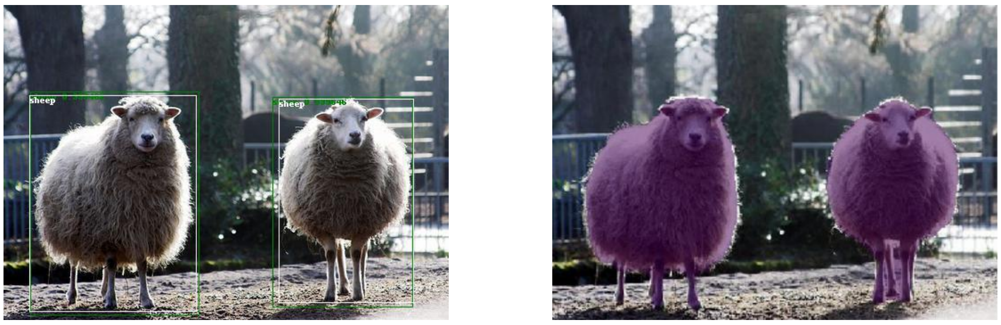


Figure 3.1 – The outputs of our pipeline. On the left, the bounding boxes were generated as a solution for object detection. The right image shows generated class maps for semantic segmentation.

Object detection and semantic segmentation are two fundamental problems in object-level scene understanding. The task of object detection is to identify on an image all objects of predefined categories and localize them via bounding boxes. Semantic segmentation operates at a finer scale; its aim is to parse an image and associate a class label to each pixel. Despite the similarities of the two tasks, only few works have tackled them jointly [51], [88], [163], [180].

Yet, there is a strong motivation to address both problems simultaneously. On the one hand, good segmentation is sufficient to perform detection in some cases. As Figure 3.1 suggests, an object may be sometimes identified and localized from segmentation only by simply looking at connected components of pixels sharing the same label. In the more general case, it is easy to conduct a simple experiment showing that ground-truth segmentation is a meaningful clue for detection, using for instance ground-truth segmentation as the input of an object detection pipeline. On the other hand, correctly identified detections are also useful for segmentation as shown by the success of weakly supervised segmentation techniques that learn from bounding box annotation only [127]. Our goal is to solve efficiently both problems at the same time, by exploiting image data annotated at the global object level (via bounding boxes), at the pixel level (via partially or fully annoated segmentation maps), or at both levels.

As most recent image recognition pipelines, our approach is based on convolutional neural networks [98], which are widely adopted for object detection [62] and semantic segmentation [111]. A popular line of work in object detection [61,62,142] advocates for a two stage pipeline. During the first stage, object location proposals are generated. Each of the proposed regions is classified independently during the second stage. Despite being highly accurate, the paradigm remains expensive and relies on a region-based strategy (see also [103]) that makes the network architecture inappropriate for semantic segmentation.

To avoid the constraints imposed by a two-stage pipeline, we choose instead to base our work on the Single Shot Detection (SSD) [110] approach, which consists of a fully-convolutional model to perform object detection in one forward pass. Besides the fact that it allows all computations to be performed in real time, the pipeline is more generic, offers more degrees of freedom in network architecture design and opens new perspectives to solve our multi-task problem.

Interestingly, recent work on semantic segmentations are also moving in the same direction, see for instance [111]. Specific to semantic segmentation, [111] also introduces new ideas such as the joint use of feature maps of different resolutions, in order to obtain more accurate classification. The idea was then improved by adding deconvolutional layers at all scales to better aggregate context, in addition to using skip and residual connections [144]. Deconvolutional layers turned out to be useful to estimate precise segmentations, and are thus good candidates to design architectures where localization is important.

In this chapter, we consider a multi-task scene understanding problem consisting of joint object detection and semantic segmentation. For that purpose, we propose a novel pipeline called BlitzNet, which is made available as an open-source software package. BlitzNet is able to provide accurate segmentation and object bounding boxes in real time. With a single network for solving both problems, we not only reduce the computational cost but also enable the tasks to share learned knowledge and benefit from each other in terms of accuracy. Moreover, shared representations enable the tasks to learn from each others annotations and leads to an increase in semantic segmentation accuracy when more images are annotated with bounding boxes only. This is of great importance given how much more time and effort pixel-wise annotation requires comparing to bounding boxes.

Outline. The chapter is organized as follows: Section 3.1 presents our real-time multi-task pipeline called BlitzNet and elaborates on architecture choices important for a unified CNN architecture. Section 3.2 is devoted to our experiments with main detection and segmentation datasets, and finally Section 3.3 concludes the chapter.

3.1 Multi-task System Design

In this section, we introduce the BlitzNet architecture and discuss its different building blocks.

3.1.1 Global View of the Pipeline

The joint object detection and segmentation pipeline is presented in Figure 3.2. The input image is first processed by a convolutional neural network to produce

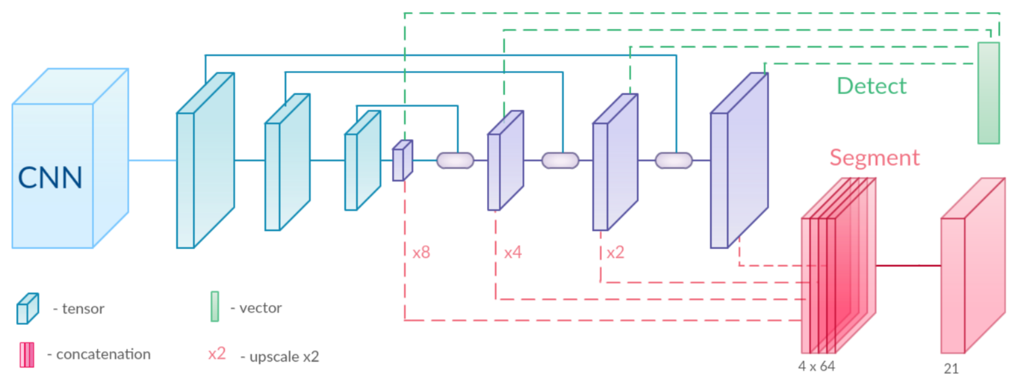


Figure 3.2 – The BlitzNet architecture, which performs object detection and segmentation with one fully convolutional network. On the left, CNN denotes a feature extractor, here ResNet-50 [76]; it is followed by the downscale-stream (in blue) and the last part of the net is the upscale-stream (in purple), which consists of a sequence of deconvolution layers interleaved with ResSkip blocks (see Figure 3.3). The localization and classification of bounding boxes (top) and pixelwise segmentation (bottom) are performed in a multiscale fashion by single convolutional layers operating on the output of deconvolution layers.

a map that carries high-level features. Because of its high performance for classification and good trade-off for speed, we use the network ResNet-50 [76] as our feature encoder.

Then, the resolution of the feature map is iteratively reduced to perform a multi-scale search of bounding boxes, following the SSD approach [110]. Inspired by the hourglass architecture [120] for pose estimation and an earlier work on semantic segmentation [124], the feature maps are then up-scaled via deconvolutional layers in order to predict subsequently precise segmentation maps. Recent DSSD approach [55] uses a similar strategy for object detection the top part of our architecture presented in Figure 3.2 may be seen as a variant of DSSD with a simpler "deconvolution module", called ResSkip, that involves residual and skip connections.

Finally, prediction is achieved by single convolutional layers, one for detection, and one for segmentation, in one forward pass, which is the main originality of our work.

3.1.2 SSD and Downscale Stream

The Single Shot MultiBox Detector [110] tiles an input image with a regular grid of anchor boxes and then uses a convolutional neural network to classify these boxes and predict corrections to their initial coordinates. In the original paper [110], the base network VGG-16 [156] is followed by a cascade of convolutional and pooling layers to form a sequence of feature maps with progressively decreasing spatial resolution and increasing field of view. In [110], each of these layers is processed separately in order to classify and predict coordinates correction for a set of default bounding boxes of a particular scale. At test time, the set of predicted bounding boxes is filtered by non-maximum suppression (NMS) to form the final output.

Our pipeline uses such a cascade (see Figure 3.2), but the classification of bounding boxes and pixels to build the segmentation maps is performed in subsequent layers, called deconvolutional layers, which will be described next.

3.1.3 Deconvolution Layers and ResSkip Blocks

Modeling visual context is often a key to complicated scenes parsing, which is typically achieved by pooling layers in a convolutional neural network, leading to large receptive fields for each output neuron. Sometimes features of an object could be less representative then its surrounding. Even though theoretical receptive field of very deep networks covers the whole image the work of [113] shows that the effective receptive field is much smaller and important semantic information about a region could be missing. For semantic segmentation, precise localization is equally important, and [124] proposes to use deconvolutional operations to solve that issue. Later, this process was improved in [120] by adding skip connections. Apart from combining high- and low-level features it also eases the learning process [76].

Like [55] for object detection and [120] for pose estimation, we also use such a mechanism with skip connections that combines feature maps from the downscale and upscale streams (see Figure 3.2). More precisely, maps from the downscale and upscale streams are combined with a simple strategy, which we call ResSkip, presented in Figure 3.3.

First, incoming feature maps are upsampled to the size of corresponding skip connection via bilinear interpolation. Then both skip connection feature maps and upsampled maps are concatenated and passed through a block (1×1 convolution, 3×3 convolution, 1×1 convolution) and summed with the upsampled input through a residual connection. The goal of such construction is to learn how to enrich spatially coarse but semantically rich features obtained by upscaling the main stream with fine localization information delivered from early layers by skip-connections. The benefits of this topology will be justified and discussed in more details in the experimental section.

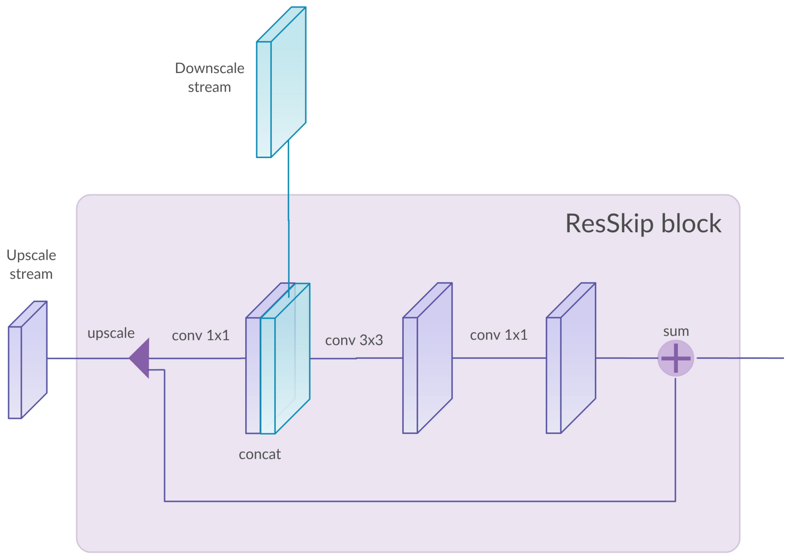


Figure 3.3 – ResSkip block integrating feature maps from the upscale and downscale streams, with skip connection.

3.1.4 Multiscale Detection and Segmentation

The problem of semantic segmentation and object detection share several key properties. They both require per-region classification, based on the pixels inside an object while taking into account its surrounding, and benefit from rich features that include localization information. Instead of training a separate network to perform these two tasks, we train a single one that allows weight sharing, such that both tasks can benefit from each other.

In our pipeline, most of the weights are shared between the tasks. Computations specific to object detection are performed by a single convolutional layer that predicts a class label and coordinate offset for each bounding box in the feature maps of the upscale stream. Similarly, a single convolutional layer is used to predict the pixel labels and produce segmentation maps. To achieve this we resize all the activations of the upscale stream to the resolutions of the last layer, concatenate them and feed to the final classification layer.

3.1.5 Speeding up Non-Maximum Suppression

Increasing the number of anchor boxes heavily affects inference time because it performs NMS on a potentially huge number of proposals (in the worst case scenario, it may be all of them). Indeed, we observed that by using sliding window proposals, addition of small scale proposals slows down the inference even more than increasing image resolution. Surprisingly, non-maximum suppression may then become the bottleneck at inference time. We observed that this occurred sometimes for particular object classes that return a lot of bounding box candidates.

Therefore, we suggest a different post-processing strategy to accelerate detection when there are too many proposals. For each class, we pre-select the top 400 boxes with largest scores, and perform NMS leaving only 50 of them. Overall, the final detection is the top 200 highest scoring boxes per image after non-maximum suppression. This strategy yields a reasonable computational time for NMS, and has marginal impact on accuracy.

3.1.6 Training and Loss Functions

Given labeled training data where each data point is annotated with segmentation maps, or bounding boxes, or with both, we consider a loss function which is simply the sum of two loss functions of the two task. Note that we tried reweighting the two loss functions, but we did not observe noticeable improvements in terms of accuracy.

For segmentation, the loss is the cross-entropy between predicted and target class distribution of pixels [24]. Specifically, we use a 1×1 convolutional operation with 64 channels to map each layer of the upscale-stream to an intermediate representation. After this, each layer is upscaled to the size of the last layer using bilinear interpolation and all maps are concatenated, which results in feature map with $64 \times k$ channels, k being the total number of layers in the up-scale stream. This final representation is mapped to c feature maps, where c is the number of classes, by using 3×3 convolutions to predict posterior class probabilities.

For detection, we use the same loss function as [110] when performing tiling of the input image with anchor boxes and matching them to ground truth bounding boxes. We use activations of each layer in the upscale-stream to regress corrections for coordinates of the anchor boxes and to predict the class probability distribution. We use the same data augmentation suggested in the original SSD pipeline, namely photometric distortions, random crops, horizontal flips and zoom-out operation.

3.2 Experimental Results

We now present various experiments conducted on the COCO, Pascal VOC 2007 and 2012 datasets, for which both bounding box annotations and segmentation maps are available. Section 3.2.1 discusses in more details the datasets and the metrics we used; Section 3.2.2 presents technical details that are useful to make our work reproducible, and then each subsequent subsection is devoted to a particular experiment. The last two sections discuss the inference speed and clarify particular choices in the network architecture. Our code is now available as an open-source software package at `http://thoth.inrialpes.fr/research/blitznet/`.

3.2.1 Datasets and Metrics

We use the COCO [108], VOC07, and VOC12 datasets [47]. All images in the VOC datasets are annotated with ground truth bounding boxes of objects and only a subset of VOC12 is annotated with target segmentation masks. The VOC07 dataset is divided into 2 subsets, trainval (5011 images) and test (4952 images). The VOC12-train subset contains 5717 images annotated for detection and 1464 of them have segmentation ground truth as well (VOC12-train-seg), while VOC12-val has 5823 images for detection and 1449 images for segmentation (we call this subset VOC12-val-seg). Both datasets have 20 object classes.

The COCO dataset includes 80 object categories for detection and instance segmentation. For the task of detection, there are 80k images for training and 40k for validation. There is no either a protocol for evaluation of semantic segmentation or even annotations to train it from. In this work, we are interested particularly in semantic segmentation masks so we obtain them from instance segmentation annotations by combining instances of one category.

To carry out more extensive experiments we leverage extra annotations for VOC12 segmentation provided by [70], which gives a total of 10,582 fully annotated images for training that we call VOC12-train-seg-aug. We still keep the original PASCAL annotations in VOC12 val-seg, even if a more precise annotation is available in [70], for a fair comparison with other methods that do not benefit from these extra annotations.

In VOC12 and VOC07 datasets, a predicted bounding box is correct if its intersection over union with the ground truth bounding box is higher than 0.5. The metric for evaluation detection performance is the mean average precision (mAP) and the quality of predicted segmentation masks is measured with mean intersection over union (mIoU).

3.2.2 Experimental Setup

In this section, we discuss the common setup to all experiments. BlitzNet is coded in Python and TensorFlow. All experiments were conducted on a single Titan X GPU (Maxwell architecture), which makes the speed comparison with previous work easy, as long as they use the same GPU.

Optimization Setup. In all our experiments, unless explicitly stated otherwise, we use the Adam algorithm [85], with a mini-batch size of 32 images. The initial

network	backbone	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
SSD300* [110]	VGG-16	77.6	79.2	84.0	75.6	69.9	50.9	86.7	85.9	88.6	60.1	81.4	76.8	86.2	87.3	84.2	79.5	52.7	79.3	79.4	87.7	77.2
SSD300* (our reimpl)	ResNet-50	75.3	75.3	85.1	72.5	67.4	45.5	85.7	83.9	82.8	57.2	79.1	76.7	83.1	86.5	83.3	77.5	50.1	74.4	79.4	86.5	73.3
BlitzNet300 (s8)	ResNet-50	78.5	79.7	85.9	80.1	72.1	50.9	87.0	84.6	88.2	62.3	83.7	77.1	87.3	85.0	84.7	79.2	54.9	81.5	80.0	87.0	78.0
BlitzNet300 (s4)	ResNet-50	78.2	86.8	85.1	78.3	70.4	47.5	85.4	85.0	86.2	59.0	81.8	77.9	86.9	86.1	85.4	78.6	54.9	81.9	81.1	87.7	78.2
BlitzNet300 + seg (s4)	ResNet50	**79.1**	86.7	86.2	78.9	73.1	47.6	85.7	86.1	87.7	59.3	85.1	78.4	86.3	87.9	84.2	79.1	58.5	82.5	81.7	85.7	81.8
SSD512* [110]	VGG-16	79.6	84.9	85.8	80.7	73.0	58.0	87.8	88.4	87.6	63.6	85.4	73.1	86.3	87.7	83.7	82.6	55.3	81.5	79.1	86.4	80.3
BlitzNet512 (s8)	ResNet-50	80.7	87.7	85.4	83.6	73.3	58.5	86.6	87.9	88.5	63.7	87.3	77.6	87.3	88.1	86.2	81.3	57.1	84.9	79.8	87.9	81.5
R-FCN [103]	ResNet-101	80.5	79.9	87.2	81.5	72.0	69.8	86.8	88.5	89.8	67.0	88.1	74.5	89.8	90.6	79.9	81.2	53.7	81.8	81.5	85.9	79.9
Faster RCNN	ResNet-101	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	89.8	56.7	87.8	69.4	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72.0
YOLO [139]	YOLO net	63.4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	- -
BlitzNet512 + seg (s8)	ResNet50	**81.5**	87.0	87.6	83.5	75.7	59.1	87.6	88.0	88.8	64.1	88.4	80.9	87.5	88.5	86.9	81.5	60.6	86.5	79.3	87.5	81.7

Table 3.1 – **Comparison of detection performance on Pascal VOC 2007 test set.** The models where trained on VOC07 trainval + VOC12 trainval. The models that have suffix "+ seg" where trained for segmentation jointly with data from VOC12 trainval and extra annotations provided by [70]. The values in columns correspond to average precision per class (%).

network	backbone	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
SSD300* [110]	VGG-16	75.8	88.1	82.9	74.4	61.9	47.6	82.7	78.8	91.5	58.1	80.0	64.1	89.4	85.7	85.5	82.6	50.2	79.8	73.6	86.6	72.1
BlitzNet300	ResNet50	75.4	87.4	82.1	74.5	61.6	45.9	81.5	78.3	91.4	58.2	80.3	64.9	89.1	83.5	85.7	81.5	50.5	79.9	74.7	84.8	71.1
BlitzNet300 + COCO	ResNet50	80.2	91.0	86.5	80.0	70.1	54.7	84.4	84.1	92.5	65.1	83.5	69.2	91.2	88.1	88.5	85.7	55.8	85.4	79.3	89.8	78.2
R-FCN [103]	ResNet-101	77.6	86.9	83.4	81.5	63.8	62.4	81.6	81.1	93.1	58.0	83.8	60.8	92.7	86.0	84.6	84.4	59.0	80.8	68.6	86.1	72.9
Faster RCNN	ResNet-101	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLO [139]	YOLOnet	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD512* [110]	VGG-16	78.5	90.0	85.3	77.7	64.3	58.5	85.1	84.3	92.6	61.3	83.4	65.1	89.9	88.5	88.2	85.5	54.4	82.4	70.7	87.1	75.6
BlitzNet512	ResNet50	79.0	89.9	85.2	80.4	67.2	53.6	82.9	83.6	93.8	62.5	84.0	65.8	91.6	86.6	87.6	84.6	56.8	84.7	73.9	88.0	75.7
BlitzNet512 + COCO	ResNet50	83.8	93.1	89.4	84.7	75.5	65.0	86.6	87.4	94.5	69.9	88.8	71.7	92.5	91.6	91.1	88.9	61.2	90.4	79.2	91.8	83.0

Table 3.2 – **Comparison of detection performance on Pascal VOC 2012 test set.** The models where trained on VOC07 trainval + VOC12 trainval. The BlitzNet models where trained for segmentation jointly with data from VOC12 trainval and extra annotations provided by [70]. Suffix '+ COCO' means that the model was pretrained on the COCO dataset. The reported values correspond to average precision per class (%). Detailed results of submissions are available on the VOC12 test server.

learning rate is set to 10^{-4} and decreased twice during training by a factor 10. We also use a weight decay parameter of 5×10^{-4}.

Modeling setup. As already mentioned, we use ResNet-50 [76] as a feature extractor, 512 feature maps for each layer in down-scale and up-scale streams, 64 channels for intermediate representations in the segmentation branches; BlitzNet300 takes input images of size 300×300 and BlitzNet512 uses 512×512 images. Different versions of the network vary in the stride of the last layer of the upscaling-stream. Strides 4 and 8 in the result tables are denoted as (s4) and (s8) suffix respectively.

network	seg	det	mIoU	mAP
BlitzNet300		✓	-	78.9
BlitzNet300	✓	✓	**72.8**	**80.0**
BlitzNet300	✓		72.4	-

Table 3.3 – **The effect of joint learning on both tasks.** The networks where trained on VOC12 train-seg-aug, and tested on VOC12 val.

network	seg	det	mIoU	mAP
BlitzNet300		✓	-	83.0
BlitzNet300	✓	✓	**75.7**	**83.6**
BlitzNet300	✓		72.4	-

Table 3.4 – **The effect of extra data with bounding box annotations on segmentation performance.** The networks were trained on VOC12 trainval (aug) + VOC07 tainval. Detection performance is measured in average precision (%) and mean IoU is the metric for segmentation segmentation(%).

3.2.3 PASCAL VOC

In this subsection we perform benchmarking of the proposed solution on the task of object detection using Pascal VOC 2007 dataset. This allows us to measure the gains achieved by incorporating semantic segmentation into the pipeline. In the second experiment we study the influence of both tasks on each other using VOC12 dataset with segmentation annotations.

Detection. In this experiment, we train our networks on the union of VOC07 trainval set and VOC12 trainval set; then, we test them on the VOC07 test set. The results are reported in the Table 3.1. For experiments that involve segmentation, we leverage ground truth segmentation masks during training if they are available in VOC12 train-seg-aug or in VOC12 val-seg. When using images of size 300×300 as input, the stochastic gradient descent algorithm is performed by training for 65K iterations with the initial learning rate, which is then decreased after 35K and 50K steps. When training on 512×512 images, we choose the batch size of 16 and learn for $75K$ iterations decreasing the learning rate after 45K and 60K steps.

The results show that BlitzNet300 outperforms SSD300 and YOLO with a 78.5 mAP, while being a real time detector. BlitzNet512 (s8) performs 0.8% better than R-FCN - the most accurate competitive model, scoring 81.2% mAP. We further improve the results by training for detection and segmentation jointly achieving 79.1% and 81.5% mAP with BlitzNet300 (s4) and BlitzNet512 (s8) respectively.

We think that the performance gain for BlitzNet300 over BlitzNet512 could be

network	seg	det	mIoU	mAP
BlitzNet512		✓	-	33.2
BlitzNet512	✓	✓	**53.5**	**34.1**
BlitzNet512	✓		48.3	-

Table 3.5 – **The effect of joint training tested on COCO minival2014.** The networks were trained on COCO train.

explained by the larger stride used for the last layer, which is 4, vs 8 for BlitzNet512, and seems to be helpful for better learning finer details. Unfortunately, training BlitzNet512 with stride 4 was impossible because of memory limitations on our single GPU.

Detection + Segmentation. In this experiment, we use VOC12 train-seg-aug for training and VOC12 val-seg for testing both segmentation and detection. We train the models for 40K steps with the initial learning rate, and then decrease it after 25K and 35K iterations. As Table 3.3 shows, joint training improves accuracy on both tasks comparing to learning a single task. Detection improves by more than 1% while segmentation mIoU grows by 0.4%. We argue that this result could be explained by feature sharing in the universal architecture.

3.2.4 Improving Segmentation with Bounding Box Annotations

To confirm that the tasks benefit from each other due sharing common representations, we conducted another experiment by adding the VOC07 trainval images to VOC12 train-seg-aug for training. Then, the proportion of images that have segmentation annotations to the ones that have detection ones only is 2/1, in contrast to the previous experiments where all the images where annotated for both tasks. To deal with cases where a mini-batch has no images to train for segmentation, we set the corresponding loss to 0 and hence there is no signal to back-propagate through the segmentation stream. Otherwise we use all images that have target segmentation masks in a batch to update the weights. The results presented in Table 3.4 show an improvement of 3.3%. Detection mAP also improves by 0.6%. Figure 3.5 shows that extra data for detection helps to improve classification results and to mitigate confusion between similar categories. In Table 3.2, we report results for these models on the VOC12 test server, which again shows that our results are competitive. More qualitative results, including failure cases, are presented in Appendix C.1.

method	minival2014			test-dev2015		
	int	0.5	0.75	int	0.5	0.75
BlitzNet300	29.7	49.4	31.2	29.8	49.7	31.1
BlitzNet512	34.1	55.1	35.9	34.2	55.5	35.8

Table 3.6 – **Detection performance of BlitzNet on the COCO dataset, with minival2014 and test-dev2015 splits** The networks were trained on COCO trainval dataset. Detection performance is measured in average precision (%) with different criteria, namely, minimum Jaccard overlap between annotated and predicted bounding box is 0.5, 0.75 or integrated from 0.5 to 0.95 % (column "int").

network	backbone	mAP %	FPS	# proposals	input resolution
Faster-RCNN [142]	VGG-16	73.2	7	-	$\sim 1000 \times 600$
R-FCN [103]	ResNet-101	80.5	9	-	$\sim 1000 \times 600$
SSD300* [110]	VGG-16	77.1	46	8732	300×300
SSD512* [110]	VGG-16	80.6	19	24564	512×512
YOLO [139]	YOLO net	63.4	46	-	-
BlitzNet300 (s4)	ResNet-50	79.1	24	45390	300×300
BlitzNet512 (s8)	ResNet-50	81.5	19.5	32766	512×512

Table 3.7 – Comparison of inference time on PASCAL VOC 2007, when running on a Titan X (Maxwell) GPU.

3.2.5 Microsoft COCO Dataset

To further validate the proposed framework, we conduct experiments on the COCO dataset [108]. Here, as explained in Section 3.2.1, we obtain segmentation masks and again training the model on different types of data, i.e., detection, segmentation and both, to study the influence of joint training on detection accuracy.

We train the BlitzNet300 or BlitzNet512 models for 700k iterations in total, starting from the initial learning rate 10^{-4} and then decreasing it after the 400k and 550k iterations by the factor of 10. Table 3.5 shows clear benefits from joint training for both of the tasks on the COCO dataset. To be comparable with other methods, we also report the detection results on COCO test-dev2015 in Table 3.6. Our results are also publicly available on the COCO evaluation test server.

3.2.6 Inference Speed Comparison

In Table 3.7 and Figure 3.4, we report speed comparison to other state-of-the-art detection pipelines. Our approach is the most accurate among the real time

Block type	mAP	mIoU
Hourglass-style [120]	78.7	75.6
Refine-style [132]	78.0	**76.1**
ResSkip (no res)	78.4	75.3
ResSkip (ours)	**79.1**	75.7

Table 3.8 – **The effect of fusion block type on performance, measured on detection (VOC07-test) and segmentation (VOC12-val)** The networks were trained on VOC12-train (aug) + VOC07 tainval, see Sec. 3.2.1. Detection performance is measured in average precision (%) and mean IoU is the metric for segmentation segmentation(%).

detectors working 24 frames per second (FPS) and in the setting close to real time (19 FPS), it provides the most accurate detections among the counterparts, while also providing semantic segmentation mask. Note that all methods are run using the same GPU (Titan X, Maxwell architecture).

3.2.7 Study of the Network Architecture

The BlitzNet pipeline simultaneously operates with several types of data. To demonstrate the effectiveness of the ResSkip block, we set up the following experiment: we leave the pipeline unchanged while only substituting this block with another one. We consider in particular fusion blocks that appear in the state-of-the-art approaches on semantic segmentation [120, 132, 144] where different types of upsampling and linear projections are applied to the input layers before they are fused by summation/concatenation. Table 3.8 shows that our ResSkip block performs similar or better (on average) than all counterparts, which may be due to the fact that its design uses similar skip-connections as the Backbone network ResNet50, making the overall architecture more homogeneous.

Optimal parameters for the size of intermediate representations in segmentation stream (64) as well as the number of channels in the upscale-stream (512) where found by using a validation set. We did not conduct experiments by changing the number of layers in the upscale-stream as long as our architecture is designed to be symmetric with respect to the convolutions and the deconvolutions steps. Reducing the number of the steps will result in a smaller number of layers in the upscale stream, which may deteriorate the performance as noted in [110].

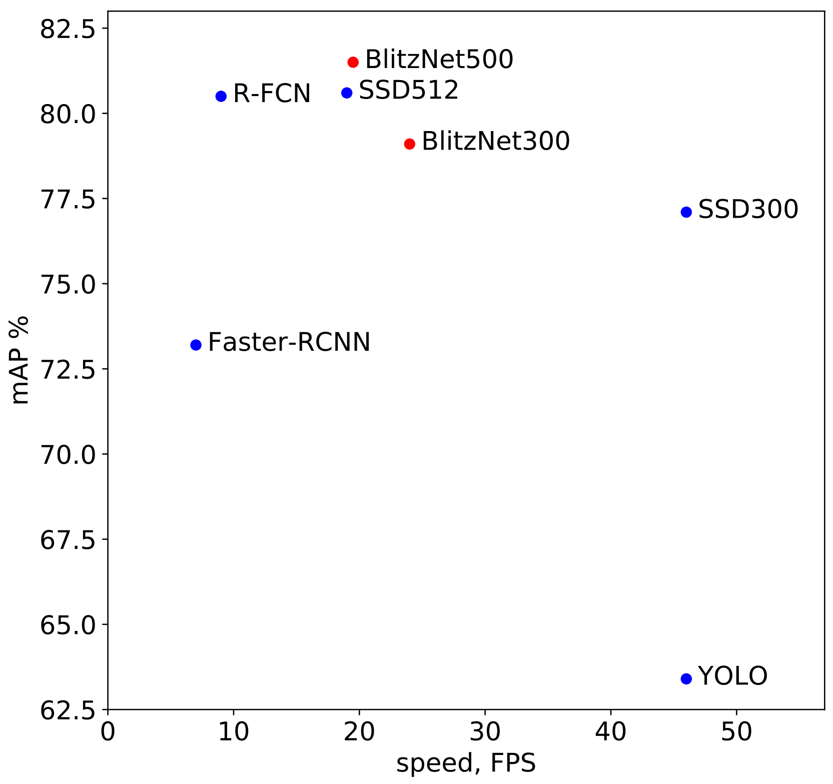


Figure 3.4 – **Speed comparison with other methods.** The detection accuracy of different methods measured in mAP is depicted on y-axis. x-coordinate is their speed, in FPS.

3.3 Conclusion

In this chapter, we introduce a joint approach for object detection and semantic segmentation. By using a single fully-convolutional network to solve both problems at the same time, learning is facilitated by weight sharing between the two tasks, and inference is performed in real time. The main contribution however lies in the way detection and segmentation benefit from each other during the training. The method learns from two types of annotation on each task and achieves state-of-the-art performance among real-time systems, on both problems. Moreover, adding extra images with only bounding box annotations considerably improves semantic segmentation accuracy. This has potential to reduce the amount of work spent on manual image pixel-wise annotation and to open up new ways for annotating scene understanding datasets more effectively.

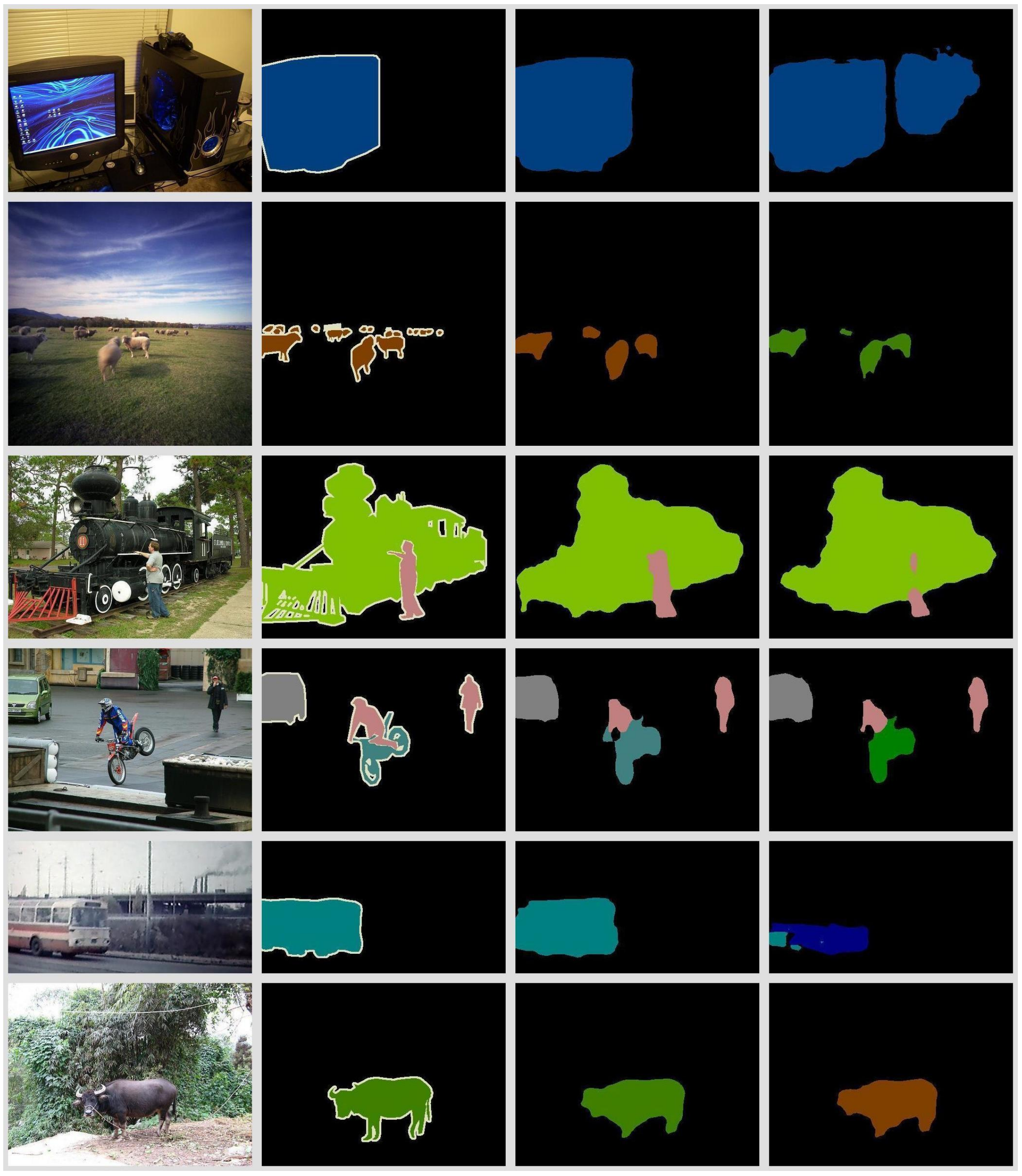


Figure 3.5 – **Effect of extra data annotated for detection on the quality of estimated segmentation masks.** The first column displays test images; the second column contains its segmentation ground truth masks. The third column corresponds to segmentations predicted by BlitzNet300 trained on VOC12 train-segmentation augmented with extra segmentation masks and VOC07. The last row is segmentation masks produced by the same architecture but trained without VOC07.

Chapter 4

The role of visual context for scene understanding

Chapter abstract: Performing data augmentation is known to be important when training deep neural networks for visual recognition, as it helps to reduce overfitting and improves generalization. While simple image transformations can already improve predictive performance in most vision tasks, larger gains can be obtained by leveraging task-specific prior knowledge. In this chapter, we consider object detection, semantic and instance segmentation - the problems where collecting annotations is laborious - and augment the training images by blending objects in existing scenes, using instance segmentation annotations. We observe that randomly pasting objects on images hurts the performance, unless the object is placed in the right context. To resolve this issue, we propose an explicit context model based on a CNN, which predicts whether an image region is suitable for placing an object or not. Our approach improves object detection, semantic and instance segmentation on the PASCAL and COCO datasets, with significant gains in a limited annotation scenario. Moreover, we reduce the need for expensive pixel-wise instance annotations and use weak supervision to extract objects given their bounding boxes. See Sections 2.1-2.4 for related work.

Code for this project is available online [122] (see Appendix B). The concept of modeling visual context with a CNN was first introduced in:

Nikita Dvornik, Julien Mairal, Cordelia Schmid. Modeling Visual Context is Key to Augmenting Object Detection Datasets. Proceedings of the IEEE European Conference on Computer Vision (ECCV), 2018.

However, the material of the chapter is based on a more recent work:

Nikita Dvornik, Julien Mairal, Cordelia Schmid. On the Importance of Visual Context for Data Augmentation in Scene Understanding. IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI) - under minor revision.

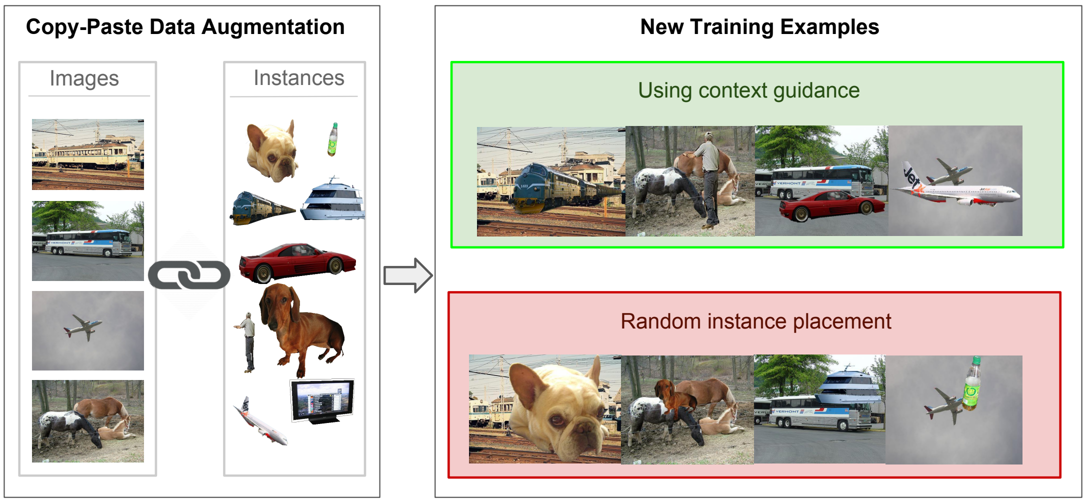


Figure 4.1 – **Examples of data-augmented training examples produced by our approach.** Images and objects are taken from the VOC'12 dataset that contains segmentation annotations. We compare the output obtained by pasting the objects with our context model vs. those obtained with random placements. Even though the results are not perfectly photorealistic and display blending artefacts, the visual context surrounding objects is more often correct with the explicit context model.

Current deep learning systems use data available for training in a completely different way, comparing to classical computer vision methods. As [162] shows for image classification, even if the original training set has hundreds of millions images, adding extra labeled data will further improve the performance by a considerable margin. The problem with scene understanding tasks, such as object detection or segmentation, is that the annotation process is laborious and time-consuming. Expanding original training set by labeling extra images may be simply not feasible. Thus, an alternative way of improving scene understanding performance, is to develop new specialized methods that use limited data more effectively.

To make better use of already annotated data, one can use artificial data augmentation, i.e., creating new training samples by altering original training examples. This may improve performance on a test set significantly if the augmentation strategy is chosen well according to the task. For most vision problems, generic input image transformations such as cropping, rescaling, adding noise, or adjusting colors are usually helpful and may substantially improve generalization. Developing more elaborate augmentation strategies requires then prior knowledge about the task. For example, all categories in the Pascal VOC [47] or ImageNet [145] datasets are invariant to horizontal flips (e.g. a flipped car is still a car). However, flipping would be harmful for hand-written digits from the MNIST dataset [97] (e.g., a flipped "5" is not a digit).

A more ambitious data augmentation technique consists of leveraging segmentation annotations, either obtained manually, or from an automatic segmentation system, and create new images with objects placed at various positions in existing scenes [45, 58, 66]. While not achieving perfect photorealism, this strategy with random placements has proven to be surprisingly effective for *object instance detection* [45], which is a fine-grained detection task consisting of retrieving instances of a particular object from an image collection; in contrast, *object detection/segmentation* and *semantic segmentation* focus on distinguishing between object categories rather than objects themselves and have to account for rich intra-class variability. For these tasks, the random-placement strategy simply does not work, as shown in the experimental section. Placing training objects at unrealistic positions probably forces the detector to become invariant to contextual information and to focus instead on the object's appearance.

Along the same lines, the authors of [66] have proposed to augment datasets for text recognition by adding text on images in a realistic fashion. There, placing text with the right geometrical context proves to be critical. Significant improvements in accuracy are obtained by first estimating the geometry of the scene, before placing text on an estimated plane. Also related, the work of [58] is using successfully such a data augmentation technique for object detection in indoor scene environments. Modeling context has been found to be critical as well and has been achieved by also estimating plane geometry and objects are typically placed on detected tables or counters, which often occur in indoor scenes.

In this chapter, we consider more general tasks of scene understanding such as object detection, semantic and instance segmentation, which require more generic context modeling than estimating planes and surfaces as done for instance in [58,66]. To this end, the first contribution of our chapter is methodological: we propose a context model based on a convolutional neural network. The model estimates the likelihood of a particular object category to be present inside a box given its neighborhood, and then automatically finds suitable locations on images to place new objects and perform data augmentation. A brief illustration of the output produced by this approach is presented in Figure 4.1. The second contribution is experimental: We show with extensive tests on the COCO [108] and VOC'12 benchmarks using different network architectures that context modeling is in fact a key to obtain good results for detection and segmentation tasks and that substantial improvements over non-data-augmented baselines may be achieved when few labeled examples are available. We also show that having expensive pixel-level annotations of objects is not necessary for our method to work well and demonstrate improvement in detection results when using only bounding-box

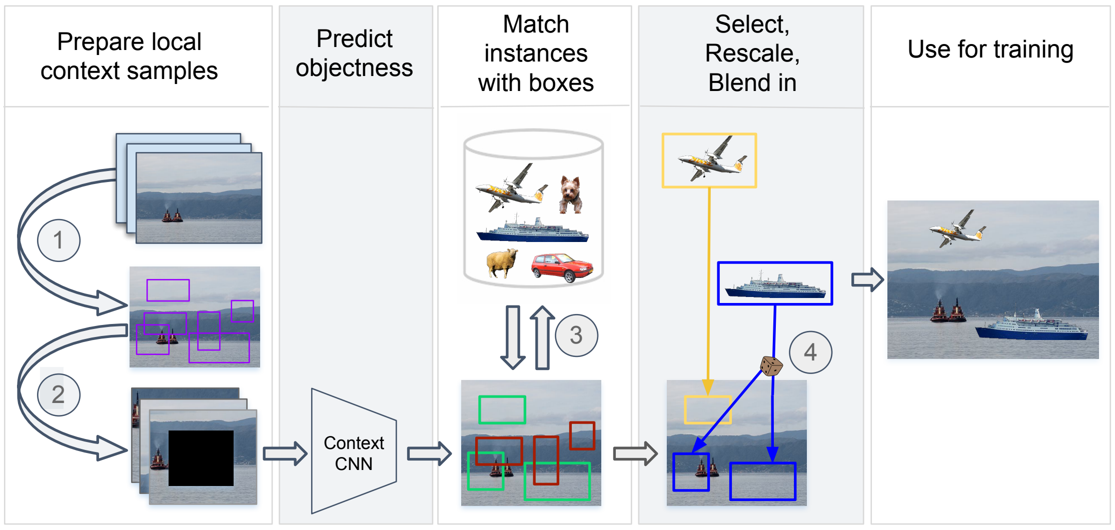

Prepare local context samples
Predict objectness
Match instances with boxes
Select, Rescale, Blend in
Use for training
1
2
Context CNN
3
4

Figure 4.2 – **Illustration of our data augmentation approach.** We select an image for augmentation and 1) generate 200 candidate boxes that cover the image. Then, 2) for each box we find a neighborhood that contains the box entirely, crop this neighborhood and mask all pixels falling inside the bounding box; this "neighborhood" with masked pixels is then fed to the context neural network module and 3) object instances are matched to boxes that have high confidence scores for the presence of an object category. 4) We select at most two instances that are rescaled and blended into the selected bounding boxes. The resulting image is then used for training the object detector.

annotations to extract object masks automatically.

Outline. The chapter is organized as follows: First, we revisit the works studying visual context in scene understanding in Section 4.1. Then, we motivate the need for context modeling in scene understanding, show how to model visual context with CNN, and describe in detail the full augmentation pipeline in Section 4.2. In Section 4.3, we present the experiments where we augment for object detection, semantic and instance segmentation and demonstrate improved performance across different tasks and datasets using various models. Section 4.4 concludes the chapter.

4.1 Related Work

In this section, we discuss related work for visual context modeling for object detection and semantic segmentation and methods suitable for automatic object segmentation.

Modeling visual context for object detection. Relatively early, visual context has been modeled by computing statistical correlation between low-level features of the global scene and descriptors representing an object [166, 167]. Later, the authors of [50] introduced a simple context re-scoring approach operating on appearance-based detections. To encode more structure, graphical models were then widely used in order to jointly model appearance, geometry, and contextual relations [29, 65]. Then, deep learning approaches such as convolutional neural networks started to be used [61, 110, 142]; as mentioned previously, their features already contain implicitly contextual information. Yet, the work of [30] explicitly incorporates higher-level context clues and combines a conditional random field model with detections obtained by Faster-RCNN. With a similar goal, recurrent neural networks are used in [14] to model spatial locations of discovered objects. Another complementary direction in context modeling with convolutional neural networks use a deconvolution pipeline that increases the field of view of neurons and fuse features at different scales [14, 44, 55], showing better performance essentially on small objects. The works of [11, 38] analyze different types of contextual relationships, identifying the most useful ones for detection, as well as various ways to leverage them. However, despite these efforts, an improvement due to purely contextual information has always been relatively modest [179, 185].

Modeling visual context for semantic segmentation. While object detection operates on image's rectangular regions, in semantic segmentation the neighboring pixels with similar values are usually organized together in so-called superpixels [143]. This allows defining contextual relations between such regions. The work of [77] introduces "context clusters" that are discovered and learned from region features. They are later used to define a specific class model for each context cluster. In the work of [178] the authors tile an image with superpixels at different scales and use this representation to build global and local context descriptors. The work of [155] computes texton features [101] for each pixel of an image and defines shape filers on them. This enables the authors to compute local and middle-range concurrence statistics and enrich region features with context information. Modern CNN-based methods on the contrary rarely define an explicit context model and mostly rely on large receptive fields [111]. Moreover, by engineering the network's architecture one can explicitly require local pixel descriptors used for classification to carry global image information too, which enables reasoning with context. To achieve this goal encoder-decoder architectures [8, 44] use deconvolutional operations to propagate coarse semantic image-level information to the final layers while refining details with local information from earlier layers using skip-connections. As an alternative, one can use dilated convolutions [25, 184] that do not down-sample the representation but rather up-sample the filters by introducing "wholes" in them. Doing so is computationally efficient and allows to account for global image statistics in pixel classification. Even though visual context is implicitly present in the networks outputs, it is possible to define an explicit context model [25, 105] on top of them. This usually results in moderate improvement in model's accuracy.

Automatic Instance Segmentation The task of instance segmentation is challenging and requires considerable amount of annotated data [108] in order to achieve good results. Segmentation annotations are the most labor-demanding since they require pixel-level precision. The need to distinguish between instances of one class makes annotating "crowd scenes" extremely time-consuming. If data for this problem comes without labels, tedious and expensive process of annotation may suggests considering other solutions that do not require full supervision. The work of [104] uses various image statistics and hand-crafted descriptors that do not require learning along with annotated image tags, in order to build a segmentation proposal system. With very little supervision, they learn to descriminate between "good" and "bad" instance masks and as a result are able to automatically discover good quality instance segments within the dataset. As an alternative, one can use weakly-supervised methods to estimate instance masks. The authors of [189] use only category image-level annotations in order to train an object segmentation system. This is done by exploiting class-peak responses obtained using pre-trained classification network and propagating them spatially to cover meaningful image segments. It is beneficial to use instance-level annotations, such as object boxes and corresponding categories, if those are available, in order to improve the system's performance. The work of [84] proposes a rather simple yet efficient framework for doing so. By providing the network with extra information, which is a rectangular region containing an object, a system learns to discover instance masks automatically inside those regions. Alternatively, the system could be trained to provide semantic segmentation masks in a weakly-supervised fashion. Together with bounding boxes, one may use it to approximate instance masks.

4.2 Context-driven Data Augmentation by Copy-pasting Instances

In this section, we present a simple experiment to motivate our context-driven data augmentation, and present the full pipeline in details. We start by describing a naive solution to augmenting an object detection dataset, which is to perform copy-paste data augmentation agnostic to context by placing objects at random locations. Next, we explain why it fails for our task and propose a natural solution based on explicit context modeling by a CNN. We show how to apply the context model to perform augmentation for object detection and semantic segmentation and how to blend the object into existing scenes. The full pipeline is depicted in

Figure. 4.2.

4.2.1 Copy-paste Data Augmentation with Random Placement is not Effective for Object Detection

In [45], data augmentation is performed by positioning segmented objects at random locations in new scenes. As mentioned previously, the strategy was shown to be effective for object *instance* detection, as soon as an appropriate procedure is used for preventing the object detector to overfit blending artefacts—that is, the main difficulty is to prevent the detector to "detect artefacts" instead of detecting objects of interest. This is achieved by using various blending strategies to smooth object boundaries such as Poisson blending [134], and by adding "distractors" - objects that do not belong to any of the dataset categories, but which are also synthetically pasted on random backgrounds. With distractors, artefacts occur both in positive and negative examples, for each of the categories, preventing the network to overfit them. According to [45], this strategy can bring substantial improvements for the object instance detection/retrieval task, where modeling the fine-grain appearance of an object instance seems to be more important than modeling visual context as in the general category object detection task.

Unfortunately, the augmentation strategy described above does not improve the results on the general object detection task and may even hurt the performance as we show in the experimental section. To justify the initial claim, we follow [45] as close as possible and conduct the following experiment on the PASCAL VOC12 dataset [47]. Using provided instance segmentation masks we extract objects from images and store them in a so-called instance-database. They are used to augment existing images in the training dataset by placing the instances at random locations. In order to reduce blending artifacts we use one of the following strategies: smoothing the edges using Gaussian or linear blur, applying Poisson blending [134] in the segmented region, blurring the whole image by simulating a slight camera motion or leaving the pasted object untouched. As distractors, we used objects extracted from the COCO dataset [108] belonging to categories not present in the PASCAL VOC[1].

For any combination of blending strategy, by using distractors or not, the naive data augmentation approach with random placement did not improve upon the baseline without data augmentation for the classical object detection task. A possible explanation may be that for instance object detection, the detector does not need to learn intra-class variability of object/scene representations and seems to concentrate only on appearance modeling of specific instances, which is not the

1. Note that external data from COCO was used only in this preliminary experiment and not in the experiments reported later in Section 4.3.

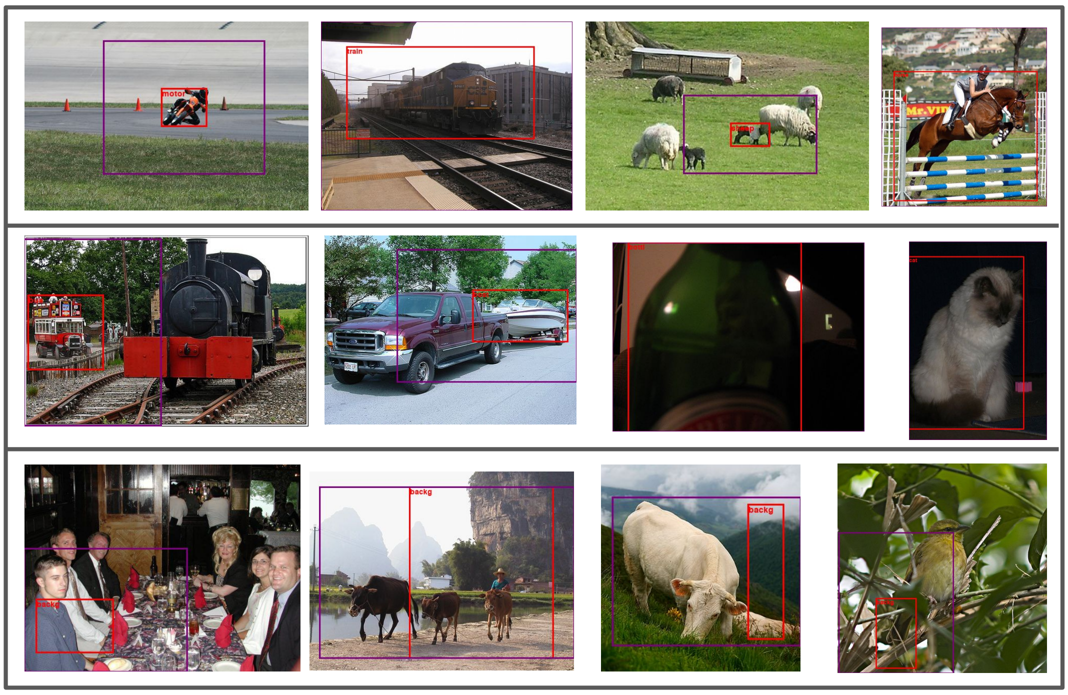


Figure 4.3 – **Contextual images - examples of inputs to the context model**. A subimage bounded by a magenta box is used as an input to the context model after masking-out the object information inside a red box. The top row lists examples of positive samples encoding real objects surrounded by regular and predictable context. Positive training examples with ambiguous or uninformative context are given in the second row. The bottom row depicts negative examples enclosing background. This figure shows that contextual images could be ambiguous to classify correctly and the task of predicting the category given only the context is challenging.

case for category-level object detection. This experiment was the key motivation for proposing a context model, which we now present.

4.2.2 Explicit Context Modeling by CNN

The core idea behind the proposed method is that it is possible to some extent to guess the category of an object just by looking at its visual surroundings. That is precisely what we are modeling by a convolutional neural network, which takes contextual neighborhood of an object as input and is trained to predict the object's class. Here, we describe the training data and the learning procedure in more details.

Contextual data generation. In order to train the contextual model we use a dataset that comes with bounding box and object class annotations. Each ground-truth bounding box in the dataset is able to generate positive "contextual images" that are used as input to the system. As depicted in the Figure 4.3, a "contextual image" is a sub-image of an original training image, fully enclosing the selected bounding box, whose content is masked out. Such a contextual image only carries information about visual neighborhood that defines middle-range context and no explicit information about the deleted object. In order to increase the amount of training samples, we generate multiple context images from one corresponding bounding box by randomly varying the size of the context neighborhood and up-scaling the box to be cut out, as illustrated in Figure. 4.4. Background "contextual images" are generated from bounding boxes that do not contain an object. More formally, we build contextual images from bounding boxes whose maximum intersection over union with any of the object boxes in an image is smaller than 0.3. To prevent distinguishing between positive and background images only by looking at the box shape and to force true visual context modeling, we estimate the shape distribution of positive boxes and sample the background ones from it. The shape is fully characterized by scale s and aspect ratio a. We model their joint distribution empirically by building a 2d histogram 30×30, smoothing it linearly between the bins and drawing a pair (s, a) from this distribution in order to construct a background box. Since in natural images there is more background boxes than the ones actually containing an object, we alleviate the imbalance by sampling background boxes 3 times more often, following sampling strategies in [110, 142].

Model training. Given the set of all contexts, gathered from all training data, we train a convolutional neural network to predict the presence of each object in the masked bounding box. The input to the network are the "contextual images" obtained during the data generation step. These contextual images are resized to 300×300 pixels, and the output of the network is a label in $\{0, 1, ..., C\}$, where C is the number of object categories. The 0-th class represents background and corresponds to a negative "context image". For such a multi-class image classification problem, we use the classical ResNet50 network [76] pre-trained on ImageNet, and change the last layer to be a softmax with $C + 1$ activations (see experimental section for details).

4.2.3 Context-driven Data Augmentation

Once the context model is trained, we use it to provide locations where to paste objects. In this section, we elaborate on the context network inference and describe the precise procedure used for blending new objects into existing scenes.

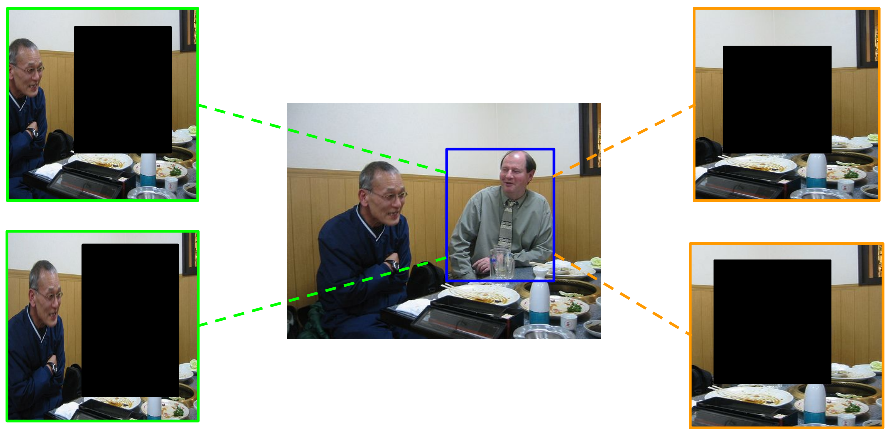


Figure 4.4 – **Different contextual images obtained from a single bounding box.** A single ground-truth bounding box (in blue) is able to generate a set of different context images (in green and orange) by varying the size of the initial box and the context neighborhood. While the orange contextual images may be recognized as a chair, the green ones make it more clear that the person was masked out. This motivates the need to evaluate several context images for one box during the context estimation phase.

Selection of candidate locations for object placement. A location for pasting an object is represented as a bounding box. For a single image, we sample 200 boxes at random from the shape distribution used in 4.2.2 and later select the successful placement candidates among them. These boxes are used to build corresponding contextual images, that we feed to the context model as input. As output, the model provides a set of scores in range between 0 and 1, representing the presence likelihood of each object category in a given bounding box, by considering its visual surrounding. The top scoring boxes are added to the final candidate set. Since the model takes into account not only the visual surroundings but a box's geometry too, we need to consider all possible boxes inside an image to maximize the recall. However this is too costly and using 200 candidates was found to provide good enough bounding boxes among the top scoring ones.
After analyzing the context model's output we made the following observation: if an object of category c is present in an image it is a confident signal for the model to place another object of this class nearby. The model ignores this signal only if no box of appropriate shape was sampled in the object's neighborhood. This often happens when only 200 candidate locations are sampled; however, evaluating more locations would introduce a computational overhead. To fix this issue, we propose a simple heuristic, which consists of drawing boxes in the neighborhood of

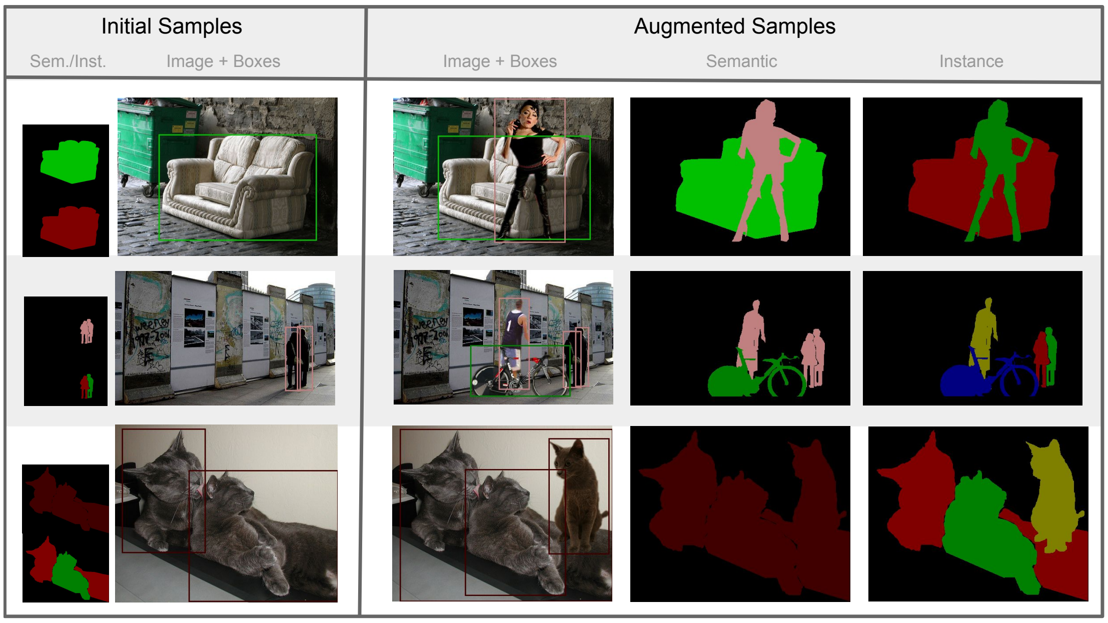


Figure 4.5 – **Data augmentation for different types of annotations.** The first column contains samples from the training dataset with corresponding semantic/instance segmentation and bounding box annotations. Columns 2-4 present the result of applying context-driven augmentation to the initial sample with corresponding annotations.

this object and adding them to the final candidate set. The added boxes have the same geometry (up to slight distortions) as the neighboring object's box.

Candidate scoring process. As noted before, we use the context model to score the boxes by using its softmax output. Since the process of generating a contextual image is not deterministic, predictions on two contextual images corresponding to the same box may differ substantially, as illustrated in Figure 4.4. We alleviate this effect by sampling 3 contextual images for one location and average the predicted scores. After the estimation stage we retain the boxes where an object category has score greater than 0.7; These boxes together with the candidates added at the previous step form the final candidate set that will be used for object placement.

Blending objects in their environment. Whenever a bounding box is selected by the previous procedure, we need to blend an object at the corresponding location. This step follows closely the findings of [45]. We consider different types of blending techniques (Gaussian or linear blur, simple copy-pasting with no post-processing, or generating blur on the whole image to imitate motion), and randomly choose one of them in order to introduce a larger diversity of blending artefacts.

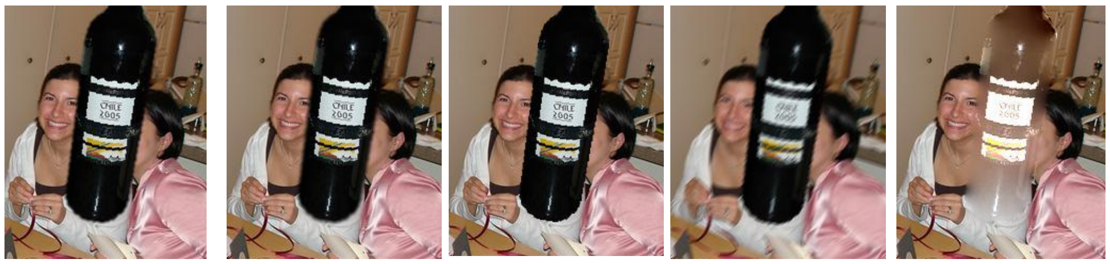


Figure 4.6 – **Different kinds of blending used in experiments.** From left to right: linear smoothing of boundaries, Gaussian smoothing, no processing, motion blur of the whole image, Poisson blending [134].

Figure 4.6 presents the blending techniques mentioned above. We also do not consider Poisson blending in our approach, which was considerably slowing down the data generation procedure. Unlike [45] and unlike our preliminary experiment described in Section 4.2.1, we do not use distractors, which were found to be less important for our task than in [45]. As a consequence, we do not need to exploit external data to perform data augmentation.

Updating image annotation. Once an image is augmented by blending in a new object, we need to modify the annotation accordingly. In this work, we consider data augmention for both object detection semantic/instance segmentation, as illustrated in Figure 4.5. Once a new object is placed in the scene, we generate a bounding box for object detection by drawing the tightest box around that object. In case where an initial object is too occluded by the blended one, i.e., the IoU between their boxes is higher than 0.8, we delete the bounding box of the original object from the annotations. For semantic segmentation, we start by considering augmentation on instance masks (Figure 4.5, column 4) and then convert them to semantic masks (Figure 4.5, column 3). If a new instance occludes more than 80% of an object already present in the scene, we discard annotations for all pixels belonging to the latter instance. To obtain semantic segmentation masks from instance segmentations, each instance pixel is labeled with the corresponding objects class.

4.3 Experimental Results

In this section, we use the proposed context model to augment object detection and segmentation datasets. We start by presenting experimental and implementation details in Sections 4.3.1 and 4.3.2 respectively. In Section 4.3.3 we present a preliminary experiment that motivates the proposed solution. In Sections 4.3.4 and 4.3.4 we study the effect of context-driven data augmentation when augmenting an object detection dataset. For this purpose we consider the Pascal VOC12 dataset that has instance segmentation annotations and we demonstrate the applicability of our method to different families of object detectors. We study the scalability of our approach in Section 4.3.5 by using the COCO dataset for object detection and instance segmentation. We show benefits of our method in Section 4.3.6 by augmenting the VOC12 for semantic segmentation. In Section 4.3.7, we use weakly-supervised learning for estimating object masks and evaluate our approach on the Pascal VOC12 dataset using only bounding box annotations. Finally, Section 4.3.8 studies how the amount of data available for training the context model influences the final detection performance.

4.3.1 Dataset, Tools, and Metrics

Datasets. In our experiments, we use the Pascal VOC'12 [47] and COCO [108] datasets. In the VOC'12 dataset, we only consider a subset that contains segmentation annotations. The training set contains 1 464 images and is dubbed `VOC12train-seg` later on. Following standard practice, we use the test set of VOC'07 to evaluate the detection performance, which contains 4 952 images with the same 20 object categories as VOC'12. We call this image set `VOC07-test`. When evaluating segmentation performance, we use the validation set of the VOC'12 annotated with segmentation masks `VOC12val-seg` that contains 1 449 images.
The COCO dataset [108] is used for large-scale object detection experiments. It includes 80 object categories for detection and instance segmentation. For both tasks, there are 118K images for training that we denote as `COCO-train2017` and 5K for validation and testing denoted as `COCO-val2017`.

Models. To test our data-augmentation strategy we chose a single model capable of performing both object detection and semantic segmentation. BlitzNet [44] is an encoder-decoder architecture, which is able to solve either of the tasks, or both simultaneously if trained with box and segmentation annotations together. The open-source implementation is available online. If used to solve the detection task, BlitzNet achieves close to the state-of-the-art results (79.1% mAP) on `VOC07-test` when trained on the union of the full training and validation parts of VOC'07 and VOC'12, namely `VOC07-train+val` and `VOC12train+val` (see [44]); this network is similar to the DSSD detector of [55] that was also used in the Focal Loss paper [107]. When used as a segmentor, BlitzNet resembles the classical U-Net architecture [144] and also achieves results comparable to the state-of-the-art on VOC'12-test set (75.5% mIoU). The advantage of such class of models is that it is relatively fast (it may work in real time) and supports training with big batches of images without further modification.
To make the evaluation extensive, we also consider a different region-based class of

detectors. For that purpose we employ an open-source implementation of Faster-RCNN [177] which uses ResNet50 [76] architecture as a feature extractor. Finally, when tackling object detection and instance segmentation on COCO, we use Mask-RCNN [75] that solves both tasks simultaneously. For each region proposal the network outputs estimated class probabilities, regressed box offsets and a predicted instance mask. We run the official implementation of [63] that uses ResNet50 as a backbone, followed by an FPN [106] module. This setup corresponds to the current state of the art in object detection and instance segmentation.

Evaluation metric. In VOC'07, a bounding box is considered to be correct if its Intersection over Union (IoU) with a ground truth box is higher than 0.5. The metric for evaluating the quality of object detection and instance segmentation for one object class is the average precision (AP). Mean Average Precision (mAP) is used to report the overall performance on the dataset. Mean Intersection Over Union (mIoU) is used to measure performance on semantic segmentation.

4.3.2 Implementation Details

Training the context model. After preparing the "contextual images" as described in 4.2.2, we re-scale them to the standard size 300×300 and stack them in batches of size 32. We use ResNet50 [76] with ImageNet initialization to train a contextual model in all our experiments. Since we have access only to the training set at any stage of the pipeline we define two strategies for training the context model. When the amount of positive samples is scarce, we train and apply the model on the same data. To prevent overfitting, we use early stopping. In order to determine when to stop the training procedure, we monitor both training error on our training set and validation error on the validation set. The moment when the loss curves start diverging noticeably is used as a stopping point. We call this training setting "small-data regime". When the size of the training set is moderate and we are in "normal-data regime", we split it in two parts ensuring that for each class, there is a similar number of positive examples in both splits. The context model is trained on one split and applied to another one. We train the model with ADAM optimizer [85] starting with learning rate 10^{-4} and decreasing it by the factor of 10 once during the learning phase. The number of steps depends on a dataset. We sample 3 times more background contextual images, as noted in Section 4.2.2. Visual examples of augmented images produced when using the context model are presented in Figure 4.7. Overall, training the context model is about 4-5 times faster than training the detector.

Training detection and segmentation models. In this work, the BlitzNet model takes images of size 300×300 as an input and produces a task-specific output. When used as a detector, the output is a set of candidate object boxes with classification scores and in case of segmentation it is an estimated semantic map of size 75×75;

like our context model, it uses ResNet50 [76] pre-trained on ImageNet as a backbone. The models are trained by following [44], with the ADAM optimizer [85] starting from learning rate 10^{-4} and decreasing it later during training by a factor 10 (see Sections 4.3.4 and 4.3.6 for number of epochs used in each experiment). In addition to our data augmentation approach obtained by copy-pasting objects, all experiments also include classical data augmentation steps obtained by random-cropping, flips, and color transformations, following [44]. For the Faster-RCNN detector training, we consider the classical model of [142] with ResNet50 backbone and closely follow the instructions of [177]. On the Pascal VOC12 dataset, training images are rescaled to have both sides between 600 and 1000 pixels before being passed to the network. The model is trained with the Momentum optimizer for 9 epochs in total. The starting learning rate is set to 10^{-2} and divided by 10 after first 8 epochs of training.

When training Mask-RCNN [75], the images are rescaled to have a maximum size of 1333 pixel on one side or a minimum one of 800 pixels and then grouped in a batch of size 8. We set the starting learning rate to $2 \cdot 10^{-2}$ which is decreased by a factor of 10 twice later during training. For both Faster-RCNN and Mask-RCNN standard data augmentation includes only horizontal flipping.

Selecting and blending objects. Since we widely use object instances extracted from the training images in all our experiments, we create a database of objects cut out from the `VOC12train-seg` or `COCO-train` sets to quickly access them during training. For a given candidate box, an instance is considered as matching if after scaling it by a factor in $[0.5, 1.5]$ the re-scaled instance's bounding box fits inside the candidate's one and takes at least 80% of its area. The scaling factor is kept close to 1 not to introduce scaling artefacts. When blending the objects into the new background, we follow [45] and use randomly one of the following methods: adding Gaussian or linear blur on the object boundaries, generating blur on the whole image by imitating motion, or just paste an image with no blending. By introducing new instances in a scene we may also introduce heavy occlusions of existing objects. The strategy for resolving this issue depends on the task and is clarified in Sections 4.3.4 and 4.3.6.

4.3.3 Why is Random Placement not Working?

As we discovered in the Section 4.2.1, random copy-paste data augmentation does not bring improvement when used to augment object detection datasets. There are multiple possible reasons for observing this behavior, such as violation of context constraints imposed by the dataset, objects looking "out of the scene" due to different illumination conditions or simply artifacts introduced due to blending techniques. To investigate this phenomenon, we conduct a study, that aims to better understand (i) the importance of visual context for object detection, (ii)

Method	aero	bike	bird	boat	bottle	average
Base-DA	58.8	64.3	48.8	47.8	33.9	48.7
Random-DA	60.2	66.5	55.1	41.9	29.7	48.3
Removing context	44.0	46.8	42.0	20.9	15.5	33.9
Enlarge + Reblend-DA	60.1	63.4	51.6	48.0	34.8	51.6

Table 4.1 – Ablation study on the first five categories of VOC'12. All models are learned independently. We compare classical data augmentation techniques (Base-DA), approaches obtained by copy-pasting objects, either randomly (Random-DA) or by preserving context (Enlarge+Reblend-DA). The line "Removing context" corresponds to the first experiment described in Section 4.3.3; Enlarge-Reblend corresponds to the second experiment.

the role of illumination conditions and (iii) the impact of blending artefacts. For simplicity, we choose the first 5 categories of VOC'12, namely *aeroplane, bike, bird, boat, bottle*, and train independent detectors per category.

Baseline when no object is in context. To confirm the negative influence of random placing, we consider one-category detection, where only objects of one selected class are annotated with bounding boxes and everything else is considered as background. Images that do not contain objects of the selected category become background images. After training 5 independent detectors as a baseline, we construct a similar experiment by learning on the same number of instances, but considering as positive examples only objects that have been synthetically placed in a random context. This is achieved by removing from the training data all the images that have an object from the category we want to model, and replacing it by an instance of this object placed on a background image. The main motivation for such study is to consider the extreme case where (i) no object is placed in the right context; (iii) all objects may suffer from rendering artefacts. As shown in Table 4.1, the average precision degrades significantly by about 14% compared to the baseline. As a conclusion, either visual context is indeed crucial for learning, or blending artefacts is also a critical issue. The purpose of the next experiment is to clarify this ambiguity.

Impact of blending when the context is right. In the previous experiment, we have shown that the lack of visual context and the presence of blending artefacts may explain the performance drop observed in the third row of Table 4.1. Here, we propose a simple experiment showing that neither (iii) blending artefacts nor (ii) illumination difference are critical when objects are placed in the right context: the experiment consists of extracting each object instance from the dataset, up-scale it

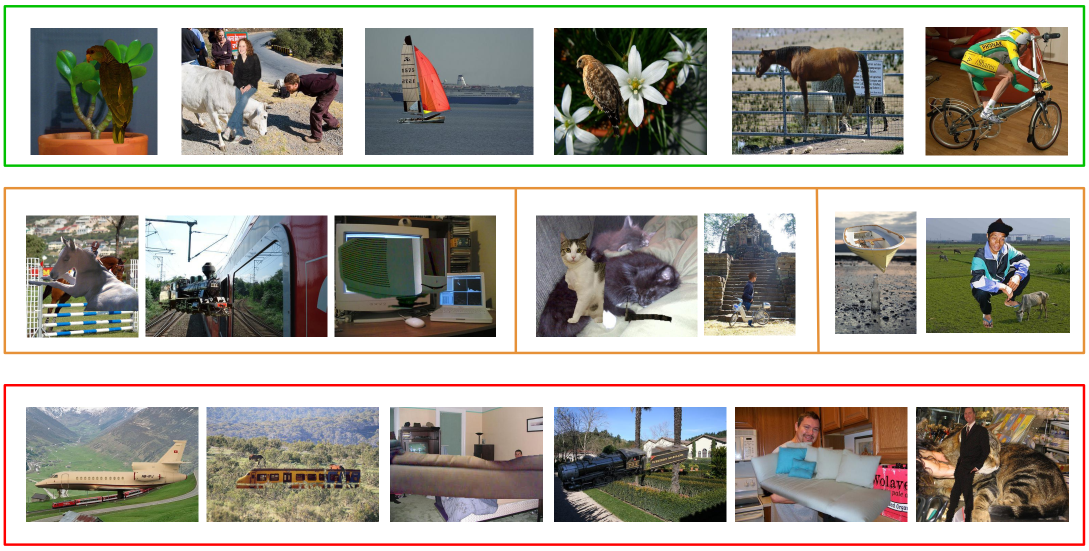


Figure 4.7 – **Examples of instance placement with context model guidance.** The figure presents samples obtained by placing a matched examples into the box predicted by the context model. The top row shows generated images that are visually almost indistinguishable from the real ones. The middle row presents samples of good quality although with some visual artifacts. For the two leftmost examples, the context module proposed an appropriate object class, but the pasted instances do not look visually appealing. Sometimes, the scene does not look natural because of the segmentation artifacts as in the two middle images. The two rightmost examples show examples where the category seems to be in the right environment, but not perfectly placed. The bottom row presents some failure cases.

by a random factor slightly greater than one (in the interval $[1.2, 1.5]$), and blend it back at the same location, such that it covers the original instance. To mimic the illumination change we apply a slight color transformation to the segmented object. As a result, the new dataset benefits slightly from data augmentation (thanks to object enlargement), but it also suffers from blending artefacts for *all object instances*. As shown on the forth row of Table 4.1, this approach improves over the baseline, which suggests that the lack of visual context is probably the key explaining the result observed before. The experiment also confirms that the presence of difference in illumination and blending artefacts is not critical for the object detection task. Visual examples of such artefacts are presented in Figure 4.8.

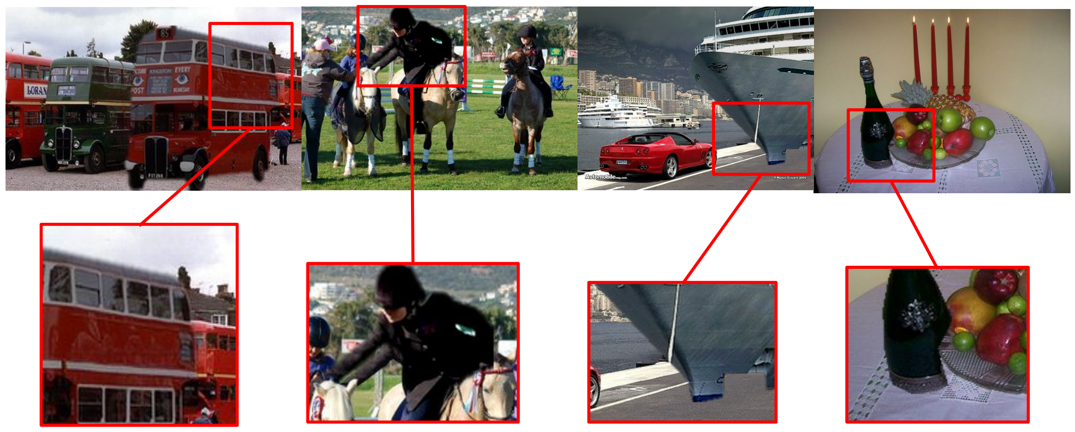


Figure 4.8 – **Illustration of artifacts arising from enlargement augmentation.** In the enlargement data augmentation, an instance is cut out of the image, up-scaled by a small factor and placed back at the same location. This approach leads to blending artefacts. Modified images are given in the top row. Zoomed parts of the images centered on blending artifacts are presented in the bottom line.

method	aero	bike	bird	boat	bott.	bus	car	cat	chair	cow	table	dog	horse	mbike	pers.	plant	sheep	sofa	train	tv	avg.
Base-DA	58.8	64.3	48.8	47.8	33.9	66.5	69.7	68.0	40.4	59.0	61.0	56.2	72.1	64.2	66.7	36.6	54.5	53.0	73.4	63.6	58.0
Random-DA	60.2	66.5	55.1	41.9	29.7	66.5	70.0	70.1	37.4	57.4	45.3	56.7	68.3	66.1	67.0	37.0	49.9	55.8	72.1	62.6	56.9
Enlarge-DA	60.1	63.4	51.6	48.0	34.8	68.8	72.1	70.4	41.1	63.7	62.3	56.3	70.1	67.8	65.3	37.9	58.1	61.2	75.5	65.9	59.7
Context-DA	68.9	73.1	62.5	57.6	38.9	72.5	74.8	77.2	42.9	69.7	59.5	63.9	76.1	70.2	69.2	43.9	58.3	59.7	77.2	64.8	64.0
Impr. Cont.	**10.1**	**8.7**	**13.7**	**9.2**	5.0	6.0	5.1	**9.2**	2.5	**10.7**	1.5	**7.5**	4.0	6.0	2.5	**7.3**	3.8	6.7	4.2	1.2	5.8

Table 4.2 – Comparison of detection accuracy on `VOC07-test` for the single-category experiment. The models are trained independently for each category, by using the 1 464 images from `VOC12train-seg`. The first row represents the baseline experiment that uses standard data augmentation techniques. The second row uses in addition copy-pasting of objects with random placements. “Enlarge-DA” augmentation blends up-scaled instances back in their initial location, which is given in row 3. The forth row presents the results achieved by our context-driven approach and the last row presents the improvement it brings over the baseline. The numbers represent average precision per class in %. Large improvements over the baseline (greater than 7%) are in bold. All numbers are averaged over 3 experiments.

model	CDA	aero	bike	bird	boat	bott.	bus	car	cat	chair	cow	table	dog	horse	mbike	pers.	plant	sheep	sofa	train	tv	avg.
BlitzNet300		63.6	73.3	63.2	57.0	31.5	76.0	71.5	79.9	40.0	71.6	61.4	74.6	80.9	70.4	67.9	36.5	64.9	63.0	79.3	64.7	64.6
	✓	69.9	73.8	63.9	62.6	35.3	78.3	73.5	80.6	42.8	73.8	62.7	74.5	81.1	73.2	68.9	38.1	67.8	64.3	79.3	66.1	**66.5**
F-RCNN		65.8	70.9	66.5	54.6	45.9	72.7	72.9	80.3	36.8	70.3	48.0	78.9	70.7	70.6	66.3	33.1	64.7	59.8	71.8	61.1	63.1
	✓	67.4	67.7	64.9	58.0	50.4	71.6	74.9	80.4	36.8	70.2	56.4	75.7	73.7	71.6	71.5	39.4	68.6	63.5	67.7	60.1	**64.5**

Table 4.3 – Comparison of detection accuracy on `VOC07-test` for the multiple-category experiment. The model is trained on all categories at the same time, by using the 1 464 images from `VOC12train-seg`. The first column specifies the detector used in the experiment, the second column notes if Context-driven Data Augmentation (CDA) was used. The numbers represent average precision per class in %.

4.3.4 Object Detection Augmentation on VOC PASCAL

In this subsection, we are conducting experiments on object detection by augmenting the PASCAL VOC'12 dataset. In order to measure the impact of the proposed technique in a "small data regime", we pick the single-category detection scenario and also consider a more standard multi-category setting. We test single-shot region-based families of detectors—with BlitzNet and Faster-RCNN respectively—and observe improved performance in both cases.

Single-category Object Detection

In this section, we conduct an experiment to better understand the effect of the proposed data augmentation approach, dubbed "Context-DA" in the different tables, when compared to a baseline with random object placement "Random-DA", and when compared to standard data augmentation techniques called "Base-DA". The study is conducted in a single-category setting, where detectors are trained independently for each object category, resulting in a relatively small number of positive training examples per class. This allows us to evaluate the importance of context when few labeled samples are available and see if conclusions drawn for a category easily generalize to other ones.

The baseline with random object placements on random backgrounds is conducted in a similar fashion as our context-driven approach, by following the strategy described in the previous section. For each category, we treat all images with no object from this category as background images, and consider a collection of cut instances as discussed in Section 4.3.1. During training, we augment a negative (background) image with probability 0.5 by pasting up to two instances on it, either at randomly selected locations (Random-DA), or using our context model in the selected bounding boxes with top scores (Context-DA). The instances are re-scaled by a random factor in $[0.5, 1.5]$ and blended into an image using a randomly selected

method	aero	bike	bird	boat	bott.	bus	car	cat	chair	cow	table	dog	horse	mbike	pers.	plant	sheep	sofa	train	tv	avg.
Base-DA	79.0	43.7	65.8	57.9	53.8	83.8	77.9	76.7	19.2	56.6	46.6	67.6	59.0	73.1	77.9	46.8	69.4	37.8	73.7	70.3	63.3
Random-DA	78.1	47.1	75.4	57.8	57.2	83.5	76.2	76.6	20.5	57.0	43.1	69.2	57.5	71.5	78.2	40.0	63.3	42.0	74.5	64.1	63.1
Enlarge-DA	77.2	45.4	67.9	57.9	61.0	84.1	78.8	76.3	20.3	58.4	46.9	67.5	60.5	73.9	78.1	45.2	71.1	38.8	73.6	71.1	64.1
Context-DA	81.7	46.4	73.4	60.7	59.4	85.3	78.8	79.1	20.6	60.0	48.0	68.1	62.2	75.3	78.8	47.6	71.6	39.9	73.6	70.3	65.4
Impr. Cont.	**2.7**	**2.7**	**7.6**	**2.8**	**4.6**	1.5	1.1	2.3	1.4	**3.4**	1.4	0.5	**3.2**	2.3	2.2	0.9	0.8	2.1	-0.1	0	2.1

Table 4.4 – Comparison of segmentation accuracy on `VOC12val-seg`. The model is trained on all 20 categories by using the 1 464 images from `VOC12train-seg`. Base-DA represents the baseline experiment that uses standard data augmentation techniques. Context-DA uses also our context-driven data augmentation. Random-DA is its context-agnostic analogue. Enlarge-DA corresponds to randomly enlarging an instance and blending it back. The last row presents absolute improvement over the baseline. The numbers represent IoU per class in %. Categories enjoying an improvement higher than 2.5% are in bold. All numbers are averaged over 3 independent experiments.

blending method mentioned in Section 4.3.1. For all models, we train the object detection network for 6K iterations and decrease the learning rate after 2K and 4K iterations by a factor 10 each time. The context model was trained in "small-data regime" for 2K iterations and the learning rate was dropped once after 1.5K steps. The results for this experiment are presented in Table 4.2.

The conclusions are the following: random placement indeed hurts the performance on average. Only the category bird seems to benefit significantly from it, perhaps because birds tend to appear in various contexts in this dataset and some categories significantly suffer from random placement such as boat, table, and sheep. Importantly, the visual context model always improves upon the random placement one, on average by 7%, and upon the baseline that uses only classical data augmentation, on average by 6%. Interestingly, we identify categories for which visual context is crucial (aeroplane, bird, boat, bus, cat, cow, dog, plant), for which context-driven data augmentation brings more than 7% improvement and some categories that display no significant gain or losses (chair, table, persons, tv), where the difference with the baseline is less noticeable (around 1-3%).

Multiple-Categories Object Detection

In this section, we conduct the same experiment as in Section 4.3.4, but we train a single multiple-category object detector instead of independent ones per category. Network parameters are trained with more labeled data (on average 20 times more than for models learned in Table 4.2). When training the context model, we follow the "normal-data strategy" described in Section 4.3.2 and train

Model	DA	@0.5:0.95	@0.5	@0.75	S	M	L
Object Detection							
BlitzNet300		27.3	46.0	28.1	10.7	26.8	46.0
BlitzNet300	Rnd	26.8	45.0	27.6	9.3	26.0	45.7
BlitzNet300	Cont	**28.0**	**46.7**	**28.9**	10.7	**27.8**	**47.0**
Mask-RCNN		38.6	59.7	42.0	22.1	41.5	50.6
Mask-RCNN	Rnd	36.9	57.3	39.7	20.5	39.6	48.0
Mask-RCNN	Cont	**39.1**	**60.3**	**42.3**	**22.4**	**42.2**	**51.2**
Instance Segmentation							
Mask-RCNN		34.5	56.5	36.3	15.7	37.1	52.1
Mask-RCNN	Rnd	33.6	55.2	35.8	14.8	35.5	50.0
Mask-RCNN	Cont	**34.8**	**57.0**	**36.5**	**15.9**	**37.6**	**52.5**

Table 4.5 – Comparison of object detection and instance segmentation accuracy on `COCO-val2017` for the multiple-category experiment. The model is trained on all categories at the same time, by using the 118 783 images from `COCO-train2017`. The first column specifies a model used to solve a task, the second column notes if Context-driven (Cont) or random-placement (Rnd) Data Augmentation was used. For different IoU thresholds @0.5:0.95, @0.5 and @0.75) and for different object size (S, M, L), the numbers represent mAP in %. Best results are in bold.

the model for 8K iterations, decreasing the learning rate after 6K steps. The results are presented in Table 4.3 and show a modest average improvement of 2.1% for a single shot and 1.4% for a region-based detector on average over the corresponding baselines, which is relatively consistent across categories. This confirms that data augmentation is crucial when few labeled examples are available.

4.3.5 Object Detection and Instance Segmentation Augmentation on COCO

In order to test our augmentation technique at large scale, we use in this section the COCO dataset [108] whose training set size is by 2 orders of magnitude larger than `voc12train-seg`, and consider both object detection and instance

Aug. type	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	pers.	plant	sheep	sofa	train	tv	avg.
Inst seg	**68.9**	**73.1**	**62.5**	**57.6**	**38.9**	**72.5**	**74.8**	**77.2**	**42.9**	**69.7**	59.5	**63.9**	76.1	70.2	**69.2**	**43.9**	58.3	**59.7**	77.2	64.8	**64.0**
Gt seg	67.8	70.3	61.5	56.6	38.2	71.2	74.7	75.7	41.6	68.3	59.0	63.2	75.6	71.0	68.7	42.6	59.5	59.1	**78.4**	**65.3**	63.4
Weak seg	68.9	71.3	59.0	54.2	37.3	71.9	74.5	75.2	40.8	67.6	**59.8**	62.8	**76.4**	**71.3**	68.4	43.8	**59.9**	57.2	76.6	64.4	63.0
No	58.8	64.3	48.8	47.8	33.9	66.5	69.7	68.0	40.4	59.0	61.0	56.2	72.1	64.2	66.7	36.6	54.5	53.0	73.4	63.6	58.0

Table 4.6 – Comparison of detection accuracy on `VOC07-test` for the single-category experiment. The models are trained independently on each category, by using the `VOC12train-seg`. The first column specifies the type of object mask used for augmentation: ground-truth instance segmentations (Inst. Seg.), ground-truth semantic segmentation (GT Seg.), or weakly-supervised semantic segmentations (Weak Seg.). Inst. seg. stands for the original instance segmentation ground truth masks. The numbers represent AP per class in %. The best result for a category is in bold. All numbers are averaged over 3 independent experiments.

Aug. type	aero	bike	bird	boat	bott.	bus	car	cat	chair	cow	table	dog	horse	mbike	pers.	plant	sheep	sofa	train	tv	avg.
Inst. seg	**69.9**	73.8	**63.9**	**62.6**	35.3	**78.3**	73.5	80.6	**42.8**	**73.8**	62.7	**74.5**	81.1	73.2	68.9	38.1	**67.8**	64.3	79.3	**66.1**	**66.5**
GT Seg.	68.7	74.5	60.1	60.0	34.9	75.4	**74.4**	**81.7**	41.1	72.4	**64.2**	74.4	**81.3**	**74.6**	**69.6**	**39.7**	67.6	64.2	**80.4**	65.5	66.2
Weak Seg.	69.2	**75.2**	63.2	59.8	**35.6**	77.1	73.4	78.7	41.3	72.9	62.8	72.7	79.6	72.5	68.1	39.2	67.6	**66.1**	79.5	64.2	65.9
No	63.6	73.3	63.2	57.0	31.5	76.0	71.5	79.9	40.0	71.6	61.4	74.6	80.9	70.4	67.9	36.5	64.9	63.0	79.3	64.7	64.6

Table 4.7 – Comparison of detection accuracy on `VOC07-test` for the multy-category experiment depending on the type of object masks used for augmentation. The models are trained on all categories together, by using the 1 464 images from `VOC12train-seg`. The first column specifies the type of object mask used for augmentation: ground-truth instance segmentations (Inst. Seg.), ground-truth semantic segmentation (GT Seg.), or weakly-supervised semantic segmentations (Weak Seg.). Inst. seg. stands for the original instance segmentation ground truth masks. The numbers represent AP per class in %. The best result for a category is in bold. All numbers are averaged over 3 independent experiments.

segmentation tasks.

Object Detection with BlitzNet

By design, the experiment is identical to the one presented in Section 4.3.4. However, for the COCO dataset we need to train a new context model. This is done by training for 350K iterations (decay at 250K) as described in Section 4.3.2. The non data-augmented baseline was trained according to [44]; when using our augmentation pipeline, we train the detector for 700K iterations and decrease the learning rate by a factor of 10 after 500K and 600K iterations. Table 4.5 shows that we are able to achieve a modest improvement of 0.7%, and that data augmentation still works and does not degrade the performance regardless the large amount of data available for training initially.

Detection and Segmentation with Mask-RCNN

For this experiment, we use Mask-RCNN [75] that jointly solves object detection and instance segmentation. When training the baseline model, we closely follow original guidelines[2] and train the model with 2x schedule (for 180K iterations) to maximize the baseline's performance. Training the model with 1x schedule (for 90K iterations) results in underfitting, while training with x4 schedule (for 360K iterations), results in overfitting. In order to improve the performance of Mask-RCNN for both tasks, we train the model with x4 schedule and use the context-driven data augmentation. In order to reduce pasting artifacts negatively affecting Mask-RCNN, we decrease the augmentation probability during the training. More precisely, augmentation probability is set to 0.5 in the beginning of the training and then linearly decreased to 0 by the end of the training procedure. Training with constant augmentation probability did not improve the performance over the x2 baseline. On the other hand, gradually reducing augmentation probability results in less aggressive regularization and brings more benefits when training on a large dataset, such as COCO. As Table 4.5 shows, following this augmentation strategy results in a 0.5% an 0.3% mAP improvement for detection and segmentation respectively, when comparing to the most accurate baseline Mask-RCNN, trained with x2 schedule. Augmenting the training data with random placement strategy hurts the performance substantially, which highlights the importance of context for data augmentation.

4.3.6 Semantic Segmentation Augmentation

In this section, we demonstrate the benefits of the proposed data augmentation technique for the task of semantic segmentation by using the VOC'12 dataset. First, we set up the baseline by training the BlitzNet300 [44] architecture for semantic segmentation. Standard augmentation techniques such as flipping, cropping, color transformations and adding random noise were applied during the training, as described in the original paper. We use `voc12train-seg` subset for learning the model parameters. Following the training procedure described in Section 4.3.2, we train the model for 12K iterations starting from the learning rate of 10^{-4} and decreasing it twice by the factor of 10, after 7K and 10K steps respectively. Next, we perform data augmentation of the training set with the proposed context-driven strategy and train the same model for 15K iterations, dropping the learning rate at 8K and 12K steps. In order to blend new objects in and to augment the

2. `https://github.com/facebookresearch/Detectron/blob/master/MODEL_ZOO.md`

ground truth we follow routines described in Section 4.2.3. We also carry out an experiment where new instances are placed at random locations, which represents a context-agnostic counterpart of our method. We summarize the results of all 3 experiments in Table 4.4. As we can see from the table, performing copy-paste augmentation at random locations for semantic segmentation slightly degrades the model's performance by 0.2%. However when objects are placed in the right context, we achieve a boost of 2.1% in mean intersection over union. These results resemble the case of object detection a lot and therefore highlight the role of context in scene understanding. We further analyze the categories that benefit from our data augmentation technique more than the others. If improvement for a class AP over the baseline is higher than 2.5%, Table 4.4 marks the result in bold. Again, we can notice correlation with the detection results from Section 4.3.4 which demonstrates the importance of context for the categories that benefit from our augmentation strategy in both cases.

4.3.7 Reducing the need for pixel-wise object annotation

Our data augmentation technique requires instance-level segmentations, which are not always available in realistic scenarios. In this section, we relax the annotation requirements for our approach and show that it is possible to use the method when only bounding boxes are available.

Semantic segmentation + bounding box annotations. Instance segmentation masks provide annotations to each pixel in an image and specify (i) an instance a pixel belongs to and (ii) class of that instance. If these annotations are not available, one may approximate them with semantic segmentation and bounding box annotations. Figure 4.9 illustrates possible annotation types and the difference between them. Semantic segmentation annotations are also pixel-wise, however they annotate each pixel only with the object category. Instance-specific information could be obtained from object bounding boxes, however this type of annotation is not pixel-wise and in some cases is not sufficient to assign each pixel to the correct instance. As Figure 4.9 suggests, as long as a pixel in semantic map is covered by only one bounding box, it uniquely defines the object it belongs to (row 1); otherwise, if more than one box covers the pixel, it is not clear which object it comes from (row 2). When deriving approximate instance masks from semantic segmentation and bounding boxes (see Figure 4.9, column 2), we randomly order the boxes and assign pixels from a semantic map to the corresponding instances. Whenever a pixel could be assigned to multiple boxes we choose a box that comes first in the ordering. Once the procedure for obtaining object masks is established we are back to the initial setting and follow the proposed data augmentation routines described above. As could be seen in Tables 4.6 and 4.7 detection performance expiriences a slight drop of 0.6% in single-category and 0.3% in multi-category

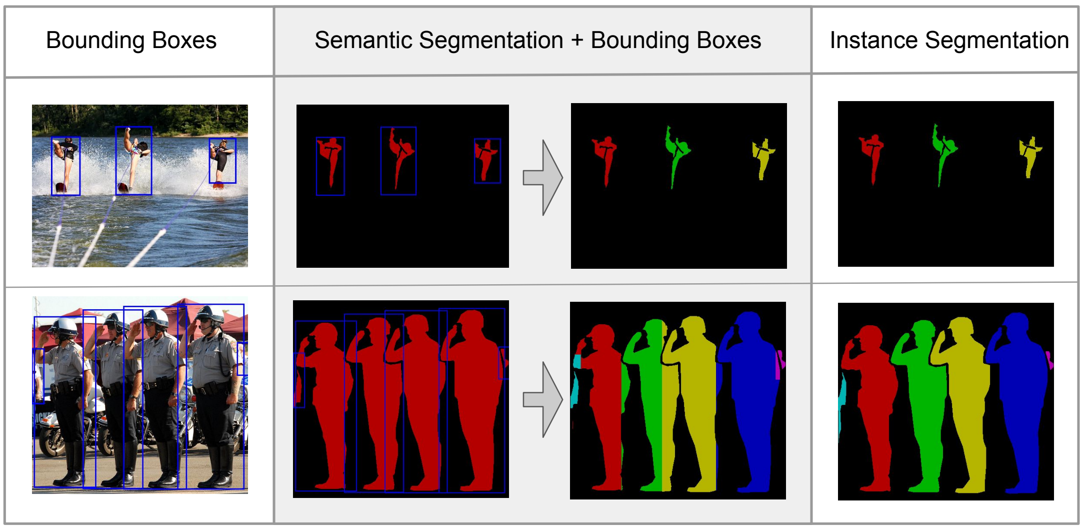


Figure 4.9 – **Possible types of instance-level annotation.** The left column presents an image annotated with object boxes. Column 2 shows semantic segmentation annotations with object boxes on top and approximate instance segmentations derived from it. The last column presents the original instance segmentation annotations.

settings respectively, comparing to using instance segmentation masks. These results are promising and encourage us to explore less elaborate annotations for the purpose of data augmentation.

Bounding box annotations only. Since we have an established procedure for performing data augmentation with semantic segmentation and bounding boxes annotations, the next step to reducing pixel-wise annotation is to approximate segmentation masks. We employ weakly-supervised learning to estimate segmentations from available bounding boxes. The work of [84] proposes an effective solution to this problem. When trained on the `VOC12train` dataset, augmented with more training examples according to [65, 84], it achieves 65.7% mIoU on the `VOC12val-set`. Unfortunately, we have found that naively applying this solution for estimating segmentation masks and using them for augmentation results in worse performance. The reason for that was low quality of estimated masks. First, inaccurate object boundaries result in non-realistic instances and may introduce biases in the augmented dataset. But more importantly, confusion between classes may hampers the performance. For example, augmenting a category "cow" with examples of a "sheep" class may hurt the learning process. Hence, we need a model with a more discriminative classifier. To this end we propose the following modifications to the segmentation method: we change the architecture from

% of data used	0	5	10	25	50	75	100
Det. mAP	64.6	65.3	66.1	66.4	66.7	66.9	66.9
Seg. mIoU	63.3	64.6	65.1	65.3	65.5	65.9	66.0

Table 4.8 – Object detection and semantic segmentation performance depending on amount of data used for building the context model. First row depicts the portion (in %) of the `VOC07trainval+VOC12trainval` used for training the context model. Second column corresponds to performance of baseline models. The second row gives the final detection mAP % evaluated on `VOC07test`, while the third row lists segmentation mIou in % on `VOC12val-seg`. For both tasks we used BlitzNet300 trained on augmented `VOC12train-seg`.

DeepLab_v1 [24] to DeepLab_v4 [25], perform multi-scale inference and process the resulting masks with a conditional random field. The later helps to refine the object edges, which was found not necessary in the original work of [25], when learning with full supervision. By training on the same data as the original method of [84] but with the proposed modifications we achieve 75.8% mIoU, which is more than 10% improvement to the initial pipeline. This accuracy seems to be sufficient to use automatically-estimated segmentation masks for augmentation purposes.

When the semantic maps are estimated, we follow the augmentation routines of the previous section with only one difference; specifically, an instance is kept if the bounding box of its segmentation covers at least 40% of its corresponding ground truth box. Otherwise, the instance mask is considered as missing and the object does not contribute to data augmentation. The results of applying this strategy to the single- and multy-category object detection are presented in Table 4.6 and 4.7, respectively. Table 4.6 shows which categories are unable to provide high-quality masks, even though the quality seems to be sufficient to improve upon the non-augmented baseline. It is surprising that by using object boxes instead of segmentation masks we lose only 0.6% of mAP in the multi-class scenario while still outperforming non-augmented training by 1.6%. These results show that the method is widely applicable even in the absence of segmentation annotations.

4.3.8 Studying the Importance of Context Modeling Quality for Scene Understanding

First, we make an assumption that the quality of a context model is mainly influenced by the amount of data it has received for training. Hence, to study this relation, we mine a bigger dataset `VOC07-trainval+VOC12-trainval` which results

method	aero	bike	bird	boat	bott.	bus	car	cat	chair	cow	table	dog	horse	mbike	pers.	plant	sheep	sofa	train	tv	avg.
Base-DA	63.4	69.0	51.3	51.5	32.8	68.9	70.0	70.4	36.2	62.3	60.5	63.7	73.4	65.8	64.3	32.4	61.2	56.2	74.1	59.9	59.4
Context-DA	64.8	71.7	54.6	51.8	34.4	71.3	72.3	68.1	40.1	64.4	63.1	63.6	76.1	67.1	65.8	37.0	63.4	53.3	74.7	60.9	61.1
Impr. Cont.	1.4	2.7	3.3	0.3	1.6	2.4	2.3	-2.3	3.9	2.1	2.6	-0.1	2.7	1.3	1.5	4.6	2.2	-2.9	0.6	1.0	1.7

Table 4.9 – Little data, multi-category object detection on `VOC07-test`. The models are trained independently for each category, by using the 600 images from `VOC12train-seg`. The first row represents the baseline experiment that uses standard data augmentation techniques. The second row presents the results achieved by our context-driven approach and the last row presents the improvement it brings over the baseline. The numbers represent average precision per class in %.

in 16551 images. Then, we proceed by taking subsets of this dataset of increasing size and train the context model on them. Finally, we use the obtained context models to augment `VOC12-trainval` and train BlitzNet300 on it for detection and segmentation. Table 4.8 summarizes the object detection performance on `VOC07-test` and semantic segmentation performance on `VOC12val-seg`. In the current experiment, 10% of the full set (1655 images) is roughly equal to the size of the `VOC12train-seg` (1464 images) initially used for training the context model. As we increase the data size used for context modeling, we can see how both detection and segmentation improve; however, this gain diminishes as the data size keeps growing. This probably mean that to improve scene understanding, the context model has to get visual context "approximately right" and further improvement is most likely limited by other factors such as unrealistic generated scenes and limited number of instances that are being copy-pasted. On the other hand, if the context model did not receive sufficient amount of data for training, as in the case of using only 5% of the full set, our augmentation strategy tends to the random one and shows little improvement.

4.4 Conclusion

In this chapter, we introduce a data augmentation technique dedicated to object-level scene understanding problems. From a methodological point of view, we show that this approach is effective and goes beyond traditional augmentation methods. One of the keys to obtain significant improvements in terms of accuracy is to introduce an appropriate context model which allows us to automatically find realistic locations for objects to be pasted and blended in a new scenes. While the role of explicit context modeling has been unclear so far for detection and segmentation, we show that it is in fact crucial when performing data augmentation and learn with fewer labeled data, which is one of the major issues deep learning models are facing today.

Chapter 5

Ensemble Methods for Few-Shot Classification

Chapter abstract: Few-shot classification consists of learning a predictive model that is able to effectively adapt to a new class, given only a few annotated samples. To solve this challenging problem, meta-learning has become a popular paradigm that advocates the ability to "learn to adapt". Recent works have shown, however, that simple learning strategies without meta-learning could be competitive. In this chapter, we go a step further and show that by addressing the fundamental high-variance issue of few-shot learning classifiers, it is possible to significantly outperform current meta-learning techniques. Our approach consists of designing an ensemble of deep networks to leverage the variance of the classifiers, and introducing new strategies to encourage the networks to cooperate, while encouraging prediction diversity. Evaluation is conducted on the mini-ImageNet, tiered-ImageNet and CUB datasets, where we show that even a single network obtained by distillation yields state-of-the-art results. See Section 2.5 for related work

Source code implementing the whole pipeline is publicly available [123] (it is given in Appendix B). Appendix C contains additional results and implementation details regarding the presented method. The material of this part is essentially based on the following work.

Nikita Dvornik, Cordelia Schmid, Julien Mairal. Diversity with Cooperation: Ensemble Methods for Few-Shot Classification. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2019.

It is well known by now, that with massively annotated datasets for training, convolutional neural networks can achieve outstanding results in many visual recognition tasks, such as classification [92], object detection [44, 110, 142], or semantic segmentation [44, 111, 144]. The problem is, however, that when such datasets are not available, the models' performance degrades significantly [57]. Annotating a large corpus is expensive and not always feasible, depending on the task at hand. Thus, improving the generalization capabilities of deep neural networks and removing the need for huge sets of annotations is of utmost importance.

While such a grand challenge may be addressed from different complementary points of views, e.g., large-scale unsupervised learning [23], self-supervised learning [39, 60], or by developing regularization techniques dedicated to deep networks [19, 183], we choose in this chapter to focus on variance-reduction principles based on ensemble methods.

Specifically, we are interested in few-shot classification, where a classifier is first trained from scratch on a medium-sized annotated corpus—that is, without leveraging external data or a pre-trained network, and then we evaluate its ability to adapt to new classes, for which only very few annotated samples are provided (typically 1 or 5). Unfortunately, simply fine-tuning a convolutional neural network on a new classification task with very few samples has been shown to provide poor results [52], which has motivated the community to develop dedicated approaches.

The dominant paradigm in few-shot learning builds upon meta-learning [52, 137, 152, 159, 164, 172], which is formulated as a principle to learn how to adapt to new learning problems. These approaches split a large annotated corpus into classification tasks, and the goal is to transfer knowledge across tasks in order to improve generalization. While the meta-learning principle seems appealing for few-shot learning, its empirical benefits have not been clearly established yet. There is indeed strong evidence [27, 59, 135] that training CNNs from scratch using meta-learning performs substantially worse than if CNN features are trained in a standard fashion—that is, by minimizing a classical loss function relying on corpus annotations; on the other hand, learning only the last layer with meta-learning has been found to produce better results [59, 135]. Then, it was recently shown in [27] that simple distance-based classifiers could achieve similar accuracy as meta-learning approaches.

This work goes a step further and shows that meta-learning-free approaches can be improved and significantly outperform the current state of the art in few-shot learning. Our angle of attack consists of using ensemble methods to reduce the variance of few-shot learning classifiers, which is inevitably high given the small number of annotations. Given an initial medium-sized dataset (following the standard setting of few-shot learning), the most basic ensemble approach consists of first training several CNNs independently before freezing them and removing the

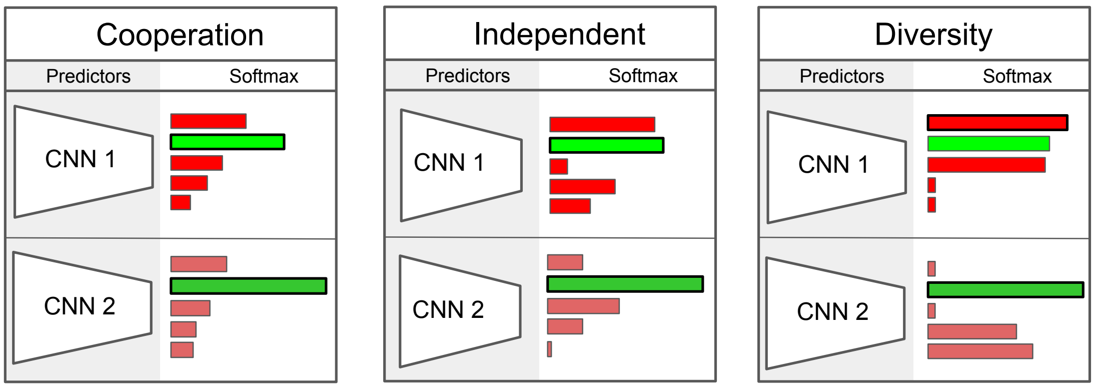


Figure 5.1 – **Illustration of the cooperation and diversity strategies on two networks.** All networks receive the same image as input and compute corresponding class probabilities with softmax. Cooperation encourages the non-ground truth probabilities (in red) to be similar, after normalization, whereas diversity encourages orthogonality.

last prediction layer. Then, given a new class (with few annotated samples), we build a mean centroid classifier for each network and estimate class probabilities—according to a basic probabilistic model—of test samples based on the distance to the centroids [117,159]. The obtained probabilities are then averaged over networks, resulting in higher accuracy.

While we show that the basic ensemble method where networks are trained independently already performs well, we introduce penalty terms that allow the networks to cooperate during training, while encouraging enough diversity of predictions, as illustrated in Figure 5.1. The motivation for cooperation is that of easier learning and regularization, where individual networks from the ensemble can benefit from each other. The motivation for encouraging diversity is classical for ensemble methods [37], where a collection of weak learners making diverse predictions often performs better together than a single strong one. Whereas these two principles seem in contradiction with each other at first sight, we show that both principles are in fact useful and lead to significantly better results than the basic ensemble method. Finally, we also show that a single network trained by distillation [79] to mimic the behavior of the ensemble also performs well, which brings a significant speed-up at test time. In summary, our contributions are three-fold:

— We introduce mechanisms to encourage cooperation and diversity for learning an ensemble of networks. We study these two principles for few-shot learning and characterize the regimes where they are useful.
— We show that it is possible to significantly outperform current state-of-the-art techniques for few-shot classification without using meta-learning.
— As a minor contribution, we also show how to distill an ensemble into a single network with minor loss in accuracy, by using additional unlabeled data.

Outline. The chapter is organized as follows: Section 5.1 presents related work in ensemble methods. Then, Section 5.2 presents our pipeline for few-shot classification and introduces new ways of training ensembles with cooperation and diversity penalties. We perform detailed study of the method in Section 5.3, apply it to few-shot classification problems on *mini*-ImageNet, *tiered*-ImageNet and CUB datasets and compare our solution to existing methods. Finally, Section 5.4 concludes the chapter.

5.1 Related Work

It is well known that ensemble methods reduce the variance of estimators and subsequently may improve the quality of prediction [54]. To gain accuracy from averaging, various randomization or data augmentation techniques are typically used to encourage a high diversity of predictions [20,37]. While individual classifiers of the ensemble may perform poorly, the quality of the average prediction turns out to be sometimes surprisingly high.

Even though ensemble methods are costly at training time for neural networks, it was shown that a single network trained to mimic the behavior of the ensemble could perform almost equally well [79]—a procedure called distillation—thus removing the overhead at test time. To improve the scalability of distillation in the context of highly-parallelized implementations, an online distillation procedure is proposed in [5]. There, each network is encouraged to agree with the averaged predictions made by other networks of the ensemble, which results in more stable models. The objective of our work is however significantly different. The form of cooperation they encourage between networks is indeed targeted to scalability and stability (due to industrial constraints), but online distilled networks do not necessarily perform better than the basic ensemble strategy. Our goal, on the other hand, is to improve the quality of prediction and do better than basic ensembles.

To this end, we encourage cooperation in a different manner, by encouraging predictions between networks to match in terms of class probabilities conditioned on the prediction not being the ground truth label. While we show that such a strategy alone is useful in general when the number of networks is small, encouraging diversity becomes crucial when this number grows. Finally, we show that distillation can help to reduce the computational overhead at test time.

5.2 Ensemble Methods for Few-shot Classification

In this section, we present our approach for few-shot classification, starting with preliminary components.

5.2.1 Mean-centroid classifiers

We now explain how to perform few-shot classification with a fixed feature extractor and a mean centroid classifier.

Few-shot classification with prototype classifier. During the meta-training stage, we are given a dataset D_b with annotations, which we use to train a prediction function f_θ represented by a CNN. Formally, after training the CNN on D_b, we remove the final prediction layer and use the resulting vector $\tilde{f}_\theta(x)$ as a set of visual features for a given image x. The parameters θ represent the weights of the network, which are frozen after this training step.

During meta-testing, we are given a new dataset $D_q = \{x_i, y_i\}_{i=1}^{nk}$, where n is a number of new categories and k is the number of available examples for each class. The (x_i, y_i)'s represent image-label pairs. Then, we build a mean centroid classifier, leading to the class prototypes

$$c_j = \frac{1}{k}\sum_{i=1}^{k} \tilde{f}_\theta(x_i), \qquad j = 1, ..., n. \tag{5.1}$$

Finally, a test sample x is assigned to the nearest centroid's class. Simple mean-centroid classifiers have proven to be effective in the context of few-shot classification [27, 117, 159], which is confirmed in the following experiment.

Motivation for mean-centroid classifier. We report here an experiment showing that a more complex model than (5.1) does not necessarily lead to better results for few-shot learning. Consider indeed a parametrized version of (5.1):

$$c_j = \sum_{i=1}^{nk} \alpha_i^j \tilde{f}_\theta(x_i), \qquad j = 1, ..., n, \tag{5.2}$$

where the weights α_i^j can be learned with gradient descent by maximizing the likelihood of the probabilistic model

$$p_j(y = l|x) = \frac{\exp(-d(\tilde{f}_\theta(x), c_l))}{\sum_{j=1}^{n} \exp(-d(\tilde{f}_\theta(x), c_j)} \tag{5.3}$$

where $d(\cdot,\cdot)$ is a distance function, such as Euclidian distance or negative cosine similarity. Since the coefficients are learned from data and not set arbitrarily to $1/k$ as in (5.1), one would potentially expect this method to produce better classifiers if appropriately regularized. When we run the evaluation of the aforementioned classifiers on 1000 5-shot learning tasks sampled from `miniImagenet-test` (see experimental section for details about this dataset), we get similar results on average: $77.28 \pm 0.46\%$ for (5.1) vs. $77.01 \pm 0.50\%$ for (5.2), confirming that learning meaningful parameters in this very-low-sample regime is difficult.

5.2.2 Learning ensembles of deep networks

During meta-training, one needs to minimize the following loss function over a training set $\{x_i, y_i\}_{i=1}^{m}$:

$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m} \ell(y_i, \sigma(f_\theta(x_i))) + \lambda\|\theta\|_2^2, \tag{5.4}$$

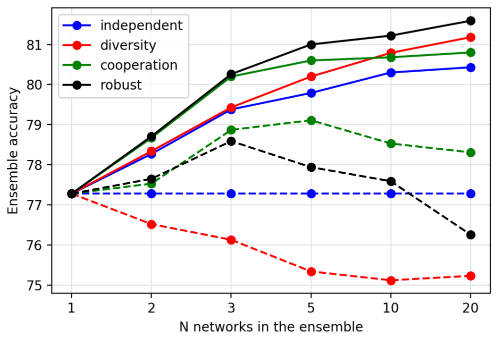
where f_θ is a CNN as before. The cost function $\ell(\cdot,\cdot)$ is the cross-entropy between ground-truth labels and predicted class probabilities $p = \sigma(f_\theta(x))$, where σ is the normalized exponential function, and λ is a weight decay parameter.

When training an ensemble of K networks f_{θ_k} independently, one would solve (5.4) for each network separately. While these terms may look identical, solutions provided by deep neural networks will typically differ when trained with different initializations and random seeds, making ensemble methods appealing in this context.

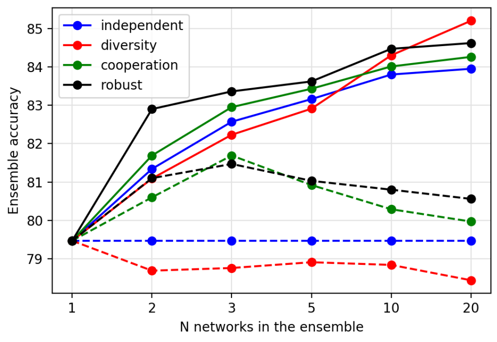
In this chapter, we are interested in ensemble of networks, but we also want to model relationships between its members; this may be achieved by considering a pairwise penalty function ψ, leading to the joint formulation:

$$\begin{aligned} L(\bar{\theta}) = \sum_{j=1}^{K}\left(\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \sigma(f_{\theta_j}(x_i))) + \lambda\|\theta_j\|_2^2\right) \\ + \frac{\gamma}{n(K-1)}\sum_{i=1}^{n}\sum_{\substack{j,l \\ j\neq l}}^{K} \psi(y_i, f_{\theta_j}(x_i), f_{\theta_l}(x_i)), \end{aligned} \tag{5.5}$$

where $\bar{\theta}$ is the vector obtained by concatenating all the parameters θ_j. By carefully designing the function ψ and setting up appropriately the parameter γ, it is possible to achieve desirable properties of the ensemble, such as diversity of predictions or collaboration during training.



(a) *mini*-ImageNet 5-shots



(b) CUB 5-shots

Figure 5.2 – **Accuracies of different ensemble strategies (one for each color) for various numbers of networks.** Solid lines give the ensemble accuracy after aggregating predictions. The average performance of single models from the ensemble is plotted with a dashed line. Best viewed in color.

5.2.3 Encouraging diversity and cooperation

To reduce the high variance of few-shot learning classifiers, we use ensemble methods trained with a particular interaction function ψ, as in (5.5). Then, once the parameters θ_j have been learned during meta-training, classification in meta-testing is performed by considering a collection of K mean-centroid classifiers associated to the basic probabilistic model presented in Eq. (5.3). Given a test image, the K class probabilities are averaged. Such a strategy was found to perform empirically better than a voting scheme.

As we show in the experimental section, the choice of pairwise relationship function ψ significantly influences the quality of the ensemble. Here, we describe three different strategies, which all provide benefits in different regimes, starting by a criterion encouraging diversity of predictions.

Diversity. One way to encourage diversity consists of introducing randomization in the learning procedure, e.g., by using data augmentation [20, 54] or various initializations. Here, we also evaluate the effect of an interaction function ψ that acts directly on the network predictions. Given an image x, two models parametrized by θ_i and θ_j respectively lead to class probabilities $p_i = \sigma(f_{\theta_i}(x))$ and $p_j = \sigma(f_{\theta_j}(x))$. During training, p_i and p_j are naturally encouraged to be close to the assignment vector e_y in $\{0, 1\}^d$ with a single non-zero entry at position y, where y is the class label associated to x and d is the number of classes.

From [79], we know that even though only the largest entry of p_i or p_j is used to make predictions, other entries—typically not corresponding to the ground truth label y—carry important information about the network. It becomes then natural to consider the probabilities $\hat{p}_i$ and $\hat{p}_j$ conditioned on not being the ground truth label. Formally, these are obtained by setting to zero the entry y in p_i and p_j renormalizing the corresponding vectors such that they sum to one. Then, we consider the following diversity penalty

$$\phi(\hat{p}_i, \hat{p}_j) = \cos(\hat{p}_i, \hat{p}_j). \tag{5.6}$$

When combined with the loss function, the resulting formulation encourages the networks to make the right prediction according to the ground-truth label, but then they are also encouraged to make different second-best, third-best, and so on, choice predictions (see Figure 5.1). This penalty turns out to be particularly effective when the number of networks is large, as shown in the experimental section. It typically worsens the performance of individual classifiers on average, but make the ensemble prediction more accurate.

Cooperation. Apparently opposite to the previous principle, encouraging the conditional probabilities $\hat{p}_i$ to be similar—though with a different metric—may also improve the quality of prediction by allowing the networks to cooperate for better learning. Our experiments show that such a principle alone may be effective, but it appears to be mostly useful when the number of training networks is small, which suggests that there is a trade-off between cooperation and diversity that needs to be found.

Specifically, our experiments show that using the negative cosine—in other words, the opposite of (5.6)—is ineffective. However, a penalty such as the symmetrized KL-divergence turned out to provide the desired effect:

$$\phi(\hat{p}_i, \hat{p}_j) = \frac{1}{2}(\mathrm{KL}(\hat{p}_i||\hat{p}_j) + \mathrm{KL}(\hat{p}_j||\hat{p}_i)). \tag{5.7}$$

By using this penalty, we managed to obtain more stable and faster training, resulting in better performing individual networks, but also—perhaps surprisingly—a better ensemble. Since this formulation doesn't match the outputs directly, genuine diversity of the networks still effects positively the final ensemble accuracy. Unfortunately, we also observed that the gain of ensembling diminishes with the number of networks in the ensemble since the individual members become too similar.

Robustness and cooperation. Given experiments conducted with the two previous penalties, a trade-off between cooperation and diversity seems to correspond to two regimes (low vs. high number of networks). This motivated us to develop an approach designed to achieve the best trade-off. When considering the cooperation penalty (5.7), we try to increase diversity of prediction by several additional means. i) We randomly drop some networks from the ensemble at each training iteration, which causes the networks to learn on different data streams and reduces the speed of knowledge propagation. ii) We introduce Dropout within each network to increase randomization. iii) We feed each network with a different (crop, color) transformation of the same image, which makes the ensemble more robust to input image transformations. Overall, this strategy was found to perform best in most scenarios (see Figure 5.2).

5.2.4 Ensemble distillation

As most ensemble methods, our ensemble strategy introduces a significant computional overhead at training time. To remove the overhead at test time, we use a variant of knowledge distillation [79] to compress the ensemble into a single network f_w. Given the meta-training dataset D_b, we consider the following cost function on example (x, y):

$$\ell(x,y) = (1-\alpha)\cdot \hat{\ell}(e_y, \sigma(f_w(x))) \\ - \alpha \cdot T^2 \cdot \hat{\ell}\left(\tfrac{1}{K}\textstyle\sum_{k=1}^{K} \sigma\left(\tfrac{f_{\theta_k}(x)}{T}\right),\ \sigma\left(\tfrac{f_w(x)}{T}\right)\right), \quad (5.8)$$

where, $\hat{\ell}$ is cross-entropy, e_y is a one-hot embedding of the true label y. The second term performs distillation with parameter T (see [79]). It encourages the single model f_w to be similar to the average output of the ensemble. In our experiments, we are able to obtain a model with performance relatively close to that of the ensemble (see Section 5.3).

Modeling out-of-distribution behavior. When distillation is performed on the dataset D_b, the network f_w mimics the behavior of the ensemble on a specific data distribution. However, new categories are introduced at test time. Therefore, we also tried distillation by using additional unnannotated data, which yields slightly better performance.

5.3 Experimental Results

We now present experiments to study the effect of cooperation and diversity for ensemble methods, and start with experimental and implementation details.

5.3.1 Experimental Setup

Datasets. We use *mini*-ImageNet [137] and *tiered*-ImageNet [141] which are derived from the original ImageNet [145] dataset and Caltech-UCSD Birds (CUB) 200-2011 [174]. *Mini*-ImageNet consists of 100 categories—64 for training, 16 for validation and 20 for testing—with 600 images each. *Tiered*-ImageNet is also a subset of ImageNet that includes 351 class for training, 97 for validation and 160 for testing which is 779,165 images in total. The splits are chosen such that the training classes are sufficiently different from the test ones, unlike in *mini*-ImageNet. The CUB dataset consists of 11,788 images of birds of more than 200 species. We adopt train, val, and test splits from [181], which were originally created by randomly splitting all 200 species in 100 for training, 50 for validation, and 50 for testing.

Evaluation. In few-shot classification, the test set is used to sample N 5-way classification problems, where only k examples of each category are provided for training and 15 for evaluation. We follow [52,59,125,135,137] and test our algorithms for $k = 1$ and 5 and N is set to 1 000. Each time, classes and corresponding train/test examples are sampled at random. For all our experiments we report the mean accuracy (in %) over 1 000 tasks and 95% confidence interval.

Implementation details. For all experiments, we use the Adam optimizer [85] with an initial learning rate 10^{-4}, which is decreased by a factor 10 once during training when no improvement in validation accuracy is observed for p consecutive epochs. For *mini*-ImageNet, we use $p = 10$, and 20 for the CUB dataset. When distilling an ensemble into one network, p is doubled. We use random crops and color augmentation during training as well as weight decay with parameter $\lambda = 5 \cdot 10^{-4}$. All experiments are conducted with the ResNet18 architecture [76], which allows us to train our ensembles of 20 networks on a single GPU. Input images are then re-scaled to the size 224×224, and organized in mini-batches of size 16. Validation accuracy is computed by running 5-shot evaluation on the validation set. During the meta-testing stage, we take central crops of size 224×224 from images and feed them to the feature extractor. No other preprocessing is used at test time. When building a mean centroid classifier, the distance d in (5.3) is computed as the negative cosine similarity [159], which is rescaled by a factor 10.
For a fair comparison, we have also evaluated ensembles composed of ResNet18 [76] with input image size 84×84 and WideResNet28 [186] with input size 80×80. All details are reported in Appendix C.2. For reproducibility purposes, our implementation is made available online and can be found in Appendix B.

5-shot

Ensemble type	1	2	3	5	10	20
Independent	77.28 ± 0.46	78.27 ± 0.45	79.38 ± 0.43	80.02 ± 0.43	80.30 ± 0.43	80.57 ± 0.42
Diversity	77.28 ± 0.46	78.34 ± 0.46	79.18 ± 0.43	79.89 ± 0.43	80.82 ± 0.42	81.18 ± 0.42
Cooperation	77.28 ± 0.46	78.67 ± 0.46	80.20 ± 0.42	80.60 ± 0.43	80.72 ± 0.42	80.80 ± 0.42
Robust	77.28 ± 0.46	78.71 ± 0.45	80.26 ± 0.43	81.00 ± 0.42	81.22 ± 0.43	81.59 ± 0.42
Distilled Ensembles						
Robust-*dist*	–	79.44 ± 0.44	79.84 ± 0.44	80.01 ± 0.42	80.25 ± 0.44	80.63 ± 0.42
Robust-*dist*++	–	79.16 ± 0.46	80.00 ± 0.44	80.25 ± 0.42	80.35 ± 0.44	81.19 ± 0.43

1-shot

Ensemble type	1	2	3	5	10	20
Independent	58.71 ± 0.62	60.04 ± 0.60	60.83 ± 0.63	61.34 ± 0.61	61.93 ± 0.61	62.06 ± 0.61
Diversity	58.71 ± 0.63	59.95 ± 0.61	61.27 ± 0.62	61.43 ± 0.61	62.23 ± 0.61	62.47 ± 0.62
Cooperation	58.71 ± 0.62	60.20 ± 0.61	61.46 ± 0.61	61.61 ± 0.61	62.06 ± 0.61	62.12 ± 0.62
Robust	58.71 ± 0.62	60.91 ± 0.62	62.36 ± 0.60	62.70 ± 0.61	62.97 ± 0.62	63.95 ± 0.61
Distilled Ensembles						
Robust-*dist*	–	62.33 ± 0.62	62.64 ± 0.60	63.14 ± 0.61	63.01 ± 0.62	63.06 ± 0.61
Robust-*dist* ++	–	62.07 ± 0.62	62.81 ± 0.60	63.39 ± 0.61	63.20 ± 0.62	63.73 ± 0.62

Table 5.1 – **Few-shot classification accuracy on *mini*-ImageNet.** The first column gives the strategy, the top row indicates the number N of networks in an ensemble. Here, *dist* means that an ensemble was distilled into a single network, and '++' indicates that extra unannotated images were used for distillation. We performed 1 000 independent experiments on *mini*-ImageNet-test and report the average with 95% confidence interval. All networks are trained on *mini*-ImageNet-train set.

5.3.2 Ensembles for Few-Shot Classification

In this section, we study the effect of ensemble training with pairwise interaction terms that encourage cooperation or diversity. For that purpose, we analyze the link between the size of ensembles and their 1- and 5-shot classification performance on the *mini*-ImageNet and CUB datasets.

Details about the three strategies. When models are trained jointly, the data stream is shared across all networks and weight updates happen simultaneously. This is achieved by placing all models on the same GPU and optimizing the loss (5.5). When training a diverse ensemble, we use the cosine function (5.6) and selected the parameter $\gamma = 1$ that performed best on the validation set among the tested values (10^i, for $i = -2, \ldots, 2$) for $n = 5$ and $n = 10$ networks. Then, this value was kept for other values of n. To enforce cooperation between networks, we use the symmetrized KL function (5.7) and selected the parameter $\gamma = 10$ in the same manner. Finally, the robust ensemble strategy is trained with the

5-shot

Full Ensemble	1	2	3	5	10	20
Independent	79.47 ± 0.49	81.34 ± 0.46	82.57 ± 0.46	83.16 ± 0.45	83.80 ± 0.45	83.95 ± 0.46
Diversity	79.47 ± 0.49	81.09 ± 0.45	82.23 ± 0.46	82.91 ± 0.46	84.30 ± 0.44	85.20 ± 0.43
Cooperation	79.47 ± 0.49	81.69 ± 0.46	82.95 ± 0.47	83.43 ± 0.47	84.01 ± 0.44	84.26 ± 0.44
Robust	79.47 ± 0.49	82.90 ± 0.46	83.36 ± 0.46	83.62 ± 0.45	84.47 ± 0.46	84.62 ± 0.44
Distilled Ensembles						
Robust-*dist*	–	82.72 ± 0.47	82.95 ± 0.46	83.27 ± 0.46	83.61 ± 0.46	83.57 ± 0.45
Robust-*dist*++	–	82.53 ± 0.48	83.04 ± 0.45	83.37 ± 0.46	83.22 ± 0.46	83.21 ± 0.44

1-shot

Ensemble type	1	2	3	5	10	20
Independent	64.25 ± 0.73	66.60 ± 0.72	67.64 ± 0.71	68.07 ± 0.70	68.93 ± 0.70	69.64 ± 0.69
Diversity	64.25 ± 0.73	65.99 ± 0.71	66.71 ± 0.72	68.19 ± 0.71	69.35 ± 0.70	70.07 ± 0.70
Cooperation	64.25 ± 0.73	67.21 ± 0.71	67.93 ± 0.70	68.22 ± 0.70	68.69 ± 0.70	68.80 ± 0.68
Robust	64.25 ± 0.73	67.33 ± 0.71	68.01 ± 0.72	68.53 ± 0.70	68.59 ± 0.70	69.47 ± 0.69
Distilled Ensembles						
Robust-*dist*	–	67.47 ± 0.71	67.29 ± 0.72	68.09 ± 0.70	68.71 ± 0.71	68.77 ± 0.71
Robust-*dist*++	–	67.01 ± 0.74	67.62 ± 0.72	68.68 ± 0.71	68.38 ± 0.70	68.68 ± 0.69

Table 5.2 – **Few-shot classification accuracy on CUB.** The first column gives the type of ensemble and the top row indicates the number of networks in an ensemble. Here, *dist* means that an ensemble was distilled into a single network, and '++' indicates that extra unannotated images were used for distillation. We performed 1000 independent experiments on CUB-test and report the average with 95% confidence interval. All networks are trained on CUB-train set.

cooperation relationship penalty and the same parameter γ, but we use Dropout with probability 0.1 before the last layer; each of the network is dropped from the ensemble with probability 0.2 at every iteration; different networks receive different transformation of the same image, i.e., different random crops and color augmentation.

Results. Tables 5.1, 5.2 summarize the few-shot classification accuracy of ensembles trained with our strategies and compare with basic ensembles. On the *mini*-ImageNet dataset, the results for 1- and 5-shot classification are consistent with each other. Training with cooperation allows smaller ensembles ($n \leq 5$) to perform better, which leads to higher individual accuracy of the ensemble members, as seen in Figure 5.2. However, when $n \geq 10$, cooperation is less effective, as opposed to the diversity strategy, which benefits from larger n. As we can see from Figure 5.2, individual members of the ensemble become worse, but the ensemble accuracy improves substantially. Finally, the robust strategy seems to perform best for all values of n in almost all settings. The situation for the CUB dataset is similar, although we notice that robust ensembles perform similarly as the diversity

Method	Input size	Network	5-shot	1-shot
TADAM [125]	84	ResNet	76.70 ± 0.30	58.50 ± 0.30
Cosine + Attention [59]	224	ResNet	73.00 ± 0.64	56.20 ± 0.86
Linear Classifier [27]	224	ResNet	74.27 ± 0.63	51.75 ± 0.80
Cosine Classifier [27]	224	ResNet	75.68 ± 0.63	51.87 ± 0.77
PPA [135]	80	WideResNet	73.74 ± 0.19	59.60 ± 0.41
LEO [146]	80	WideResNet	77.59 ± 0.12	61.76 ± 0.08
FEAT [181]	80	WideResNet	78.32 ± 0.16	61.72 ± 0.11
Robust 20-*dist*++ (ours)	224	ResNet	**81.19** ± 0.43	**63.73** ± 0.62
Robust 20-*dist*++ (ours)	84	ResNet	75.62 ± 0.48	59.48 ± 0.62
Robust 20-*dist*++ (ours)	80	WideResNet	81.17 ± 0.43	63.28 ± 0.62
Robust 20 Full (ours)	224	ResNet	81.59 ± 0.42	63.95 ± 0.61
Robust 20 Full (ours)	84	ResNet	76.90 ± 0.42	59.38 ± 0.65
Robust 20 Full (ours)	80	WideResNet	81.94 ± 0.44	63.46 ± 0.62

Table 5.3 – **Comparison of distilled ensembles to other methods on 1- and 5-shot *mini*ImageNet.** The two last columns display the accuracy on 1- and 5-shot learning tasks. To evaluate our methods we performed 1 000 independent experiments on *Mini*ImageNet-test and report the average and 95% confidence interval. Here, '++' means that extra non-annotated images were used to perform distillation. The last model is a full ensemble and should not be directly compared to the rest of the table.

strategy for $n = 20$.

5.3.3 Distilling an ensemble

We distill robust ensembles of all sizes to study knowledge transferability with growing ensemble size. To do so, we use the meta-training dataset and optimize the loss (5.8) with parameters $T = 10$ and $\alpha = 0.8$. For the strategy using external data, we randomly add at each iteration 8 images (without annotations) from the COCO [108] dataset to the 16 annotated samples from the meta-training data. Those images contribute only to the distillation part of the loss (5.8). Table 5.1 and 5.1 display model accuracies for *mini*-ImageNet and CUB datasets respectively. For 5-shot classification on *mini*-ImageNet, the difference between ensemble and its distilled version is rather low (around 1%), while adding extra non-annotated data helps reducing this gap. Surprisingly, 1-shot classification accuracy is slightly higher for distilled models than for their corresponding full ensembles. On the CUB dataset, distilled models stop improving after $n = 5$, even though the performance of full ensembles keeps growing. This seems to indicate

Method	Input size	Network	5-shot	1-shot
TADAM [125]	84	ResNet	81.92 $\pm$ 0.30	62.13 $\pm$ 0.31
LEO [146]	80	WideResNet	81.44 $\pm$ 0.12	66.33 $\pm$ 0.09
Mean Centroid (one network)	224	ResNet	83.89 $\pm$ 0.33	68.33 $\pm$ 0.32
Robust 20-*dist* (ours)	224	ResNet	**85.43** $\pm$ 0.21	**70.44** $\pm$ 0.32
Robust 20 Full	224	ResNet	86.49 $\pm$ 0.22	71.71 $\pm$ 0.31

Table 5.4 – **Comparison of distilled ensembles to other methods on 1- and 5-shot *tiered*-ImageNet [141].** To evaluate our methods we performed 5 000 independent experiments on *tiered*-ImageNet-test and report the average accuracy with 95% confidence interval.

that the capacity of the single network may have been reached, which suggests using a more complex architecture here. Consistently with such hypothesis, adding extra data is not as helpful as for *mini*-ImageNet, most likely because data distributions of COCO and CUB are more different.

In Tables 5.3, 5.4, we also compare the performance of our distilled networks with other baselines from the literature, including current state-of-the-art meta-learning approaches, showing that our approach does significantly better on the *mini*-ImageNet [137] and *tiered*-ImageNet [141] datasets.

5.3.4 Study of relationship penalties

There are many possible ways to model relationship between the members of an ensemble. In this subsection, we study and discuss such particular choices.

Input to relationship function. As noted by [79], class probabilities obtained by the softmax layer of a network seem to carry a lot of information and are useful for distillation. However, after meta-training, such probabilities are often close to binary vectors with a dominant value associated to the ground-truth label. To make small values more noticeable, distillation uses a parameter T, as in (5.8). Given such a class probability computed by a network, we experimented such a strategy consisting of introducing new probabilities $\hat{p} = \sigma(p/T)$, where the contributions of non ground-truth values are emphasized. When used within our diversity (5.6) or cooperation (5.7) penalties, we however did not see any improvement over the basic ensemble method. Instead, we found that computing the class probabilities conditioned on not being the ground truth label, as explained in Section 5.2.3, would perform much better.

This is illustrated on the following experiment with two network ensembles of size $n = 5$, where we compared the two strategies. We enforce similarity on

Purpose (Sign)	L2	-cos	KL_{sim}
Cooperation (+)	80.14 $\pm$ 0.43	80.29 $\pm$ 0.44	80.72 $\pm$ 0.42
Diversity (-)	80.54 $\pm$ 0.44	80.82 $\pm$ 0.42	79.81 $\pm$ 0.43

Table 5.5 – **Evaluating different relationship criteria on *mini*-Imagenet 5-shot** The first row indicates which function was used as a relationship criteria, the first column indicates for which purpose the function is used and the corresponding sign. To evaluate our methods, we performed 1 000 independent experiments on CUB-test and report the average accuracy with 95% confidence intervals. All ensembles are trained on *mini*-ImageNet-train.

the full probability vectors in the first one, computed with softmax at $T = 10$ following [5], and with conditionally non-ground-truth probabilities for the second one as defined in Section 5.2.3. When using the cooperation training formulation, the second strategy turns out to perform about 1% better than the first one (79.79 % vs 80.60%), when tested on *Mini*ImageNet. Similar observations have been made using the diversity criterion. In comparison, the basic ensemble method without interactions achieves about 80%.

Choice of relationship function. In principle, any similarity measure could be used to design a penalty encouraging cooperation. Here, we show that in fact, selecting the right criterion for comparing probability vectors (cosine similarity, L2 distance, symmetrized KL divergence), is crucial depending on the desired effect (cooperation or diversity). In Table 5.5, we perform such a comparison for an ensemble with $n = 5$ networks on the *Mini*ImageNet dataset for a 5−shot classification task, when plugging the above function in the formulation 5.5, with a specific sign. The parameter γ for each experiment is chosen such that the performance on the validation set is maximized.

When looking for diversity, the cosine similarity performs slightly better than negative L2 distance, although the accuracies are within error bars. Using negative KL_{sim} with various γ was either not distinguishable from independent training or was hurting the performance for larger values of γ (not reported on the table). As for cooperation, positive KL_{sim} gives better results than L2 distance or negative cosine similarity. We believe that this behavior is due to important difference in the way these functions compare small values in probability vectors. While negative cosine or L2 losses would penalize heavily the largest difference, KL_{sim} concentrates on values that are close to 0 in one vector and are greater in the second one.

Method	*mini*-ImageNet → CUB
MatchingNet [172]	53.07 ± 0.74
MAML [52]	51.34 ± 0.72
ProtoNet [159]	62.02 ± 0.70
Linear Classifier [27]	**65.57** ± 0.70
Cosine Classifier [27]	62.04 ± 0.76
Robust 20-*dist*++ (ours)	64.23 ± 0.58
Robust 20 Full (ours)	65.04 ± 0.57
Diverse 20 Full (ours)	66.17 ± 0.55

Table 5.6 – **5-shot classification accuracy under domain shift.** The last two models are full ensembles and should not be directly compared with the rest of the table. We performed 1 000 independent experiments on `CUB-test` from [27] and report the average and confidence interval here. All ensembles are trained on *mini*-ImageNet.

5.3.5 Performance under domain shift

Finally, we evaluate the performance of ensemble methods under domain shift. We proceed by meta-training the models on the *mini*-ImageNet training set and evaluate the model on the CUB-test set. The following setting was first proposed by [27] and aims at evaluating the performance of algorithms to adapt when the difference between training and testing distributions is large. To compare to the results reported in the original work, we adopt their CUB test split. Table 5.6 compares our results to the ones in [27]. We can see that neither the full robust ensemble nor its distilled version are able to do better than training a linear classifier on top of a frozen network. Yet, it does significantly better than distance-based approaches (denoted by cosine classifier in the table). However, if a diverse ensemble is used, it achieves the best accuracy. This is not surprising and highlights the importance of having diverse models when ensembling weak classifiers.

5.4 Conclusion

In this chapter, we show that distance-based classifiers for few-shot learning suffer from high variance, which can be significantly reduced by using an ensemble of classifiers. Unlike traditional ensembling paradigms where diversity of predictions is encouraged by various randomization and data augmentation techniques, we show that encouraging the networks to cooperate during training is also important.

The overall performance of a single network obtained by distillation (with no computational overhead at test time) leads to state-of-the-art performance for few shot learning, without relying on the meta-learning paradigm. While such a result may sound negative for meta-learning approaches, it may simply mean that further research in this area is required to improve upon classical learning methods in few-shot image recognition.

Chapter 6

Conclusion

In this thesis we focused on problems with scarce annotations, such as object detection, instance and semantic segmentation as well as few-shot learning. Our goal was to use annotated data in a more effective way by discovering connections between tasks and their annotations, explicitly modeling context to perform data augmentation or reducing the variance of classifiers with small data support.

Outline. The chapter is organized as follows: In Section 6.1 we summarize the contributions of the thesis. Section 6.2 gives directions for future research inspired by the work presented in the manuscript.

6.1 Summary of Contributions

In the current section we list all 3 contribution as they are presented in the thesis.

Joint object detection and semantic segmentation with BlitzNet.
In Chapter 3, we address the problem of object detection and semantic segmentation on datasets annotated for both tasks. We notice connections between the tasks and their annotations that allows us to design a unified framework solving both problems together. The main pipeline consists of a fully-convolutional neural network, that given an image predicts object bounding boxes and semantic segmentation masks. The system is trained jointly on both tasks, which allows it to benefit from presence of two types of annotations, i.e., on the region level (bounding boxes) and on the pixel level (segmentation masks). Such design principles lead to a system that:

1. Improves performance for object detection and semantic segmentation thanks to joint training and leads to state-of-the-art results.
2. Solves the two problems in real time thanks to computations shared between the tasks and efficient design.
3. Unifies the tasks of object detection and semantic segmentation in a universal pipeline that learns to combine local and global image information to improve accuracy on small objects.

The model works well for object detection and semantic segmentation. However, it does not support instance masks prediction by default as it requires higher model capacity and different system design. Other method limitations include still limited robustness to object scale, handling complicated scenes with many objects and false detections caused by ambiguous object context.

Context-driven data augmentation for scene understanding.
The contribution of Chapter 4 consists of designing a new data augmentation strategy for scene understanding problems on object level, such as object detection, semantic and instance segmentation. We increase the number of scenes within a dataset by copy-pasting objects from one image to another, using instance segmentation masks. The key observation was that an object has to be placed in the right visual context, otherwise it negatively affects the task's accuracy. The context was modeled explicitly by a CNN; it takes as input visual neighborhood around a given image region and predict likelihood for different object categories to be present at this location. Pasting objects at locations proposed by the context model results in an augmentation pipeline that:

1. Synthesizes new plausible images for object-level scene understanding tasks and allows to effectively increase the size of initial dataset.
2. Improves performance on the tasks of semantic and instance segmentation and object detection, especially when annotations are scarce.
3. Allows to improve accuracy even on large-scale datasets with state-of-the-art recognition systems for object detection and instance segmentation.

Moreover, by using weakly-supervised learning, we could relax the need for pixel-wise annotations of instances to only bounding boxes to perform copy-pasting. However, image-level annotations are too fuzzy and the need for object boxes remains unsolved. Even though the augmentation procedure is automatic, it takes time to train the context model and to predict locations suitable for pasting. In fact, this step becomes a computational bottle-neck when augmenting datasets with large images, because the amount of computations here grows quadratically with input image resolution.

Ensemble methods for few-shot learning
The contribution of Chapter 5 consists of introducing a new approach for few-shot learning. Our method was the first to address high variance of classifiers that inevitably arises in few-shot tasks. We proposed to train a diverse set of feature extractors, build mean centroid classifiers on top and ensemble their predictions. To speed up the inference, we distilled an ensemble into a single network with minor accuracy losses. This distilled classifier allowed us to achieve state-of-the-art accuracy on all standard benchmarks. To make ensembles more accurate and robust, we designed a joint training strategy that promotes cooperation between network during the training stage. This mechanism implements:

1. Sharing networks' knowledge during training by penalizing difference between softmax activations. This allows the networks to improve each others predictions when training.
2. Robustness to small image transformations emerging when matching softmax activations across different transformations of one image. That further improves accuracy of individual ensemble member, while increasing output diversity between the networks.
3. Simplified ensemble distillation due to more aligned network's activations. Thanks to this property, the gap in test accuracy between the full and distilled models is minimized.

Even though the method is accurate, it is agnostic to a change in training and testing data statistic, and hence is ignoring an important source of information. Implementing some form of feature adaptation may further improve the pipeline.

6.2 Future research and perspectives

In this section we give possible directions for future works, based on the contributions of this manuscripts.

Methods for Scene Understanding

BlitzNet with instance segmentation. The BlitzNet pipeline, introduced in Chapter 3, is designed to solve object detection and instance segmentation simultaneously. As it learns jointly on these two tasks, it benefits from different annotation types. Including an extra source of information, namely instance masks, should positively influence the system's performance. Unfortunately, the model is not designed to do instance segmentation. Following the detection-based paradigm for instance segmentation, as in [82], one would naturally extend BlitzNet to predict instance masks from object-specific vectors, currently used to predict box offsets and classification scores. Such design has two obvious flaws: high dimentionality of object vectors necessary to store information about the whole object mask, and the complexity of learnable upsampling, needed to convert this vector into a mask. To address these problems, one may leverage system's multitasking and enforce consistency between semantic and instance segmentation, which will regularize the training as well. This may be done by forcing the union of all individual instance segmentations projected on the image plane to match the mask, predicted by the semantic segmentation head. This is similar in spirit to existing segmentation-based methods for instance segmentation [7, 9, 90]. However, in its core, it is still a detection-based framework that enjoys better accuracy. Such strategy offers a more explicit way to link semantic and instance segmentation problems and their annotations, and as a result may simplify the two tasks. In a similar fashion, it is possible to introduce additional constraints on the neighboring predicted masks; they must match at intersecting locations, when projected onto the image plane. Doing so will teach the network about the output and task structure and should be helpful for reducing the model's complexity.

Adaptive proposal greed for one-stage detectors. Most popular one-stage object detectors follow the training procedure proposed by SSD [110], detailed in Section 2.3. All of them predict class categories and regress box offsets with respect to a grid of diverse bounding boxes, fixed prior to training. Scale and aspect ratio parameters of those boxes are chosen manually, however their optimal value is unknown and generally depends on a dataset. Benefits of adapting the initial proposal set to data were clearly demostrated in [139, 140]. The authors used clustering on location and shape of ground-truth training boxes, to select proposal box configurations maximizing foreground object coverage. However, this step is agnostic to the training procedure of the detector itself. Instead, we think that one may let the initial tiling parameters be variable and optimize them with back-propagation. Doing so must result in optimal tiling parameters and improve precision/recall of the system.

Data Augmentation for Scene Understanding

Instance-specific data augmentation with context guidance. The context model introduced in Chapter 4 predicts the probability of presence of *any* object of a *specific* category, given visual context around a pasting location. In this formulation, all objects of the same category are equally likely to be pasted onto a chosen location. However, intuitively, this assumption is too restrictive, as some instances may fit a scene better than others, due to more suitable scale, appearance, style, or other factors. An example of such scenario is given in Figure 6.1. To alleviate this shortcoming, one may assess the quality of generated images by using another model, that given an image, predicts photo-realism score. For example, it

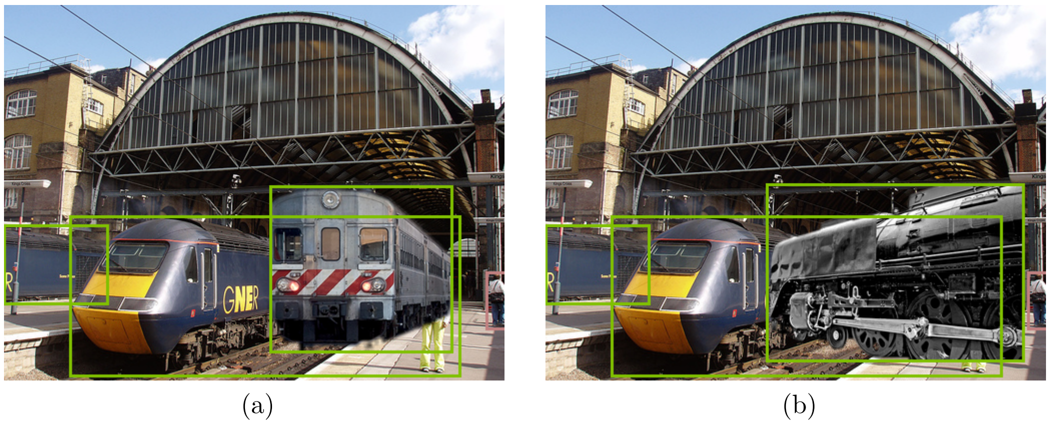


Figure 6.1 – Two images, augmented by pasting a different instance on the same location, proposed by the context model. Image (a) looks more visually convincing than (b) due to a better fit of pasted instance. This motivates the need for automatic selection of more successful image augmentations.

is possible to use a discriminator of a Generative Adversarial Network (GAN) [64] that models the dataset's distributions. Although, it is not clear if this strategy will produce desirable results. Otherwise, one can train such classifier explicitly, using artificial data. Positive examples would be composed by pasting slightly transformed instances back at their location, while negative examples would be generated at random. Such model may capture desired photo-realism properties and score instances that suit a given scene higher. If such photo-realism model is available, one may generate several images using the context-driven data augmentation, score all of them with the photo-realism model, and select the top-scored one for training a scene-understanding system.

Generative context model. The context model proposed in Sectino 4 is a discriminative model operating on image regions, i.e., given an object's location and visual content of its neighborhood it outputs how likely this location is to contain an object. As we typically evaluate hundreds of such locations with a CNN to find good pasting candidates, this step becomes a computational bottleneck. Instead, using a conditional generative model that proposes such locations, given an input image, is conceptually more appealing and beneficial in terms of computational complexity. Such model could be implemented by using a conditional GAN, that given an image, generates rectangles to be cut out off the image, in order to hide an object of a specific class. If such system is implemented, it will allow context-driven data augmentation to work in real-time, without the need to pre-compute pasting locations off-line.

Data augmentation by inpainting with new instances. Another way of introducing new instances in an image, instead of copy-pasting the existing ones, is to train a generative model like GAN [64] to generate new objects. It is possible to learn the distribution of objects appearance, conditioned on their class, and then use it to sample new examples in a controlled way. Moreover, recent progress in generative modeling [109] makes it possible to do realistic image inpainting, where an arbitrary part of an image is deleted or masked-out, and later restored in a way that is natural and coherent with the rest of the image. Hence, combining these two tasks into one generative framework for class-conditioned object inpainting seems promising. For example, when having a confident box for placing an object of category c inside (for example, proposed by the context model), one can inpaint its interior with the proposed generative model, by conditioning on the category c and the image itself. Importantly, in this case we not only condition on the class of an object that has to appear in the box but on its surroundings as well, though only implicitly.

Alternatively, it is possible to find a middle ground between copy-pasting instances on images and inpainting empty boxes with imaginary objects. It amounts to using existing objects (copy) and blending them inside a scene using a generative model (paste). This may be implemented by taking a GAN proposed earlier in this section and additionally conditioning the generator on instance's appearance. This will force the model to inpaint an empty box with the given real object, although adapted to fit the visual surrounding better. A similar approach was implemented in [176] and produced visually appealing results, although it solves a simpler task, namely pure object blending, with no inpainting. Since generative models are known to produce more realistic samples when learning conditional distributions, compared to marginal ones [148], there is hope that produced samples will be of sufficient quality to improve scene understanding performance when used as a data augmentation technique.

Methods for Few-shot Learning

Data augmentation for few-shot learning. A problem central to few-shot learning is the difference between training and testing set statistics, as highlighted in Section 2.5. This means that even if we have properly trained a network (did not overfit training images), say, for dog classification, obtained CNN will probably extract little useful features from images of birds. Hence, we can say that overfitting happened on the level of categories, or features. An increasingly popular [27, 59, 165] approach to this problem is to train a more general feature extractor on training categories, such that obtained features are more useful for new classes. Applying data augmentation to regularize the training is an option, but, unfortunately, most augmentation methods are designed to fight overfitting on image level; they preserve discriminative object's features, while only modifying image's appearance [80, 92, 187]. Fortunately, as supported by the results in Section 4, a data augmentation strategy leveraging prior knowledge about the task may be more useful. To this end, a sensible data augmentation method is to corrupt most discriminative object parts in training images. Doing so will force the network to pay more attention to other object features, that must be used for reasoning, and to extract more distinct features as a consequence. The overall idea could be summarized as using data augmentation to diversifying CNN's filter bank, which should be beneficial for solving a set of problems unknown beforehand.

Feature adaptation for ensemble methods. To tackle few-shot problems, in Chapter 5, we propose to learn a rather general feature extractor that reduces the variance of few-shot classifiers. This alone provided state-of-the-art accuracy on several few-shot learning benchmarks. However, the method is completely agnostic to the domain shift between training and testing sets. There is a considerable body of literature in few-shot classification that adress this problem by feature adaptation [18, 125, 146]. This is done by either adjusting the weights of a CNN feature extractor, based on observed samples of new categories [18, 125], or by first extracting image representations with a CNN, and then mapping them to another, task-specific space [146]. In fact, the benefits of feature adaptation are orthogonal to our contributions in Chapter 5. Thus, it should be possible to further improve our low-variance image embeddings by adjusting them to new classes, which may lead to higher accuracy on few-shot learning problems.

Learning on large datasets without annotations. The main motivation of this manuscript to address effective learning methods on limited datasets was poor performance of unsupervised approaches, even when trained on much bigger corpuses. However, learning good quality image representations with no annotations at all is of foremost importance, as it will allow to leverage billions of images freely available on the web. Hence, developing methods for learning without human supervision is essential for further development. One way to do unsupervised learning is to use an encoder-decoder neural network and train it with a reconstruction loss [86, 171]. Another possibility is to use self-supervised learning, where the structure of input is used to produce surrogate labels for learning feature representations [40, 60, 129]. Clustering, a classical unsupervised learning approach, has been recently turned into a way to learn image representations too [23]. Even though conceptually different, when used for feature extraction, all these methods produce similar results on image classification, and are far behind fully supervised methods. A common point between these paradigms, is that they all use hand-crafted objectives for unsupervised learning, that are only a rough proxy for downstream performance, for example on classification. Thus, using a "smarter" objective may significantly improve unsupervised representation learning.

One way to learn a smarter learning procedure is to use meta-learning [152, 164].

Even though not very successful for few-shot image classification, this paradigm offers new tools for learning prior knowledge [57] or neural network update rules [4], that is beyond the scope of standard learning tools. Importantly, such methods incorporate two nested levels of optimization and allow to take into account two different objectives, one of which could be a desired down-stream task [114]. Hence, it should be possible to use annotated data to learn an unsupervised learning algorithm, that will achieve good performance when tested against the ground-truth labels. The obtained learning algorithm can then be used to learn feature representations from unlabeled data corpuses. Importantly, such learning schemes must be transferable across datasets to be useful, thus learning them must be addressed carefully. With this approach, the new unsupervised learning rules must outperform their hand-crafted counterparts and may open up new perspective for learning algorithms in general.

Appendix A

Publications

This thesis has led to several publications summarized below.

Publications in International Conferences

— Nikita Dvornik[1], Konstantin Shmelkov[1], Julien Mairal, Cordelia Schmid.
BlitzNet: A Real-Time Deep Network for Scene Understanding.
Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017. [44], `https://hal.archives-ouvertes.fr/hal-01573361/document`

— Nikita Dvornik, Julien Mairal, Cordelia Schmid.
Modeling Visual Context is Key to Augmenting Object Detection Datasets.
Proceedings of the IEEE European Conference on Computer Vision (ECCV), 2018. [41], `https://hal.archives-ouvertes.fr/hal-01844474/document`

— Nikita Dvornik, Cordelia Schmid, Julien Mairal.
Diversity with Cooperation: Ensemble Methods for Few-Shot Classification.
Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2019. [42], `https://hal.archives-ouvertes.fr/hal-02080004`

Journal Publications - under minor revision

— Nikita Dvornik, Julien Mairal, Cordelia Schmid.
On the Importance of Visual Context for Data Augmentation in Scene Understanding.
IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI) - submitted in September, 2018, [43], `https://hal.archives-ouvertes.fr/hal-01869784v2`

1. Equal contribution

Appendix B

Software

Several source code bases created during this thesis are available online.

— Source code of BlitzNet:
https://github.com/dvornikita/blitznet/
This code allows to train and evaluate BlitzNet on benchmark datasets such as PASCAL VOC [47] and COCO [108], otherwise the documentation provides the instructions on how to adapt the system to a custom dataset. To improve reproducibility, the repository includes a fast-to-run demo, models already pre-trained on the benchmark datasets, and a script to measure inference speed. Additionally, the repository provides an interactive interface. It allows to easily run the pipeline locally with images stored on a hard-drive or from the web, by providing an image link. The BlitzNet engine was implemented using TensorFlow software package [2], while the interface is built with Tk.

— Source code of the context-driven data augmentation pipeline:
https://github.com/dvornikita/context_aug
This repository provides two main tools necessary for data augmentation. First, it implements the CNN context model with the training and inference routines. We also released the weights of already trained context models. The second part implements the data augmentation pipeline, that given outputs of the context model, selects and blends new objects in the image. The scripts to extract all labeled instances from a dataset and to perform matching of instances with candidate locations are provided. The project is implemented using TensorFlow [2].

— Source code of the ensemble methods for few-shot learning:
`https://github.com/dvornikita/fewshot_ensemble`
This code implements the entire procedure to perform few-shot learning with ensemble methods. First, it allows to train a single or ensemble model for standard classification on a meta-training set. One can train ensembles independently or with proposed penalties of cooperation, diversity or robustness. The code for building mean-centroid classifiers and evaluating few-shot classification accuracy on new categories is provided as well. The method was implemented using PyTorch [128] library.

Appendix C

Additional Results and Details

C.1 Additional results for BlitzNet

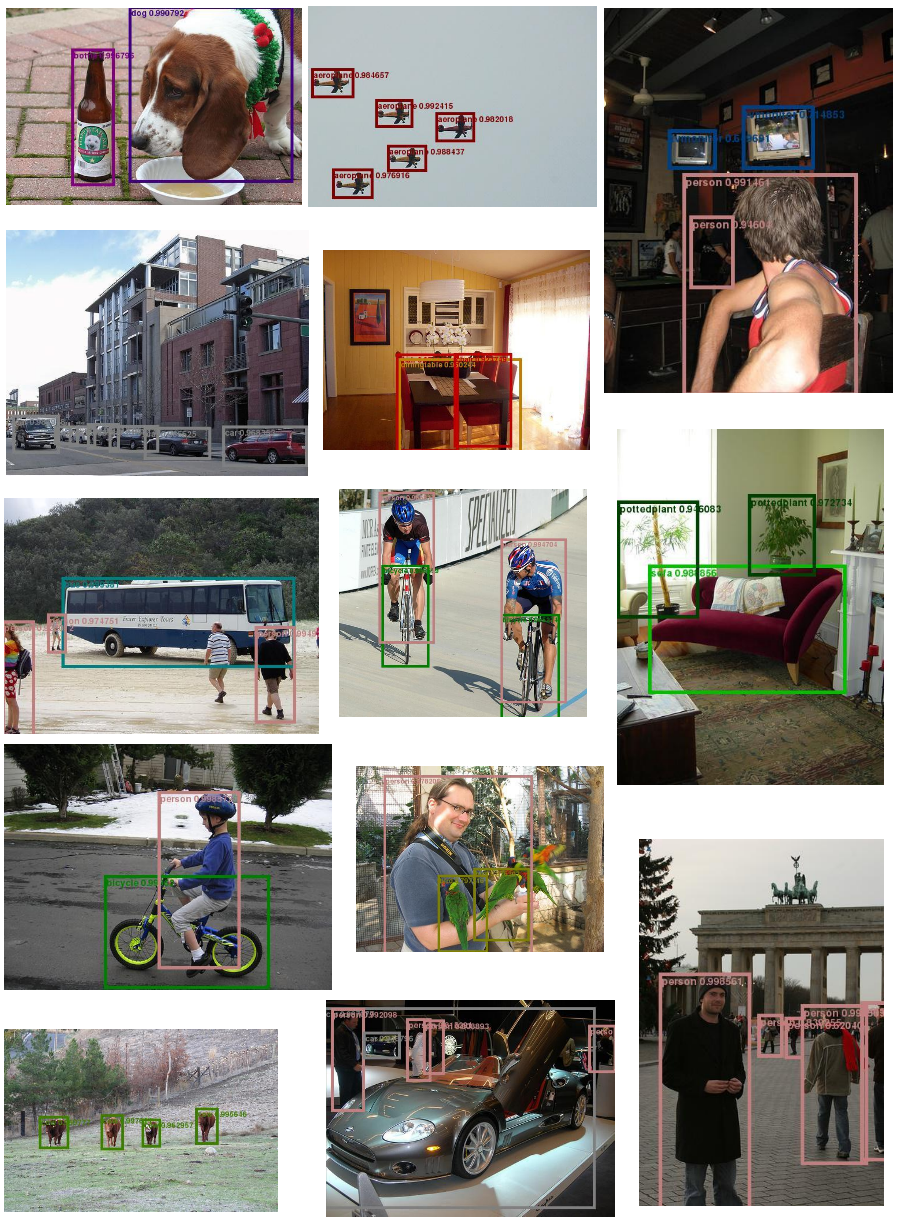


Figure C.1 – **Qualitative results for the taks of object detection.** The results are obtained by the BlitzNet512 trained on VOC07 and VOC12 train-val augmented with extra segmentation masks.

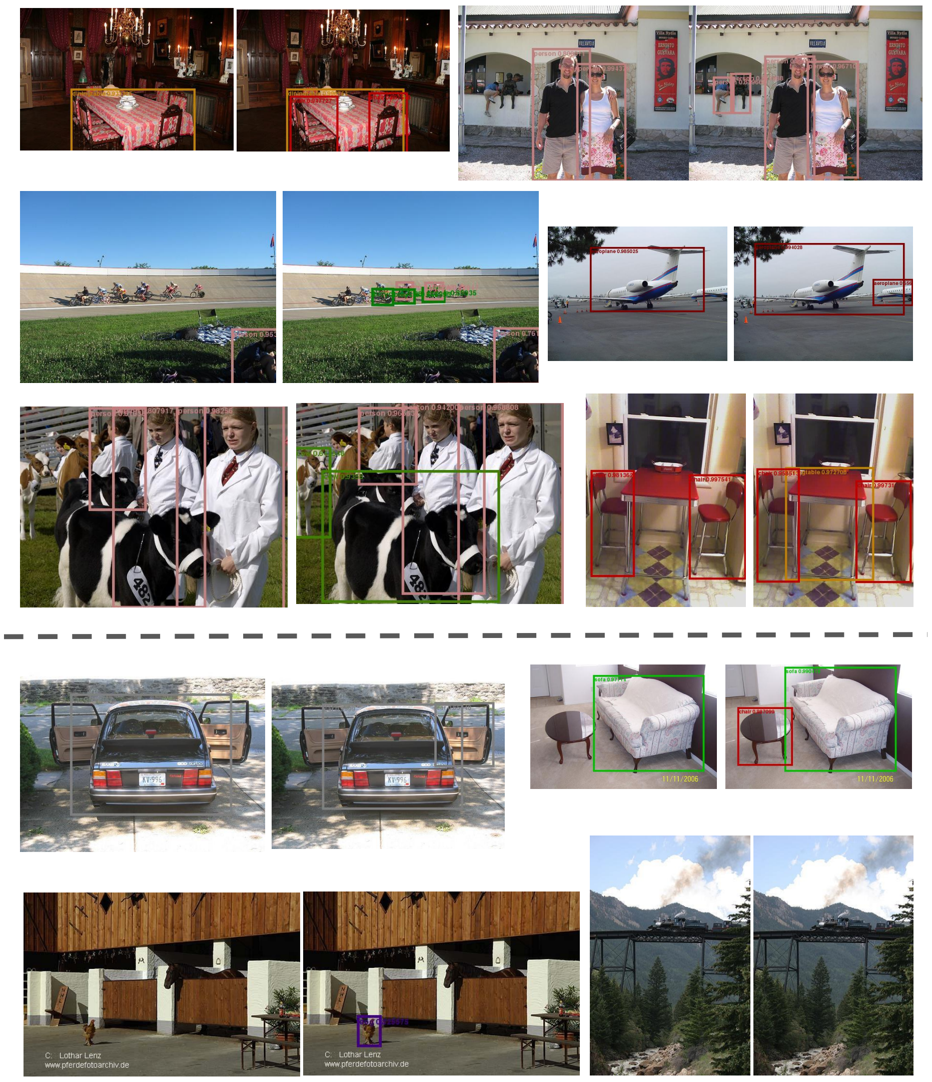


Figure C.2 – **Improved and failure cases of detection by BlitzNet300 comparing to SSD300.** Each pair of images corresponds to the results of detection by SSD300 (left) and BlitzNet300 (right). The cases of improved detection are presened on the top part of the figure and the cases where both methods still fail are placed below the dashed line. It's clear that our pipeline provides more accurate detections in presence of small objects, complicated scenes and objects consisting of several parts with different appearance. The failure cases indicate that modern pipelines still struggle to handle ambiguous big objects (top left), intraclass variability (top right), misleading context (bottom right) and highly occluded objects (bottom left)

C.2 Implementation Details of Network Ensembles

In this section, we elaborate on training, testing and distillation details of the ensemble methods Proposed in Chapter 5. The details are presented for different datasets, network architectures and input image resolution.

Training ResNet18 on 84x84 images on mini-ImageNet. For all experiments, we use ResNet18 with input image size 84x84, train with the Adam optimizer with an initial learning rate $3 \cdot 10^{-4}$, which is decreased by a factor 10 once during training when no improvement in validation accuracy is observed for p consecutive epochs. We use $p = 20$ for training individual models, $p = 30$ for training ensembles and when distilling the model. When distilling an ensemble into one network, p is doubled. We use random crops and color augmentation during training as well as weight decay with parameter $\lambda = 5 \cdot 10^{-4}$. At training time we use random crop, color transformation and adding random noise as data augmentation. During the meta-testing stage, we take central crops of size 224×224 from images and feed them to the feature extractor. No other preprocessing is used at test time. The parameters used in distillation are the same as in Section 5.3.3 of the manuscript.

Training WideResNet28 on 80x80 images on mini-ImageNet. For all experiments, we use WideResNet28 with input image size 80x80, train with the Adam optimizer with an initial learning rate $1 \cdot 10^{-4}$, which is decreased by a factor 10 once during training when no improvement in validation accuracy is observed for p consecutive epochs. We use $p = 20$ for training individual models, $p = 30$ for training ensembles and when distilling the model. When distilling an ensemble into one network, p is doubled. We use random crops and color augmentation during training as well as weight decay with parameter $\lambda = 5 \cdot 10^{-4}$. We also set a dropout rate inside convolutional blocks to be 0.5 as described in. At training time we use random crop and color transformation only as data augmentation. During the meta-testing stage, we take central crops of size 80×80 from images and feed them to the feature extractor. No other preprocessing is used at test time. The parameters used in distillation are the same as in Section 5.3.3 of the manuscript. Here, the maximal ensemble size we evaluated is 10 and not 20 due to memory limitations on available GPUs. Therefore, to construct an ensemble of size 20 we merge two ensembles of size 10, that were trained independently.

Training ResNet18 on 224x224 images on tiered-ImageNet For all experiments, we use ResNet18 with input image size 224x224, train with the Adam optimizer with an initial learning rate $3 \cdot 10^{-4}$, which is decreased by a factor 10 once during training when no improvement in validation accuracy is observed for p consecutive epochs. We use $p = 20$ for training individual models, ensembles and for distilation. We use random crops and color augmentation during training as well as weight decay with parameter $\lambda = 1 \cdot 10^{-4}$. At training time we use random crop and color transformation. During the meta-testing stage, we take central crops of size 224×224 from images and feed them to the feature extractor. No other preprocessing is used at test time. The parameters used in distillation are the same as in Section 5.3.3 of the manuscript.

C.3 Additional Results for Few-shot Classification

In this section we report and analyze the performance of different ensemble types depending on their size for different network architectures and input image resolutions.

Few-shot Classification with ResNet18 on 84x84 images on mini-ImageNet. The results for 1- and 5-shot classification on *mini*-ImageNet are presented in Table C.2 and summarized in Figure C.3. We can see that Cooperation training is the most successful here for all ensemble sizes < 20 and other training strategies that introduce diversity tend to perform worse. This happens because single networks are far from overfitting the training set (as opposed to the case with 224x224 input size) and forcing diversity acts as harmful regularization. In contrary, cooperation training enforces useful learning signal and helps ensemble members achieve higher accuracy. Only for $n = 20$ where diversity matters more, robust ensembles perform the best.

Few-shot Classification with WideResNet28 on 80x80 images on mini-ImageNet. Results for 1- and 5-shot classification on *mini*-ImageNet are presented in Table C.1 and summarized in Figure C.3. In this case we can see again that Diverse training does not help since the networks do not memorize the training set. Robust ensembles outperform other training regimes emphasizing the importance of the proposed solution that generalizes across architectures.

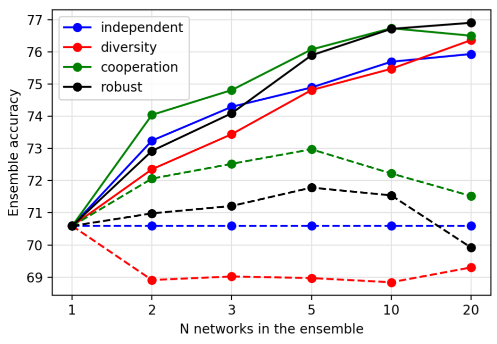
5-shot

Ensemble type	1	2	3	5	10	20
Independent	70.59 ± 0.51	73.24 ± 0.49	74.29 ± 0.48	74.89 ± 0.47	75.69 ± 0.47	75.93 ± 0.47
Diversity	70.59 ± 0.51	72.35 ± 0.47	73.44 ± 0.49	74.81 ± 0.48	75.47 ± 0.48	76.36 ± 0.47
Cooperation	70.59 ± 0.51	74.04 ± 0.47	74.81 ± 0.47	76.37 ± 0.48	76.73 ± 0.48	76.50 ± 0.47
Robust	70.59 ± 0.51	72.92 ± 0.50	73.09 ± 0.43	75.69 ± 0.42	76.71 ± 0.47	76.90 ± 0.48
Distilled Ensembles						
Robust-*dist*	–	73.04 ± 0.50	73.58 ± 0.49	74.35 ± 0.48	74.69 ± 0.49	75.24 ± 0.49
Robust-*dist*++	–	73.50 ± 0.49	74.17 ± 0.49	74.84 ± 0.49	75.12 ± 0.44	75.62 ± 0.48

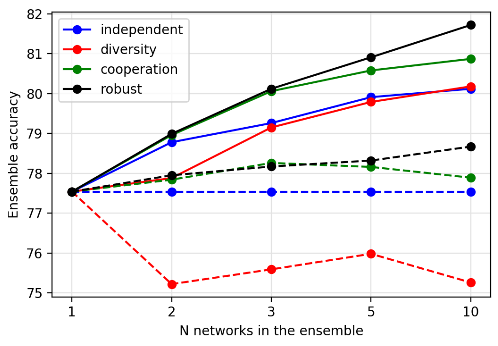
1-shot

Ensemble type	1	2	3	5	10	20
Independent	53.31 ± 0.64	55.72 ± 0.60	56.85 ± 0.64	57.90 ± 0.63	58.21 ± 0.63	58.56 ± 0.61
Diversity	53.31 ± 0.64	54.61 ± 0.62	55.90 ± 0.62	57.06 ± 0.63	57.49 ± 0.62	58.93 ± 0.64
Cooperation	53.31 ± 0.64	55.80 ± 0.64	57.13 ± 0.63	58.18 ± 0.64	58.63 ± 0.63	58.73 ± 0.62
Robust	53.31 ± 0.64	55.95 ± 0.62	56.27 ± 0.64	58.51 ± 0.65	59.38 ± 0.65	59.48 ± 0.65
Distilled Ensembles						
Robust-*dist*	–	56.84 ± 0.64	56.58 ± 0.65	57.13 ± 0.63	57.41 ± 0.65	58.11 ± 0.64
Robust-*dist* ++	–	56.53 ± 0.62	57.03 ± 0.64	57.48 ± 0.65	58.05 ± 0.63	58.67 ± 0.65

Table C.1 – **Few-shot classification accuracy on *Mini*ImageNet, using ResNet18 and 84x84 image size.** The first column gives the strategy, the top row indicates the number N of networks in an ensemble. Here, *dist* means that an ensemble was distilled into a single network, and '++' indicates that extra unannotated images were used for distillation. We performed 1 000 independent experiments on *mini*-ImageNet-test and report the average with 95% confidence interval. All networks are trained on *mini*-ImageNet-train set.



(a) ResNet18 with 84x84 input



(b) WideResNet28 with 80x80 input

Figure C.3 – **Dependency of ensemble accuracy on network architecture and input size for different ensemble strategies (one for each color) and various numbers of networks on *Mini*ImageNet 5-shots classification.** Solid lines give the ensemble accuracy after aggregating predictions. The average performance of single models from the ensemble is plotted with a dashed line. Best viewed in color.

5-shot

Ensemble type	1	2	3	5	10
Independent	77.54 ± 0.45	78.78 ± 0.45	79.26 ± 0.43	79.91 ± 0.44	80.12 ± 0.43
Diversity	77.54 ± 0.45	77.88 ± 0.45	79.15 ± 0.44	79.79 ± 0.44	80.18 ± 0.44
Cooperation	77.54 ± 0.45	78.96 ± 0.46	80.06 ± 0.44	80.58 ± 0.45	80.87 ± 0.43
Robust	77.54 ± 0.45	78.99 ± 0.45	80.12 ± 0.43	80.91 ± 0.43	81.72 ± 0.44
Distilled Ensembles					
Robust-*dist*	–	79.44 ± 0.44	79.84 ± 0.44	80.01 ± 0.42	80.85 ± 0.43
Robust-*dist*++	–	79.16 ± 0.46	80.00 ± 0.44	80.25 ± 0.42	81.11 ± 0.43

1-shot

Ensemble type	1	2	3	5	10
Independent	59.02 ± 0.63	60.07 ± 0.62	60.58 ± 0.61	61.24 ± 0.63	62.05 ± 0.61
Diversity	59.02 ± 0.63	58.87 ± 0.62	60.63 ± 0.61	61.30 ± 0.62	62.28 ± 0.61
Cooperation	59.02 ± 0.63	60.22 ± 0.62	61.03 ± 0.61	62.07 ± 0.61	62.42 ± 0.61
Robust	59.02 ± 0.63	60.92 ± 0.62	62.03 ± 0.62	62.78 ± 0.61	63.39 ± 0.62
Distilled Ensembles					
Robust-*dist*	–	61.07 ± 0.62	61.57 ± 0.61	62.24 ± 0.61	62.80 ± 0.62
Robust-*dist* ++	–	61.37 ± 0.62	62.01 ± 0.60	62.45 ± 0.62	63.25 ± 0.62

Table C.2 – **Few-shot classification accuracy on *Mini*ImageNet, using WideResNet28 and 80x80 image size.** The first column gives the strategy, the top row indicates the number N of networks in an ensemble. Here, *dist* means that an ensemble was distilled into a single network, and '++' indicates that extra unannotated images were used for distillation. We performed 1 000 independent experiments on *mini*-ImageNet-test and report the average with 95% confidence interval. All networks are trained on *mini*-ImageNet-train set.

Bibliography

[1] Cs143 computer vision course, brown univirsity. http://cs.brown.edu/courses/cs143/2011/results/proj3/senewman/.

[2] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 1, 2015.

[3] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels. Technical report, 2010.

[4] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[5] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. In *International Conference on Learning Representations (ICLR)*, 2018.

[6] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[7] Anurag Arnab and Philip HS Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[9] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] Moshe Bar and Shimon Ullman. Spatial context in recognition. *Perception*, 25:343–352, 1996.

[11] Ehud Barnea and Ohad Ben-Shahar. On the utility of context (or the lack thereof) for object detection. *arXiv preprint arXiv:1711.05471*, 2017.

[12] Evgeniy Bart and Shimon Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 672–679, 2005.

[13] Ruud Barth, Jochen Hemming, and Eldert J van Henten. Improved part segmentation performance by optimising realism of synthetic images using cycle generative adversarial networks. *arXiv preprint arXiv:1803.06301*, 2018.

[14] 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 (CVPR)*, 2016.

[15] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[16] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)*, 2019.

[18] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems (NIPS)*, pages 523–531, 2016.

[19] Alberto Bietti, Grégoire Mialon, Dexiong Chen, and Julien Mairal. A kernel perspective for regularizing deep neural networks. 2019.

[20] Leo Breiman. Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383, 1996.

[21] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011.

[22] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 778–792, 2010.

[23] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

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

[25] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40:834–848, 2018.

[26] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[27] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations (ICLR)*, 2019.

[28] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. *arXiv preprint arXiv:1903.12174*, 2019.

[29] Myung Jin Choi, Joseph J Lim, Antonio Torralba, and Alan S Willsky. Exploiting hierarchical context on a large database of object categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[30] Wenqing Chu and Deng Cai. Deep feature based contextual model for object detection. *Neurocomputing*, 275:1035–1042, 2018.

[31] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[32] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints.

[33] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[34] 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 (CVPR)*, 2016.

[35] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[37] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, 2000.

[38] Santosh K Divvala, Derek Hoiem, James H Hays, Alexei A Efros, and Martial Hebert. An empirical study of context in object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[39] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[40] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[41] Nikita Dvornik, Julien Mairal, and Cordelia Schmid. Modeling visual context is key to augmenting object detection datasets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[42] Nikita Dvornik, Julien Mairal, and Cordelia Schmid. Modeling visual context is key to augmenting object detection datasets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[43] Nikita Dvornik, Julien Mairal, and Cordelia Schmid. On the importance of visual context for data augmentation in scene understanding. *arXiv preprint arXiv:1809.02492*, 2018.

[44] Nikita Dvornik, Konstantin Shmelkov, Julien Mairal, and Cordelia Schmid. Blitznet: A real-time deep network for scene understanding. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[45] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[46] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[47] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[48] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35:1915–1929, 2012.

[49] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(4):594–611, 2006.

[50] 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 (PAMI)*, 32(9):1627–1645, 2010.

[51] Sanja Fidler, Roozbeh Mottaghi, Alan Yuille, and Raquel Urtasun. Bottom-up segmentation for top-down detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[52] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. 2017.

[53] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321:321–331, 2018.

[54] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. 2001.

[55] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. DSSD: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[56] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. 2015.

[57] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. 2018.

[58] Georgios Georgakis, Arsalan Mousavian, Alexander C Berg, and Jana Kosecka. Synthesizing training data for object detection in indoor scenes. In *Robotics: Science and Systems (RSS)*, 2017.

[59] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[60] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations (ICLR)*, 2018.

[61] Ross Girshick. Fast R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[62] 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 (CVPR)*, 2014.

[63] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018.

[64] 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 (NIPS)*, 2014.

[65] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[66] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[67] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947, 2000.

[68] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Understanding real world indoor scenes with synthetic data.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[69] Stephen José Hanson and Lorien Y Pratt. Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems (NIPS)*, 1989.

[70] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[71] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[72] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[73] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[74] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. *arXiv preprint arXiv:1811.08883*, 2018.

[75] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[76] 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 (CVPR)*, 2016.

[77] Xuming He, Richard S Zemel, and Debajyoti Ray. Learning and incorporating top-down cues in image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006.

[78] Alex Hernández-García and Peter König. Data augmentation instead of explicit regularization. *arXiv preprint arXiv:1806.03852*, 2018.

[79] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

[80] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018.

[81] Herve Jegou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(9):1704–1716, 2011.

[82] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1219–1228, 2018.

[83] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. *ACM Transactions on Graphics (TOG)*, 30(6):157, 2011.

[84] Anna Khoreva, Rodrigo Benenson, Jan Hendrik Hosang, Matthias Hein, and Bernt Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[85] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[86] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[87] Gregory Koch. Siamese neural networks for one-shot image recognition. 2015.

[88] Iasonas Kokkinos. UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[89] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques.* 2009.

[90] Shu Kong and Charless C Fowlkes. Recurrent pixel embedding for instance grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[91] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[92] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[93] Sanjiv Kumar and Martial Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. 2003.

[94] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv:1811.00982*, 2018.

[95] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[96] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[97] Yann LeCun. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[98] 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.

[99] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[100] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869, 2017.

[101] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision (IJCV)*, 43:29–44, 2001.

[102] Hengduo Li, Bharat Singh, Mahyar Najibi, Zuxuan Wu, and Larry S Davis. An analysis of pre-training on object detection. *arXiv preprint arXiv:1904.05871*, 2019.

[103] Yi Li, Kaiming He, Jian Sun, et al. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[104] Zicheng Liao, Ali Farhadi, Yang Wang, Ian Endres, and David Forsyth. Building a dictionary of image fragments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[105] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid. Exploring context with deep structured models for semantic segmentation.

IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI), 40:1352–1366, 2018.

[106] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[107] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[108] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[109] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

[110] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[111] 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 (CVPR)*, 2015.

[112] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[113] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[114] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. 2015.

[115] Dhruv Kumar Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[116] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[117] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(11):2624–2637, 2013.

[118] Yair Movshovitz-Attias, Takeo Kanade, and Yaser Sheikh. How useful is photo-realistic rendering for visual learning? In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[119] Naila Murray, Luca Marchesotti, and Florent Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2408–2415, 2012.

[120] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[121] Dvornik Nikita. Implementation of blitznet. `https://github.com/dvornikita/blitznet`, 2017.

[122] Dvornik Nikita. Implementation of context-driven data augmentation pipeline. `https://github.com/dvornikita/context_aug`, 2018.

[123] Dvornik Nikita. Implementation of ensemble methods for few-shot classification. `https://github.com/dvornikita/fewshot_ensemble`, 2019.

[124] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[125] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[126] Constantine Papageorgiou, Michael Oren, and Tomaso A. Poggio. A general framework for object detection. *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 555–562, 1998.

[127] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a deep convolutional network for

semantic image segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[128] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[129] 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 (CVPR)*, 2016.

[130] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning deep object detectors from 3d models. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[131] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 143–156, 2010.

[132] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[133] David C Plaut et al. Experiments on learning by back propagation. 1986.

[134] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics (SIGGRAPH'03)*, 22(3):313–318, 2003.

[135] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[136] Weichao Qiu and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[137] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[138] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[139] 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 (CVPR)*, 2016.

[140] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[141] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations (ICLR)*, 2018.

[142] 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 (NIPS)*, 2015.

[143] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003.

[144] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[145] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[146] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[147] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Semantic foggy scene understanding with synthetic data. *International Journal of Computer Vision (IJCV)*, 126:973–992, 2018.

[148] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations (ICLR)*, 2017.

[149] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision (IJCV)*, 105(3):222–245, 2013.

[150] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[151] Robert E Schapire. A brief introduction to boosting.

[152] 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.

[153] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[154] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. How good is my gan? In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[155] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006.

[156] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[157] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[158] Leon Sixt, Benjamin Wild, and Tim Landgraf. Rendergan: Generating realistic labeled data. *Frontiers in Robotics and AI*, 5:66, 2018.

[159] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[160] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[161] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[162] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 843–852, 2017.

[163] Marvin Teichmann, Michael Weber, Marius Zoellner, Roberto Cipolla, and Raquel Urtasun. Multinet: Real-time joint semantic reasoning for autonomous driving. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1013–1020, 2018.

[164] Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. 1998.

[165] Pavel Tokmakov, Yu-Xiong Wang, and Martial Hebert. Learning compositional representations for few-shot recognition. *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[166] Antonio Torralba. Contextual priming for object detection. *International Journal of Computer Vision*, 53(2):169–191, 2003.

[167] Antonio Torralba and Pawan Sinha. Statistical context priming for object detection. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2001.

[168] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[169] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International Journal of Computer Vision (IJCV)*, 104(2):154–171, 2013.

[170] Jakob Verbeek and Bill Triggs. Scene segmentation with conditional random fields learned from partially labeled images. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[171] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. 2008.

[172] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[173] Paul Viola, Michael Jones, et al. Robust real-time object detection. 2001.

[174] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[175] Chao Weng, Dong Yu, Michael L Seltzer, and Jasha Droppo. Deep neural networks for single-channel multi-talker speech recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(10):1670–1679, 2015.

[176] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Gp-gan: Towards realistic high-resolution image blending. In *ACM International Conference on Multimedia (ACMMM)*, 2019.

[177] Jianwei Yang, Jiasen Lu, Dhruv Batra, and Devi Parikh. A faster pytorch implementation of faster r-cnn. *https://github.com/jwyang/faster-rcnn.pytorch*, 2017.

[178] Jimei Yang, Brian Price, Scott Cohen, and Ming-Hsuan Yang. Context driven scene parsing with attention to rare classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[179] Bangpeng Yao and Li Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[180] Jian Yao, Sanja Fidler, and Raquel Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[181] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Learning embedding adaptation for few-shot learning. *arXiv preprint arXiv:1812.03664*, 2018.

[182] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 467–483, 2016.

[183] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

[184] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.

[185] Ruichi Yu, Xi Chen, Vlad I Morariu, and Larry S Davis. The role of context selection in object detection. In *British Machine Vision Conference (BMVC)*, 2016.

[186] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, 2016.

[187] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018.

[188] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*, 2017.

[189] Yanzhao Zhou, Yi Zhu, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Weakly supervised instance segmentation using class peak response. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[190] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. *arXiv preprint arXiv:1906.11172*, 2019.