HAL open science

Approaches for incremental learning and image generation

Konstantin Shmelkov

To cite this version:

Konstantin Shmelkov. Approaches for incremental learning and image generation. Mathematical Physics [math-ph]. Université Grenoble Alpes, 2019. English. NNT : 2019GREAM010 . tel-02183259

HAL Id: tel-02183259

https://theses.hal.science/tel-02183259

Submitted on 15 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Communauté UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Konstantin SHMELKOV

Thèse dirigée par **Cordelia SCHMID**, directeur de recherche , INRIA
et codirigée par **Karteek Alahari**, CR, INRIA
préparée au sein du **Laboratoire Laboratoire Jean Kuntzmann** dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Approches pour l'apprentissage incrémental et la génération des images

Approaches for incremental learning and image generation

Thèse soutenue publiquement le **29 mars 2019**,
devant le jury composé de :

Madame Svetlana Lazebnik
Associate Professor, University of Illinois at Urbana-Champaign, Rapporteur
Monsieur Victor Lempitsky
Associate Professor, Skolkovo Institute of Science and Technology, Rapporteur
Monsieur Josef Sivic
Directeur de Recherche, Inria, Examinateur
Monsieur Florent Perronnin
Directeur de Recherche, NAVER LABS Europe, Président
Madame Cordelia Schmid
Directeur de Recherche, Inria, Directeur de thèse
Monsieur Karteek Alahari
Chargé de Recherche, Inria, Co-directeur de thèse

Abstract

This dissertation explores two related topics in the context of deep learning: incremental learning and image generation. Incremental learning studies training of models with the objective function evolving over time, e.g., addition of new categories to a classification task. Image generation seeks to learn a distribution of natural images for generating new images resembling original ones.

Incremental learning is a challenging problem due to the phenomenon called *catastrophic forgetting*: any significant change to the objective during training causes a severe degradation of previously learned knowledge. We present a learning framework to introduce new classes to an object detection network. It is based on the idea of knowledge distillation to counteract catastrophic forgetting effects: fixed copy of the network evaluates old samples and its output is reused in an auxiliary loss to stabilize learning of new classes. Our framework mines these samples of old classes on the fly from incoming images, in contrast to other solutions that keep a subset of samples in memory.

On the second topic of image generation, we build on the Generative Adversarial Network (GAN) model. Recently, GANs significantly improved the quality of generated images. However, they suffer from poor coverage of the dataset: while individual samples have great quality, some modes of the original distribution may not be captured. In addition, existing GAN evaluation methods are focused on image quality, and thus do not evaluate how well the dataset is covered, in contrast to the likelihood measure commonly used for generative models. We present two approaches to address these problems.

The first method evaluates class-conditional GANs using two complementary measures based on image classification — GAN-train and GAN-test, which approximate recall (diversity) and precision (quality of the image) of GANs respectively. We evaluate several recent GAN approaches based on these two measures, and demonstrate a clear difference in performance. Furthermore, we observe that the increasing difficulty of the dataset, from CIFAR10 over CIFAR100 to ImageNet, shows an inverse correlation with the quality of the GANs, as clearly evident from our measures.

Inspired by our study of GAN models, we present a method to explicitly enforce dataset coverage during the GAN training phase. We develop a generative model that combines GAN image quality with VAE architecture in the feature space engendered by a flow-based model Real-NVP. This allows us to evaluate a valid likelihood and simultaneously relax the independence assumption in RGB space which is common for VAEs. We achieve Inception score and FID competitive with state-of-the-art GANs, while maintaining good likelihood for this class of models.

Keywords: incremental learning, lifelong learning, object detection, generative adversarial networks, convolutional neural networks, computer vision, machine learning.

Résumé

Cette thèse explore deux sujets liés dans le contexte de l'apprentissage profond : l'apprentissage incrémental et la génération des images. L'apprentissage incrémental étudie l'entrainement des modèles dont la fonction objective évolue avec le temps (exemple : Ajout de nouvelles catégories à une tâche de classification). La génération d'images cherche à apprendre une distribution d'images naturelles pour générer de nouvelles images ressemblant aux images de départ.

L'apprentissage incrémental est un problème difficile dû au phénomène appelé *l'oubli catastrophique* : tout changement important de l'objectif au cours de l'entrainement provoque une grave dégradation des connaissances acquises précédemment. Nous présentons un cadre d'apprentissage permettant d'introduire de nouvelles classes dans un réseau de détection d'objets. Il est basé sur l'idée de la distillation du savoir pour lutter les effets de l'oubli catastrophique : une copie fixe du réseau évalue les anciens échantillons et sa sortie est réutilisée dans un objectif auxiliaire pour stabiliser l'apprentissage de nouvelles classes. Notre framework extrait ces échantillons d'anciennes classes à la volée à partir d'images entrantes, contrairement à d'autres solutions qui gardent un sous-ensemble d'échantillons en mémoire.

Pour la génération d'images, nous nous appuyons sur le modèle du réseau adverse génératif (en anglais generative adversarial network ou GAN). Récemment, les GANs ont considérablement amélioré la qualité des images générées. Cependant, ils offrent une pauvre couverture de l'ensemble des données : alors que les échantillons individuels sont de grande qualité, certains modes de la distribution d'origine peuvent ne pas être capturés. De plus, contrairement à la mesure de vraisemblance couramment utilisée pour les modèles génératives, les méthodes existantes d'évaluation GAN sont axées sur la qualité de l'image et n'évaluent donc pas la qualité de la couverture du jeu de données. Nous présentons deux approches pour résoudre ces problèmes.

La première approche évalue les GANs conditionnels à la classe en utilisant deux mesures complémentaires basées sur la classification d'image — GAN-train et GAN-test, qui approchent respectivement le rappel (diversité) et la précision (qualité d'image) des GANs. Nous évaluons plusieurs approches GANs récentes en fonction de ces deux mesures et démontrons une différence de performance importante. De plus, nous observons que la difficulté croissante du jeu de données, de CIFAR10 à ImageNet, indique une corrélation inverse avec la qualité des GANs, comme le montre clairement nos mesures.

Inspirés par notre étude des modèles GANs, la seconde approche applique explicitement la couverture d'un jeux de données pendant la phase d'entrainement de GAN. Nous développons un modèle génératif combinant la qualité d'image GAN et l'architecture VAE dans l'espace latente engendré par un modèle basé sur le flux, Real-NVP. Cela nous permet d'évaluer une

vraisemblance correcte et d'assouplir simultanément l'hypothèse d'indépendance dans l'espace RVB qui est courante pour les VAE. Nous obtenons le score Inception et la FID en concurrence avec les GANs à la pointe de la technologie, tout en maintenant une bonne vraisemblance pour cette classe de modèles.

Mots-clefs : apprentissage incrémental, apprentissage tout au long de la vie, detection d'objet, réseaux adverses génératifs, réseaux de neurones convolutionnels, vision par ordinateur, apprentissage machine

I was astonished to discover that working in the laboratory — doing science in collaboration with interesting and creative people — is dramatically different from taking courses and reading about science.

Eric Kandel

Acknowledgements

I would like to thank my supervisors, Dr Cordelia Schmid and Dr Karteek Alahari, for teaching me how the research actually works and exercising infinite patience during my PhD. I would also like to thank my co-authors, Nikita Dvornik, Thomas Lucas, Julien Mairal and Jakob Verbeek, for enriching discussions and outstanding example of scientific ethics. I am very grateful to jury members who agreed to participate in my defense and critically review the manuscript. Special thanks go to Nathalie Gillot, who helped me tremendously in fights against bureaucracy. I would like to express my gratitude towards my friends for their support and company on the journey of inner discovery, they helped me to grasp subtle differences between art and science. I cannot express how grateful I am for all the support and love I received from my parents. Last but not least I would like to thank my fellow labmates at Inria for tolerating me during my daily struggles and challenging deadline periods.

Contents

Contents xi

1 Introduction 1
1.1 Goals and challenges . . . 3
1.2 Contributions . . . 10

I Incremental learning in object detection 15

2 Related work 19
2.1 Catastrophic forgetting . . . 20
2.2 Class-incremental learning . . . 23

3 Incremental learning of new classes 25
3.1 Object detection network . . . 26
3.2 Dual-network learning . . . 27
3.3 Sampling strategy . . . 29
3.4 Experiments . . . 29
3.5 Other alternatives . . . 38
3.6 Conclusion . . . 38

II Generative models 41

4 Related work 45
4.1 Coverage-driven training and maximum likelihood estimation . . . 46
4.2 Adversarial models and quality-driven training . . . 48

4.3 Hybrid approaches 50
4.4 Conditioning of generative models 51
4.5 Evaluation of GAN models 51
4.6 Data augmentation with GANs 53

5 Evaluation of conditional GANs 55
5.1 GAN-train and GAN-test 55
5.2 Datasets and methods 57
5.3 Experiments 59
5.4 Summary 67

6 Adversarial training of partially invertible variational autoencoders 69
6.1 Coverage and quality driven training 70
6.2 Experimental evaluation 72
6.3 Model refinements and implementation details 81
6.4 Conclusion 84

7 Conclusion 85
7.1 Summary of contributions 86
7.2 Future work 87

A Publications 91

B Software 93

C BlitzNet: a real-time deep network for scene understanding 95
C.1 Introduction 95
C.2 Related work 97
C.3 Scene understanding with BlitzNet 98
C.4 Experiments 102
C.5 Conclusion 110

Bibliography 113

Chapter 1

Introduction

Computer languages of the future will be more concerned with goals and less with procedures specified by the programmer.

— Marvin Minsky, *Turing Award Lecture "Form and Content in Computer Science"*

Contents

1.1 Goals and challenges **3**
1.1.1 Incremental learning 4
1.1.2 Generative models 7
1.2 Contributions **10**
1.2.1 Incremental learning of object detectors without catastrophic forgetting 10
1.2.2 How good is my GAN? 11
1.2.3 Adversarial training of partially invertible variational autoencoders 13
1.2.4 Outline of the thesis 14

The quest for artificial intelligence is an ongoing adventure to design machines capable of understanding the real world and to act accordingly on the level of,

or even surpassing human intelligence. Over the last 70 years the focus of AI research has gradually moved from symbolic reasoning [126] and hand-crafted algorithms [176] to statistical learning from data [20]. Today we train neural networks that make sense of raw data and can learn millions of parameters in an end-to-end manner [58]. This paradigm is known as deep learning.

Artificial neural networks (ANNs) are loosely inspired by biological neural networks, but most similarities disappear on closer inspection. Subtle differences suggest why their properties are so different from humans: ANNs are typically trained in supervised regimes with a lot of data using backpropagation, while the human brain learns with very little supervision and backpropagation is considered biologically implausible [19]. Humans are capable of continuous learning throughout their life, adding new objectives and refining prior knowledge, which demonstrates the amazing flexibility of the human brain. ANNs, on the other hand, assume objectives are known before training and all the data are i.i.d., which makes them incapable of learning from ever changing streams of data.

Advances in connectionist research over the last two decades enabled end-to-end training of neural networks on huge amounts of raw data. In particular, deep learning has achieved incredible success in recent years: neural networks have reached super-human performance in image classification [69–71], significantly advanced state-of-the-art in several computer vision tasks (e.g., object detection [56, 99, 148], semantic segmentation [27, 28, 101], instance segmentation [33, 68]) as well as machine translation [14, 72, 174], speech recognition [8, 15, 63], and playing Go [167, 168]. A common feature of these tasks is large amounts of annotated raw sensory data, where learned representations proved to be better than hand-crafted representations of the past.

Computer vision seeks to understand the world from images and videos, which are imperfect projections of reality, and partially capture its semantic variety. It is an important subfield of AI research: if the main AI objective is to automate tasks that the human brain can do, then computer vision automates what the human visual system accomplishes. A major part of computer vision is essentially pattern recognition: discovery of regularities in data, based on previously extracted statistical information. Among basic pattern recognition problems is image classification: an image is associated with a class label, which can be interpreted as high-level semantic description. Convolutional neural networks (CNNs) are empirically very successful at solving this problem.

However, when trained for classification CNNs tend to learn minimalist representations in the sense that they keep enough information to separate the classes, and throw the rest away. One manifestation of this is that the only way to extend an already trained network to teach it a few more classes (that differ significantly from the original training set) is a complete retraining. Typically, this problem

is mitigated by choosing many different classes (for example, 1000 classes of ImageNet [155]). More classes means more data, hence longer training time for every round of retraining. Naturally it raises a question: is there a way to avoid wasting computational resources, to reuse pretrained networks and to extend them to new classes in a short time? Computational resources become cheaper every year, but the amount of available data grows at a much faster rate: Instagram alone (relatively large social network, but still a small part of internet) has more than 80 million images and videos uploaded every day! [1]

Computer vision is often described as a reduction of information: rich visual description gets distilled to high-level semantic description, e.g., a class label, segmentation mask or 3D structure. Perhaps the key to extendable neural networks is connected with the inverse task: computer graphics that creates new images and videos out of semantic description. As Geoffrey Hinton describes it: "In order to do computer vision, first learn how to do computer graphics". Indeed, computer graphics roughly corresponds to inversion of semantic description back to raw pixel space. Humans have the ability to visualise rich semantic concepts with very little description, for instance by visualising the image of a bear when hearing the word "bear". It seems to be a more difficult problem than pattern recognition because the latter is essentially reduction of information, while the former involves reconstruction of fairly complex structure from a short description. Although generative models achieved impressive success during the last few years [64, 78, 198], they are still far from generating convincing high-resolution images on an arbitrary subject, let alone videos [100]. In contrast, the human capability of imagination seems to be interlinked with episodic memory (recalling events of the past) as well as planning of future actions ("what will happen if I do X?") as evidenced by neuroscience research [196]. The ability to recall the past when needed is crucial to maintain learning throughout our life. While CNNs excel in pattern recognition, they are deprived of memory, and thus struggle at continuous learning.

1.1 Goals and challenges

This dissertation discusses two independent, but related topics: incremental learning and generative adversarial networks. The first one is dedicated to designing a framework where number of detected classes can be extended multiple times, without complete retraining. The second one is dedicated to modern generative models, particularly to adversarial ones. It discusses a common problem of adversarial models "mode-dropping", how to evaluate its extent, and introduces a new generative model to reduce effects of mode-dropping.

1. https://www.brandwatch.com/blog/instagram-stats/

1.1.1 Incremental learning

Several applications of computer vision are connected to robotics. Indeed, robots would benefit from having a powerful vision system, ideally a human-level one. This would make them capable of understanding and navigating the visual world, but is a challenging problem because of a plethora of hardware limitations. The deep learning revolution has raised hopes that we will be able to build autonomous agents equipped with human-level vision and understanding in the near future. However, currently adopted lifecycle of deep learning models is hardly compatible with the autonomous agent dream [173]. They require a lot of data collected beforehand and most often, manually annotated as well as enormous computational resources to train [2] (state-of-the-art models already use GPU-weeks as a unit of training time). After this long training is completed, they are deployed most often in the cloud. Furthermore, they can only be replaced by newer networks trained from scratch. There is no practical way of altering an objective or extending a network in any significant way after training, or even during the training phase. It means that all the required information about the problem should be fixed before the training starts.

The real world is infinitely diverse, making it impossible to precisely describe and predict what an autonomous agent might encounter in the future. Ideally, a neural network should be prepared to work with data outside of the training distribution, so-called *open set recognition* [17, 18]. It means that the network can reject a sample as unknown if it cannot classify it confidently. This setup, however, presents a bigger challenge than it seems.

Another way to tackle the real world complexity is to continuously expand the initial scope of the problem, and continue training after deployment. This is what humans tend to do: continue learning throughout their whole life. However, early connectionist research [51, 112] showed that neural networks are particularly vulnerable to changing the objective on the fly. This phenomenon is called *catastrophic forgetting (or interference)* because new objective leads previous knowledge to be discarded extremely fast.

Catastrophic forgetting makes any kind of evolving training objective difficult. It can be mitigated partially by keeping all available data, and carefully balancing the new and the old objectives, but often requires a tedious manual tuning of hyperparameters and training schedule. The holy grail of learning without catastrophic forgetting is, of course, continuous retraining on-the-fly performed by autonomous agents, but it is currently out of reach: modern neural networks require large amounts of data and computational resources for training, even assuming we have a perfect solution for catastrophic forgetting.

2. https://blog.openai.com/ai-and-compute/

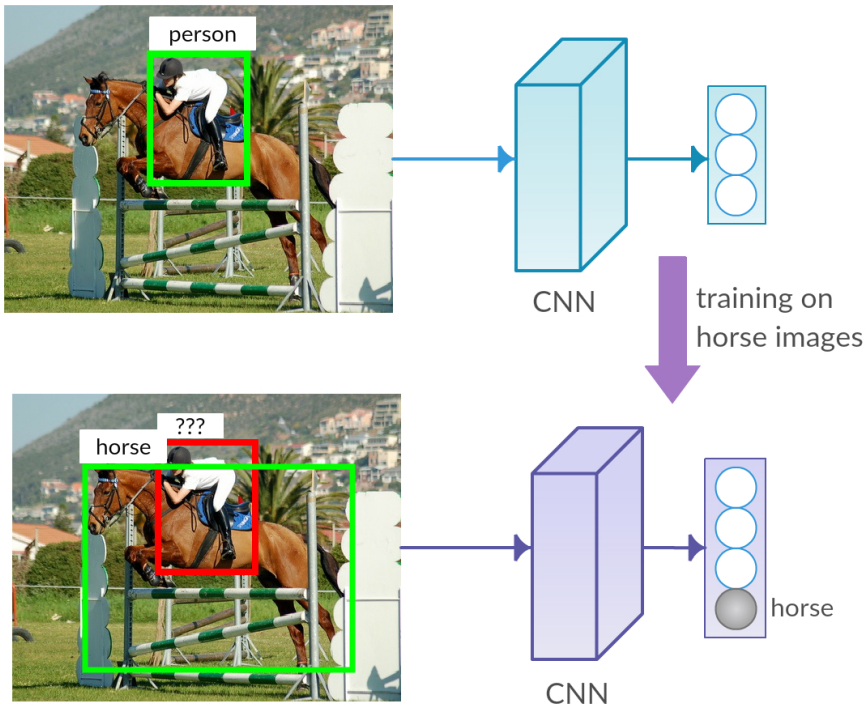

Figure 1.1 – Catastrophic forgetting. An object detector network originally trained for three classes, including *person*, detects the rider (top). When the network is retrained with images of the new class *horse*, it detects the horse in the test image, but fails to localize the rider (bottom).

Incremental learning is a setup where the training objective does not change radically, but rather expands over time with new classes and new samples following the evolution of datasets. It is desirable for networks deployed in the cloud, because it avoids retraining from scratch, and thus wasting computational resources. In certain cases, it is even required, e.g., a supermarket like Amazon Go, where new products can arrive every day. Catastrophic forgetting heavily impedes the incremental learning process: even when all incoming new data are stored and added to the constantly growing training set, the resulting network has subpar performance. Thus catastrophic forgetting is a big and important challenge in incremental learning.

Let us illustrate how catastrophic forgetting manifests itself in class-incremental learning of object detection task without memory. Consider the example in Figure 1.1: a trained model can detect a class *person* and recognizes a rider. A horse is filtered out as background because it was not among the classes considered for training. Note that the horse is not "never seen" data: *the network has been*

explicitly trained to recognize patches with horses as background. Now we train the model on images with annotated horses so it can reliably recognize the class *horse*. However, due to catastrophic forgetting the network loses the ability to recognize other classes including *person*. This is not the only problem here. In the absence of prior training data, the object detector will fail because existing representations are too crude to separate a new class from the background. Incremental learning requires the model to discover more powerful representations and move the decision boundary to free enough space for a new class. In the image classification task (in the absence of dedicated background class), an image of a new unseen class is often misclassified with high confidence as one of the previously learned classes. Strictly speaking, entirely preventing forgetting in this case impedes learning new classes. Thus, incremental learning introduces a challenge to gradually relearn new outputs for potentially known inputs, without disturbing the unrelated representations, which makes it more difficult than applications where catastrophic forgetting is the only issue.

Catastrophic forgetting in neural networks is strikingly different from the human brain: while we do forget information we do not use anymore, it is a gradual slow process that is observed in adults. However, there is an interesting evidence that human infants experience asymmetric interference reminiscent of catastrophic forgetting in neural networks. In [139], asymmetric interference was observed in a sequential category learning task in 3- and 4-month-old infants. They learned two categories, "dog" and "cat", sequentially from a series of images. It turned out that subsequent learning of "dog" category disrupts the previously learned "cat" category. However, learning of "cat" after the initial exposure to "dog" does not interfere with previously learned "dog" category. This experiment inspired a hypothesis that neural networks can model short-term memory in human infants [110].

Growing evidence from neuroscience [121, 196] suggests that the hippocampus, the brain region that is ultimately responsible for short-term memory and participating in its consolidation into long-term memory, is also linked to our imagination. It is reasonable to hypothesize that our ability to *imagine* real and unreal objects (in addition to memory retrieval) enables incremental learning and prevents catastrophic forgetting from damaging our previously learned skills. It is possible that when we encounter new objects we mentally compare them with what we can remember or imagine. This allows us to separate a new category from familiar data, essentially an outlier detection task. It is often a failure of discriminative models: they misclassify images that look nothing like training set while maintaining high confidence. We prefer discriminative models because they are easier to train than generative ones, especially when classes are very different [20]. However, a model trained to differentiate between sharks and bears may concentrate on background cues and will likely recognize a ship as shark. At

the same time, a generative model capable of outputing images of both sharks and bears should be able to guess that ships are nothing like sharks. Indeed, generative models can be used to detect outliers in a learned distribution [135]. Besides, in the context of incremental learning, a generative model has to learn to generate samples of a new class without losing the ability to model the old ones: it is a much more natural case of catastrophic forgetting that does not require modification of existing representations.

1.1.2 Generative models

As discussed above, generative models may be more robust to catastrophic forgetting and are more suitable for incremental learning than their discriminative counterparts. However, natural images are notoriously difficult to model. Successful recent generative models can be divided into two broad families, which are trained in fundamentally different ways. The first is trained using likelihood-based criteria, which ensure that all the training data points are well covered by the model. This category includes variational autoencoders [83, 84], autoregressive models such as PixelCNNs [159, 188], and flow-based models such as Real-NVP [38]. The second category is trained based on a signal that measures to what extent (statistics of) samples from the model can be distinguished from (statistics of) the training data, i.e., based on the quality of samples drawn from the model. This is the case for generative adversarial networks (GANs) [59] and its numerous variants, as well as moment matching methods [95]. Currently, GAN is the model of choice for natural image generation mainly due to high fidelity of samples.

GAN [59] is a deep neural net architecture composed of a pair of competing networks: a generator and a discriminator. This model is trained by alternately optimizing two objective functions so that the generator learns to produce samples resembling real images, and the discriminator learns to better discriminate between real and fake data. Such a paradigm has huge potential, as it can learn to generate any data distribution. This has been exploited with great success in several computer vision problems, such as text-to-image [199] and image-to-image [76, 202] translation, video-to-video translation [190], pose transfer [123], facial animation [138], super-resolution [93], and realistic natural image generation [78].

Since the original GAN model [59] was proposed, many variants have appeared in the past few years, for example, to improve the quality of generated images [30, 36, 78, 115], or to stabilize the training procedure [12, 64, 109, 115, 127, 200]. GANs have also been modified to generate images of a given class by conditioning on additional information, such as the class label [42, 114, 116, 129]. With many GAN modifications being regularly proposed in the literature, a critical question is *how these models can be evaluated and compared to each other*.

Evaluation and comparison of GANs, or equivalently, the images generated by GANs, is challenging. This is in part due to the lack of an explicit likelihood measure [180], which is commonplace in comparable probabilistic models [84, 159]. Thus, much of the previous work has resorted to a mere subjective visual evaluation in the case of images synthesized by GANs. As seen from the sample images generated by a state-of-the-art GAN [115] in Figure 1.5, it is impossible to judge their quality precisely with a subjective evaluation. Recent work in the past two years has begun to target this challenge through quantitative measures for evaluating GANs [73, 78, 105, 158].

Inception score (IS) [158] and Fréchet Inception distance (FID) [73] were suggested as ad-hoc measures correlated with the visual quality of generated images. Inception score measures the quality of a generated image by computing the KL-divergence between the (logit) response produced by this image and the marginal distribution, i.e., the average response of all the generated images, using an Inception network [175] trained on ImageNet. In other words, Inception score does not compare samples with a target distribution, and is limited to quantifying the diversity of generated samples. Fréchet Inception distance compares Inception activations (responses of the penultimate layer of the Inception network) between real and generated images. This comparison however approximates the activations of real and generated images as Gaussian distributions (cf. equation (4.12)), computing their means and covariances, which are too crude to capture subtle details. Both these measures rely on an ImageNet-pretrained Inception network, which is far from ideal for other datasets, such as faces and biomedical imaging. Overall, IS and FID are useful measures to evaluate how training advances, but they guarantee no correlation with performance on real-world tasks. Besides, while they are somewhat sensitive to the diversity of generated samples, *they do not allow to distinguish sample quality from diversity*. Both measures provide essentially single numbers that does not allow to evaluate the quality/diversity trade-off.

An alternative evaluation is to compute the distance of the generated samples to the real data manifold in terms of precision and recall [105]. Here, high precision implies that the generated samples are close to the data manifold, and high recall shows that the generator outputs samples that cover the manifold well. These measures remain idealistic as they are impossible to compute on natural image data, whose manifold is unknown. Indeed, the evaluation in [105] is limited to using synthetic data composed of gray-scale triangles.

Ideally we would like to evaluate likelihood estimation on held-out data. It is a mathematically justified measure of how well we cover the dataset and has been used in most generative models (VAE [84], PixelRNN [188], Real-NVP [38]) as a training objective, except in the case of GANs. While being useful for evaluation, likelihood presents a number of drawbacks. Likelihood-based models are trained

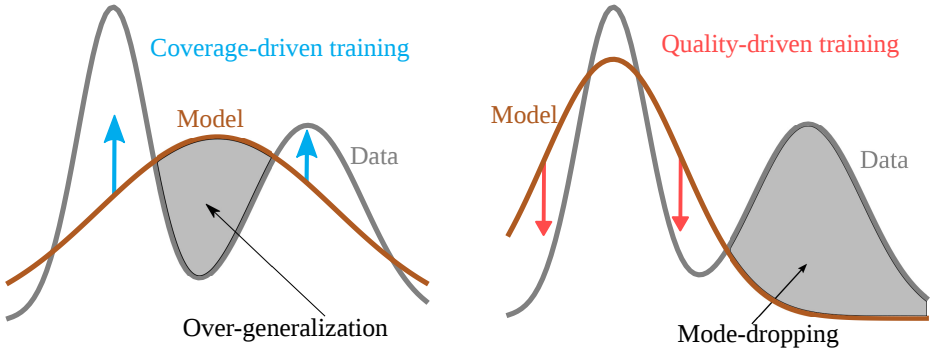

Figure 1.2 – Intuitive explanation of the difference between coverage-driven (CDT) and quality-driven training (QDT), in a one dimensional setting. CDT pulls probability mass towards points from regions of high density of the distribution underlying the data, while QDT pushes mass out of low-density regions. If the model is not flexible enough (in the example, it has too few modes), these training procedures lead to very different compromises in practice.

to put probability mass on all the elements of the training set, however fitting all natural images perfectly would require infinite flexibility (even a finite, but reasonably big dataset is already too difficult to fit perfectly). The lack of flexibility forces models to *over-generalize* and assign probability mass to non-realistic images in order to cover all modes. Limiting factors in such models are the use of fully factorized decoders in variational autoencoders, restriction to the class of fully invertible functions in Real-NVP, and the lack of latent variables to induce global pixel dependencies in autoregressive models. Adversarial training on the other hand explicitly ensures that the model does not generate samples that can be distinguished from training images. This goes, however, at the expense of covering all the training samples, a phenomenon known as "mode collapse" [12]. Moreover, adversarial trained models typically have a low-dimensional support, which prevents the use of likelihood to assess coverage of held-out data. Figure 1.2 gives an intuitive illustration of the complementary approaches of coverage (likelihood-based) and quality (adversarial) driven training.

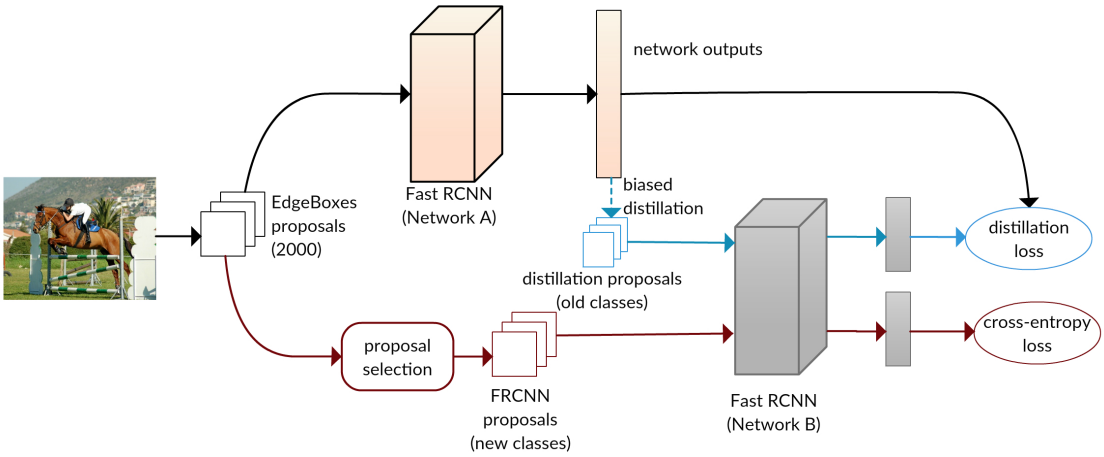

Figure 1.3 – Overview of our framework for learning object detectors incrementally. It is composed of a frozen copy of the detector (Network A) and the detector (Network B) adapted for the new class(es).

1.2 Contributions

1.2.1 Incremental learning of object detectors without catastrophic forgetting

Our first contribution is dedicated to class-incremental learning in an object detection framework. We focus on a challenging setup that does not store old training images in memory. Using only the training samples for the new classes, we propose a method for not only adapting the old network to the new classes, but also ensuring performance on the old classes does not degrade (or only moderately). The core of our approach is a training procedure balancing the interplay between predictions on the new classes, i.e., cross-entropy loss, and a new distillation loss, which minimizes the discrepancy between responses for old classes from the original and the new networks. The overall approach is illustrated in Figure 1.3.

We use a frozen copy of the original detection network to compute the distillation loss. This loss is related to the concept of "knowledge distillation" proposed in [74] to transfer knowledge from a big neural network to a much smaller one. We specifically target the problem of object detection, which has the additional challenge of localizing objects with bounding boxes, unlike other attempts [96, 144] limited to the image classification task. We demonstrate experimental results on the PASCAL VOC 2007 [46] and COCO [98] datasets using Fast R-CNN [56] as the network. Our results show that we can add new classes incrementally to an existing network affecting the performance on the original classes only moderately and with no access to the original training data. We also evaluate variants of our

method empirically, and show the influence of distillation and the loss function.

This work was published in ICCV'17 [164] and is presented in Chapter 3.

1.2.2 How good is my GAN?

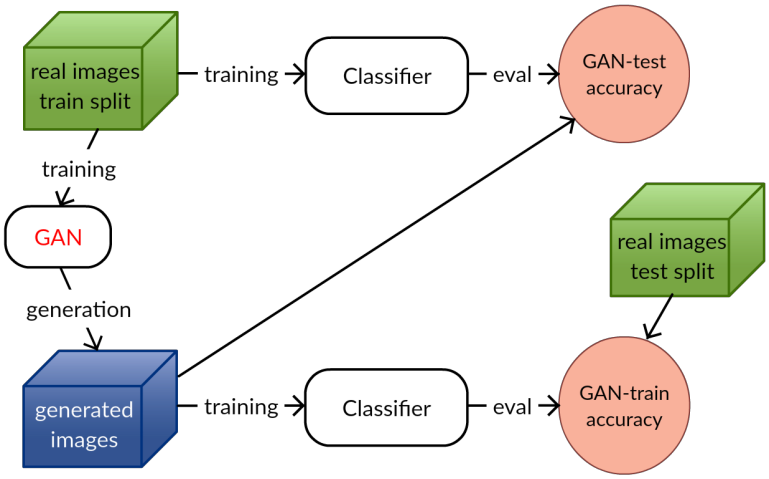

Figure 1.4 – Illustration of GAN-train and GAN-test. GAN-train learns a classifier on GAN generated images and measures the performance on real test images. This evaluates the diversity and realism of GAN images. GAN-test learns a classifier on real images and evaluates it on GAN images. This measures how realistic GAN images are.

Incremental learning without memory turned out to be a quite challenging problem in object detection. The setup we proposed in Chapter 3 is not easily adaptable to image classification. Our pipeline relies on co-occurring instances of objects, which are common in object detection datasets and virtually absent in image classification datasets. Generative models present an alternative to a memory buffer because they can learn a distribution, and then generate new samples for replay. GANs produce high quality samples, so they are a natural choice for this application. In order to find the most appropriate GAN model for this, we first need to be a able to evaluate them. Such an evaluation of GANs is a difficult problem. In particular, the extent of mode-collapse is a very important part of generative model performance, but it is hard to evaluate this quantitatively. Common evaluation methods consider unsupervised GANs and focus mostly on image quality. Even measures more sensitive to mode-dropping (FID) do not give a breakdown between quality and diversity.

We propose new evaluation measures to compare class-conditional GAN architectures with GAN-train and GAN-test scores. We rely on a neural net architecture

for image classification for both these measures. To compute GAN-train, we train a classification network (bottom classifier in Figure 1.4) with images generated by a GAN, and then evaluate its performance on a test set composed of real-world images. Intuitively, this measures the difference between the learned (i.e., generated image) and the target (i.e., real image) distributions. We can conclude that generated images are similar to real ones if the classification network, which learns features for discriminating images generated for different classes, can correctly classify real images. In other words, GAN-train is akin to a recall measure, as a good GAN-train performance shows that the generated samples are diverse enough. However, GAN-train also requires a sufficient precision, i.e., sample quality, which impacts the classifier.

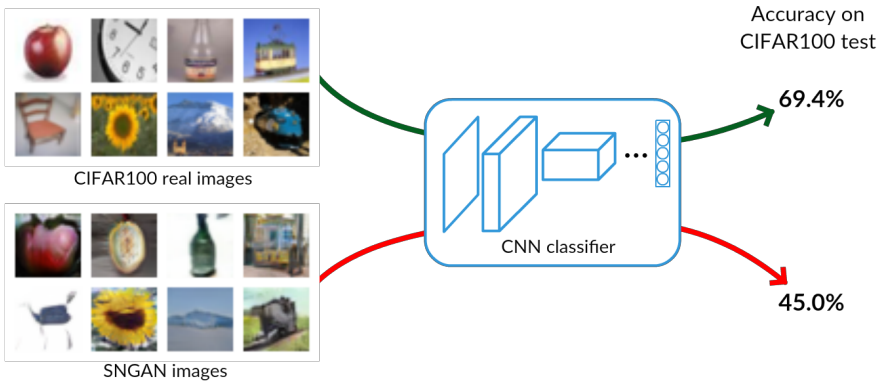

Figure 1.5 – State-of-the-art GANs, e.g., SNGAN [115], generate realistic images, which are difficult to evaluate subjectively in comparison to real images. Our new image classification accuracy-based measure (GAN-train is shown here) overcomes this issue, showing a clear difference between real and generated images.

Our second measure, GAN-test, is the accuracy of a network trained on real images and evaluated on the generated images (top classifier in Figure 1.5). This measure is similar to precision, with a high value denoting that the generated samples are a realistic approximation of the (unknown) distribution of natural images. In addition to these two measures, we study the utility of images generated by GANs for augmenting training data. This can be interpreted as a measure of the diversity of the generated images. The utility of our evaluation approach, in particular, when a subjective inspection is insufficient, is illustrated with the GAN-train measure in Figure 1.5.

As shown in our extensive experimental results in Section 5.3, these measures are much more informative to evaluate GANs, compared to all the previous measures discussed, including cases where human studies are inconclusive. In particular, we evaluate two state-of-the-art GAN models: WGAN-GP [64] and SNGAN [115],

along with other generative models [141,159] to provide baseline comparisons. Image classification performance is evaluated on MNIST [92], CIFAR10, CIFAR100 [87], and the ImageNet [155] datasets. Experimental results show that the quality of GAN images decreases significantly as the complexity of the dataset increases.

This work was published in ECCV'18 [163] and is presented in Chapter 5.

1.2.3 Adversarial training of partially invertible variational autoencoders

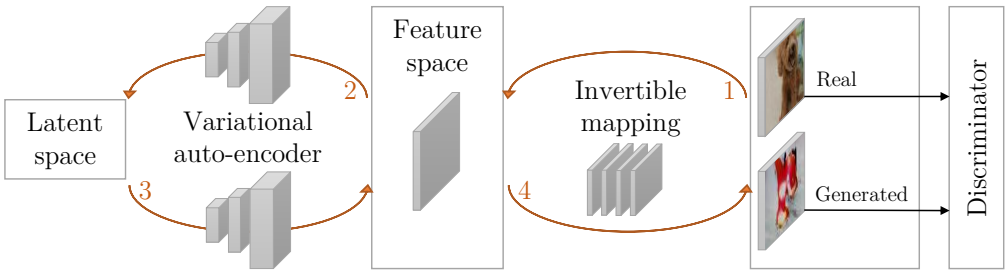

Figure 1.6 – Overview of our model: An invertible non-linear mapping f maps an image x to a feature space with the same dimension (arrows 1 and 4). The encoder takes $f(x)$ and maps it to a posterior distribution $q_\phi(z|x)$ over the latent variable (arrow 2), while the decoder maps z to a distribution over the feature space (arrow 3). The discriminator $D(x)$ assesses sample quality in the image space.

As discussed above, a generative model can be used instead of memory to implement incremental learning in image classification. It requires, however, a model generating high quality samples with diversity comparable to the original data. Modern generative models are typically divided into several classes: GANs, variational autoencoders (VAEs) and flow-based models (e.g., Real-NVP [38]). VAEs generate bad samples on natural images while GANs achieve good enough quality, but suffer from mode collapse. It is desirable to modify the procedure for training GANs to explicitly penalize mode dropping, and enforce full coverage of the dataset, while maintaining GAN image quality. We propose a model that uses adversarial mechanisms to explicitly discourage over-generalization together with maximum-likelihood training to avoid mode collapse. Our model (illustrated in Figure 1.6) is a novel extension of VAEs that uses invertible transformations close to the output, thus avoiding naive independence assumptions on pixels given the latent variables (which are typical in VAEs). As compared to Real-NVP, our model is computationally much more efficient, since most of the model consists of common non-invertible feed-forward layers.

To experimentally validate our coverage and quality driven training procedure and the different components of our model, we conduct a comprehensive and exhaustive set of experiments on the CIFAR-10 natural image dataset. We evaluate our models using likelihood scores on held-out training data, as well as Inception score (IS) and Fréchet Inception distance (FID). Our models obtain likelihood scores competitive with the state-of-the-art likelihood-based models, while achieving at a sample quality typical of GANs. We provide additional qualitative and quantitative experimental results on the CelebA dataset, STL-10, ImageNet, and LSUN-Bedrooms, that further confirm the observations made on CIFAR-10.

This work is submitted to ICML'19 and is presented in Chapter 6.

1.2.4 Outline of the thesis

The thesis is split into two parts. Part I is dedicated to incremental learning. We present related work on catastrophic forgetting and incremental learning in Chapter 2. Then we explore how to incrementally learn new classes in an object detection setup in Chapter 3. We do not store old training data which is a challenging task in incremental learning because most of the existing solutions are based on replay mechanism. Further in Part II, we discuss generative models as type of memory. Related work on generative models (in particular, GANs and VAEs) and their evaluation methods is presented in Chapter 4. In Chapter 5 we present a new way to evaluate conditional GANs, and quantify one of the major problems of GANs: "mode-dropping". It is necessary to cover all the modes of a data distribution to succeed in incremental learning supported by generative models. So, in Chapter 6 we propose a new generative model combining the high image quality of GANs with theoretical guarantees of data coverage, enjoyed by models directly optimizing maximum likelihood. In Appendix A we list several publications made while working on this thesis. In Appendix B we provide links to the released source code accompanying these publications. In Appendix C we present a real-time framework for simultaneous object detection and semantic segmentation.

Part I

Incremental learning in object detection

Incremental (or lifelong) learning addresses situations where a learner encounters different tasks in a form of stream [181]. It should master them sequentially and preferably exploit previously obtained knowledge to learn recent tasks faster. It is a metacognitive skill, so tasks can be arbitrary different from each other just like humans can learn vastly different skills over their life. In practice, it is a very broad definition involving a complex evaluation. Despite being studied for decades, incremental learning still represents a problem without off-the-shelf solutions. In computer vision it is often evaluated as a series of essentially the same task applied to different datasets. Implicit assumption here is that tasks are approximately independent and do not influence each other.

A more practical definition would be a gradual extension of tasks when a current one includes a previous task as an easier case. For example, in image classification it is a setup when every task adds new classes while keeping old ones intact. This way a learner recognizes more and more classes with every task. Note that it obviously violates independence assumption. An important subtlety here is how much training data we keep for every task: one can accumulate more and more data [32] or keep a small subset for each task [144] or not use old data at all.

In deep learning naive solutions to incremental learning fail because of catastrophic forgetting [112]: neural networks are not able to retain knowledge without constant reinforcement during the whole training time. They forget in a fundamentally different way than humans do and require special mechanisms to enable consolidation of knowledge, even in the simplest incremental learning setup with series of independent tasks.

The rest of this part is organized as follows: in Chapter 2 we review the related work on catastrophic forgetting and incremental learning, describe in detail interconnections and differences between these topics. In Chapter 3 we introduce our method to perform incremental learning in object detection task and present extensive experimental validation on PASCAL VOC 2007 dataset.

Chapter 2

Related work

Contents

2.1 Catastrophic forgetting **20**
2.1.1 Architectural methods 21
2.1.2 Regularization approaches 21
2.1.3 Knowledge distillation 22
2.2 Class-incremental learning **23**

The problem of incremental learning has a long history in machine learning and artificial intelligence [26, 137, 161, 181]. One of the main goals of AI research is to create autonomous agents capable to adapt to constantly drifting distribution of the real world rather than frozen models trained on carefully curated datasets for very specific problems.

Incremental learning has a few subtly different definitions in the literature [77, 80, 133]. It can be interpreted as learning *multiple independent tasks sequentially* or when the main objective evolves over time, typically becoming more challenging. Either way, sequential training of neural networks is complicated by *catastrophic forgetting* (or interference): when the network's training objective changes, it quickly forgets previously learned knowledge [51, 60, 112].

It is not always the case we are interested in keeping performance on old tasks intact. In transfer learning and domain adaptation goals are different: previous task is a way to improve performance on a given task (typically lacking enough training data) through first training a network on some other task. Transfer learning uses knowledge acquired from one task to help learn another [194]. Domain adaptation

transfers the knowledge acquired for a task from a data distribution to other (but related) data [47]. Fine-tuning [57], a very common type of transfer learning in computer vision, is way to alter the output layer of the original network for a new task and to tune all the parameters of the network for a new objective. CNNs learned for image classification [88] are often used to train other vision tasks such as object detection [57, 130, 194] and semantic segmentation [27]. Overall, existing transfer learning and domain adaptation methods require at least unlabeled data for both the tasks or domains, and in its absence, the new network quickly forgets all the knowledge acquired in the source domain [51, 60, 112, 143].

If we consider the image classification problem, then sequential multi-task learning would correspond to classification on many datasets with disjoint sets of classes learned one after another. On the other hand, *class-incremental* learning assumes that the set of classes is occasionally extended and new data is added. Even more extreme version requires to only use new data. We are going to review existing solutions for catastrophic forgetting problem and discuss their potential in class-incremental setup.

2.1 Catastrophic forgetting

Catastrophic forgetting is the predisposition of neural networks to quickly forget already learned knowledge upon learning new information. This phenomenon is believed to be caused by two factors [50, 91]. First, the internal representations in hidden layers are often overlapping, and a small change in a single neuron can affect multiple representations at the same time [50]. Second, all the parameters in feedforward networks are involved in computations for every data point, and a backpropagation update affects all of them in each training step [91]. The problem of addressing these issues in neural networks has its origin in classical connectionist networks several years ago [50, 51, 112].

Recent approaches specifically targeted against catastrophic forgetting are evaluated on sequential multi-task learning, typically on permuted MNIST (all image pixels are permuted according to a fixed permutation, which generates a different dataset with the same classification objective) or on multiple image classification datasets.

These methods can be divided in two groups: architectural and regularization approaches. Architectural methods change the network architecture to accommodate for new knowledge. Regularization methods add another term in the training objective that encourages the network to remember previously learned knowledge.

2.1.1 Architectural methods

Architectural approaches include growing the capacity of the network with new layers [157] or splitting a big network into independent blocks a sequence of which is optimized for any given task [48]. The downside of these approaches is the rapid increase in the number of new parameters to be learned. Besides, both of them require adding another head to network output which makes them unsuitable for class-incremental learning.

Another solution is to employ binary masks to selectively turn off certain weights [108]: idea inspired by weight pruning techniques for model compression. Masked out weights can be retrained for a new objective and then pruned again to accommodate for another task. While it allows to squeeze multiple classification tasks in a single network, it can not be used to infer outputs for all of them in parallel. Therefore, it can not be used for class-incremental learning.

2.1.2 Regularization approaches

Strong per-parameter regularization penalizes changes in weights relevant for a given task. Such approach called elastic weight consolidation (EWC) was developed in [85]. Answering the question "which parameters should not be changed much" in full generality seems to be intractable for modern neural networks, however various relaxations were suggested recently [7, 197]. Alternative to regularization loss is Gradient Episodic Memory [102]. This approach modifies gradients using inner optimization loop to prevent them from affecting important parameters.

EWC is a weighted L_2-penalty selectively penalizing big changes in certain weights. Suppose we trained a network with a loss $\mathcal{L}_A$ for task A and θ^*_A are the weights learned at the end of training of A. Then we want to approximate a conditional posterior on weights $p(\theta|\mathcal{D}_A)$ given the dataset $\mathcal{D}_A$. In general case, it is intractable: EWC approximates it as a Gaussian distribution with mean θ^*_A and diagonal variance given by the diagonal of Fisher information matrix F. So, coefficients F_i are basically second derivatives of the loss near a minimum (when the network converged on task A) and thus can be computed efficiently. Then the total loss to train on task B without forgetting task A is:

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \frac{\lambda}{2} \sum_i F_i(\theta_i - \theta^*_{A,i})^2, \tag{2.1}$$

where $\mathcal{L}_B(\theta)$ is the loss of task B.

In [197] per-parameter regularization coefficients are estimated online during SGD optimization rather than computed at convergence. In [7] these coefficients are computed from gradients of network's outputs rather than the loss function (as in EWC).

Gradient Episodic Memory (GEM) is quite different: for every task it stores a subset of examples which provide a number of proxy losses. It allows to evaluate if an updated network obtains worse loss on previous tasks. The idea of GEM is to project gradients on every update onto subspace of updates that do not degrade proxy losses of previous tasks. It enables backward knowledge transfer (performance on old tasks may become better). However, it introduces inner loop of quadratic optimization for each gradient descent step which slows training down by an order of magnitude. Quite unsurprisingly, performance of GEM depends on the size of memory and reaches performance of joint multi-task training when the memory size is close to training set size.

Overall, regularization approaches are developed for multi-task learning and do not perform well in class-incremental learning.

2.1.3 Knowledge distillation

An alternative to per-parameter regularization is to constrain network activations in order to cancel out undesired gradients and prevent representation shift when it can be avoided. It is typically achieved via knowledge distillation [23, 74]. This was originally proposed to transfer knowledge between different neural networks—from a large network to a smaller one for efficient deployment. The method in [74] encouraged the large (old) and the small (new) networks to produce similar responses. It has found several applications in domain adaptation and model compression [66, 156, 185].

Typically a small network is trained with usual cross-entropy loss on soft targets q_i produced by a large one

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \tag{2.2}$$

where z_i are logits of the large network and $T \geq 1$ is softmax temperature to smooth logit distribution. Alternatively, distillation can be represented as L_2-loss on logits.

Li and Hoiem [96] use knowledge distillation for one of the classical vision tasks, image classification, formulated in a deep learning framework. However, their evaluation is limited to the case where the old network is trained on a dataset, while the new network is trained on a different one, e.g., Places365 for the old and PASCAL VOC for the new, ImageNet for the old and PASCAL VOC for the new, etc. It is clearly an example of sequential multi-task learning because (i) different datasets often contain dissimilar classes, (ii) there is little confusion between datasets—it is in fact possible to identify a dataset simply from an image [183].

Our method is significantly different from [96] in two ways. First, we deal with the more difficult problem of learning incrementally on the same dataset, i.e., the

addition of classes to the network. As shown in [144], [96] fails in a similar setting of learning image classifiers incrementally. Second, we address the object detection task, where it is very common for the old and the new classes to co-occur, unlike the classification task.

2.2 Class-incremental learning

Class-incremental learning introduces new challenges besides catastrophic forgetting: it is not enough to just freeze network activations for old classes, we have to train the network to recognize new classes: *discover new representations capable to discriminate new classes against old ones as well as new classes between themselves*. It could be represented as multi-task learning (classification among old classes and classification among new classes), but it would require an oracle at test time to guess if an image belongs to old or new classes.

In object detection all objects not included in initial set of classes should be recognized as background. It makes it very close to open set recognition we discussed above. A lot of objects present on images from PASCAL VOC 2007 or COCO are not annotated and thus should be considered background. To enable incremental learning, the network should learn distinctive features of new objects and change decision boundary of background class to create a space for new ones. It is even more difficult challenge than catastrophic forgetting. In close set recognition (classical image classification task) there is no dedicated background class. New classes can be recognized as any of old classes which forces us to alter decision boundaries for all old classes. *This is why none of the methods described above work for class-incremental learning [80].*

Some of the more recent work, e.g., [32, 39], focuses on continuously updating the training set with data acquired from the Internet. They are: (i) restricted to learning with a fixed data representation [39], or (ii) keep all the collected data to retrain the model [32]. Other work partially addresses these issues by learning classifiers without access to the ensemble of data [113, 151], but uses a fixed image representation.

Rebuffi *et al.* [144] address some of the drawbacks in [96] with their incremental learning approach for image classification (iCaRL). They also use knowledge distillation, but decouple the classifier and the representation learning on top of heavily engineered framework to select and process samples for the memory. More precisely, nearest mean-of-exemplars over stored in memory examples is used as classifier. Representations are then updated in a separate step via auxillary sigmoid classification layer and composite loss (binary cross-entropy for new classes and knowledge distillation for old ones). After that, examplar set is recomputed to include new classes.

Very recent extension [25] of iCaRL trains classifier and representation learning end-to-end. Some of non-standard iCaRL elements (exemplar selection using herding and Nearest Exemplar Mean) are removed in [77] without any performance loss: it looks like exemplars can be chosen randomly and a softmax classifier can be used when distillation loss is unbiased. Our approach is aimed at learning the representation and classifiers jointly, without storing any of original training examples to avoid catastrophic forgetting. We use a standard end-to-end framework to solve object detection rather than non-standard pipeline in image classification.

A lot of class-incremental learning frameworks (including ours and [144]) are inspired by biological models of human memory [111]: the brain contains the hippocampus and the neocortex. The hippocampus is believed to be a quick learner prone to forgetting just like artificial neural networks. The neocortex, however, regularly consolidates recent knowledge learned by hippocampus and integrates it with prior information. This dual network paradigm fits well with knowledge distillation which describes exactly a way to transfer knowledge from one network to another.

Early connectionist research [9, 52] established that regular replay (rehearsal) of old training samples helps to mitigate catastrophic forgetting, yet still leaving a noticeable gap with offline models. Besides, it requires to keep all training data which does not scale well on modern datasets. Possible workaround is to select a small subset of representative images to use them for replay [144]. Another way is to draw these samples from a probabilistic model, so-called "pseudorehearsal" [152]. Recently this method was applied to image classification CNN where a GAN is used as a generator of pseudosamples [162]. However, natural image generation is not exactly an easy task, especially not for object detection datasets. Besides, incremental learning of generative models is even less explored area of research.

However, in [192] it was demonstrated that pseudorehearsal works for GANs and arguably even better than for discriminative models. Indeed, the main problem (covering all possible inputs) is circumvented because GANs uses random noise as input. It demonstrates that class-incremental learning for GANs is a practical idea.

FearNet [79] is another implementation of dual network system based on pseudorehearsal sampled from a simple generative model for embeddings. While it achieves impressive results in incremental setup, this work is not focused on representation learning and thus uses pre-trained on ImageNet embeddings as input. It makes FearNet effectively unable to learn representations over time.

In summary, none of the previous work addresses the problem of learning classifiers for object detection incrementally, without using previously seen training samples.

Chapter 3

Incremental learning of new classes

Contents

3.1 Object detection network **26**
3.2 Dual-network learning **27**
3.3 Sampling strategy **29**
3.4 Experiments **29**
3.4.1 Datasets and evaluation 29
3.4.2 Implementation details 29
3.4.3 Addition of one class 31
3.4.4 Addition of multiple classes 32
3.4.5 Sequential addition of multiple classes 34
3.5 Other alternatives **38**
3.6 Conclusion **38**

Our overall approach for incremental learning of a CNN model for object detection is illustrated in Figure 3.1. It contains a frozen copy of the original detector (denoted by Network A in the figure), which is used to: (i) select proposals corresponding to the old classes, i.e., distillation proposals, and (ii) compute the distillation loss. Network B in the figure is the adapted network for the new classes. It is obtained by increasing the number of outputs in the last layer of the original network, such that the new output layer includes the old as well as the new classes.

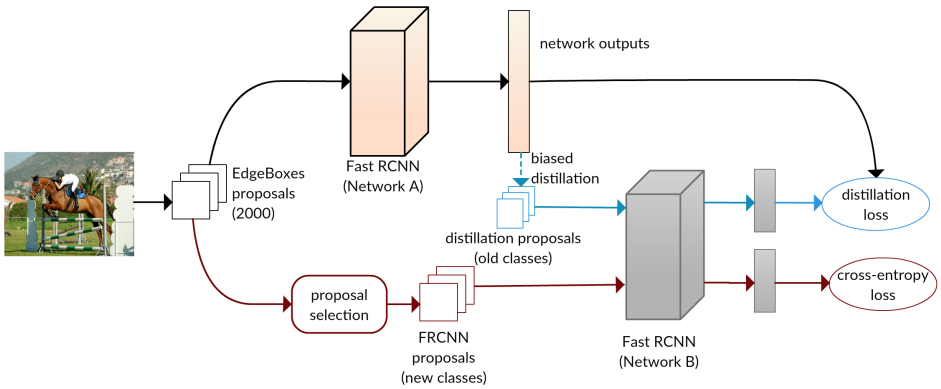

Figure 3.1 – Overview of our framework for learning object detectors incrementally. It is composed of a frozen copy of the detector (Network A) and the detector (Network B) adapted for the new class(es).

In order to avoid catastrophic forgetting, we constrain the learning process of the adapted network. We achieve this by incorporating a distillation loss, to preserve the performance on the old classes, as an additional term in the standard cross-entropy loss function (see Section 3.2). Specifically, we evaluate each new training sample on the frozen copy (Network A) to choose a diverse set of proposals (distillation proposals in Figure 3.1), and record their responses. With these responses in hand, we compute a distillation loss which measures the discrepancy between the two networks for the distillation proposals. This loss is added to the cross-entropy loss on the new classes to make up the loss function for training the adapted detection network. As we show in the experimental evaluation, the distillation loss as well as the strategy to select the distillation proposals are critical in preserving the performance on the old classes (see Section 3.4).

In the remainder of this section, we provide details of the object detector network (Section 3.1), the loss functions and the learning algorithm (Section 3.2), and strategies to sample the object proposals (Section 3.3).

3.1 Object detection network

We use a variant of a popular framework for object detection—Fast R-CNN [56], which is a proposal-based detection method built with pre-computed object proposals, e.g., [10, 203]. We chose this instead of the more recent Faster R-CNN [148], which integrates the computation of category-specific proposals into the network, because we need proposals agnostic to object categories, such as EdgeBoxes [203], MCG [10]. We use EdgeBoxes [203] proposals for PASCAL VOC 2007 and MCG [10]

for COCO. This allows us to focus on the problem of learning the representation and the classifier, given a pre-computed set of generic object proposals.

In our variant of Fast R-CNN, we replaced the VGG-16 trunk with a deeper ResNet-50 [70] component, which is faster and more accurate than VGG-16. We follow the suggestions in [70] to combine Fast R-CNN and ResNet architectures. The network processes the whole image through a sequence of residual blocks. Before the last strided convolution layer we insert a RoI pooling layer, which performs maxpooling over regions of varied sizes, i.e., proposals, into a 7×7 feature map. Then we add the remaining residual blocks, a layer for average pooling over spatial dimensions, and two fully connected layers: a softmax layer for classification (PASCAL or COCO classes, for example, along with the background class) and a regression layer for bounding box refinement, with independent corrections for each class.

The input to the network is an image and about 2000 precomputed object proposals represented as bounding boxes. During inference, the high-scoring proposals are refined according to bounding box regression. Then, a per-category non-maxima suppression (NMS) is performed to get the final detection results. The loss function to train the Fast R-CNN detector, corresponding to a RoI, is given by:

$$\mathcal{L}_{\text{rcnn}}(\mathbf{p}, k^*, t, t^*) = -\log \mathbf{p}_{k^*} + [k^* \geq 1] R(t - t^*), \tag{3.1}$$

where $\mathbf{p}$ is the set of responses of the network for all the classes (i.e., softmax output), k^* is a groundtruth class, t is an output of bounding box refinement layer, and $t*$ is the ground truth bounding box proposal. The first part of the loss denotes log-loss over classes, and the second part is localization loss. For more implementation details about Fast R-CNN, refer to the original paper [56].

3.2 Dual-network learning

First, we train a Fast R-CNN to detect the original set of classes C_A. We refer to this network as $\mathbf{A}(C_A)$. The goal now is to add a new set of classes C_B to this. We make two copies of $\mathbf{A}(C_A)$: one that is frozen to recognize classes C_A through distillation loss, and the second $\mathbf{B}(C_B)$ that is extended to detect the new classes C_B, which were not present or at least not annotated in the source images. The extension is done only in the last fully connected layers, i.e., classification and bounding box regression. We create sibling (i.e., fully-connected) layers [57] for new classes only and concatenate their outputs with the original ones. The new layers are initialized randomly in the same way as the corresponding layers in Fast R-CNN. Our goal is to train $\mathbf{B}(C_B)$ to recognize classes $C_A \cup C_B$ using only new data and annotations for C_B.

The distillation loss represents the idea of "keeping all the answers of the network the same or as close as possible". If we train $\mathbf{B}(C_B)$ without distillation, average precision on the old classes will degrade quickly, after a few hundred SGD iterations. This is a manifestation of catastrophic forgetting. We illustrate this in Section 3.4.3 and Section 3.4.4. We compute the distillation loss by applying the frozen copy of $\mathbf{A}(C_A)$ to any new image. Even if no object is detected by $\mathbf{A}(C_A)$, the unnormalized logits (softmax input) carry enough information to "distill" the knowledge of the old classes from $\mathbf{A}(C_A)$ to $\mathbf{B}(C_B)$. This process is illustrated in Figure 3.1.

For each image we randomly sample 64 RoIs out of 128 with the smallest background score. The logits computed for these RoIs by $\mathbf{A}(C_A)$ serve as targets for the old classes in the L_2 distillation loss shown below. The logits for the new classes C_B are not considered in this loss. We subtract the mean over the class dimension from these unnormalized logits (y) of each RoI to obtain the corresponding centered logits $\bar{y}$ used in the distillation loss. Bounding box regression outputs t_A (of the same set of proposals used for computing the logit loss) also constrain the loss of the network $\mathbf{B}(C_B)$. We chose to use L_2 loss instead of a cross-entropy loss for regression outputs because it demonstrates more stable training and performs better (see Section 3.4.4). The distillation loss combining the logits and regression outputs is written as:

$$\mathcal{L}_{\text{dist}}(y_A, t_A, y_B, t_B) = \frac{1}{N|C_A|} \sum \Big[(\bar{y}_A - \bar{y}_B)^2 + (t_A - t_B)^2 \Big], \tag{3.2}$$

where N is the number of RoIs sampled for distillation (i.e., 64 in this case), $|C_A|$ is the number of old classes, and the sum is over all the RoIs for the old classes. We distill logits without any smoothing, unlike [74], because most of the proposals already produce a smooth distribution of scores. Moreover, in our case, both the old and the new networks are similar with almost the same parameters (in the beginning), and so smoothing the logits distribution is not necessary to stabilize the learning.

The values of the bounding box regression are also distilled because we update all the layers, and any update of the convolutional layers will affect them indirectly. As box refinements are important to detect objects accurately, their values should be conserved as well. This is an easier task than keeping the classification scores because bounding box refinements for each class are independent, and are not linked by the softmax.

The overall loss $\mathcal{L}$ to train the model incrementally is a weighted sum of the distillation loss (3.2), and the standard Fast R-CNN loss (3.1) that is applied only to new classes C_B, where groundtruth bounding box annotation is available. In essence,

$$\mathcal{L} = \mathcal{L}_{\text{rcnn}} + \lambda \mathcal{L}_{\text{dist}}, \tag{3.3}$$

where the hyperparameter λ balances the two losses. We set λ to 1 in all the experiments with cross-validation (see Section 3.4.4).

The interplay between the two networks $\mathbf{A}(C_A)$ and $\mathbf{B}(C_B)$ provides the necessary supervision that prevents the catastrophic forgetting in the absence of original training data used by $\mathbf{A}(C_A)$. After the training of $\mathbf{B}(C_B)$ is completed, we can add more classes by freezing the newly trained network and using it for distillation. We can thus add new classes sequentially. Since $\mathbf{B}(C_B)$ is structurally identical to $\mathbf{A}(C_A \cup C_B)$, the extension can be repeated to add more classes.

3.3 Sampling strategy

As mentioned before, we choose 64 proposals out of 128 with the lowest background score, thus biasing the distillation to non-background proposals. We noticed that proposals recognized as confident background do not provide strong learning cues to conserve the original classes. One possibility is using an *unbiased distillation* that randomly samples 64 proposals out of the whole set of 2000 proposals. However, when doing so, the detection performance on old classes is noticeably worse because most of the distillation proposals are now background, and carry no strong signal about the object categories. Therefore, it is advantageous to select non-background proposals. We demonstrate this empirically in Section 3.4.5.

3.4 Experiments

3.4.1 Datasets and evaluation

We evaluate our method on the PASCAL VOC 2007 detection benchmark and the Microsoft COCO challenge dataset. VOC 2007 consists of 5K images in the trainval split and 5K images in the test split for 20 object classes. COCO on the other hand has 80K images in the training set and 40K images in the validation set for 80 object classes (which includes all the classes from VOC). We use the standard mean average precision (mAP) at 0.5 IoU threshold as the evaluation metric. We also report mAP weighted across different IoU from 0.5 to 0.95 on COCO, as recommended in the COCO challenge guidelines. Evaluation of the VOC 2007 experiments is done on the test split, while for COCO, we use the first 5000 images from the validation set.

3.4.2 Implementation details

We use SGD with Nesterov momentum [122] to train the network in all the experiments. We set the learning rate to 0.001, decay to 0.0001 after 30K iterations,

method	old	new	all
A(1-19)	68.4	-	-
+**B**(20) w/o distillation	25.0	52.1	26.4
+**B**(20) w frozen trunk	53.5	43.1	52.9
+**B**(20) w all layers frozen	69.1	41.6	66.6
+**B**(20) w frozen trunk and distill.	68.7	43.2	67.4
+**B**(20) w distillation	68.3	58.3	67.8
+**B**(20) w cross-entropy distill.	68.1	52.0	67.3
+**B**(20) w/o bbox distillation	68.5	62.7	68.3
A(1-20)	69.6	73.9	69.8

Table 3.1 – **VOC 2007 test** average precision (%). Experiments demonstrating the addition of "tvmonitor" class to a pretrained network under various setups. Classes 1-19 are the old classes, and "tvmonitor" (class 20) is the new one.

and momentum to 0.9. In the second stage of training, i.e., learning the extended network with new classes, we used a learning rate of 0.0001. The $\mathbf{A}(C_A)$ network is trained for 40K iterations on PASCAL VOC 2007 and for 400K iterations on COCO. The $\mathbf{B}(C_B)$ network is trained for 3K-5K iterations when only one class is added, and for the same number of iterations as $\mathbf{A}(C_A)$ when many classes are added at once. Following Fast R-CNN [56], we regularize with weight decay of 0.00005 and take batches of two images each. All the layers of $\mathbf{A}(C_A)$ and $\mathbf{B}(C_B)$ networks are finetuned unless stated otherwise.

The integration of ResNet into Fast R-CNN (see Section 3.1) is done by adding a RoI pooling layer before the *conv5_1* layer, and replacing the final classification layer by two sibling fully connected layers. The batch normalization layers are frozen, and as in Fast R-CNN, no dropout is used. RoIs are considered as detections if they have a score more than 0.5 for any of the classes. We apply per-class NMS with an IoU threshold of 0.3. Training is image-centric, and a batch is composed of 64 proposals per image, with 16 of them having an IoU of at least 0.5 with a groundtruth object. All the proposals are filtered to have IoU less than 0.7, as in [203].

We use TensorFlow [5] to develop our incremental learning framework. Each experiment begins with choosing a subset of classes to form the set C_A. Then, a network is learned only on the subset of the training set composed of all the images containing at least one object from C_A. Annotations for other classes in these images are ignored. With the new classes chosen to form the set C_B, we learn the extended network as described in Section 3.2 with the subset of the training set containing at least one object from C_B. As in the previous case, annotations of all the other classes, including those of the original classes C_A, are ignored.

method	old	new	all
A(1-10)	65.8	-	-
+**B**(11-20) w/o distillation	12.8	64.5	38.7
+**B**(11-20) w distillation	63.2	63.1	63.1
+**B**(11-20) w/o bbox distillation	58.7	63.1	60.9
+**B**(11-20) w EWC [85]	31.6	61.0	46.3
A(1-20)	68.4	71.3	69.8

Table 3.2 – **VOC 2007 test** average precision (%). Experiments demonstrating the addition of 10 classes, all at once, to a pretrained network. Classes 1-10 are the old classes, and 11-20 the new ones.

For computational efficiency, we precomputed the responses of the frozen network $\mathbf{A}(C_A)$ on the training data (as every image is typically used multiple times).

3.4.3 Addition of one class

In the first experiment we take 19 classes in alphabetical order from the VOC dataset as C_A, and the remaining one as the only new class C_B. We then train the **A**(1-19) network on the VOC trainval subset containing any of the 19 classes, and the **B**(20) network is trained on the trainval subset containing the new class. A summary of the evaluation of these networks on the VOC test set is shown in Table 3.1, with the full results in Table 3.6.

A baseline approach for addition of a new class is to add an output to the last layer and freeze the rest of the network. This freezing, where the weights of the network's convolutional layers are fixed ("**B**(20) w frozen trunk" in the tables), results in a lower performance on the new class as the previously learned representations have not been adapted for it. Furthermore, it does not prevent degradation of the performance on the old classes, where mAP drops by almost 15%. When we freeze all the layers, including the old output layer ("**B**(20) w all layers frozen"), or apply distillation loss ("**B**(20) w frozen trunk and distill."), the performance on the old classes is maintained, but that on the new class is poor. This shows that finetuning of convolutional layers is necessary to learn the new classes.

When the network **B**(20) is trained without the distillation loss ("**B**(20) w/o distillation" in the tables), it can learn the 20th class, but the performance decreases significantly on the other (old) classes. As seen in Table 3.6, the AP on classes like "cat", "person" drops by over 60%. The same training procedure with distillation loss largely alleviates this catastrophic forgetting. Without distillation, the new network has 25.0% mAP on the old classes compared to 68.3% with distillation,

and 69.6% mAP of baseline Fast R-CNN trained jointly on all classes ("A(1-20)"). With distillation the performance is similar to that of the old network **A**(1-19), but is lower for certain classes, e.g., "bottle". The 20th class "tvmonitor" does not get the full performance of the baseline (73.9%), with or without distillation, and is less than 60%. This is potentially due to the size of the training set. The **B**(20) network is trained only a few hundred images containing instances of this class. Thus, the "tvmonitor" classifier does not see the full diversity of negatives.

We also performed the "addition of one class" experiment with each of the VOC categories being the new class. The behavior for each class is very similar to the "tvmonitor" case described above. The mAP varies from 66.1% (for new class "sheep") to 68.3% ("tvmonitor") with mean 67.38% and standard deviation of 0.6%.

3.4.4 Addition of multiple classes

In this scenario we train the network **A**(1-10) on the first 10 VOC classes (in alphabetical order) with the VOC trainval subset corresponding to these classes. In the second stage of training we used the remaining 10 classes as C_B and trained only on the images containing the new classes. Table 3.2 shows a summary of the evaluation of these networks on the VOC test set, with the full results in Table 3.7.

Training the network **B**(11-20) on the 10 new classes with distillation (for the old classes) achieves 63.1% mAP ("**B**(11-20) w distillation" in the tables) compared to 69.8% of the baseline network trained on all the 20 classes ("**A**(1-20)"). Just as in the previous experiment of adding one class, performance on the new classes is slightly worse than with the joint training of all the classes. For example, as seen in Table 3.7, the performance for "person" is 73.2% vs 79.1%, and 72.5% vs 76.8% for the "train" class. The mAP on new classes is 63.1% for the network with distillation versus 71.3% for the jointly trained model. However, without distillation, the network achieves only 12.8% mAP ("+**B**(11-20) w/o distillation") on the old classes. Note that the method without bounding box distillation ("+**B**(11-20) w/o bbox distillation") is inferior to our full method ("+**B**(11-20) w distillation").

We also performed the 10-class experiment for different values of λ in (3.3), the hyperparameter controlling the relative importance of distillation and Fast R-CNN loss. Results shown in Figure 3.2 demonstrate that when the distillation is weak ($\lambda = 0.1$) the new classes are easier to learn, but the old ones are more easily forgotten. When distillation is strong ($\lambda = 10$), it destabilizes training and impedes learning the new classes. Setting λ to 1 is a good trade-off between learning new classes and preventing catastrophic forgetting.

We also compare our approach with elastic weight consolidation (EWC) [85], which is an alternative to distillation and applies per-parameter regularization selectively to alleviate catastrophic forgetting. We reimplemented EWC and verified

method	old	new	all
A(1-15)	70.5	-	-
+**B**(16-20) w distill.	68.4	58.4	65.9
+**B**(16)(17)...(20) w distill.	66.0	51.6	62.4
+**B**(16)(17)...(20) w unbiased distill.	45.8	46.5	46.0
+**A**(16)+...+**A**(20)	70.5	37.8	62.4
A(1-20)	70.9	66.7	69.8

Table 3.3 – **VOC 2007 test** average precision (%). Experiments demonstrating the addition of 5 classes, all at once, and incrementally to a pretrained network. Classes 1-15 are the old ones, and 16-20 the new classes.

method	mAP@.5	mAP@[.5, .95]
A(1-40)+**B**(41-80)	37.4	21.3
A(1-80)	38.1	22.6

Table 3.4 – **COCO minival** (first 5000 validation images) average precision (%). We compare the model learned incrementally on half the classes with the baseline trained on all jointly.

that it produces results comparable to those reported in [85] on MNIST, and then adapted it to our object detection task. We do this by using the Fast R-CNN batches during the training phase (as done in Section 3.4.2), and by replacing log loss with the Fast R-CNN loss. Our approach outperforms EWC for this case, when we add 10 classes at once, as shown in Table 3.2 and Table 3.7.

We evaluated the influence of the number of new classes in incremental learning. To this end, we learn a network for 15 classes first, and then train for the remaining 5 classes, all added at once on VOC. These results are summarized in Table 3.3, with the per-class results shown in Table 3.8. The network **B**(16-20) has better overall performance than **B**(11-20): 65.9% mAP versus 63.1% mAP. As in the experiment with 10 classes, the performance is lower for a few classes, e.g., "table", "horse", for example, than the initial model **A**(1-15). The performance on the new classes is lower than jointly trained baseline Fast R-CNN **A**(1-20). Overall, mAP of **B**(16-20) is lower than baseline Fast R-CNN (65.9% versus 69.8%).

The evaluation on COCO, shown in Table 3.4, is done with the first 40 classes in the initial set, and the remaining 40 in the new second stage. The network **B**(41-80) trained with the distillation loss obtains 37.4% mAP in the PASCAL-style metric and 21% mAP in the COCO-style metric. The baseline network trained on 80 classes is similar in performance with 38.1% and 22.6% mAP respectively. We observe that our proposed method overcomes catastrophic forgetting, just as in the

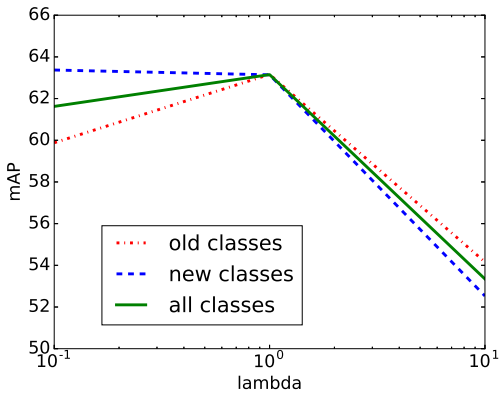

Figure 3.2 – The influence of λ, in the loss function (3.3), on the mAP performance for the **B**(11-20) network trained with distillation.

case of VOC seen earlier.

We also studied if distillation depends on the distribution of images used in this loss. To this end, we used the model **A**(1-10) trained on VOC, and then performed the second stage learning in two settings: **B**(11-20) learned on the subset of VOC as before, and another model trained for the same set of classes, but using a subset of COCO. From Table 3.5 we see that indeed, distillation works better when background samples have exactly the same distribution in both stages of training. However, it is still very effective even when the dataset in the second stage is different from the one used in the first.

3.4.5 Sequential addition of multiple classes

In order to evaluate incremental learning of classes added sequentially, we update the frozen copy of the network with the one learned with the new class, and then repeat the process with another new class. For example, we take a network learned for 15 classes of VOC, train it for the 16th on the subset containing only this class, and then use the 16-class network as the frozen copy to then learn the 17th class. This is then continued until the 20th class. We denote this incremental extension as **B**(16)(17)(18)(19)(20).

Results of adding classes sequentially are shown in Table 3.8 and Table 3.9. After adding the 5 classes we obtain 62.4% mAP (row 3 in Table 3.8), which is lower than 65.9% obtained by adding all the 5 classes at once (row 2). Table 3.9 shows intermediate evaluations after adding each class. We observe that the performance of the original classes remains stable at each step in most cases, but for a few classes, which is not recovered in the following steps. We empirically evaluate the

	+COCO-10cls	+VOC-10cls
mAP (old classes)	61.4	63.2
mAP (new classes)	48.2	63.1
mAP (all classes)	54.8	63.2

Table 3.5 – **VOC 2007 test** average precision (%). The second stage of training, where 10 classes (11-20th) are added, is done on the subset of COCO images (+COCO-10cls), and is compared to the one trained on the VOC subset (+VOC-10cls).

importance of using biased non-background proposals (cf. Section 3.3). Here we add the 5 classes one by one, but use unbiased distillation ("**B**(16)(17)(18)(19)(20) w unbiased distill." in Table 3.3 and Table 3.8), i.e., randomly sampled proposals are used for distillation. This results in much worse overall performance (46% vs 62.4%) and some classes ("person", "chair") suffer from a significant performance drop of 10-20%. We also performed sequential addition experiment with 10 classes, and present the results in Table 3.10. Although the drop in mAP is more significant than for the previous experiment with 5 classes, it is far from catastrophic forgetting.

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
A(1-19)	69.9	79.4	69.5	55.7	45.6	78.4	78.9	79.8	44.8	76.2	63.8	78.0	80.8	77.6	70.2	40.9	67.8	64.5	77.5	-	68.4
+**B**(20) w/o distillation	35.9	36.1	26.4	16.5	9.1	26.4	36.2	18.2	9.1	51.5	9.1	26.6	50.0	26.2	9.1	9.1	43.7	9.1	28.0	52.1	26.4
+**B**(20) w frozen trunk	61.3	71.9	62.5	46.2	34.5	70.6	71.6	62.4	9.1	68.3	27.1	61.6	80.0	70.6	35.9	24.6	53.8	34.9	68.9	43.1	52.9
+**B**(20) w all layers frozen	68.8	78.4	70.2	51.8	52.8	76.1	78.7	78.8	50.1	74.5	65.5	76.9	80.2	76.3	69.8	40.4	62.0	63.7	75.5	41.6	66.6
+**B**(20) w frozen trunk and distill.	74.4	78.1	69.8	54.7	52.1	75.7	79.0	78.5	48.5	74.4	62.3	77.0	80.2	77.2	69.7	44.5	68.6	64.5	74.7	43.2	67.4
+**B**(20) w distillation	70.2	79.3	69.6	56.4	40.7	78.5	78.8	80.5	45.0	75.7	64.1	77.8	80.8	78.0	70.4	42.3	67.6	64.6	77.5	58.3	67.8
+**B**(20) w cross-entropy distill.	69.1	79.1	69.5	52.8	45.4	78.1	78.9	79.5	44.8	75.5	64.2	77.2	80.8	77.9	70.2	42.7	66.8	64.6	76.1	52.0	67.3
+**B**(20) w/o bbox distillation	69.4	79.3	69.5	57.4	45.4	78.4	79.1	80.5	45.7	76.3	64.8	77.2	80.8	77.5	70.1	42.3	67.5	64.4	76.7	62.7	68.3
A(1-20)	70.2	77.9	70.4	54.1	47.4	78.9	78.6	79.8	50.8	75.9	65.6	78.0	80.5	79.1	76.3	47.7	69.3	65.6	76.8	73.9	69.8

Table 3.6 – **VOC 2007 test** per-class average precision (%) under different settings when the "tvmonitor" class is added.

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
A(1-10)	69.9	76.7	68.9	54.9	48.7	72.9	78.8	75.5	48.8	62.7	-	-	-	-	-	-	-	-	-	-	65.8
+**B**(11-20) w/o distillation	25.5	9.1	23.5	17.3	9.1	9.1	9.1	16.2	0.0	9.1	61.5	67.7	76.0	72.2	68.9	34.8	63.6	62.7	72.5	65.2	38.7
+**B**(11-20) w distillation	69.9	70.4	69.4	54.3	48.0	68.7	78.9	68.4	45.5	58.1	59.7	72.7	73.5	73.2	66.3	29.5	63.4	61.6	69.3	62.2	63.1
+**B**(11-20) w/o bbox distillation	68.8	69.8	60.6	46.4	46.7	65.9	71.3	66.3	43.6	47.3	58.5	70.6	73.4	70.6	66.3	33.6	63.1	62.1	69.4	63.1	60.9
+**B**(11-20) w EWC [85]	54.5	18.2	52.8	20.8	25.8	53.2	45.0	27.3	9.1	9.1	49.6	61.2	76.1	73.6	67.1	35.8	57.8	55.2	67.9	65.3	46.3
A(1-20)	70.2	77.9	70.4	54.1	47.4	78.9	78.6	79.8	50.8	75.9	65.6	78.0	80.5	79.1	76.3	47.7	69.3	65.6	76.8	73.9	69.8

Table 3.7 – **VOC 2007 test** per-class average precision (%) under different settings when 10 classes are added at once.

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
A(1-15)	70.8	79.1	69.8	59.2	53.3	76.9	79.3	79.1	47.8	70.0	62.0	76.6	80.4	77.5	76.2	-	-	-	-	-	70.5
+**B**(16-20) w distill.	70.5	79.2	68.8	59.1	53.2	75.4	79.4	78.8	46.6	59.4	59.0	75.8	71.8	78.6	69.6	33.7	61.5	63.1	71.7	62.2	65.9
+**B**(16)(17)(18)(19)(20) w distill.	70.0	78.1	61.0	50.9	46.3	76.0	78.8	77.2	46.1	66.6	58.9	67.7	71.6	71.4	69.6	25.6	57.1	46.5	70.7	58.2	62.4
+**B**(16)(17)(18)(19)(20) w unbiased distill.	62.2	71.2	52.3	43.8	24.9	60.7	62.9	53.4	9.1	34.9	42.5	34.8	54.3	70.9	9.1	18.7	53.2	48.9	58.2	53.5	46.0
A(1-20)	70.2	77.9	70.4	54.1	47.4	78.9	78.6	79.8	50.8	75.9	65.6	78.0	80.5	79.1	76.3	47.7	69.3	65.6	76.8	73.9	69.8

Table 3.8 – **VOC 2007 test** per-class average precision (%) under different settings when 5 classes are added at once or sequentially.

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
A(1-15)	70.8	79.1	69.8	59.2	53.3	76.9	79.3	79.1	47.8	70.0	62.0	76.6	80.4	77.5	76.2	-	-	-	-	-	70.5
+**B**(16)	70.5	78.3	69.6	60.4	52.4	76.8	79.4	79.2	47.1	70.2	56.7	77.0	80.3	78.1	70.0	26.3	-	-	-	-	67.0
+**B**(16)(17)	70.3	78.9	67.7	59.2	47.0	76.3	79.3	77.7	48.0	58.8	60.2	67.4	71.6	78.6	70.2	27.9	46.8	-	-	-	63.9
+**B**(16)(17)(18)	69.8	78.2	67.0	50.4	46.9	76.5	78.6	78.0	46.4	58.6	58.6	67.5	71.8	78.5	69.9	26.1	56.2	45.3	-	-	62.5
+**B**(16)(17)(18)(19)	70.4	78.8	67.3	49.8	46.4	75.6	78.4	78.0	46.0	59.5	59.2	67.2	71.8	71.3	69.8	25.9	56.1	48.2	65.0	-	62.4
+**B**(16)(17)(18)(19)(20)	70.0	78.1	61.0	50.9	46.3	76.0	78.8	77.2	46.1	66.6	58.9	67.7	71.6	71.4	69.6	25.6	57.1	46.5	70.7	58.2	62.4
A(1-20)	70.2	77.9	70.4	54.1	47.4	78.9	78.6	79.8	50.8	75.9	65.6	78.0	80.5	79.1	76.3	47.7	69.3	65.6	76.8	73.9	69.8

Table 3.9 – **VOC 2007 test** per-class average precision (%) when 5 classes are added sequentially.

method	**A**(1-10)	+table	+dog	+horse	+mbike	+persn	+plant	+sheep	+sofa	+train	+tv
mAP	67.1	65.1	62.5	59.9	59.8	59.2	57.3	49.1	49.8	48.7	49.0

Table 3.10 – **VOC 2007 test** average precision (%) when adding 10 classes sequentially. Unlike other tables each column here shows the mAP of a network trained on all the previous classes and the new class. For example, the mAP shown for "+dog" is the result of the network trained on the first ten classes, "table", and the new class "dog".

3.5 Other alternatives

Learning multiple networks. Another solution for learning multiple classes is to train a new network for each class, and then combine their detections. This is an expensive strategy at test time, as each network has to be run independently, including the extraction of features. This may seem like a reasonable thing to do as evaluation of object detection is done independently for each class, However, learning is usually not independent. Although we can learn a decent detection network for 10 classes, it is much more difficult when learning single classes independently. To demonstrate this, we trained a network for 1-15 classes and then separate networks for each of the 16-20 classes. This results in 6 networks in total (row "+**A**(16)+...+**A**(20)" in Table 3.3), compared to incremental learning of 5 classes implemented with a single network ("+**B**(16)(17)...(20) w distill."). The results confirm that new classes are difficult to learn in isolation.

Varying distillation loss. As noted in [74], knowledge distillation can also be expressed as a cross-entropy loss. We compared this with L_2-based loss on the one class extension experiment ("**B**(20) w cross-entropy distill." in Table 3.1 and Table 3.6). Cross-entropy distillation works as well as L_2 distillation keeping old classes intact (67.3% vs 67.8%), but performs worse than L_2 on the new class "tvmonitor" (52% vs 58.3%). We also observed that cross-entropy is more sensitive to the training schedule. According to [74], both formulations should be equivalent in the limit of a high smoothing factor for logits (cf. Section 3.2), but our choice of not smoothing leads to this different behavior.

Bounding box regression distillation. Addition of 10 classes (Table 3.2) without distilling bounding box regression values performs consistently worse than the full distillation loss. Overall **B**(11-20) without distilling bounding box regression gets 60.9% vs 63.1% with the full distillation. However, on a few new classes the performance can be higher than with the full distillation (Table 3.7). This is also the case for **B**(20) without bounding box distillation (Table 3.6) that has better performance on "tvmonitor" (62.7% vs 58.3%). This is not the case when other categories are chosen as the new class. Indeed, bounding box distillation shows an improvement of 2% for the "sheep" class.

3.6 Conclusion

In this chapter, we have presented an approach for incremental learning of object detectors for new classes, without access to the training data corresponding to the old classes. We address the problem of catastrophic forgetting in this context, with

a loss function that optimizes the performance on the new classes, in addition to preserving the performance on the old classes. Our extensive experimental analysis demonstrates that our approach performs well, even in the extreme case of adding new classes one by one.

Part II

Generative models

As we discussed in Chapter 2, a lot of frameworks for incremental learning rely on a form of knowledge distillation to prevent catastrophic forgetting. This requires some samples to distill on. One way to get them is just to stash all (or subset of) training images from the past. However, this approach does not scale well; size of training set quickly becomes an issue even in cloud computing, let alone any embodied agent we may imagine. Another way is to generate them artificially or mine from incoming data.

It is interesting to note that we do not have to use real images for distillation. Samples have to be close enough to real data and cover it well. Knowledge distillation can be compared to polynomial interpolation: given enough of input-output pairs we can exactly recover the original function even if these pairs were never used to fit it in the first place. This is essentially the idea behind "pseudorehearsals" [9, 52, 152] that were initially suggested to fight catastrophic forgetting in fully connected shallow networks with low-dimensional inputs. In this situation they could be sampled from a very simple probabilistic model. Although this idea should in principle work in convolutional networks applied to natural images, sampling random natural images is more challenging task.

Generative models offer exactly this: random natural images. Currently GANs demonstrate the best quality of images, they are easy to condition on class labels (after all, in order to use knowledge distillation we want not just any random image, but image of a particular class learned before). Recent work [162] shows that GANs can be extended dynamically and generated samples indeed help with incremental learning. Even more effective method to dynamically extend conditional GANs was suggested in [192]. These studies suggest that GANs represent an effective method to generate "pseudorehearsals".

However, there is still a significant gap between data a GAN can be trained to generate and data an ordinary discriminative CNN can learn. Besides, GANs suffer from mode-dropping, are notoriously difficult to train and to evaluate correctly. All these problems contribute to the gap between generative and discriminative models.

The rest of this part is organized as follows: in Chapter 4 we review the related work on generative models with a particular focus on GANs and their issues and then discuss common ways to evaluate them. In Chapter 5 we suggest a new measure to evaluate conditional GANs and compare it to accepted GAN metrics. Further in Chapter 6 we propose a new generative model elegantly combining properties of adversarial models as well as coverage guarantees of likelihood-based models. We demonstrate extensive evaluation of our model on CIFAR-10 and STL-10 and show generated samples on CelebA and LSUN.

Application of generative models to incremental learning is left for future work.

Chapter 4

Related work

Contents

4.1 Coverage-driven training and maximum likelihood estimation . **46**
4.1.1 Variational autoencoders 46
4.1.2 Autoregressive models 47
4.1.3 Flow-based models 48
4.2 Adversarial models and quality-driven training **48**
4.3 Hybrid approaches **50**
4.4 Conditioning of generative models **51**
4.5 Evaluation of GAN models **51**
4.5.1 Inception score 52
4.5.2 Fréchet Inception distance 52
4.5.3 Other evaluation measures 53
4.6 Data augmentation with GANs **53**

The classic and most common approach to training generative models is to make sure that they cover the training data sampled from an unknown distribution p^* well. Using a score $s_c(x, p_\theta)$ that evaluates how well a sample x is covered by the model p_θ, training proceeds by maximizing the objective $\mathcal{L}_C(p_\theta) = \int_{x \in X} p^*(x) s_c(x, p_\theta) \mathrm{d}x$. We refer to this training procedure as *coverage-driven training* (CDT). The other approach to generative modeling is to make sure that samples from the model fit the distribution p^* underlying the training data. Given a score $s_q(x, p^*)$ that evaluates how plausible a sample x is under p^*, a model p_θ can be trained by

maximizing $\mathcal{L}_Q(p_\theta) = \int_{x \in X} p_\theta(x) s_q(x, p^*) \mathrm{d}x$. We refer to this procedure as *quality-driven training* (QDT). In practice p^* is unavailable, and the integral over p^* is therefore approximated using the empirical average over the training data. In what follows we argue that well-recognized failure modes of modern generative models, at least partially, stem from the choice of one of these procedures.

4.1 Coverage-driven training and maximum likelihood estimation

Among coverage-driven training methods, maximum likelihood estimation (MLE) is the most common. It maximizes the probability of data observed from p^* under p_θ w.r.t. θ, the parameter vector of the model, using the log-score $\mathcal{L}_C(p_\theta) = \mathbb{E}_{x \sim p^*}[\log p_\theta(x)]$. This is equivalent to minimizing $\mathrm{D}_{\mathrm{KL}}(p^* \mid\mid p_\theta)$, the Kullback-Liebler (KL) divergence between p^* and p_θ. This yields models that typically cover all the modes of the data, but tend to put mass in spurious regions of the target space; a phenomenon known as *over-generalization* [20], and manifested by unrealistic samples in the context of generative image models. Intuitively, all the modes of the data are well covered because p_θ is explicitly optimized to cover all the samples from p^*. Conversely, what does not happen is sampling x from p_θ and assessing its quality, ideally using the inaccessible $p^*(x)$ as a score. put mass in spurious regions of the space without being heavily penalized.

Putting mass on samples from p^* takes it away from spurious regions, so in principle an infinitely flexible model with infinite training data should fit p^* without evaluating it on samples from p_θ. Assuming p^* is more complex than what p_θ can model, however, covering all the modes of p^* will push p_θ to assign mass to regions of low probability under p^*. Even if p_θ were infinitely expressive, given a finite dataset it would over-fit by putting Diracs on all the training samples. The model is thus *forced to* put mass on points that are not in the training set, but we cannot control where this occurs, since p^* cannot be evaluated. So regularization is necessary: if a simple model explains the data well it should generalize and spill some mass in the right places. Using a held-out validation or test set, we can assess if the model generalizes and assigns mass to the held-out samples from p^*, but not how much has been spilled elsewhere. Therefore, over-generalization is an inherent problem for MLE-based models, and having a mechanism that drives quality as well as coverage in the learned distribution is therefore desirable.

4.1.1 Variational autoencoders

Variational autoencoders (VAEs) [84, 150] is a class of deep generative latent variables models optimizing likelihood. VAE consists of two neural networks:

encoder and decoder. Decoder network implements a distribution $p_\theta(x|z)$ over observations x given latent variables z and has parameters θ. Typically z has unit Gaussian as a basic prior $p(z)$. Then the generative model can be obtained via marginalization of latent variables:

$$p_\theta(x) = \int p(z)p_\theta(x|z)dz \tag{4.1}$$

However, this integral is intractable thus making direct optimization of marginal likelihood impossible. Encoder network approximates a posterior distribution $q_\phi(z|x)$ and has parameters ϕ. It helps to define a variational lower bound on data log-likelihood:

$$\begin{aligned} \log p_\theta(x) \geq \mathrm{ELBO}(x, \theta, \phi) &= \log p_\theta(x) - \mathrm{D_{KL}}(q_\phi(z|x) \mid\mid p_\theta(x|z)) \\ &= \mathop{\mathbb{E}}_{z\sim q_\phi}[\log p_\theta(x|z)] - \mathrm{D_{KL}}(q_\phi(z|x) \mid\mid p(z)) \end{aligned} \tag{4.2}$$

The first term of this formula is reconstruction loss maximizing data log-likelihood $p_\theta(x|z)$ under posterior estimate $q_\phi(z|x)$. The second term can be seen as regularization of the posterior q_ϕ to be close to the prior $p(z)$.

To achieve efficient sampling VAE assumes pixels to be conditionally independent:

$$p_\theta(x|z) = \prod_{i=1}^{N} p(x_i|z) \tag{4.3}$$

This implies that all low-level variability should be handled by latent variables z. In the end it makes VAE samples very blurry.

4.1.2 Autoregressive models

Autoregressive models [55, 89] address the main drawback of VAEs: instead of conditional independence they factorize multivariate distribution sequentially,

$$p_\theta(x) = \prod_{i=1}^{N} p(x_i|x_{<i}) \tag{4.4}$$

where $x_{<i} = x_1, \dots x_{i-1}$. Conditional distributions are modeled by a deep neural network.

In computer vision autoregressive models usually use scan line pixel ordering. The most successful model (by obtained log-likelihood on held-out data) is PixelCNN++ [159] which integrates a lot of improvements on top of PixelRNN [188]. It achieves scan line ordering with convolutional layers via masking applied when needed.

However, autoregressive factorization comes with a price: autoregressive models are frustratingly slow to sample and thus hardly scale to high resolution. Even though clever caching of intermediate computations [142] can accelerate inference significantly, the speed is still far behind both GANs and VAEs. Another way is to partially relax autoregressive nature and render some groups of pixels independently [147].

4.1.3 Flow-based models

Flow-based model [37] is defined as follows:

$$z_0 \sim p(z_0) \qquad z_t = f_t(z_{t-1}) \quad \forall t = 1 \dots T \qquad z_T = x \tag{4.5}$$

where $p(z_0)$ is a simple tractable distribution, usually spherical multivariate Gaussian distribution $\mathcal{N}(0; \mathbf{I})$ and each of f_t is an invertible function (bijection). Such sequence of f_t is called a *normalizing flow* [149]. Assuming that f_t have known Jacobian, probability density function of z_T can be computed via change of variable rule:

$$\log p(z_T) = \log p(z_0) + \sum_{t=1}^{T} \log \left| \det \frac{\mathrm{d}z_t}{\mathrm{d}z_{t-1}} \right| \tag{4.6}$$

This way log-likelihood can be computed exactly, encoding and decoding is essentially done by a single neural network (due to bijective nature of f_t).

Normalizing flow was used in Inverse Autoregressive Flow (IAF) [83] to integrate autoregressive decomposition into VAE. It made reconstructions crisp and allowed VAE to achieve likelihood competitive with autoregressive models. In general, normalizing flow can be seen as a way to learn more flexible posteriors.

RealNVP [38] uses a rich family of functions to choose f_t from and generates quite good samples (quality is still worse than GAN samples though). Likelihood, on the other hand, was noticeably lower than autoregressive models. Very recently a successor of RealNVP called Glow [82] has reached competitive likelihood numbers albeit at a cost of huge networks and thus slower inference.

4.2 Adversarial models and quality-driven training

We have argued that models trained with maximum likelihood would benefit from having a mechanism that samples $x \sim p_\theta$ and evaluate it under p^*. Intuitively speaking, this is what happens when we ask human experts to assess a model by comparing its samples with their own implicit, expert approximation of p^*. One expects a good model p_θ to produce realistic samples, which is reasonably modeled by the idea that the cross-entropy between p_θ and p^* should be small.

One also expects the model to produce samples that are as diverse as possible, which is directly related to the entropy of p_θ. Combining these ideas leads to the KL divergence $\mathrm{D_{KL}}(p_\theta \,||\, p^*) = \mathbb{E}_{x\sim p_\theta}[\log p_\theta(x) - \log p^*(x)]$.

GANs use a similar mechanism: samples are drawn from the model and evaluated by a discriminator D. Originally, [59] trained the discriminator with the loss

$$\mathcal{L}_{\mathrm{GAN}} = \int_x p^*(x) \log D(x) + p_\theta(x) \log(1 - D(x)) \mathrm{d}x, \tag{4.7}$$

and showed that, given $f(y) = a \log y + b \log(1 - y)$ is minimized for $y = \frac{a}{a+b}$, the optimal discriminator is given by

$$D^*(x) = \frac{p^*(x)}{p^*(x) + p_\theta(x)}. \tag{4.8}$$

Substituting the optimal discriminator, $\mathcal{L}_{\mathrm{GAN}}$ equals (up to additive and multiplicative constants) the Jensen-Shannon divergence:

$$\mathrm{JSD}(p^* || p_\theta) = \frac{1}{2}(\mathrm{D_{KL}}(p^* \,||\, (p_\theta + p^*)/2) + \mathrm{D_{KL}}(p_\theta \,||\, (p_\theta + p^*)/2)). \tag{4.9}$$

This loss, approximated by the discriminator, is symmetric and contains two KL divergence terms. While one of them is an integral on p^* the term that approximates it, $\int_x p^*(x) \log D(x)$, is independent from the generative model. Therefore, it cannot be used to perform coverage-driven training, and instead, the generator is trained either to minimize $\log(1 - D(G(z)))$ or to maximize $\log D(G(z))$ [59], where $G(z)$ is the deterministic generator that maps latent variables z to the data space. Assuming $D = D^*$, the first of these generator objective functions becomes:

$$\log(1 - D^*(G(z))) = \int_x p_\theta(x) \log \frac{p_\theta(x)}{p_\theta(x) + p^*(x)} = \mathrm{D_{KL}}(p_\theta \,||\, p_\theta + p^*). \tag{4.10}$$

As argued above, it seems reasonable that human assessment of generative models is also based on a quality-driven objective similar to that of the GAN generator, integrating a score function across samples from p_θ. This could explain why images produced by GANs typically correlate well with human judgment. The main failure case of GANs, and more generally of quality-driven models, is that they typically do not cover the full support of the data. This well-recognized phenomenon is known as *mode-dropping* [12,20]. GANs fail to provide an objective measure of this phenomenon, and of their performance in general. For instance, defining a valid likelihood requires adding volume to the low-dimensional manifold learned by GANs to define a density under which training and test data have non-zero density. Furthermore, computing the density of a datapoint under the defined density requires marginalizing out the latent variables. This is non trivial in

the absence of a readily available inference model. Quality-driven training mirrors the shortcomings of coverage-driven training: the reverse KL divergence explicitly targets sample quality, and only implicitly how well the model covers the training data.

As one of the main failure modes of GANs, mode-collapse has recently received considerable attention. One line of research is focused on allowing the discriminator to access batch statistics [103, 158]. Another line of research focuses on restricting discriminator to a certain functional class: it was demonstrated in [12] that Lipschitz-continuous discriminator together with Earth-Mover distance as loss function heavily improve the stability of training. Conditioning a network to be Lipschitz-continuous is not trivial, initial suggestion (weight clipping) was eventually replaced by more flexible gradient penalty [64]. These contributions allowed to move from simple convolutional architectures like [141] to ResNets. Even better results were obtained with spectral normalization of discriminator [115]: weights of every layer are divided by first singular value estimated via power method. Interestingly, results do not depend on loss function which hints that discriminator conditioning is more important than loss function. It was also suggested that conditioning of generator in the same way is beneficial for GAN reproducibility [128]. More detailed discussion of contemporary GAN models can be found in Section 5.2.1.

There are many recent advances in GANs that allow generation of high resolution samples when conditioning on image labels. In [116] combining spectral normalization with a projection discriminator yields convincing samples on ImageNet. SAGAN [198] uses a self-attention mechanism to allow the discriminator to better focus on salient long range dependencies. These contributions are orthogonal to our work and could be leveraged to further improve image quality. The same goes for the work of [78], where the depth of the convolutional architectures used to build the two competing networks is progressively increased as training advances.

4.3 Hybrid approaches

Some recent works are exploring models combining the benefits of auto-encoders and adversarial training. [90] use a VAE to model a feature space, taken to be one of the intermediate layers of a discriminator. This goes beyond the pixel-wise reconstruction loss in RGB space but does not yield a density model in the RGB image space. [42] and [40] learn an encoder and decoder model using a discriminator that, given a pair (x, z), predicts if z was encoded from a real image, or if x decoded from a z sampled from the prior. This procedure is fully adversarial and does not allow explicit optimization of a log-likelihood. This work is extended by [29], which showed that it is possible to approximately optimize the symmetric KL in a fully adversarial setup and introduced reconstruction losses in RGB space to improve

the correspondence between reconstructions and the ground truth. [186] collapse the encoder and a discriminator into one network that encodes real images and samples, and tries to separate their posteriors, yielding another fully adversarial approach. [107] replace the regularization term on the latent variables of a VAE with a discriminator that compares latent codes from the prior and from the posterior. This regularization is more flexible, but does not lead to a valid density model on images.

To go beyond the VAE independence assumption in pixel space, some recent works [31,65,104] have proposed using autoregressive decoders. These models suffer from slow sampling because of their autoregressive components, and could not be directly improved with a quality-driven mechanism.

4.4 Conditioning of generative models

Most generative models can be conditioned on additional information. In the simplest case it can be a class label. One of well-tested methods of GAN conditioning is ACGAN [129]: latent variables are concatenated with one hot class labels, discriminator has additional head to recognize a category and the GAN loss is augmented with categorical cross-entropy. It allows to train a model generating an image of a given class, however does not necessarily facilitates training of bigger models.

Recently, along with spectral normalization a new conditioning method was suggested [116]: generator is conditioned using conditional batch normalization [43] while discriminator uses cosine similarity to insert class information into GAN loss. This method demonstrated great results on ImageNet generation and was adopted by followup work [22, 198].

In principle, a lot of different information can be used to condition GANs: another image (like semantic segmentation or grayscale version) [76, 202], pose information [106, 123], text description [199].

VAEs can be conditioned on labels or semantic information with fairly straightforward modification of objective function [170]. PixelCNN can be also easily conditioned on arbitrary vector (class label or richer semantic information) [187] just by replacing all probabilities by conditional on some vector h probabilities in the model definition (Eq. 4.4).

4.5 Evaluation of GAN models

We present existing quantitative measures to evaluate GANs: scores based on an Inception network, i.e., IS and FID, a Wasserstein-based distance metric,

precision and recall scores, and a technique built with data augmentation.

4.5.1 Inception score

One of the most common ways to evaluate GANs is the Inception score [158]. It uses an Inception network [175] pre-trained on ImageNet to compute logits of generated images. The score is given by:

$$\mathrm{IS}(G) = \exp\left(\mathop{\mathbb{E}}_{\mathbf{x}\backsim p_g}[D_{\mathrm{KL}}(p(y|\mathbf{x}) \parallel p(y))]\right), \tag{4.11}$$

where $\mathbf{x}$ is a generated image sampled from the learned generator distribution p_g, $\mathbb{E}$ is the expectation over the set of generated images, D_{KL} is the KL-divergence between the conditional class distribution $p(y|\mathbf{x})$ (for label y, according to the Inception network) and the marginal class distribution $p(y) = \mathop{\mathbb{E}}_{\mathbf{x}\backsim p_g}[p(y|\mathbf{x})]$. By definition, Inception score does not consider real images at all, and so cannot measure how well the generator approximates the real distribution. This score is limited to measuring only the diversity of generated images. Some of its other limitations, as noted in [16], are: high sensitivity to small changes in weights of the Inception network, and large variance of scores.

4.5.2 Fréchet Inception distance

The recently proposed Fréchet Inception distance (FID) [73] compares the distributions of Inception embeddings (activations from the penultimate layer of the Inception network) of real ($p_r(\mathbf{x})$) and generated ($p_g(\mathbf{x})$) images. Both these distributions as modeled as multi-dimensional Gaussians parameterized by their respective mean and covariance. The distance measure is defined between the two Gaussian distributions as:

$$d^2((\mathbf{m}_r, \mathbf{C}_r), (\mathbf{m}_g, \mathbf{C}_g)) = \|\mathbf{m}_r - \mathbf{m}_g\|^2 + \mathrm{Tr}(\mathbf{C}_r + \mathbf{C}_g - 2(\mathbf{C}_r\mathbf{C}_g)^{\frac{1}{2}}), \tag{4.12}$$

where $(\mathbf{m}_r, \mathbf{C}_r)$, $(\mathbf{m}_g, \mathbf{C}_g)$ denote the mean and covariance of the real and generated image distributions respectively. FID is inversely correlated with Inception score, and suffers from the same issues discussed earlier.

The two Inception-based measures cannot separate image quality from image diversity. For example, low IS or FID values can be due to the generated images being either not realistic (low image quality) or too similar to each other (low diversity), with no way to analyze the cause. In contrast, our measures can distinguish when generated images become less diverse from worse image quality.

4.5.3 Other evaluation measures

Sliced Wasserstein distance (SWD) [78] was used to evaluate high-resolution GANs. It is a multi-scale statistical similarity computed on local image patches extracted from the Laplacian pyramid representation of real and generated images. A total of 128 7×7 local patches for each level of the Laplacian pyramid are extracted per image. While SWD is an efficient approximation, using randomized projections [140], of the Wasserstein-1 distance between the real and generated images, its utility is limited when comparing a variety of GAN models, with not all of them producing high-resolution images (see our evaluation in Section 5.3).

Precision and recall measures were introduced [105] in the context of GANs, by constructing a synthetic data manifold. This makes it possible to compute the distance of an image sample (generated or real) to the manifold, by finding its distance to the closest point from the manifold. In this synthetic setup, precision is defined as the fraction of the generated samples whose distance to the manifold is below a certain threshold. Recall, on the other hand, is computed by considering a set of test samples. First, the latent representation $\tilde{\mathbf{z}}$ of each test sample $\mathbf{x}$ is estimated, through gradient descent, by inverting the generator G. Recall is then given by the fraction of test samples whose L2-distance to $G(\tilde{\mathbf{z}})$ is below the threshold. High recall is equivalent to the GAN capturing most of the manifold, and high precision implies that the generated samples are close to the manifold. Although these measures bring the flavor of techniques used widely to evaluate discriminative models to GANs, they are impractical for real images as the data manifold is unknown, and their use is limited to evaluations on synthetic data [105].

4.6 Data augmentation with GANs

Augmenting training data is an important component of learning neural networks. This can be achieved by increasing the size of the training set [88] or incorporating augmentation directly in the latent space [191]. A popular technique is to increase the size of the training set with minor transformations of data, which has resulted in a performance boost, e.g., for image classification [88]. GANs provide a natural way to augment training data with the generated samples. Indeed, GANs have been used to train classification networks in a semi-supervised fashion [34, 184] or to facilitate domain adaptation [21]. Modern GANs generate images realistic enough to improve performance in applications, such as, biomedical imaging [24, 53], person re-identification [201] and image enhancement [195]. They can also be used to refine training sets composed of synthetic images for applications such as eye gaze and hand pose estimation [165]. GANs are also used to learn complex 3D distributions and replace computationally intensive simulations in

physics [119, 131] and neuroscience [117]. Ideally, GANs should be able to recreate the training set with different variations. This can be used to compress datasets for learning incrementally, without suffering from catastrophic forgetting as new classes are added [162]. We will study the utility of GANs for training image classification networks with data augmentation (see Section 5.3.5), and analyze it as an evaluation measure.

In summary, evaluation of generative models is not a easy task [180], especially for models like GANs. We bring a new dimension to this problem with our GAN-train and GAN-test performance-based measures, and show through our extensive analysis that they are complementary to all the above schemes.

Chapter 5

Evaluation of conditional GANs

Contents

5.1 GAN-train and GAN-test **55**
5.2 Datasets and methods **57**
5.2.1 Evaluated methods 58
5.3 Experiments **59**
5.3.1 Implementation details of evaluation measures 59
5.3.2 Implementation details of generative models 60
5.3.3 Generative model evaluation 61
5.3.4 GAN image diversity 65
5.3.5 GAN data augmentation 65
5.3.6 Dataset compression 67
5.3.7 t-SNE embeddings of datasets 67
5.4 Summary **67**

5.1 GAN-train and GAN-test

An important characteristic of a conditional GAN model is that generated images should not only be realistic, but also recognizable as coming from a given class. An optimal GAN that perfectly captures the target distribution can generate a new set of images S_g, which are indistinguishable from the original training set S_t. Assuming both these sets have the same size, a classifier trained on either of them

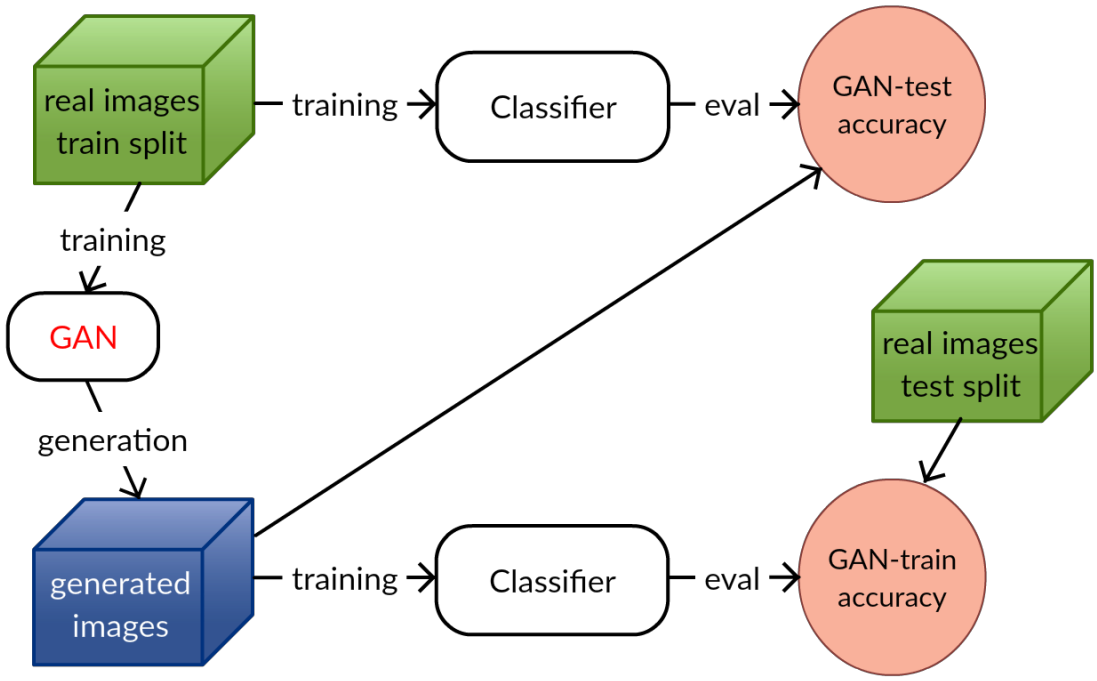

Figure 5.1 – Illustration of GAN-train and GAN-test. GAN-train learns a classifier on GAN generated images and measures the performance on real test images. This evaluates the diversity and realism of GAN images. GAN-test learns a classifier on real images and evaluates it on GAN images. This measures how realistic GAN images are.

should produce roughly the same validation accuracy. This is indeed true when the dataset is simple enough, for example, MNIST [162] (see also Section 5.3.3). Motivated by this optimal GAN characteristic, we devise two scores to evaluate GANs, as illustrated in Figure 5.1.

GAN-train is the accuracy of a classifier trained on S_g and tested on a validation set of real images S_v. When a GAN is not perfect, GAN-train accuracy will be lower than the typical validation accuracy of the classifier trained on S_t. It can happen due to many reasons, e.g., (i) mode dropping reduces the diversity of S_g in comparison to S_t, (ii) generated samples are not realistic enough to make the classifier learn relevant features, (iii) GANs can mix-up classes and confuse the classifier. Unfortunately, GAN failures are difficult to diagnose. When GAN-train accuracy is close to validation accuracy, it means that GAN images are high quality and as diverse as the training set. As we will show in Section 5.3.4, diversity varies with the number of generated images. We will analyze this with the evaluation discussed at the end of this section.

GAN-test is the accuracy of a classifier trained on the original training set S_t, but tested on S_g. If a GAN learns well, this turns out be an easy task because both the sets have the same distribution. Ideally, GAN-test should be close to the validation accuracy. If it significantly higher, it means that the GAN overfits, and simply memorizes the training set. On the contrary, if it is significantly lower, the GAN does not capture the target distribution well and the image quality is poor. Note that this measure does not capture the diversity of samples because a model that memorizes exactly one training image perfectly will score very well. GAN-test accuracy is related to the precision score in [105], quantifying how close generated images are to a data manifold.

To provide an insight into the diversity of GAN-generated images, we measure GAN-train accuracy with generated sets of different sizes, and compare it with the validation accuracy of a classifier trained on real data of the corresponding size. If all the generated images were perfect, the size of S_g where GAN-train is equal to validation accuracy with the reduced-size training set, would be a good estimation of the number of distinct images in S_g. In practice, we observe that GAN-train accuracy saturates with a certain number of GAN-generated samples (see Figure 5.3(a) and Figure 5.3(b) discussed in Section 5.3.4). This is a measure of the diversity of a GAN, similar to recall from [105], measuring the fraction of the data manifold covered by a GAN.

5.2 Datasets and methods

Datasets. For comparing the different GAN methods and PixelCNN++, we use several image classification datasets with an increasing number of labels: MNIST [92], CIFAR10 [87], CIFAR100 [87] and ImageNet1k [155]. CIFAR10 and CIFAR100 both have 50k 32×32 RGB images in the training set, and 10k images in the validation set. CIFAR10 has 10 classes while CIFAR100 has 100 classes. ImageNet1k has 1000 classes with 1.3M training and 50k validation images. We downsample the original ImageNet images to two resolutions in our experiments, namely 64×64 and 128×128. MNIST has 10 classes of 28×28 grayscale images, with 60k samples for training and 10k for validation.

We exclude the CIFAR10/CIFAR100/ImageNet1k validation images from GAN training to enable the evaluation of test accuracy. This is not done in a number of GAN papers and may explain minor differences in IS and FID scores compared to the ones reported in these papers.

5.2.1 Evaluated methods

Among the plethora of GAN models in literature, it is difficult to choose the best one, especially since appropriate hyperparameter fine-tuning appears to bring all major GANs within a very close performance range, as noted in a study [105]. We choose to perform our analysis on Wasserstein GAN (WGAN-GP), one of the most widely-accepted models in literature at the moment, and SNGAN, a very recent model showing state-of-the-art image generation results on ImageNet. Additionally, we include two baseline generative models, DCGAN [141] and PixelCNN++ [159]. We summarize all the models included in our experimental analysis below.

Wasserstein GAN. WGAN [12] replaces the discriminator separating real and generated images with a critic estimating Wasserstein-1 (i.e., earth-mover's) distance between their corresponding distributions. The success of WGANs in comparison to the classical GAN model [59] can be attributed to two reasons. Firstly, the optimization of the generator is easier because the gradient of the critic function is better behaved than its GAN equivalent. Secondly, empirical observations show that the WGAN value function better correlates with the quality of the samples than GANs [12].

In order to estimate the Wasserstein-1 distance between the real and generated image distributions, the critic must be a K-Lipschitz function. The original paper [12] proposed to constrain the critic through weight clipping to satisfy this Lipschitz requirement. This, however, can lead to unstable training or generate poor samples [64]. An alternative to clipping weights is the use of a gradient penalty as a regularizer to enforce the Lipschitz constraint. In particular, we penalize the norm of the gradient of the critic function with respect to its input. This has demonstrated stable training of several GAN architectures [64].

We use the gradient penalty variant of WGAN, conditioned on data in our experiments, and refer to it as WGAN-GP in the rest of the chapter. Label conditioning is an effective way to use labels available in image classification training data [129]. Following ACGAN [129], we concatenate the noise input $\mathbf{z}$ with the class label in the generator, and modify the discriminator to produce probability distributions over the sources as well as the labels.

SNGAN. Variants have also analyzed other issues related to training GANs, such as the impact of the performance control of the discriminator on training the generator. Generators often fail to learn the multimodal structure of the target distribution due to unstable training of the discriminator, particularly in high-dimensional spaces [115]. More dramatically, generators cease to learn when the supports of the real and the generated image distributions are disjoint [11]. This occurs since the discriminator quickly learns to distinguish these distributions, resulting in the gradients of the discriminator function, with respect to the input, becoming zeros, and thus failing to update the generator model any further.

SNGAN [115] introduces spectral normalization to stabilize training the discriminator. This is achieved by normalizing each layer of the discriminator (i.e., the learnt weights) with the spectral norm of the weight matrix, which is its largest singular value. Miyato *et al.* [115] showed that this regularization outperforms other alternatives, including gradient penalty, and in particular, achieves state-of-the-art image synthesis results on ImageNet. We use the class-conditioned version of SNGAN [116] in our evaluation. Here, SNGAN is conditioned with projection in the discriminator network, and conditional batch normalization [43] in the generator network.

DCGAN. Deep convolutional GANs (DCGANs) is a class of architecture that was proposed to leverage the benefits of supervised learning with CNNs as well as the unsupervised learning of GAN models [141]. The main principles behind DCGANs are using only convolutional layers and batch normalization for the generator and discriminator networks. Several instantiations of DCGAN are possible with these broad guidelines, and in fact, many do exist in literature [64, 115, 129]. We use the class-conditioned variant presented in [129] for our analysis.

PixelCNN++. The original PixelCNN [188] belongs to a class of generative models with tractable likelihood. It is a deep neural net which predicts pixels sequentially along both the spatial dimensions. The spatial dependencies among pixels are captured with a fully convolutional network using masked convolutions. PixelCNN++ proposes improvements to this model in terms of regularization, modified network connections and more efficient training [159].

5.3 Experiments

5.3.1 Implementation details of evaluation measures

We compute Inception score with the WGAN-GP code [1] corrected for the 1008 classes problem [16]. The mean value of this score computed 10 times on 5k splits is reported in all our evaluations, following standard protocol.

We found that there are two variants for computing FID. The first one is the original implementation [2] from the authors [73], where all the real images and at least 10k generated images are used. The second one is from the SNGAN [115] implementation, where 5k generated images are compared to 5k real images. Estimation of the covariance matrix is also different in both these cases. Hence, we include these two versions of FID in the paper to facilitate comparison in the future. The original implementation is referred to as FID, while our implementation [4] of the 5k version is denoted as FID-5K. Implementation of SWD is taken from the official NVIDIA repository [3].

5.3.2 Implementation details of generative models

SNGAN in our experiments refers to the model with ResNet architecture, hinge loss, spectral normalization [115], conditional batch normalization [43] and conditioning via projection [116]. We evaluate two variants of WGAN-GP: one with 2.5M and the other with 10M parameters. Both these variants use ResNet architecture from [64], Wasserstein loss [12] with gradient penalty [64] and conditioning via an auxiliary classifier [129], with the only difference being that, there are twice as many filters in each layer of the generator for the 10M model over the 2.5M variant. We use DCGAN with a simple convnet architecture [141], classical GAN loss, conditioned via an auxiliary classifier [129], as in [115].

We reimplemented WGAN-GP, SNGAN, DCGAN, and validated our implementations on CIFAR10 to ensure that they match the published results. Our implementations are available online [4]. For PixelCNN++, we used the reference implementation [1] for training and the accelerated version [2] for inference.

For all the CIFAR experiments, SNGAN and WGAN-GP are trained for 100k iterations with a linearly decaying learning rate, starting from 0.0002 to 0, using the Adam optimizer [81] ($\beta_1 = 0$, $\beta_2 = 0.9$, 5 critic steps per generator step, and generator batch size 64). ACGAN loss scaling coefficients are 1 and 0.1 for the critic's and generator's ACGAN losses respectively. Gradient penalty is 10 as recommended [64]. DCGAN is also trained for 100k iterations with learning rate 0.0002, Adam parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$, and batch size of 100.

In the case of ImageNet, we follow [116] for ResNet architecture and the protocol. We trained for 250k iterations with learning rate 0.0002, linearly decaying to 0, starting from the 200k-th iteration, for resolution 64×64. We use the Adam optimizer [81] ($\beta_1 = 0$, $\beta_2 = 0.9$, 5 critic steps per generator step, generator batch size 64). For 128×128 resolution, we train for 450k iterations, with learning rate linearly decaying to 0, starting from the 400k-th iteration. WGAN-GP for ImageNet uses the same ResNet and training schedule as SNGAN. Its gradient penalty and ACGAN loss coefficients are identical to the CIFAR case.

The classifier for computing GAN-train and GAN-test is a preactivation variant of ResNet-32 from [71] in the CIFAR10 and CIFAR100 evaluation. Training schedule is identical to the original paper: 64k iterations using momentum optimizer with learning rate 0.1 dropped to 0.01, after 32k iterations, and 0.001, after 48k iterations (batch size 128). We also use standard CIFAR data augmentation: 32×32 crops are randomly sampled from the padded image (4 pixels on each side), which are flipped horizontally. The classifier in the case of ImageNet is ResNet-32 for 64×64 and ResNet-34 for 128×128 with momentum optimizer for 400k iterations, and learning rate 0.1, dropped to 0.01 after 200k steps (batch size 128). We use a

1. https://github.com/openai/pixel-cnn
2. https://github.com/PrajitR/fast-pixel-cnn

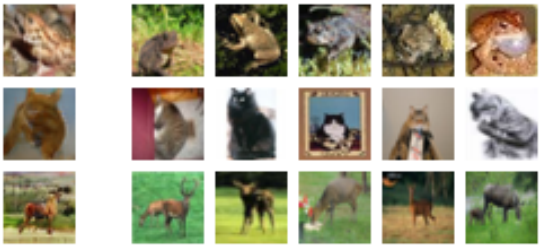

Figure 5.2 – First column: SNGAN-generated images. Other columns: 5 images from CIFAR10 "train" closest to GAN image from the first column in feature space of baseline CIFAR10 classifier.

central crop for both training and testing. We also compute our measures with a random forest classifier [75]. Here, we used the scikit-learn implementation [134] with 100 trees and no depth limitation.

To train SNGAN on MNIST we use the same GAN architecture as in the case of CIFAR10 (adjusted to a single input channel), and a simple convnet architecture (four convolutional layers with 32, 64, 128, 256 filters with batch normalization and max pooling in between, and global average pooling before the final output layer) as the baseline classifier. The GAN training schedule is also the same as in the case of CIFAR10. We train the baseline classifier for 64k iterations using a momentum optimizer with learning rate 0.1 dropped to 0.01, after 32k iterations, and 0.001, after 48k iterations (using a batch size of 128).

5.3.3 Generative model evaluation

MNIST. We validate our claim (from Section 5.1) that a GAN can perfectly reproduce a simple dataset on MNIST. A four-layer convnet classifier trained on real MNIST data achieves 99.3% accuracy on the test set. In contrast, images generated with SNGAN achieve a GAN-train accuracy of 99.0% and GAN-test accuracy of 99.2%, highlighting their high image quality as well as diversity.

CIFAR10. Table 5.1 shows a comparison of state-of-the-art GAN models on CIFAR10. We observe that the relative ranking of models is consistent across different metrics: FID, GAN-train and GAN-test accuracies. Both GAN-train and GAN-test are quite high for SNGAN and WGAN-GP (10M). This implies that both the image quality and the diversity are good, but are still lower than that of real images (92.8 in the first row). Note that PixelCNN++ has low diversity because GAN-test is much higher than GAN-train in this case. This is in line with its relatively poor Inception score and FID (as shown in [105] FID is quite sensitive to mode dropping).

model	IS	FID-5K	FID	GAN-train	GAN-test	SWD 16	SWD 32
real images	11.33	9.4	2.1	92.8	-	2.8	2.0
SNGAN	8.43	18.8	11.8	82.2	87.3	3.9	24.4
WGAN-GP (10M)	8.21	21.5	14.1	79.5	85.0	3.8	6.2
WGAN-GP (2.5M)	8.29	22.1	15.0	76.1	80.7	3.4	6.9
DCGAN	6.69	42.5	35.6	65.0	58.2	6.5	24.7
PixelCNN++	5.36	121.3	119.5	34.0	47.1	14.9	56.6

Table 5.1 – CIFAR10 experiments. IS: higher is better. FID and SWD: lower is better. SWD values here are multiplied by 10^3 for better readability. GAN-train and GAN-test are accuracies given as percentage (higher is better).

Note that SWD does not correlate well with other metrics: it is consistently smaller for WGAN-GP (especially SWD 32). We hypothesize that this is because SWD approximates the Wasserstein-1 distance between patches of real and generated images, which is related to the optimization objective of Wasserstein GANs, but not other models (e.g., SNGAN). This suggests that SWD is unsuitable to compare WGAN and other GAN losses. It is also worth noting that WGAN-GP (10M) shows only a small improvement over WGAN-GP (2.5M) despite a four-fold increase in the number of parameters. In Figure 5.2 we show SNGAN-generated images on CIFAR10 and their nearest neighbors from the training set in the feature space of the classifier we use to compute the GAN-test measure. Note that SNGAN consistently finds images of the same class as a generated image, which are close to an image from the training set.

To highlight the complementarity of GAN-train and GAN-test, we emulate a simple model by subsampling/corrupting the CIFAR10 training set, in the spirit of [73]. GAN-train/test now corresponds to training/testing the classifier on modified data. We observe that GAN-test is insensitive to subsampling unlike GAN-train (where it is equivalent to training a classifier on a smaller split). Salt and pepper noise, ranging from 1% to 20% of replaced pixels per image, barely affects GAN-train, but degrades GAN-test significantly (from 82% to 15%).

Through this experiment on modified data, we also observe that FID is insufficient to distinguish between the impact of image diversity and quality. For example, FID between CIFAR10 train set and train set with Gaussian noise ($\sigma = 5$) is 27.1, while FID between train set and its random 5k subset with the same noise is 29.6. This difference may be due to lack of diversity or quality or both. GAN-test, which measures the quality of images, is identical (95%) in both these cases. GAN-train, on the other hand, drops from 91% to 80%, showing that the 5k train set lacks diversity. Together, our measures, address one of the main drawbacks of FID.

CIFAR100. Our results on CIFAR100 are summarized in Table 5.2. It is a more challenging dataset than CIFAR10, mainly due to the larger number of classes and fewer images per class; as evident from the accuracy of a convnet for classification trained with real images: 92.8 vs 69.4 for CIFAR10 and CIFAR100 respectively. SNGAN and WGAN-GP (10M) produce similar IS and FID, but very different GAN-train and GAN-test accuracies. This makes it easier to conclude that SNGAN has better image quality and diversity than WGAN-GP (10M). It is also interesting to note that WGAN-GP (10M) is superior to WGAN-GP (2.5M) in all the metrics, except SWD. WGAN-GP (2.5M) achieves reasonable IS and FID, but the quality of the generated samples is very low, as evidenced by GAN-test accuracy. SWD follows the same pattern as in the CIFAR10 case: WGAN-GP shows a better performance than others in this measure, which is not consistent with its relatively poor image quality. PixelCNN++ exhibits an interesting behavior, with high GAN-test accuracy, but very low GAN-train accuracy, showing that it can generate images of acceptable quality, but they lack diversity. A high FID in this case also hints at significant mode dropping.

Random forests. We verify if our findings depend on the type of classifier by using random forests [75,134] instead of CNN for classification. This results in GAN-train, GAN-test scores of 15.2%, 19.5% for SNGAN, 10.9%, 16.6% for WGAN-GP (10M), 3.7%, 4.8% for WGAN-GP (2.5M), and 3.2%, 3.0% for DCGAN respectively. Note that the relative ranking of these GANs remains identical for random forests and CNNs.

Human study. We designed a human study with the goal of finding which of the measures (if any) is better aligned with human judgement. The subjects were asked to choose the more realistic image from two samples generated for a particular class of CIFAR100. Five subjects evaluated SNGAN vs one of the following: DCGAN, WGAN-GP (2.5M), WGAN-GP (10M) in three separate tests. They made 100 comparisons of randomly generated image pairs for each test, i.e., 1500 trials in total. All of them found the task challenging, in particular for both WGAN-GP tests.

We use Student's t-test for statistical analysis of these results. In SNGAN vs DCGAN, subjects chose SNGAN 368 out of 500 trials, in SNGAN vs WGAN-GP (2.5M), subjects preferred SNGAN 274 out of 500 trials, and in SNGAN vs WGAN-GP (10M), SNGAN was preferred 230 out of 500. The preference of SNGAN over DCGAN is statistically significant ($p < 10^{-7}$), while the preference over WGAN-GP (2.5M) or WGAN-GP (10M) is insignificant ($p = 0.28$ and $p = 0.37$ correspondingly). We conclude that the quality of images generated needs to be significantly different, as in the case of SNGAN vs DCGAN, for human studies to be conclusive. They are insufficient to pick out the subtle, but performance-critical, differences, unlike our measures.

model	IS	FID-5K	FID	GAN-train	GAN-test	SWD 16	SWD 32
real images	14.9	10.8	2.4	69.4	-	2.7	2.0
SNGAN	9.30	23.8	15.6	45.0	59.4	4.0	15.6
WGAN-GP (10M)	9.10	23.5	15.6	26.7	40.4	6.0	9.1
WGAN-GP (2.5M)	8.22	28.8	20.6	5.4	4.3	3.7	7.7
DCGAN	6.20	49.7	41.8	3.5	2.4	9.9	20.8
PixelCNN++	6.27	143.4	141.9	4.8	27.5	8.5	25.9

Table 5.2 – CIFAR100 experiments. Refer to the caption of Table 5.1 for details.

res	model	IS	FID-5K	FID	GAN-train top-1	GAN-train top-5	GAN-test top-1	GAN-test top-5
64px	real images	63.8	15.6	2.9	55.0	78.8	-	-
	SNGAN	12.3	44.5	34.4	3	8.4	12.9	28.9
	WGAN-GP	11.3	46.7	35.8	0.1	0.7	0.1	0.5
128px	real images	203.2	17.4	3.0	59.1	81.9	-	-
	SNGAN*	35.3	44.9	33.2	9.3	21.9	39.5	63.4
	WGAN-GP	11.6	91.6	79.5	0.1	0.5	0.1	0.5

Table 5.3 – ImageNet experiments. SNGAN* refers to the model provided by [115], trained for 850k iterations. Refer to the caption of Table 5.1 for details.

ImageNet. On this dataset, which is one of the more challenging ones for image synthesis [115], we analyzed the performance of the two best GAN models based on our CIFAR experiments, i.e., SNGAN and WGAN-GP. As shown in Table 5.3, SNGAN achieves a reasonable GAN-train accuracy and a relatively high GAN-test accuracy at 128×128 resolution. This suggests that SNGAN generated images have good quality, but their diversity is much lower than the original data. This may be partly due to the size of the generator (150Mb) being significantly smaller in comparison to ImageNet training data (64Gb for 128×128). Despite this difference in size, it achieves GAN-train accuracy of 9.3% and 21.9% for top-1 and top-5 classification results respectively. In comparison, the performance of WGAN-GP is dramatically poorer; see last row for each resolution in the table.

In the case of images generated at 64×64 resolution, GAN-train and GAN-test accuracies with SNGAN are lower than their 128×128 counterparts. GAN-test accuracy is over four times better than GAN-train, showing that the generated images lack in diversity. It is interesting to note that WGAN-GP produces Inception score and FID very similar to SNGAN, but its images are insufficient to train a reasonable classifier and to be recognized by an ImageNet classifier, as shown by

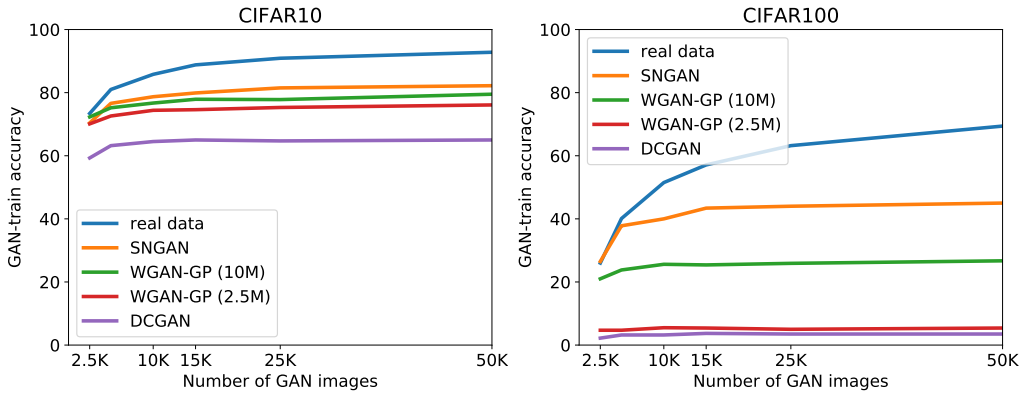

Figure 5.3 – The effect of varying the size of the generated image set on GAN-train accuracy. For comparison, we also show the result (in blue) of varying the size of the real image training dataset. (Best viewed in pdf.)

the very low GAN-train and GAN-test scores.

5.3.4 GAN image diversity

We further analyze the diversity of the generated images by evaluating GAN-train accuracy with varying amounts of generated data. A model with low diversity generates redundant samples, and increasing the quantity of data generated in this case does not result in better GAN-train accuracy. In contrast, generating more samples from a model with high diversity produces a better GAN-train score. We show this analysis in Figure 5.3, where GAN-train accuracy is plotted with respect to the size of the generated training set on CIFAR10 and CIFAR100.

In the case of CIFAR10, we observe that GAN-train accuracy saturates around 15-20k generated images, even for the best model SNGAN (see Figure 5.3a). With DCGAN, which is weaker than SNGAN, GAN-train saturates around 5k images, due to its relatively poorer diversity. Figure 5.3b shows no increase in GAN-train accuracy on CIFAR100 beyond 25k images for all the models. The diversity of 5k SNGAN-generated images is comparable to the same quantity of real images; see blue and orange plots in Figure 5.3b. WGAN-GP (10M) has very low diversity beyond 5k generated images. WGAN-GP (2.5M) and DCGAN perform poorly on CIFAR100, and are not competitive with respect to the other methods.

5.3.5 GAN data augmentation

We analyze the utility of GANs for data augmentation, i.e., for generating additional training samples, with the best-performing GAN model (SNGAN) under

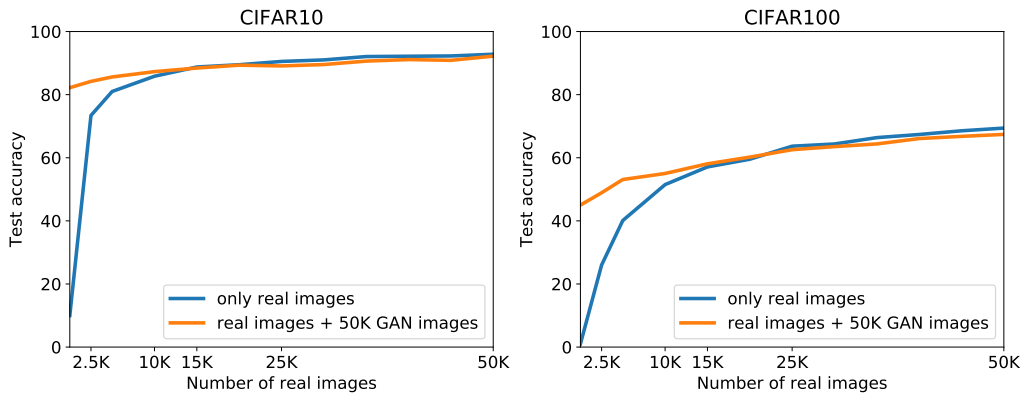

Figure 5.4 – The impact of training a classifier with a combination of real and SNGAN generated images.

Num real images	real C10	real+GAN C10	real C100	real+GAN C100
2.5k	73.4	67.0	25.6	23.9
5k	80.9	77.9	40.0	33.5
10k	85.8	83.5	51.5	45.5

Table 5.4 – Data augmentation when SNGAN is trained with reduced real image set. Classifier is trained either on this data (real) or a combination of real and SNGAN generated images (real+GAN). Performance is shown as % accuracy.

two settings. First, in Figure 5.4a and Figure 5.4b, we show the influence of training the classifier with a combination of real images from the training set and 50k GAN-generated images on the CIFAR10 and CIFAR100 datasets respectively. In this case, SNGAN is trained with all the images from the original training set. From both the figures, we observe that adding 2.5k or 5k real images to the 50k GAN-generated images improves the accuracy over the corresponding real-only counterparts. However, adding 50k real images does not provide any noticeable improvement, and in fact, reduces the performance slightly in the case of CIFAR100 (Figure 5.4b). This is potentially due to the lack of image diversity.

This experiment provides another perspective on the diversity of the generated set, given that the generated images are produced by a GAN learned from the entire CIFAR10 (or CIFAR100) training dataset. For example, augmenting 2.5k real images with 50k generated ones results in a better test accuracy than the model trained only on 5k real images. Thus, we can conclude that the GAN model generates images that have more diversity than the 2.5k real ones. This is however, assuming that the generated images are as realistic as the original data. In practice, the generated images tend to be lacking on the realistic front, and are

more diverse than the real ones. These observations are in agreement with those from Section 5.3.4, i.e., SNGAN generates images that are at least as diverse as 5k randomly sampled real images.

In the second setting, SNGAN is trained in a low-data regime. In contrast to the previous experiment, we train SNGAN on a reduced training set, and then train the classifier on a combination of this reduced set, and the same number of generated images. Results in Table 5.4 show that on both CIFAR10 and CIFAR100 (C10 and C100 respectively in the table), the behaviour is consistent with the whole dataset setting (50k images), i.e., accuracy drops slightly.

5.3.6 Dataset compression

To complete the discussion on dataset memorization, we compare the size of the generator and the real dataset. In our CIFAR experiments SNGAN has 8.3M parameters (33Mb), WGAN-GP has either 2.5M or 10M parameters (10Mb or 40Mb respectively). In comparison, CIFAR10/100 have 50k training images which amounts to 150Mb of uncompressed data.

SNGAN for ImageNet has 42M parameters (168Mb) and WGAN-GP has 48M parameters (192Mb). The entire ImageNet training set takes 16Gb in 64×64 resolution and 64Gb in 128×128 resolution. This difference may partially explain why compression of CIFAR10/100 into a GAN is relatively easier than ImageNet.

5.3.7 t-SNE embeddings of datasets

In Figure 5.5 we show t-SNE embedding of randomly selected images from the CIFAR100 train and test set splits, and images generated from the four GAN models (500 images per split or model) of 5 classes (apple, aquarium fish, baby, bear, bicycle). The t-SNE visualizations are generated with the embeddings of these images in the feature space of the baseline classifier trained on CIFAR100, i.e., the classifier used to compute GAN-test accuracy. Note that the quality of this clustering is in line with the GAN-test accuracy in Table 5.2 in the main paper. For example, the images generated by SNGAN and WGAN-GP (10M), which produce the two best GAN-test accuracies, lie close to the training images, while those from WGAN-GP (2.5M) and DCGAN form a point cloud that does not correspond to any of the clusters.

5.4 Summary

This chapter presents steps towards addressing the challenging problem of evaluating and comparing images generated by GANs. To this end, we present

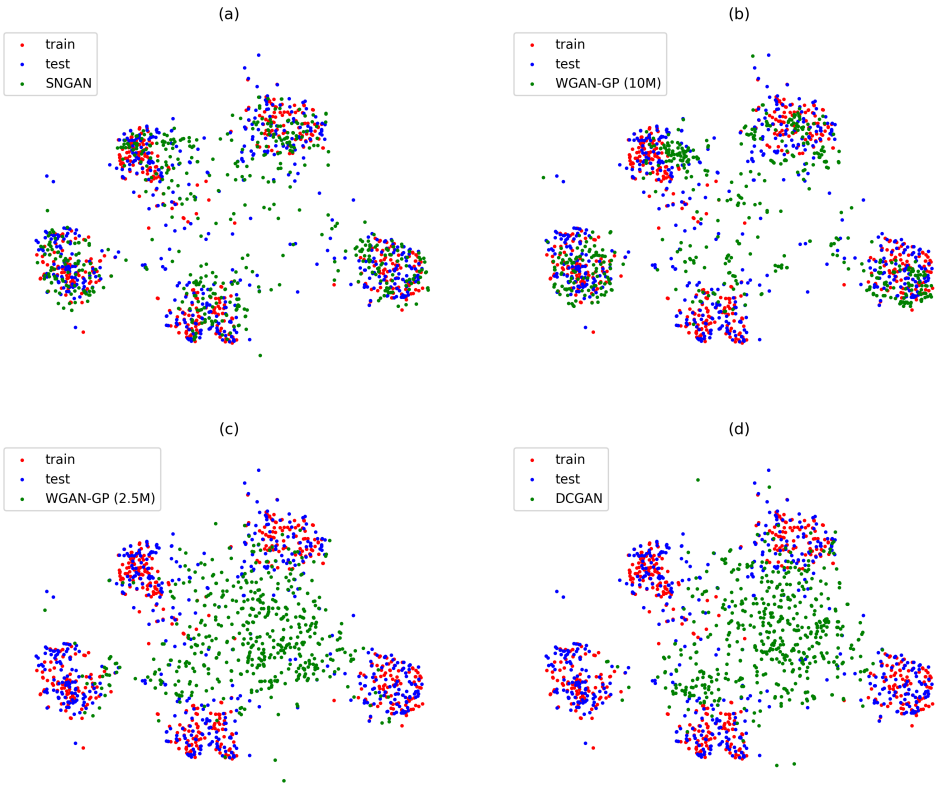

Figure 5.5 – t-SNE [189] embedding of images from 5 CIFAR100 classes, embedded in the feature space of the baseline CIFAR100 classifier. We use 500 images each from the train and test sets, along with 500 images each generated with (a) SNGAN, (b) WGAN-GP (10M), (c) WGAN-GP (2.5M), and (d) DCGAN.

new quantitative measures, GAN-train and GAN-test, which are motivated by precision and recall scores popularly used in the evaluation of discriminative models. We evaluate several recent GAN approaches as well as other popular generative models with these measures. Our extensive experimental analysis demonstrates that GAN-train and GAN-test not only highlight the difference in performance of these methods, but are also complementary to existing scores.

Chapter 6

Adversarial training of partially invertible variational autoencoders

Contents

6.1 Coverage and quality driven training **70**
6.1.1 Partially Invertible Variational Autoencoders 70
6.1.2 Improving samples with adversarial training 72
6.2 Experimental evaluation **72**
6.2.1 Evaluation protocol 73
6.2.2 Comparison to GAN and VAE baselines 73
6.2.3 Qualitative influence of the feature space flexibility . . 75
6.2.4 Evaluation of architectural refinements 76
6.2.5 Comparison to the state of the art 77
6.2.6 Results on additional datasets 78
6.2.7 Class-conditional results 80
6.3 Model refinements and implementation details **81**
6.3.1 Top-down sampling of hierarchical latent variables . . 81
6.3.2 Inverse autoregressive flow 82
6.3.3 Gradient penalty 83
6.3.4 Architecture and training hyper-parameters 83
6.4 Conclusion **84**

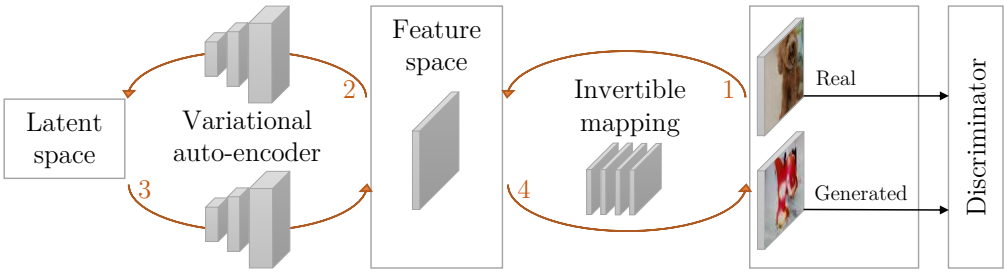

Figure 6.1 – Overview of our model: An invertible non-linear mapping f maps an image x to a feature space with the same dimension (arrows 1 and 4). The encoder takes $f(x)$ and maps it to a posterior distribution $q_\phi(z|x)$ over the latent variable (arrow 2), while the decoder maps z to a distribution over the feature space (arrow 3). The discriminator $D(x)$ assesses sample quality in the image space.

6.1 Coverage and quality driven training

In this section we describe our generative model, and how we train it to ensure both quality and coverage.

6.1.1 Partially Invertible Variational Autoencoders

Adversarial training requires continuous sampling from the model during training. As VAEs and flow-based models allow for efficient feed-forward sampling, they are suitable likelihood-based models to build our approach on. VAEs rely on an inference network $q_\phi(z|x)$, or "encoder", to construct a variational evidence lower-bound (ELBO),

$$\mathcal{L}_{elbo}(x, \phi, \theta) = \mathop{\mathbb{E}}_{q_\phi(z|x)} [\log(p_\theta(x|z)) - \mathrm{D_{KL}}(q_\phi(z|x) \,||\, p_\theta(z))] \leq \log p_\theta(x). \tag{6.1}$$

The "decoder" $p_\theta(x|z)$ has a convolutional architecture similar to that of a GAN generator, with the exception that the decoder maps the latent variable z to a distribution over images, rather to a single image. Another difference is that in a VAE the prior $p_\theta(z)$ is typically learned, and more flexible than in a GAN, which can significantly improve the likelihoods on held-out data [83]. Our generative model uses a latent variable hierarchy with top-down sampling similar to [13, 83, 172], see Section 6.3.1. It also leverages inverse auto-regressive flow [83] to obtain accurate posterior approximations, beyond commonly used factorized Gaussian approximations, see Section 6.3.2.

Typically, strong independence assumptions are also made on the decoder, e.g. by constraining it to a fully factorized Gaussian, i.e. $p_\theta(x|z) = \prod_{i=1} \mathcal{N}(x_i; \mu_i(z), \sigma_i(z))$.

In this case, all dependency structure across the pixels has to be modeled by the latent variable z, any correlations not captured by z are treated as independent per-pixel noise. Unless z captures each and every aspect of the image structure, this is a poor model for natural images, and leads the model to over-generalize with independent per-pixel noise around blurry non-realistic examples. Using the decoder to produce a sparse Cholesky decomposition of the inverse covariance matrix alleviates this problem to some extent [41], but retains a limiting assumption of linear-Gaussian dependency across pixel values.

Flow-based models offer a more flexible alternative, allowing to depart from Gaussian or other parametric distributions. Models such as NVP [38] map an image $x \in X$ from RGB space to a latent code $y \in Y$ using a bijection $f : X \to Y$, and rely on the change of variable formula to compute the likelihood

$$p_X(x) = p_Y(f(x)) \left| \det \left(\frac{\partial f(x)}{\partial x^T} \right) \right| . \tag{6.2}$$

To sample x, we first sample y from a parametric prior, e.g. a unit Gaussian, and use the reverse mapping f^{-1} to find the corresponding x. Despite allowing for exact inference and efficient sampling, current flow-based approaches are worse than state-of-the-art likelihood-based approaches in terms held-out likelihood, and sample quality.

In our model we use invertible network layers to map RGB images to an abstract feature space $f(x)$. A VAE is then trained to model the distribution of $f(x)$. This results in a non-factorial and non-parametric form of $p_\theta(x|z)$ in the space of RGB images. See Figure 6.1 for a schematic illustration of the model. Although the likelihood of this model is intractable to compute, we can rely on a lower bound for training:

$$\mathcal{L}_C(p_\theta) = - \mathop{\mathbb{E}}_{p^*(x)} \left[\mathcal{L}_{elbo}(f(x), \phi, \theta) + \log \left| \det \frac{\partial f(x)}{\partial x^T} \right| \right] \leq \mathop{\mathbb{E}}_{p^*(x)} \left[-\log p_\theta(x) \right] . \tag{6.3}$$

The bound is obtained by combining the VAE variational lower bound of Eq. (6.1) with the change of variable formula of Eq. (6.2). Our model combines benefits from VAE and NVP: it uses efficient non-invertible (convolutional) layers of VAE, while using a limited number of invertible layers as in NVP to avoid factorization in the conditional distribution $p_\theta(x|z)$. An alternative interpretation of our model is to see it as a variant of NVP with a complex non-parametric prior distribution rather than a unit Gaussian. The Jacobian in Eq. (6.3) pushes the model to increases the volume around training images in feature space, and the VAE measures their density in that space. Experimentally, we find our partially invertible non-factorial decoder to improve both sample quality as well as the likelihood of held-out data.

6.1.2 Improving samples with adversarial training

When optimizing $\mathcal{L}_C(p_\theta)$, the regularization term $D_{KL}(q_\phi(z|x)||p_\theta(z))$ drives the posterior $q_\phi(z|x)$ and the prior $p_\theta(z)$ closer together. Ideally, the posterior marginalized across real images and the prior match, i.e. $p^*_\theta(z) = \int_x p^*(x) q^*_\phi(z|x)$. In this is the case, latent variables $z \sim p_\theta(z)$, and mapped through the feedforward decoder, should result in realistic samples. Adversarial training be leveraged for quality-driven training of the prior, thus enrich its training signal as previously discussed.

For quality-driven training, we train a discriminator using the modified objective proposed by [171] that combines both generator losses considered by [59]:

$$\mathcal{L}_Q(p_\theta) = -\mathop{\mathbb{E}}_{p_\theta(z)} \ln \frac{D(G_\theta(z))}{1 - D(G_\theta(z))}. \tag{6.4}$$

Assuming the discriminator D is trained to optimality at every step, it is easy to demonstrate that the generator is trained to optimize $\mathrm{D_{KL}}(p_\theta \,||\, p^*)$. To regularize the training of the discriminator, we use the gradient penalty introduced by [64], see App. 6.3.3 for details.

The training procedure alternates between two steps, similar to that of GANs. In the first step, the discriminator is trained to maximize $\mathcal{L}_Q(p_\theta)$, bringing it closer to it's optimal value $\mathcal{L}^*_Q(p_\theta) = \mathrm{D_{KL}}(p_\theta \,||\, p^*)$. In the second step, the generative model is trained to minimize $\mathcal{L}_C(p_\theta) + \mathcal{L}_Q(p_\theta)$, the sum of the coverage-based loss in Eq. (6.3), and the quality-based loss in Eq. (6.4). Assuming that the discriminator is trained to optimality at every step, the generator is trained to minimize a bound on the sum of two symmetric KL divergences:

$$\mathcal{L}_C(p_\theta) + \mathcal{L}^*_Q(p_\theta) \geq \mathrm{D_{KL}}(p^* \,||\, p_\theta) + \mathrm{D_{KL}}(p_\theta \,||\, p^*) + \mathcal{H}(p^*), \tag{6.5}$$

where the entropy of the data generating distribution, $\mathcal{H}(p^*)$, is an additive constant that does not depend on the learned generative model p_θ.

6.2 Experimental evaluation

Below, we present our evaluation protocol (Section 6.2.1), followed by an ablation study to assess the importance of the components of our model (Section 6.2.2). In Section 6.2.4 we improve quantitative and qualitative performance using recent advances from the VAE and GAN literature. We then compare to the state of the art on the CIFAR-10 and STL-10 datasets (Section 6.2.5), and present additional results at higher resolutions and on other datasets (Section 6.2.6). Finally, we evaluate a class-conditional version of our model using the image classification framework of [163].

6.2.1 Evaluation protocol

To evaluate the how well models cover held-out data, we use the bits per dimension (BPD) measure. This measure is defined as the negative log-likelihood on held-out data, averaged across pixels and color channels [38]. Due to their degenerate low-dimensional support, GANs do not define a valid density in the image space, which prevents measuring BPD. To endow a GAN with a full support and a valid likelihood, we train a VAE "around it". In particular, we train an isotropic noise parameter σ that does not depend on z, as in our VAE decoder, as well as an inference network. As we train these, the weights of the GAN generator are kept fixed. For both GANs and VAEs, we use the inference network to compute a lower-bound to approximate the likelihood, i.e. an upper bound on BPD.

To evaluate the sample quality, we report Fréchet Inception distance (FID) [73] and Inception score (IS) [158], which are commonly used to quantitatively evaluate GANs [22, 198]. Although IS and FID are used to evaluate sample quality, these metrics are also sensitive to coverage. In fact, any metric evaluating sample quality only would be degenerate, as collapsing to the mode of the target distribution would maximize it. However, in practice both metrics correlate stronger with sample quality than with support coverage. We evaluate all measures using held-out data not used during training, which improves over common practice in the GAN literature, where train data is often used for evaluation.

6.2.2 Comparison to GAN and VAE baselines

Experimental setup. We evaluate our approach on the CIFAR-10 dataset, using 50k/10k train/test images of 32×32 pixels (standard split). We train our GAN baseline to optimize $\mathcal{L}_Q(p_\theta)$, and use the architecture of SNGAN [115], which is stable and trains quickly. The same architecture and training hyper-parameters are used for all models in this experiments, see Section 6.3 for details.

We train our VAE baseline by optimizing $\mathcal{L}_C(p_\theta)$. We use the GAN generator architecture for the decoder, which produces the mean of a factorizing Gaussian distribution over pixel RGB values. We add a trainable isotropic variance σ, to ensure a valid density model. In the VAE model some feature maps in the decoder are treated as conditional latent variables, allowing for hierarchical top-down sampling. Experimentally, we find that similar top-down sampling is not effective for the GAN model.

To train the generator for both coverage and quality, we optimize the sum of $\mathcal{L}_C(p_\theta)$ and $\mathcal{L}_Q(p_\theta)$. We refer to the model trained in this way as CQG. We refer to model that also includes invertible layers in the decoder as CQFG. The small invertible model uses a single scale with three invertible layers, each composed of two residual blocks and increases the number of weights in the generator by roughly

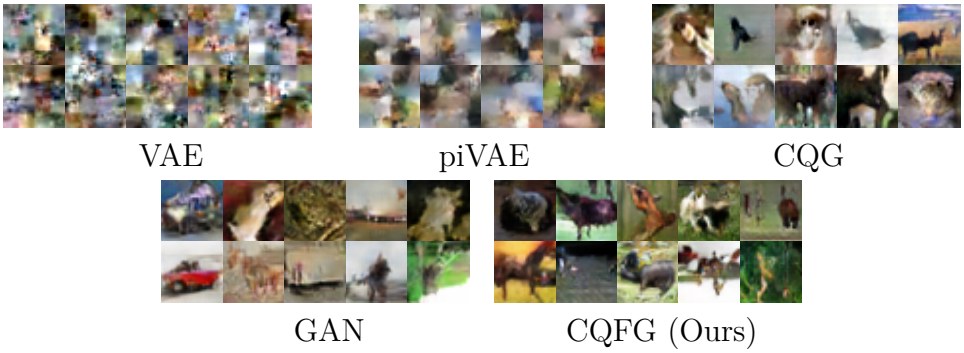

Figure 6.2 – Samples from GAN and VAE baselines, and our CQG and CQFG models, all trained on CIFAR-10.

	$\mathcal{L}_Q$	$\mathcal{L}_C$	Flow	BPD ↓	IS ↑	FID ↓
GAN	✓			[7.0]	6.8	31.4
VAE		✓		4.4	2.0	171.0
piVAE		✓	✓	3.5	3.0	112.0
CQG	✓	✓		4.4	5.1	58.6
CQFG	✓	✓	✓	3.9	7.1	28.0

Table 6.1 – Results for the GAN and VAE baselines, VAE with invertible flow layers (piVAE), and our models with (CQFG) and without (CQG) invertible layers. [Square brackets] denote that the value is approximated as described in Section 6.2.1.

1.4% so we also slightly increase the width of the generator in the CQG version for fair comparison. All implementation details can be found in Section 6.3.

Analysis of results. Form the experimental results in Table 6.1 we make several observations. As expected, the GAN baseline yields better sample quality (IS and FID) than the VAE baseline, e.g. obtaining inception scores of 6.8 and 2.0, respectively. Conversely, the VAE achieves better coverage, with a BPD of 4.4, compared to an estimated 7.0 for the GAN. The same generator trained for both quality and coverage, CQG, achieves *the same* BPD as the VAE baseline. The same quality of this model is in between that of the GAN and the VAE baselines. In Figure 6.2 we show samples from the different models, and these confirm the quantitative observations.

When adding the invertible layers to the VAE decoder, but using maximum likelihood training with $L_C(p_\theta)$ (piVAE), leads to improves sample quality with IS increasing from 2.0 to 3.0 and FID dropping from 171.0 to 112.0. Note that the quantitative sample quality is below that of the GAN baseline and our CQG

model. When we combine the non-factorial decoder with coverage and quality driven training, CQFG, we obtain quantitative sample quality that is somewhat better than that of the GAN baseline: IS improving from 6.8 to 7.1, and FID decreasing from 31.4 to 28.0. The samples in Figure 6.2 confirm the high sample quality of the CQFG model. Note that the CQFG model also achieves a better BPD than the VAE baseline. These experimental observations demonstrate the importance of our contributions: our non-factorial decoder trained for coverage and quality improves both VAE and GAN in terms of held-out likelihood, and improves VAE sample quality to, or slightly beyond, that of the GAN.

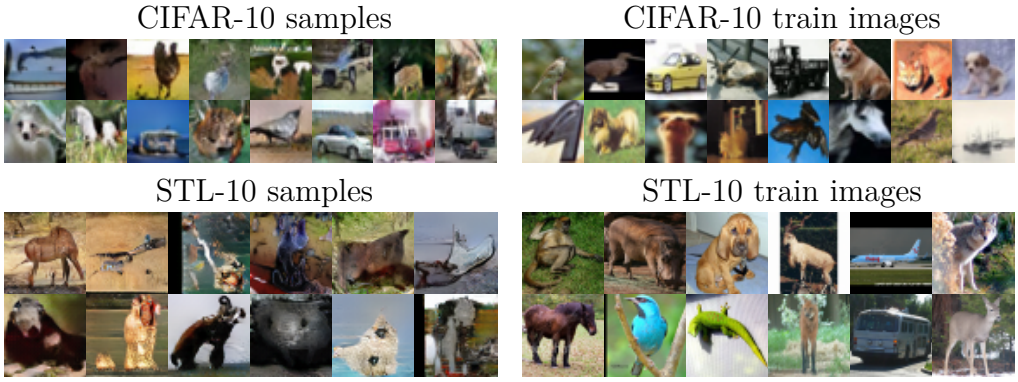

Figure 6.3 – Random samples from our CQFG (large-D) trained on the CIFAR-10 and STL-10 (48×48) datasets.

6.2.3 Qualitative influence of the feature space flexibility

In this section we experiment with different architectures to implement the invertible mapping used to build the feature space as presented in Section 6.1.1. To assess the impact of the expressiveness of the invertible model on the behavior of our framework, we modify various standard parameters of the architecture. Popular invertible models such as NVP [38] readily offer the possibility of extracting latent representation at several scales, separating global factors of variations from low level detail, thus we experiment with varying number of scales. An other way of increasing the flexibility of the model is to change the number of residual blocks used in each invertible layer. Note that all the models evaluated so far in the main body of the paper are based on a single scale and two residual blocks. In addition to our CQFG models, we also compare with similar models trained with maximum likelihood estimation (MLE). Models are trained first with maximum-likelihood estimation, then with both coverage and quality driven criterions.

The results in Table 6.2 show that factoring out features at two scales rather than one is helpful in terms of BPD. For the CQFG models, however, the IS and FID deteriorate with more scales, and so a tradeoff between must be struck. For the MLE models, the visual quality of samples also improves when using multiple scales, as reflected in better IS and FID scores. Their quality, however, remains far worse than those produced with the coverage and quality training used for the CQFG models. Samples in the maximum-likelihood setting are provided in Figure 6.4. With three or more scales, models exhibit symptoms of overfitting: train BPD keeps decreasing while test BPD starts increasing, and IS and FID also degrade.

Scales	Blocks	BPD ↓	IS ↑	FID ↓
1	2	3.77	**7.9**	**20.1**
2	2	3.48	6.9	27.7
2	4	**3.46**	6.9	28.9
3	3	3.49	6.5	31.7

(a) CQFG models

Scales	Blocks	BPD ↓	IS ↑	FID ↓
1	2	3.52	3.0	112.0
2	2	**3.41**	**4.5**	85.5
3	2	3.45	4.4	**78.7**
4	1	3.49	4.1	82.4

(b) piVAE models

Table 6.2 – Evaluation on CIFAR-10 of different architectures of the invertible layers of the model.

In Figure 6.4 we show samples obtained using VAE models trained with MLE. The models include one without invertible decoder layers, and with NVP layers using one, two and three scales. The samples illustrate the dramatic impact of using invertible NVP layers in these autoencoders.

6.2.4 Evaluation of architectural refinements

To further improve quantitative and qualitative performance, we proceed to include two recent advances in the VAE and GAN literature. First, [64] have shown a deeper discriminator with residual connections to be beneficial to training. We using such improved discriminators, we find it useful to make similar changes to

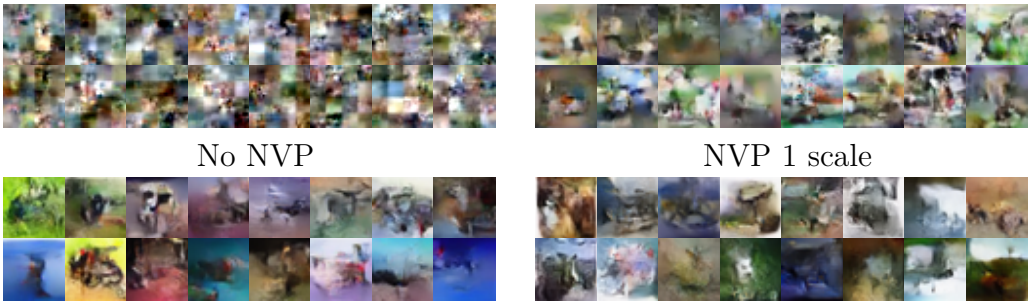

Figure 6.4 – Samples from MLE models (Table 6.2b) showing qualitative influence of multi-scale feature space.

	IAF	Res	BPD ↓	IS ↑	FID ↓
GAN			[7.0]	6.8	31.4
GAN		✓	—	7.4	24.0
CQFG			3.9	7.1	28.0
CQFG		✓	3.8	7.5	26.0
CQFG	✓	✓	3.8	7.9	20.1
CQFG (large-D)	✓	✓	3.7	8.1	18.6

Table 6.3 – Evaluation of architectures using residual (Res) layers and inverse autoregressive flow (IAF) posterior approximation.

the generator to mirror these modifications. Second, [83] improve VAE encoders by introducing inverse auto-regressive flow (IAF) to allow for more accurate posterior approximations that go beyond factorized Gaussian approximations that are commonly used.

The results in Table 6.3 show consistent improvements across all metrics when adding residual connections and IAF. Increasing the size of the discriminator (denoted "large D") yields further improvements in IS and FID, while slightly degrading the BPD from 3.77 to 3.74. These results show that our model benefits from recent architectural advances in GANs and VAEs. In the remainder of our experiments we use the CQFG (large D).

6.2.5 Comparison to the state of the art

In Table 6.4 we compare the performance of our models with previously proposed hybrid approaches, as well as state-of-the-art adversarial and likelihood based

models. Many entries in the table are missing, since the likelihood of held-out data is not defined for most adversarial methods, and most likelihood-based models do not report IS or FID scores. We present results for two variants of our CQFG model, the large-D variant from Table 6.3, as well as a variant that uses two scales in the invertible layers rather than one, denoted "S2". See Section 6.2.3 for details. The latter model achieves better BPD at the expense of worse IS and FID.

Compared to the best hybrid approaches, our large-D model yields a substantial improvement in IS to 8.1, while our S2 model yields a comparable value of 6.9. Compared to adversarially trained models, our large-D model obtains results that are comparable to the best results obtained by SNGAN using residual connections and hinge-loss. We note that the use of spectral normalization and hinge-loss for adversarial training could potentially improve our results, but we leave this for future work. Our S2 model is comparable to the basic SNGAN (somewhat better FID, somewhat worse IS) that does not use residual connections and hinge-loss. On STL-10 (48×48 pixels) our models trained using 100k/8k train/test images, also achieve competitive IS and FID scores; being only outperformed by SNGAN (Res-Hinge).

Using our S2 model we obtain a BPD of 3.5 that is comparable to Real-NVP, while our large-D model obtains a slightly worse value of 3.7. We computed IS and FID scores for VAE-IAF and PixelCNN++ using publicly released code and parameters. We find that these IS and FID scores are substantially worse than the ones we measured both of our model variants. To the best of our knowledge, we are the first to report BPD measurements on STL-10, and can therefore not compare to previous work in this metric.

We display samples from our CQFG (large-D) model on both datasets in Figure 6.3.

6.2.6 Results on additional datasets

To further validate our approach we train our CQFG (large-D) model on three additional datasets, and on STL-10 at 96×96 resolution. The architecture and training procedure are unchanged from the preceding experiments, up to the addition of convolutional layers to adapt to the increased resolution. For the CelebA dataset we used 196k/6.4k train/test images, resized to 96×96, and used central image crops of both 178×178 and 96×96 pixels. We also train on STL-10 (100k/8k train/test images) resized to 96×96, on the LSUN-bedrooms dataset (3M/300 train/test images) at 64×64 resolution, and on ImageNet (1.2M/50k train/test images) resized to 64×64 pixels.

We show samples and train images for these datasets in Figure 6.5, and quantitative evaluation results in Table 6.5. The fact that our model works without changing the architecture and training hyper-parameters shows the stability of

	CIFAR-10			STL-10		
	BPD ↓	IS ↑	FID ↓	BPD ↓	IS ↑	FID ↓
Hybrid models						
AGE		5.9				
ALI		5.3				
SVAE		6.8				
α-GAN		6.8				
SVAE-r		7.0				
CQFG (Ours)	3.7	8.1	18.6	4.0	8.6	52.7
CQFG (S2) (Ours)	3.5	6.9	28.9	3.8	8.6	52.1
Adversarial models						
SNGAN		7.4	29.3		8.3	53.1
BatchGAN		7.5	23.7		8.7	51
WGAN-GP		7.9				
SNGAN (Res-Hinge)		8.2	21.7		9.1	40.1
Likelihood-based models						
Real-NVP	3.5					
VAE-IAF	3.1	[3.8]	[73.5]			
PixelCNN++	2.9	[5.4]	[121.3]			

Table 6.4 – Comparison of our models on CIFAR-10 and STL-10 (48×48) with state-of-the-art likelihood based, adversarial and hybrid generative models. [Square brackets] denote that we computed the values using samples obtained with the code and checkpoints released by the authors of the corresponding models. AGE: [186], ALI: [42], SVAE: [29], SNGAN: [115], BatchGAN: [103], WGAN-GP: [64], NVP: [38], VAE-IAF: [83], PixelCNN++: [159], α-GAN: [154].

our approach. On the CelebA and LSUN datasets, our CQF generator produces compelling samples despite the high resolution of the images. The samples for STL-10 and ImageNet are less realistic due to the larger variability in these datasets; recall that we do not condition on class labels for generation. On CelebA, all scores improve significantly when using central crops of 96×96, due to the reduced variability in the smaller crop which removes the background from the images.

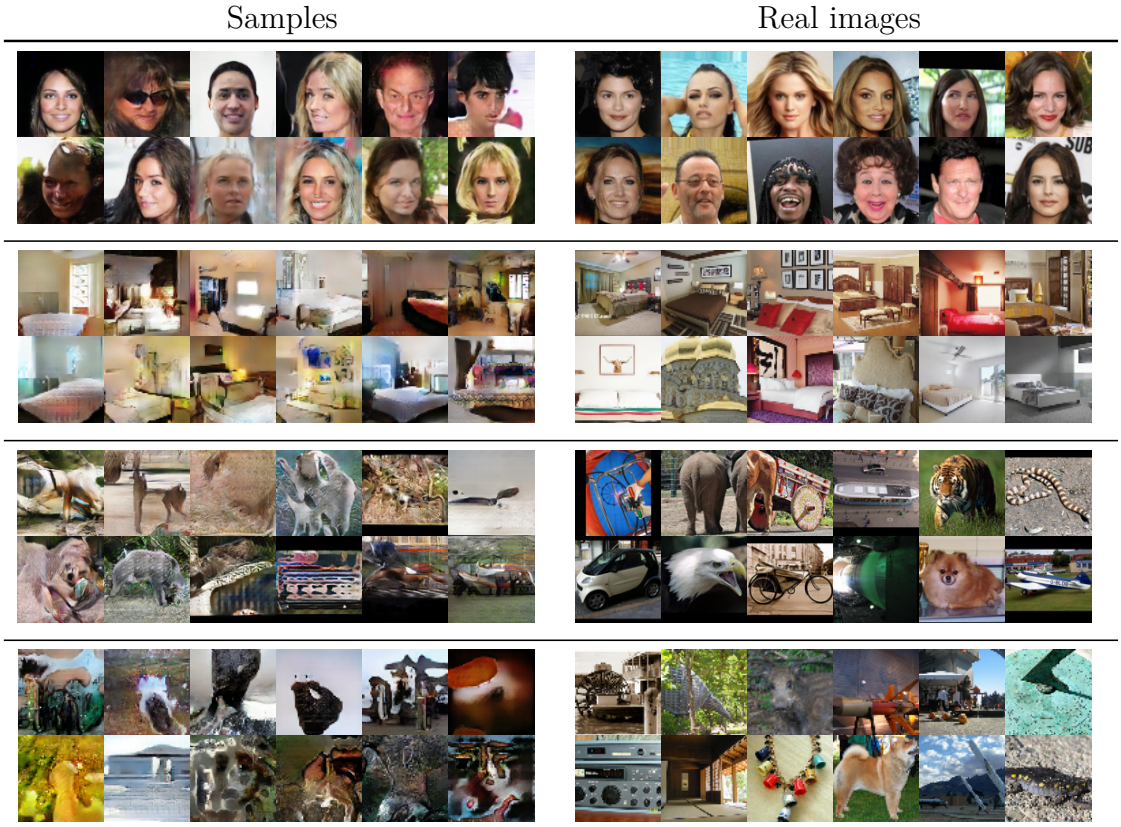

Figure 6.5 – Samples and train images of CelebA (crop 178), LSUN-Bedrooms, STL-10 96×96, and ImageNet.

6.2.7 Class-conditional results

We also evaluate a class-conditional model and rely on two measures recently proposed by [163]. The first measure, GAN-test, is obtained by training a classifier on natural image data and evaluating it on the samples of a class-conditional generative model. This measure is sensitive to sample quality only. The second measure, GAN-train, is obtained by training a classifier on generated samples and evaluating it on natural images. This measure requires is sensitive both to quality and coverage. For a given GAN-test level, variations in GAN-train can be attributed to different coverage.

To perform this evaluation we develop a class conditional version of our CQFG model. The discriminator is conditioned using the class conditioning introduced by [114]. GAN generators are typically made class-conditional using conditional batch normalization [35,44], however batch normalization is known to be detrimental

	Resolution	BPD↓	IS↑	FID↓
CelebA, crop 178	96×96	2.85	—	24.3
CelebA, crop 96	96×96	2.45	—	13.8
STL-10	96×96	3.85	8.8	100.8
ImageNet	64×64	4.90	7.6	69.9
LSUN-Bedrooms	64×64	4.01	—	61.9

Table 6.5 – Quantitative evaluation of our CQFG (large-D) model on additional datasets. IS is not reported on CelebA and LSUN because it is not informative on these datasets.

in VAEs [83], as we verified in practice. To address this issue, we propose conditional weight normalization (CWN). As in weight normalization [160], we separate the training of the scale and the direction of the weight matrix. Additionally, the scaling factor $g(y)$ of the weight matrix $\mathbf{v}$ is conditioned on the class label y:

$$\mathbf{w} = \frac{g(y)}{\|\mathbf{v}\|}\mathbf{v}, \tag{6.6}$$

We also make the network biases conditional on the class label. Otherwise, the architecture is the same one used for the experiments in Table 6.1.

In Table 6.6 we report the GAN-train and GAN-test measures on CIFAR-10. Our CQFG model obtain a slightly higher GAN-test score than the GAN baseline, which shows that it achieves comparable if not better sample quality, which is inline with the results in terms of IS and FID scores in Section 6.2.2. Moreover, with CQFG we obtain a substantially better GAN-train score, going from 29.7 to 73.4. Having established similar GAN-test performance, this demonstrates significantly improved sample diversity of the CQFG model as compared to the GAN baseline. This shows that the coverage-driven training improves the coverage of the learned model.

6.3 Model refinements and implementation details

In this section we give more details about architectural changes made in Section 6.2.4 as well as about implementation, in general.

6.3.1 Top-down sampling of hierarchical latent variables

Flexible priors and posteriors for the variational autoencoder model can be obtained by sampling hierarchical latent variables at different layers in the network.

model	GAN-test (%)	GAN-train (%)
GAN	71.8	29.7
CQFG	**76.9**	**73.4**
DCGAN†	58.2	65.0

Table 6.6 – GAN-test and GAN-train measures for class conditional CQFG and GAN models on CIFAR-10. The performance of the DCGAN† model, though not directly comparable, is provided as a reference point.

In the generative model p_θ, latent variables $\mathbf{z}$ can be split into L groups, each one at a different layer, and the density over $\mathbf{z}$ split thus:

$$q(\mathbf{z}) = q(\mathbf{z}_L) \prod_{i=1}^{L-1} q(\mathbf{z}_i|\mathbf{z}_{i+1})$$

Additionally, to allow the chain of latent variables to be sampled in the same order when encoding-decoding and when sampling, top-down sampling is used, as proposed in [13, 83, 172]. With top-down sampling, the encoder (symmetric to the decoder) extracts deterministic features h_i at different levels as the image is being encoded, constituting the bottom-up deterministic pass. While decoding the image, these previously extracted deterministic features h_i are used for top-down sampling and help determining the posterior over latent variables at different depths in the decoder. These posteriors are also conditioned on the latent variables sampled at lower feature resolutions, using normal densities:

$$\begin{aligned} q_\phi(\mathbf{z}_1|x) &= \mathcal{N}(\mathbf{z}_1|\mu_1(x, h_1), \sigma_1^2(x), h_1) \\ q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1}) &= \mathcal{N}(\mathbf{z}_i|\mu_i(x, \mathbf{z}_{i-1}, h_{i-1}), \sigma_i^2(x, \mathbf{z}_{i-1}, h_{i-1})) \end{aligned}$$

This constitutes the stochastic top-down pass.

We refer the reader to [13, 83, 172] for more detail.

6.3.2 Inverse autoregressive flow

To increase the flexibility of posteriors used over latent variables in variational inference, The authors of [83] have proposed a type of normalizing flow called inverse autoregressive flow (IAF). The main appeals of this normalizing flow are its scalability to high dimensionality and its ability to leverage autoregressive neural network (such as those introduced in [188]). First, a latent variable vector is sampled using the reparametrization trick [84]:

$$\epsilon \sim \mathcal{N}(0, I) z_0 = \mu_0 + \sigma_0 \epsilon.$$

Then mean and variance parameters μ_1 and σ_1 are computed as functions of z_0 using autoregressive models, and a new latent variable z_1 is obtained:

$$z_1 = \mu_1(z_0) + \sigma_1(z_0)z_0.$$

Because σ_1 and μ_1 are implemented by auto-regressive networks, the jacobian $\frac{dz_1}{dz_0}$ is triangular with the values of σ_1 on the diagonal and the density under the new latent variable remains efficient to compute. In theory this transformation can be repeated an arbitrary number of times for increased flexibility, in practice typically a single step is used.

6.3.3 Gradient penalty

A body of work on Generative Adversarial Networks centers around the idea of regularizing the discriminator by enforcing Lipschitz continuity, for instance in [12, 64, 115, 179]. In this work we use the approach of [64], that enforces the Lipschitz constraint with a gradient penalty term added to the loss:

$$\mathcal{L}_{Grad} = \lambda + \mathbb{E}_{\hat{x}}[(||\Delta_{\hat{x}} D(\hat{x})||_2 - 1)^2],$$

where $\hat{x}$ is obtained by interpolating between real and generated data:

$$\begin{aligned} \epsilon &\sim U_{[0,1]} \\ \hat{x} &= \epsilon x + (1-\epsilon)\tilde{x} \end{aligned}$$

We add this term to the loss used to train the discriminator that yields our quality driven criterion.

6.3.4 Architecture and training hyper-parameters

We used Adamax [81] with learning rate 0.002, $\beta_1 = 0.9$, $\beta_2 = 0.999$ for all experiments. All CIFAR-10 experiments use batch size 64, other experiments in high resolution use batch size 32. To stabilize the adversarial training we use the gradient penalty [64] with coefficient 100, and 1 discriminator update per generator update. We experimented with different weighting coefficient between the two loss components, and found that values in the range 10 to 100 on the adversarial component work best in practice. In this range, no significant influence on the final performance of the model is observed, though the training dynamics in early training are improved with higher values. With values significantly smaller than 10, discriminator collapses was observed in a few isolated cases. All experiments reported here use coefficient 100.

For experiments with hierarchical latent variables we use 32 of them per layer. In the generator we use ELU nonlinearity, in discriminator with residual blocks

we use ReLU while in simple convolutional discriminator we use leaky ReLU with slope 0.2.

Unless stated otherwise we use three NVP layers with a single scale and two residual blocks that we train only with the likelihood loss. Regardless of the number of scales, the VAE decoder always outputs a tensor of the same dimension as the target image, which is then fed to the NVP layers. Just like in reference implementations we use both batch normalization and weight normalization in NVP and only weight normalization in IAF.

We use the reference implementations of IAF and NVP released by the authors.

Discriminator	Generator
	conv 3×3, 16
	IAF block 32
conv 3×3, 16	IAF block down 64
ResBlock 32	IAF block down 128
ResBlock down 64	IAF block down 256
ResBlock down 128	$h \sim \mathcal{N}(0; 1)$
ResBlock down 256	IAF block up 256
Average pooling	IAF block up 128
dense 1	IAF block up 64
	IAF block 32
	conv 3×3, 3

Table 6.7 – Residual architectures for experiments from Table 6.3 and Table 6.2

6.4 Conclusion

We presented CQGF, a generative model that leverages invertible network layers to relax the conditional pixel independence assumption commonly made in VAE models. Since our model allows for efficient feed-forward sampling, we are able to train our model using a maximum likelihood criterion that ensure coverage of the data generating distribution, as well as an adversarial criterion that ensures high sample quality. We provide quantitative and qualitative experimental results on a collection of five datasets (CIFAR-10, STL-10, CelebA, ImageNet and LSUN-Bedrooms). We obtain IS and FID scores comparable to state-of-the-art GAN models, and held-out likelihood scores that are comparable to recent pure likelihood-based models.

Chapter 7

Conclusion

There is great potential to use computer vision technology in a constructive and benevolent way.

— Fei Fei Li

Contents

7.1 Summary of contributions **86**

7.1.1 Incremental learning 86

7.1.2 Evaluation of GANs 86

7.1.3 Hybrid generative model 86

7.2 Future work **87**

7.2.1 Comprehensive benchmark on incremental learning . . 87

7.2.2 Critical periods in deep networks 88

7.2.3 Catastrophic forgetting in mainstream applications . . 89

7.2.4 Autoregressive decoder in CQFG 90

In this chapter we summarize our contributions in incremental learning and generative modeling in Section 7.1. We conclude with ideas for future research in Section 7.2.

7.1 Summary of contributions

7.1.1 Incremental learning

We have introduced a framework to enable class-incremental learning for object detection in Chapter 3. Our model is validated for iterated single-class and multiple-class addition scenarios in PASCAL VOC 2007 and MS COCO datasets. The core idea is replay of samples corresponding to previously learned classes with a training procedure based on knowledge distillation. Our framework does not store old samples for replay, and instead mines them on the fly from new training images, using a frozen copy of the network obtained from the preceding training steps. This allowed us to obtain good results without keeping old samples for replay even in the extreme scenario of adding classes one by one.

Our framework provides a solution to challenges specific to incremental learning of object detection, e.g., background class, imbalanced class distribution of outputs, catastrophic forgetting in bounding box regression module. It is universal enough to be adapted for recent object detection systems with sliding window or learned proposals, e.g., SSD [99] and Faster-RCNN [148].

7.1.2 Evaluation of GANs

We developed a framework to evaluate class-conditional GANs beyond commonly used Inception-based metrics. In Chapter 5 we presented two measures, GAN-train and GAN-test, that allow us to separately evaluate image quality and image diversity. Both measures are computed using a separate CNN classifier. GAN-test is the accuracy of the CNN trained on real images, and evaluated on GAN-generated ones. It shows how similar generated images are to real ones, even if there are very few distinct generated images. GAN-train is the accuracy of the same CNN, but trained on GAN-generated samples and evaluated on real data. To obtain high GAN-train accuracy the model should not only have good samples, but also generate many different samples to prevent overfitting from crippling the classifier. Both measures are complementary, enable a more comprehensive view on the performance of GAN models, and can detect "mode collapse" (models that generate very few, high quality samples), a common GAN problem.

7.1.3 Hybrid generative model

Motivated by the mode collapse problem [12], we built a new generative model that combines the sample quality of adversarial training and the mathematical guarantees of likelihood-based models. In Chapter 6, we presented this model based on VAE architecture, together with a Real-NVP induced feature space.

It alleviates the unrealistic assumption of independence in RGB space, which is common for VAE models, and significantly improves the quality of samples. Adversarial training allows to obtain Inception score and FID comparable with state-of-the-art GAN models and alleviate overgeneralization typically encountered in likelihood-based models, e.g., VAE, Real-NVP. At the same time state-of-the-art VAE architecture (IAF) allows our model to achieve perfect reconstructions and competitive test likelihood. We provide extensive ablation studies on CIFAR-10 as well as quantitative and qualitative evaluation on CelebA, STL-10, ImageNet and LSUN-Bedrooms datasets.

7.2 Future work

7.2.1 Comprehensive benchmark on incremental learning

As we discussed in Chapter 2, catastrophic forgetting and incremental learning are not reduced one to another nor can be used interchangeably. Several researchers believe that catastrophic forgetting is the only obstacle for incremental learning. While it is the main issue, it is not the only one. Thus, solutions evaluated on catastrophic forgetting benchmarks are assumed to perform well on incremental learning problems as well. In practice, it is seldom the case. In order to facilitate tracking of progress in the field we need a comprehensive and well-defined evaluation framework that specifically targets incremental learning challenges.

A common benchmark for incremental learning is to split a dataset into number of class groups, for example, CIFAR-100 split in 20 groups of 5 classes, and consider each group as a separate training stage. In the end, we want a network architecturally similar to the one we obtain using offline training, i.e., having a single output layer for 100 classes and capable of identifying any test image without any additional information. However, some papers targeting catastrophic forgetting implement this benchmark as a sequence of 20 independent tasks. So, at the end of the training phase, the resulting network can identify an image only when a task descriptor (number of class group) is provided. *Essentially it corresponds to classifying an image as one of 5 classes using prior information to what split it belongs.* Architecturally, such a network has multiple output layers because different splits are treated as independent tasks.

This formulation is significantly easier than any incremental learning benchmark, and often provides a false impression that ideas to solve catastrophic forgetting can be trivially applied to incremental learning challenges. Another pitfall is to allow the network to revisit old tasks and relearn forgotten classes. It makes performing well on the benchmark easier, but the exact protocol detailing the number and order of revisits is often described too briefly, and complicates fair comparison

between approaches.

These issues are due to the experimental methodology, not a particular dataset. Incremental learning benchmarks should be designed differently than those for catastrophic forgetting, which are usually a sequence of independent classification tasks with task descriptors available at test time. Given our experience with object detection incremental learning in Chapter 3 we suggest to begin with the image classification task as it is easier to analyze mathematically and fundamental for a variety of applications.

We argue that a good benchmark on class-incremental learning should satisfy following criteria:

— no task descriptors at test time,
— a single output layer for all the learned classes,
— no revisits of old tasks,
— enough classes to create a meaningful sequence of tasks (10 classes is a really lower limit),
— enough data per class to enable incremental learning without having to constantly fight overfitting.

Evaluation procedure for incremental learning should be kept separate from catastrophic forgetting benchmarks because small technicalities (task descriptors available at test time, task revisits with unclear order) often create a false impression that existing solutions are on par with offline training.

7.2.2 Critical periods in deep networks

A recent work [6] demonstrates that critical development periods common for biological neural networks also arise in artificial neural networks. In particular, it is shown that a network experiencing a degradation of input data (e.g., blur) in the very beginning of training never recovers completely even if it is given enough iterations. It is strikingly similar to mammals that never develop binocular vision if they had one eye blocked during first few weeks or months of their life [118]. Further, the authors show that the critical period is followed by a period of compression (decreasing of information stored in weights). Empirical decomposition of the training process agrees well with information bottleneck theoretical analysis in [166] that divides the SGD optimization in two phases: drift phase (big gradients with low SNR) and diffusion phase (small gradients with high SNR). The latter takes significantly more epochs and corresponds to representation compression.

We speculate that the interplay between these periods may be connected to incremental learning. When the critical period is already closed, learning new classes substantially different from already learned ones results in low performance on the new classes. On the other hand, moving to new classes before the end of critical period results in inferior performance on the old classes (due to lack of

compression period). There is a chance that the critical period is a fundamental problem arising from information theory, and looking for a way to switch between these periods back and forth seems to be a promising avenue in incremental learning research.

It is interesting to note that the existence of critical periods does not complicate incremental learning for humans. This is likely due to critical periods mainly affecting low-level visual representations that are learned and fixed early in life. It would mean that these representations are sufficient to navigate the visual world and incrementally learn to recognize new object classes never seen before. Studying the limits of incremental learning in humans may shed light on this question. Learning to process other forms of visual data (for example, biomedical imaging data) often takes a lot of time for people (up to a few decades of training in some domains). This indirect evidence suggests that incremental learning without assumptions on data similarity becomes much harder once we are past the critical period. If it is indeed the case, then pre-training on datasets of even larger scale than ones we use today can be a solution for deep networks.

7.2.3 Catastrophic forgetting in mainstream applications

There is an interesting facet of catastrophic forgetting that is often overlooked in literature: catastrophic forgetting during training of the same task. In general, catastrophic forgetting is an issue when the distribution of input data shifts during training. Usually deep networks are trained using mini-batches uniformly sampled from a large dataset. However, a mini-batch is just an approximation of true data distribution, there is nothing that prevents the network from forgetting information learned from mini-batches in previous iterations. It is a case of shifting distribution, and it inevitably hinders the learning process. For example, a classifier on ImageNet is typically trained with relatively small mini-batches (256 samples). Therefore, the network observes each class on average only 1 batch out of 4. From this point of view large batches may enable convergence within fewer steps because of reduced catastrophic forgetting between batches [62].

In [182] it was empirically demonstrated that forgetting of individual training examples occurs in practice, and has interesting regularities, e.g., some samples are never forgotten once learned, while others are frequently forgotten and this property of individual images (if they are ever forgotten during training) is conserved across multiple random seeds and different architectures!

Another example of learning from an online stream of data shifting over time is GANs: generator output shifts over training, discriminator forgets old images and then generator cycles back to forgotten samples. In [178] this effect was demonstrated on a toy dataset, and hypothesized that it may contribute to instabilities of

GAN training as well as provide an intuition why certain heuristics help convergence of GANs.

Catastrophic forgetting within a simple task due to training distribution shift is another challenging case: any embodied agent learning from real environment will be confronted with the problem that a stream of data is not i.i.d. It provides an interesting connection between catastrophic forgetting and mainstream applications of deep networks, and suggests that even an imperfect catastrophic solution may alleviate example forgetting and accelerate convergence of CNN classifiers and GANs.

7.2.4 Autoregressive decoder in CQFG

In the CQFG (coverage-quality-feature space generator) generative model presented in Chapter 6, we used a VAE decoder due to its sampling speed. Fast sampling is important to assure adversarial training as it requires evaluation of generated samples in every iteration. However, it is interesting to replace VAE in CQFG with an autoregressive decoder, which provides a more flexible model, as discussed in Chapter 4. Autoregressive models do not use the independence assumption in the output space unlike VAE. Empirically state-of-the-art autoregressive models produce images of higher quality and achieve better likelihood than VAE as shown in Table 6.4. However, sampling is significantly slower, which is not a problem for purely coverage-based training, but an issue for our model as it slows down adversarial training significantly.

An interesting solution to accelerate sampling from PixelCNN is suggested in [147]: partial relaxation of the main assumption. This way pixels close to each other are modeled with autoregressive decomposition while pixels far away are assumed independent. To an extent, it corresponds to our intuition about natural images: neighboring pixels are indeed more correlated than pixels far away. This allows a significant speedup of sampling: $O(\log N)$ compared to $O(N)$ for regular PixelCNN, where N is a number of pixels.

It is plausible that Real-NVP will remain useful for relaxed PixelCNN decoder: some independence is still present in the decoder output. However, the Real-NVP part can be made much smaller, and thus faster and easier to train because it is optimized in a limited function space due to the bijection requirement.

We hope such a modification of CQFG will lead to better-quality images (potentially even surpassing GANs) and better likelihood, much closer to the state-of-the-art coverage-based models without making training impossibly long.

Appendix A

Publications

This thesis has led to several publications summarized below.

— Konstantin Shmelkov, Cordelia Schmid, Karteek Alahari.
Incremental Learning of Object Detectors without Catastrophic Forgetting.
Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017. [164], https://hal.archives-ouvertes.fr/hal-01573623/document

— 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. [45], https://hal.archives-ouvertes.fr/hal-01573361/document (not included in the thesis)

— Konstantin Shmelkov, Cordelia Schmid, Karteek Alahari.
How good is my GAN?
Proceedings of the European Conference on Computer Vision (ECCV), 2018. [163], https://hal.archives-ouvertes.fr/hal-01850447/document

— Thomas Lucas[1], Konstantin Shmelkov[1], Karteek Alahari, Cordelia Schmid, Jakob Verbeek.
Adversarial training of partially invertible variational autoencoders, technical report, 2019. https://hal.archives-ouvertes.fr/hal-02007787/document

1. Equal contribution

Appendix B

Software

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

— Source code of incremental learning in object detection:
https://github.com/kshmelkov/incremental_detectors

— Source code of BlitzNet:
https://github.com/dvornikita/blitznet/

— Source code of GAN-train/GAN-test measures:
https://github.com/kshmelkov/gan_evaluation

Appendix C

BlitzNet: a real-time deep network for scene understanding

The following presents an ICCV 2017 paper [45] I co-authored with another PhD student Nikita Dvornik during my PhD.

C.1 Introduction

Object detection and semantic segmentation are two fundamental problems for scene understanding in computer vision. 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 [49, 86, 177, 193].

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 C.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 [132]. The 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.

(a) The outputs of object detection. (b) The results of semantic segmentation.

Figure C.1 – The outputs of our pipeline.

As most recent image recognition pipelines, our approach is based on convolutional neural networks [91], which are widely adopted for object detection [57] and semantic segmentation [101]. More precisely, deep neural networks were first used as feature extractors to classify a large number of candidate bounding boxes [57], which is computationally expensive. The improved version [56] reduces the computational cost but relies on shallow techniques for extracting bounding box proposals and does not allow end-to-end training. This issue was later solved in [148] by making the object proposal mechanism a part of the neural network. Yet, the approach remains expensive and relies on a region-based strategy (see also [94]) that makes the network architecture inappropriate for semantic segmentation.

To match the real-time speed requirement, we choose instead to base our work on the Single Shot Detection (SSD) [99] 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, imposes less constraints on the network architecture 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 [101]. Specific to semantic segmentation, [101] 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 [153]. 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 the multi-task scene understanding problem consisting of joint object detection and semantic segmentation. For that purpose, we propose a novel pipeline called BlitzNet, which will be released 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, the computational cost is reduced, and we show also that the two tasks benefit from each other in terms of accuracy.

The chapter is organized as follows: Section C.2 discusses related work; Section C.3 presents our real-time multi-task pipeline called BlitzNet. Finally, Section C.4 is devoted to our experiments, and Section C.5 concludes the chapter.

C.2 Related work

Before we introduce our approach, we now present techniques for object detection, semantic segmentation, and previous attempts to combine both tasks.

Object detection. The field of object detection has been recently dominated by variants of the R-CNN architecture [56, 148], where bounding-box proposals are independently classified by a convolutional neural network, and then filtered by a non-maximum suppression algorithm. It provides great accuracy, but relatively low inference speed since it requires a significant amount of computation per proposal. R-FCN [94] is a fully-convolutional variant that further improves detection and significantly reduces the computational cost per proposal. Its region-based mechanism is however dedicated to object detection only.

SSD [99] is a recent state-of-the-art object detector, which uses a sliding window approach instead of generated proposals to classify all boxes directly. SSD creates a scale pyramid to find objects of various sizes in one forward pass. Because of its speed and high accuracy, we have chosen to build our work on, and subsequently improve, the SSD approach. Finally, YOLO [145, 146] also provides real-time object detection and shares some ideas with SSD.

Semantic segmentation and deconvolutional layers. Deconvolutional architectures consist of adding to a classical convolutional neural networks with feature pooling, a sequence of layers whose purpose is to increase the resolution of the output feature maps. This idea is natural in the context of semantic segmentation [125], since segmentation maps are expected to have the same resolution as input images. Yet, it was also successfully evaluated in other contexts, such

as pose estimation [124], and object detection, as extensions of SSD [54] and Faster-R-CNN [97].

Joint semantic segmentation and object detection. The idea of joint semantic segmentation and object detection was investigated first for shallow approaches in [49, 61, 120, 193]. There, it was shown that learning both tasks simultaneously could be better than learning them independently.

More recently, UberNet [86] integrates multiple vision tasks such as semantic segmentation and object detection into a single deep neural network. The detection part is based on the Faster R-CNN approach and is thus neither fully-convolutional nor real-time. Closely related to our work, but dedicated to autonomous driving, [177] also proposes to integrate semantic segmentation and object detection via a deep network. There, the VGG16 network [169] is used to compute image features (encoding step), and then two different sub-networks are used for the prediction of object bounding boxes and segmentation maps (decoding).

Our work is inspired by these previous attempts, but goes a step further in integrating the two tasks, with a fully convolutional approach where network weights are shared for both tasks until the last layer, which has advantages in terms of speed, feature sharing, and simplicity for training.

C.3 Scene understanding with BlitzNet

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

C.3.1 Global view of the pipeline

The joint object detection and segmentation pipeline is presented in Figure C.2. The input image is first processed by a convolutional neural network to produce 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 [70] 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 [99]. Inspired by the hourglass architecture [124] for pose estimation and an earlier work on semantic segmentation [125], the feature maps are then up-scaled via deconvolutional layers in order to predict subsequently precise segmentation maps. Recent DSSD approach [54] uses a similar strategy for object detection the top part of our architecture presented in Figure C.2 may be seen as a variant of DSSD with a

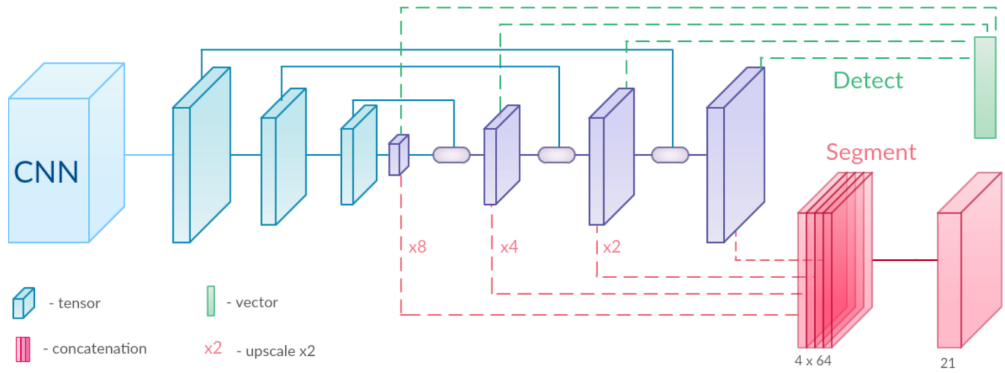

Figure C.2 – The BlitzNet architecture, which performs object detection and segmentation with one fully convolutional network. On the left, CNN denotes a feature extractor, here ResNet-50 [70]; it is followed by the downscale-stream (in blue) and the last part of the net is the upscale-stream (in purple), which consists of a sequence of deconvolution layers interleaved with ResSkip blocks (see Figure C.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.

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.

C.3.2 SSD and downscale stream

The Single Shot MultiBox Detector [99] 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 [99], the base network VGG-16 [169] 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 [99], 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 C.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.

C.3.3 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. For semantic segmentation, precise localization is equally important, and [125] proposes to use deconvolutional operations to solve that issue. Later, this process was improved in [124] by adding skip connections. Apart from combining high- and low-level features it also eases the learning process [70].

Like [54] for object detection and [124] for pose estimation, we also use such a mechanism with skip connections that combines feature maps from the downscale and upscale streams (see Figure C.2). More precisely, maps from the downscale and upscale streams are combined with a simple strategy, which we call ResSkip, presented in Figure C.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 benefits of this topology will be justified and discussed in more details in the experimental section.

C.3.4 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. Object detection is performed by a single convolutional layer that predicts a class and coordinate corrections 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 upscale all the activations of the upscale stream, concatenate them and feed to the final classification layer.

C.3.5 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

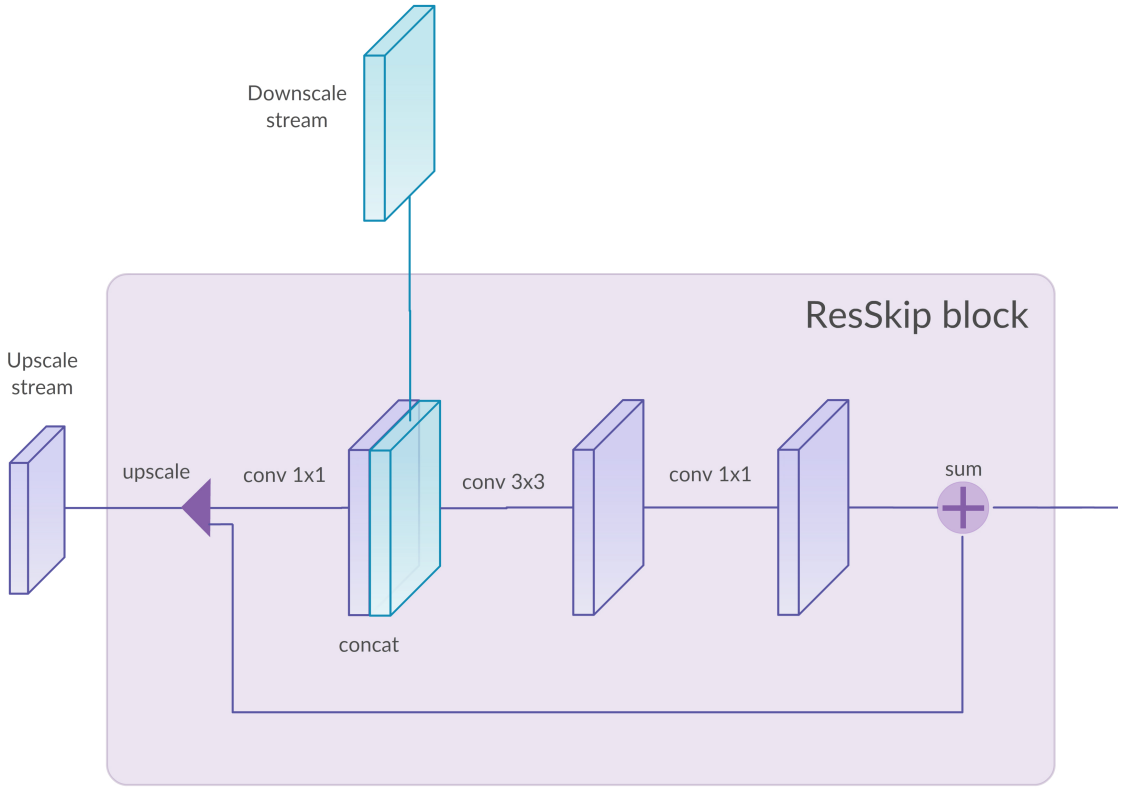

Figure C.3 – ResSkip block integrating feature maps from the upscale and downscale streams, with skip connection.

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.

C.3.6 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 [27]. 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. This 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 [99] 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.

C.4 Experiments

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 C.4.1 discusses in more details the datasets and the metrics we used; Section C.4.2 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/`.

C.4.1 Datasets and metrics

We use the COCO [98], VOC07, and VOC12 datasets [46]. 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 [67], 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 [67], 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).

C.4.2 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 [81], with a mini-batch size of 32 images. The initial 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 [70] 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.

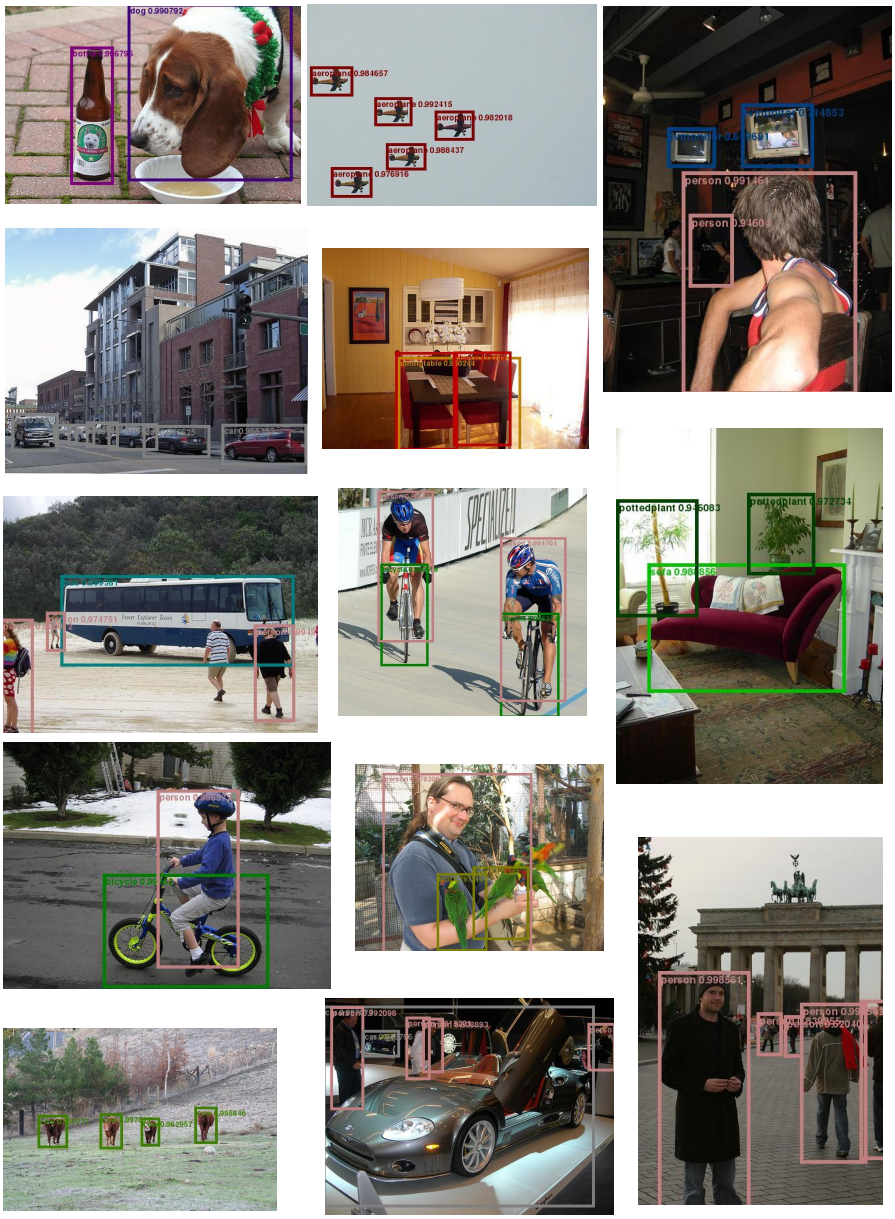

Figure C.4 – **Qualitative results for the taks of object detection.** The results are obtained by the BlitzNet512 trained on VOC07 and VOC12 train-val augmented with extra segmentation masks.

network	backbone	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
SSD300* [99]	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* [99]	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 [94]	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 [145]	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 C.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 [67]. 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* [99]	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 [94]	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 [145]	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* [99]	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 C.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 [67]. 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.

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

Table C.3 – **The effect of joint learning on both tasks.** The networks where trained on VOC12 train-seg-aug, and tested on VOC12 val.

C.4.3 PASCAL VOC 2007

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 C.1. For experiments that involve segmentation, we

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

Table C.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(%).

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 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.

C.4.4 PASCAL VOC 2012

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 C.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.

To confirm this fact, 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

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

Table C.5 – **The effect of joint training tested on COCO minival2014.** The networks were trained on COCO train.

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 C.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").

to train for segmentation, we set the corresponding loss to 0 and do not back propagate with respect to these images, otherwise we use all images that have target segmentation masks in a batch to update the weights. The results presented in Table C.4 show an improvement of 3.3%. Detection also improves in mAP by 0.6%. Figure C.5 shows that extra data for detection helps to improve classification results and to mitigate confusion between similar categories. In Table C.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 the supplementary material.

network	backbone	mAP %	FPS	# proposals	input resolution
Faster-RCNN [148]	VGG-16	73.2	7	-	$\sim 1000 \times 600$
R-FCN [94]	ResNet-101	80.5	9	-	$\sim 1000 \times 600$
SSD300* [99]	VGG-16	77.1	46	8732	300×300
SSD512* [99]	VGG-16	80.6	19	24564	512×512
YOLO [145]	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 C.7 – Comparison of inference time on PASCAL VOC 2007, when running on a Titan X (Maxwell) GPU.

Block type	mAP	mIoU
Hourglass-style [124]	78.7	75.6
Refine-style [136]	78.0	**76.1**
ResSkip (no res)	78.4	75.3
ResSkip (ours)	**79.1**	75.7

Table C.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. C.4.1. Detection performance is measured in average precision (%) and mean IoU is the metric for segmentation segmentation(%).

C.4.5 Microsoft COCO dataset

To further validate the proposed framework, we conduct experiments on the COCO dataset [98]. Here, as explained in Section C.4.1, 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 C.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 C.6. Our results are also publicly available on the COCO evaluation test server.

C.4.6 Inference speed comparison

In Table C.7 and Figure C.6, we report speed comparison to other state-of-the-art detection pipelines. Our approach is the most accurate among the real time 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).

C.4.7 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-

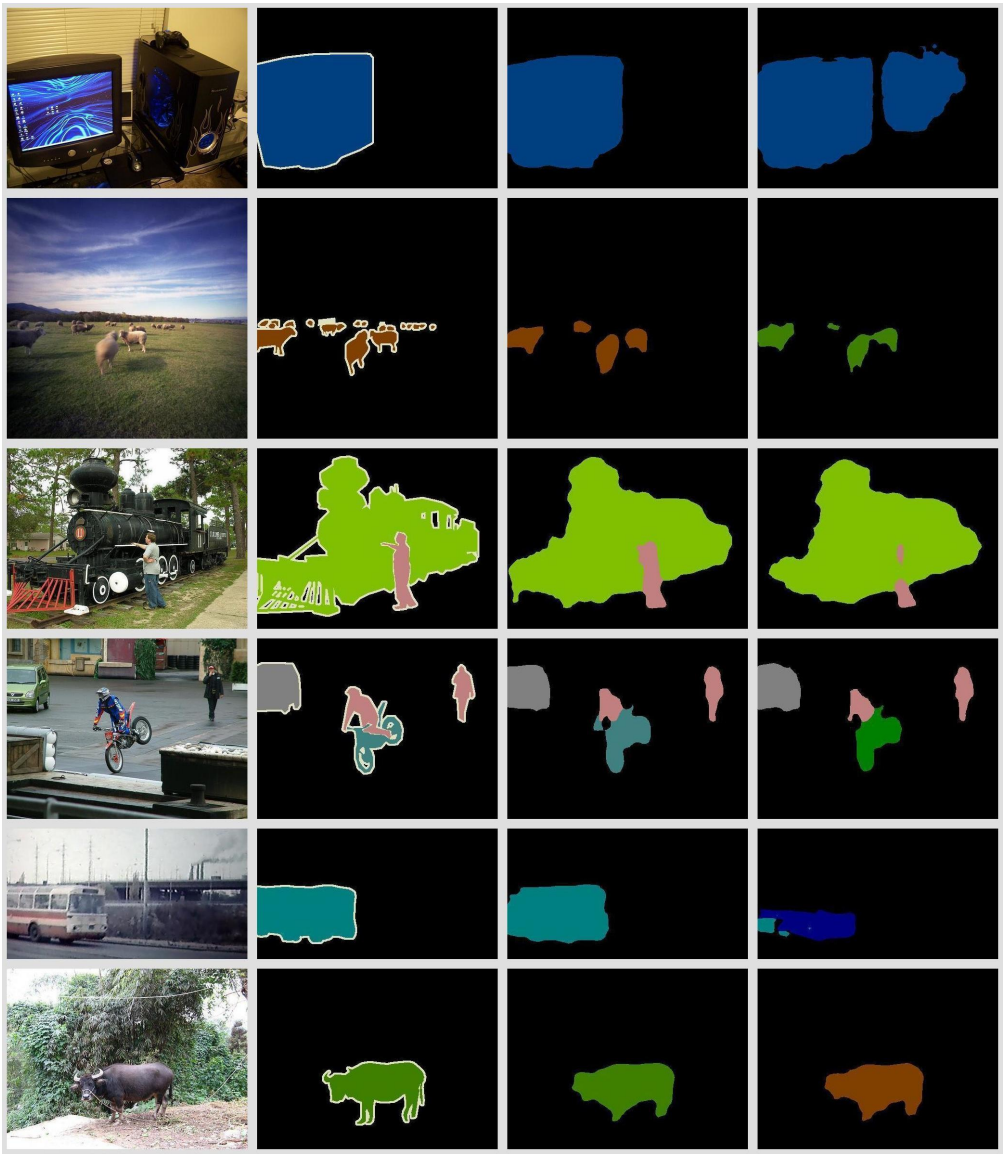

Figure C.5 – **Effect of extra data annotated for detection on the quality of estimated segmentation masks.** The first column displays test images; the second column contains its segmentation ground truth masks. The third column corresponds to segmentations predicted by BlitzNet300 trained on VOC12 train-segmentation augmented with extra segmentation masks and VOC07. The last row is segmentation masks produced by the same architecture but trained without VOC07.

art approaches on semantic segmentation. [124] [136] [153]. Table C.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 [99].

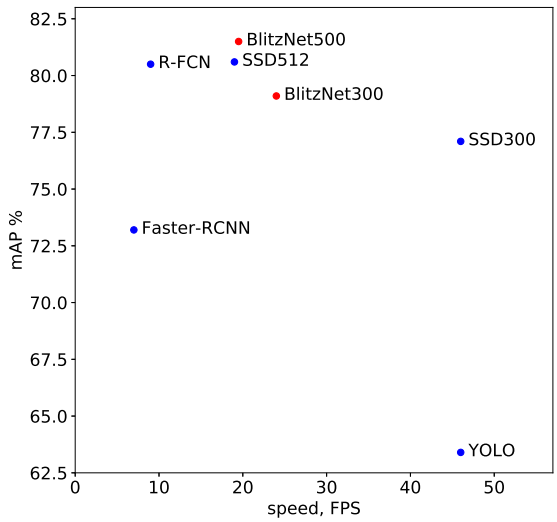

Figure C.6 – **Speed comparison with other methods.** The detection accuracy of different methods measured in mAP is depicted on y-axis. x-coordinate is their speed, in FPS.

C.5 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. Moreover, we show that our pipeline is competitive in terms of accuracy, and that the two tasks benefit from each other.

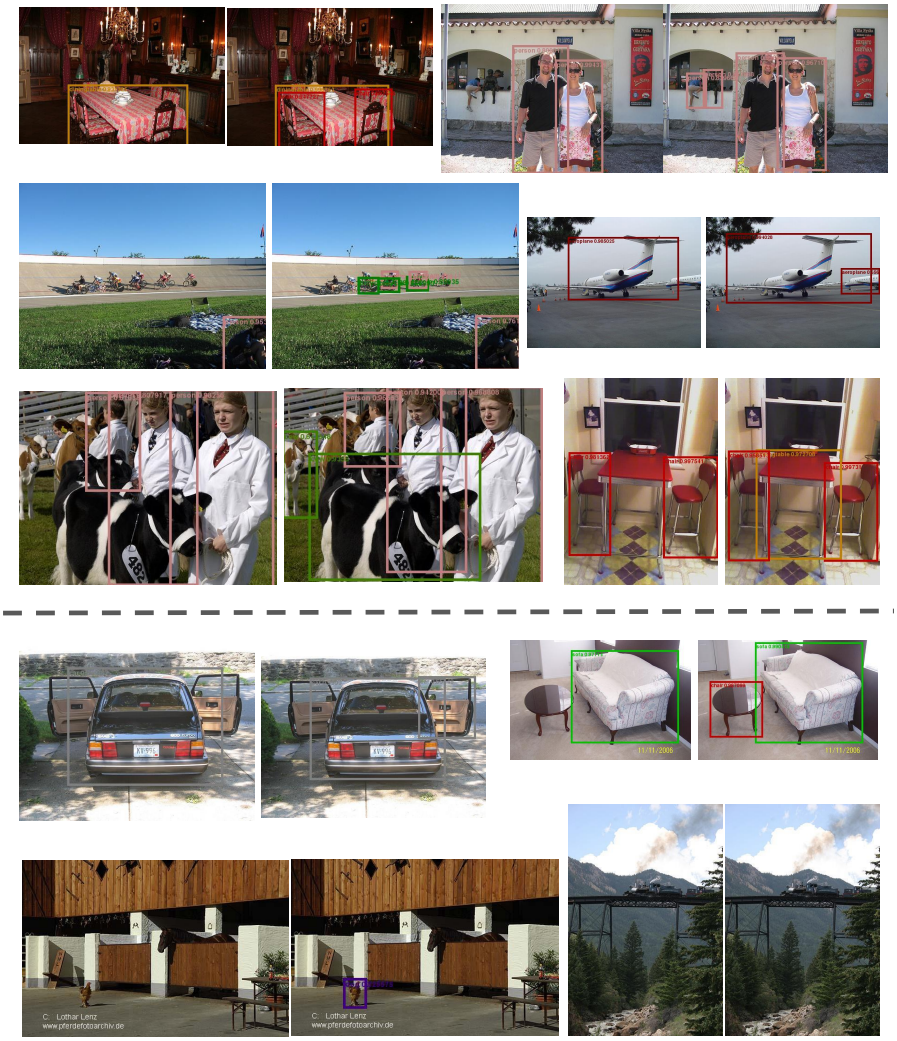

Figure C.7 – **Improved and failure cases of detection by BlitzNet300 comparing to SSD300.** Each pair of images corresponds to the results of detection by SSD300 (left) and BlitzNet300 (right). The cases of improved detection are presened on the top part of the figure and the cases where both methods still fail are placed below the dashed line. It is 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).

Bibliography

[1] https://github.com/igul222/improved_wgan_training.

[2] https://github.com/bioinf-jku/TTUR.

[3] https://github.com/tkarras/progressive_growing_of_gans.

[4] Source code. http://thoth.inrialpes.fr/research/ganeval.

[5] 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. *tensorflow. org*, 2015.

[6] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *ICLR*, 2019.

[7] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

[8] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *ICML*, 2016.

[9] Bernard Ans, Stéphane Rousset, Robert M French, and Serban Musca. Self-refreshing memory in artificial neural networks: Learning temporal sequences without catastrophic forgetting. *Connection Science*, 16(2):71–99, 2004.

[10] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.

[11] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017.

[12] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

[13] Philip Bachman. An architecture for deep, hierarchical generative models. In *NIPS*, 2016.

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[15] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *ICASSP*, 2016.

[16] Shane Barratt and Rishi Sharma. A note on the Inception score. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[17] Abhijit Bendale and Terrance E Boult. Towards open world recognition. In *CVPR*, 2015.

[18] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *CVPR*, 2016.

[19] Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. In *NIPS Workshop Brains and Bits: Neuroscience Meets Machine Learning*, 2016.

[20] Christopher Bishop. *Pattern recognition and machine learning*. Spinger-Verlag, 2006.

[21] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017.

[22] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[23] Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *KDD*, 2006.

[24] Francesco Calimeri, Aldo Marzullo, Claudio Stamile, and Giorgio Terracina. Biomedical data augmentation using generative adversarial neural networks. In *ICANN*, 2017.

[25] Francisco M. Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, 2018.

[26] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *NIPS*, 2000.

[27] 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 *ICLR*, 2015.

[28] 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*, 40(4):834–848, 2018.

[29] Liqun Chen, Shuyang Dai, Yunchen Pu, Erjin Zhou, Chunyuan Li, Qinliang Su, Changyou Chen, and Lawrence Carin. Symmetric variational autoencoder and connections to adversarial learning. In *AISTATS*, 2018.

[30] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016.

[31] Xi Chen, Diederik Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. In *ICLR*, 2017.

[32] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. NEIL: Extracting visual knowledge from web data. In *ICCV*, 2013.

[33] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016.

[34] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good semi-supervised learning that requires a bad GAN. In *NIPS*, 2017.

[35] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *NIPS*, 2017.

[36] Emily L Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a Laplacian pyramid of adversarial networks. In *NIPS*, 2015.

[37] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. In *ICLR Workshop*, 2015.

[38] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *ICLR*, 2017.

[39] Santosh Divvala, Ali Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014.

[40] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *ICLR*, 2017.

[41] Garoe Dorta, Sara Vicente, Lourdes Agapito, Neill D. F. Campbell, and Ivor Simpson. Structured uncertainty prediction networks. In *CVPR*, 2018.

[42] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *ICLR*, 2017.

[43] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2017.

[44] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2017.

[45] Nikita Dvornik, Konstantin Shmelkov, Julien Mairal, and Cordelia Schmid. Blitznet: A real-time deep network for scene understanding. In *ICCV*, 2017.

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

[47] Basura Fernando, Rahaf Aljundi, Rémi Emonet, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised domain adaptation based on subspace alignment. In *Domain Adaptation in Computer Vision Applications*, pages 81–94. Springer, 2017.

[48] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. PathNet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[49] Sanja Fidler, Roozbeh Mottaghi, Alan Yuille, and Raquel Urtasun. Bottom-up segmentation for top-down detection. In *CVPR*, 2013.

[50] Robert M French. Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference. *Cognitive Science Society Conference*, 1994.

[51] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[52] Robert M French, Bernard Ans, and Stephane Rousset. Pseudopatterns and dual-network memory models: Advantages and shortcomings. In *Connectionist models of learning, development and evolution*, pages 13–22. Springer London, 2001.

[53] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using GAN for improved liver lesion classification. In *ISBI*, 2018.

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

[55] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: masked autoencoder for distribution estimation. In *ICML*, 2015.

[56] Ross Girshick. Fast R-CNN. In *ICCV*, 2015.

[57] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[58] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[59] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[60] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *ICLR*, 2014.

[61] Stephen Gould, Tianshi Gao, and Daphne Koller. Region-based segmentation and object detection. In *NIPS*, 2009.

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

[63] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

[64] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *NIPS*, 2017.

[65] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. PixelVAE: A latent variable model for natural images. In *ICLR*, 2017.

[66] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *CVPR*, 2016.

[67] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[68] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[69] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015.

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[72] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, 2015.

[73] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NIPS*, 2017.

[74] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS*, 2014.

[75] Tin Kam Ho. Random decision forests. In *ICDAR*, 1995.

[76] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[77] Khurram Javed and Faisal Shafait. Revisiting distillation and incremental classifier learning. In *ACCV*, 2018.

[78] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

[79] Ronald Kemker and Christopher Kanan. FearNet: Brain-inspired model for incremental learning. In *ICLR*, 2018.

[80] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *AAAI*, 2018.

[81] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[82] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NIPS*, 2018.

[83] Diederik P. Kingma, Tim Salimans, Rafal Józefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational autoencoders with inverse autoregressive flow. In *NIPS*, 2016.

[84] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *ICLR*, 2014.

[85] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017.

[86] Iasonas Kokkinos. UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *CVPR*, 2017.

[87] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[89] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *AISTATS*, 2011.

[90] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, , Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016.

[91] 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*, 1(4):541–551, 1989.

[92] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[93] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[94] Yi Li, Kaiming He, Jian Sun, et al. R-FCN: Object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[95] Yujia Li, Kevin Swersky, and Richard S. Zemel. Generative moment matching networks. In *ICML*, 2015.

[96] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, 2016.

[97] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[98] 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 *ECCV*, 2014.

[99] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[100] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017.

[101] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[102] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NIPS*, 2017.

[103] Thomas Lucas, Corentin Tallec, Yann Ollivier, and Jakob Verbeek. Mixed batches and symmetric discriminators for GAN training. In *ICML*, 2018.

[104] Thomas Lucas and Jakob Verbeek. Auxiliary guided autoregressive variational autoencoders. In *ECML*, 2018.

[105] Mario Lucic, Karol Kurach, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Are GANs created equal? A large-scale study. In *NIPS*, 2018.

[106] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *NIPS*, 2017.

[107] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. In *ICLR*, 2016.

[108] Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018.

[109] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, 2017.

[110] Denis Mareschal, Paul C. Quinn, and Robert M. French. Asymmetric interference in 3- to 4-month-olds' sequential category learning. *Cognitive Science*, 26(3):377–389, 2002.

[111] James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

[112] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989.

[113] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriella Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *PAMI*, 35(11):2624–2637, 2013.

[114] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[115] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[116] Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. In *ICLR*, 2018.

[117] Manuel Molano-Mazon, Arno Onken, Eugenio Piasini, and Stefano Panzeri. Synthesizing realistic neural population activity patterns using generative adversarial networks. In *ICLR*, 2018.

[118] Hirofumi Morishita and Takao K Hensch. Critical period revisited: impact on vision. *Current opinion in neurobiology*, 18(1):101–107, 2008.

[119] Lukas Mosser, Olivier Dubrule, and Martin J Blunt. Reconstruction of three-dimensional porous media using generative adversarial neural networks. *Physical Review E*, 96(4):043309, 2017.

[120] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[121] Sinéad L Mullally and Eleanor A Maguire. Memory, imagination, and predicting the future: a common brain mechanism? *The Neuroscientist*, 20(3):220–234, 2014.

[122] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[123] Natalia Neverova, Rıza Alp Güler, and Iasonas Kokkinos. Dense pose transfer. In *ECCV*, 2018.

[124] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[125] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[126] Peter Norvig. *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1992.

[127] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *NIPS*, 2016.

[128] Augustus Odena, Jacob Buckman, Catherine Olsson, Tom B Brown, Christopher Olah, Colin Raffel, and Ian Goodfellow. Is generator conditioning causally related to GAN performance? In *ICML*, 2018.

[129] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *ICML*, 2017.

[130] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

[131] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. Accelerating science with generative adversarial networks: An application to 3D particle showers in multilayer calorimeters. *Physical review letters*, 120(4):042003, 2018.

[132] 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 *ICCV*, 2015.

[133] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *arXiv preprint arXiv:1802.07569*, 2018.

[134] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830, 2011.

[135] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[136] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *ECCV*, 2016.

[137] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 31(4):497–508, 2001.

[138] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. GANimation: Anatomically-aware facial animation from a single image. In *ECCV*, 2018.

[139] Paul C Quinn, Peter D Eimas, and Stacey L Rosenkrantz. Evidence for representations of perceptually similar natural categories by 3-month-old and 4-month-old infants. *Perception*, 22(4):463–475, 1993.

[140] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *Intl. Conf. Scale Space and Variational Methods in Computer Vision*, 2011.

[141] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[142] Prajit Ramachandran, Tom Le Paine, Pooya Khorrami, Mohammad Babaeizadeh, Shiyu Chang, Yang Zhang, Mark A Hasegawa-Johnson, Roy H Campbell, and Thomas S Huang. Fast generation for convolutional autoregressive models. In *ICLR Workshop*, 2017.

[143] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

[144] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, 2017.

[145] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[146] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *CVPR*, 2017.

[147] Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Dan Belov, and Nando de Freitas. Parallel multiscale autoregressive density estimation. In *ICML*, 2017.

[148] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[149] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

[150] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[151] Marko Ristin, Matthieu Guillaumin, Juergen Gall, and Luc Van Gool. Incremental learning of NCM forests for large-scale image classification. In *CVPR*, 2014.

[152] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

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

[154] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.

[155] 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. *IJCV*, 115(3):211–252, 2015.

[156] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *ICLR*, 2016.

[157] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[158] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *NIPS*, 2016.

[159] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

[160] Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016.

[161] Jeffrey C. Schlimmer and Douglas H. Fisher. A case study of incremental concept induction. In *AAAI*, 1986.

[162] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NIPS*, 2017.

[163] K. Shmelkov, C. Schmid, and K. Alahari. How good is my GAN? In *ECCV*, 2018.

[164] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ICCV*, 2017.

[165] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017.

[166] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[167] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[168] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[169] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[170] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015.

[171] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised MAP inference for image super-resolution. In *ICLR*, 2017.

[172] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *NIPS*, 2016.

[173] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[174] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[175] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. In *CVPR*, 2015.

[176] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, 2010.

[177] 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*, 2018.

[178] Hoang Thanh-Tung, Truyen Tran, and Svetha Venkatesh. On catastrophic forgetting and mode collapse in generative adversarial networks. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[179] Hoang Thanh-Tung, Truyen Tran, and Svetha Venkatesh. Improving generalization and stability of generative adversarial networks. In *ICLR*, 2019.

[180] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. In *ICLR*, 2016.

[181] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *NIPS*, 1996.

[182] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *ICLR*, 2019.

[183] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR*, 2011.

[184] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A Bayesian data augmentation approach for learning deep models. In *NIPS*, 2017.

[185] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015.

[186] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. It takes (only) two: Adversarial generator-encoder networks. In *AAAI*, 2018.

[187] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with PixelCNN decoders. In *NIPS*, 2016.

[188] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016.

[189] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *JMLR*, 9(Nov):2579–2605, 2008.

[190] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *NIPS*, 2018.

[191] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *CVPR*, 2018.

[192] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay GANs: learning to generate images from new categories without forgetting. In *NIPS*, 2018.

[193] Jian Yao, Sanja Fidler, and Raquel Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012.

[194] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

[195] Kyongsik Yun, Jessi Bustos, and Thomas Lu. Predicting rapid fire growth (flashover) using conditional generative adversarial networks. *arXiv preprint arXiv:1801.09804*, 2018.

[196] Peter Zeidman and Eleanor A Maguire. Anterior hippocampus: the anatomy of perception, imagination and episodic memory. *Nature Reviews Neuroscience*, 17(3):173, 2016.

[197] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

[198] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[199] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[200] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial networks. In *ICLR*, 2017.

[201] Zhun Zhong, Liang Zheng, Zhedong Zheng, Shaozi Li, and Yi Yang. Camera style adaptation for person re-identification. In *CVPR*, 2018.

[202] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[203] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.