

# Transferrable Representations for Visual Recognition

*Jeffrey Donahue*



Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2017-106

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-106.html>

May 14, 2017

Copyright © 2017, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

# **Transferrable Representations for Visual Recognition**

by

Jeffrey Donahue

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair

Professor Jitendra Malik

Professor Alexei Efros

Professor Bruno Olshausen

Spring 2017

# **Transferrable Representations for Visual Recognition**

Copyright © 2017

by

Jeffrey Donahue



## Abstract

Transferrable Representations for Visual Recognition

by

Jeffrey Donahue

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

The rapid progress in visual recognition capabilities over the past several years can be attributed largely to improvements in generic and transferrable *feature representations*, particularly learned representations based on *convolutional networks (convnets)* trained “end-to-end” to predict visual semantics given raw pixel intensity values. In this thesis, we analyze the structure of these convnet representations and their generality and transferability to other tasks and settings.

We begin in Chapter 2 by examining the hierarchical semantic structure that naturally emerges in convnet representations from large-scale supervised training, even when this structure is unobserved in the training set. Empirically, the resulting representations generalize surprisingly well to classification in related yet distinct settings.

Chapters 3 and 4 showcase the flexibility of convnet-based representations for prediction tasks where the inputs or targets have more complex structure. Chapter 3 focuses on representation transfer to the object detection and semantic segmentation tasks in which objects must be *localized* within an image, as well as labeled. Chapter 4 augments convnets with *recurrent* structure to handle recognition problems with sequential inputs (e.g., video activity recognition) or outputs (e.g., image captioning). Across each of these domains, end-to-end *fine-tuning* of the representation for the target task provides a substantial additional performance benefit.

Finally, we address the necessity of label supervision for representation learning. In Chapter 5 we propose an unsupervised learning approach based on generative models, demonstrating that some of the transferrable semantic structure learned by supervised convnets can be learned from images alone.

To my parents, Jim and Linda

# Contents

<b>Contents</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Classification: DeCAF</b>	<b>4</b>
2.1 Background . . . . .	5
2.2 Deep Convolutional Activation Features . . . . .	6
2.2.1 Feature generalization and visualization . . . . .	7
2.3 Evaluation . . . . .	8
2.3.1 Object recognition . . . . .	9
2.3.2 Domain adaptation . . . . .	10
2.3.3 Subcategory recognition . . . . .	12
2.3.4 Scene recognition . . . . .	14
2.4 Discussion . . . . .	15
<b>3 Localization: R-CNN</b>	<b>16</b>
3.1 Object detection . . . . .	19
3.1.1 Module design . . . . .	19
3.1.2 Test-time detection . . . . .	20
3.1.3 Training . . . . .	21
3.1.4 Results on PASCAL VOC 2010-12 . . . . .	22
3.1.5 Results on ILSVRC2013 detection . . . . .	23
3.1.6 Visualization, ablation, and modes of error . . . . .	23
3.1.7 The ILSVRC2013 detection dataset . . . . .	29
3.2 Semantic segmentation . . . . .	33
3.3 Discussion . . . . .	34
3.4 Appendix . . . . .	35
3.4.1 Object proposal transformations . . . . .	35
3.4.2 Positive vs. negative examples and softmax . . . . .	35
3.4.3 Bounding-box regression . . . . .	37
3.4.4 Additional feature visualizations . . . . .	38
3.4.5 Per-category segmentation results . . . . .	39

3.4.6	Analysis of cross-dataset redundancy . . . . .	39
<b>4</b>	<b>Sequences: LRCN</b>	<b>46</b>
4.1	Background . . . . .	48
4.2	Long-term Recurrent Convolutional Networks . . . . .	51
4.3	Activity recognition . . . . .	53
4.3.1	Evaluation . . . . .	54
4.4	Image captioning . . . . .	57
4.4.1	Evaluation . . . . .	59
4.5	Video description . . . . .	66
4.5.1	Evaluation . . . . .	67
4.6	Related work . . . . .	68
4.6.1	Prior work . . . . .	68
4.6.2	Contemporaneous and subsequent work . . . . .	69
4.7	Discussion . . . . .	71
<b>5</b>	<b>Unsupervised Learning: BiGAN</b>	<b>73</b>
5.1	Background . . . . .	75
5.2	Bidirectional Generative Adversarial Networks . . . . .	76
5.2.1	Optimal discriminator, generator, & encoder . . . . .	77
5.2.2	Optimal generator & encoder are inverses . . . . .	78
5.2.3	Relationship to autoencoders . . . . .	79
5.2.4	Learning . . . . .	79
5.2.5	Generalized BiGAN . . . . .	80
5.3	Evaluation . . . . .	81
5.3.1	Baseline methods . . . . .	81
5.3.2	Permutation-invariant MNIST . . . . .	82
5.3.3	ImageNet . . . . .	83
5.4	Discussion . . . . .	85
5.5	Appendix . . . . .	87
5.5.1	Additional proofs . . . . .	87
5.5.2	Learning details . . . . .	92
5.5.3	Model and training details . . . . .	93
<b>6</b>	<b>Conclusion</b>	<b>96</b>
6.1	Frontiers and future directions . . . . .	96

## Acknowledgments

I've been extraordinarily lucky to learn from, work with, and enjoy the company of so many brilliant, kind, and funny people over my six years at Berkeley. None of this would have been possible without them.

Thanks to Trevor Darrell for being a great and enthusiastic advisor, always patient and willing to support, brainstorm with, and guide me on any research path I was excited about, and for organizing countless events from bar nights to Tahoe ski retreats that have made being a part of the group so memorable.

Thanks to the rest of my committee – Jitendra Malik, Alyosha Efros, and Bruno Olshausen – for their feedback and guidance with this dissertation, for their memorable teaching and conversations that have informed and shaped my research, and for their infectious enthusiasm.

Thanks to Kristen Grauman, my undergraduate advisor at UT Austin, from whom I learned how to do research and write a paper, and whose inspiring teaching first got me interested in computer vision.

Thanks to the great (former) postdocs and research scientists I've had the chance to collaborate with, including Kate Saenko, Erik Rodner, Marcus Rohrbach, and Sergio Guadarrama. I worked especially closely with and learned an enormous amount from Ross Girshick and Philipp Krähenbühl.

Thanks to all my friends and collaborators in Berkeley (including some visitors) who have made it such an intellectually rich and thoroughly enjoyable place to be (and a sad place to leave...for now at least): Yangqing Jia, Oriol Vinyals, Jon Barron, Sergey Karayev, Judy Hoffman, Jon Long, Ning Zhang, Allie Janoch, Hyun Oh Song, Georgia Gkioxari, Saurabh Gupta, Pulkit Agrawal, Evan Shelhamer, Lisa Anne Hendricks, Eric Tzeng, Richard Zhang, Subha Venugopalan, Carl Doersch, Deepak Pathak, Samaneh Azadi, Kate Rakelly, Max Argus, Erin Grant, Parsa Mahmoudieh, Zeynep Akata, David Fouhey, and many more.

Thanks to everyone on the Visual Discovery team at Pinterest for the fun and welcoming home away from home across the bay, including Kevin Jing, Jiajing Xu, Dmitry Kislyuk, Andrew Zhai, David Liu, Stephanie Rogers, and Raymond Shiao.

Thanks to my brother Chris and to friends across the country who I've been able to stay in touch with over the years – particularly Marc Legrand, Allen Farris, Vishal Ganesan, and Michael Qian – for their moral support.

Finally, thanks to my parents for all their love and support throughout the years, and for always encouraging me to pursue education. This thesis is dedicated to them.

# Chapter 1

## Introduction

The last decade of advancement in visual recognition has been largely fueled by improvements in feature representations. Previously, improving feature representations meant sitting down and thinking very hard about how to process an image locally to make its higher level semantics more accessible to the machine learning algorithm downstream. More recently, however, *end-to-end learning* techniques in which visual feature representations are learned directly from raw pixel intensity values – primarily using convolutional networks or *convnets* – have facilitated rapid progress and come to supplant the hand-engineered approaches.

Though convnets had been proposed by the early 1980s (Fukushima, 1980), and even put into real-world use for handwritten digit recognition by the 1990s (LeCun et al., 1998), the deep learning revolution in mainstream computer vision began at the end of 2012 with a groundbreaking submission (Krizhevsky et al., 2012) from Alex Krizhevsky and his collaborators at the University of Toronto to the ImageNet classification challenge (Russakovsky et al., 2015). The Toronto team trained a deep convnet, now affectionately known as *AlexNet*, to predict image categories directly from pixel intensity inputs. Compared with the next best method, which used a variety of traditional computer vision techniques based on hand-engineered features, *AlexNet* reduced the classification error rate by around 40%.

To some, the reason for the success of this new approach seemed immediately clear: the use of end-to-end learning with minimal data preprocessing to optimize the representation for performance on the task at hand, rather than a hand-designed representation based on intuition and trial-and-error alone. Yet despite the compelling ImageNet results, much of the community remained skeptical of the generality of the approach.

With over 1.2 million labeled images, the ImageNet challenge dataset was much larger than typical datasets in computer vision, which often consist of just a few labeled examples of each category. Was this complex model with millions of parameters learning anything that could generalize to other settings? In particular, could these deep learning based approaches possibly be applicable in more common settings with

just a few labeled examples, or would each new application always require an additional painstaking and expensive annotation effort to label thousands or millions of images?

Furthermore, classification is just one challenge in computer vision – arguably the simplest one – and *AlexNet* struck many as an inscrutable black box whose structure was quite rigid and inflexible, taking input images of a fixed resolution and producing predictions from a fixed and finite discrete hypothesis space. As such, there was palpable doubt that deep learning approaches could facilitate similar progress in more complex visual recognition tasks like object detection and semantic segmentation, where objects need not only be identified but *localized* as well, or in tasks where the inputs or outputs are *dynamic* or *sequential* rather than drawn from a fixed hypothesis space.

In Chapter 2 we address the first set of questions, demonstrating that these deep hierarchical models infer structure and regularities in the data not explicitly specified by the labels on which they are trained (Figure 1.1), and furthermore that their learned representations are highly *transferrable* to related classification tasks. When activations in the intermediate layers of this network are treated as “features” and fed into simple classifiers, respectable classification accuracy can be achieved with as little as a single sample per class, and with more labeled samples, results quickly approach or exceed the prior state of the art.

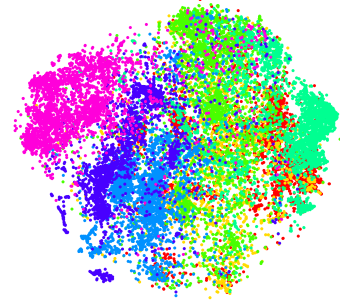


Figure 1.1. Deep convnets learn representations that organize the visual world into “supercategories” they were never explicitly trained to recognize (Chapter 2).

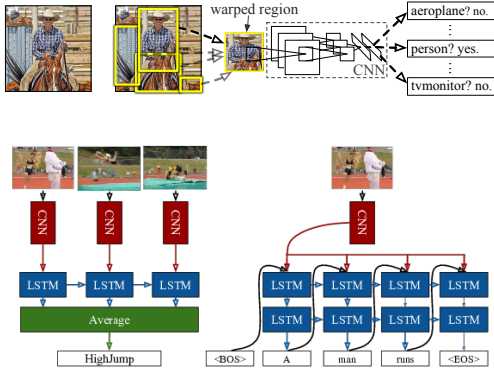


Figure 1.2. When appropriately equipped, convnets go beyond classification, handling tasks with more complex structure (Chapters 3 & 4).

*AlexNet*, the initial skepticism as to whether deep convnets would come to proliferate

With Chapters 3 and 4 we further demonstrate that with a bit of additional instrumentation, these networks are quite capable not only of classification, but of handling more complex localization problems, as well as tasks with dynamic inputs or outputs (Figure 1.2). And while surprisingly good performance comes quickly and easily by transferring an ImageNet-pretrained convnet representation as a simple feature extractor, the real power of these approaches is their capacity to be learned end-to-end, and this power is apparent from the significant further performance gains that can be had by *fine-tuning* the pre-trained weights for the target task.

Given these triumphs and innumerable many others over the last four years since

and dominate all of supervised learning in computer vision has all but evaporated. While network architectures, optimization techniques, and other aspects of supervised deep learning are still being refined and improved to this day, by now there is broad (if at times begrudging) consensus within the community that given a sufficiently large labeled dataset, a convolutional network with appropriate structure can adeptly handle just about any computer vision problem one throws at it.

What is less clear, however, is how these networks might be able to learn *unsupervised* to exploit the virtually infinite supply of unlabeled data available in the wild. Intuition suggests that it should be possible for a model to gain useful knowledge about the structure of the visual world just by looking at it, without always being explicitly told what it's looking at, as in fully supervised learning.

In Chapter 5 we'll discuss approaches that aim to address this important research frontier, focusing particularly on a purely generative model based on Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), a powerful class of models capable of synthesizing realistic images from simple latent distributions. Bidirectional GANs (BiGANs) learn semantic feature representations of the visual world by generating it (Figure 1.3).

Though generative models for unsupervised representation learning are still in their infancy, there are a number of potential areas for improvement, as well as broad applications outside of pure visual recognition in areas like reinforcement learning. And despite the relative strength and maturity of supervised representation learning for visual recognition, a variety of techniques could further enhance their efficiency and accuracy. In Chapter 6 we'll conclude and discuss some of these exciting future research directions.

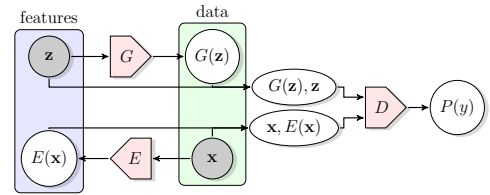


Figure 1.3. Unsupervised BiGANs learn the structure of the visual world by generating it (Chapter 5).



## Chapter 2

# A Deep Convolutional Activation Feature for Generic Visual Recognition<sup>1</sup>

Discovery of effective representations that capture salient semantics for a given task is a key goal of perceptual learning. Performance with conventional visual representations, based on flat feature representations involving quantized gradient filters, has been impressive but has likely plateaued in recent years.

It has long been argued that deep or layered compositional architectures should be able to capture salient aspects of a given domain through discovery of salient clusters, parts, mid-level features, and/or hidden units (Fidler and Leonardis, 2007; Hinton and Salakhutdinov, 2006; Krizhevsky et al., 2012; Singh et al., 2012; Zhu et al., 2007). Such models have been able to perform better than traditional hand-engineered representations in many domains, especially those where good features have not already been engineered (Le et al., 2011). Recent results have shown that moderately deep unsupervised models outperform the state-of-the art gradient histogram features in part-based detection models (Ren and Ramanan, 2013).

Deep models have recently been applied to large-scale visual recognition tasks, trained via back-propagation through layers of convolutional filters (LeCun et al., 1989). These models perform extremely well in domains with large amounts of training data, and had early success in digit classification tasks (LeCun et al., 1998). With the advent of large scale sources of category-level training data; e.g., ImageNet (Deng et al., 2009), and efficient implementation with online approximate model averaging (“dropout”) (Krizhevsky et al., 2012), they have recently outperformed all known methods on a large scale recognition challenge (Deng et al., 2012).

---

<sup>1</sup>This chapter is based on joint work with Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell (Donahue et al., 2014), presented primarily as it appears in the ICML 2014 proceedings. Statements throughout the chapter (such as references to “prior work”) should be read with this context in mind.

With limited training data, however, fully-supervised deep architectures with the representational capacity of (Krizhevsky et al., 2012) will generally dramatically overfit the training data. In fact, many conventional visual recognition challenges have tasks with few training examples; e.g., when a user is defining a category “on-the-fly” using specific examples, or for fine-grained recognition challenges (Welinder et al., 2010), attributes (Bourdev et al., 2011), and/or domain adaptation (Saenko et al., 2010).

In this chapter we investigate semi-supervised multi-task learning of deep convolutional representations, where representations are learned on a set of related problems but applied to new tasks which have too few training examples to learn a full deep representation. Our model can either be considered as a deep architecture for transfer learning based on a supervised pre-training phase, or simply as a new visual feature *DeCAF* defined by the convolutional network weights learned on a set of pre-defined object recognition tasks. Our work is also related to representation learning schemes in computer vision which form an intermediate representation based on learning classifiers on related tasks (Li et al., 2010; Quattoni et al., 2008; Torresani et al., 2010).

Our main result is the empirical validation that a generic visual feature based on a convolutional network weights trained on ImageNet outperforms a host of conventional visual representations on standard benchmark object recognition tasks, including Caltech-101 (Li et al., 2004), the Office domain adaptation dataset (Saenko et al., 2010), the Caltech-UCSD Birds fine-grained recognition dataset (Welinder et al., 2010), and the SUN-397 scene recognition database (Xiao et al., 2010).

Further, we analyze the semantic salience of deep convolutional representations, comparing visual features defined from such networks to conventional representations. In Section 2.2, we visualize the semantic clustering properties of deep convolutional features compared to baseline representations, and find that convolutional features appear to cluster semantic topics more readily than conventional features. Finally, while conventional deep learning can be computationally expensive, we note that the run-time and resource computation of deep-learned convolutional features are not exceptional in comparison to existing features such as HOG (Dalal and Triggs, 2005) or KDES (Bo et al., 2010).

## 2.1 Background

Deep convolutional networks have a long history in computer vision, with early examples showing successful results on using supervised back-propagation networks to perform digit recognition (LeCun et al., 1989). More recently, these networks, in particular the convolutional network proposed by Krizhevsky et al. (2012), have achieved competition-winning numbers on large benchmark datasets consisting of more than one million images, such as ImageNet (Deng et al., 2012).

Learning from related tasks also has a long history in machine learning beginning

with Caruana (1997) and Thrun (1996). Later works such as Argyriou et al. (2006) developed efficient frameworks for optimizing representations from related tasks, and Ando and Zhang (2005) explored how to transfer parameter manifolds to new tasks. In computer vision, forming a representation based on sets of trained classifiers on related tasks has recently been shown to be effective in a variety of retrieval and classification settings, specifically using classifiers based on visual category detectors (Li et al., 2010; Torresani et al., 2010). A key question for such learning problems is to find a feature representation that captures the object category related information while discarding noise irrelevant to object category information such as illumination.

Transfer learning across tasks using deep representations has been extensively studied, especially in an unsupervised setting (Mesnil et al., 2012; Raina et al., 2007). However, reported successes with such models in convolutional networks have been limited to relatively small datasets such as CIFAR and MNIST, and efforts on larger datasets have had only modest success (Le et al., 2012). We investigate the “supervised pre-training” approach proven successful in computer vision and multimedia settings using a concept-bank paradigm (Kennedy and Hauptmann, 2006; Li et al., 2010; Torresani et al., 2010) by learning the features on large-scale data in a supervised setting, then transferring them to different tasks with different labels.

To evaluate the generality of a representation formed from a deep convolutional feature trained on generic recognition tasks, we consider training and testing on datasets known to have a degree of dataset bias with respect to ImageNet. We evaluate on the SUN-397 scene dataset, as well as datasets used to evaluate domain adaptation performance directly (Chopra et al., 2013; Kulis et al., 2011). This evaluates whether the learned features could undo the domain bias by capturing the real semantic information instead of overfitting to domain-specific appearances.

## 2.2 Deep Convolutional Activation Features

In our approach, a deep convolutional model is first trained in a fully supervised setting using a state-of-the-art method (Krizhevsky et al., 2012). We then extract various features from this network, and evaluate the efficacy of these features on generic vision tasks. Even though the forward pass computed by the architecture in this section does achieve state-of-the-art performance on ILSVRC-2012, two questions remain:

- Do features extracted from the CNN generalize to other datasets?
- How do these features perform versus depth?

We address these questions both qualitatively and quantitatively, via visualizations of semantic clusters below, and experimental comparison to current baselines in the following section.

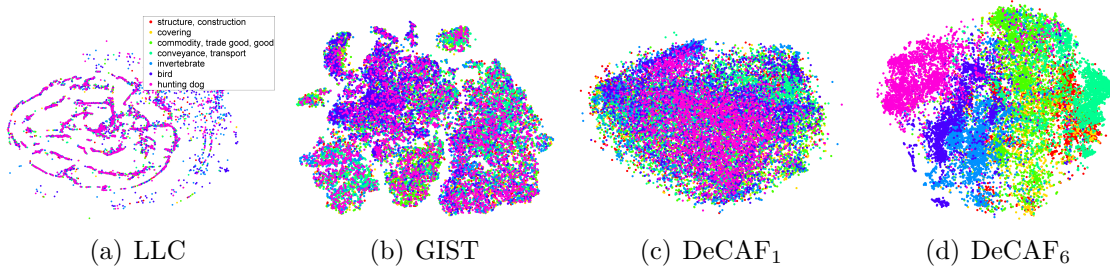


Figure 2.1. This figure shows several t-SNE feature visualizations on the ILSVRC-2012 validation set. (a) LLC, (b) GIST, and features derived from our CNN: (c) DeCAF<sub>1</sub>, the first pooling layer, and (d) DeCAF<sub>6</sub>, the second to last hidden layer (best viewed in color).

### 2.2.1 Feature generalization and visualization

We visualized the model features to gain insight into the semantic capacity of DeCAF and other features that have been typically employed in computer vision. In particular, we compare the features described in Section 2.2 with GIST features (Oliva and Torralba, 2001) and LLC features (Wang et al., 2010).

We visualize features in the following way: we run the t-SNE algorithm (van der Maaten and Hinton, 2008) to find a 2-dimensional embedding of the high-dimensional feature space, and plot them as points colored depending on their semantic category in a particular hierarchy. We did this on the validation set of ILSVRC-2012 to avoid overfitting effects (as the deep CNN used in this chapter was trained only on the training set), and also use an independent dataset, SUN-397 (Xiao et al., 2010), to evaluate how dataset bias affects our results (see, e.g., Torralba and Efros (2011) for a deeper discussion of this topic).

One would expect features closer to the output (softmax) layer to be linearly separable, so it is not very interesting (and also visually quite hard) to represent the 1000 classes on the t-SNE derived embedding.

We first visualize the semantic segregation of the model by plotting the embedding of labels for higher levels of the WordNet hierarchy; for example, a strong feature for visual recognition should cluster indoor and outdoor instances separately, even though there is no explicit modeling through the supervised training of the CNN. Figure 2.1 shows the features extracted on the validation set using the first pooling layer, and the second to last fully connected layer, showing a clear semantic clustering in the latter but not in the former. This is compatible with common deep learning knowledge that the first layers learn “low-level” features, whereas the latter layers learn semantic or “high-level” features. Furthermore, other features such as GIST or LLC fail to capture the semantic difference in the image (although they show interesting clustering

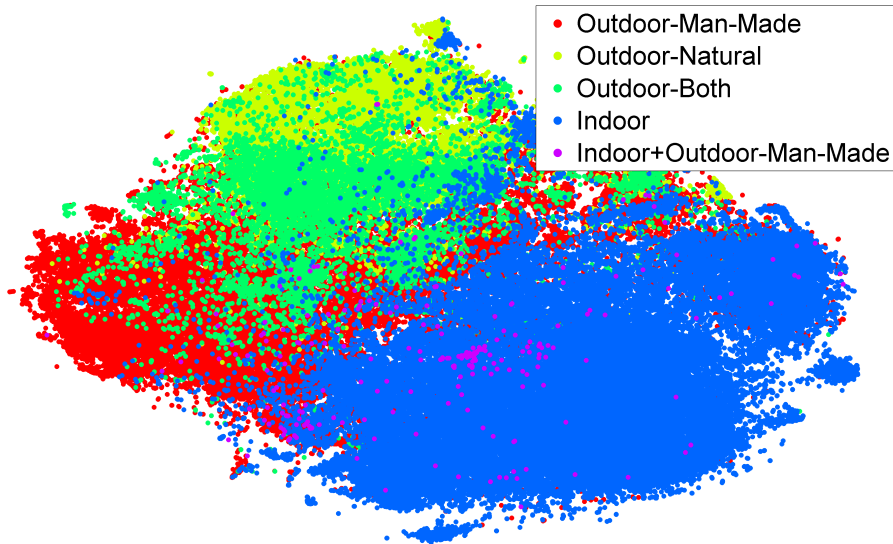


Figure 2.2. In this figure we show how our features trained on ILSVRC-2012 generalized to SUN-397 when considering semantic groupings of labels (best viewed in color).

structure).<sup>2</sup>

More interestingly, in Figure 2.2 we can see the top performing features (DeCAF<sub>6</sub>) on the SUN-397 dataset. Even there, the features show very good clustering of semantic classes (e.g., indoor vs. outdoor). This suggests DeCAF is a good feature for general object recognition tasks. Consider the case where the object class that we are trying to detect is not in the original object pool of ILSVRC-2012. The fact that these features cluster several intermediate nodes of WordNet implies that these features are an excellent starting point for generalizing to unseen classes.

## 2.3 Evaluation

In this section, we present experimental results evaluating DeCAF on multiple standard computer vision benchmarks, comparing many possible featurization and classification approaches. In each of the experiments, we take the activations of the  $n^{\text{th}}$  hidden layer of the deep convolutional neural network described in Section 2.2 as a feature DeCAF <sub>$n$</sub> . DeCAF<sub>7</sub> denotes features taken from the final hidden layer – i.e., just before propagating through the final fully connected layer to produce the class predictions. DeCAF<sub>6</sub> is the activations of the layer before DeCAF<sub>7</sub>, and DeCAF<sub>5</sub> the layer before DeCAF<sub>6</sub>. DeCAF<sub>5</sub> is the first set of activations that has been fully

<sup>2</sup>Some of the features were very high dimensional (e.g. LLC had 16K dimension), in which case we preprocess them by randomly projecting them down to 512 dimensions – random projections are cheap to apply and tend to preserve distances well, which is all the t-SNE algorithm cares about.

	DeCAF <sub>5</sub>	DeCAF <sub>6</sub>	DeCAF <sub>7</sub>
LogReg	$63.29 \pm 6.6$	$84.30 \pm 1.6$	$84.87 \pm 0.6$
LogReg with Dropout	-	$86.08 \pm 0.8$	$85.68 \pm 0.6$
SVM	$77.12 \pm 1.1$	$84.77 \pm 1.2$	$83.24 \pm 1.2$
SVM with Dropout	-	<b><math>86.91 \pm 0.7</math></b>	$85.51 \pm 0.9$
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	

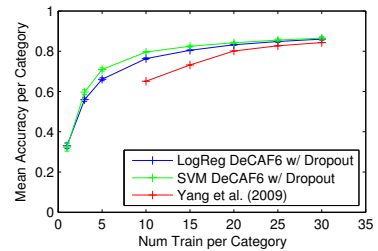


Figure 2.3. Left: average accuracy per class on Caltech-101 with 30 training samples per class across three hidden layers of the network and two classifiers. Our result from the training protocol/classifier combination with the best validation accuracy – SVM with Layer 6 (+ dropout) features – is shown in **bold**. Right: average accuracy per class on Caltech-101 at varying training set sizes.

propagated through the convolutional layers of the network. We chose not to evaluate features from any earlier in the network, as the earlier convolutional layers are unlikely to contain a richer semantic representation than the later features which form higher-level hypotheses from the low to mid-level local information in the activations of the convolutional layers. Because we are investigating the use of the network’s hidden layer activations as features, all of its weights are frozen to those learned on the Deng et al. (2012) dataset.<sup>3</sup> All images are preprocessed using the procedure described for the ILSVRC images in Section 2.2, taking features on the center  $224 \times 224$  crop of the  $256 \times 256$  resized image.

We present results on multiple datasets to evaluate the strength of DeCAF for basic object recognition, domain adaptation, fine-grained recognition, and scene recognition. These tasks each differ somewhat from that for which the architecture was trained, together representing much of the contemporary visual recognition spectrum.

### 2.3.1 Object recognition

To analyze the ability of the deep features to transfer to basic-level object category recognition, we evaluate them on the Caltech-101 dataset (Li et al., 2004). In addition to directly evaluating linear classifier performance on DeCAF<sub>6</sub> and DeCAF<sub>7</sub>, we also report results using a regularization technique called “dropout” proposed by Hinton et al. (2012). At training time, this technique works by randomly setting half of the activations (here, our features) in a given layer to 0. At test time, all activations are multiplied by 0.5. Dropout was used successfully by Krizhevsky et al. (2012) in layers 6 and 7 of their network; hence we study the effect of the technique when applied to the features derived from these layers.

<sup>3</sup>We also experimented with the equivalent feature using randomized weights and found it to have performance comparable to traditional hand-designed features.

In each evaluation, the classifier, a logistic regression (LogReg) or support vector machine (SVM), is trained on a random set of 30 samples per class (including the background class), and tested on the rest of the data, with parameters cross-validated for each split on a 25 train/5 validation subsplit of the training data. The results in Figure 2.3, left, are reported in terms of mean accuracy per category averaged over five data splits.

Our top-performing method (based on validation accuracy) trains a linear SVM on DeCAF<sub>6</sub> with dropout, with test set accuracy of 86.9%. The DeCAF<sub>5</sub> features perform substantially worse than either the DeCAF<sub>6</sub> or DeCAF<sub>7</sub> features, and hence we do not evaluate them further in this chapter. The DeCAF<sub>7</sub> features generally have accuracy about 1-2% lower than the DeCAF<sub>6</sub> features on this task. The dropout regularization technique uniformly improved results by 0-2% for each classifier/feature combination. When trained on DeCAF, the SVM and logistic regression classifiers perform roughly equally well on this task.

We compare our performance against the current state-of-the-art on this benchmark from Yang et al. (2009), a method employing a combination of 5 traditional hand-engineered image features followed by a multi-kernel based classifier. Our top-performing method training a linear SVM on a single feature outperforms this method by 2.6%. Our method also outperforms by over 20% the two-layer convolutional network of Jarrett et al. (2009), demonstrating the importance of the depth of the network used for our feature. Note that unlike our method, these approaches from the literature do not implicitly leverage an outside large-scale image database like ImageNet. The performance edge of our method over these approaches demonstrates the importance of multi-task learning when performing object recognition with sparse data like that available in the Caltech-101 benchmark.

We also show how performance of the two DeCAF<sub>6</sub> with dropout methods above vary with the number of training cases per category, plotted in Figure 2.3, right, trained with fixed parameters and evaluated under the same metric as before. Our one-shot learning results (e.g., 33.0% for SVM) suggest that with sufficiently strong representations like DeCAF, useful models of visual categories can often be learned from just a single positive example.

### 2.3.2 Domain adaptation

We next evaluate DeCAF for use on the task of domain adaptation. For our experiments we use the benchmark *Office* dataset (Saenko et al., 2010). The dataset contains three domains: **Amazon**, which consists of product images taken from **amazon.com**; and **Webcam** and **Dslr**, which consists of images taken in an office environment using a webcam or digital SLR camera, respectively.

In the domain adaptation setting, we are given a training (source) domain with labeled training data and a distinct test (target) domain with either a small amount of labeled data or no labeled data. We will experiment within the supervised domain

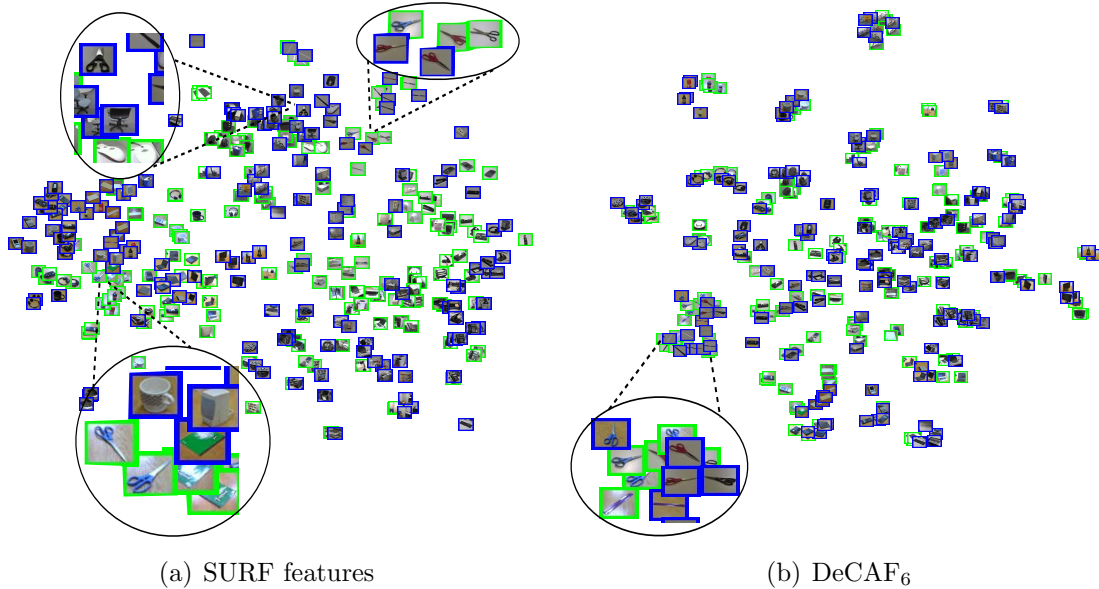


Figure 2.4. Visualization of the webcam (green) and dslr (blue) domains using the original released SURF features (a) and DeCAF<sub>6</sub> (b). The figure is best viewed by zooming in to see the images in local regions. All images from the scissor class are shown enlarged. They are well clustered and overlapping in both domains with our representation, while SURF only clusters a subset and places the others in disjoint parts of the space, closest to distinctly different categories such as chairs and mugs.

adaptation setting, where there is a small amount of labeled data available from the target domain.

Most prior work for this dataset uses SURF (Bay et al., 2006) interest point features (available for download with the dataset). To illustrate the ability of DeCAF to be robust to resolution changes, we use the t-SNE (van der Maaten and Hinton, 2008) algorithm to project both SURF and DeCAF<sub>6</sub>, computed for *Webcam* and *Dslr*, into a 2D visualizable space (See Figure 2.4). We visualize an image on the point in space corresponding to its low dimension projected feature vector. We find that DeCAF not only provides better within category clustering, but also clusters same category instances across domains. This indicates qualitatively that DeCAF removed some of the domain bias between the *Webcam* and *Dslr* domains.

We validate this conclusion with a quantitative experiment on the *Office* dataset. Table 2.1 presents multi-class accuracy averaged across 5 train/test splits for the domain shifts *Amazon*  $\rightarrow$  *Webcam* and *Dslr*  $\rightarrow$  *Webcam*. We use the standard experimental setup first presented in Saenko et al. (2010). To compare SURF with the DeCAF<sub>6</sub>, and DeCAF<sub>7</sub> deep convolutional features, we report the multi-class accuracy for each, using an SVM and Logistic Regression both trained in 3 ways: with only source data (S), only target data (T), and source and target data (ST). We also report results



	Amazon $\rightarrow$ Webcam			Dslr $\rightarrow$ Webcam		
	SURF	DeCAF <sub>6</sub>	DeCAF <sub>7</sub>	SURF	DeCAF <sub>6</sub>	DeCAF <sub>7</sub>
Logistic Reg. (S)	$9.63 \pm 1.4$	$48.58 \pm 1.3$	$53.56 \pm 1.5$	$24.22 \pm 1.8$	$88.77 \pm 1.2$	$87.38 \pm 2.2$
SVM (S)	$11.05 \pm 2.3$	$52.22 \pm 1.7$	$53.90 \pm 2.2$	$38.80 \pm 0.7$	$91.48 \pm 1.5$	$89.15 \pm 1.7$
Logistic Reg. (T)	$24.33 \pm 2.1$	$72.56 \pm 2.1$	$74.19 \pm 2.8$	$24.33 \pm 2.1$	$72.56 \pm 2.1$	$74.19 \pm 2.8$
SVM (T)	$51.05 \pm 2.0$	$78.26 \pm 2.6$	$78.72 \pm 2.3$	$51.05 \pm 2.0$	$78.26 \pm 2.6$	$78.72 \pm 2.3$
Logistic Reg. (ST)	$19.89 \pm 1.7$	$75.30 \pm 2.0$	$76.32 \pm 2.0$	$36.55 \pm 2.2$	$92.88 \pm 0.6$	$91.91 \pm 2.0$
SVM (ST)	$23.19 \pm 3.5$	$80.66 \pm 2.3$	$79.12 \pm 2.1$	$46.32 \pm 1.1$	<b><math>94.79 \pm 1.2</math></b>	$92.96 \pm 2.0$
Daume III (2007)	$40.26 \pm 1.1$	<b><math>82.14 \pm 1.9</math></b>	$81.65 \pm 2.4$	$55.07 \pm 3.0$	$91.25 \pm 1.1$	$89.52 \pm 2.2$
Hoffman et al. (2013)	$37.66 \pm 2.2$	$80.06 \pm 2.7$	$80.37 \pm 2.0$	$53.65 \pm 3.3$	$93.25 \pm 1.5$	$91.45 \pm 1.5$
Gong et al. (2012)	$39.80 \pm 2.3$	$75.21 \pm 1.2$	$77.55 \pm 1.9$	$39.12 \pm 1.3$	$88.40 \pm 1.0$	$88.66 \pm 1.9$
Chopra et al. (2013)		58.85			78.21	

Table 2.1. DeCAF dramatically outperforms the baseline SURF feature available with the *Office* dataset as well as the deep adaptive method of Chopra et al. (2013). We report average multi class accuracy using both non-adaptive and adaptive classifiers, changing only the input feature from SURF to DeCAF. Most surprisingly, in the case of *Dslr* $\rightarrow$ *Webcam* the domain shift is largely non-existent with DeCAF.

for three adaptive methods run with each DeCAF we consider as input. Finally, for completeness we report a recent and competing deep domain adaptation result from Chopra et al. (2013). DeCAF dramatically outperforms the baseline SURF feature available with the *Office* dataset as well as the deep adaptive method of Chopra et al. (2013).

### 2.3.3 Subcategory recognition

We tested the performance of DeCAF on the task of subcategory recognition. To this end, we adopted one of its most popular tasks - the Caltech-UCSD birds dataset (Welinder et al., 2010), and compare the performance against several state-of-the-art baselines.

Following common practice in the literature, we adopted two approaches to perform classification. Our first approach adopts an ImageNet-like pipeline, in which we followed the existing protocol by cropping the images regions  $1.5\times$  the size of the provided bounding boxes, resizing them  $256\times 256$  and then feeding them into the CNN pipeline to get the features for classification. We computed DeCAF<sub>6</sub> and trained a multi-class logistic regression on top of the features.

Our second approach, we tested DeCAF in a pose-normalized setting using the

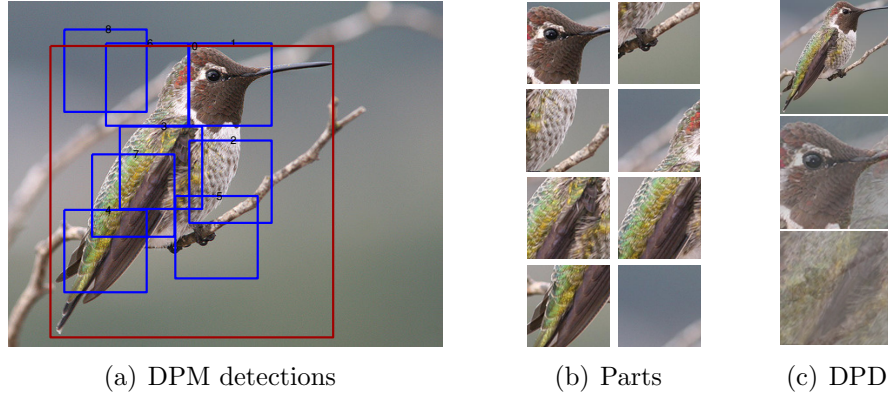


Figure 2.5. Pipeline of deformable part descriptor (DPD) on a sample test images. It uses DPM for part localization and then use learned pooling weights for final pose-normalized representation.

deformable part descriptors (DPD) method (Zhang et al., 2013). Inspired by the deformable parts model (Felzenszwalb et al., 2010), DPD explicitly utilizes the part localization to do semantic pooling. Specifically, after training a weakly-supervised DPM on bird images, the pool weight for each part of each component is calculated by using the key-point annotations to get cross-component semantic part correspondence. The final pose-normalized representation is computed by pooling the image features of predicted part boxes using the pooling weights. Based on the DPD implementation provided by the authors, we applied DeCAF in the same pre-trained DPM model and part predictions and used the same pooling weights. Figure 2.5 shows the DPM detections and visualization of pooled DPD features on a sample test image. As our first approach, we resized each predicted part box to  $256 \times 256$  and computed DeCAF<sub>6</sub> to replace the KDES image features (Bo et al., 2010) used by Zhang et al. (2013).

Our performance as well as those from the literature are listed in Table 2.2. DeCAF together with a simple logistic regression already obtains a significant performance increase over existing approaches, indicating that such features, although not specifically designed to model subcategory-level differences, captures such information well. In addition, explicitly taking more structured information such as part locations still helps, and provides another significant performance increase, obtaining an accuracy of 64.96%, compared to the 50.98% accuracy reported in Zhang et al. (2013). It also outperforms POOF (Berg and Belhumeur, 2013), which is the best part-based approach for fine-grained categorization published so far.

To the best of our knowledge, this is the best accuracy reported so far in the literature.

We note again that in all the experiments above, no fine-tuning is carried out on the CNN layers since our main interest is to analyze how DeCAF generalizes to

Method	Accuracy
DeCAF <sub>6</sub>	58.75
DPD + DeCAF <sub>6</sub>	<b>64.96</b>
DPD (Zhang et al., 2013)	50.98
POOF (Berg and Belhumeur, 2013)	56.78

Table 2.2. Accuracy on the Caltech-UCSD bird dataset.

different tasks. To obtain the best possible result one may want to perform a full back-propagation. However, the fact that we see a significant performance increase without fine-tuning suggests that DeCAF may serve as a good off-the-shelf visual representation without heavy computation.

### 2.3.4 Scene recognition

Finally, we evaluate DeCAF on the SUN-397 large-scale scene recognition database (Xiao et al., 2010). Unlike object recognition, wherein the goal is to identify and classify an object which is usually the primary focus of the image, the goal of a scene recognition task is to classify the *scene* of the entire image. In the SUN-397 database, there are 397 semantic scene categories including *abbey*, *diner*, *mosque*, and *stadium*. Because DeCAF is learned on ILSVRC, an object recognition database, we are applying it to a task for which it was not designed. Hence we might expect this task to be very challenging for these features, unless they are highly generic representations of the visual world.

Based on the success of using dropout with DeCAF<sub>6</sub> and DeCAF<sub>7</sub> for the object recognition task detailed in Section 2.3.1, we train and evaluate linear classifiers on these dropped-out features on the SUN-397 database. Table 2.3 gives the classification accuracy results averaged across 5 splits of 50 training images and 50 test images. Parameters are fixed for all methods, but we select the top-performing method by cross-validation, training on 42 images and testing on the remaining 8 in each split.

Our top-performing method in terms of cross-validation accuracy was to use DeCAF<sub>7</sub> with the SVM classifier, resulting in 40.94% test performance. Comparing against the method of Xiao et al. (2010), the current state-of-the-art method, we see a performance improvement of 2.9% using only DeCAF. Note that, like the state-of-the-art method used as a baseline in Section 2.3.1, this method uses a large set of traditional vision features and combines them with a multi-kernel learning method. The fact that a simple linear classifier on top of our single image feature outperforms the multi-kernel learning baseline built on top of many traditional features demonstrates the ability of DeCAF to generalize to other tasks and its representational power as compared to traditional hand-engineered features.

	DeCAF <sub>6</sub>	DeCAF <sub>7</sub>
LogReg	<b>40.94 ± 0.3</b>	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	

Table 2.3. Average accuracy per class on SUN-397 with 50 training samples and 50 test samples per class, across two hidden layers of the network and two classifiers. Our result from the training protocol/classifier combination with the best validation accuracy – Logistic Regression with DeCAF<sub>7</sub> – is shown in **bold**.

## 2.4 Discussion

In this work, we analyze the use of deep features applied in a semi-supervised multi-task framework. In particular, we demonstrate that by leveraging an auxiliary large labeled object database to train a deep convolutional architecture, we can learn features that have sufficient representational power and generalization ability to perform semantic visual discrimination tasks using simple linear classifiers, reliably outperforming current state-of-the-art approaches based on sophisticated multi-kernel learning techniques with traditional hand-engineered features. Our visual results demonstrate the generality and semantic knowledge implicit in these features, showing that the features tend to cluster images into interesting semantic categories on which the network was never explicitly trained. Our numerical results consistently and robustly demonstrate that our multi-task feature learning framework can substantially improve the performance of a wide variety of existing methods across a spectrum of visual recognition tasks, including domain adaptation, fine-grained part-based recognition, and large-scale scene recognition. The ability of a visual recognition system to achieve high classification accuracy on tasks with sparse labeled data has proven to be an elusive goal in computer vision research, but our multi-task deep learning framework and fast open-source implementation are significant steps in this direction. While our current experiments focus on contemporary recognition challenges, we expect our feature to be very useful in detection, retrieval, and category discovery settings as well.

## Chapter 3

# Object Detection by Regions with Convolutional Network Features<sup>1</sup>

Features matter. The last decade of progress on various visual recognition tasks has been based considerably on the use of SIFT (Lowe, 2004) and HOG (Dalal and Triggs, 2005). But if we look at performance on the canonical visual recognition task, PASCAL VOC object detection (Everingham et al., 2010), it is generally acknowledged that progress has been slow during 2010-2012, with small gains obtained by building ensemble systems and employing minor variants of successful methods.

SIFT and HOG are blockwise orientation histograms, a representation we could associate roughly with complex cells in V1, the first cortical area in the primate visual pathway. But we also know that recognition occurs several stages downstream, which suggests that there might be hierarchical, multi-stage processes for computing features that are even more informative for visual recognition.

Fukushima’s “neocognitron” (Fukushima, 1980), a biologically-inspired hierarchical and shift-invariant model for pattern recognition, was an early attempt at just such a process. The neocognitron, however, lacked a supervised training algorithm. Building on Rumelhart et al. (1985), LeCun et al. (1989) showed that stochastic gradient descent via backpropagation was effective for training convolutional neural networks (CNNs), a class of models that extend the neocognitron.

CNNs saw heavy use in the 1990s (e.g., LeCun et al. (1998)), but then fell out of fashion with the rise of support vector machines. In 2012, Krizhevsky et al. (2012) rekindled interest in CNNs by showing substantially higher image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Deng et al., 2009, 2012). Their success resulted from training a large CNN on 1.2 million labeled images, together with a few twists on LeCun’s CNN (e.g.,  $\max(x, 0)$  rectifying

---

<sup>1</sup>This chapter is based on joint work with Ross Girshick, Trevor Darrell, and Jitendra Malik (Girshick et al., 2014, 2016), presented primarily as it appears in *TPAMI* 2016, an extended version of the CVPR 2014 paper. Statements throughout the chapter (such as references to “prior work”) should be read with this context in mind.

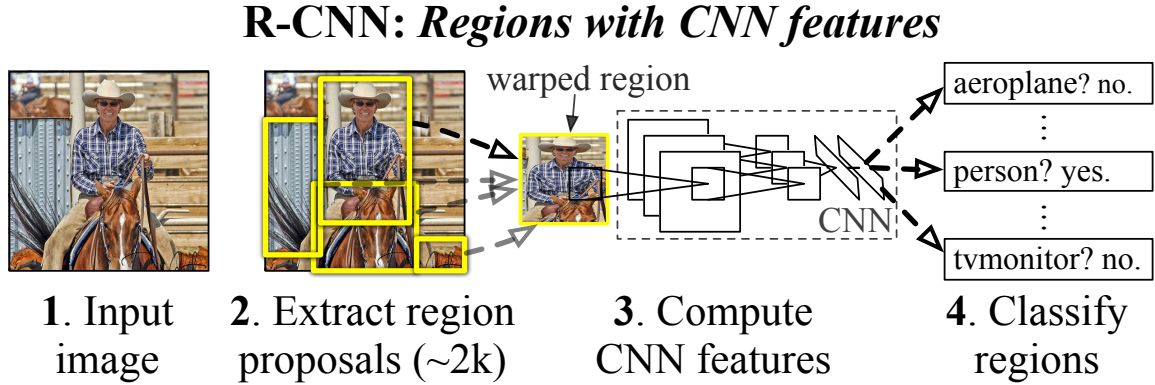


Figure 3.1. **Object detection system overview.** Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, Uijlings et al. (2013) reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%. On the 200-class **ILSVRC2013 detection dataset**, **R-CNN’s mAP is 31.4%**, a large improvement over OverFeat (Sermanet et al., 2013a), which had the previous best result at 24.3%.

non-linearities and “dropout” regularization).

The significance of the ImageNet result was vigorously debated during the ILSVRC 2012 workshop. The central issue can be distilled to the following: To what extent do the CNN classification results on ImageNet generalize to object detection results on the PASCAL VOC Challenge?

We answer this question by bridging the gap between image classification and object detection. This chapter shows that a CNN can lead to dramatically higher object detection performance on PASCAL VOC as compared to systems based on simpler HOG-like features. To achieve this result, we focused on two problems: localizing objects with a deep network and training a high-capacity model with only a small quantity of annotated detection data.

Unlike image classification, detection requires localizing (likely many) objects within an image. One approach frames localization as a regression problem. However, work from Szegedy et al. (2013), concurrent with our own, indicates that this strategy may not fare well in practice (they report a mAP of 30.5% on VOC 2007 compared to the 58.5% achieved by our method). An alternative is to build a sliding-window detector. CNNs have been used in this way for at least two decades, typically on constrained object categories, such as faces (Rowley et al., 1998; Vaillant et al., 1994) and pedestrians (Sermanet et al., 2013b). In order to maintain high spatial

resolution, these CNNs typically only have two convolutional and pooling layers. We also considered adopting a sliding-window approach. However, units high up in our network, which has five convolutional layers, have very large receptive fields ( $195 \times 195$  pixels) and strides ( $32 \times 32$  pixels) in the input image, which makes precise localization within the sliding-window paradigm an open technical challenge.

Instead, we solve the CNN localization problem by operating within the “recognition using regions” paradigm (Gu et al., 2009), which has been successful for both object detection (Uijlings et al., 2013) and semantic segmentation (Carreira and Sminchisescu, 2012). At test time, our method generates around 2000 category-independent region proposals for the input image, extracts a fixed-length feature vector from each proposal using a CNN, and then classifies each region with category-specific linear SVMs. We use a simple technique (affine image warping) to compute a fixed-size CNN input from each region proposal, regardless of the region’s shape. Figure 3.1 presents an overview of our method and highlights some of our results. Since our system combines region proposals with CNNs, we dub the method R-CNN: Regions with CNN features.

In this chapter, we provide a head-to-head comparison of R-CNN and the recently proposed OverFeat (Sermanet et al., 2013a) detection system by running R-CNN on the 200-class ILSVRC2013 detection dataset. OverFeat uses a sliding-window CNN for detection and until now was the best performing method on ILSVRC2013 detection. We show that R-CNN significantly outperforms OverFeat, with a mAP of 31.4% versus 24.3%.

A second challenge faced in detection is that labeled data is scarce and the amount currently available is insufficient for training a large CNN. The conventional solution to this problem is to use *unsupervised* pre-training, followed by supervised fine-tuning (e.g., Sermanet et al. (2013b)). The second principle contribution of this chapter is to show that *supervised* pre-training on a large auxiliary dataset (ILSVRC), followed by domain-specific fine-tuning on a small dataset (PASCAL), is an effective paradigm for learning high-capacity CNNs when data is scarce. In our experiments, fine-tuning for detection improves mAP performance by 8 percentage points. After fine-tuning, our system achieves a mAP of 54% on VOC 2010 compared to 33% for the highly-tuned, HOG-based deformable part model (DPM) (Felzenszwalb et al., 2010; Girshick et al.). We also point readers to contemporaneous work of Donahue et al. (2014), who show that Krizhevsky’s CNN can be used (without fine-tuning) as a blackbox feature extractor, yielding excellent performance on several recognition tasks including scene classification, fine-grained sub-categorization, and domain adaptation.

Our system is also quite efficient. The only class-specific computations are a reasonably small matrix-vector product and greedy non-maximum suppression. This computational property follows from features that are shared across all categories and that are also two orders of magnitude lower-dimensional than previously used region features (*cf.* Uijlings et al. (2013)).

Understanding the failure modes of our approach is also critical for improving it,

and so we report results from the detection analysis tool of Hoiem et al. (2012). As an immediate consequence of this analysis, we demonstrate that a simple bounding-box regression method significantly reduces mislocalizations, which are the dominant error mode.

Before developing technical details, we note that because R-CNN operates on regions it is natural to extend it to the task of semantic segmentation. With minor modifications, we also achieve competitive results on the PASCAL VOC segmentation task, with an average segmentation accuracy of 47.9% on the VOC 2011 test set.

## 3.1 Object detection

Our object detection system consists of three modules. The first generates category-independent region proposals. These proposals define the set of candidate detections available to our detector. The second module is a large convolutional neural network that extracts a fixed-length feature vector from each region. The third module is a set of class-specific linear SVMs. In this section, we present our design decisions for each module, describe their test-time usage, detail how their parameters are learned, and show detection results on PASCAL VOC 2010-12 and on ILSVRC2013.

### 3.1.1 Module design

**Region proposals.** A variety of recent papers offer methods for generating category-independent region proposals. Examples include: objectness (Alexe et al., 2012), selective search (Uijlings et al., 2013), category-independent object proposals (Endres and Hoiem, 2010), constrained parametric min-cuts (CPMC) (Carreira and Sminchisescu, 2012), multi-scale combinatorial grouping (Arbeláez et al., 2014), and Cireşan et al. (Cireşan et al., 2013), who detect mitotic cells by applying a CNN to regularly-spaced square crops, which are a special case of region proposals. While R-CNN is agnostic to the particular region proposal method, we use selective search to enable a controlled comparison with prior detection work (e.g., Uijlings et al. (2013); Wang et al. (2013a)).

**Feature extraction.** We extract a 4096-dimensional feature vector from each region proposal using the Caffe (Jia et al., 2014) implementation of the CNN described by Krizhevsky et al. (Krizhevsky et al., 2012). Features are computed by forward propagating a mean-subtracted  $227 \times 227$  RGB image through five convolutional layers and two fully connected layers. We refer readers to Jia et al. (2014); Krizhevsky et al. (2012) for more network architecture details.

In order to compute features for a region proposal, we must first convert the image data in that region into a form that is compatible with the CNN (its architecture requires inputs of a fixed  $227 \times 227$  pixel size). Of the many possible transformations of our arbitrary-shaped regions, we opt for the simplest. Regardless of the size or





Figure 3.2. **Warped training samples** from VOC 2007 train.

aspect ratio of the candidate region, we warp all pixels in a tight bounding box around it to the required size. Prior to warping, we dilate the tight bounding box so that at the warped size there are exactly  $p$  pixels of warped image context around the original box (we use  $p = 16$ ). Figure 3.2 shows a random sampling of warped training regions. Alternatives to warping are discussed in Appendix 3.4.1.

### 3.1.2 Test-time detection

At test time, we run selective search on the test image to extract around 2000 region proposals (we use selective search’s “fast mode” in all experiments). We warp each proposal and forward propagate it through the CNN in order to compute features. Then, for each class, we score each extracted feature vector using the SVM trained for that class. Given all scored regions in an image, we apply a greedy non-maximum suppression (for each class independently) that rejects a region if it has an intersection-over-union (IoU) overlap with a higher scoring selected region larger than a learned threshold.

**Run-time analysis.** Two properties make detection efficient. First, all CNN parameters are shared across all categories. Second, the feature vectors computed by the CNN are low-dimensional when compared to other common approaches, such as spatial pyramids with bag-of-visual-word encodings. The features used in the UVA detection system (Uijlings et al., 2013), for example, are two orders of magnitude larger than ours (360k vs. 4k-dimensional).

The result of such sharing is that the time spent computing region proposals and features (13s/image on a GPU or 53s/image on a CPU) is amortized over all classes. The only class-specific computations are dot products between features and SVM weights and non-maximum suppression. In practice, all dot products for an image are batched into a single matrix-matrix product. The feature matrix is typically  $2000 \times 4096$  and the SVM weight matrix is  $4096 \times N$ , where  $N$  is the number of classes.

This analysis shows that R-CNN can scale to thousands of object classes without resorting to approximate techniques, such as hashing. Even if there were 100k classes, the resulting matrix multiplication takes only 10 seconds on a modern multi-core

CPU. This efficiency is not merely the result of using region proposals and shared features. The UVA system, due to its high-dimensional features, would be two orders of magnitude slower while requiring 134GB of memory just to store 100k linear predictors, compared to just 1.5GB for our lower-dimensional features.

It is also interesting to contrast R-CNN with the recent work from Dean et al. on scalable detection using DPMs and hashing (Dean et al., 2013). They report a mAP of around 16% on VOC 2007 at a run-time of 5 minutes per image when introducing 10k distractor classes. With our approach, 10k detectors can run in about a minute on a CPU, and because no approximations are made mAP would remain at 59% (Section 3.1.6.2).

### 3.1.3 Training

**Supervised pre-training.** We discriminatively pre-trained the CNN on a large auxiliary dataset (ILSVRC2012 classification) using *image-level annotations* only (bounding-box labels are not available for this data). Pre-training was performed using the open source Caffe CNN library (Jia et al., 2014). In brief, our CNN nearly matches the performance of Krizhevsky et al. (Krizhevsky et al., 2012), obtaining a top-1 error rate 2.2 percentage points higher on the ILSVRC2012 classification validation set. This discrepancy is due to simplifications in the training process.

**Domain-specific fine-tuning.** To adapt our CNN to the new task (detection) and the new domain (warped proposal windows), we continue stochastic gradient descent (SGD) training of the CNN parameters using only warped region proposals. Aside from replacing the CNN’s ImageNet-specific 1000-way classification layer with a randomly initialized  $(N + 1)$ -way classification layer (where  $N$  is the number of object classes, plus 1 for background), the CNN architecture is unchanged. For VOC,  $N = 20$  and for ILSVRC2013,  $N = 200$ . We treat all region proposals with  $\geq 0.5$  IoU overlap with a ground-truth box as positives for that box’s class and the rest as negatives. We start SGD at a learning rate of 0.001 (1/10th of the initial pre-training rate), which allows fine-tuning to make progress while not clobbering the initialization. In each SGD iteration, we uniformly sample 32 positive windows (over all classes) and 96 background windows to construct a mini-batch of size 128. We bias the sampling towards positive windows because they are extremely rare compared to background.

**Object category classifiers.** Consider training a binary classifier to detect cars. It’s clear that an image region tightly enclosing a car should be a positive example. Similarly, it’s clear that a background region, which has nothing to do with cars, should be a negative example. Less clear is how to label a region that partially overlaps a car. We resolve this issue with an IoU overlap threshold, below which regions are defined as negatives. The overlap threshold, 0.3, was selected by a grid search over  $\{0, 0.1, \dots, 0.5\}$  on a validation set. We found that selecting this threshold carefully is important. Setting it to 0.5, as in Uijlings et al. (2013), decreased mAP by 5 points.

VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 (Girshick et al.) <sup>†</sup>	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA (Uijlings et al., 2013)	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets (Wang et al., 2013a)	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM (Fidler et al., 2013) <sup>†</sup>	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	<b>71.8</b>	<b>65.8</b>	<b>53.0</b>	<b>36.8</b>	<b>35.9</b>	<b>59.7</b>	<b>60.0</b>	<b>69.9</b>	<b>27.9</b>	<b>50.6</b>	<b>41.4</b>	<b>70.0</b>	<b>62.0</b>	<b>69.0</b>	<b>58.1</b>	<b>29.5</b>	<b>59.4</b>	<b>39.3</b>	<b>61.2</b>	<b>52.4</b>	<b>53.7</b>

Table 3.1. **Detection average precision (%) on VOC 2010 test.** R-CNN is most directly comparable to UVA and Regionlets since all methods use selective search region proposals. Bounding-box regression (BB) is described in Section 3.1.6.5. At publication time, SegDPM was the top-performer on the PASCAL VOC leaderboard. <sup>†</sup>DPM and SegDPM use context rescoring not used by the other methods.

Similarly, setting it to 0 decreased mAP by 4 points. Positive examples are defined simply to be the ground-truth bounding boxes for each class.

Once features are extracted and training labels are applied, we optimize one linear SVM per class. Since the training data is too large to fit in memory, we adopt the standard hard negative mining method (Felzenszwalb et al., 2010; Sung and Poggio, 1994). Hard negative mining converges quickly and in practice mAP stops increasing after only a single pass over all images.

In Appendix 3.4.2 we discuss why the positive and negative examples are defined differently in fine-tuning versus SVM training. We also discuss the trade-offs involved in training detection SVMs rather than simply using the outputs from the final softmax layer of the fine-tuned CNN.

### 3.1.4 Results on PASCAL VOC 2010-12

Following the PASCAL VOC best practices (Everingham et al., 2010), we validated all design decisions and hyperparameters on the VOC 2007 dataset (Section 3.1.6.2). For final results on the VOC 2010-12 datasets, we fine-tuned the CNN on VOC 2012 train and optimized our detection SVMs on VOC 2012 trainval. We submitted test results to the evaluation server only once for each of the two major algorithm variants (with and without bounding-box regression).

Table 3.1 shows complete results on VOC 2010. We compare our method against four strong baselines, including SegDPM (Fidler et al., 2013), which combines DPM detectors with the output of a semantic segmentation system (Carreira et al., 2012) and uses additional inter-detector context and image-classifier rescoring. The most germane comparison is to the UVA system from Uijlings et al. (2013), since our systems use the same region proposal algorithm. To classify regions, their method builds a four-level spatial pyramid and populates it with densely sampled SIFT, Extended OpponentSIFT, and RGB-SIFT descriptors, each vector quantized with 4000-word codebooks. Classification is performed with a histogram intersection kernel SVM. Compared to their multi-feature, non-linear kernel SVM approach, we achieve a large improvement in mAP, from 35.1% to 53.7% mAP, while also being much faster

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN pool <sub>5</sub>	51.8	60.2	36.4	27.8	23.2	52.8	60.6	49.2	18.3	47.8	44.3	40.8	56.6	58.7	42.4	23.4	46.1	36.7	51.3	55.7	44.2
R-CNN fc <sub>6</sub>	59.3	61.8	43.1	34.0	25.1	53.1	60.6	52.8	21.7	47.8	42.7	47.8	52.5	58.5	44.6	25.6	48.3	34.0	53.1	58.0	46.2
R-CNN fc <sub>7</sub>	57.6	57.9	38.5	31.8	23.7	51.2	58.9	51.4	20.0	50.5	40.9	46.0	51.6	55.9	43.3	23.3	48.1	35.3	51.0	57.4	44.7
R-CNN FT pool <sub>5</sub>	58.2	63.3	37.9	27.6	26.1	54.1	66.9	51.4	26.7	55.5	43.4	43.1	57.7	59.0	45.8	28.1	50.8	40.6	53.1	56.4	47.3
R-CNN FT fc <sub>6</sub>	63.5	66.0	47.9	37.7	29.9	62.5	70.2	60.2	32.0	57.9	47.0	53.5	60.1	64.2	52.2	31.3	55.0	50.0	57.7	63.0	53.1
R-CNN FT fc <sub>7</sub>	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
R-CNN FT fc <sub>7</sub> BB	<b>68.1</b>	<b>72.8</b>	<b>56.8</b>	<b>43.0</b>	<b>36.8</b>	<b>66.3</b>	<b>74.2</b>	<b>67.6</b>	<b>34.4</b>	<b>63.5</b>	<b>54.5</b>	<b>61.2</b>	<b>69.1</b>	<b>68.6</b>	<b>58.7</b>	<b>33.4</b>	<b>62.9</b>	<b>51.1</b>	<b>62.5</b>	<b>64.8</b>	<b>58.5</b>
DPM v5 (Girshick et al.)	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
DPM ST (Lim et al., 2013)	23.8	58.2	10.5	8.5	27.1	50.4	52.0	7.3	19.2	22.8	18.1	8.0	55.9	44.8	32.4	13.3	15.9	22.8	46.2	44.9	29.1
DPM HSC (Ren and Ramanan, 2013)	32.2	58.3	11.5	16.3	30.6	49.9	54.8	23.5	21.5	27.7	34.0	13.7	58.1	51.6	39.9	12.4	23.5	34.4	47.4	45.2	34.3

Table 3.2. **Detection average precision (%) on VOC 2007 test.** Rows 1-3 show R-CNN performance without fine-tuning. Rows 4-6 show results for the CNN pre-trained on ILSVRC 2012 and then fine-tuned (FT) on VOC 2007 trainval. Row 7 includes a simple bounding-box regression (BB) stage that reduces localization errors (Section 3.1.6.5). Rows 8-10 present DPM methods as a strong baseline. The first uses only HOG, while the next two use different feature learning approaches to augment or replace HOG.

(Section 3.1.2). Our method achieves similar performance (53.3% mAP) on VOC 2011/12 test.

### 3.1.5 Results on ILSVRC2013 detection

We ran R-CNN on the 200-class ILSVRC2013 detection dataset using the same system hyperparameters that we used for PASCAL VOC. We followed the same protocol of submitting test results to the ILSVRC2013 evaluation server only twice, once with and once without bounding-box regression.

Figure 3.3 compares R-CNN to the entries in the ILSVRC 2013 competition and to the post-competition OverFeat result (Sermanet et al., 2013a). R-CNN achieves a mAP of 31.4%, which is significantly ahead of the second-best result of 24.3% from OverFeat. To give a sense of the AP distribution over classes, box plots are also presented and a table of per-class APs follows at the end of the chapter in Table 3.8. Most of the competing submissions (OverFeat, NEC-MU, UvA-Euvison, Toronto A, and UIUC-IFP) used convolutional neural networks, indicating that there is significant nuance in how CNNs can be applied to object detection, leading to greatly varying outcomes.

In Section 3.1.7, we give an overview of the ILSVRC2013 detection dataset and provide details about choices that we made when running R-CNN on it.

### 3.1.6 Visualization, ablation, and modes of error

#### 3.1.6.1 Visualizing learned features

First-layer filters can be visualized directly and are easy to understand (Krizhevsky et al., 2012). They capture oriented edges and opponent colors. Understanding the

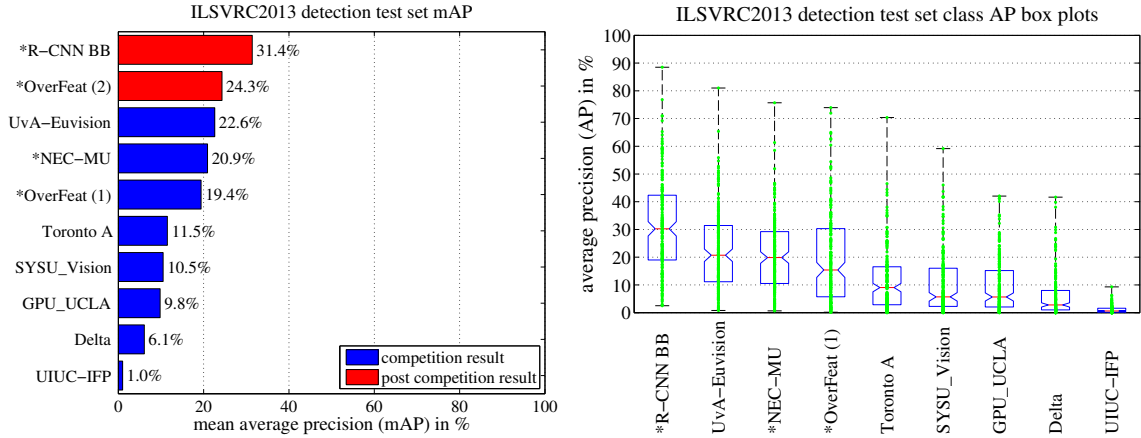


Figure 3.3. **(Left) Mean average precision on the ILSVRC2013 detection test set.** Methods preceded by \* use outside training data (images and labels from the ILSVRC classification dataset in all cases). **(Right) Box plots for the 200 average precision values per method.** A box plot for the post-competition OverFeat result is not shown because per-class APs are not yet available (per-class APs for R-CNN are in Table 3.8 and also included in the tech report source uploaded to arXiv.org; see `R-CNN-ILSVRC2013-APs.txt`). The red line marks the median AP, the box bottom and top are the 25th and 75th percentiles. The whiskers extend to the min and max AP of each method. Each AP is plotted as a green dot over the whiskers (best viewed digitally with zoom).

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN T-Net	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
R-CNN T-Net BB	68.1	72.8	56.8	43.0	36.8	66.3	74.2	67.6	34.4	63.5	54.5	61.2	69.1	68.6	58.7	33.4	62.9	51.1	62.5	64.8	58.5
R-CNN O-Net	71.6	73.5	58.1	42.2	39.4	70.7	76.0	74.5	38.7	71.0	56.9	74.5	67.9	69.6	59.3	<b>35.7</b>	62.1	64.0	66.5	<b>71.2</b>	62.2
R-CNN O-Net BB	<b>73.4</b>	<b>77.0</b>	<b>63.4</b>	<b>45.4</b>	<b>44.6</b>	<b>75.1</b>	<b>78.1</b>	<b>79.8</b>	<b>40.5</b>	<b>73.7</b>	<b>62.2</b>	<b>79.4</b>	<b>78.1</b>	<b>73.1</b>	<b>64.2</b>	35.6	<b>66.8</b>	<b>67.2</b>	<b>70.4</b>	71.1	<b>66.0</b>

Table 3.3. **Detection average precision (%) on VOC 2007 test for two different CNN architectures.** The first two rows are results from Table 3.2 using Krizhevsky et al.’s architecture (T-Net). Rows three and four use the recently proposed 16-layer architecture from Simonyan and Zisserman (O-Net) (Simonyan and Zisserman, 2015).

subsequent layers is more challenging. Zeiler et al. (2011) presents a visually attractive deconvolutional approach. We propose a simple (and complementary) non-parametric method that directly shows what the network learned.

The idea is to single out a particular unit (feature) in the network and use it as if it were an object detector in its own right. That is, we compute the unit’s activations on a large set of held-out region proposals (about 10 million), sort the proposals from highest to lowest activation, perform non-maximum suppression, and then display the

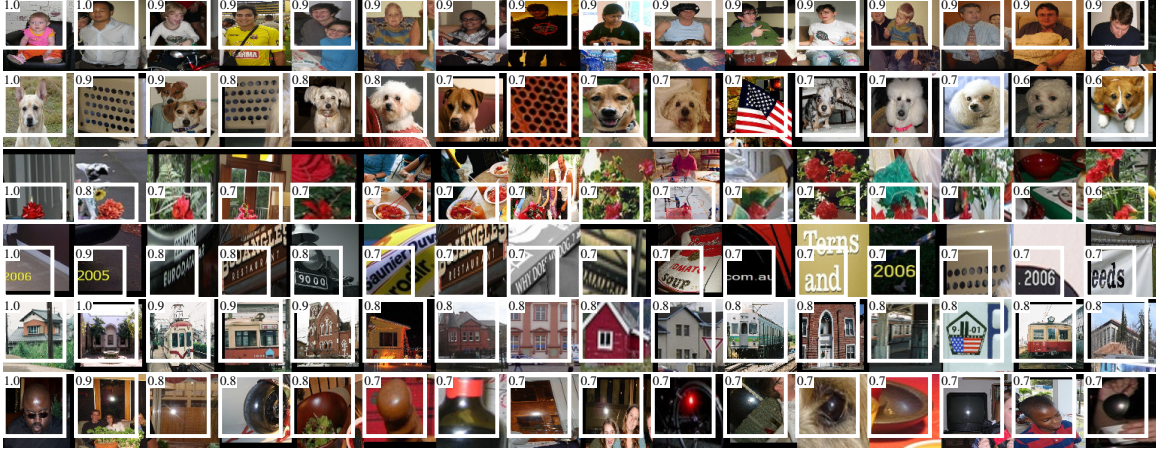


Figure 3.4. **Top regions for six  $\text{pool}_5$  units.** Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

top-scoring regions. Our method lets the selected unit “speak for itself” by showing exactly which inputs it fires on. We avoid averaging in order to see different visual modes and gain insight into the invariances computed by the unit.

We visualize units from layer  $\text{pool}_5$ , which is the max-pooled output of the network’s fifth and final convolutional layer. The  $\text{pool}_5$  feature map is  $6 \times 6 \times 256 = 9216$ -dimensional. Ignoring boundary effects, each  $\text{pool}_5$  unit has a receptive field of  $195 \times 195$  pixels in the original  $227 \times 227$  pixel input. A central  $\text{pool}_5$  unit has a nearly global view, while one near the edge has a smaller, clipped support.

Each row in Figure 3.4 displays the top 16 activations for a  $\text{pool}_5$  unit from a CNN that we fine-tuned on VOC 2007 trainval. Six of the 256 functionally unique units are visualized (Appendix 3.4.4 includes more). These units were selected to show a representative sample of what the network learns. In the second row, we see a unit that fires on dog faces and dot arrays. The unit corresponding to the third row is a red blob detector. There are also detectors for human faces and more abstract patterns such as text and triangular structures with windows. The network appears to learn a representation that combines a small number of class-tuned features together with a distributed representation of shape, texture, color, and material properties. The subsequent fully connected layer  $\text{fc}_6$  has the ability to model a large set of compositions of these rich features.

### 3.1.6.2 Ablation studies

**Performance layer-by-layer, without fine-tuning.** To understand which layers are critical for detection performance, we analyzed results on the VOC 2007 dataset for

each of the CNN’s last three layers. Layer  $\text{pool}_5$  was briefly described in Section 3.1.6.1. The final two layers are summarized below.

Layer  $\text{fc}_6$  is fully connected to  $\text{pool}_5$ . To compute features, it multiplies a  $4096 \times 9216$  weight matrix by the  $\text{pool}_5$  feature map (reshaped as a 9216-dimensional vector) and then adds a vector of biases. This intermediate vector is component-wise half-wave rectified ( $x \leftarrow \max(0, x)$ ).

Layer  $\text{fc}_7$  is the final layer of the network. It is implemented by multiplying the features computed by  $\text{fc}_6$  by a  $4096 \times 4096$  weight matrix, and similarly adding a vector of biases and applying half-wave rectification.

We start by looking at results from the CNN *without fine-tuning* on PASCAL, i.e. all CNN parameters were pre-trained on ILSVRC 2012 only. Analyzing performance layer-by-layer (Table 3.2 rows 1-3) reveals that features from  $\text{fc}_7$  generalize worse than features from  $\text{fc}_6$ . This means that 29%, or about 16.8 million, of the CNN’s parameters can be removed without degrading mAP. More surprising is that removing *both*  $\text{fc}_7$  and  $\text{fc}_6$  produces quite good results even though  $\text{pool}_5$  features are computed using *only 6%* of the CNN’s parameters. Much of the CNN’s representational power comes from its convolutional layers, rather than from the much larger densely connected layers. This finding suggests potential utility in computing a dense feature map, in the sense of HOG, of an arbitrary-sized image by using only the convolutional layers of the CNN. This representation would enable experimentation with sliding-window detectors, including DPM, on top of  $\text{pool}_5$  features.

**Performance layer-by-layer, with fine-tuning.** We now look at results from our CNN after having fine-tuned its parameters on VOC 2007 trainval. The improvement is striking (Table 3.2 rows 4-6): fine-tuning increases mAP by 8.0 percentage points to 54.2%. The boost from fine-tuning is much larger for  $\text{fc}_6$  and  $\text{fc}_7$  than for  $\text{pool}_5$ , which suggests that the  $\text{pool}_5$  features learned from ImageNet are general and that most of the improvement is gained from learning domain-specific non-linear classifiers on top of them.

**Comparison to recent feature learning methods.** Relatively few feature learning methods have been tried on PASCAL VOC detection. We look at two recent approaches that build on deformable part models. For reference, we also include results for the standard HOG-based DPM (Girshick et al.).

The first DPM feature learning method, DPM ST (Lim et al., 2013), augments HOG features with histograms of “sketch token” probabilities. Intuitively, a sketch token is a tight distribution of contours passing through the center of an image patch. Sketch token probabilities are computed at each pixel by a random forest that was trained to classify  $35 \times 35$  pixel patches into one of 150 sketch tokens or background.

The second method, DPM HSC (Ren and Ramanan, 2013), replaces HOG with histograms of sparse codes (HSC). To compute an HSC, sparse code activations are solved for at each pixel using a learned dictionary of 100  $7 \times 7$  pixel (grayscale) atoms.



The resulting activations are rectified in three ways (full and both half-waves), spatially pooled, unit  $\ell_2$  normalized, and then power transformed ( $x \leftarrow \text{sign}(x)|x|^\alpha$ ).

All R-CNN variants strongly outperform the three DPM baselines (Table 3.2 rows 8-10), including the two that use feature learning. Compared to the latest version of DPM, which uses only HOG features, our mAP is more than 20 percentage points higher: 54.2% vs. 33.7%—a 61% *relative improvement*. The combination of HOG and sketch tokens yields 2.5 mAP points over HOG alone, while HSC improves over HOG by 4 mAP points (when compared internally to their private DPM baselines—both use non-public implementations of DPM that underperform the open source version (Girshick et al.)). These methods achieve mAPs of 29.1% and 34.3%, respectively.

### 3.1.6.3 Network architectures

Most results in this chapter use the network architecture from Krizhevsky et al. (2012). However, we have found that the choice of architecture has a large effect on R-CNN detection performance. In Table 3.3 we show results on VOC 2007 test using the 16-layer deep network recently proposed by Simonyan and Zisserman (2015). This network was one of the top performers in the recent ILSVRC 2014 classification challenge. The network has a homogeneous structure consisting of 13 layers of  $3 \times 3$  convolution kernels, with five max pooling layers interspersed, and topped with three fully-connected layers. We refer to this network as “O-Net” for OxfordNet and the baseline as “T-Net” for TorontoNet.

To use O-Net in R-CNN, we downloaded the publicly available pre-trained network weights for the `VGG_ILSVRC_16_layers` model from the Caffe Model Zoo.<sup>2</sup> We then fine-tuned the network using the same protocol as we used for T-Net. The only difference was to use smaller minibatches (24 examples) as required in order to fit within GPU memory. The results in Table 3.3 show that R-CNN with O-Net substantially outperforms R-CNN with T-Net, increasing mAP from 58.5% to 66.0%. However there is a considerable drawback in terms of compute time, with the forward pass of O-Net taking roughly 7 times longer than T-Net.

### 3.1.6.4 Detection error analysis

We applied the excellent detection analysis tool from Hoiem et al. (2012) in order to reveal our method’s error modes, understand how fine-tuning changes them, and to see how our error types compare with DPM. A full summary of the analysis tool is beyond the scope of this chapter and we encourage readers to consult Hoiem et al. (2012) to understand some finer details (such as “normalized AP”). Since the analysis is best absorbed in the context of the associated plots, we present the discussion within the captions of Figure 3.5 and Figure 3.6.

<sup>2</sup><https://github.com/BVLC/caffe/wiki/Model-Zoo>



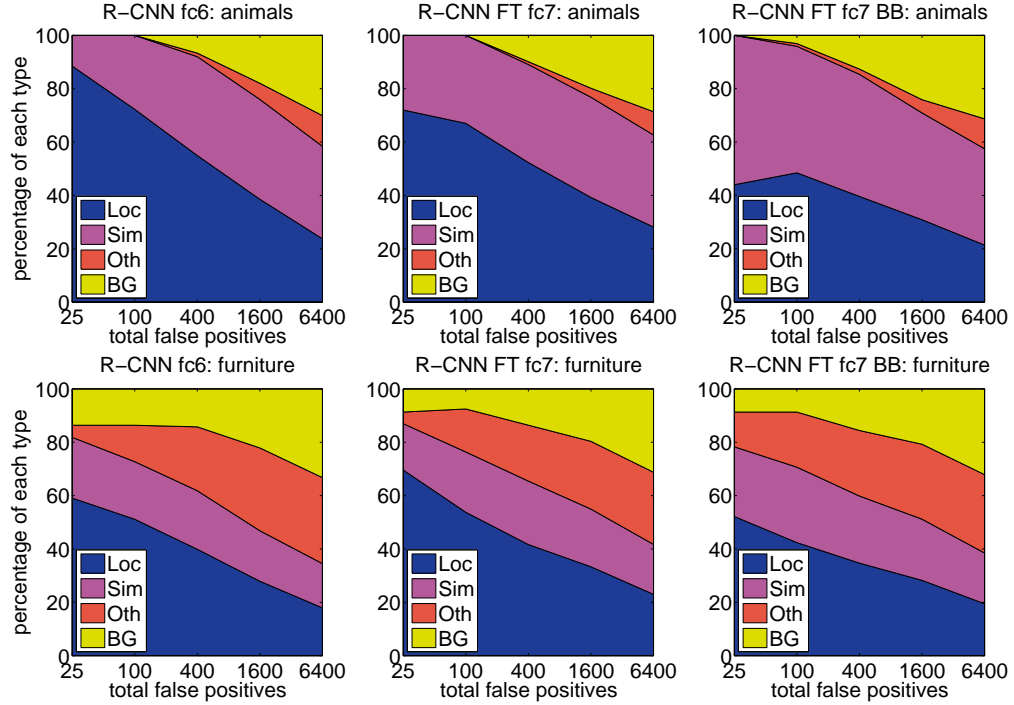


Figure 3.5. **Distribution of top-ranked false positive (FP) types.** Each plot shows the evolving distribution of FP types as more FPs are considered in order of decreasing score. Each FP is categorized into 1 of 4 types: Loc—poor localization (a detection with an IoU overlap with the correct class between 0.1 and 0.5, or a duplicate); Sim—confusion with a similar category; Oth—confusion with a dissimilar object category; BG—a FP that fired on background. Compared with DPM (see Hoiem et al. (2012)), significantly more of our errors result from poor localization, rather than confusion with background or other object classes, indicating that the CNN features are much more discriminative than HOG. Loose localization likely results from our use of bottom-up region proposals and the positional invariance learned from pre-training the CNN for whole-image classification. Column three shows how our simple bounding-box regression method fixes many localization errors.

### 3.1.6.5 Bounding-box regression

Based on the error analysis, we implemented a simple method to reduce localization errors. Inspired by the bounding-box regression employed in DPM (Felzenszwalb et al., 2010), we train a linear regression model to predict a new detection window given the  $\text{pool}_5$  features for a selective search region proposal. Full details are given in Appendix 3.4.3. Results in Table 3.1, Table 3.2, and Figure 3.5 show that this simple approach fixes a large number of mislocalized detections, boosting mAP by 3 to 4 points.

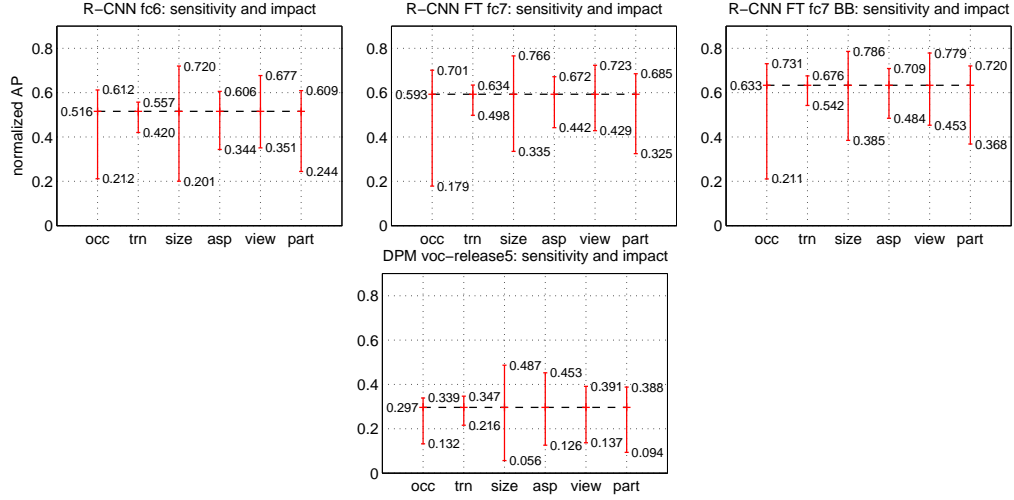


Figure 3.6. **Sensitivity to object characteristics.** Each plot shows the mean (over classes) normalized AP (see Hoiem et al. (2012)) for the highest and lowest performing subsets within six different object characteristics (occlusion, truncation, bounding-box area, aspect ratio, viewpoint, part visibility). We show plots for our method (R-CNN) with and without fine-tuning (FT) and bounding-box regression (BB) as well as for DPM voc-release5. Overall, fine-tuning does not reduce sensitivity (the difference between max and min), but does substantially improve both the highest and lowest performing subsets for nearly all characteristics. This indicates that fine-tuning does more than simply improve the lowest performing subsets for aspect ratio and bounding-box area, as one might conjecture based on how we warp network inputs. Instead, fine-tuning improves robustness for all characteristics including occlusion, truncation, viewpoint, and part visibility.

### 3.1.6.6 Qualitative results

Qualitative detection results on ILSVRC2013 are presented in Figure 3.8 and Figure 3.9 at the end of the chapter. Each image was sampled randomly from the val<sub>2</sub> set and all detections from all detectors with a precision greater than 0.5 are shown. Note that these are not curated and give a realistic impression of the detectors in action. More qualitative results are presented in Figure 3.10 and Figure 3.11, but these have been curated. We selected each image because it contained interesting, surprising, or amusing results. Here, also, all detections at precision greater than 0.5 are shown.

### 3.1.7 The ILSVRC2013 detection dataset

In Section 3.1 we presented results on the ILSVRC2013 detection dataset. This dataset is less homogeneous than PASCAL VOC, requiring choices about how to use

it. Since these decisions are non-trivial, we cover them in this section.

### 3.1.7.1 Dataset overview

The ILSVRC2013 detection dataset is split into three sets: train (395,918), val (20,121), and test (40,152), where the number of images in each set is in parentheses. The val and test splits are drawn from the same image distribution. These images are scene-like and similar in complexity (number of objects, amount of clutter, pose variability, etc.) to PASCAL VOC images. The val and test splits are exhaustively annotated, meaning that in each image all instances from all 200 classes are labeled with bounding boxes. The train set, in contrast, is drawn from the ILSVRC2013 *classification* image distribution. These images have more variable complexity with a skew towards images of a single centered object. Unlike val and test, the train images (due to their large number) are not exhaustively annotated. In any given train image, instances from the 200 classes may or may not be labeled. In addition to these image sets, each class has an extra set of negative images. Negative images are manually checked to validate that they do not contain any instances of their associated class. The negative image sets were not used in this work. More information on how ILSVRC was collected and annotated can be found in Deng et al. (2009); Russakovsky et al. (2015).

The nature of these splits presents a number of choices for training R-CNN. The train images cannot be used for hard negative mining, because annotations are not exhaustive. Where should negative examples come from? Also, the train images have different statistics than val and test. Should the train images be used at all, and if so, to what extent? While we have not thoroughly evaluated a large number of choices, we present what seemed like the most obvious path based on previous experience.

Our general strategy is to rely heavily on the val set and use some of the train images as an auxiliary source of positive examples. To use val for both training and validation, we split it into roughly equally sized “val<sub>1</sub>” and “val<sub>2</sub>” sets. Since some classes have very few examples in val (the smallest has only 31 and half have fewer than 110), it is important to produce an approximately class-balanced partition. To do this, a large number of candidate splits were generated and the one with the smallest maximum relative class imbalance was selected.<sup>3</sup> Each candidate split was generated by clustering val images using their class counts as features, followed by a randomized local search that may improve the split balance. The particular split used here has a maximum relative imbalance of about 11% and a median relative imbalance of 4%. The val<sub>1</sub>/val<sub>2</sub> split and code used to produce them will be publicly available to allow other researchers to compare their methods on the val splits used in this report.

---

<sup>3</sup>Relative imbalance is measured as  $|a - b|/(a + b)$  where  $a$  and  $b$  are class counts in each half of the split.

test set	val <sub>2</sub>	val <sub>2</sub>	val <sub>2</sub>	val <sub>2</sub>	val <sub>2</sub>	val <sub>2</sub>	test	test
SVM training set	val <sub>1</sub>	val <sub>1</sub> +train <sub>5k</sub>	val <sub>1</sub> +train <sub>1k</sub>	val <sub>1</sub> +train <sub>1k</sub>	val <sub>1</sub> +train <sub>1k</sub>	val <sub>1</sub> +train <sub>1k</sub>	val+train <sub>1k</sub>	val+train <sub>1k</sub>
CNN fine-tuning set	n/a	n/a	n/a	val <sub>1</sub>	val <sub>1</sub> +train <sub>1k</sub>	val <sub>1</sub> +train <sub>1k</sub>	val <sub>1</sub> +train <sub>1k</sub>	val <sub>1</sub> +train <sub>1k</sub>
bbox reg set	n/a	n/a	n/a	n/a	n/a	val <sub>1</sub>	n/a	val
CNN feature layer	fc <sub>6</sub>	fc <sub>6</sub>	fc <sub>6</sub>	fc <sub>7</sub>	fc <sub>7</sub>	fc <sub>7</sub>	fc <sub>7</sub>	fc <sub>7</sub>
mAP	20.9	24.1	24.1	26.5	29.7	<b>31.0</b>	30.2	<b>31.4</b>
median AP	17.7	21.0	21.4	24.8	29.2	<b>29.6</b>	29.0	<b>30.3</b>

Table 3.4. **ILSVRC2013 ablation study** of data usage choices, fine-tuning, and bounding-box regression.

### 3.1.7.2 Region proposals

We followed the same region proposal approach that was used for detection on PASCAL. Selective search (Uijlings et al., 2013) was run in “fast mode” on each image in val<sub>1</sub>, val<sub>2</sub>, and test (but not on images in train). One minor modification was required to deal with the fact that selective search is not scale invariant and so the number of regions produced depends on the image resolution. ILSVRC image sizes range from very small to a few that are several mega-pixels, and so we resized each image to a fixed width (500 pixels) before running selective search. On val, selective search resulted in an average of 2403 region proposals per image with a 91.6% recall of all ground-truth bounding boxes (at 0.5 IoU threshold). This recall is notably lower than in PASCAL, where it is approximately 98%, indicating significant room for improvement in the region proposal stage.

### 3.1.7.3 Training data

For training data, we formed a set of images and boxes that includes all selective search and ground-truth boxes from val<sub>1</sub> together with up to  $N$  ground-truth boxes per class from train (if a class has fewer than  $N$  ground-truth boxes in train, then we take all of them). We’ll call this dataset of images and boxes val<sub>1</sub>+train <sub>$N$</sub> . In an ablation study, we show mAP on val<sub>2</sub> for  $N \in \{0, 500, 1000\}$  (Section 3.1.7.5).

Training data is required for three procedures in R-CNN: (1) CNN fine-tuning, (2) detector SVM training, and (3) bounding-box regressor training. CNN fine-tuning was run for 50k SGD iteration on val<sub>1</sub>+train <sub>$N$</sub>  using the exact same settings as were used for PASCAL. Fine-tuning on a single NVIDIA Tesla K20 took 13 hours using Caffe. For SVM training, all ground-truth boxes from val<sub>1</sub>+train <sub>$N$</sub>  were used as positive examples for their respective classes. Hard negative mining was performed on a randomly selected subset of 5000 images from val<sub>1</sub>. An initial experiment indicated that mining negatives from all of val<sub>1</sub>, versus a 5000 image subset (roughly half of it), resulted in only a 0.5 percentage point drop in mAP, while cutting SVM training time in half. No negative examples were taken from train because the annotations are not exhaustive. The extra sets of verified negative images were not used. The bounding-box regressors were trained on val<sub>1</sub>.

#### 3.1.7.4 Validation and evaluation

Before submitting results to the evaluation server, we validated data usage choices and the effect of fine-tuning and bounding-box regression on the  $\text{val}_2$  set using the training data described above. All system hyperparameters (e.g., SVM C hyperparameters, padding used in region warping, NMS thresholds, bounding-box regression hyperparameters) were fixed at the same values used for PASCAL. Undoubtedly some of these hyperparameter choices are slightly suboptimal for ILSVRC, however the goal of this work was to produce a preliminary R-CNN result on ILSVRC without extensive dataset tuning. After selecting the best choices on  $\text{val}_2$ , we submitted exactly two result files to the ILSVRC2013 evaluation server. The first submission was without bounding-box regression and the second submission was with bounding-box regression. For these submissions, we expanded the SVM and bounding-box regressor training sets to use  $\text{val} + \text{train}_{1k}$  and  $\text{val}$ , respectively. We used the CNN that was fine-tuned on  $\text{val}_1 + \text{train}_{1k}$  to avoid re-running fine-tuning and feature computation.

#### 3.1.7.5 Ablation study

Table 3.4 shows an ablation study of the effects of different amounts of training data, fine-tuning, and bounding-box regression. A first observation is that mAP on  $\text{val}_2$  matches mAP on test very closely. This gives us confidence that mAP on  $\text{val}_2$  is a good indicator of test set performance. The first result, 20.9%, is what R-CNN achieves using a CNN pre-trained on the ILSVRC2012 classification dataset (no fine-tuning) and given access to the small amount of training data in  $\text{val}_1$  (recall that half of the classes in  $\text{val}_1$  have between 15 and 55 examples). Expanding the training set to  $\text{val}_1 + \text{train}_N$  improves performance to 24.1%, with essentially no difference between  $N = 500$  and  $N = 1000$ . Fine-tuning the CNN using examples from just  $\text{val}_1$  gives a modest improvement to 26.5%, however there is likely significant overfitting due to the small number of positive training examples. Expanding the fine-tuning set to  $\text{val}_1 + \text{train}_{1k}$ , which adds up to 1000 positive examples per class from the train set, helps significantly, boosting mAP to 29.7%. Bounding-box regression improves results to 31.0%, which is a smaller relative gain than what was observed in PASCAL.

#### 3.1.7.6 Relationship to OverFeat

There is an interesting relationship between R-CNN and OverFeat: OverFeat can be seen (roughly) as a special case of R-CNN. If one were to replace selective search region proposals with a multi-scale pyramid of regular square regions and change the per-class bounding-box regressors to a single bounding-box regressor, then the systems would be very similar (modulo some potentially significant differences in how they are trained: CNN detection fine-tuning, using SVMs, etc.). It is worth noting that OverFeat has a significant speed advantage over R-CNN: it is about 9x faster, based on a figure of 2 seconds per image quoted from Sermanet et al. (2013a). This

speed comes from the fact that OverFeat’s sliding windows (i.e., region proposals) are not warped at the image level and therefore computation can be easily shared between overlapping windows. Sharing is implemented by running the entire network in a convolutional fashion over arbitrary-sized inputs. Speeding up R-CNN should be possible in a variety of ways and remains as future work.

## 3.2 Semantic segmentation

Region classification is a standard technique for semantic segmentation, allowing us to easily apply R-CNN to the PASCAL VOC segmentation challenge. To facilitate a direct comparison with the current leading semantic segmentation system (called O<sub>2</sub>P for “second-order pooling”) (Carreira et al., 2012), we work within their open source framework. O<sub>2</sub>P uses CPMC to generate 150 region proposals per image and then predicts the quality of each region, for each class, using support vector regression (SVR). The high performance of their approach is due to the quality of the CPMC regions and the powerful second-order pooling of multiple feature types (enriched variants of SIFT and LBP). We also note that Farabet et al. (2013) recently demonstrated good results on several dense scene labeling datasets (not including PASCAL) using a CNN as a multi-scale per-pixel classifier.

We follow Arbeláez et al. (2012); Carreira et al. (2012) and extend the PASCAL segmentation training set to include the extra annotations made available by Hariharan et al. (2011). Design decisions and hyperparameters were cross-validated on the VOC 2011 validation set. Final test results were evaluated only once.

**CNN features for segmentation.** We evaluate three strategies for computing features on CPMC regions, all of which begin by warping the rectangular window around the region to  $227 \times 227$ . The first strategy (*full*) ignores the region’s shape and computes CNN features directly on the warped window, exactly as we did for detection. However, these features ignore the non-rectangular shape of the region. Two regions might have very similar bounding boxes while having very little overlap. Therefore, the second strategy (*fg*) computes CNN features only on a region’s foreground mask. We replace the background with the mean input so that background regions are zero after mean subtraction. The third strategy (*full+fg*) simply concatenates the *full* and *fg* features; our experiments validate their complementarity.

	<i>full</i> R-CNN		<i>fg</i> R-CNN		<i>full+fg</i> R-CNN	
O <sub>2</sub> P (Carreira et al., 2012)	fc <sub>6</sub>	fc <sub>7</sub>	fc <sub>6</sub>	fc <sub>7</sub>	fc <sub>6</sub>	fc <sub>7</sub>
46.4	43.0	42.5	43.7	42.1	<b>47.9</b>	45.8

Table 3.5. **Segmentation mean accuracy (%) on VOC 2011 validation.** Column 1 presents O<sub>2</sub>P; 2-7 use our CNN pre-trained on ILSVRC 2012.

VOC 2011 test	bg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
R&P (Arbeláez et al., 2012)	83.4	46.8	18.9	36.6	31.2	42.7	57.3	47.4	44.1	8.1	39.4	<b>36.1</b>	36.3	49.5	48.3	50.7	26.3	47.2	22.1	42.0	43.2	40.8
O <sub>2</sub> P (Carreira et al., 2012)	<b>85.4</b>	<b>69.7</b>	22.3	45.2	<b>44.4</b>	46.9	66.7	57.8	56.2	<b>13.5</b>	<b>46.1</b>	32.3	41.2	<b>59.1</b>	55.3	51.0	<b>36.2</b>	50.4	<b>27.8</b>	46.9	<b>44.6</b>	47.6
ours ( <i>full+fg</i> R-CNN fc <sub>6</sub> )	84.2	66.9	<b>23.7</b>	<b>58.3</b>	37.4	<b>55.4</b>	<b>73.3</b>	<b>58.7</b>	<b>56.5</b>	9.7	45.5	29.5	<b>49.3</b>	40.1	<b>57.8</b>	<b>53.9</b>	33.8	<b>60.7</b>	22.7	<b>47.1</b>	41.3	<b>47.9</b>

Table 3.6. **Segmentation accuracy (%) on VOC 2011 test.** We compare against two strong baselines: the “Regions and Parts” (R&P) (Arbeláez et al., 2012) and the second-order pooling (O<sub>2</sub>P) method (Carreira et al., 2012). Without any fine-tuning, our CNN achieves top segmentation performance, outperforming R&P and roughly matching O<sub>2</sub>P.

**Results on VOC 2011.** Table 3.5 shows a summary of our results on the VOC 2011 validation set compared with O<sub>2</sub>P. (See Appendix 3.4.5 for complete per-category results.) Within each feature computation strategy, layer fc<sub>6</sub> always outperforms fc<sub>7</sub> and the following discussion refers to the fc<sub>6</sub> features. The *fg* strategy slightly outperforms *full*, indicating that the masked region shape provides a stronger signal, matching our intuition. However, *full+fg* achieves an average accuracy of 47.9%, our best result by a margin of 4.2% (also modestly outperforming O<sub>2</sub>P), indicating that the context provided by the *full* features is highly informative even given the *fg* features. Notably, training the 20 SVRs on our *full+fg* features takes an hour on a single core, compared to 10+ hours for training on O<sub>2</sub>P features.

In Table 3.6 we present results on the VOC 2011 test set, comparing our best-performing method, fc<sub>6</sub> (*full+fg*), against two strong baselines. Our method achieves the highest segmentation accuracy for 11 out of 21 categories, and the highest overall segmentation accuracy of 47.9%, averaged across categories (but likely ties with the O<sub>2</sub>P result under any reasonable margin of error). Still better performance could likely be achieved by fine-tuning.

### 3.3 Discussion

In recent years, object detection performance had stagnated. The best performing systems were complex ensembles combining multiple low-level image features with high-level context from object detectors and scene classifiers. This chapter presents a simple and scalable object detection algorithm that gives a 30% relative improvement over the best previous results on PASCAL VOC 2012.

We achieved this performance through two insights. The first is to apply high-capacity convolutional neural networks to bottom-up region proposals in order to localize and segment objects. The second is a paradigm for training large CNNs when labeled training data is scarce. We show that it is highly effective to pre-train the network—*with supervision*—for a auxiliary task with abundant data (image classification) and then to fine-tune the network for the target task where data is scarce (detection). We conjecture that the “supervised pre-training/domain-specific fine-tuning” paradigm will be highly effective for a variety of data-scarce vision

problems.

We conclude by noting that it is significant that we achieved these results by using a combination of classical tools from computer vision *and* deep learning (bottom-up region proposals and convolutional neural networks). Rather than opposing lines of scientific inquiry, the two are natural and inevitable partners.

## 3.4 Appendix

### 3.4.1 Object proposal transformations

The convolutional neural network used in this work requires a fixed-size input of  $227 \times 227$  pixels. For detection, we consider object proposals that are arbitrary image rectangles. We evaluated two approaches for transforming object proposals into valid CNN inputs.

The first method (“tightest square with context”) encloses each object proposal inside the tightest square and then scales (isotropically) the image contained in that square to the CNN input size. Figure 3.7 column (B) shows this transformation. A variant on this method (“tightest square without context”) excludes the image content that surrounds the original object proposal. Figure 3.7 column (C) shows this transformation. The second method (“warp”) anisotropically scales each object proposal to the CNN input size. Figure 3.7 column (D) shows the warp transformation.

For each of these transformations, we also consider including additional image context around the original object proposal. The amount of context padding ( $p$ ) is defined as a border size around the original object proposal in the transformed input coordinate frame. Figure 3.7 shows  $p = 0$  pixels in the top row of each example and  $p = 16$  pixels in the bottom row. In all methods, if the source rectangle extends beyond the image, the missing data is replaced with the image mean (which is then subtracted before inputting the image into the CNN). A pilot set of experiments showed that warping with context padding ( $p = 16$  pixels) outperformed the alternatives by a large margin (3-5 mAP points). Obviously more alternatives are possible, including using replication instead of mean padding. Exhaustive evaluation of these alternatives is left as future work.

### 3.4.2 Positive vs. negative examples and softmax

Two design choices warrant further discussion. The first is: Why are positive and negative examples defined differently for fine-tuning the CNN versus training the object detection SVMs? To review the definitions briefly, for fine-tuning we map each object proposal to the ground-truth instance with which it has maximum IoU overlap (if any) and label it as a positive for the matched ground-truth class if the IoU is at least 0.5. All other proposals are labeled “background” (i.e., negative examples for





Figure 3.7. **Different object proposal transformations.** (A) the original object proposal at its actual scale relative to the transformed CNN inputs; (B) tightest square with context; (C) tightest square without context; (D) warp. Within each column and example proposal, the top row corresponds to  $p = 0$  pixels of context padding while the bottom row has  $p = 16$  pixels of context padding.

all classes). For training SVMs, in contrast, we take only the ground-truth boxes as positive examples for their respective classes and label proposals with less than 0.3 IoU overlap with all instances of a class as a negative for that class. Proposals that fall into the grey zone (more than 0.3 IoU overlap, but are not ground truth) are ignored.

Historically speaking, we arrived at these definitions because we started by training SVMs on features computed by the ImageNet pre-trained CNN, and so fine-tuning was not a consideration at that point in time. In that setup, we found that our particular label definition for training SVMs was optimal within the set of options we evaluated (which included the setting we now use for fine-tuning). When we started using fine-tuning, we initially used the same positive and negative example definition as we were using for SVM training. However, we found that results were much worse than those obtained using our current definition of positives and negatives.

Our hypothesis is that this difference in how positives and negatives are defined is not fundamentally important and arises from the fact that fine-tuning data is limited. Our current scheme introduces many “jittered” examples (those proposals with overlap between 0.5 and 1, but not ground truth), which expands the number of positive examples by approximately 30x. We conjecture that this large set is needed

when fine-tuning the *entire* network to avoid overfitting. However, we also note that using these jittered examples is likely suboptimal because the network is not being fine-tuned for precise localization.

This leads to the second issue: Why, after fine-tuning, train SVMs at all? It would be cleaner to simply apply the last layer of the fine-tuned network, which is a 21-way softmax regression classifier, as the object detector. We tried this and found that performance on VOC 2007 dropped from 54.2% to 50.9% mAP. This performance drop likely arises from a combination of several factors including that the definition of positive examples used in fine-tuning does not emphasize precise localization and the softmax classifier was trained on randomly sampled negative examples rather than on the subset of “hard negatives” used for SVM training.

This result shows that it’s possible to obtain close to the same level of performance without training SVMs after fine-tuning. We conjecture that with some additional tweaks to fine-tuning the remaining performance gap may be closed. If true, this would simplify and speed up R-CNN training with no loss in detection performance.

### 3.4.3 Bounding-box regression

We use a simple bounding-box regression stage to improve localization performance. After scoring each selective search proposal with a class-specific detection SVM, we predict a new bounding box for the detection using a class-specific bounding-box regressor. This is similar in spirit to the bounding-box regression used in deformable part models (Felzenszwalb et al., 2010). The primary difference between the two approaches is that here we regress from features computed by the CNN, rather than from geometric features computed on the inferred DPM part locations.

The input to our training algorithm is a set of  $N$  training pairs  $\{(P^i, G^i)\}_{i=1,\dots,N}$ , where  $P^i = (P_x^i, P_y^i, P_w^i, P_h^i)$  specifies the pixel coordinates of the center of proposal  $P^i$ ’s bounding box together with  $P^i$ ’s width and height in pixels. Hence forth, we drop the superscript  $i$  unless it is needed. Each ground-truth bounding box  $G$  is specified in the same way:  $G = (G_x, G_y, G_w, G_h)$ . Our goal is to learn a transformation that maps a proposed box  $P$  to a ground-truth box  $G$ .

We parameterize the transformation in terms of four functions  $d_x(P)$ ,  $d_y(P)$ ,  $d_w(P)$ , and  $d_h(P)$ . The first two specify a scale-invariant translation of the center of  $P$ ’s bounding box, while the second two specify log-space translations of the width and height of  $P$ ’s bounding box. After learning these functions, we can transform an input proposal  $P$  into a predicted ground-truth box  $\hat{G}$  by applying the transformation

$$\hat{G}_x = P_w d_x(P) + P_x \quad (3.1)$$

$$\hat{G}_y = P_h d_y(P) + P_y \quad (3.2)$$

$$\hat{G}_w = P_w \exp(d_w(P)) \quad (3.3)$$

$$\hat{G}_h = P_h \exp(d_h(P)). \quad (3.4)$$

Each function  $d_\star(P)$  (where  $\star$  is one of  $x, y, h, w$ ) is modeled as a linear function of the  $\text{pool}_5$  features of proposal  $P$ , denoted by  $\phi_5(P)$ . (The dependence of  $\phi_5(P)$  on the image data is implicitly assumed.) Thus we have  $d_\star(P) = \mathbf{w}_\star^\top \phi_5(P)$ , where  $\mathbf{w}_\star$  is a vector of learnable model parameters. We learn  $\mathbf{w}_\star$  by optimizing the regularized least squares objective (ridge regression):

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^\top \phi_5(P^i))^2 + \lambda \|\hat{\mathbf{w}}_\star\|^2. \quad (3.5)$$

The regression targets  $t_\star$  for the training pair  $(P, G)$  are defined as

$$t_x = (G_x - P_x)/P_w \quad (3.6)$$

$$t_y = (G_y - P_y)/P_h \quad (3.7)$$

$$t_w = \log(G_w/P_w) \quad (3.8)$$

$$t_h = \log(G_h/P_h). \quad (3.9)$$

As a standard regularized least squares problem, this can be solved efficiently in closed form.

We found two subtle issues while implementing bounding-box regression. The first is that regularization is important: we set  $\lambda = 1000$  based on a validation set. The second issue is that care must be taken when selecting which training pairs  $(P, G)$  to use. Intuitively, if  $P$  is far from all ground-truth boxes, then the task of transforming  $P$  to a ground-truth box  $G$  does not make sense. Using examples like  $P$  would lead to a hopeless learning problem. Therefore, we only learn from a proposal  $P$  if it is *nearby* at least one ground-truth box. We implement “nearness” by assigning  $P$  to the ground-truth box  $G$  with which it has maximum IoU overlap (in case it overlaps more than one) if and only if the overlap is greater than a threshold (which we set to 0.6 using a validation set). All unassigned proposals are discarded. We do this once for each object class in order to learn a set of class-specific bounding-box regressors.

At test time, we score each proposal and predict its new detection window only once. In principle, we could iterate this procedure (i.e., re-score the newly predicted bounding box, and then predict a new bounding box from it, and so on). However, we found that iterating does not improve results.

### 3.4.4 Additional feature visualizations

Figure 3.12 shows additional visualizations for 20  $\text{pool}_5$  units. For each unit, we show the 24 region proposals that maximally activate that unit out of the full set of approximately 10 million regions in all of VOC 2007 test.

We label each unit by its (y, x, channel) position in the  $6 \times 6 \times 256$  dimensional  $\text{pool}_5$  feature map. Within each channel, the CNN computes exactly the same function of the input region, with the (y, x) position changing only the receptive field.

VOC 2011 val	bg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
O <sub>2</sub> P (Carreira et al., 2012)	<b>84.0</b>	<b>69.0</b>	21.7	47.7	42.2	42.4	<b>64.7</b>	<b>65.8</b>	57.4	<b>12.9</b>	37.4	20.5	43.7	35.7	52.7	51.0	<b>35.8</b>	<b>51.0</b>	28.4	59.8	49.7	46.4
full R-CNN fc <sub>6</sub>	81.3	56.2	23.9	42.9	40.7	38.8	59.2	56.5	53.2	11.4	34.6	16.7	48.1	37.0	51.4	46.0	31.5	44.0	24.3	53.7	51.1	43.0
full R-CNN fc <sub>7</sub>	81.0	52.8	<b>25.1</b>	43.8	40.5	42.7	55.4	57.7	51.3	8.7	32.5	11.5	48.1	37.0	50.5	46.4	30.2	42.1	21.2	57.7	<b>56.0</b>	42.5
fg R-CNN fc <sub>6</sub>	81.4	54.1	21.1	40.6	38.7	<b>53.6</b>	59.9	57.2	52.5	9.1	36.5	<b>23.6</b>	46.4	38.1	53.2	51.3	32.2	38.7	<b>29.0</b>	53.0	47.5	43.7
fg R-CNN fc <sub>7</sub>	80.9	50.1	20.0	40.2	34.1	40.9	59.7	59.8	52.7	7.3	32.1	14.3	48.8	42.9	54.0	48.6	28.9	42.6	24.9	52.2	48.8	42.1
full+fg R-CNN fc <sub>6</sub>	83.1	60.4	23.2	48.4	<b>47.3</b>	52.6	61.6	60.6	<b>59.1</b>	10.8	<b>45.8</b>	20.9	<b>57.7</b>	43.3	<b>57.4</b>	<b>52.9</b>	34.7	48.7	28.1	60.0	48.6	<b>47.9</b>
full+fg R-CNN fc <sub>7</sub>	82.3	56.7	20.6	<b>49.9</b>	44.2	43.6	59.3	61.3	57.8	7.7	38.4	15.1	53.4	<b>43.7</b>	50.8	52.0	34.1	47.8	24.7	<b>60.1</b>	55.2	45.7

Table 3.7. Per-category segmentation accuracy (%) on the VOC 2011 validation set.

### 3.4.5 Per-category segmentation results

In Table 3.7 we show the per-category segmentation accuracy on VOC 2011 val for each of our six segmentation methods in addition to the O<sub>2</sub>P method (Carreira et al., 2012). These results show which methods are strongest across each of the 20 PASCAL classes, plus the background class.

### 3.4.6 Analysis of cross-dataset redundancy

One concern when training on an auxiliary dataset is that there might be redundancy between it and the test set. Even though the tasks of object detection and whole-image classification are substantially different, making such cross-set redundancy much less worrisome, we still conducted a thorough investigation that quantifies the extent to which PASCAL test images are contained within the ILSVRC 2012 training and validation sets. Our findings may be useful to researchers who are interested in using ILSVRC 2012 as training data for the PASCAL image classification task.

We performed two checks for duplicate (and near-duplicate) images. The first test is based on exact matches of flickr image IDs, which are included in the VOC 2007 test annotations (these IDs are intentionally kept secret for subsequent PASCAL test sets). All PASCAL images, and about half of ILSVRC, were collected from flickr.com. This check turned up 31 matches out of 4952 (0.63%).

The second check uses GIST (Oliva and Torralba, 2001) descriptor matching, which was shown in Douze et al. (2009) to have excellent performance at near-duplicate image detection in large (> 1 million) image collections. Following Douze et al. (2009), we computed GIST descriptors on warped  $32 \times 32$  pixel versions of all ILSVRC 2012 trainval and PASCAL 2007 test images.

Euclidean distance nearest-neighbor matching of GIST descriptors revealed 38 near-duplicate images (including all 31 found by flickr ID matching). The matches tend to vary slightly in JPEG compression level and resolution, and to a lesser extent cropping. These findings show that the overlap is small, less than 1%. For VOC 2012, because flickr IDs are not available, we used the GIST matching method only. Based on GIST matches, 1.5% of VOC 2012 test images are in ILSVRC 2012 trainval. The slightly higher rate for VOC 2012 is likely due to the fact that the two datasets were collected closer together in time than VOC 2007 and ILSVRC 2012 were.

class	AP	class	AP	class	AP	class	AP	class	AP
accordion	50.8	centipede	30.4	hair spray	13.8	pencil box	11.4	snowplow	69.2
airplane	50.0	chain saw	14.1	hamburger	34.2	pencil sharpener	9.0	soap dispenser	16.8
ant	31.8	chair	19.5	hammer	9.9	perfume	32.8	soccer ball	43.7
antelope	53.8	chime	24.6	hamster	46.0	person	41.7	sofa	16.3
apple	30.9	cocktail shaker	46.2	harmonica	12.6	piano	20.5	spatula	6.8
armadillo	54.0	coffee maker	21.5	harp	50.4	pineapple	22.6	squirrel	31.3
artichoke	45.0	computer keyboard	39.6	hat with a wide brim	40.5	ping-pong ball	21.0	starfish	45.1
axe	11.8	computer mouse	21.2	head cabbage	17.4	pitcher	19.2	stethoscope	18.3
baby bed	42.0	corkscrew	24.2	helmet	33.4	pizza	43.7	stove	8.1
backpack	2.8	cream	29.9	hippopotamus	38.0	plastic bag	6.4	strainer	9.9
bagel	37.5	croquet ball	30.0	horizontal bar	7.0	plate rack	15.2	strawberry	26.8
balance beam	32.6	crutch	23.7	horse	41.7	pomegranate	32.0	stretcher	13.2
banana	21.9	cucumber	22.8	hotdog	28.7	popsicle	21.2	sunglasses	18.8
band aid	17.4	cup or mug	34.0	iPod	59.2	porcupine	37.2	swimming trunks	9.1
banjo	55.3	diaper	10.1	isopod	19.5	power drill	7.9	swine	45.3
baseball	41.8	digital clock	18.5	jellyfish	23.7	pretzel	24.8	syringe	5.7
basketball	65.3	dishwasher	19.9	koala bear	44.3	printer	21.3	table	21.7
bathing cap	37.2	dog	76.8	ladle	3.0	puck	14.1	tape player	21.4
beaker	11.3	domestic cat	44.1	ladybug	58.4	punching bag	29.4	tennis ball	59.1
bear	62.7	dragonfly	27.8	lamp	9.1	purse	8.0	tick	42.6
bee	52.9	drum	19.9	laptop	35.4	rabbit	71.0	tie	24.6
bell pepper	38.8	dumbbell	14.1	lemon	33.3	racket	16.2	tiger	61.8
bench	12.7	electric fan	35.0	lion	51.3	ray	41.1	toaster	29.2
bicycle	41.1	elephant	56.4	lipstick	23.1	red panda	61.1	traffic light	24.7
binder	6.2	face powder	22.1	lizard	38.9	refrigerator	14.0	train	60.8
bird	70.9	fig	44.5	lobster	32.4	remote control	41.6	trombone	13.8
bookshelf	19.3	filing cabinet	20.6	maillot	31.0	rubber eraser	2.5	trumpet	14.4
bow tie	38.8	flower pot	20.2	maraca	30.1	rugby ball	34.5	turtle	59.1
bow	9.0	flute	4.9	microphone	4.0	ruler	11.5	tv or monitor	41.7
bowl	26.7	fox	59.3	microwave	40.1	salt or pepper shaker	24.6	unicycle	27.2
brassiere	31.2	french horn	24.2	milk can	33.3	saxophone	40.8	vacuum	19.5
burrito	25.7	frog	64.1	miniskirt	14.9	scorpion	57.3	violin	13.7
bus	57.5	frying pan	21.5	monkey	49.6	screwdriver	10.6	volleyball	59.7
butterfly	88.5	giant panda	42.5	motorcycle	42.2	seal	20.9	waffle iron	24.0
camel	37.6	goldfish	28.6	mushroom	31.8	sheep	48.9	washer	39.8
can opener	28.9	golf ball	51.3	nail	4.5	ski	9.0	water bottle	8.1
car	44.5	golfcart	47.9	neck brace	31.6	skunk	57.9	watercraft	40.9
cart	48.0	guacamole	32.3	oboe	27.5	snail	36.2	whale	48.6
cattle	32.3	guitar	33.1	orange	38.8	snake	33.8	wine bottle	31.2
cello	28.9	hair dryer	13.0	otter	22.2	snowmobile	58.8	zebra	49.6

Table 3.8. Per-class average precision (%) on the ILSVRC2013 detection test set.













Figure 3.10. Curated examples. Each image was selected because we found it impressive, surprising, interesting, or amusing. Viewing digitally with zoom is recommended.





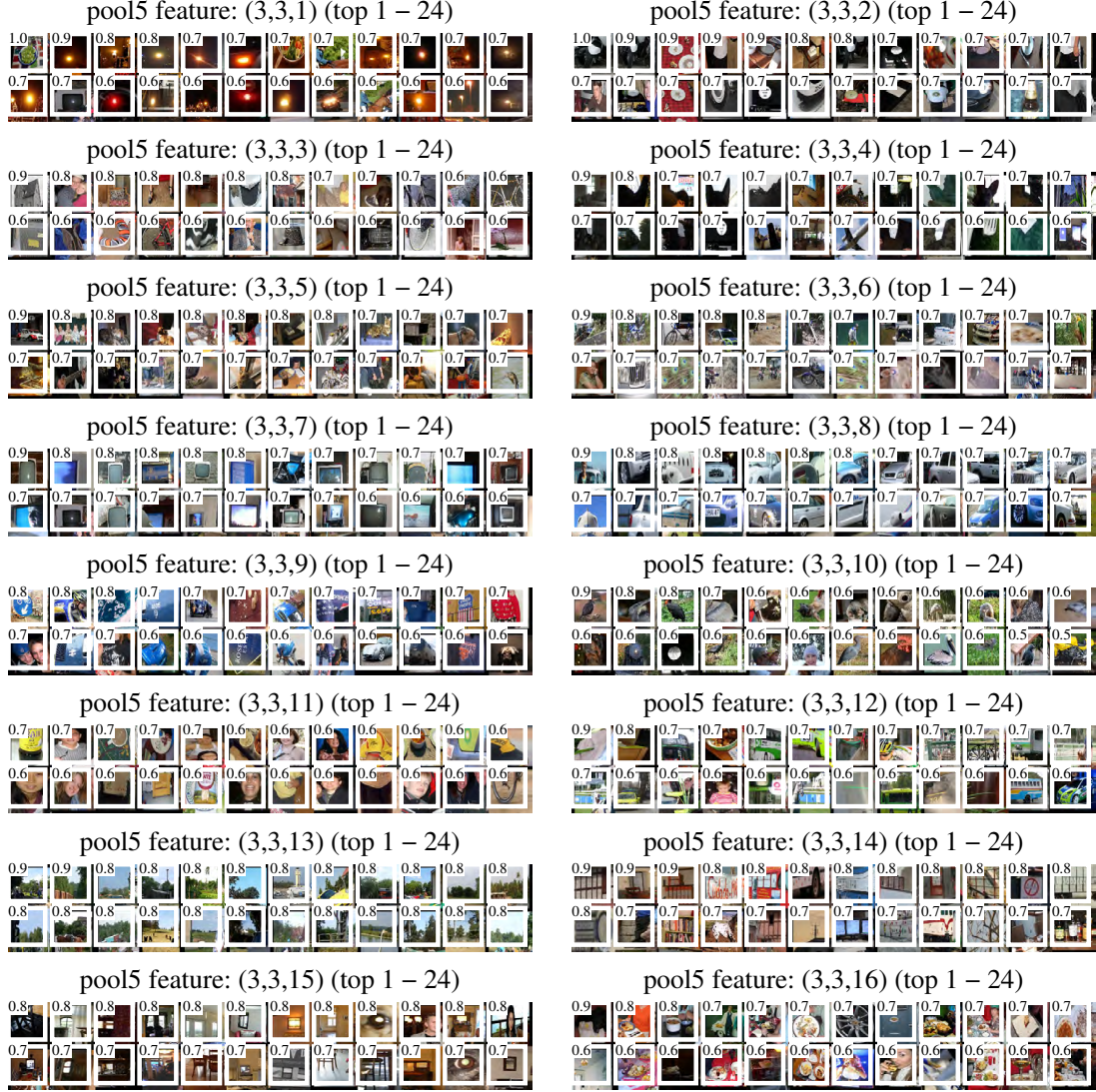


Figure 3.12. We show the 24 region proposals, out of the approximately 10 million regions in VOC 2007 test, that most strongly activate each of 20 units. Each montage is labeled by the unit's (y, x, channel) position in the  $6 \times 6 \times 256$  dimensional pool<sub>5</sub> feature map. Each image region is drawn with an overlay of the unit's receptive field in white. The activation value (which we normalize by dividing by the max activation value over all units in a channel) is shown in the receptive field's upper-left corner. Best viewed digitally with zoom.

## Chapter 4

# Long-term Recurrent Convolutional Networks for Visual Recognition and Description<sup>1</sup>

Recognition and description of images and videos is a fundamental challenge of computer vision. Dramatic progress has been achieved by supervised convolutional neural network (CNN) models on image recognition tasks, and a number of extensions to process video have been recently proposed. Ideally, a video model should allow processing of variable length input sequences, and also provide for variable length outputs, including generation of full-length sentence descriptions that go beyond conventional one-versus-all prediction tasks. In this chapter we propose *Long-term Recurrent Convolutional Networks* (LRCNs), a class of architectures for visual recognition and description which combines convolutional layers and long-range temporal recursion and is end-to-end trainable (Figure 4.1). We instantiate our architecture for specific video activity recognition, image caption generation, and video description tasks as described below.

Research on CNN models for video processing has considered learning 3D spatio-temporal filters over raw sequence data (Baccouche et al., 2011; Ji et al., 2013), and learning of frame-to-frame representations which incorporate instantaneous optic flow or trajectory-based models aggregated over fixed windows or video shot segments (Karpathy et al., 2014a; Simonyan and Zisserman, 2014). Such models explore two extrema of perceptual time-series representation learning: either learn a fully general time-varying weighting, or apply simple temporal pooling. Following the same inspiration that motivates current deep convolutional models, we advocate for video

---

<sup>1</sup>This chapter is based on joint work with Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell (Donahue et al., 2015, 2016), presented primarily as it appears in *TPAMI* 2016, an extended version of the CVPR 2015 paper. Statements throughout the chapter (such as references to “prior work”) should be read with this context in mind.

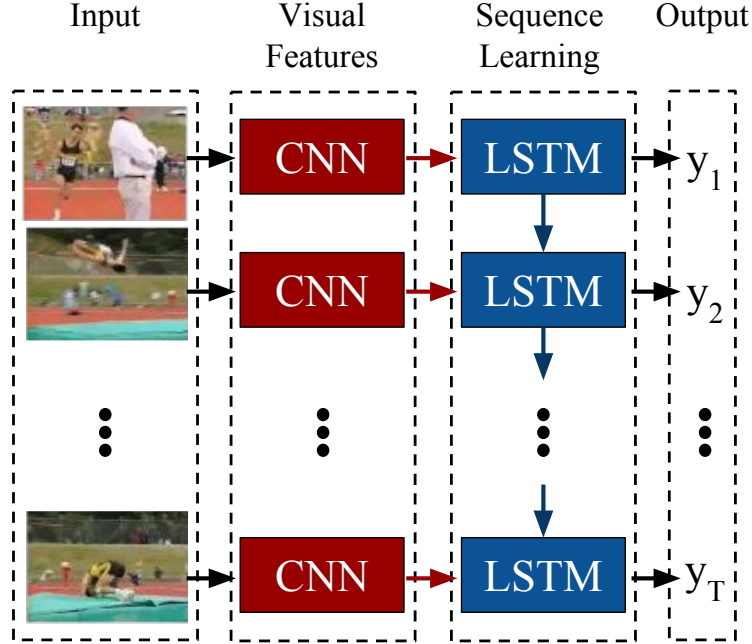


Figure 4.1. We propose *Long-term Recurrent Convolutional Networks* (LRCNs), a class of architectures leveraging the strengths of rapid progress in CNNs for visual recognition problems, and the growing desire to apply such models to time-varying inputs and outputs. LRCN processes the (possibly) variable-length visual input (left) with a CNN (middle-left), whose outputs are fed into a stack of recurrent sequence models (*LSTMs*, middle-right), which finally produce a variable-length prediction (right). Both the CNN and LSTM weights are shared across time, resulting in a representation that scales to arbitrarily long sequences.

recognition and description models which are also deep over temporal dimensions; i.e., have temporal recurrence of latent variables. Recurrent Neural Network (RNN) models are “deep in time” – explicitly so when unrolled – and form implicit compositional representations in the time domain. Such “deep” models predated deep spatial convolution models in the literature (Rumelhart et al., 1985; Williams and Zipser, 1989).

The use of RNNs in perceptual applications has been explored for many decades, with varying results. A significant limitation of simple RNN models which strictly integrate state information over time is known as the “vanishing gradient” effect: the ability to backpropagate an error signal through a long-range temporal interval becomes increasingly difficult in practice. *Long Short-Term Memory* (LSTM) units, first proposed in Hochreiter and Schmidhuber (1997), are recurrent modules which enable long-range learning. LSTM units have hidden state augmented with nonlinear mechanisms to allow state to propagate without modification, be updated, or be reset,



using simple learned gating functions. LSTMs have recently been demonstrated to be capable of large-scale learning of speech recognition (Graves and Jaitly, 2014) and language translation models (Cho et al., 2014a; Sutskever et al., 2014).

We show here that convolutional networks with recurrent units are generally applicable to visual time-series modeling, and argue that in visual tasks where static or flat temporal models have previously been employed, LSTM-style RNNs can provide significant improvement when ample training data are available to learn or refine the representation. Specifically, we show that LSTM type models provide for improved recognition on conventional video activity challenges and enable a novel end-to-end optimizable mapping from image pixels to sentence-level natural language descriptions. We also show that these models improve generation of descriptions from intermediate visual representations derived from conventional visual models.

We instantiate our proposed architecture in three experimental settings (Figure 4.3). First, we show that directly connecting a visual convolutional model to deep LSTM networks, we are able to train video recognition models that capture temporal state dependencies (Figure 4.3 left; Section 4.3). While existing labeled video activity datasets may not have actions or activities with particularly complex temporal dynamics, we nonetheless observe significant improvements on conventional benchmarks.

Second, we explore end-to-end trainable image to sentence mappings. Strong results for machine translation tasks have recently been reported (Cho et al., 2014a; Sutskever et al., 2014); such models are encoder-decoder pairs based on LSTM networks. We propose a multimodal analog of this model, and describe an architecture which uses a visual convnet to encode a deep state vector, and an LSTM to decode the vector into a natural language string (Figure 4.3 middle; Section 4.4). The resulting model can be trained end-to-end on large-scale image and text datasets, and even with modest training provides competitive generation results compared to existing methods.

Finally, we show that LSTM decoders can be driven directly from conventional computer vision methods which predict higher-level discriminative labels, such as the semantic video role tuple predictors in Rohrbach et al. (2013) (Figure 4.3, right; Section 4.5). While not end-to-end trainable, such models offer architectural and performance advantages over previous statistical machine translation-based approaches.

## 4.1 Background

Traditional recurrent neural networks (RNNs, Figure 4.2, left) model temporal dynamics by mapping input sequences to hidden states, and hidden states to outputs

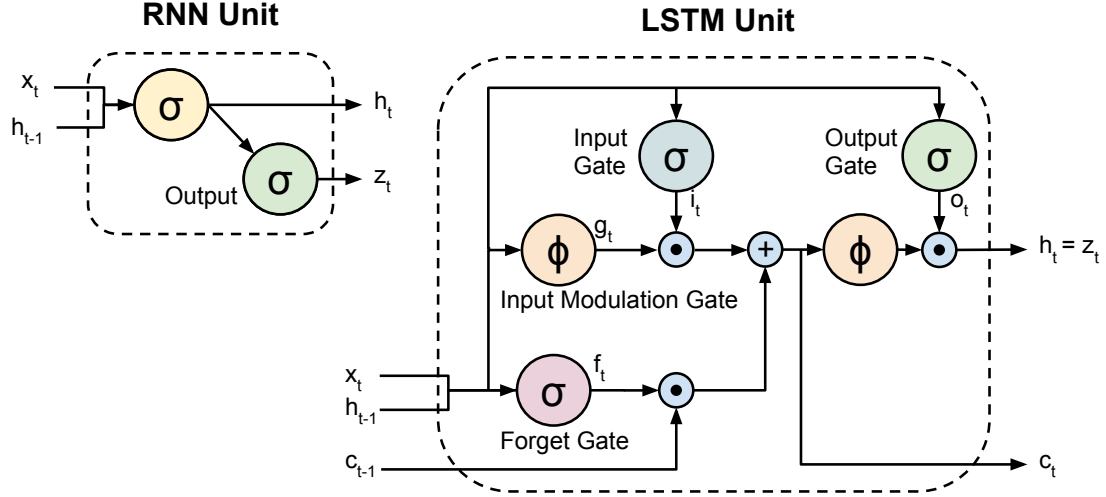


Figure 4.2. A diagram of a basic RNN cell (left) and an LSTM memory cell (right) used in this chapter (from Zaremba and Sutskever (2014), a slight simplification of the architecture described in Graves (2013), which was derived from the LSTM initially proposed in Hochreiter and Schmidhuber (1997)).

via the following recurrence equations (Figure 4.2, left):

$$\begin{aligned} h_t &= g(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \\ z_t &= g(W_{hz}h_t + b_z) \end{aligned}$$

where  $g$  is an element-wise non-linearity, such as a sigmoid or hyperbolic tangent,  $x_t$  is the input,  $h_t \in \mathbb{R}^N$  is the hidden state with  $N$  hidden units, and  $z_t$  is the output at time  $t$ . For a length  $T$  input sequence  $\langle x_1, x_2, \dots, x_T \rangle$ , the updates above are computed sequentially as  $h_1$  (letting  $h_0 = 0$ ),  $z_1$ ,  $h_2$ ,  $z_2$ ,  $\dots$ ,  $h_T$ ,  $z_T$ .

Though RNNs have proven successful on tasks such as speech recognition (Vinyals et al., 2012) and text generation (Sutskever et al., 2011), it can be difficult to train them to learn long-term dynamics, likely due in part to the vanishing and exploding gradients problem (Hochreiter and Schmidhuber, 1997) that can result from propagating the gradients down through the many layers of the recurrent network, each corresponding to a particular time step. LSTMs provide a solution by incorporating memory units that explicitly allow the network to learn when to “forget” previous hidden states and when to update hidden states given new information. As research on LSTMs has progressed, hidden units with varying connections within the memory unit have been proposed. We use the LSTM unit as described in Zaremba and Sutskever (2014) (Figure 4.2, right), a slight simplification of the one described in Graves and Jaitly (2014), which was derived from the original LSTM unit proposed in Hochreiter and Schmidhuber (1997). Letting  $\sigma(x) = (1 + e^{-x})^{-1}$  be the *sigmoid* non-linearity which

squashes real-valued inputs to a  $[0, 1]$  range, and letting  $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1$  be the *hyperbolic tangent* non-linearity, similarly squashing its inputs to a  $[-1, 1]$  range, the LSTM updates for time step  $t$  given inputs  $x_t$ ,  $h_{t-1}$ , and  $c_{t-1}$  are:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ g_t &= \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned}$$

$x \odot y$  denotes the element-wise product of vectors  $x$  and  $y$ .

In addition to a hidden unit  $h_t \in \mathbb{R}^N$ , the LSTM includes an input gate  $i_t \in \mathbb{R}^N$ , forget gate  $f_t \in \mathbb{R}^N$ , output gate  $o_t \in \mathbb{R}^N$ , input modulation gate  $g_t \in \mathbb{R}^N$ , and memory cell  $c_t \in \mathbb{R}^N$ . The memory cell unit  $c_t$  is a sum of two terms: the previous memory cell unit  $c_{t-1}$  which is modulated by  $f_t$ , and  $g_t$ , a function of the current input and previous hidden state, modulated by the input gate  $i_t$ . Because  $i_t$  and  $f_t$  are sigmoidal, their values lie within the range  $[0, 1]$ , and  $i_t$  and  $f_t$  can be thought of as knobs that the LSTM learns to selectively forget its previous memory or consider its current input. Likewise, the output gate  $o_t$  learns how much of the memory cell to transfer to the hidden state. These additional cells seem to enable the LSTM to learn complex and long-term temporal dynamics for a wide variety of sequence learning and prediction tasks. Additional depth can be added to LSTMs by stacking them on top of each other, using the hidden state  $h_t^{(\ell-1)}$  of the LSTM in layer  $\ell - 1$  as the input to the LSTM in layer  $\ell$ .

Recently, LSTMs have achieved impressive results on language tasks such as speech recognition (Graves and Jaitly, 2014) and machine translation (Cho et al., 2014a; Sutskever et al., 2014). Analogous to CNNs, LSTMs are attractive because they allow end-to-end fine-tuning. For example, Graves and Jaitly (2014) eliminates the need for complex multi-step pipelines in speech recognition by training a deep bidirectional LSTM which maps spectrogram inputs to text. Even with no language model or pronunciation dictionary, the model produces convincing text translations. Sutskever et al. (2014) and Cho et al. (2014a) translate sentences from English to French with a multi-layer LSTM encoder and decoder. Sentences in the source language are mapped to a hidden state using an encoding LSTM, and then a decoding LSTM maps the hidden state to a sequence in the target language. Such an encoder-decoder scheme allows an input sequence of arbitrary length to be mapped to an output sequence of different length. The sequence-to-sequence architecture for machine translation circumvents the need for language models.

The advantages of LSTMs for modeling sequential data in vision problems are twofold. First, when integrated with current vision systems, LSTM models are straightforward to fine-tune end-to-end. Second, LSTMs are not confined to fixed

length inputs or outputs allowing simple modeling for sequential data of varying lengths, such as text or video. We next describe a unified framework to combine recurrent models such as LSTMs with deep convolutional networks to form end-to-end trainable networks capable of complex visual and sequence prediction tasks.

## 4.2 Long-term Recurrent Convolutional Networks

This work proposes a Long-term Recurrent Convolutional Network (LRCN) model combining a deep hierarchical visual feature extractor (such as a CNN) with a model that can learn to recognize and synthesize temporal dynamics for tasks involving sequential data (inputs or outputs), visual, linguistic, or otherwise. Figure 4.1 depicts the core of our approach. LRCN works by passing each visual input  $x_t$  (an image in isolation, or a frame from a video) through a feature transformation  $\phi_V(\cdot)$  with parameters  $V$ , usually a CNN, to produce a fixed-length vector representation  $\phi_V(x_t)$ . The outputs of  $\phi_V$  are then passed into a recurrent sequence learning module.

In its most general form, a recurrent model has parameters  $W$ , and maps an input  $x_t$  and a previous time step hidden state  $h_{t-1}$  to an output  $z_t$  and updated hidden state  $h_t$ . Therefore, inference must be run sequentially (i.e., from top to bottom, in the *Sequence Learning* box of Figure 4.1), by computing in order:  $h_1 = f_W(x_1, h_0) = f_W(x_1, 0)$ , then  $h_2 = f_W(x_2, h_1)$ , etc., up to  $h_T$ . Some of our models stack multiple LSTMs atop one another as described in Section 4.1.

To predict a distribution  $P(y_t)$  over outcomes  $y_t \in C$  (where  $C$  is a discrete, finite set of outcomes) at time step  $t$ , the outputs  $z_t \in \mathbb{R}^{d_z}$  of the sequential model are passed through a linear prediction layer  $\hat{y}_t = W_z z_t + b_z$ , where  $W_z \in \mathbb{R}^{|C| \times d_z}$  and  $b_z \in \mathbb{R}^{|C|}$  are learned parameters. Finally, the predicted distribution  $P(y_t)$  is computed by taking the softmax of  $\hat{y}_t$ :  $P(y_t = c) = \text{softmax}(\hat{y}_t) = \frac{\exp(\hat{y}_{t,c})}{\sum_{c' \in C} \exp(\hat{y}_{t,c'})}$ .

The success of recent deep models for object recognition (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; Szegedy et al., 2015) suggests that strategically composing many “layers” of non-linear functions can result in powerful models for perceptual problems. For large  $T$ , the above recurrence indicates that the last few predictions from a recurrent network with  $T$  time steps are computed by a very “deep” ( $T$  layer) non-linear function, suggesting that the resulting recurrent model may have similar representational power to a  $T$  layer deep network. Critically, however, the sequence model’s weights  $W$  are reused at every time step, forcing the model to learn generic time step-to-time step dynamics (as opposed to dynamics conditioned on  $t$ , the sequence index) and preventing the parameter size from growing in proportion to the maximum sequence length.

In most of our experiments, the visual feature transformation  $\phi$  corresponds to the activations in some layer of a deep CNN. Using a visual transformation  $\phi_V(\cdot)$  which is time-invariant and independent at each time step has the important advantage of



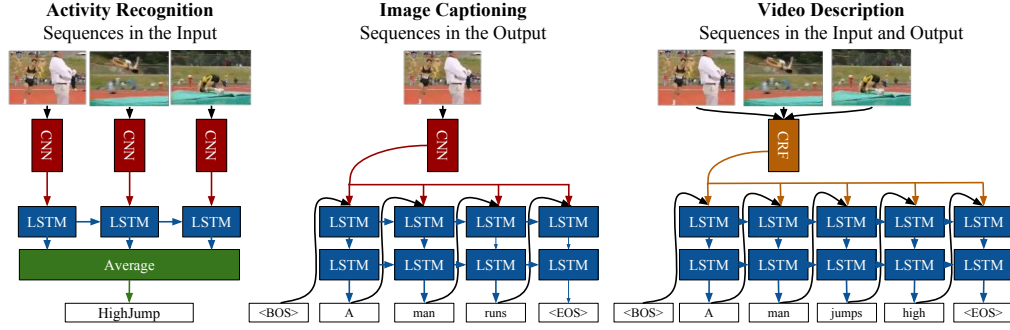


Figure 4.3. Task-specific instantiations of our LRCN model for activity recognition, image description, and video description.

making the expensive convolutional inference and training parallelizable over all time steps of the input, facilitating the use of fast contemporary CNN implementations whose efficiency relies on independent batch processing, and end-to-end optimization of the visual and sequential model parameters  $V$  and  $W$ .

We consider three vision problems (activity recognition, image description and video description), each of which instantiates one of the following broad classes of sequential learning tasks:

1. Sequential input, static output (Figure 4.3, left):  $\langle x_1, x_2, \dots, x_T \rangle \mapsto y$ . The visual activity recognition problem can fall under this umbrella, with videos of arbitrary length  $T$  as input, but with the goal of predicting a single label like *running* or *jumping* drawn from a fixed vocabulary.
2. Static input, sequential output (Figure 4.3, middle):  $x \mapsto \langle y_1, y_2, \dots, y_T \rangle$ . The image captioning problem fits in this category, with a static (non-time-varying) image as input, but a much larger and richer label space consisting of *sentences* of any length.
3. Sequential input and output (Figure 4.3, right):  $\langle x_1, x_2, \dots, x_T \rangle \mapsto \langle y_1, y_2, \dots, y_{T'} \rangle$ . In tasks such as video description, both the visual input and output are time-varying, and in general the number of input and output time steps may differ (i.e., we may have  $T \neq T'$ ). In video description, for example, the number of frames in the video should not constrain the length of (number of words in) the natural language description.

In the previously described generic formulation of recurrent models, each instance has  $T$  inputs  $\langle x_1, x_2, \dots, x_T \rangle$  and  $T$  outputs  $\langle y_1, y_2, \dots, y_T \rangle$ . Note that this formulation does not align cleanly with any of the three problem classes described above – in the first two classes, either the input or output is static, and in the third class, the input

length  $T$  need not match the output length  $T'$ . Hence, we describe how we adapt this formulation in our hybrid model to each of the above three problem settings.

With sequential inputs and static outputs (class 1), we take a late-fusion approach to merging the per-time step predictions  $\langle y_1, y_2, \dots, y_T \rangle$  into a single prediction  $y$  for the full sequence. With static inputs  $x$  and sequential outputs (class 2), we simply duplicate the input  $x$  at all  $T$  time steps:  $\forall t \in \{1, 2, \dots, T\} : x_t := x$ . Finally, for a sequence-to-sequence problem with (in general) different input and output lengths (class 3), we take an “encoder-decoder” approach, as proposed for machine translation by Cho et al. (2014b); Sutskever et al. (2014). In this approach, one sequence model, the *encoder*, maps the input sequence to a fixed-length vector, and another sequence model, the *decoder*, unrolls this vector to a sequential output of arbitrary length. Under this type of model, a run of the full system on one instance occurs over  $T + T' - 1$  time steps. For the first  $T$  time steps, the encoder processes the input  $x_1, x_2, \dots, x_T$ , and the decoder is inactive until time step  $T$ , when the encoder’s output is passed to the decoder, which in turn predicts the first output  $y_1$ . For the latter  $T' - 1$  time steps, the decoder predicts the remainder of the output  $y_2, y_3, \dots, y_{T'}$  with the encoder inactive. This encoder-decoder approach, as applied to the video description task, is depicted in Section 4.5, Figure 4.5 (left).

Under the proposed system, the parameters  $(V, W)$  of the model’s visual and sequential components can be jointly optimized by maximizing the likelihood of the ground truth outputs  $y_t$  at each time step  $t$ , conditioned on the input data and labels up to that point  $(x_{1:t}, y_{1:t-1})$ . In particular, for a training set  $\mathcal{D}$  of labeled sequences  $(x_t, y_t)_{t=1}^T \in \mathcal{D}$ , we optimize parameters  $(V, W)$  to minimize the expected negative log likelihood of a sequence sampled from the training set  $\mathcal{L}(V, W, \mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{(x_t, y_t)_{t=1}^T \in \mathcal{D}} \sum_{t=1}^T \log P(y_t | x_{1:t}, y_{1:t-1}, V, W)$ .

One of the most appealing aspects of the described system is the ability to learn the parameters “end-to-end,” such that the parameters  $V$  of the visual feature extractor learn to pick out the aspects of the visual input that are relevant to the sequential classification problem. We train our LRCN models using stochastic gradient descent, with backpropagation used to compute the gradient  $\nabla_{V, W} \mathcal{L}(V, W, \tilde{\mathcal{D}})$  of the objective  $\mathcal{L}$  with respect to all parameters  $(V, W)$  over minibatches  $\tilde{\mathcal{D}} \subset \mathcal{D}$  sampled from the training dataset  $\mathcal{D}$ .

We next demonstrate the power of end-to-end trainable hybrid convolutional and recurrent networks by exploring three applications: activity recognition, image captioning, and video description.

### 4.3 Activity recognition

Activity recognition is an instance of the first class of sequential learning tasks described above: each frame in a length  $T$  sequence is the input to a single convolutional network (i.e., the convnet weights are tied across time). We consider both RGB and

flow as inputs to our recognition system. Flow is computed with Brox et al. (2004) and transformed into a “flow image” by scaling and shifting  $x$  and  $y$  flow values to a range of  $[-128, +128]$ . A third channel for the flow image is created by calculating the flow magnitude.

During training, videos are resized to  $240 \times 320$  and we augment our data by using  $227 \times 227$  crops and mirroring. Additionally, we train the LRCN networks with video clips of 16 frames, even though the UCF101 videos are generally much longer (on the order of 100 frames when extracting frames at 30 FPS). Training on shorter video clips can be seen as analogous to training on image crops and is a useful method of data augmentation. LRCN is trained to predict the video’s activity class at each time step. To produce a single label prediction for an entire video clip, we average the label probabilities – the outputs of the network’s softmax layer – across all frames and choose the most probable label. At test time, we extract 16 frame clips with a stride of 8 frames from each video and average across all clips from a single video.

The CNN base of LRCN in our activity recognition experiments is a hybrid of the *CaffeNet* (Jia et al., 2014) reference model (a minor variant of *AlexNet* (Krizhevsky et al., 2012)) and the network used by Zeiler & Fergus (Zeiler and Fergus, 2014). The network is pre-trained on the 1.2M image ILSVRC-2012 (Russakovsky et al., 2015) classification training subset of the ImageNet (Deng et al., 2009) dataset, giving the network a strong initialization to facilitate faster training and avoid overfitting to the relatively small video activity recognition datasets. When classifying center crops, the top-1 classification accuracy is 60.2% and 57.4% for the hybrid and *CaffeNet* reference models, respectively.

We compare LRCN to a single frame baseline model. In our baseline model,  $T$  video frames are individually classified by a CNN. As in the LSTM model, whole video classification is done by averaging scores across all video frames.

### 4.3.1 Evaluation

We evaluate our architecture on the UCF101 dataset (Soomro et al., 2012) which consists of over 12,000 videos categorized into 101 human action classes. The dataset is split into three splits, with just under 8,000 videos in the training set for each split.

We explore various hyperparameters for the LRCN activity recognition architecture. To explore different variants, we divide the first training split of UCF101 into a smaller training set ( $\approx 6,000$  videos) and a validation set ( $\approx 3,000$  videos). We find that the most influential hyperparameters include the number of hidden units in the LSTM and whether  $fc_6$  or  $fc_7$  features are used as input to the LSTM. We compare networks with 256, 512, and 1024 LSTM hidden units. When using flow as an input, more hidden units leads to better performance with 1024 hidden units yielding a 1.7% boost in accuracy in comparison to a network with 256 hidden units on our validation set. In contrast, for networks with RGB input, the number of hidden units has little impact on the performance of the model. We thus use 1024 hidden units for flow

inputs, and 256 for RGB inputs. We find that using  $fc_6$  as opposed to  $fc_7$  features improves accuracy when using flow as input on our validation set by 1%. When using RGB images as input, the difference between using  $fc_6$  or  $fc_7$  features is quite small; using  $fc_6$  features only increases accuracy by 0.2%. Because both models perform better with  $fc_6$  features, we train our final models using  $fc_6$  features (denoted by LRCN- $fc_6$ ). We also considered subsampling the frames input to the LSTM, but found that this hurts performance compared with using all frames. Additionally, when training the LRCN network end-to-end, we found that aggressive dropout (0.9) was needed to avoid overfitting.

Table 4.1 reports the average accuracy across the three standard test splits of UCF101. Columns 2-3, compare video classification of LRCN against the baseline single frame architecture for both RGB and flow inputs. LRCN yields the best results for both RGB and flow and improves upon the baseline network by 0.83% and 2.91% respectively. RGB and flow networks can be combined by computing a weighted average of network scores as proposed in Simonyan and Zisserman (2014). Like Simonyan and Zisserman (2014), we report two weighted averages of the predictions from the RGB and flow networks in Table 4.1 (right). Since the flow network outperforms the RGB network, weighting the flow network higher unsurprisingly leads to better accuracy. In this case, LRCN outperforms the baseline single-frame model by 3.40%.

Table 4.2 compares LRCN’s accuracy with the single frame baseline model for individual classes on Split 1 of UCF101. For the majority of classes, LRCN improves performance over the single frame model. Though LRCN performs worse on some classes including *Knitting* and *Mixing*, in general when LRCN performs worse, the loss in accuracy is not as substantial as the gain in accuracy for classes like *BoxingPunchingBag* and *HighJump*. Consequently, accuracy is higher overall.

Table 4.3 compares accuracies for the LRCN flow and LRCN RGB models for individual classes on Split 1 of UCF101. Note that for some classes the LRCN flow model outperforms the LRCN RGB model and vice versa. One explanation is that activities which are better classified by the LRCN RGB model are best determined by which objects are present in the scene, while activities which are better classified by the LRCN flow model are best classified by the kind of motion in the scene. For example, activity classes like *Typing* are highly correlated with the presence of certain objects, such as a keyboard, and are thus best learned by the LRCN RGB model. Other activities such as *SoccerJuggling* include more generic objects which are frequently seen in other activities (soccer balls, people) and are thus best identified from class-specific motion cues. Because RGB and flow signals are complementary, the best models take both into account.

LRCN shows clear improvement over the baseline single-frame system and is comparable to accuracy achieved by other deep models. Simonyan and Zisserman (2014) report the results on UCF101 by computing a weighted average between flow and RGB networks and achieve 87.6%. Karpathy et al. (2014a) reports 65.4% accuracy on UCF101, which is substantially lower than LRCN.

Model	Single Input Type		Weighted Average	
	RGB	Flow	$1/2, 1/2$	$1/3, 2/3$
Single frame	67.37	74.37	75.46	78.94
LRCN- $\text{fc}_6$	<b>68.20</b>	<b>77.28</b>	80.90	<b>82.34</b>

Table 4.1. Activity recognition: Comparing single frame models to LRCN networks for activity recognition on the UCF101 (Soomro et al., 2012) dataset, with RGB and flow inputs. Average values across all three splits are shown. LRCN consistently and strongly outperforms a model based on predictions from the underlying convolutional network architecture alone.

Label	$\Delta$	Label	$\Delta$
BoxingPunchingBag	40.82	BoxingSpeedBag	-16.22
HighJump	29.73	Mixing	-15.56
JumpRope	28.95	Knitting	-14.71
CricketShot	28.57	Typing	-13.95
Basketball	28.57	Skiing	-12.50
WallPushups	25.71	BaseballPitch	-11.63
Nunchucks	22.86	BrushingTeeth	-11.11
ApplyEyeMakeup	22.73	Skijet	-10.71
HeadMassage	21.95	Haircut	-9.10
Drumming	17.78	TennisSwing	-8.16

Table 4.2. Activity recognition: comparison of improvement  $\Delta$  in LRCN’s per-class recognition accuracy versus the single-frame baseline. Here we report results on all three splits of UCF101 (Soomro et al., 2012).  $\Delta$  is the difference between LRCN’s accuracy and the single-frame model’s accuracy.

Label	$\Delta$	Label	$\Delta$
BoxingPunchingBag	57.14	Typing	-44.19
PushUps	53.33	TennisSwing	-42.86
JumpRope	50.00	FieldHockeyPenalty	-32.50
SoccerJuggling	48.72	BrushingTeeth	-30.56
HandstandWalking	44.12	CuttingInKitchen	-30.30
Basketball	40.00	Skijet	-28.57
BodyWeightSquats	38.46	Mixing	-26.67
Lunges	37.84	Skiing	-25.00
Nunchucks	34.29	Knitting	-20.59
WallPushups	34.29	FloorGymnastics	-19.44

Table 4.3. Activity recognition: comparison of per-class recognition accuracy between the flow and RGB LRCN models.  $\Delta$  is the difference between LRCN flow accuracy and LRCN RGB accuracy.

## 4.4 Image captioning

In contrast to activity recognition, the static image captioning task requires only a single invocation of a convolutional network since the input consists of a single image. At each time step, both the image features and the previous word are provided as inputs to the sequence model, in this case a stack of LSTMs (each with 1000 hidden units), which is used to learn the dynamics of the time-varying output sequence, natural language.

At time step  $t$ , the input to the bottom-most LSTM is the embedded word from the previous time step  $y_{t-1}$ . Input words are encoded as “one-hot” vectors: vectors  $y \in \mathbb{R}^K$  with a single non-zero component  $y_i = 1$  denoting the  $i^{\text{th}}$  word in the vocabulary, where  $K$  is the number of words in the vocabulary, plus one additional entry for the <BOS> (beginning of sequence) token which is always taken as  $y_0$ , the “previous word” at the first time step ( $t = 1$ ). These one-hot vectors are then projected into an embedding space with dimension  $d_e$  by multiplication  $W_e y_t$  with a learned parameter matrix  $W_e \in \mathbb{R}^{d_e \times K}$ . The result of a matrix-vector multiplication with a one-hot vector is the column of the matrix corresponding to the index of the single non-zero component of the one-hot vector.  $W_e$  can therefore be thought of as a “lookup table,” mapping each of the  $K$  words in the vocabulary to a  $d_e$ -dimensional vector.

The visual feature representation  $\phi_V(x)$  of the image  $x$  may be input to the sequence model – a stack of  $L$  LSTMs – by concatenating it at each time step either with (1) the embedded previous word  $W_e y_{t-1}$  and fed into the first LSTM of the stack, or (2) the hidden state  $h_t^{(\ell-1)}$  output from LSTM  $\ell - 1$  and fed into LSTM  $\ell$ , for some  $\ell \in 2, \dots, L$ . These choices are depicted in Figure 4.4. We refer to the latter choice as

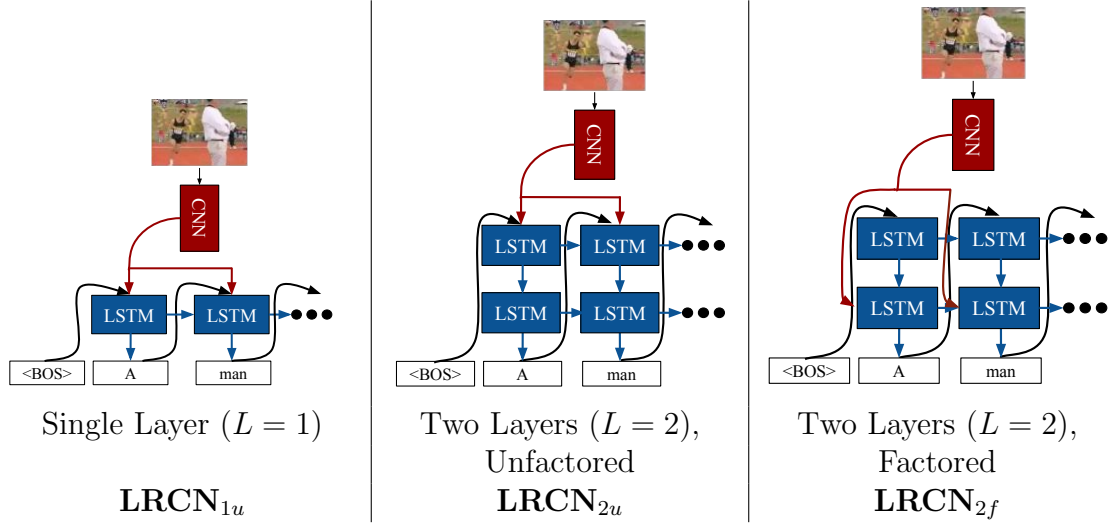


Figure 4.4. Three variants of the LRCN image captioning architecture that we experimentally evaluate. We explore the effect of depth in the LSTM stack, and the effect of the “factorization” of the modalities.

“factored,” as it forces a sort of separation of responsibilities by “blinding” the first  $\ell - 1$  LSTMs and forcing all of the capacity of their hidden states at time step  $t$  to represent only the partial caption  $y_{1:t-1}$  independent of the visual input, while the LSTMs starting from  $\ell$  are responsible for fusing the lower layer’s hidden state given by the partial caption with the visual feature representation  $\phi_V(x)$  to produce a joint hidden state representation  $h_t^{(\ell)}$  of the visual and language inputs up to time step  $t$  from which the next word  $y_t$  can be predicted. In the factored case, the hidden state  $h_t$  for the lower layers is conditionally independent of the image  $x$  given the partial caption  $y_{1:t-1}$ .

The outputs of the final LSTM in the stack are the inputs to a learned linear prediction layer with a softmax producing a distribution  $P(y_t|y_{1:t-1}, \phi_V(x))$  over words  $y_t$  in the model’s vocabulary, including the  $\langle \text{EOS} \rangle$  token denoting the end of the caption, allowing the model to predict captions of varying length. The visual model  $\phi_V$  used for our image captioning experiments is either the *CaffeNet* (Jia et al., 2014) reference model, a variant of *AlexNet* (Krizhevsky et al., 2012), or the more modern and computationally expensive *VGGNet* (Simonyan and Zisserman, 2015) model pre-trained for ILSVRC-2012 (Russakovsky et al., 2015) classification.

Without any explicit language modeling or impositions on the structure of the generated captions, the described LRCN system learns mappings from images input as pixel intensity values to natural language descriptions that are often semantically descriptive and grammatically correct.

At training time, the previous word inputs  $y_{1:t-1}$  at time step  $t$  are from the *ground truth* caption. For inference of captions on a novel image  $x$ , the input is a sample

	<b>R@1</b>	<b>R@5</b>	<b>R@10</b>	<b>Med<math>r</math></b>
Caption to Image (Flickr30k)				
DeViSE (Frome et al., 2013)	6.7	21.9	32.7	25
SDT-RNN (Socher et al., 2014)	8.9	29.8	41.1	16
DeFrag (Karpathy et al., 2014b)	10.3	31.4	44.5	13
m-RNN (Mao et al., 2015a)	12.6	31.2	41.5	16
ConvNet (Kiros et al., 2015)	11.8	34.0	46.3	13
LRCN <sub>2f</sub> (ours)	<b>17.5</b>	<b>40.3</b>	<b>50.8</b>	<b>9</b>
Image to Caption (Flickr30k)				
DeViSE (Frome et al., 2013)	4.5	18.1	29.2	26
SDT-RNN (Socher et al., 2014)	9.6	29.8	41.1	16
DeFrag (Karpathy et al., 2014b)	16.4	40.2	54.7	8
m-RNN (Mao et al., 2015a)	18.4	40.2	50.9	10
ConvNet (Kiros et al., 2015)	14.8	39.2	50.9	10
LRCN <sub>2f</sub> (ours)	<b>23.6</b>	<b>46.6</b>	<b>58.3</b>	<b>7</b>

Table 4.4. Image description: retrieval results for the Flickr30k (Young et al., 2014) datasets. **R@K** is the average recall at rank  $K$  (high is good). **Med $r$**  is the median rank (low is good).

$\tilde{y}_t \sim P(y_t | \tilde{y}_{1:t-1}, \phi_V(x))$  from the model’s predicted distribution at the previous time step, and generation continues until an **<EOS>** (end of sequence) token is generated.

#### 4.4.1 Evaluation

We evaluate our image description model for retrieval and generation tasks. We first demonstrate the effectiveness of our model by quantitatively evaluating it on the image and caption retrieval tasks proposed by Hodosh et al. (2013) and seen in Frome et al. (2013); Karpathy et al. (2014b); Kiros et al. (2015); Mao et al. (2015a); Socher et al. (2014). We report results on Flickr30k (Young et al., 2014), and COCO 2014 (Lin et al., 2014) datasets, both with five captions annotated per image.

##### 4.4.1.1 Retrieval

Retrieval results on the Flickr30k (Young et al., 2014) dataset are recorded in Table 4.4. We report median rank, **Med $r$** , of the first retrieved ground truth image or caption and Recall@ $K$ , the number of images or captions for which a correct caption or image is retrieved within the top  $K$  results. Our model consistently outperforms the strong baselines from recent work (Frome et al., 2013; Karpathy et al., 2014b;



Kiros et al., 2015; Mao et al., 2015a; Socher et al., 2014) as can be seen in Table 4.4. Here, we note that the *VGGNet* model in Kiros et al. (2015) (called *OxfordNet* in their work) outperforms our model on the retrieval task. However, *VGGNet* is a stronger convolutional network (Simonyan and Zisserman, 2015) than that used for our results on this task. The strength of our sequence model (and integration of the sequence and visual models) can be more directly measured against the *ConvNet* (Kiros et al., 2015) result, which uses a very similar base CNN architecture (AlexNet (Krizhevsky et al., 2012), where we use CaffeNet) pretrained on the same data.

We also ablate the model’s retrieval performance on a randomly chosen subset of 1000 images (and 5000 captions) from the COCO 2014 (Lin et al., 2014) validation set. Results are recorded in Table 4.5. The first group of results for each task examines the effectiveness of an LSTM compared with a “vanilla” RNN as described in Section 4.1. These results demonstrate that the use of the LSTM unit compared to the simpler RNN architecture is an important element of our model’s performance on this task, justifying the additional complexity and suggesting that the LSTM’s gating mechanisms allowing for “long-term” memory may be quite useful, even for relatively simple sequences.

Within the second and third result groups, we compare performance among the three sequence model architectural variants depicted in Figure 4.4. For both tasks and under all metrics, the two layer, unfactored variant ( $\text{LRCN}_{2u}$ ) performs worse than the other two. The fact that  $\text{LRCN}_{1u}$  outperforms  $\text{LRCN}_{2u}$  indicates that stacking additional LSTM layers alone is not beneficial for this task. The other two variants ( $\text{LRCN}_{2f}$  and  $\text{LRCN}_{1u}$ ) perform similarly across the board, with  $\text{LRCN}_{2f}$  appearing to have a slight edge in the image to caption task under most metrics, but the reverse for caption to image retrieval.

Unsurprisingly, finetuning the CNN (indicated by the “FT?” column of Table 4.5) and using a more powerful CNN (*VGGNet* (Simonyan and Zisserman, 2015) rather than CaffeNet) each improve results substantially across the board. Finetuning boosts the  $\text{R@}k$  metrics by 3-5% for CaffeNet, and 5-8% for *VGGNet*. Switching from CaffeNet to *VGGNet* improves results by around 8-12% for the caption to image task, and by roughly 11-17% for the image to caption task.

#### 4.4.1.2 Generation

We evaluate LRCN’s caption generation performance on the COCO2014 (Lin et al., 2014) dataset using the official metrics on which COCO image captioning submissions are evaluated. The BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) metrics were designed for automatic evaluation of machine translation methods. ROUGE-L (Lin, 2004) was designed for evaluating summarization performance. CIDEr-D (Vedantam et al., 2015) was designed specifically to evaluate the image captioning task.

In Table 4.6 we evaluate variants of our model along the same axes as done for

Vision Model		Sequence Model			Retrieval Performance			
CNN	FT?	Unit	$L$	Factor?	R@1	R@5	R@10	Med $r$
					Caption to Image			
CaffeNet	-	RNN	2	✓	21.3	51.7	67.2	5
CaffeNet	-	LSTM	2	✓	<b>25.0</b>	<b>56.2</b>	<b>70.6</b>	<b>4</b>
CaffeNet	-	LSTM	1	-	<b>25.2</b>	<b>56.2</b>	<b>70.8</b>	<b>4</b>
CaffeNet	-	LSTM	2	-	23.4	54.8	69.3	5
CaffeNet	-	LSTM	2	✓	25.0	<b>56.2</b>	70.6	<b>4</b>
CaffeNet	✓	LSTM	1	-	<b>28.5</b>	<b>60.0</b>	74.5	<b>4</b>
CaffeNet	✓	LSTM	2	-	25.6	57.2	72.2	<b>4</b>
CaffeNet	✓	LSTM	2	✓	27.2	59.6	<b>74.7</b>	<b>4</b>
VGGNet	-	LSTM	2	✓	33.5	68.1	80.8	3
VGGNet	✓	LSTM	2	✓	<b>39.3</b>	<b>74.7</b>	<b>85.9</b>	<b>2</b>
					Image to Caption			
CaffeNet	-	RNN	2	✓	30.2	61.0	72.6	4
CaffeNet	-	LSTM	2	✓	<b>33.8</b>	<b>65.3</b>	<b>75.3</b>	<b>3</b>
CaffeNet	-	LSTM	1	-	32.3	64.5	<b>75.6</b>	<b>3</b>
CaffeNet	-	LSTM	2	-	29.9	60.8	72.7	<b>3</b>
CaffeNet	-	LSTM	2	✓	<b>33.8</b>	<b>65.3</b>	75.3	<b>3</b>
CaffeNet	✓	LSTM	1	-	36.1	<b>68.4</b>	79.5	3
CaffeNet	✓	LSTM	2	-	33.1	63.7	76.9	3
CaffeNet	✓	LSTM	2	✓	<b>36.3</b>	67.3	<b>80.6</b>	<b>2</b>
VGGNet	-	LSTM	2	✓	46.0	77.4	88.3	2
VGGNet	✓	LSTM	2	✓	<b>53.3</b>	<b>84.3</b>	<b>91.9</b>	<b>1</b>

Table 4.5. Retrieval results (image to caption and caption to image) for a randomly chosen subset (1000 images) of the COCO 2014 (Lin et al., 2014) validation set. R@ $K$  is the average recall at rank  $K$  (high is good). Med $r$  is the median rank (low is good).

Generation Strategy			Vision Model		Sequence Model			Generation Performance (COCO 2014 Validation Set)						
Beam Width	Sample $N$	Sample $T$	CNN	FT?	Unit	$L$	Factor?	B1	B2	B3	B4	C	M	R
1	-	-	CaffeNet	-	RNN	2	✓	0.638	0.454	0.315	0.220	0.660	0.209	0.473
1	-	-	CaffeNet	-	LSTM	2	✓	<b>0.646</b>	<b>0.462</b>	<b>0.321</b>	<b>0.224</b>	<b>0.674</b>	<b>0.210</b>	<b>0.477</b>
1	-	-	CaffeNet	-	LSTM	1	-	<b>0.654</b>	<b>0.475</b>	<b>0.333</b>	<b>0.231</b>	0.661	0.209	<b>0.480</b>
1	-	-	CaffeNet	-	LSTM	2	-	0.653	0.470	0.328	0.230	<b>0.682</b>	<b>0.212</b>	<b>0.480</b>
1	-	-	CaffeNet	-	LSTM	2	✓	0.646	0.462	0.321	0.224	0.674	0.210	0.477
1	-	-	CaffeNet	✓	LSTM	1	-	<b>0.661</b>	<b>0.485</b>	<b>0.344</b>	<b>0.241</b>	0.702	0.216	<b>0.489</b>
1	-	-	CaffeNet	✓	LSTM	2	-	0.659	0.478	0.338	0.238	0.716	0.217	0.486
1	-	-	CaffeNet	✓	LSTM	2	✓	0.659	0.478	0.336	0.237	<b>0.717</b>	<b>0.218</b>	0.486
1	-	-	VGGNet	-	LSTM	2	✓	0.674	0.494	0.351	0.248	0.773	0.227	0.497
1	-	-	VGGNet	✓	LSTM	2	✓	<b>0.695</b>	<b>0.519</b>	<b>0.374</b>	<b>0.268</b>	<b>0.839</b>	<b>0.237</b>	<b>0.512</b>
-	100	1.5	CaffeNet	-	RNN	2	✓	0.647	0.466	0.334	0.244	0.703	0.212	0.479
-	100	1.5	CaffeNet	-	LSTM	2	✓	<b>0.657</b>	<b>0.478</b>	<b>0.344</b>	<b>0.251</b>	<b>0.720</b>	<b>0.215</b>	<b>0.485</b>
-	100	1.5	CaffeNet	-	LSTM	1	-	<b>0.664</b>	<b>0.490</b>	<b>0.354</b>	0.254	0.704	0.211	0.488
-	100	1.5	CaffeNet	-	LSTM	2	-	<b>0.664</b>	0.486	0.352	<b>0.257</b>	<b>0.732</b>	<b>0.216</b>	<b>0.489</b>
-	100	1.5	CaffeNet	-	LSTM	2	✓	0.657	0.478	0.344	0.251	0.720	0.215	0.485
-	100	1.5	CaffeNet	✓	LSTM	1	-	<b>0.679</b>	<b>0.507</b>	<b>0.370</b>	<b>0.268</b>	0.753	0.219	<b>0.499</b>
-	100	1.5	CaffeNet	✓	LSTM	2	-	0.672	0.495	0.361	0.265	0.762	<b>0.222</b>	0.495
-	100	1.5	CaffeNet	✓	LSTM	2	✓	0.670	0.493	0.358	0.264	<b>0.764</b>	<b>0.222</b>	0.495
-	100	1.5	VGGNet	-	LSTM	2	✓	0.690	0.514	0.377	0.278	0.828	0.231	0.508
-	100	1.5	VGGNet	✓	LSTM	2	✓	<b>0.711</b>	<b>0.541</b>	<b>0.402</b>	<b>0.300</b>	<b>0.896</b>	<b>0.242</b>	<b>0.524</b>
1	-	-	VGGNet	✓	LSTM	2	✓	0.695	0.519	0.374	0.268	0.839	0.237	0.512
2	-	-	VGGNet	✓	LSTM	2	✓	0.707	0.533	0.394	0.291	0.879	0.242	0.520
3	-	-	VGGNet	✓	LSTM	2	✓	<b>0.708</b>	<b>0.536</b>	<b>0.399</b>	0.298	<b>0.888</b>	<b>0.243</b>	<b>0.521</b>
4	-	-	VGGNet	✓	LSTM	2	✓	0.706	0.534	0.398	0.299	<b>0.888</b>	<b>0.243</b>	<b>0.521</b>
5	-	-	VGGNet	✓	LSTM	2	✓	0.704	0.533	0.398	<b>0.300</b>	<b>0.888</b>	0.242	0.520
10	-	-	VGGNet	✓	LSTM	2	✓	0.699	0.528	0.395	0.298	0.886	0.241	0.518
-	1	2.0	VGGNet	✓	LSTM	2	✓	0.658	0.472	0.327	0.224	0.733	0.222	0.483
-	10	2.0	VGGNet	✓	LSTM	2	✓	0.708	0.534	0.391	0.286	0.868	0.239	0.519
-	25	2.0	VGGNet	✓	LSTM	2	✓	0.712	0.540	0.398	0.294	0.885	0.241	0.523
-	100	2.0	VGGNet	✓	LSTM	2	✓	<b>0.714</b>	<b>0.543</b>	<b>0.402</b>	<b>0.297</b>	<b>0.889</b>	<b>0.242</b>	<b>0.524</b>
-	100	1.0	VGGNet	✓	LSTM	2	✓	0.674	0.494	0.357	0.261	0.805	0.228	0.494
-	100	1.5	VGGNet	✓	LSTM	2	✓	0.711	0.541	<b>0.402</b>	<b>0.300</b>	<b>0.896</b>	<b>0.242</b>	<b>0.524</b>
-	100	2.0	VGGNet	✓	LSTM	2	✓	<b>0.714</b>	<b>0.543</b>	<b>0.402</b>	0.297	0.889	<b>0.242</b>	<b>0.524</b>

Table 4.6. Image caption generation performance (under the BLEU 1-4 (Papineni et al., 2002) (B1-B4), CIDEr-D (Vedantam et al., 2015) (C), METEOR (Banerjee and Lavie, 2005) (M), and ROUGE-L (Lin, 2004) (R) metrics) across various network architectures and generation strategies.

		Generation Performance (COCO 2014 Test Set)						
Method		B1	B2	B3	B4	C	M	R
Vinyals et al. (2015a)	NIC	0.895	0.802	0.694	0.587	<b>0.946</b>	<b>0.346</b>	<b>0.682</b>
Devlin et al. (2015a)	MSR Captivator	<b>0.907</b>	<b>0.819</b>	<b>0.710</b>	<b>0.601</b>	0.937	0.339	0.680
Mao et al. (2015b)	m-RNN (2015)	0.890	0.798	0.687	0.575	0.935	0.325	0.666
*	LRCN (sample)	0.895	0.804	0.695	0.585	0.934	0.335	0.678
Fang et al. (2015)	MSR	0.880	0.789	0.678	0.567	0.925	0.331	0.662
Devlin et al. (2015b)	Nearest Neighbor	0.872	0.770	0.655	0.542	0.916	0.318	0.648
Lin et al. (2014)	Human	0.880	0.744	0.603	0.471	0.910	0.335	0.626
Mao et al. (2015a)	m-RNN (2014)	0.890	0.801	0.690	0.578	0.896	0.320	0.668
Donahue et al. (2015)	LRCN (greedy)	0.871	0.772	0.653	0.534	0.891	0.322	0.656
Xu et al. (2015)	Show, Attend, and Tell	0.872	0.768	0.644	0.523	0.878	0.323	0.651
Kiros et al. (2015)	MLBL	0.848	0.747	0.633	0.517	0.752	0.294	0.635
Karpathy and Li (2015)	NeuralTalk	0.828	0.701	0.566	0.446	0.692	0.280	0.603

Table 4.7. Image caption generation results from top-performing methods in the 2015 COCO caption challenge competition, sorted by performance under the CIDEr-D metric. (We omit submissions that did not provide a reference to a report describing their method; see full results at <http://mscoco.org/dataset/#captions-leaderboard>.) All results except for our updated result (denoted by *LRCN, this work*) were competition entries (submitted by May 2015). Our updated result differs from our original competition entry only by generation strategy (sampling with  $N = 100$ ,  $T = 1.5$ , rather than beam search with width 1; i.e., greedy search); the visual and recurrent architectures (and trained weights) are the same.

the retrieval tasks in Table 4.5. In the topmost set of results, we show performance across various CNN and recurrent architectures for a simple generation strategy – beam search with beam width 1 (i.e., simply choosing the most probable word at each time step). In the middle set of results, we show performance across the same set of architectures for a more sophisticated and computationally intensive generation strategy found to be the best performing (in terms of performance under the CIDEr-D metric) among those explored in the bottom-most set of results, which explores various generation strategies while fixing the choice of network. In the first two sets of results, we vary the visual input CNN architecture (either CaffeNet (Jia et al., 2014), an architecture similar to AlexNet (Krizhevsky et al., 2012), or the more modern VGGNet (Simonyan and Zisserman, 2015)) and whether its weights are finetuned (FT?). Keeping the visual input CNN fixed with CaffeNet, we also vary the choice of recurrent architecture, comparing a stack of “vanilla” RNNs with LSTMs (Hochreiter and Schmidhuber, 1997), as well as the number of layers in the stack  $L$ , and (for  $L = 2$ ) whether the layers are “factored” (i.e., whether the visual input is passed into the second layer).

In the last of the three groups of results, we additionally explore and evaluate various caption generation strategies that can be employed for a given network. The simplest strategy, and the one employed for most of our generation results in our prior work (Donahue et al., 2015), is to generate captions greedily; i.e., by simply choosing the most probable word at each time step. This is equivalent to (and denoted in Table 4.6 by) beam search with beam width 1. In general, beam search with beam width  $N$  approximates the most likely caption by retaining and expanding only the  $N$  current most likely partial captions, according to the model. We find that of the beam search strategies, a beam width of 3-5 gives the best generation numbers – performance saturates quickly and even degrades for larger beam width (e.g., 10).

An alternative, non-deterministic generation strategy is to randomly sample  $N$  captions from the model’s distribution and choose the most probable among these. Under this strategy we also examine the effect of applying various choices of scalar factors (inverse of the “temperature”)  $T$  to the real-valued predictions input to the softmax producing the distribution. For larger values of  $T$  the samples are greedier and less diverse, with  $T = \infty$  being equivalent to beam search with beam width 1. Larger values of  $N$  suggest using smaller values of  $T$ , and vice versa – for example, with large  $N$  and large  $T$ , most of the  $\mathcal{O}(N)$  computation is wasted as many of the samples will be redundant. We assess saturation as the number of samples  $N$  grows, and find that  $N = 100$  samples with  $T = 2$  improves little over  $N = 25$ . We also varied the temperature  $T$  among values 1, 1.5, and 2 (all with  $N = 100$ ) and found  $T = 1.5$  to perform the best.

We adopt the best-performing generation strategy from the bottom-most set of results in Table 4.6 (sampling with  $T = 1.5$ ,  $N = 100$ ) as the strategy for the middle set of results in the table, which ablates LRCN architectures. We also record generation performance for all architectures (Table 4.6, top set of results) with the

simpler generation strategy used in our earlier work (Donahue et al., 2015) for ease of comparison with this work and for future researchers. For the remainder of this discussion, we will focus on the middle set of results, and particularly on the CIDEr-D (Vedantam et al., 2015) (C) metric, as it was designed specifically for automatic evaluation of image captioning systems. We see again that the LSTM unit outperforms an RNN unit for generation, though not as significantly as for retrieval. Between the sequence model architecture choices (depicted in Figure 4.4) of the number of layers  $L$  and whether to factor, we see that in this case the two-layer models (LRCN<sub>2f</sub> and LRCN<sub>2u</sub>) perform similarly, outperforming the single layer model (LRCN<sub>1u</sub>). Interestingly, of the three variants, LRCN<sub>2f</sub> is the only one to perform best for both retrieval and generation.

We see again that fine-tuning (FT) the visual representation and using a stronger vision model (VGGNet (Simonyan and Zisserman, 2015)) improves results significantly. Fine-tuning improves CIDEr-D by roughly 0.04 points for CaffeNet, and by roughly 0.07 points for VGGNet. Switching from finetuned CaffeNet to VGGNet improves CIDEr-D by 0.13 points.

In Table 4.7 we compare generation performance with contemporaneous and recent work submitted to the 2015 COCO caption challenge using our best-performing method (under the CIDEr-D metric) from the results on the validation set described above – generating a caption for a single image by taking the best of  $N = 100$  samples with a scalar factor of  $T = 1.5$  applied to the softmax inputs, using an LRCN model which pairs a fine-tuned VGGNet with our LRCN<sub>2f</sub> (two layer, factored) sequence model architecture. Our results are competitive with the contemporary work, performing 4th best in CIDEr-D (0.934, compared with the best result of 0.946 from (Vinyals et al., 2015a)), and 3rd best in METEOR (0.335, compared with 0.346 from (Vinyals et al., 2015a)).

In addition to standard quantitative evaluations, we also employ Amazon Mechanical Turk workers (“Turkers”) to evaluate the generated sentences. Given an image and a set of descriptions from different models, we ask Turkers to rank the sentences based on correctness, grammar and relevance. We compared sentences from our model to the ones made publicly available by Kiros et al. (2015). As seen in Table 4.8, our fine-tuned (FT) LRCN model performs on par with the Nearest Neighbour (NN) on correctness and relevance, and better on grammar.

We show sample captions in Figure 4.6. We additionally note some properties of the captions our model generates. When using the VGG model to generate sentences in the validation set, we find that 33.7% of our generated sentences exactly match a sentence in the training set. Furthermore, we find that when using a beam size of one, our model generates 42% of the vocabulary words used by human annotators when describing images in the validation set. Some words, such as “lady” and “guy”, are not generated by our model but are commonly used by human annotators, but synonyms such as “woman” and “man” are two of the most common words generated by our model.

	Correctness	Grammar	Relevance
TreeTalk (Kuznetsova et al., 2014)	4.08	4.35	3.98
VGGNet (Kiros et al., 2015)	3.71	3.46	3.70
NN (Kiros et al., 2015)	<b>3.44</b>	3.20	<b>3.49</b>
LRCN $fc_8$ (ours)	3.74	3.19	3.72
LRCN FT (ours)	<b>3.47</b>	<b>3.01</b>	<b>3.50</b>
Captions	2.55	3.72	2.59

Table 4.8. Image description: Human evaluator rankings from 1-6 (low is good) averaged for each method and criterion. We evaluated on 785 Flickr images selected by the authors of Kiros et al. (2015) for the purposes of comparison against this similar contemporary approach.

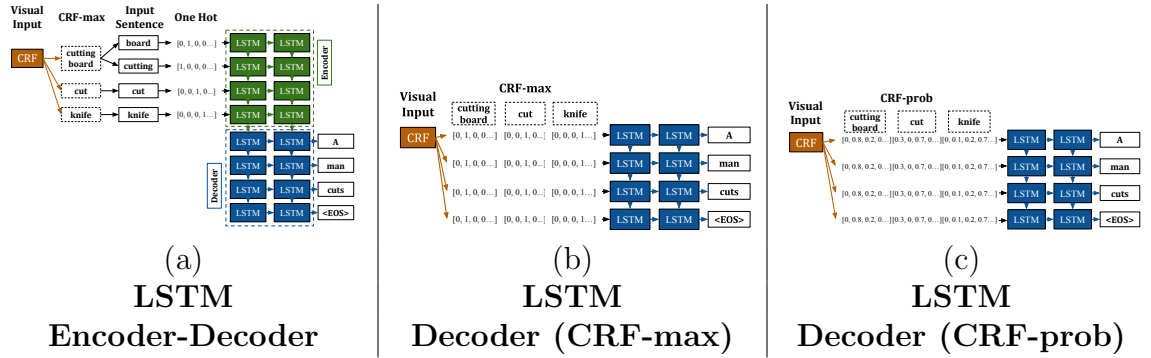


Figure 4.5. Our approaches to video description. (a) LSTM encoder & decoder with CRF max (b) LSTM decoder with CRF max (c) LSTM decoder with CRF probabilities.

## 4.5 Video description

In video description the LSTM framework allows us to model the video as a variable length input stream. However, due to the limitations of available video description datasets, we rely on more “traditional” activity and video recognition processing for the input and use LSTMs for generating a sentence. We first distinguish the following architectures for video description (see Figure 4.5). For each architecture, we assume we have predictions of activity, tool, object, and locations present in the video from a CRF based on the full video input. In this way, we observe the video as whole at each time step, not incrementally frame by frame.

(a) **LSTM encoder & decoder with CRF max.** (Figure 4.5(a)) This architecture is motivated by the video description approach presented in Rohrbach et al. (2013). They first recognize a semantic representation of the video using the

Architecture	Input	BLEU
SMT (Rohrbach et al., 2013)	CRF max	24.9
SMT (Rohrbach et al., 2014)	CRF prob	26.9
(a) LSTM Encoder-Decoder (ours)	CRF max	25.3
(b) LSTM Decoder (ours)	CRF max	27.4
(c) LSTM Decoder (ours)	CRF prob	<b>28.8</b>

Table 4.9. Video description: Results on detailed description of TACoS multi-level (Rohrbach et al., 2014), in %, see Section 4.5 for details.

maximum a posteriori (MAP) estimate of a CRF with video features as unaries. This representation, e.g.,  $\langle \text{knife, cut, carrot, cutting board} \rangle$ , is concatenated into an input sequence (*knife cut carrot cutting board*) which is translated to a natural language sentence (*a person cuts a carrot on the board*) using statistical machine translation (SMT) (Koehn et al., 2007). We replace SMT with an encoder-decoder LSTM, which encodes the input sequence as a fixed length vector before decoding to a sentence.

**(b) LSTM decoder with CRF max.** (Figure 4.5(b)) In this variant we provide the full visual input representation at each time step to the LSTM, analogous to how an image is provided as an input to the LSTM in image captioning.

**(c) LSTM decoder with CRF probabilities.** (Figure 4.5(c)) A benefit of using LSTMs for machine translation compared to phrase-based SMT (Koehn et al., 2007) is that it can naturally incorporate probability vectors during training *and* test time which allows the LSTM to learn uncertainties in visual generation rather than relying on MAP estimates. The architecture is the the same as in (b), but we replace max predictions with probability distributions.

### 4.5.1 Evaluation

We evaluate our approach on the TACoS multilevel (Rohrbach et al., 2014) dataset, which has 44,762 video/sentence pairs (about 40,000 for training/validation). We compare to Rohrbach et al. (2013) who use max prediction as well as a variant presented in Rohrbach et al. (2014) which takes CRF probabilities at test time and uses a word lattice to find an optimal sentence prediction. Since we use the max prediction as well as the probability scores provided by Rohrbach et al. (2014), we have an identical visual representation. Rohrbach et al. (2014) uses dense trajectories (Wang et al., 2013b) and SIFT features as well as temporal context reasoning modeled in a CRF. In this set of experiments we use the two-layered, unfactored version of LRCN, as described for image description.

Table 4.9 shows the BLEU-4 score. The results show that (1) the LSTM outperforms an SMT-based approach to video description; (2) the simpler decoder architecture (b) and (c) achieve better performance than (a), likely because the input does not



need to be memorized; and (3) our approach achieves 28.8%, clearly outperforming the best reported number of 26.9% on TACoS multilevel by Rohrbach et al. (2014).

More broadly, these results show that our architecture is not restricted only to input from deep networks, but can be cleanly integrated with fixed or variable length inputs from other vision systems.

## 4.6 Related work

We present previous literature pertaining to the three tasks discussed in this work. Additionally, we discuss subsequent extensions which combine convolutional and recurrent networks to achieve improved results on activity recognition, image captioning, and video description as well as related new tasks such as visual question answering.

### 4.6.1 Prior work

**Activity recognition.** State-of-the-art shallow models combine spatio-temporal features along dense trajectories (Wang and Schmid, 2013) and encode features as bags of words or Fisher vectors for classification. Such shallow features track how low level features change through time but cannot track higher level features. Furthermore, by encoding features as bags of words or Fisher vectors, temporal relationships are lost.

Many deep architectures proposed for activity recognition stack a fixed number of video frames for input to a deep network. Karpathy et al. (2014a) propose a fusion convolutional network which fuses layers which correspond to different input frames at various levels of a deep network. Simonyan and Zisserman (2014) proposes a two stream CNN which combines one CNN trained on RGB frames and one CNN trained on a stack of 10 flow frames. When combining RGB and flow by averaging softmax scores, results are comparable to state-of-the-art shallow models on UCF101 (Soomro et al., 2012) and HMDB51 (Kuehne et al., 2011). Results are further improved by using an SVM to fuse RGB and flow as opposed to simply averaging scores. Alternatively, Ji et al. (2013) and Baccouche et al. (2011) propose learning deep spatio-temporal features with 3D convolutional neural networks. Baccouche et al. (2011) and Baccouche et al. (2010) propose extracting visual and motion features and modeling temporal dependencies with recurrent networks. This architecture most closely resembles our proposed architecture for activity classification, though it differs in two key ways. First, we integrate 2D CNNs that can be pre-trained on large image datasets. Second, we combine the CNN and LSTM into a single model to enable end-to-end fine-tuning.

**Image captioning.** Several early works (Farhadi et al., 2010; Kulkarni et al., 2011; Mitchell et al., 2012; Yang et al., 2011) on image captioning combine object and

scene recognition with template or tree based approaches to generate captions. Such sentences are typically simple and are easily distinguished from more fluent human generated descriptions. Kuznetsova et al. (2012, 2014) address this by composing new sentences from existing caption fragments which, though more human like, are not necessarily accurate or correct.

More recently, a variety of deep and multi-modal models (Frome et al., 2013; Kiros et al., 2014; Mao et al., 2015a; Socher et al., 2014) have been proposed for image and caption retrieval, as well as caption generation. Though some of these models rely on deep convolutional nets for image feature extraction (Frome et al., 2013; Kiros et al., 2014), recently researchers have realized the importance of also including temporally deep networks to model text. Socher et al. (2014) propose an RNN to map sentences into a multi-modal embedding space. By mapping images and language into the same embedding space, they are able to compare images and descriptions for image and annotation retrieval tasks. Mao et al. (2015a) propose a model for caption generation that is more similar to the model proposed in this work: predictions for the next word are based on previous words in a sentence and image features. Kiros et al. (2014) propose an encoder-decoder model for image caption retrieval which relies on both a CNN and LSTM encoder to learn an embedding of image-caption pairs. Their model uses a neural language decoder to enable sentence generation. As evidenced by the rapid growth of image captioning, visual sequence models like LRCN are increasingly important for describing the visual world using natural language.

**Video description.** Recent approaches to describing video with natural language have made use of templates, retrieval, or language models (Barbu et al., 2012; Das et al., 2013; Guadarrama et al., 2013; Khan et al., 2011,?; Rohrbach et al., 2013; Tan et al., 2011; Thomason et al., 2014). To our knowledge, we present the first application of deep models to the video description task. Most similar to our work is Rohrbach et al. (2013), which use phrase-based SMT (Koehn et al., 2007) to generate a sentence. In Section 4.5 we show that phrase-based SMT can be replaced with LSTMs for video description as has been shown previously for language translation (Sak et al., 2014; Sutskever et al., 2014).

### 4.6.2 Contemporaneous and subsequent work

Similar work in activity recognition and visual description was conducted contemporaneously with our work, and a variety of subsequent work has combined convolutional and recurrent networks to both improve upon our results and achieve exciting results on other sequential visual tasks.

**Activity recognition.** Contemporaneous with our work, Ng et al. (2015) train a network which combines CNNs and LSTMs for activity recognition. Because

activity recognition datasets like UCF101 are relatively small in comparison to image recognition datasets, Ng et al. (2015) pretrain their network using the Sports-1M (Karpathy et al., 2014a) dataset which includes over a million videos mined from YouTube. By training a much larger network (four stacked LSTMs) and pretraining on a large video dataset, Ng et al. (2015) achieve 88.6% on the UCF101 dataset.

Yeung et al. (2015) also combines a convolutional network with an LSTM to predict multiple activities per frame. Unlike LRCN, Yeung et al. (2015) focuses on frame-level (rather than video-level) predictions, which allows their system to label multiple activities that occur in different temporal locations of a video clip. Like we show for activity recognition, Yeung et al. (2015) demonstrates that including temporal information improves upon a single frame baseline. Additionally, Yeung et al. (2015) employ an attention mechanism to further improve results.

**Image captioning.** Karpathy and Li (2015) and Vinyals et al. (2015a) also propose models which combine a CNN with a recurrent network for image captioning. Though similar to LRCN, the architectures proposed in Karpathy and Li (2015) and Vinyals et al. (2015a) differ in how image features are input into the sequence model. In contrast to our system, in which image features are input at each time step, Karpathy and Li (2015) and Vinyals et al. (2015a) only input image features at the first time step. Furthermore, they do not explore a “factored” representation (Figure 4.4). Subsequent work (Xu et al., 2015) has proposed attention to focus on which portion of the image is observed during sequence generation. By including attention, Xu et al. (2015) aim to visually focus on the current word generated by the model. Other works aim to address specific limitations of captioning models based on combining convolutional and recurrent architectures. For example, methods have been proposed to integrate new vocabulary with limited (Mao et al., 2015b) or no (Hendricks et al., 2016) examples of images and corresponding captions.

**Video description.** In this work, we rely on intermediate features for video description, but end-to-end trainable models for visual captioning have since been proposed. Venugopalan et al. (2015a) propose creating a video feature by pooling high level CNN features across frames. The video feature is then used to generate descriptions in the same way an image is used to generate a description in LRCN. Though achieving good results, by pooling CNN features, temporal information from the video is lost. Consequently, Venugopalan et al. (2015b) propose an LSTM to encode video frames into a fixed length vector before sentence generation with an LSTM. Using an end-to-end trainable “sequence-to-sequence” model which can exploit temporal structure in video, Venugopalan et al. (2015b) improve upon results for video description. Yao et al. (2015) propose a similar model, adding a temporal attention mechanism which weights video frames differently when generating each word in a sentence.

**Visual grounding.** Rohrbach et al. (2015) combine CNNs with LSTMs for visual grounding. The model first encodes a phrase which describes part of an image using an LSTM, then learns to attend to the appropriate location in the image to accurately reconstruct the phrase. In order to reconstruct the phrase, the model must learn to visually ground the input phrase to the appropriate location in the image.

**Natural language object retrieval.** In this work, we present methods for image retrieval based on a natural language description. In contrast, Hu et al. (2016) use a model based on LRCN for *object retrieval*, which returns the bounding box around a given object as opposed to an entire image. In order to adapt LRCN to the task of object retrieval, Hu et al. (2016) include local convolutional features which are extracted from object proposals and the spatial configuration of object proposals in addition to a global image feature. By including local features, Hu et al. (2016) effectively adapt LRCN for object retrieval.

## 4.7 Discussion

We’ve presented LRCN, a class of models that is both spatially and temporally deep, and flexible enough to be applied to a variety of vision tasks involving sequential inputs and outputs. Our results consistently demonstrate that by learning sequential dynamics with a deep sequence model, we can improve upon previous methods which learn a deep hierarchy of parameters only in the visual domain, and on methods which take a fixed visual representation of the input and only learn the dynamics of the output sequence.

As the field of computer vision matures beyond tasks with static input and predictions, deep sequence modeling tools like LRCN are increasingly central to vision systems for problems with sequential structure. The ease with which these tools can be incorporated into existing visual recognition pipelines makes them a natural choice for perceptual problems with time-varying visual input or sequential outputs, which these methods are able to handle with little input preprocessing and no hand-designed features.



A female tennis player in action on the court.



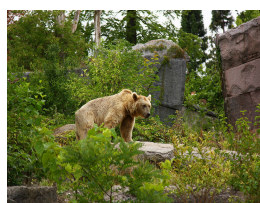
A group of young men playing a game of soccer



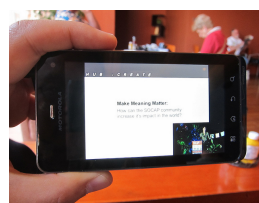
A man riding a wave on top of a surfboard.



A baseball game in progress with the batter up to plate.



A brown bear standing on top of a lush green field.



A person holding a cell phone in their hand.



A close up of a person brushing his teeth.



A woman laying on a bed in a bedroom.



A black and white cat is sitting on a chair.



A large clock mounted to the side of a building.



A bunch of fruit that are sitting on a table.



A toothbrush holder sitting on top of a white sink.

Figure 4.6. Image description: images with corresponding captions generated by our finetuned LRCN model. These are images 1-12 of our randomly chosen validation set from COCO 2014 (Lin et al., 2014). We used beam search with a beam size of 5 to generate the sentences.

## Chapter 5

# Unsupervised Representation Learning with Bidirectional Generative Adversarial Networks<sup>1</sup>

Deep convolutional networks (convnets) have become a staple of the modern computer vision pipeline. After training these models on a massive database of image-label pairs like ImageNet (Russakovsky et al., 2015), the network easily adapts to a variety of similar visual tasks, achieving impressive results on image classification (Donahue et al., 2014; Razavian et al., 2014; Zeiler and Fergus, 2014) or localization (Girshick et al., 2014; Long et al., 2015) tasks. In other perceptual domains such as natural language processing or speech recognition, deep networks have proven highly effective as well (Bahdanau et al., 2015; Graves et al., 2013; Sutskever et al., 2014; Vinyals et al., 2015b). However, all of these recent results rely on a supervisory signal from large-scale databases of hand-labeled data, ignoring much of the useful information present in the structure of the data itself.

Meanwhile, Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have emerged as a powerful framework for learning generative models of arbitrarily complex data distributions. The GAN framework learns a *generator* mapping samples from an arbitrary latent distribution to data, as well as an adversarial *discriminator* which tries to distinguish between real and generated samples as accurately as possible. The generator’s goal is to “fool” the discriminator by producing samples which are as close to real data as possible. When trained on databases of natural images, GANs produce impressive results (Denton et al., 2015; Radford et al., 2016).

Interpolations in the latent space of the generator produce smooth and plausible semantic variations, and certain directions in this space correspond to particular semantic attributes along which the data distribution varies. For example, Radford

---

<sup>1</sup>This chapter is based on joint work with Philipp Krähenbühl and Trevor Darrell (Donahue et al., 2017), presented primarily as it appears in the ICLR 2017 proceedings.

et al. (2016) showed that a GAN trained on a database of human faces learns to associate particular latent directions with gender and the presence of eyeglasses.

A natural question arises from this ostensible “semantic juice” flowing through the weights of generators learned using the GAN framework: can GANs be used for unsupervised learning of rich feature representations for arbitrary data distributions? An obvious issue with doing so is that the generator maps latent samples to generated data, but the framework does not include an *inverse* mapping from data to latent representation.

Hence, we propose a novel unsupervised feature learning framework, *Bidirectional Generative Adversarial Networks* (BiGAN). The overall model is depicted in Figure 5.1. In short, in addition to the generator  $G$  from the standard GAN framework (Goodfellow et al., 2014), BiGAN includes an *encoder*  $E$  which maps data  $\mathbf{x}$  to latent representations  $\mathbf{z}$ . The BiGAN discriminator  $D$  discriminates not only in data space ( $\mathbf{x}$  versus  $G(\mathbf{z})$ ), but jointly in data and latent space (tuples  $(\mathbf{x}, E(\mathbf{x}))$  versus  $(G(\mathbf{z}), \mathbf{z})$ ), where the latent component is either an encoder output  $E(\mathbf{x})$  or a generator input  $\mathbf{z}$ .

It may not be obvious from this description that the BiGAN encoder  $E$  should learn to invert the generator  $G$ . The two modules cannot directly “communicate” with one another: the encoder never “sees” generator outputs ( $E(G(\mathbf{z}))$  is not computed), and vice versa. Yet, in Section 5.2, we will both argue intuitively and formally prove that the encoder and generator must learn to invert one another in order to fool the BiGAN discriminator.

Because the BiGAN encoder learns to predict features  $\mathbf{z}$  given data  $\mathbf{x}$ , and prior work on GANs has demonstrated that these features capture semantic attributes of the data, we hypothesize that a trained BiGAN encoder may serve as a useful feature representation for related semantic tasks, in the same way that fully supervised visual models trained to predict semantic “labels” given images serve as powerful feature representations for related visual tasks. In this context, a latent representation  $\mathbf{z}$  may be thought of as a “label” for  $\mathbf{x}$ , but one which came for “free,” without the need for supervision.

An alternative approach to learning the inverse mapping from data to latent representation is to directly model  $p(\mathbf{z}|G(\mathbf{z}))$ , predicting generator input  $\mathbf{z}$  given generated data  $G(\mathbf{z})$ . We’ll refer to this alternative as a *latent regressor*, later arguing (Section 5.3.1) that the BiGAN encoder may be preferable in a feature learning context, as well as comparing the approaches empirically.

BiGANs are a robust and highly generic approach to unsupervised feature learning, making no assumptions about the structure or type of data to which they are applied, as our theoretical results will demonstrate. Our empirical studies will show that despite their generality, BiGANs are competitive with contemporary approaches to self-supervised and weakly supervised feature learning designed specifically for a notoriously complex data distribution – natural images.

Dumoulin et al. (2016) independently proposed an identical model in their concurrent work, exploring the case of a stochastic encoder  $E$  and the ability of such

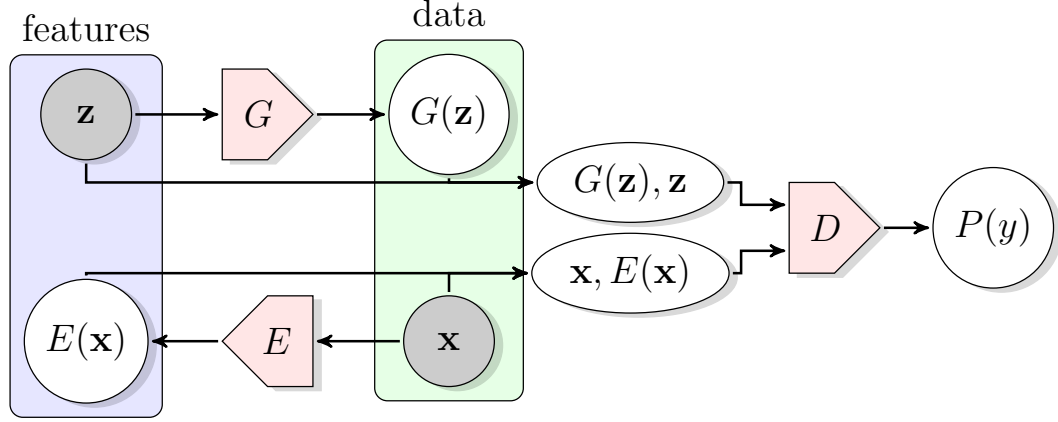


Figure 5.1. The structure of Bidirectional Generative Adversarial Networks (BiGAN).

models to learn in a semi-supervised setting.

## 5.1 Background

Let  $p_{\mathbf{x}}(\mathbf{x})$  be the distribution of our data for  $\mathbf{x} \in \Omega_{\mathbf{x}}$  (e.g. natural images). The goal of generative modeling is capture this data distribution using a probabilistic model. Unfortunately, exact modeling of this probability density function is computationally intractable (Hinton et al., 2006; Salakhutdinov and Hinton, 2009) for all but the most trivial models. Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) instead model the data distribution as a transformation of a fixed latent distribution  $p_{\mathbf{z}}(\mathbf{z})$  for  $\mathbf{z} \in \Omega_{\mathbf{z}}$ . This transformation, called a *generator*, is expressed as a deterministic feed forward network  $G : \Omega_{\mathbf{z}} \rightarrow \Omega_{\mathbf{x}}$  with  $p_G(\mathbf{x}|\mathbf{z}) = \delta(\mathbf{x} - G(\mathbf{z}))$  and  $p_G(\mathbf{x}) = \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [p_G(\mathbf{x}|\mathbf{z})]$ . The goal is to train a generator such that  $p_G(\mathbf{x}) \approx p_{\mathbf{x}}(\mathbf{x})$ .

The GAN framework trains a generator, such that no discriminative model  $D : \Omega_{\mathbf{x}} \mapsto [0, 1]$  can distinguish samples of the data distribution from samples of the generative distribution. Both generator and discriminator are learned using the adversarial (minimax) objective  $\min_G \max_D V(D, G)$ , where

$$V(D, G) := \mathbb{E}_{\mathbf{x} \sim p_{\mathbf{x}}} [\log D(\mathbf{x})] + \underbrace{\mathbb{E}_{\mathbf{x} \sim p_G} [\log (1 - D(\mathbf{x}))]}_{\mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log (1 - D(G(\mathbf{z})))]} \quad (5.1)$$

Goodfellow et al. (2014) showed that for an ideal discriminator the objective  $C(G) := \max_D V(D, G)$  is equivalent to the Jensen-Shannon divergence between the two distributions  $p_G$  and  $p_{\mathbf{x}}$ .

The adversarial objective 5.1 does not directly lend itself to an efficient optimization, as each step in the generator  $G$  requires a full discriminator  $D$  to be learned. Furthermore, a perfect discriminator no longer provides any gradient information



to the generator, as the gradient of any global or local maximum of  $V(D, G)$  is 0. To provide a strong gradient signal nonetheless, Goodfellow et al. (2014) slightly alter the objective between generator and discriminator updates, while keeping the same fixed point characteristics. They also propose to optimize (5.1) using an alternating optimization switching between updates to the generator and discriminator. While this optimization is not guaranteed to converge, empirically it works well if the discriminator and generator are well balanced.

Despite the empirical strength of GANs as generative models of arbitrary data distributions, it is not clear how they can be applied as an unsupervised feature representation. One possibility for learning such representations is to learn an inverse mapping regressing from generated data  $G(\mathbf{z})$  back to the latent input  $\mathbf{z}$ . However, unless the generator perfectly models the data distribution  $p_{\mathbf{x}}$ , a nearly impossible objective for a complex data distribution such as that of high-resolution natural images, this idea may prove insufficient.

## 5.2 Bidirectional Generative Adversarial Networks

In Bidirectional Generative Adversarial Networks (BiGANs) we not only train a generator, but additionally train an encoder  $E : \Omega_{\mathbf{x}} \rightarrow \Omega_{\mathbf{z}}$ . The encoder induces a distribution  $p_E(\mathbf{z}|\mathbf{x}) = \delta(\mathbf{z} - E(\mathbf{x}))$  mapping data points  $\mathbf{x}$  into the latent feature space of the generative model. The discriminator is also modified to take input from the latent space, predicting  $P_D(Y|\mathbf{x}, \mathbf{z})$ , where  $Y = 1$  if  $\mathbf{x}$  is real (sampled from the real data distribution  $p_{\mathbf{x}}$ ), and  $Y = 0$  if  $\mathbf{x}$  is generated (the output of  $G(\mathbf{z})$ ,  $\mathbf{z} \sim p_{\mathbf{z}}$ ).

The BiGAN training objective is defined as a minimax objective

$$\min_{G, E} \max_D V(D, E, G) \quad (5.2)$$

where

$$V(D, E, G) := \mathbb{E}_{\mathbf{x} \sim p_{\mathbf{x}}} \left[ \underbrace{\mathbb{E}_{\mathbf{z} \sim p_E(\cdot|\mathbf{x})} [\log D(\mathbf{x}, \mathbf{z})]}_{\log D(\mathbf{x}, E(\mathbf{x}))} \right] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \left[ \underbrace{\mathbb{E}_{\mathbf{x} \sim p_G(\cdot|\mathbf{z})} [\log (1 - D(\mathbf{x}, \mathbf{z}))]}_{\log(1 - D(G(\mathbf{z}), \mathbf{z}))} \right]. \quad (5.3)$$

We optimize this minimax objective using the same alternating gradient based optimization as Goodfellow et al. (2014). See Section 5.2.4 for details.

BiGANs share many of the theoretical properties of GANs (Goodfellow et al., 2014), while additionally guaranteeing that at the global optimum,  $G$  and  $E$  are each other's inverse. BiGANs are also closely related to autoencoders with an  $\ell_0$  loss function. In the following sections we highlight some of the appealing theoretical properties of BiGANs.

**Definitions.** Let  $p_{G\mathbf{Z}}(\mathbf{x}, \mathbf{z}) := p_G(\mathbf{x}|\mathbf{z})p_{\mathbf{Z}}(\mathbf{z})$  and  $p_{E\mathbf{X}}(\mathbf{x}, \mathbf{z}) := p_E(\mathbf{z}|\mathbf{x})p_{\mathbf{X}}(\mathbf{x})$  be the joint distributions modeled by the generator and encoder respectively.  $\Omega := \Omega_{\mathbf{X}} \times \Omega_{\mathbf{Z}}$  is the joint latent and data space. For a region  $R \subseteq \Omega$ ,

$$\begin{aligned} P_{E\mathbf{X}}(R) &:= \int_{\Omega} p_{E\mathbf{X}}(\mathbf{x}, \mathbf{z}) \mathbf{1}_{[(\mathbf{x}, \mathbf{z}) \in R]} d(\mathbf{x}, \mathbf{z}) = \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \int_{\Omega_{\mathbf{Z}}} p_E(\mathbf{z}|\mathbf{x}) \mathbf{1}_{[(\mathbf{x}, \mathbf{z}) \in R]} d\mathbf{z} d\mathbf{x} \\ P_{G\mathbf{Z}}(R) &:= \int_{\Omega} p_{G\mathbf{Z}}(\mathbf{x}, \mathbf{z}) \mathbf{1}_{[(\mathbf{x}, \mathbf{z}) \in R]} d(\mathbf{x}, \mathbf{z}) = \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \int_{\Omega_{\mathbf{X}}} p_G(\mathbf{x}|\mathbf{z}) \mathbf{1}_{[(\mathbf{x}, \mathbf{z}) \in R]} d\mathbf{x} d\mathbf{z} \end{aligned}$$

are probability measures over that region. We also define

$$P_{\mathbf{X}}(R_{\mathbf{X}}) := \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \mathbf{1}_{[\mathbf{x} \in R_{\mathbf{X}}]} d\mathbf{x} \quad P_{\mathbf{Z}}(R_{\mathbf{Z}}) := \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \mathbf{1}_{[\mathbf{z} \in R_{\mathbf{Z}}]} d\mathbf{z}$$

as measures over regions  $R_{\mathbf{X}} \subseteq \Omega_{\mathbf{X}}$  and  $R_{\mathbf{Z}} \subseteq \Omega_{\mathbf{Z}}$ . We refer to the set of features and data samples in the support of  $P_{\mathbf{X}}$  and  $P_{\mathbf{Z}}$  as  $\hat{\Omega}_{\mathbf{X}} := \text{supp}(P_{\mathbf{X}})$  and  $\hat{\Omega}_{\mathbf{Z}} := \text{supp}(P_{\mathbf{Z}})$  respectively.  $D_{\text{KL}}(P \parallel Q)$  and  $D_{\text{JS}}(P \parallel Q)$  respectively denote the Kullback-Leibler (KL) and Jensen-Shannon divergences between probability measures  $P$  and  $Q$ . By definition,

$$\begin{aligned} D_{\text{KL}}(P \parallel Q) &:= \mathbb{E}_{\mathbf{x} \sim P} [\log f_{PQ}(\mathbf{x})] \\ D_{\text{JS}}(P \parallel Q) &:= \frac{1}{2} (D_{\text{KL}}(P \parallel \frac{P+Q}{2}) + D_{\text{KL}}(Q \parallel \frac{P+Q}{2})), \end{aligned}$$

where  $f_{PQ} := \frac{dP}{dQ}$  is the Radon-Nikodym (RN) derivative of measure  $P$  with respect to measure  $Q$ , with the defining property that  $P(R) = \int_R f_{PQ} dQ$ . The RN derivative  $f_{PQ} : \Omega \mapsto \mathbb{R}_{\geq 0}$  is defined for any measures  $P$  and  $Q$  on space  $\Omega$  such that  $P$  is absolutely continuous with respect to  $Q$ : i.e., for any  $R \subseteq \Omega$ ,  $P(R) > 0 \implies Q(R) > 0$ .

### 5.2.1 Optimal discriminator, generator, & encoder

We start by characterizing the optimal discriminator for any generator and encoder, following Goodfellow et al. (2014). This optimal discriminator then allows us to reformulate objective (5.3), and show that it reduces to the Jensen-Shannon divergence between the joint distributions  $P_{E\mathbf{X}}$  and  $P_{G\mathbf{Z}}$ .

**Proposition 1** *For any  $E$  and  $G$ , the optimal discriminator  $D_{EG}^* := \arg\max_D V(D, E, G)$  is the Radon-Nikodym derivative  $f_{EG} := \frac{dP_{E\mathbf{X}}}{d(P_{E\mathbf{X}} + P_{G\mathbf{Z}})} : \Omega \mapsto [0, 1]$  of measure  $P_{E\mathbf{X}}$  with respect to measure  $P_{E\mathbf{X}} + P_{G\mathbf{Z}}$ .*

*Proof.* Given in Appendix 5.5.1.1.

This optimal discriminator now allows us to characterize the optimal generator and encoder.

**Proposition 2** *The encoder and generator's objective for an optimal discriminator  $C(E, G) := \max_D V(D, E, G) = V(D_{EG}^*, E, G)$  can be rewritten in terms of the Jensen-Shannon divergence between measures  $P_{E\mathbf{X}}$  and  $P_{G\mathbf{Z}}$  as  $C(E, G) = 2 D_{\text{JS}}(P_{E\mathbf{X}} \parallel P_{G\mathbf{Z}}) - \log 4$ .*

*Proof.* Given in Appendix 5.5.1.2.

**Theorem 1** *The global minimum of  $C(E, G)$  is achieved if and only if  $P_{E\mathbf{X}} = P_{G\mathbf{Z}}$ . At that point,  $C(E, G) = -\log 4$  and  $D_{EG}^* = \frac{1}{2}$ .*

*Proof.* From Proposition 2, we have that  $C(E, G) = 2 D_{\text{JS}}(P_{E\mathbf{X}} \parallel P_{G\mathbf{Z}}) - \log 4$ . The Jensen-Shannon divergence  $D_{\text{JS}}(P \parallel Q) \geq 0$  for any  $P$  and  $Q$ , and  $D_{\text{JS}}(P \parallel Q) = 0$  if and only if  $P = Q$ . Therefore, the global minimum of  $C(E, G)$  occurs if and only if  $P_{E\mathbf{X}} = P_{G\mathbf{Z}}$ , and at this point the value is  $C(E, G) = -\log 4$ . Finally,  $P_{E\mathbf{X}} = P_{G\mathbf{Z}}$  implies that the optimal discriminator is chance:  $D_{EG}^* = \frac{dP_{E\mathbf{X}}}{d(P_{E\mathbf{X}} + P_{G\mathbf{Z}})} = \frac{dP_{E\mathbf{X}}}{2dP_{E\mathbf{X}}} = \frac{1}{2}$ .  $\square$

The optimal discriminator, encoder, and generator of BiGAN are similar to the optimal discriminator and generator of the GAN framework (Goodfellow et al., 2014). However, an important difference is that BiGAN optimizes a Jensen-Shannon divergence between a joint distribution over both data  $\mathbf{X}$  and latent features  $\mathbf{Z}$ . This joint divergence allows us to further characterize properties of  $G$  and  $E$ , as shown below.

### 5.2.2 Optimal generator & encoder are inverses

We first present an intuitive argument that, in order to “fool” a perfect discriminator, a deterministic BiGAN encoder and generator must invert each other. (Later we will formally state and prove this property.) Consider a BiGAN discriminator input pair  $(\mathbf{x}, \mathbf{z})$ . Due to the sampling procedure,  $(\mathbf{x}, \mathbf{z})$  must satisfy at least one of the following two properties:

$$(a) \mathbf{x} \in \hat{\Omega}_{\mathbf{X}} \wedge E(\mathbf{x}) = \mathbf{z} \qquad (b) \mathbf{z} \in \hat{\Omega}_{\mathbf{Z}} \wedge G(\mathbf{z}) = \mathbf{x}$$

If *only* one of these properties is satisfied, a perfect discriminator can infer the source of  $(\mathbf{x}, \mathbf{z})$  with certainty: if only (a) is satisfied,  $(\mathbf{x}, \mathbf{z})$  must be an encoder pair  $(\mathbf{x}, E(\mathbf{x}))$  and  $D_{EG}^*(\mathbf{x}, \mathbf{z}) = 1$ ; if only (b) is satisfied,  $(\mathbf{x}, \mathbf{z})$  must be a generator pair  $(G(\mathbf{z}), \mathbf{z})$  and  $D_{EG}^*(\mathbf{x}, \mathbf{z}) = 0$ .

Therefore, in order to fool a perfect discriminator at  $(\mathbf{x}, \mathbf{z})$  (so that  $0 < D_{EG}^*(\mathbf{x}, \mathbf{z}) < 1$ ),  $E$  and  $G$  must satisfy *both* (a) and (b). In this case, we can substitute the equality  $E(\mathbf{x}) = \mathbf{z}$  required by (a) into the equality  $G(\mathbf{z}) = \mathbf{x}$  required by (b), and vice versa, giving the inversion properties  $\mathbf{x} = G(E(\mathbf{x}))$  and  $\mathbf{z} = E(G(\mathbf{z}))$ .

Formally, we show in Theorem 2 that the optimal generator and encoder invert one another almost everywhere on the support  $\hat{\Omega}_{\mathbf{X}}$  and  $\hat{\Omega}_{\mathbf{Z}}$  of  $P_{\mathbf{X}}$  and  $P_{\mathbf{Z}}$ .

**Theorem 2** *If  $E$  and  $G$  are an optimal encoder and generator, then  $E = G^{-1}$  almost everywhere; that is,  $G(E(\mathbf{x})) = \mathbf{x}$  for  $P_{\mathbf{X}}$ -almost every  $\mathbf{x} \in \Omega_{\mathbf{X}}$ , and  $E(G(\mathbf{z})) = \mathbf{z}$  for  $P_{\mathbf{Z}}$ -almost every  $\mathbf{z} \in \Omega_{\mathbf{Z}}$ .*

*Proof.* Given in Appendix 5.5.1.4.

While Theorem 2 characterizes the encoder and decoder at their optimum, due to the non-convex nature of the optimization, this optimum might never be reached. Experimentally, Section 5.3 shows that on standard datasets, the two are approximate inverses; however, they are rarely exact inverses. It is thus also interesting to show what objective BiGAN optimizes in terms of  $E$  and  $G$ . Next we show that BiGANs are closely related to autoencoders with an  $\ell_0$  loss function.

### 5.2.3 Relationship to autoencoders

As argued in Section 5, a model trained to predict features  $\mathbf{z}$  given data  $\mathbf{x}$  should learn useful semantic representations. Here we show that the BiGAN objective forces the encoder  $E$  to do exactly this: in order to fool the discriminator at a particular  $\mathbf{z}$ , the encoder must invert the generator at that  $\mathbf{z}$ , such that  $E(G(\mathbf{z})) = \mathbf{z}$ .

**Theorem 3** *The encoder and generator objective given an optimal discriminator  $C(E, G) := \max_D V(D, E, G)$  can be rewritten as an  $\ell_0$  autoencoder loss function*

$$C(E, G) = \mathbb{E}_{\mathbf{x} \sim p_{\mathbf{x}}} \left[ \mathbf{1}_{[E(\mathbf{x}) \in \hat{\Omega}_{\mathbf{z}} \wedge G(E(\mathbf{x})) = \mathbf{x}]} \log f_{EG}(\mathbf{x}, E(\mathbf{x})) \right] + \\ \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \left[ \mathbf{1}_{[G(\mathbf{z}) \in \hat{\Omega}_{\mathbf{x}} \wedge E(G(\mathbf{z})) = \mathbf{z}]} \log (1 - f_{EG}(G(\mathbf{z}), \mathbf{z})) \right]$$

with  $\log f_{EG} \in (-\infty, 0)$  and  $\log (1 - f_{EG}) \in (-\infty, 0)$   $P_{\mathbf{E}\mathbf{X}}$ -almost and  $P_{G\mathbf{Z}}$ -almost everywhere.

*Proof.* Given in Appendix 5.5.1.5.

Here the indicator function  $\mathbf{1}_{[G(E(\mathbf{x})) = \mathbf{x}]}$  in the first term is equivalent to an autoencoder with  $\ell_0$  loss, while the indicator  $\mathbf{1}_{[E(G(\mathbf{z})) = \mathbf{z}]}$  in the second term shows that the BiGAN encoder must invert the generator, the desired property for feature learning. The objective further encourages the functions  $E(\mathbf{x})$  and  $G(\mathbf{z})$  to produce valid outputs in the support of  $P_{\mathbf{Z}}$  and  $P_{\mathbf{X}}$  respectively. Unlike regular autoencoders, the  $\ell_0$  loss function does not make any assumptions about the structure or distribution of the data itself; in fact, all the structural properties of BiGAN are learned as part of the discriminator.

### 5.2.4 Learning

In practice, as in the GAN framework (Goodfellow et al., 2014), each BiGAN module  $D$ ,  $G$ , and  $E$  is a parametric function (with parameters  $\theta_D$ ,  $\theta_G$ , and  $\theta_E$ , respectively). As a whole, BiGAN can be optimized using alternating stochastic gradient steps. In one iteration, the discriminator parameters  $\theta_D$  are updated by taking one or more steps in the positive gradient direction  $\nabla_{\theta_D} V(D, E, G)$ , then the encoder parameters  $\theta_E$  and generator parameters  $\theta_G$  are together updated by

taking a step in the negative gradient direction  $-\nabla_{\theta_E, \theta_G} V(D, E, G)$ . In both cases, the expectation terms of  $V(D, E, G)$  are estimated using mini-batches of  $n$  samples  $\{\mathbf{x}^{(i)} \sim p_{\mathbf{X}}\}_{i=1}^n$  and  $\{\mathbf{z}^{(i)} \sim p_{\mathbf{Z}}\}_{i=1}^n$  drawn independently for each update step.

Goodfellow et al. (2014) found that an objective in which the real and generated labels  $Y$  are swapped provides stronger gradient signal to  $G$ . We similarly observed in BiGAN training that an “inverse” objective provides stronger gradient signal to  $G$  and  $E$ . For efficiency, we also update all modules  $D$ ,  $G$ , and  $E$  simultaneously at each iteration, rather than alternating between  $D$  updates and  $G$ ,  $E$  updates. See Appendix 5.5.2 for details.

### 5.2.5 Generalized BiGAN

It is often useful to parametrize the output of the generator  $G$  and encoder  $E$  in a different, usually smaller, space  $\Omega'_{\mathbf{X}}$  and  $\Omega'_{\mathbf{Z}}$  rather than the original  $\Omega_{\mathbf{X}}$  and  $\Omega_{\mathbf{Z}}$ . For example, for visual feature learning, the images input to the encoder should be of similar resolution to images used in the evaluation. On the other hand, generating high resolution images remains difficult for current generative models. In this situation, the encoder may take higher resolution input while the generator output and discriminator input remain low resolution.

We generalize the BiGAN objective  $V(D, G, E)$  (5.3) with functions  $g_{\mathbf{X}} : \Omega_{\mathbf{X}} \mapsto \Omega'_{\mathbf{X}}$  and  $g_{\mathbf{Z}} : \Omega_{\mathbf{Z}} \mapsto \Omega'_{\mathbf{Z}}$ , and encoder  $E : \Omega_{\mathbf{X}} \mapsto \Omega'_{\mathbf{Z}}$ , generator  $G : \Omega_{\mathbf{Z}} \mapsto \Omega'_{\mathbf{X}}$ , and discriminator  $D : \Omega'_{\mathbf{X}} \times \Omega'_{\mathbf{Z}} \mapsto [0, 1]$ :

$$\mathbb{E}_{\mathbf{x} \sim p_{\mathbf{X}}} \left[ \underbrace{\mathbb{E}_{\mathbf{z}' \sim p_E(\cdot|\mathbf{x})} [\log D(g_{\mathbf{X}}(\mathbf{x}), \mathbf{z}')] ]}_{\log D(g_{\mathbf{X}}(\mathbf{x}), E(\mathbf{x}))} \right] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{Z}}} \left[ \underbrace{\mathbb{E}_{\mathbf{x}' \sim p_G(\cdot|\mathbf{z})} [\log (1 - D(\mathbf{x}', g_{\mathbf{Z}}(\mathbf{z})))] ]}_{\log(1 - D(G(\mathbf{z}), g_{\mathbf{Z}}(\mathbf{z})))} \right]$$

An identity  $g_{\mathbf{X}}(\mathbf{x}) = \mathbf{x}$  and  $g_{\mathbf{Z}}(\mathbf{z}) = \mathbf{z}$  (and  $\Omega'_{\mathbf{X}} = \Omega_{\mathbf{X}}$ ,  $\Omega'_{\mathbf{Z}} = \Omega_{\mathbf{Z}}$ ) yields the original objective. For visual feature learning with higher resolution encoder inputs,  $g_{\mathbf{X}}$  is an image resizing function that downsamples a high resolution image  $\mathbf{x} \in \Omega_{\mathbf{X}}$  to a lower resolution image  $\mathbf{x}' \in \Omega'_{\mathbf{X}}$ , as output by the generator. ( $g_{\mathbf{Z}}$  is identity.)

In this case, the encoder and generator respectively induce probability measures  $P_{E\mathbf{X}'}$  and  $P_{G\mathbf{Z}'}$  over regions  $R \subseteq \Omega'$  of the joint space  $\Omega' := \Omega'_{\mathbf{X}} \times \Omega'_{\mathbf{Z}}$ , with  $P_{E\mathbf{X}'}(R) := \int_{\Omega_{\mathbf{X}}} \int_{\Omega'_{\mathbf{Z}}} p_{E\mathbf{X}}(\mathbf{x}, \mathbf{z}') \mathbf{1}_{[(\mathbf{x}', \mathbf{z}') \in R]} \delta(g_{\mathbf{X}}(\mathbf{x}) - \mathbf{x}') d\mathbf{z}' d\mathbf{x}' d\mathbf{x} = \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \mathbf{1}_{[(g_{\mathbf{X}}(\mathbf{x}), E(\mathbf{x})) \in R]} d\mathbf{x}$ , and  $P_{G\mathbf{Z}'}$  defined analogously. For optimal  $E$  and  $G$ , we can show  $P_{E\mathbf{X}'} = P_{G\mathbf{Z}'}$ : a generalization of Theorem 1. When  $E$  and  $G$  are deterministic and optimal, Theorem 2 – that  $E$  and  $G$  invert one another – can also be generalized:  $\exists_{\mathbf{z} \in \hat{\Omega}_{\mathbf{Z}}} \{E(\mathbf{x}) = g_{\mathbf{Z}}(\mathbf{z}) \wedge G(\mathbf{z}) = g_{\mathbf{X}}(\mathbf{x})\}$  for  $P_{\mathbf{X}}$ -almost every  $\mathbf{x} \in \Omega_{\mathbf{X}}$ , and  $\exists_{\mathbf{x} \in \hat{\Omega}_{\mathbf{X}}} \{E(\mathbf{x}) = g_{\mathbf{Z}}(\mathbf{z}) \wedge G(\mathbf{z}) = g_{\mathbf{X}}(\mathbf{x})\}$  for  $P_{\mathbf{Z}}$ -almost every  $\mathbf{z} \in \Omega_{\mathbf{Z}}$ .

## 5.3 Evaluation

We evaluate the feature learning capabilities of BiGANs by first training them unsupervised as described in Section 5.2.4, then transferring the encoder’s learned feature representations for use in auxiliary supervised learning tasks. To demonstrate that BiGANs are able to learn meaningful feature representations both on arbitrary data vectors, where the model is agnostic to any underlying structure, as well as very high-dimensional and complex distributions, we evaluate on both permutation-invariant MNIST (LeCun et al., 1998) and on the high-resolution natural images of ImageNet (Russakovsky et al., 2015).

In all experiments, each module  $D$ ,  $G$ , and  $E$  is a parametric deep (multi-layer) network. The BiGAN discriminator  $D(\mathbf{x}, \mathbf{z})$  takes data  $\mathbf{x}$  as its initial input, and at each linear layer thereafter, the latent representation  $\mathbf{z}$  is transformed using a learned linear transformation to the hidden layer dimension and added to the non-linearity input.

### 5.3.1 Baseline methods

Besides the BiGAN framework presented above, we considered alternative approaches to learning feature representations using different GAN variants.

**Discriminator.** The discriminator  $D$  in a standard GAN takes data samples  $\mathbf{x} \sim p_{\mathbf{X}}$  as input, making its learned intermediate representations natural candidates as feature representations for related tasks. This alternative is appealing as it requires no additional machinery, and is the approach used for unsupervised feature learning in Radford et al. (2016). On the other hand, it is not clear that the task of distinguishing between real and generated data requires or benefits from intermediate representations that are useful as semantic feature representations. In fact, if  $G$  successfully generates the true data distribution  $p_{\mathbf{X}}(\mathbf{x})$ ,  $D$  may ignore the input data entirely and predict  $P(Y = 1) = P(Y = 1|\mathbf{x}) = \frac{1}{2}$  unconditionally, not learning any meaningful intermediate representations.

**Latent regressor.** We consider an alternative encoder training by minimizing a reconstruction loss  $\mathcal{L}(\mathbf{z}, E(G(\mathbf{z})))$ , after or jointly during a regular GAN training, called latent regressor or joint latent regressor respectively. We use a sigmoid cross entropy loss  $\mathcal{L}$  as it naturally maps to a uniformly distributed output space. Intuitively, a drawback of this approach is that, unlike the encoder in a BiGAN, the latent regressor encoder  $E$  is trained only on generated samples  $G(\mathbf{z})$ , and never “sees” real data  $\mathbf{x} \sim p_{\mathbf{X}}$ . While this may not be an issue in the theoretical optimum where  $p_G(\mathbf{x}) = p_{\mathbf{X}}(\mathbf{x})$  exactly – i.e.,  $G$  perfectly generates the data distribution  $p_{\mathbf{X}}$  – in practice, for highly complex data distributions  $p_{\mathbf{X}}$ , such as the distribution of natural images, the generator will almost never achieve this perfect result. The fact that

BiGAN	$D$	LR	JLR	AE ( $\ell_2$ )	AE ( $\ell_1$ )
97.39	97.30	97.44	97.13	97.58	97.63

Table 5.1. One Nearest Neighbors (1NN) classification accuracy (%) on the permutation-invariant MNIST (LeCun et al., 1998) test set in the feature space learned by BiGAN, Latent Regressor (LR), Joint Latent Regressor (JLR), and an autoencoder (AE) using an  $\ell_1$  or  $\ell_2$  distance.

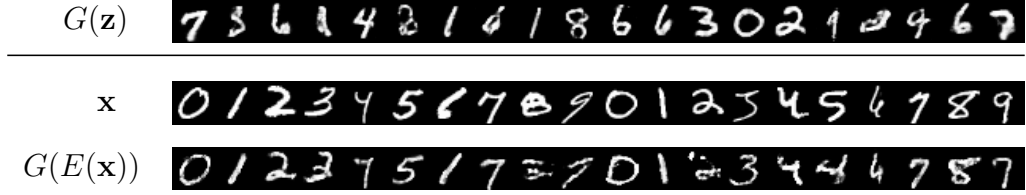


Figure 5.2. Qualitative results for permutation-invariant MNIST BiGAN training, including generator samples  $G(\mathbf{z})$ , real data  $\mathbf{x}$ , and corresponding reconstructions  $G(E(\mathbf{x}))$ .

the real data  $\mathbf{x}$  are never input to this type of encoder limits its utility as a feature representation for related tasks, as shown later in this section.

### 5.3.2 Permutation-invariant MNIST

We first present results on permutation-invariant MNIST (LeCun et al., 1998). In the permutation-invariant setting, each  $28 \times 28$  digit image must be treated as an unstructured 784D vector (Goodfellow et al., 2013). In our case, this condition is met by designing each module as a multi-layer perceptron (MLP), agnostic to the underlying spatial structure in the data (as opposed to a convnet, for example). See Appendix 5.5.3.1 for more architectural and training details. We set the latent distribution  $p_{\mathbf{z}} = [\mathcal{U}(-1, 1)]^{50}$  – a 50D continuous uniform distribution.

Table 5.1 compares the encoding learned by a BiGAN-trained encoder  $E$  with the baselines described in Section 5.3.1, as well as autoencoders (Hinton and Salakhutdinov, 2006) trained directly to minimize either  $\ell_2$  or  $\ell_1$  reconstruction error. The same architecture and optimization algorithm is used across all methods. All methods, including BiGAN, perform at roughly the same level. This result is not overly surprising given the relative simplicity of MNIST digits. For example, digits generated by  $G$  in a GAN nearly perfectly match the data distribution (qualitatively), making the latent regressor (LR) baseline method a reasonable choice, as argued in Section 5.3.1. Qualitative results are presented in Figure 5.2.

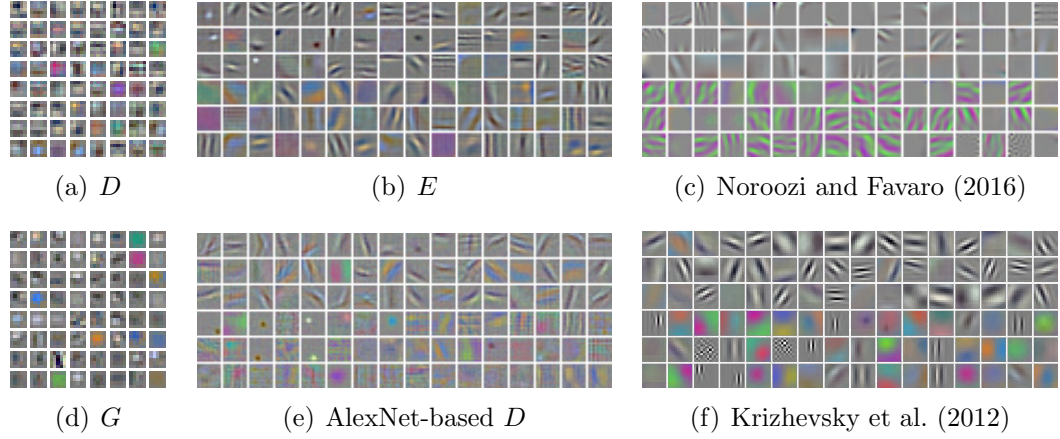


Figure 5.3. The convolutional filters learned by the three modules ( $D$ ,  $G$ , and  $E$ ) of a BiGAN (left, top-middle) trained on the ImageNet (Russakovsky et al., 2015) database. We compare with the filters learned by a discriminator  $D$  trained with the same architecture (bottom-middle), as well as the filters reported by Noroozi and Favaro (2016), and by Krizhevsky et al. (2012) for fully supervised ImageNet training (right).

### 5.3.3 ImageNet

Next, we present results from training BiGANs on ImageNet LSVRC (Russakovsky et al., 2015), a large-scale database of natural images. GANs trained on ImageNet cannot perfectly reconstruct the data, but often capture some interesting aspects. Here, each of  $D$ ,  $G$ , and  $E$  is a convnet. In all experiments, the encoder  $E$  architecture follows AlexNet (Krizhevsky et al., 2012) through the fifth and last convolution layer (*conv5*). We also experiment with an AlexNet-based discriminator  $D$  as a baseline feature learning approach. We set the latent distribution  $p_{\mathbf{z}} = [\mathcal{U}(-1, 1)]^{200}$  – a 200D continuous uniform distribution. Additionally, we experiment with higher resolution encoder input images –  $112 \times 112$  rather than the  $64 \times 64$  used elsewhere – using the generalization described in Section 5.2.5. See Appendix 5.5.3.2 for more architectural and training details.

**Qualitative results.** The convolutional filters learned by each of the three modules are shown in Figure 5.3. We see that the filters learned by the encoder  $E$  have clear Gabor-like structure, similar to those originally reported for the fully supervised AlexNet model (Krizhevsky et al., 2012). The filters also have similar “grouping” structure where one half (the bottom half, in this case) is more color sensitive, and the other half is more edge sensitive. (This separation of the filters occurs due to the AlexNet architecture maintaining two separate filter paths for computational efficiency.)



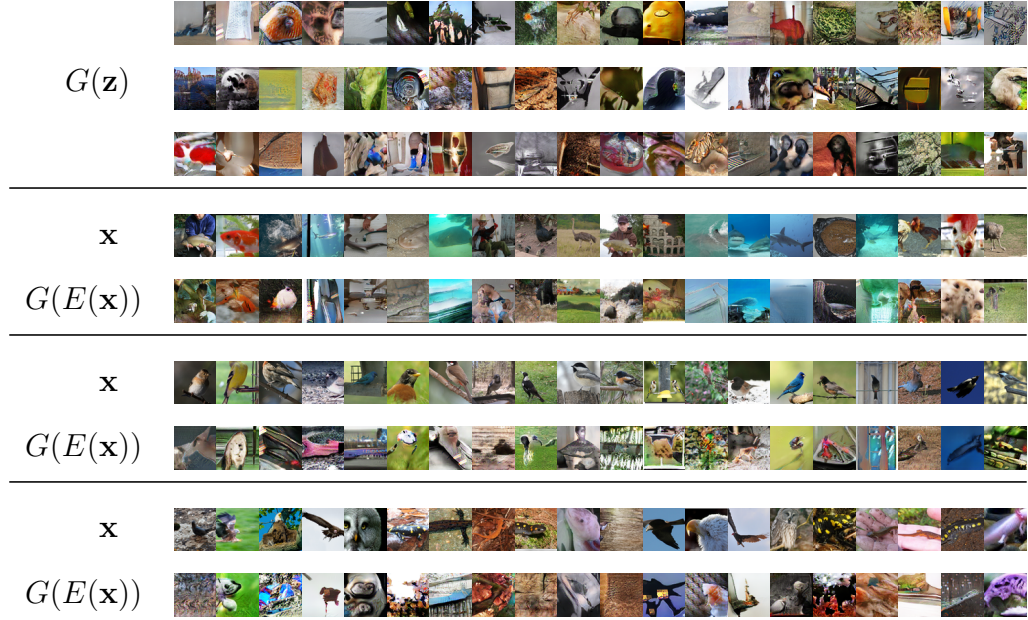


Figure 5.4. Qualitative results for ImageNet BiGAN training, including generator samples  $G(\mathbf{z})$ , real data  $\mathbf{x}$ , and corresponding reconstructions  $G(E(\mathbf{x}))$ .

In Figure 5.4 we present sample generations  $G(\mathbf{z})$ , as well as real data samples  $\mathbf{x}$  and their BiGAN reconstructions  $G(E(\mathbf{x}))$ . The reconstructions, while certainly imperfect, demonstrate empirically that the BiGAN encoder  $E$  and generator  $G$  learn approximate inverse mappings, as shown theoretically in Theorem 2. In Appendix 5.5.3.2, we present nearest neighbors in the BiGAN learned feature space.

**ImageNet classification.** Following Noroozi and Favaro (2016), we evaluate by freezing the first  $N$  layers of our pretrained network and randomly reinitializing and training the remainder fully supervised for ImageNet classification. Results are reported in Table 5.2.

**VOC classification, detection, and segmentation.** We evaluate the transferability of BiGAN representations to the PASCAL VOC (Everingham et al., 2014) computer vision benchmark tasks, including classification, object detection, and semantic segmentation. The classification task involves simple binary prediction of presence or absence in a given image for each of 20 object categories. The object detection and semantic segmentation tasks go a step further by requiring the objects to be localized, with semantic segmentation requiring this at the finest scale: pixelwise prediction of object identity. For detection, the pretrained model is used as the initialization for *Fast R-CNN* (Girshick, 2015) (FRCN) training; and for semantic segmentation, the model is used as the initialization for *Fully Convolutional Network* (Long et al.,

	conv1	conv2	conv3	conv4	conv5
Random (Noroozi and Favaro, 2016)	48.5	41.0	34.8	27.1	12.0
Wang and Gupta (2015)	51.8	46.9	42.8	38.8	29.8
Doersch et al. (2015)	53.1	47.6	48.7	<b>45.6</b>	30.4
Noroozi and Favaro (2016)*	57.1	56.0	52.4	48.3	38.1
BiGAN (ours)	<b>56.2</b>	<b>54.4</b>	<b>49.4</b>	43.9	33.3
BiGAN, $112 \times 112$ $E$ (ours)	55.3	53.2	49.3	44.4	<b>34.8</b>

Table 5.2. Classification accuracy (%) for the ImageNet LSVRC (Russakovsky et al., 2015) validation set with various portions of the network frozen, or reinitialized and trained from scratch, following the evaluation from Noroozi and Favaro (2016). In, e.g., the *conv3* column, the first three layers – conv1 through conv3 – are transferred and frozen, and the last layers – conv4, conv5, and fully connected layers – are reinitialized and trained fully supervised for ImageNet classification. BiGAN is competitive with these contemporary visual feature learning methods, despite its generality. (\*Results from Noroozi and Favaro (2016) are not directly comparable to those of the other methods as a different base convnet architecture with larger intermediate feature maps is used.)

2015) (FCN) training, in each case replacing the *AlexNet* (Krizhevsky et al., 2012) model trained fully supervised for ImageNet classification. We report results on each of these tasks in Table 5.3, comparing BiGANs with contemporary approaches to unsupervised (Krähenbühl et al., 2016) and self-supervised (Agrawal et al., 2015; Doersch et al., 2015; Pathak et al., 2016; Wang and Gupta, 2015) feature learning in the visual domain, as well as the baselines discussed in Section 5.3.1.

## 5.4 Discussion

Despite making no assumptions about the underlying structure of the data, the BiGAN unsupervised feature learning framework offers a representation competitive with existing self-supervised and even weakly supervised feature learning approaches for visual feature learning, while still being a purely generative model with the ability to sample data  $\mathbf{x}$  and predict latent representation  $\mathbf{z}$ . Furthermore, BiGANs outperform the discriminator ( $D$ ) and latent regressor (LR) baselines discussed in Section 5.3.1, confirming our intuition that these approaches may not perform well in the regime of highly complex data distributions such as that of natural images. The version in which the encoder takes a higher resolution image than output by the generator (*BiGAN*  $112 \times 112$   $E$ ) performs better still, and this strategy is not possible under the LR and  $D$  baselines as each of those modules take generator outputs as their input.

Although existing self-supervised approaches have shown impressive performance

		Classification (% mAP)			<i>FRCN</i> Detection (% mAP)	<i>FCN</i> Segmentation (% mIU)
trained layers		fc8	fc6-8	all	all	all
sup.	ImageNet (Krizhevsky et al., 2012)	<b>77.0</b>	<b>78.8</b>	<b>78.3</b>	<b>56.8</b>	<b>48.0</b>
self-sup.	Agrawal et al. (2015)	31.2	31.0	54.2	43.9	-
	Pathak et al. (2016)	30.5	34.6	56.5	44.5	<b>30.0</b>
	Wang and Gupta (2015)	28.4	<b>55.6</b>	63.1	47.4	-
	Doersch et al. (2015)	<b>44.7</b>	55.1	<b>65.3</b>	<b>51.1</b>	-
unsup.	<i>k</i> -means (Krähenbühl et al., 2016)	32.0	39.2	56.6	45.6	32.6
	Discriminator ( <i>D</i> )	30.7	40.5	56.4	-	-
	Latent Regressor (LR)	36.9	47.9	57.1	-	-
	Joint LR	37.1	47.9	56.5	-	-
	Autoencoder ( $\ell_2$ )	24.8	16.0	53.8	41.9	-
	BiGAN (ours)	37.5	48.7	58.9	46.2	34.9
	BiGAN, $112 \times 112$ <i>E</i> (ours)	<b>41.7</b>	<b>52.5</b>	<b>60.3</b>	<b>46.9</b>	<b>35.2</b>

Table 5.3. Classification and Fast R-CNN (Girshick, 2015) detection results for the PASCAL VOC 2007 (Everingham et al., 2014) test set, and FCN (Long et al., 2015) segmentation results on the PASCAL VOC 2012 validation set, under the standard mean average precision (mAP) or mean intersection over union (mIU) metrics for each task. Classification models are trained with various portions of the *AlexNet* (Krizhevsky et al., 2012) model frozen. In the *fc8* column, only the linear classifier (a multinomial logistic regression) is learned – in the case of BiGAN, on top of randomly initialized fully connected (FC) layers *fc6* and *fc7*. In the *fc6-8* column, all three FC layers are trained fully supervised with all convolution layers frozen. Finally, in the *all* column, the entire network is “fine-tuned”. BiGAN outperforms other unsupervised (*unsup.*) feature learning approaches, including the GAN-based baselines described in Section 5.3.1, and despite its generality, is competitive with contemporary self-supervised (*self-sup.*) feature learning approaches specific to the visual domain.

and thus far tended to outshine purely unsupervised approaches in the complex domain of high-resolution images, purely unsupervised approaches to feature learning or pre-training have several potential benefits.

BiGAN and other unsupervised learning approaches are agnostic to the domain of the data. The self-supervised approaches are specific to the visual domain, in some cases requiring weak supervision from video unavailable in images alone. For example, the methods are not applicable in the permutation-invariant MNIST setting explored in Section 5.3.2, as the data are treated as flat vectors rather than 2D images.

Furthermore, BiGAN and other unsupervised approaches needn't suffer from domain shift between the pre-training task and the transfer task, unlike self-supervised methods in which some aspect of the data is normally removed or corrupted in order to create a non-trivial prediction task. In the context prediction task (Doersch et al., 2015), the network sees only small image patches – the global image structure is unobserved. In the context encoder or inpainting task (Pathak et al., 2016), each image is corrupted by removing large areas to be filled in by the prediction network, creating inputs with dramatically different appearance from the uncorrupted natural images seen in the transfer tasks.

Other approaches (Agrawal et al., 2015; Wang and Gupta, 2015) rely on auxiliary information unavailable in the static image domain, such as video, egomotion, or tracking. Unlike BiGAN, such approaches cannot learn feature representations from unlabeled static images.

We finally note that the results presented here constitute only a preliminary exploration of the space of model architectures possible under the BiGAN framework, and we expect results to improve significantly with advancements in generative image models and discriminative convolutional networks alike.

## 5.5 Appendix

### 5.5.1 Additional proofs

#### 5.5.1.1 Proof of Proposition 1 (optimal discriminator)

**Proposition 1** *For any  $E$  and  $G$ , the optimal discriminator  $D_{EG}^* := \operatorname{argmax}_D V(D, E, G)$  is the Radon-Nikodym derivative  $f_{EG} := \frac{dP_{E\mathbf{X}}}{d(P_{E\mathbf{X}} + P_{G\mathbf{Z}})} : \Omega \mapsto [0, 1]$  of measure  $P_{E\mathbf{X}}$  with respect to measure  $P_{E\mathbf{X}} + P_{G\mathbf{Z}}$ .*

*Proof.* For measures  $P$  and  $Q$  on space  $\Omega$ , with  $P$  absolutely continuous with respect to  $Q$ , the RN derivative  $f_{PQ} := \frac{dP}{dQ}$  exists, and we have

$$\mathbb{E}_{\mathbf{x} \sim P} [g(\mathbf{x})] = \int_{\Omega} g \, dP = \int_{\Omega} g \frac{dP}{dQ} \, dQ = \int_{\Omega} g f_{PQ} \, dQ = \mathbb{E}_{\mathbf{x} \sim Q} [f_{PQ}(\mathbf{x}) g(\mathbf{x})]. \quad (5.4)$$

Let the probability measure  $P_{EG} := \frac{P_{E\mathbf{X}} + P_{G\mathbf{Z}}}{2}$  denote the average of measures  $P_{E\mathbf{X}}$  and  $P_{G\mathbf{Z}}$ . Both  $P_{E\mathbf{X}}$  and  $P_{G\mathbf{Z}}$  are each absolutely continuous with respect to  $P_{EG}$ . Hence

the RN derivatives  $f_{EG} := \frac{dP_{E\mathbf{X}}}{d(P_{E\mathbf{X}}+P_{G\mathbf{Z}})} = \frac{1}{2} \frac{dP_{E\mathbf{X}}}{dP_{EG}}$  and  $f_{GE} := \frac{dP_{G\mathbf{Z}}}{d(P_{E\mathbf{X}}+P_{G\mathbf{Z}})} = \frac{1}{2} \frac{dP_{G\mathbf{Z}}}{dP_{EG}}$  exist and sum to 1:

$$f_{EG} + f_{GE} = \frac{dP_{E\mathbf{X}}}{d(P_{E\mathbf{X}}+P_{G\mathbf{Z}})} + \frac{dP_{G\mathbf{Z}}}{d(P_{E\mathbf{X}}+P_{G\mathbf{Z}})} = \frac{d(P_{E\mathbf{X}}+P_{G\mathbf{Z}})}{d(P_{E\mathbf{X}}+P_{G\mathbf{Z}})} = 1. \quad (5.5)$$

We use (5.4) and (5.5) to rewrite the objective  $V$  (5.3) as a single expectation under measure  $P_{EG}$ :

$$\begin{aligned} V(D, E, G) &= \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{E\mathbf{X}}} [\log D(\mathbf{x}, \mathbf{z})] + \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{G\mathbf{Z}}} [\log (1 - D(\mathbf{x}, \mathbf{z}))] \\ &= \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{EG}} \left[ \underbrace{2f_{EG}(\mathbf{x}, \mathbf{z})}_{\frac{dP_{E\mathbf{X}}}{dP_{EG}}} \log D(\mathbf{x}, \mathbf{z}) \right] + \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{EG}} \left[ \underbrace{2f_{GE}(\mathbf{x}, \mathbf{z})}_{\frac{dP_{G\mathbf{Z}}}{dP_{EG}}} \log (1 - D(\mathbf{x}, \mathbf{z})) \right] \\ &= 2 \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{EG}} [f_{EG}(\mathbf{x}, \mathbf{z}) \log D(\mathbf{x}, \mathbf{z}) + f_{GE}(\mathbf{x}, \mathbf{z}) \log (1 - D(\mathbf{x}, \mathbf{z}))] \\ &= 2 \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{EG}} [f_{EG}(\mathbf{x}, \mathbf{z}) \log D(\mathbf{x}, \mathbf{z}) + (1 - f_{EG}(\mathbf{x}, \mathbf{z})) \log (1 - D(\mathbf{x}, \mathbf{z}))]. \end{aligned}$$

Note that  $\operatorname{argmax}_y \{a \log y + (1 - a) \log(1 - y)\} = a$  for any  $a \in [0, 1]$ . Thus,  $D_{EG}^* = f_{EG}$ .  $\square$

### 5.5.1.2 Proof of Proposition 2 (encoder and generator objective)

**Proposition 2** *The encoder and generator's objective for an optimal discriminator  $C(E, G) := \max_D V(D, E, G) = V(D_{EG}^*, E, G)$  can be rewritten in terms of the Jensen-Shannon divergence between measures  $P_{E\mathbf{X}}$  and  $P_{G\mathbf{Z}}$  as  $C(E, G) = 2 D_{JS}(P_{E\mathbf{X}} \parallel P_{G\mathbf{Z}}) - \log 4$ .*

*Proof.* Using Proposition 1 along with (5.5) ( $1 - D_{EG}^* = 1 - f_{EG} = f_{GE}$ ) we rewrite the objective

$$\begin{aligned} C(E, G) &= \max_D V(D, E, G) = V(D_{EG}^*, E, G) \\ &= \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{E\mathbf{X}}} [\log D_{EG}^*(\mathbf{x}, \mathbf{z})] + \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{G\mathbf{Z}}} [\log (1 - D_{EG}^*(\mathbf{x}, \mathbf{z}))] \\ &= \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{E\mathbf{X}}} [\log f_{EG}(\mathbf{x}, \mathbf{z})] + \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{G\mathbf{Z}}} [\log f_{GE}(\mathbf{x}, \mathbf{z})] \\ &= \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{E\mathbf{X}}} [\log (2f_{EG}(\mathbf{x}, \mathbf{z}))] + \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{G\mathbf{Z}}} [\log (2f_{GE}(\mathbf{x}, \mathbf{z}))] - \log 4 \\ &= D_{KL}(P_{E\mathbf{X}} \parallel P_{EG}) + D_{KL}(P_{G\mathbf{Z}} \parallel P_{EG}) - \log 4 \\ &= D_{KL}(P_{E\mathbf{X}} \parallel \frac{P_{E\mathbf{X}}+P_{G\mathbf{Z}}}{2}) + D_{KL}(P_{G\mathbf{Z}} \parallel \frac{P_{E\mathbf{X}}+P_{G\mathbf{Z}}}{2}) - \log 4 \\ &= 2 D_{JS}(P_{E\mathbf{X}} \parallel P_{G\mathbf{Z}}) - \log 4. \quad \square \end{aligned}$$

### 5.5.1.3 Measure definitions for deterministic $E$ and $G$

While Theorem 1 and Propositions 1 and 2 hold for any encoder  $p_E(\mathbf{z}|\mathbf{x})$  and generator  $p_G(\mathbf{x}|\mathbf{z})$ , stochastic or deterministic, Theorems 2 and 3 assume the encoder  $E$  and generator  $G$  are deterministic functions; i.e., with conditionals  $p_E(\mathbf{z}|\mathbf{x}) = \delta(\mathbf{z} - E(\mathbf{x}))$  and  $p_G(\mathbf{x}|\mathbf{z}) = \delta(\mathbf{x} - G(\mathbf{z}))$  defined as  $\delta$  functions.

For use in the proofs of those theorems, we simplify the definitions of measures  $P_{E\mathbf{X}}$  and  $P_{G\mathbf{Z}}$  given in Section 5.2 for the case of deterministic functions  $E$  and  $G$  below:

$$\begin{aligned}
P_{E\mathbf{X}}(R) &= \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \int_{\Omega_{\mathbf{Z}}} p_E(\mathbf{z}|\mathbf{x}) \mathbf{1}_{[(\mathbf{x}, \mathbf{z}) \in R]} d\mathbf{z} d\mathbf{x} \\
&= \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \left( \int_{\Omega_{\mathbf{Z}}} \delta(\mathbf{z} - E(\mathbf{x})) \mathbf{1}_{[(\mathbf{x}, \mathbf{z}) \in R]} d\mathbf{z} \right) d\mathbf{x} \\
&= \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \mathbf{1}_{[(\mathbf{x}, E(\mathbf{x})) \in R]} d\mathbf{x} \\
P_{G\mathbf{Z}}(R) &= \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \int_{\Omega_{\mathbf{X}}} p_G(\mathbf{x}|\mathbf{z}) \mathbf{1}_{[(\mathbf{x}, \mathbf{z}) \in R]} d\mathbf{x} d\mathbf{z} \\
&= \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \left( \int_{\Omega_{\mathbf{X}}} \delta(\mathbf{x} - G(\mathbf{z})) \mathbf{1}_{[(\mathbf{x}, \mathbf{z}) \in R]} d\mathbf{x} \right) d\mathbf{z} \\
&= \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \mathbf{1}_{[(G(\mathbf{z}), \mathbf{z}) \in R]} d\mathbf{z}
\end{aligned}$$

#### 5.5.1.4 Proof of Theorem 2 (optimal generator and encoder are inverses)

**Theorem 2** *If  $E$  and  $G$  are an optimal encoder and generator, then  $E = G^{-1}$  almost everywhere; that is,  $G(E(\mathbf{x})) = \mathbf{x}$  for  $P_{\mathbf{X}}$ -almost every  $\mathbf{x} \in \Omega_{\mathbf{X}}$ , and  $E(G(\mathbf{z})) = \mathbf{z}$  for  $P_{\mathbf{Z}}$ -almost every  $\mathbf{z} \in \Omega_{\mathbf{Z}}$ .*

*Proof.* Let  $R_{\mathbf{X}}^0 := \{\mathbf{x} \in \Omega_{\mathbf{X}} : \mathbf{x} \neq G(E(\mathbf{x}))\}$  be the region of  $\Omega_{\mathbf{X}}$  in which the inversion property  $\mathbf{x} = G(E(\mathbf{x}))$  does *not* hold. We will show that, for optimal  $E$  and  $G$ ,  $R_{\mathbf{X}}^0$  has measure zero under  $P_{\mathbf{X}}$  (i.e.,  $P_{\mathbf{X}}(R_{\mathbf{X}}^0) = 0$ ) and therefore  $\mathbf{x} = G(E(\mathbf{x}))$  holds  $P_{\mathbf{X}}$ -almost everywhere.

Let  $R^0 := \{(\mathbf{x}, \mathbf{z}) \in \Omega : \mathbf{z} = E(\mathbf{x}) \wedge \mathbf{x} \in R_{\mathbf{X}}^0\}$  be the region of  $\Omega$  such that  $(\mathbf{x}, E(\mathbf{x})) \in R^0$  if and only if  $\mathbf{x} \in R_{\mathbf{X}}^0$ . We'll use the definitions of  $P_{E\mathbf{X}}$  and  $P_{G\mathbf{Z}}$  for deterministic  $E$  and  $G$  (Appendix 5.5.1.3), and the fact that  $P_{E\mathbf{X}} = P_{G\mathbf{Z}}$  for optimal  $E$  and  $G$  (Theorem 1).

$$\begin{aligned}
P_{\mathbf{X}}(R_{\mathbf{X}}^0) &= \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \mathbf{1}_{[\mathbf{x} \in R_{\mathbf{X}}^0]} d\mathbf{x} \\
&= \int_{\Omega_{\mathbf{X}}} p_{\mathbf{X}}(\mathbf{x}) \mathbf{1}_{[(\mathbf{x}, E(\mathbf{x})) \in R^0]} d\mathbf{x} \\
&= P_{E\mathbf{X}}(R^0) \\
&= P_{G\mathbf{Z}}(R^0) \\
&= \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \mathbf{1}_{[(G(\mathbf{z}), \mathbf{z}) \in R^0]} d\mathbf{z} \\
&= \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \mathbf{1}_{[\mathbf{z} = E(G(\mathbf{z})) \wedge G(\mathbf{z}) \in R_{\mathbf{X}}^0]} d\mathbf{z} \\
&= \int_{\Omega_{\mathbf{Z}}} p_{\mathbf{Z}}(\mathbf{z}) \underbrace{\mathbf{1}_{[\mathbf{z} = E(G(\mathbf{z})) \wedge G(\mathbf{z}) \neq G(E(G(\mathbf{z})))]}}_{=0 \text{ for any } \mathbf{z}, \text{ as } \mathbf{z} = E(G(\mathbf{z})) \implies G(\mathbf{z}) = G(E(G(\mathbf{z})))} d\mathbf{z} \\
&= 0.
\end{aligned}$$

Hence region  $R_{\mathbf{X}}^0$  has measure zero ( $P_{\mathbf{X}}(R_{\mathbf{X}}^0) = 0$ ), and the inversion property  $\mathbf{x} = G(E(\mathbf{x}))$  holds  $P_{\mathbf{X}}$ -almost everywhere.

An analogous argument shows that  $R_{\mathbf{Z}}^0 := \{\mathbf{z} \in \Omega_{\mathbf{Z}} : \mathbf{z} \neq E(G(\mathbf{z}))\}$  has measure zero on  $P_{\mathbf{Z}}$  (i.e.,  $P_{\mathbf{Z}}(R_{\mathbf{Z}}^0) = 0$ ) and therefore  $\mathbf{z} = E(G(\mathbf{z}))$  holds  $P_{\mathbf{Z}}$ -almost everywhere.  $\square$

### 5.5.1.5 Proof of Theorem 3 (relationship to autoencoders)

As shown in Proposition 2 (Section 5.2), the BiGAN objective is equivalent to the Jensen-Shannon divergence between  $P_{E\mathbf{X}}$  and  $P_{G\mathbf{Z}}$ . We now go a step further and show that this Jensen-Shannon divergence is closely related to a standard autoencoder loss. Omitting the  $\frac{1}{2}$  scale factor, a KL divergence term of the Jensen-Shannon divergence is given as

$$\begin{aligned} D_{\text{KL}}(P_{E\mathbf{X}} \parallel \frac{P_{E\mathbf{X}} + P_{G\mathbf{Z}}}{2}) &= \log 2 + \int_{\Omega} \log \frac{dP_{E\mathbf{X}}}{d(P_{E\mathbf{X}} + P_{G\mathbf{Z}})} dP_{E\mathbf{X}} \\ &= \log 2 + \int_{\Omega} \log f dP_{E\mathbf{X}}, \end{aligned} \quad (5.6)$$

where we abbreviate as  $f$  the Radon-Nikodym derivative  $f_{EG} := \frac{dP_{E\mathbf{X}}}{d(P_{E\mathbf{X}} + P_{G\mathbf{Z}})} \in [0, 1]$  defined in Proposition 1 for most of this proof.

We'll make use of the definitions of  $P_{E\mathbf{X}}$  and  $P_{G\mathbf{Z}}$  for deterministic  $E$  and  $G$  found in Appendix 5.5.1.3. The integral term of the KL divergence expression given in (5.6) over a particular region  $R \subseteq \Omega$  will be denoted by

$$F(R) := \int_R \log \frac{dP_{E\mathbf{X}}}{d(P_{E\mathbf{X}} + P_{G\mathbf{Z}})} dP_{E\mathbf{X}} = \int_R \log f dP_{E\mathbf{X}}.$$

Next we will show that  $f > 0$  holds  $P_{E\mathbf{X}}$ -almost everywhere, and hence  $F$  is always well defined and finite. We then show that  $F$  is equivalent to an autoencoder-like reconstruction loss function.

**Proposition 3**  $f > 0$   $P_{E\mathbf{X}}$ -almost everywhere.

*Proof.* Let  $R^{f=0} := \{(\mathbf{x}, \mathbf{z}) \in \Omega : f(\mathbf{x}, \mathbf{z}) = 0\}$  be the region of  $\Omega$  in which  $f = 0$ . Using the definition of the Radon-Nikodym derivative  $f$ , the measure  $P_{E\mathbf{X}}(R^{f=0}) = \int_{R^{f=0}} f d(P_{E\mathbf{X}} + P_{G\mathbf{Z}}) = \int_{R^{f=0}} 0 d(P_{E\mathbf{X}} + P_{G\mathbf{Z}}) = 0$  is zero. Hence  $f > 0$   $P_{E\mathbf{X}}$ -almost everywhere.  $\square$

Proposition 3 ensures that  $\log f$  is defined  $P_{E\mathbf{X}}$ -almost everywhere, and  $F(R)$  is well-defined. Next we will show that  $F(R)$  mimics an autoencoder with  $\ell_0$  loss, meaning  $F$  is zero for any region in which  $G(E(\mathbf{x})) \neq \mathbf{x}$ , and non-zero otherwise.

**Proposition 4** *The KL divergence  $F$  outside the support of  $P_{G\mathbf{Z}}$  is zero:  $F(\Omega \setminus \text{supp}(P_{G\mathbf{Z}})) = 0$ .*

We'll first show that in region  $R_S := \Omega \setminus \text{supp}(P_{GZ})$ , we have  $f = 1$   $P_{EX}$ -almost everywhere. Let  $R^{f<1} := \{(\mathbf{x}, \mathbf{z}) \in R_S : f(\mathbf{x}, \mathbf{z}) < 1\}$  be the region of  $R_S$  in which  $f < 1$ . Let's assume that  $P_{EX}(R^{f<1}) > 0$  has non-zero measure. Then, using the definition of the Radon-Nikodym derivative,

$$\begin{aligned} P_{EX}(R^{f<1}) &= \int_{R^{f<1}} f \, d(P_{EX} + P_{GZ}) = \int_{R^{f<1}} \underbrace{f}_{\leq \varepsilon < 1} dP_{EX} + \underbrace{\int_{R^{f<1}} f \, dP_{GZ}}_0 \leq \varepsilon P_{EX}(R^{f<1}) \\ &< P_{EX}(R^{f<1}), \end{aligned}$$

where  $\varepsilon$  is a constant smaller than 1. But  $P_{EX}(R^{f<1}) < P_{EX}(R^{f<1})$  is a contradiction; hence  $P_{EX}(R^{f<1}) = 0$  and  $f = 1$   $P_{EX}$ -almost everywhere in  $R_S$ , implying  $\log f = 0$   $P_{EX}$ -almost everywhere in  $R_S$ . Hence  $F(R_S) = 0$ .  $\square$

By definition,  $F(\Omega \setminus \text{supp}(P_{EX})) = 0$  is also zero. The only region where  $F$  might be non-zero is  $R^1 := \text{supp}(P_{EX}) \cap \text{supp}(P_{GZ})$ .

**Proposition 5**  $f < 1$   $P_{EX}$ -almost everywhere in  $R^1$ .

Let  $R^{f=1} := \{(\mathbf{x}, \mathbf{z}) \in R^1 : f(\mathbf{x}, \mathbf{z}) = 1\}$  be the region in which  $f = 1$ . Let's assume the set  $R^{f=1} \neq \emptyset$  is not empty. By definition of the support<sup>2</sup>,  $P_{EX}(R^{f=1}) > 0$  and  $P_{GZ}(R^{f=1}) > 0$ . The Radon-Nikodym derivative on  $R^{f=1}$  is then given by

$$P_{EX}(R^{f=1}) = \int_{R^{f=1}} f \, d(P_{EX} + P_{GZ}) = \int_{R^{f=1}} 1 \, d(P_{EX} + P_{GZ}) = P_{EX}(R^{f=1}) + P_{GZ}(R^{f=1}),$$

which implies  $P_{GZ}(R^{f=1}) = 0$  and contradicts the definition of support. Hence  $R^{f=1} = \emptyset$  and  $f < 1$   $P_{EX}$ -almost everywhere on  $R^1$ , implying  $\log f < 0$   $P_{EX}$ -almost everywhere.  $\square$

**Theorem 3** *The encoder and generator objective given an optimal discriminator  $C(E, G) := \max_D V(D, E, G)$  can be rewritten as an  $\ell_0$  autoencoder loss function*

$$\begin{aligned} C(E, G) &= \mathbb{E}_{\mathbf{x} \sim p_{\mathbf{x}}} \left[ \mathbf{1}_{[E(\mathbf{x}) \in \hat{\Omega}_{\mathbf{z}} \wedge G(E(\mathbf{x})) = \mathbf{x}]} \log f_{EG}(\mathbf{x}, E(\mathbf{x})) \right] + \\ &\quad \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \left[ \mathbf{1}_{[G(\mathbf{z}) \in \hat{\Omega}_{\mathbf{x}} \wedge E(G(\mathbf{z})) = \mathbf{z}]} \log (1 - f_{EG}(G(\mathbf{z}), \mathbf{z})) \right] \end{aligned}$$

with  $\log f_{EG} \in (-\infty, 0)$  and  $\log (1 - f_{EG}) \in (-\infty, 0)$   $P_{EX}$ -almost and  $P_{GZ}$ -almost everywhere.

*Proof.* Proposition 4 ( $F(\Omega \setminus \text{supp}(P_{GZ})) = 0$ ) and  $F(\Omega \setminus \text{supp}(P_{EX})) = 0$  imply that  $R^1 := \text{supp}(P_{EX}) \cap \text{supp}(P_{GZ})$  is the only region of  $\Omega$  where  $F$  may be non-zero; hence  $F(\Omega) = F(R^1)$ . Note that

$$\begin{aligned} \text{supp}(P_{EX}) &= \{(\mathbf{x}, E(\mathbf{x})) : \mathbf{x} \in \hat{\Omega}_{\mathbf{x}}\} \\ \text{supp}(P_{GZ}) &= \{(G(\mathbf{z}), \mathbf{z}) : \mathbf{z} \in \hat{\Omega}_{\mathbf{z}}\} \\ \implies R^1 &:= \text{supp}(P_{EX}) \cap \text{supp}(P_{GZ}) = \{(\mathbf{x}, \mathbf{z}) : E(\mathbf{x}) = \mathbf{z} \wedge \mathbf{x} \in \hat{\Omega}_{\mathbf{x}} \wedge G(\mathbf{z}) = \mathbf{x} \wedge \mathbf{z} \in \hat{\Omega}_{\mathbf{z}}\} \end{aligned}$$

<sup>2</sup>We use the definition  $U \cap C \neq \emptyset \implies \mu(U \cap C) > 0$  here.



So a point  $(\mathbf{x}, E(\mathbf{x}))$  is in  $R^1$  if  $\mathbf{x} \in \hat{\Omega}_{\mathbf{X}}$ ,  $E(\mathbf{x}) \in \hat{\Omega}_{\mathbf{Z}}$ , and  $G(E(\mathbf{x})) = \mathbf{x}$ . (We can omit the  $\mathbf{x} \in \hat{\Omega}_{\mathbf{X}}$  condition from inside an expectation over  $P_{\mathbf{X}}$ , as  $P_{\mathbf{X}}$ -almost all  $\mathbf{x} \notin \hat{\Omega}_{\mathbf{X}}$  have 0 probability.) Therefore,

$$\begin{aligned} D_{\text{KL}}(P_{E\mathbf{X}} \parallel \frac{P_{E\mathbf{X}} + P_{G\mathbf{Z}}}{2}) - \log 2 &= F(\Omega) = F(R^1) \\ &= \int_{R^1} \log f(\mathbf{x}, \mathbf{z}) dP_{E\mathbf{X}} \\ &= \int_{\Omega} \mathbf{1}_{[(\mathbf{x}, \mathbf{z}) \in R^1]} \log f(\mathbf{x}, \mathbf{z}) dP_{E\mathbf{X}} \\ &= \mathbb{E}_{(\mathbf{x}, \mathbf{z}) \sim P_{E\mathbf{X}}} [\mathbf{1}_{[(\mathbf{x}, \mathbf{z}) \in R^1]} \log f(\mathbf{x}, \mathbf{z})] \\ &= \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{X}}} [\mathbf{1}_{[(\mathbf{x}, E(\mathbf{x})) \in R^1]} \log f(\mathbf{x}, E(\mathbf{x}))] \\ &= \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{X}}} [\mathbf{1}_{[E(\mathbf{x}) \in \hat{\Omega}_{\mathbf{Z}} \wedge G(E(\mathbf{x})) = \mathbf{x}]} \log f(\mathbf{x}, E(\mathbf{x}))]. \end{aligned}$$

Finally, with Propositions 3 and 5, we have  $f \in (0, 1)$   $P_{E\mathbf{X}}$ -almost everywhere in  $R^1$ , and therefore  $\log f \in (-\infty, 0)$ , taking a finite and strictly negative value  $P_{E\mathbf{X}}$ -almost everywhere.

An analogous argument (along with the fact that  $f_{EG} + f_{GE} = 1$ ) lets us rewrite the other KL divergence term

$$\begin{aligned} D_{\text{KL}}(P_{G\mathbf{Z}} \parallel \frac{P_{E\mathbf{X}} + P_{G\mathbf{Z}}}{2}) - \log 2 &= \mathbb{E}_{\mathbf{z} \sim P_{\mathbf{Z}}} [\mathbf{1}_{[G(\mathbf{z}) \in \hat{\Omega}_{\mathbf{X}} \wedge E(G(\mathbf{z})) = \mathbf{z}]} \log f_{GE}(G(\mathbf{z}), \mathbf{z})] \\ &= \mathbb{E}_{\mathbf{z} \sim P_{\mathbf{Z}}} [\mathbf{1}_{[G(\mathbf{z}) \in \hat{\Omega}_{\mathbf{X}} \wedge E(G(\mathbf{z})) = \mathbf{z}]} \log (1 - f_{EG}(G(\mathbf{z}), \mathbf{z}))] \end{aligned}$$

The Jensen-Shannon divergence is the mean of these two KL divergences, giving  $C(E, G)$ :

$$\begin{aligned} C(E, G) &= 2 D_{\text{JS}}(P_{E\mathbf{X}} \parallel P_{G\mathbf{Z}}) - \log 4 \\ &= D_{\text{KL}}(P_{E\mathbf{X}} \parallel \frac{P_{E\mathbf{X}} + P_{G\mathbf{Z}}}{2}) + D_{\text{KL}}(P_{G\mathbf{Z}} \parallel \frac{P_{E\mathbf{X}} + P_{G\mathbf{Z}}}{2}) - \log 4 \\ &= \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{X}}} [\mathbf{1}_{[E(\mathbf{x}) \in \hat{\Omega}_{\mathbf{Z}} \wedge G(E(\mathbf{x})) = \mathbf{x}]} \log f_{EG}(\mathbf{x}, E(\mathbf{x}))] + \\ &\quad \mathbb{E}_{\mathbf{z} \sim P_{\mathbf{Z}}} [\mathbf{1}_{[G(\mathbf{z}) \in \hat{\Omega}_{\mathbf{X}} \wedge E(G(\mathbf{z})) = \mathbf{z}]} \log (1 - f_{EG}(G(\mathbf{z}), \mathbf{z}))] \square \end{aligned}$$

### 5.5.2 Learning details

In this section we provide additional details on the BiGAN learning protocol summarized in Section 5.2.4. Goodfellow et al. (2014) found for GAN training that an objective in which the real and generated labels  $Y$  are swapped provides stronger gradient signal to  $G$ . We similarly observed in BiGAN training that an “inverse” objective  $\Lambda$  (with the same fixed point characteristics as  $V$ ) provides stronger gradient signal to  $G$  and  $E$ , where

$$\Lambda(D, G, E) = \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{X}}} \left[ \underbrace{\mathbb{E}_{\mathbf{z} \sim P_E(\cdot|\mathbf{x})} [\log (1 - D(\mathbf{x}, \mathbf{z}))]}_{\log(1 - D(\mathbf{x}, E(\mathbf{x})))} \right] + \mathbb{E}_{\mathbf{z} \sim P_{\mathbf{Z}}} \left[ \underbrace{\mathbb{E}_{\mathbf{x} \sim P_G(\cdot|\mathbf{z})} [\log D(\mathbf{x}, \mathbf{z})]}_{\log D(G(\mathbf{z}), \mathbf{z})} \right].$$

In practice,  $\theta_G$  and  $\theta_E$  are updated by moving in the positive gradient direction of this inverse objective  $\nabla_{\theta_E, \theta_G} \Lambda$ , rather than the negative gradient direction of the original objective.

We also observed that learning behaved similarly when all parameters  $\theta_D$ ,  $\theta_G$ ,  $\theta_E$  were updated simultaneously at each iteration rather than alternating between  $\theta_D$  updates and  $\theta_G, \theta_E$  updates, so we took the simultaneous updating (non-alternating) approach for computational efficiency. (For standard GAN training, simultaneous updates of  $\theta_D$ ,  $\theta_G$  performed similarly well, so our standard GAN experiments also follow this protocol.)

### 5.5.3 Model and training details

In the following sections we present additional details on the models and training protocols used in the permutation-invariant MNIST and ImageNet evaluations presented in Section 5.3.

**Optimization.** For unsupervised training of BiGANs and baseline methods, we use the Adam optimizer (Kingma and Ba, 2015) to compute parameter updates, following the hyperparameters (initial step size  $\alpha = 2 \times 10^{-4}$ , momentum  $\beta_1 = 0.5$  and  $\beta_2 = 0.999$ ) used by Radford et al. (2016). The step size  $\alpha$  is decayed exponentially to  $\alpha = 2 \times 10^{-6}$  starting halfway through training. The mini-batch size is 128.  $\ell_2$  weight decay of  $2.5 \times 10^{-5}$  is applied to all multiplicative weights in linear layers (but not to the learned bias  $\beta$  or scale  $\gamma$  parameters applied after batch normalization). Weights are initialized from a zero-mean normal distribution with a standard deviation of 0.02, with one notable exception: BiGAN discriminator weights that directly multiply  $\mathbf{z}$  inputs to be added to spatial convolution outputs have initializations scaled by the convolution kernel size – e.g., for a  $5 \times 5$  kernel, weights are initialized with a standard deviation of 0.5, 25 times the standard initialization.

**Software & hardware.** We implement BiGANs and baseline feature learning methods using the *Theano* (Theano Development Team, 2016) framework, based on the convolutional GAN implementation provided by Radford et al. (2016). ImageNet transfer learning experiments (Section 5.3.3) use the *Caffe* (Jia et al., 2014) framework, per the Fast R-CNN (Girshick, 2015) and FCN (Long et al., 2015) reference implementations. Most computation is performed on an NVIDIA Titan X or Tesla K40 GPU.

#### 5.5.3.1 Permutation-invariant MNIST

In all permutation-invariant MNIST experiments (Section 5.3.2),  $D$ ,  $G$ , and  $E$  each consist of two hidden layers with 1024 units. The first hidden layer is followed by a non-linearity; the second is followed by (parameter-free) batch normalization (Ioffe

and Szegedy, 2015) and a non-linearity. The second hidden layer in each case is the input to a linear prediction layer of the appropriate size. In  $D$  and  $E$ , a leaky ReLU (Maas et al., 2013) non-linearity with a “leak” of 0.2 is used; in  $G$ , a standard ReLU non-linearity is used. All models are trained for 400 epochs.

### 5.5.3.2 ImageNet

In all ImageNet experiments (Section 5.3.3), the encoder  $E$  architecture follows AlexNet (Krizhevsky et al., 2012) through the fifth and last convolution layer (*conv5*), with local response normalization (LRN) layers removed and batch normalization (Ioffe and Szegedy, 2015) (including the learned scaling and bias) with leaky ReLU non-linearity applied to the output of each convolution at unsupervised training time. (For supervised evaluation, batch normalization is not used, and the pre-trained scale and bias is merged into the preceding convolution’s weights and bias.)

In most experiments, both the discriminator  $D$  and generator  $G$  architecture are those used by Radford et al. (2016), consisting of a series of four  $5 \times 5$  convolutions (or “deconvolutions” – fractionally-strided convolutions – for the generator  $G$ ) applied with 2 pixel stride, each followed by batch normalization and rectified non-linearity.

The sole exception is our discriminator baseline feature learning experiment, in which we let the discriminator  $D$  be the AlexNet variant described above. Generally, using AlexNet (or similar convnet architecture) as the discriminator  $D$  is detrimental to the visual fidelity of the resulting generated images, likely due to the relatively large convolutional filter kernel size applied to the input image, as well as the max-pooling layers, which explicitly discard information in the input. However, for fair comparison of the discriminator’s feature learning abilities with those of BiGANs, we use the same architecture as used in the BiGAN encoder.

**Preprocessing.** To produce a data sample  $\mathbf{x}$ , we first sample an image from the database, and resize it proportionally such that its shorter edge has a length of 72 pixels. Then, a  $64 \times 64$  crop is randomly selected from the resized image. The crop is flipped horizontally with probability  $\frac{1}{2}$ . Finally, the crop is scaled to  $[-1, 1]$ , giving the sample  $\mathbf{x}$ .

**Timing.** A single epoch (one training pass over the 1.2 million images) of BiGAN training takes roughly 40 minutes on a Titan X GPU. Models are trained for 100 epochs, for a total training time of under 3 days.

**Nearest neighbors.** In Figure 5.5 we present nearest neighbors in the feature space of the BiGAN encoder  $E$  learned in unsupervised ImageNet training.
























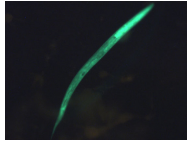






Query	#1	#2	#3	#4
				
				
				
				
				
				

Figure 5.5. For the query images used in Krähenbühl et al. (2016) (left), nearest neighbors (by minimum cosine distance) from the ImageNet LSVRC (Russakovsky et al., 2015) training set in the  $fc6$  feature space of the ImageNet-trained BiGAN encoder  $E$ . (The  $fc6$  weights are set randomly; this space is a random projection of the learned  $conv5$  feature space.)

# Chapter 6

## Conclusion

In the first few chapters of this thesis, we saw that feature representations learned by supervised convolutional networks generalize surprisingly well to related domains and tasks. Chapter 2 demonstrated state-of-the-art classification results in domains like scene recognition and fine-grained or subcategory recognition simply by pairing linear classifiers with features transferred from ImageNet-trained convnets. Chapter 3 paired pretrained convnet representations with region proposal mechanisms to address localization tasks, far exceeding the state of the art at the time in object detection and matching the state of the art in pixelwise semantic segmentation. Chapter 4 showed that convnets may be paired with recurrent networks (e.g., LSTMs) to accept sequential inputs or produce sequential outputs.

All of these results relied heavily on supervision from the 1.2 million labeled images of the ImageNet challenge database (Deng et al., 2012). In contrast, Chapter 5 proposed and evaluated an approach – Bidirectional GANs or BiGANs – for learning powerful visual feature representations *without* supervision, from unlabeled images alone. Although the representations learned by unsupervised approaches like BiGAN currently lag behind the fully supervised representations in quantitative evaluations, these approaches show significant promise as potential ways of exploiting the billions or trillions of images available “in the wild” from the internet and other sources to learn even richer and more generalizable visual feature representations than those learned from purely supervised approaches.

While much progress in both supervised and unsupervised representation learning has been made, there remain a number of important frontiers and directions for future research, some of which we’ll discuss below.

### 6.1 Frontiers and future directions

**Improving unsupervised generative models.** While there’s a great deal of excitement around GANs (Goodfellow et al., 2014), and justifiably so, their current

forms generally work best – i.e., synthesize images most convincingly – when applied to restricted domains (e.g., when trained on only scene or face images as in Radford et al. (2016), or only on dog images, as in Salimans et al. (2016)), or when trained conditionally with supervision to generate images of a particular class (Zhang et al., 2016). In contrast, when trained unconditionally on the entirety of the ImageNet 1000 class dataset, for example, the generated images produced by current GANs have little recognizable structure, mostly producing amorphous “blobs” rather than recognizable objects (see, e.g., Figure 5.4). Ian Goodfellow’s GAN tutorial (Goodfellow, 2016) provides a nice discussion of these and other issues with GAN training, as well as various recently proposed techniques to partially alleviate them.

Given the relationship of the generator and the encoder in a BiGAN, the inability of the generator to accurately represent the many modes of a large dataset like ImageNet also limits the encoder’s ability to learn visual semantics. For example, if the generator does not learn to associate a concept like “cat” with some direction in latent space, the encoder, which inverts the generator, will not learn to recognize cats in its inputs. As such, improving the BiGAN encoder’s ability to learn good feature representations from large and diverse datasets like ImageNet may require improvements to image generation.

Next we’ll discuss one potential direction for doing so.

### Generative models with discrete latent spaces.

In many domains of interest, particularly in the visual world, some of the underlying factors of variation may be *discrete* rather than continuous. For example, in distributions like MNIST or ImageNet, the latent factors for a given image include a discrete “label” (one of 10 or 1000). Despite this, the latent spaces typically used by practitioners to train GANs and other generative models are entirely continuous. For continuous latent variables  $\mathbf{z}$ , a continuous generator  $G(\mathbf{z})$  is forced to smoothly interpolate between the discrete modes of the data distribution, while still producing a sufficiently plausible image at every point in latent space between any two different modes. While this has proven empirically to be possible with a deep nonlinear  $G$ , at least for MNIST, this intuition suggests that we might be able to make the generator’s job easier by feeding it latent inputs  $\mathbf{z}$  with discrete components.

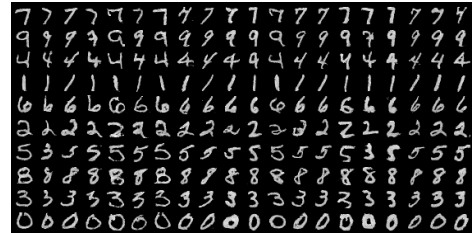


Figure 6.1. GANs and other generative models may be trained with discrete latent components.

As a proof of concept, Figure 6.1 shows the results of training a standard GAN on MNIST where latent inputs  $\mathbf{z}$  consist of a 10-way discrete categorical variable (i.e., a 10D one-hot vector) concatenated with a 10D continuous uniform variable. In each row, the outcome of the discrete categorical component is fixed; in each column, the outcome of the continuous component is fixed. The GAN, completely unsupervised,

learns a discrete clustering of the MNIST digits that exactly corresponds to our notion of digit identities, as well as learning to associate particular writing styles with the continuous components.<sup>1</sup>



Figure 6.2. Generated images from a GAN with discrete inputs trained on a subset of ImageNet.

Figure 6.2 shows the results of a similar preliminary experiment for the first 100 classes of ImageNet (mainly various bird and fish species), where the latent inputs to a GAN consist of a 100-way discrete categorical variable concatenated with a 100D continuous uniform variable. (Due to space constraints, only ten outcomes of the categorical variable are shown.) In this case, the categorical variable seems to select for both a foreground object type and background texture, somewhat failing to automatically do exactly what we “want” it to do – learn to use the categorical variable to select for object identity alone. The addition of more discrete components may help overcome this limitation – for example, with two categorical latent variables, the generator could associate one with selecting the background and the other with selecting the foreground.

In the case of MNIST, if the GAN generator with these discrete inputs were perfect, the “latent regressor” encoder we proposed earlier, trained to reconstruct  $\mathbf{z}$  given a generated image  $G(\mathbf{z})$ , would learn all the structure learned by fully supervised classifiers, as it needs to predict the categorical component of  $\mathbf{z}$  which the generator learned to associate with digit identity. Beyond that, by reconstructing the continuous components of  $\mathbf{z}$  the encoder learns to recognize handwriting style – an aspect not represented by the category label.

Training a BiGAN with discrete  $\mathbf{z}$  on the other hand is significantly more challenging. The encoder in this case is stochastic: its output  $E(\mathbf{x})$  includes a softmax component, parametrizing a categorical distribution, and

a sample  $\hat{\mathbf{z}} \sim E(\mathbf{x})$  includes a one-hot vector sampled from that distribution. This discrete sampling step is non-differentiable, and thus computing gradients of the discriminator score  $D(\mathbf{x}, \hat{\mathbf{z}})$  with respect to the encoder weights requires a method beyond standard backpropagation. One method for estimating such gradients is known as REINFORCE (Williams, 1992), which produces unbiased yet very high

<sup>1</sup>InfoGAN (Chen et al., 2016) includes a similar result, but found that a standard GAN fails to associate the categorical outcomes with digit identities. To obtain the results in Figures 6.1 & 6.2, we scale the one-hot vector representing the categorical component of  $\mathbf{z}$  by a factor  $\alpha > 1$  before it’s input to the generator, biasing the generator such that this component has greater influence.



variance gradient estimates. Preliminary attempts to train discrete BiGANs with this approach have proven difficult and unstable. (We similarly struggled with other methods like the “straight-through” estimator (Bengio et al., 2013) and Gumbel-Softmax reparameterization (Jang et al., 2016), recently proposed as alternatives to REINFORCE which are biased, but may have lower variance.)

Despite the additional modeling and optimization challenges discrete latent variables bring, they nonetheless intuitively seem a potentially important direction for improving unsupervised representation learning.

**Semi-supervised learning.** In Chapter 5 we stubbornly confined ourselves to representation learning in the purely unsupervised regime to analyze the extent to which rich representations can be learned from data alone. But this restriction makes little sense in many important real world settings. For example, as we saw in the first chapters of this thesis, if our goal is to use a convnet to make semantic predictions in a natural image domain, pre-training the convnet to predict ImageNet labels seems to be a nearly universally beneficial – or at least non-detrimental – first step. In general, whenever we have access to a supervisory signal for a domain of interest, we may as well make use of it.

Hence an important next step to demonstrate the value of these approaches in more realistic settings is to evaluate in the large-scale *semi-supervised* setting, where a model is jointly trained with an unsupervised objective on a large unlabeled database, as well as a supervised objective for any labeled samples we have access to. In the case of BiGAN, for example, the encoder predicting latent representations  $\mathbf{z}$  could also be trained to predict any available labels  $y$  with a standard softmax classifier.

Unfortunately, current unsupervised learning approaches are unlikely to improve learned visual representations given a sufficiently large labeled dataset like ImageNet. However, once unsupervised learning approaches mature (e.g., by improving generation as suggested in previous paragraphs), demonstrating benefits in the semi-supervised setting will be key for adoption of these approaches outside the ivory tower.

**Unsupervised learning for reinforcement learning.** The goal of *reinforcement learning* (RL) is to train an agent to act in response to observations from an environment, maximizing rewards accumulated over time. Reinforcement learning is, in general, thought to be more challenging than supervised learning for a variety of reasons, most of which we won’t delve into here. Among them, though, is the sparsity of typical reward signals – e.g., the reward may be zero at every step until a particular complex sequence of actions has been taken – and the resulting difficulty of representation learning, especially for high dimensional state spaces such as natural images.

A recent body of work (Dosovitskiy and Koltun, 2017; Jaderberg et al., 2017; Shelhamer et al., 2016) has shown that optimizing the agent’s policy or value network



(which takes raw states or observations as input) for auxiliary (self-)supervised or unsupervised tasks improves the agent’s ability to maximize RL returns, at least in terms of time to convergence or data efficiency. For example, Shelhamer et al. (2016) showed that pretraining an Atari agent’s convnet with unsupervised or self-supervised objectives (e.g., as a BiGAN encoder) improves convergence speed and data efficiency for many Atari games. While BiGAN does not improve convergence to the extent that the self-supervised reward or (inverse) dynamics tasks do, the self-supervised tasks are complementary to BiGAN: as described in the previous section (*semi-supervised learning*) for semantic image labels, the BiGAN encoder could also be jointly trained for prediction per the self-supervised tasks, potentially with the generator and discriminator conditioned on other components of a  $(s, a, r, s')$  transition tuple as well. Furthermore, the “joint policy and auxiliary optimization” results from Shelhamer et al. (2016) show that rather than merely pre-training for the auxiliary tasks on transitions collected from a randomly initialized agent, data efficiency can be further improved by training the RL agent jointly to optimize reward as well as an auxiliary objective. With joint training, unlike pre-training, the inputs to the auxiliary objective continue to match the (non-stationary) distribution of the agent’s observations, even as its policy is updated.

A complementary research direction in RL known as *intrinsic motivation* (Chentanez et al., 2004; Schmidhuber, 1991; Stadie et al., 2015) ponders how agents can be incentivized to be “curious” – i.e., to learn to explore the full state space. This topic is especially important in the face of sparse reward signals or high-dimensional observations. Unsupervised learning and generative modeling approaches may also prove useful in this area. For example, if a generative model can be used to obtain density estimates  $p(\mathbf{x})$  for observations  $\mathbf{x}$ , the negative of these (log-)density estimates  $-p(\mathbf{x})$  or  $-\log p(\mathbf{x})$  could be added to the standard reward signal as “exploration bonuses,” incentivizing the agent to search for low probability states. *PixelCNN* (van den Oord et al., 2016) is a recent generative model that produces such density estimates directly. The BiGAN encoder can also be used to obtain density estimates, though in a less straightforward manner.

Overall, existing evidence and intuition suggest that unsupervised learning approaches may be important tools for making reinforcement learning with sparse reward signals or high-dimensional observations more tractable.

**Multistep prediction and attention.** Since *AlexNet* (Krizhevsky et al., 2012), deep convnet architectures have been improved with a number of new techniques and design choices like batch normalization (Ioffe and Szegedy, 2015) and residual connections (He et al., 2016). These new techniques, as well as improvements in GPU hardware and low-level software, have made it possible to train larger and deeper convnets with superior recognition capabilities, as measured by accuracy on ImageNet classification and other standard vision benchmarks.

However, all of these techniques process the entirety of the input image in a

single feed-forward inference step. To the extent that vision systems found in nature can serve as inspiration for improvements in computer vision systems, this suggests we might be able to do better with a multistep prediction approach; for example using a learned *attention* mechanism where a model predicts not only what it’s looking at, but also *where* to look, potentially with multiple “glimpses” at the image.

Attention mechanisms have been explored for both recognition (Mnih et al., 2014; Ranzato, 2014) and generation (Gregor et al., 2015) in small-scale settings, but haven’t been demonstrated for high-resolution images of interest to modern computer vision researchers. Despite the current lack of empirical evidence, it is clear that for a fixed computational budget such approaches have the potential to outperform purely feed-forward networks in any domain, by allowing for more expensive computations to be performed on smaller regions of the input.

This capability may prove especially useful for localization problems such as detection and pixelwise semantic segmentation, where challenges remain with small objects and fine object boundaries. Taking segmentation as an example, a network with attention capabilities could first make a “coarse” segmentation prediction at low resolution, and then “zoom in” on regions near object boundaries, iteratively refining its predictions.

Of course, learning an attention mechanism brings additional challenges not suffered with purely feed-forward prediction. In particular, a straightforward supervised learning problem turns into a much more difficult reinforcement learning problem, depending on how the problem is formulated. This suggests that improving upon purely feed-forward models with attention, even with a restricted computational budget, is unlikely to be straightforward. Nonetheless, the potential benefits of attention mechanisms for both computational efficiency and recognition accuracy could make overcoming these barriers a worthwhile pursuit in the long run.

# Bibliography

- P. Agrawal, J. Carreira, and J. Malik, “Learning to see by moving,” in *ICCV*, 2015.
- B. Alexe, T. Deselaers, and V. Ferrari, “Measuring the objectness of image windows,” *TPAMI*, 2012.
- R. Ando and T. Zhang, “A framework for learning predictive structures from multiple tasks and unlabeled data,” *JMLR*, 2005.
- P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik, “Multiscale combinatorial grouping,” in *CVPR*, 2014.
- P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik, “Semantic segmentation using regions and parts,” in *CVPR*, 2012.
- A. Argyriou, T. Evgeniou, and M. Pontil, “Multi-task feature learning,” in *NIPS*, 2006.
- M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, “Action classification in soccer videos with long short-term memory recurrent neural networks,” in *ICANN*, 2010.
- , “Sequential deep learning for human action recognition,” in *Human Behavior Understanding*, 2011.
- D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *ICLR*, 2015.
- S. Banerjee and A. Lavie, “METEOR: An automatic metric for MT evaluation with improved correlation with human judgments,” in *ACL Workshops*, 2005.
- A. Barbu, A. Bridge, Z. Burchill, D. Coroian, S. Dickinson, S. Fidler, A. Michaux, S. Mussman, S. Narayanaswamy, D. Salvi, L. Schmidt, J. Shangquan, J. M. Siskind, J. Waggoner, S. Wang, J. Wei, Y. Yin, and Z. Zhang, “Video in sentences out,” in *UAI*, 2012.
- H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” in *ECCV*, 2006.

- Y. Bengio, N. Léonard, and A. Courville, “Estimating or propagating gradients through stochastic neurons for conditional computation,” *arXiv:1308.3432*, 2013.
- T. Berg and P. Belhumeur, “POOF: Part-based one-vs-one features for fine-grained categorization, face verification, and attribute estimation,” in *CVPR*, 2013.
- L. Bo, X. Ren, and D. Fox, “Kernel descriptors for visual recognition,” in *NIPS*, 2010.
- L. Bourdev, S. Maji, and J. Malik, “Describing people: A poselet-based approach to attribute classification,” in *ICCV*, 2011.
- T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, “High accuracy optical flow estimation based on a theory for warping,” in *ECCV*, 2004.
- J. Carreira and C. Sminchisescu, “CPMC: Automatic object segmentation using constrained parametric min-cuts,” *TPAMI*, 2012.
- J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, “Semantic segmentation with second-order pooling,” in *ECCV*, 2012.
- R. Caruana, “Multitask learning,” *Machine Learning*, 1997.
- X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets,” in *NIPS*, 2016.
- N. Chentanez, A. G. Barto, and S. P. Singh, “Intrinsically motivated reinforcement learning,” in *NIPS*, 2004.
- K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” in *SSST Workshop*, 2014.
- K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” in *EMNLP*, 2014.
- S. Chopra, S. Balakrishnan, and R. Gopalan, “DLID: Deep learning for domain adaptation by interpolating between domains,” in *ICML Workshops*, 2013.
- D. Cireşan, A. Giusti, L. Gambardella, and J. Schmidhuber, “Mitosis detection in breast cancer histology images with deep neural networks,” in *MICCAI*, 2013.
- N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *CVPR*, 2005.

- P. Das, C. Xu, R. Doell, and J. Corso, “Thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching,” in *CVPR*, 2013.
- H. Daume III, “Frustratingly easy domain adaptation,” in *ACL*, 2007.
- T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, “Fast, accurate detection of 100,000 object classes on a single machine,” in *CVPR*, 2013.
- J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F.-F. Li, “ImageNet: A large-scale hierarchical image database,” in *CVPR*, 2009.
- J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and F.-F. Li, “ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012),” <http://www.image-net.org/challenges/LSVRC/2012/>, 2012.
- E. L. Denton, S. Chintala, A. Szlam, and R. Fergus, “Deep generative image models using a Laplacian pyramid of adversarial networks,” in *NIPS*, 2015.
- J. Devlin, H. Cheng, H. Fang, S. Gupta, L. Deng, X. He, G. Zweig, and M. Mitchell, “Language models for image captioning: The quirks and what works,” in *ACL*, 2015.
- J. Devlin, S. Gupta, R. Girshick, M. Mitchell, and C. L. Zitnick, “Exploring nearest neighbor approaches for image captioning,” *arXiv:1505.04467*, 2015.
- C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” in *ICCV*, 2015.
- J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “DeCAF: A deep convolutional activation feature for generic visual recognition,” in *ICML*, 2014.
- J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” in *CVPR*, 2015.
- J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” *TPAMI*, 2016.
- J. Donahue, P. Krähenbühl, and T. Darrell, “Adversarial feature learning,” in *ICLR*, 2017.
- A. Dosovitskiy and V. Koltun, “Learning to act by predicting the future,” in *ICLR*, 2017.

- M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid, “Evaluation of GIST descriptors for web-scale image search,” in *CIVR*, 2009.
- V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville, “Adversarially learned inference,” in *ICLR*, 2016.
- I. Endres and D. Hoiem, “Category independent object proposals,” in *ECCV*, 2010.
- M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes (VOC) Challenge,” *IJCV*, 2010.
- M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes challenge: A retrospective,” *IJCV*, 2014.
- H. Fang, S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. Platt, C. L. Zitnick, and G. Zweig, “From captions to visual concepts and back,” in *CVPR*, 2015.
- C. Farabet, C. Couprie, L. Najman, and Y. LeCun, “Learning hierarchical features for scene labeling,” *TPAMI*, 2013.
- A. Farhadi, M. Hejrati, M. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth, “Every picture tells a story: Generating sentences from images,” in *ECCV*, 2010.
- P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *TPAMI*, 2010.
- S. Fidler and A. Leonardis, “Towards scalable representations of object categories: Learning a hierarchy of parts,” in *CVPR*, 2007.
- S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun, “Bottom-up segmentation for top-down detection,” in *CVPR*, 2013.
- A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov, “DeViSE: A deep visual-semantic embedding model,” in *NIPS*, 2013.
- K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, 1980.
- R. Girshick, P. Felzenszwalb, and D. McAllester, “Discriminatively trained deformable part models, release 5,” <http://www.cs.berkeley.edu/~rbg/latent-v5/>.
- R. Girshick, “Fast R-CNN,” in *ICCV*, 2015.

- R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *CVPR*, 2014.
- , “Region-based convolutional networks for accurate object detection and segmentation,” *TPAMI*, 2016.
- B. Gong, Y. Shi, F. Sha, and K. Grauman, “Geodesic flow kernel for unsupervised domain adaptation,” in *CVPR*, 2012.
- I. Goodfellow, “NIPS 2016 tutorial: Generative adversarial networks,” *arXiv:1701.00160*, 2016.
- I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “Maxout networks,” in *ICML*, 2013.
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *NIPS*, 2014.
- A. Graves, “Generating sequences with recurrent neural networks,” *arXiv:1308.0850*, 2013.
- A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *ICML*, 2014.
- A. Graves, A.-r. Mohamed, and G. E. Hinton, “Speech recognition with deep recurrent neural networks,” in *ICASSP*, 2013.
- K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, “DRAW: A recurrent neural network for image generation,” in *ICML*, 2015.
- C. Gu, J. J. Lim, P. Arbeláez, and J. Malik, “Recognition using regions,” in *CVPR*, 2009.
- S. Guadarrama, N. Krishnamoorthy, G. Malkarnenkar, S. Venugopalan, R. Mooney, T. Darrell, and K. Saenko, “YouTube2Text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition,” in *ICCV*, 2013.
- B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, “Semantic contours from inverse detectors,” in *ICCV*, 2011.
- K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.
- L. A. Hendricks, S. Venugopalan, M. Rohrbach, R. Mooney, K. Saenko, and T. Darrell, “Deep compositional captioning: Describing novel object categories without paired training data,” in *CVPR*, 2016.

- G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv:1207.0580*, 2012.
- G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, 2006.
- G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, 2006.
- S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.
- M. Hodosh, P. Young, and J. Hockenmaier, "Framing image description as a ranking task: Data, models and evaluation metrics," *JAIR*, 2013.
- J. Hoffman, E. Rodner, J. Donahue, K. Saenko, and T. Darrell, "Efficient learning of domain-invariant image representations," in *ICLR*, 2013.
- D. Hoiem, Y. Chodpathumwan, and Q. Dai, "Diagnosing error in object detectors," in *ECCV*, 2012.
- R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell, "Natural language object retrieval," in *CVPR*, 2016.
- S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.
- M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," in *ICLR*, 2017.
- E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv:1611.01144*, 2016.
- K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *ICCV*, 2009.
- S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *TPAMI*, 2013.
- Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv:1408.5093*, 2014.
- A. Karpathy and F.-F. Li, "Deep visual-semantic alignments for generating image descriptions," in *CVPR*, 2015.



- A. Karpathy, A. Joulin, and F.-F. Li, “Deep fragment embeddings for bidirectional image sentence mapping,” in *NIPS*, 2014.
- A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F.-F. Li, “Large-scale video classification with convolutional neural networks,” in *CVPR*, 2014.
- L. Kennedy and A. Hauptmann, “LSCOM lexicon definitions and annotations (version 1.0),” *IEEE Multimedia*, 2006.
- M. U. G. Khan, L. Zhang, and Y. Gotoh, “Human focused video description,” in *ICCV Workshops*, 2011.
- D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.
- R. Kiros, R. R. Salakhutdinov, and R. S. Zemel, “Multimodal neural language models,” in *ICML*, 2014.
- , “Unifying visual-semantic embeddings with multimodal neural language models,” in *TACL*, 2015.
- P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, “Moses: Open source toolkit for statistical machine translation,” in *ACL*, 2007.
- P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell, “Data-dependent initializations of convolutional neural networks,” in *ICLR*, 2016.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *NIPS*, 2012.
- H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “HMDB: a large video database for human motion recognition,” in *ICCV*, 2011.
- B. Kulis, K. Saenko, and T. Darrell, “What you saw is not what you get: Domain adaptation using asymmetric kernel transforms,” in *CVPR*, 2011.
- G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg, “Baby talk: Understanding and generating simple image descriptions,” in *CVPR*, 2011.
- P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi, “Collective generation of natural image descriptions,” in *ACL*, 2012.
- P. Kuznetsova, V. Ordonez, T. L. Berg, U. C. Hill, and Y. Choi, “TreeTalk: Composition and compression of trees for image descriptions,” in *TACL*, 2014.

- Q. V. Le, W. Zou, S. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR*, 2011.
- Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Y. Ng, "Building high-level features using large scale unsupervised learning," in *ICML*, 2012.
- Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, 1989.
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, 1998.
- F.-F. Li, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories," in *CVPR*, 2004.
- L. Li, H. Su, F.-F. Li, and E. Xing, "Object bank: A high-level image representation for scene classification & semantic feature sparsification," in *NIPS*, 2010.
- J. J. Lim, C. L. Zitnick, and P. Dollár, "Sketch tokens: A learned mid-level representation for contour and object detection," in *CVPR*, 2013.
- C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *ACL Workshops*, 2004.
- T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *arXiv:1405.0312*, 2014.
- J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.
- D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.
- A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, 2013.
- J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, "Learning like a child: Fast novel visual concept learning from sentence descriptions of images," in *ICCV*, 2015.
- J. Mao, W. Xu, Y. Yang, J. Wang, and A. Yuille, "Deep captioning with multimodal recurrent neural networks (m-RNN)," in *ICLR*, 2015.

- G. Mesnil, Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley, P. Vincent, A. Courville, and J. Bergstra, “Unsupervised and transfer learning challenge: a deep learning approach.” *JMLR*, 2012.
- M. Mitchell, X. Han, J. Dodge, A. Mensch, A. Goyal, A. Berg, K. Yamaguchi, T. Berg, K. Stratos, and H. Daumé III, “Midge: Generating image descriptions from computer vision detections,” in *EACL*, 2012.
- V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, “Recurrent models of visual attention,” in *NIPS*, 2014.
- J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *CVPR*, 2015.
- M. Noroozi and P. Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” in *ECCV*, 2016.
- A. Oliva and A. Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope,” *IJCV*, 2001.
- K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a method for automatic evaluation of machine translation,” in *ACL*, 2002.
- D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by inpainting,” in *CVPR*, 2016.
- A. Quattoni, M. Collins, and T. Darrell, “Transfer learning for image classification with sparse prototype representations,” in *CVPR*, 2008.
- A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in *ICLR*, 2016.
- R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, “Self-taught learning: Transfer learning from unlabeled data,” in *ICML*, 2007.
- M. Ranzato, “On learning where to look,” *arXiv:1405.5488*, 2014.
- A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN features off-the-shelf: an astounding baseline for recognition,” in *CVPR Workshops*, 2014.
- X. Ren and D. Ramanan, “Histograms of sparse codes for object detection,” in *CVPR*, 2013.
- A. Rohrbach, M. Rohrbach, W. Qiu, A. Friedrich, M. Pinkal, and B. Schiele, “Coherent multi-sentence video description with variable level of detail,” in *GCPR*, 2014.

- A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele, “Grounding of textual phrases in images by reconstruction,” *arXiv:1511.03745*, 2015.
- M. Rohrbach, W. Qiu, I. Titov, S. Thater, M. Pinkal, and B. Schiele, “Translating video content to natural language descriptions,” in *ICCV*, 2013.
- H. A. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *TPAMI*, 1998.
- D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” DTIC Document, Tech. Rep., 1985.
- O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and F.-F. Li, “ImageNet large scale visual recognition challenge,” *IJCV*, 2015.
- K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual category models to new domains,” in *ECCV*, 2010.
- H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao, “Sequence discriminative distributed training of long short-term memory recurrent neural networks,” in *Interspeech*, 2014.
- R. R. Salakhutdinov and G. E. Hinton, “Deep Boltzmann machines,” in *AISTATS*, 2009.
- T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” in *NIPS*, 2016.
- J. Schmidhuber, “Curious model-building control systems,” in *IJCNN*, 1991.
- P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “OverFeat: Integrated recognition, localization and detection using convolutional networks,” *arXiv:1312.6229*, 2013.
- P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, “Pedestrian detection with unsupervised multi-stage feature learning,” in *CVPR*, 2013.
- E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell, “Loss is its own reward: Self-supervision for reinforcement learning,” *arXiv:1612.07307*, 2016.
- K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *NIPS*, 2014.
- , “Very deep convolutional networks for large-scale image recognition,” in *ICLR*, 2015.

- S. Singh, A. Gupta, and A. A. Efros, “Unsupervised discovery of mid-level discriminative patches,” in *ECCV*, 2012.
- R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng, “Grounded compositional semantics for finding and describing images with sentences,” in *TACL*, 2014.
- K. Soomro, A. R. Zamir, and M. Shah, “UCF101: A dataset of 101 human actions classes from videos in the wild,” CRCV-TR-12-01, Tech. Rep., 2012.
- B. C. Stadie, S. Levine, and P. Abbeel, “Incentivizing exploration in reinforcement learning with deep predictive models,” *arXiv:1507.00814*, 2015.
- K. Sung and T. Poggio, “Example-based learning for view-based human face detection,” Massachusetts Institute of Technology, Tech. Rep. A.I. Memo No. 1521, 1994.
- I. Sutskever, J. Martens, and G. E. Hinton, “Generating text with recurrent neural networks,” in *ICML*, 2011.
- I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *NIPS*, 2014.
- C. Szegedy, A. Toshev, and D. Erhan, “Deep neural networks for object detection,” in *NIPS*, 2013.
- C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *CVPR*, 2015.
- C. C. Tan, Y.-G. Jiang, and C.-W. Ngo, “Towards textually describing complex video contents with audio-visual concept classifiers,” in *ACM MM*, 2011.
- Theano Development Team, “Theano: A Python framework for fast computation of mathematical expressions,” *arXiv:1605.02688*, 2016.
- J. Thomason, S. Venugopalan, S. Guadarrama, K. Saenko, and R. J. Mooney, “Integrating language and vision to generate natural language descriptions of videos in the wild,” in *TACL*, 2014.
- S. Thrun, “Is learning the  $n$ -th thing any easier than learning the first?” in *NIPS*, 1996.
- A. Torralba and A. A. Efros, “Unbiased look at dataset bias,” in *CVPR*, 2011.
- L. Torresani, M. Szummer, and A. Fitzgibbon, “Efficient object category recognition using classemes,” in *ECCV*, 2010.

- J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, “Selective search for object recognition,” *IJCV*, 2013.
- R. Vaillant, C. Monrocq, and Y. LeCun, “Original approach for the localisation of objects in images,” *IEE Proc on Vision, Image, and Signal Processing*, 1994.
- A. van den Oord, N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves, “Conditional image generation with PixelCNN decoders,” in *NIPS*, 2016.
- L. van der Maaten and G. E. Hinton, “Visualizing data using t-SNE,” *JMLR*, 2008.
- R. Vedantam, C. L. Zitnick, and D. Parikh, “CIDEr: Consensus-based image description evaluation,” in *CVPR*, 2015.
- S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, “Sequence to sequence–video to text,” in *ICCV*, 2015.
- S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, “Translating videos to natural language using deep recurrent neural networks,” in *NAACL*, 2015.
- O. Vinyals, S. V. Ravuri, and D. Povey, “Revisiting recurrent neural networks for robust ASR,” in *ICASSP*, 2012.
- O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. E. Hinton, “Grammar as a foreign language,” in *NIPS*, 2015.
- O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” in *CVPR*, 2015.
- H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *ICCV*, 2013.
- H. Wang, A. Kläser, C. Schmid, and C. Liu, “Dense trajectories and motion boundary descriptors for action recognition,” *IJCV*, 2013.
- J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, “Locality-constrained linear coding for image classification,” in *CVPR*, 2010.
- X. Wang and A. Gupta, “Unsupervised learning of visual representations using videos,” in *ICCV*, 2015.
- X. Wang, M. Yang, S. Zhu, and Y. Lin, “Regionlets for generic object detection,” in *ICCV*, 2013.

- P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, “Caltech-UCSD Birds 200,” California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010.
- R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning*, 1992.
- R. J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks,” in *Neural Computation*, 1989.
- J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, “SUN database: Large-scale scene recognition from abbey to zoo,” in *CVPR*, 2010.
- K. Xu, J. Ba, R. Kiros, A. Courville, R. R. Salakhutdinov, R. S. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *ICML*, 2015.
- J. Yang, Y. L., Y. Tian, L. Duan, and W. Gao, “Group-sensitive multiple kernel learning for object categorization,” in *ICCV*, 2009.
- Y. Yang, C. L. Teo, H. Daumé III, and Y. Aloimonos, “Corpus-guided sentence generation of natural images,” in *EMNLP*, 2011.
- L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, “Describing videos by exploiting temporal structure,” in *CVPR*, 2015.
- S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and F.-F. Li, “Every moment counts: Dense detailed labeling of actions in complex videos,” *arXiv:1507.05738*, 2015.
- P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, “From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions,” *TACL*, 2014.
- W. Zaremba and I. Sutskever, “Learning to execute,” *arXiv:1410.4615*, 2014.
- M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *ECCV*, 2014.
- M. D. Zeiler, G. Taylor, and R. Fergus, “Adaptive deconvolutional networks for mid and high level feature learning,” in *CVPR*, 2011.
- H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas, “StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks,” *arXiv:1612.03242*, 2016.

- N. Zhang, R. Farrell, F. Iandola, and T. Darrell, “Deformable part descriptors for fine-grained recognition and attribute prediction,” in *ICCV*, 2013.
- L. Zhu, Y. Chen, and A. Yuille, “Unsupervised learning of a probabilistic grammar for object detection and parsing,” in *NIPS*, 2007.