Learning Generalizable Visual Patterns Without Human Supervision

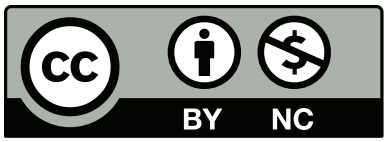
Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Simon JENNI

von Bangerten BE

Leiter der Arbeit:
Prof. Dr. Paolo FAVARO
Institut für Informatik



Learning Generalizable Visual Patterns Without Human Supervision

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Simon JENNI

von Bangerten BE

Leiter der Arbeit:
Prof. Dr. Paolo FAVARO
Institut für Informatik

Von der Philosophisch-naturwissenschaftlichen Fakultät angenommen.

Bern, 23.06.2021

Der Dekan
Prof. Dr. Z. Balogh

Abstract

Learning Generalizable Visual Patterns Without Human Supervision

Simon JENNI, Ph.D. in Computer Science

Universität Bern, 2021

Owing to the existence of large labeled datasets, Deep Convolutional Neural Networks have ushered in a renaissance in computer vision. However, almost all of the visual data we generate daily - several human lives worth of it - remains unlabeled and thus out of reach of today's dominant supervised learning paradigm. This thesis focuses on techniques that steer deep models towards learning generalizable visual patterns without human supervision. Our primary tool in this endeavor is the design of Self-Supervised Learning tasks, i.e., pretext-tasks for which labels do not involve human labor. Besides enabling the learning from large amounts of unlabeled data, we demonstrate how self-supervision can capture relevant patterns that supervised learning largely misses. For example, we design learning tasks that learn deep representations capturing shape from images, motion from video, and 3D pose features from multi-view data. Notably, these tasks' design follows a common principle: The recognition of data transformations. The strong performance of the learned representations on downstream vision tasks such as classification, segmentation, action recognition, or pose estimation validate this pretext-task design.

This thesis also explores the use of Generative Adversarial Networks (GANs) for unsupervised representation learning. Besides leveraging generative adversarial learning to define image transformation for self-supervised learning tasks, we also address training instabilities of GANs through the use of noise.

While unsupervised techniques can significantly reduce the burden of supervision, in the end, we still rely on some annotated examples to fine-tune learned representations towards a target task. To improve the learning from scarce or noisy labels, we describe a supervised learning algorithm with improved generalization in these challenging settings.

Acknowledgements

Foremost, I would like to thank my Ph.D. advisor Paolo Favaro. I am very grateful for the opportunities and the mentorship he provided me over the last few years. His BSc course on Machine Learning sparked my initial interest in the field, and the work during my MSc thesis under his supervision inspired a passion for research that led me to pursue a Ph.D. I am thankful for his outstanding support and advice, numerous inspiring discussions, and the freedoms he provided over the last four years. But most of all, I thank him for motivating and challenging me to go further than I thought I could.

I want to express my thanks to Phillip Isola and David Bommes for serving as thesis examiners. I greatly admire Phillip's work, so it was an immense pleasure to hear that he agreed to serve as an examiner.

A great thank you goes out to all the members of the Computer Vision Group in Bern I had the pleasure to work with: Givi Meishvili, Attila Szabó, Mehdi Noroozi, Meiguang Jin, Qiyang Hu, Xiaochen Wang, Adrian Wälchli, Adam Bielski, Abdelhak Lemkhenter, Josué Page, Tomoki Watanabe, Riccardo Fantinel, Florence Aellen, and the new members Llukman Çerkezi, Aram Davtyan, Sepehr Sameni, which I regrettably did not get to know as well due to COVID-19. I very much enjoyed the work atmosphere in the group and the fun times spent outside of work, be it during lunch breaks, the occasional ski-trips, social events, refreshing swims in the river during summertime, or the occasional after-work happy-hour. A special thank you goes to Givi for being an all-around great colleague, collaborator, and friend.

Much of the smooth operation and excellent working environment at Bern is due to the exceptional work of both our office manager Dragana Esser and the system admins: Peppo Brambilla and Adrian Wälchli. I am very thankful for their support.

Thanks also to Hailin Jin at Adobe Research, with whom I collaborated during an internship at the beginning of 2021. I look very much forward to starting my industrial research career at Adobe.

None of this would have been possible without the support of my family and friends. A Ph.D. can be quite tough and stressful at times, but quality time spent with family and friends provides the necessary balance and perspective to push through such obstacles. A special thank goes to my father Erwin for his continued financial support, being patient with me, giving me the freedom to explore many different career paths, and being a great father and role model.

Last but not least, I would like to thank my fiancée Céline for her emotional support, for proofreading this thesis, and for making the last four years of my life infinitely more enjoyable.

Contents

Abstract **v**

Acknowledgements **vii**

1 Introduction **1**
1.1 Limitations of Supervised Representation Learning . . . 2
1.2 Self-Supervision: Learning Without Human Annotations . . . 3
1.3 Learning Generalizable Visual Patterns . . . 4
1.4 Thesis Contributions . . . 4
1.4.1 Chapter Outline . . . 5

2 Background **7**
2.1 Autoencoders . . . 7
2.2 Generative Adversarial Learning . . . 8
2.3 Self-Supervised Learning via Proxy Tasks . . . 9
2.3.1 Methods for Learning Image Representations . . . 10
2.3.2 Methods for Learning Video Representations . . . 11
2.4 Contrastive Representation Learning . . . 12

3 Learning Visual Patterns by Damaging Them **13**
3.1 Background . . . 15
3.2 Architecture Overview . . . 15
3.3 Learning to Spot Artifacts . . . 16
3.3.1 The Damage & Repair Network . . . 16
3.3.2 Replication of Real Images . . . 18
3.3.3 Training the Discriminator . . . 19
3.3.4 Implementation . . . 19
3.4 Experiments . . . 20
3.4.1 Ablation Analysis . . . 20
3.4.2 Transfer Learning Experiments . . . 21
Classification on STL-10 . . . 22
Classification, Detection and Segmentation on PASCAL VOC . 22
Layerwise Performance on ImageNet & Places . . . 23
3.4.3 Qualitative Analysis of the Features . . . 23
3.5 Discussion . . . 24

4 Learning by Recognizing Image Transformations **29**
4.1 Background . . . 30
4.2 Learning Features by Discriminating Global Image Transformations . . 32
4.2.1 Limited Context Inpainting . . . 33
4.2.2 Random Warping . . . 34
4.2.3 Image Rotations . . . 34
4.2.4 Preventing Degenerate Learning . . . 35

4.3 On the Choice of Transformations . . . 35
4.4 Experiments . . . 36
4.4.1 Ablation Experiments . . . 36
Limited Context Inpainting . . . 36
Combination of Image Transformations . . . 38
4.4.2 Unsupervised Feature Learning Benchmarks . . . 38
4.5 Discussion . . . 42

5 Learning Motion via Temporal Transformations 45
5.1 Background . . . 47
5.2 Learning Video Dynamics . . . 47
5.2.1 Transformations of Time . . . 48
5.2.2 Training . . . 49
5.2.3 Implementation . . . 50
5.3 Experiments . . . 51
5.4 Discussion . . . 57

6 Self-Supervised Learning of 3D Pose Features 59
6.1 Background . . . 61
6.2 Unsupervised Learning of 3D Pose-Discriminative Features . . . 62
6.2.1 Classifying Synchronized and Flipped Views . . . 62
6.2.2 Static Backgrounds Introduce Shortcuts . . . 64
6.2.3 Implementation . . . 64
6.2.4 Transfer to 3D Human Pose Estimation . . . 65
6.3 Experiments . . . 65
6.3.1 Ablations . . . 65
6.3.2 Comparison to Prior Work . . . 68
6.3.3 Evaluation of the Synchronization Task . . . 68
6.4 Discussion . . . 69

7 Stabilizing Generative Adversarial Training With Random Noise 73
7.1 Background . . . 76
7.2 Matching Filtered Distributions . . . 76
7.2.1 Formulation . . . 77
7.2.2 Implementation . . . 78
7.2.3 Batch-Normalization and Mode Collapse . . . 78
7.3 Experiments . . . 79
7.3.1 Ablations . . . 80
7.3.2 Application to Different GAN Models . . . 81
7.3.3 Robustness to Hyperparameters . . . 81
7.3.4 Qualitative Results . . . 83
7.4 Discussion . . . 85

8 Generalizing from Limited and Noisy Labels 87
8.1 Background . . . 88
8.2 Learning to Generalize . . . 90
8.2.1 Bilevel Learning . . . 90
8.2.2 A Proximal Formulation . . . 91
8.3 Implementation . . . 92
8.4 Experiments . . . 93
8.4.1 Ablations . . . 93

8.4.2 Fitting Random Pixel Permutations 96
8.4.3 Memorization of Partially Corrupted Labels 96
8.4.4 Generalization on Small Datasets 98
8.5 Discussion 99

9 Conclusions **101**

Bibliography **103**

List of Figures

1.1 Problems with learning from sparse human labels. . . . 2
1.2 Recurring patterns in natural visual data. . . . 3

3.1 Images with and without artifacts. . . . 14
3.2 Overview of the model architecture. . . . 15
3.3 Influence of the repair network. . . . 17
3.4 Encoder design. . . . 18
3.5 The repair block. . . . 20
3.6 Renderings with the damage and repair network. . . . 22
3.7 Visualization of the learned features. . . . 27
3.8 Nearest-neighbor retrievals on the ImageNet validation set. . . . 28
3.9 Grad-CAM visualization. . . . 28

4.1 The importance of global image statistics. . . . 30
4.2 Learning global statistics. . . . 31
4.3 Selected image transformations. . . . 32
4.4 Training of the Limited Context Inpainting (LCI) network. . . . 33
4.5 Image statistics on CelebA. . . . 35
4.6 Nearest-neighbor accuarcy on Places. . . . 42
4.7 Comparison of nearest neighbor retrieval. . . . 43

5.1 Learning from temporal transformations. . . . 46
5.2 Training a 3D-CNN to distinguish temporal transformations. . . . 48
5.3 Time-related pseudo-tasks. . . . 53
5.4 Visualization of active pixels. . . . 54
5.5 Examples of retrievals in UCF101. . . . 56

6.1 Monocular 3D human pose estimation. . . . 60
6.2 An overview of the proposed self-supervised task. . . . 63
6.3 Examples of predictions. . . . 69
6.4 Synchronization performance on unseen test subjects. . . . 70
6.5 Synchronization retrievals on test subjects. . . . 70
6.6 Generalization across subjects. . . . 71

7.1 Addressing the limited support of the data distribution in GANs. . . . 74
7.2 Simplified scheme of the proposed GAN training. . . . 75
7.3 Wall clock time vs IS for GANs with and without distribution filtering. 78
7.4 How separate normalization of fake and real mini-batches discourages mode collapse. . . . 79
7.5 Comparison of reconstructions with and without distribution filtering. 82
7.6 Results of robustness experiments on CIFAR-10. . . . 83
7.7 Examples of the generated noise patterns. . . . 84
7.8 Reconstructions from DFGANs trained on 128×128 images. . . . 84

8.1 The training procedure of our bilevel formulation. 88
8.2 Ablation experiments on CIFAR-10. 95
8.3 CifarNet on CIFAR-10 and CIFAR-100 with varying label noise. 97
8.4 Inception network on CIFAR-10 and CIFAR-100 with varying label noise. 98
8.5 Classification on varying fractions of Pascal VOC. 99

List of Tables

3.1 Ablation experiments on STL-10. 21
3.2 Comparison to prior works on STL-10. 23
3.3 Transfer learning experiments on Pascal VOC. 24
3.4 Linear classifier experiments on ImageNet. 25
3.5 Linear classifier experiments on Places. 26

4.1 Ablation experiments for Limited Context Inpainting (LCI) on STL-10. 37
4.2 Combinations of image transformations on STL-10. 37
4.3 Combinations of image transformations on CelebA. 38
4.4 Transfer learning experiments on PASCAL VOC. 39
4.5 Linear classifier experiments on ImageNet. 40
4.6 Linear classifier experiments on Places. 41
4.7 Comparison to prior works on STL-10. 41

5.1 Ablation experiments. 50
5.2 Comparison to prior work on self-supervised video representation learning. 52
5.3 Time-related pseudo-tasks. 53
5.4 Video retrieval performance on UCF101. 56

6.1 Ablation experiments. 66
6.2 Comparison to prior work. 67

7.1 Ablation experiments on CIFAR-10 and STL-10. 80
7.2 Network architectures used for experiments on CIFAR-10 and STL-10 . 81
7.3 Application to established GAN models on CIFAR-10 and CelebA. . . 82
7.4 Hyperparameter settings for the robustness experiments. 83

8.1 Results of ablation experiments on CIFAR-10. 96
8.2 Inception network trained on data with random pixel permutations. . 96
8.3 Comparison to prior works. 97
8.4 Experiments with more realistic label noise on ImageNet. 98

Chapter 1

Introduction

> *"They think that intelligence is about noticing things are relevant (detecting patterns); in a complex world, intelligence consists in ignoring things that are irrelevant (avoiding false patterns)"*
>
> — Nassim Nicholas Taleb

Applications of computer vision can increasingly be found in our everyday lives. Phones unlock automatically by recognizing our faces. We can search our photo libraries for pictures of our pets without ever labeling them as such. The mail delivered to our doorstep relies on an algorithm to decipher handwritten addresses. Cars now observe their surroundings and brake automatically in case we run the risk of hitting an obstacle. Algorithms might analyze the X-ray images from a recent visit to the doctor, possibly finding patterns a human expert would miss.

Vision systems need to recognize patterns in visual data accurately to solve any of those tasks. Nowadays, computer vision relies predominantly on deep Convolutional Neural Networks (CNNs), a model loosely inspired by the visual system in the brain, to learn visual representations. Through the hierarchical composition of many simple building blocks (artificial neurons), CNNs become capable of pattern recognition at a level even surpassing humans in some cases. However, with the ability to find patterns comes the danger of discovering false patterns. This is especially true for supervised learning - today's dominant paradigm - where correlations between high-dimensional inputs and sparse targets can be plentiful. As a result, current methods rely on large quantities of human-annotated examples and regularization to prevent overfitting and to generalize well to unseen examples. Indeed, much of the recent success of computer vision can be attributed to the availability of large-scale human-labeled datasets and effective training and regularization strategies.

Collecting tens of thousands of human-annotated examples for any given vision application is not feasible. How come deep learning has seen widespread adoption nonetheless? A key feature of deep learning is its ability to transfer knowledge acquired through learning some task to a novel task of interest. This so-called "transfer learning" can significantly reduce the demand for human supervision on a downstream vision task (*e.g.*, object detection or segmentation). Therefore, it is the current best practice to fine-tune the parameters of a network that was pre-trained on large-scale object recognition datasets for most vision problems. Transfer learning works reasonably well when the learning tasks and data distributions are aligned between pre-training and transfer learning. However, can supervised learning actually make use of web-scale visual data for learning? What if the relevance of visual patterns differs significantly between pre-training and transfer tasks?

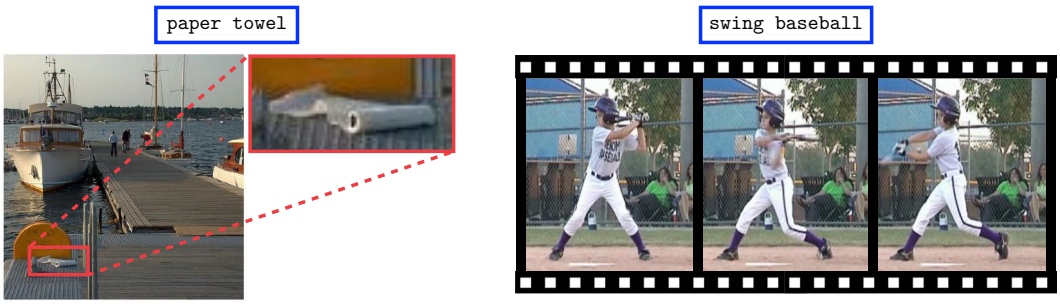


FIGURE 1.1: **Problems with learning from sparse human labels.** We show training examples from ImageNet (a) and HMDB51 (b) along with their class labels above. A single label does not capture the complexities of images and videos well. This can lead to ambiguities (a) and a failure to capture relevant patterns, *e.g.*, motion in (b).

1.1 Limitations of Supervised Representation Learning

Besides its success, supervised representation learning has some severe limitations. Gathering human annotations does not scale to the massive amounts of visual data we produce every day. For example, it is estimated that around 500 hours of video are uploaded to YouTube every minute.[1] However, even sparse human annotations become prohibitive on billions of images (both in terms of time spent and cost). Therefore, a finite number of labeled training examples might not represent all the relevant discriminative visual patterns. This is especially true for rarely occurring instances, *i.e.*, examples in the long tail of the data distribution. Furthermore, for data and tasks that differ significantly from pre-training, knowledge transfer is very limited. This is the case for many vision applications of interest, *e.g.*, in medicine, multi-camera systems in autonomous driving, or satellite imagery. Given how data-hungry deep neural networks are, are medical doctors destined to labeling millions of X-ray images for machines to learn from?

Another less obvious issue is the lack of control over what visual patterns supervised networks capture when predicting a single label from its input (see Figure 1.1). To what degree, for example, does object classification rely on texture, shape, or scene composition? How vital are motion patterns to action classification? All these features are relevant, but evidence suggests that texture dominates in image classification [1] and appearance likewise in action recognition [2]. However, an over-reliance on a single feature can lead to poor generalization. What if, in downstream tasks, a network should recognize different poses of objects with identical appearance? What if a network that relies on correlations between foreground and background textures observes an object in a novel environment at test time? Other visual patterns such as shape, motion, and scene composition could disambiguate these cases but might not be captured well through supervised learning.

As a result of the shortcomings, the knowledge captured through supervised classification tasks might be too shallow for the increasingly more complex vision tasks we aim to tackle. Indeed, humans do not appear to rely predominantly on supervised learning for visual learning, so why should we expect supervised classification tasks to produce human-level visual understanding.

[1] Source: https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/

(a) (b) (c)

FIGURE 1.2: **Recurring patterns in natural visual data.** Our visual world is full of recurring patterns, symmetries, and structure. Similar texture patterns can be found inside, and across images (a), shape and pose patterns are shared among different classes (b), and patterns of scene composition appear with regularity (c). Through self-supervised learning, we aim to exploit and learn from this natural structure.

1.2 Self-Supervision: Learning Without Human Annotations

Self-Supervised Learning (SSL) presents a viable alternative to supervised representation learning. In SSL, learning tasks are designed for which supervision can be generated automatically without human labor. Provided these tasks can learn relevant visual patterns, SSL can leverage arbitrarily large unlabeled datasets, solving supervised learning's scaling issues. Can we also hope to gain more control over what features are learned through SSL? If supervision is not provided through human annotators, what sources of learning signals are there?

Our visual world is very structured (see Figure 1.2); texture patterns reappear in the same image and among images of similar categories. Likewise, we can observe similar shape, motion, and scene composition patterns regularly all around us. This natural structure can be exploited through SSL by designing learning tasks for which the solution requires an understanding of these recurring patterns. For example, a scene's spatial structure could be captured by learning to arrange several image patches correctly [3], [4]. Provided the right task design, SSL can capture any regular visual pattern, even those not well captured through supervised learning.

How can we then design SSL tasks to learn specific task-relevant visual features (*e.g.*, motion and pose features for action recognition)? At present, there is a limited understanding of what characterizes effective pretext task designs and how the task design influences the learned representations' properties. It appears to be more of an art, with task designs based on intuitions and trial and error. This lack of design principles makes it difficult to predict if an SSL task will produce features that perform well on a given data domain and vision problem.

1.3 Learning Generalizable Visual Patterns

In the context of supervised learning, generalization refers to the ability of a learning algorithm to learn task-relevant patterns that perform well also on unseen data while ignoring patterns that do not. Since increasing the amount of training data is often not feasible, this is primarily achieved through regularization, *i.e.*, by limiting the learning algorithms' capacity. The design of CNNs and the methodology of transfer learning can be seen as a form of regularization: they introduce inductive biases in the network structure and network weights, respectively. While deep CNNs generalize surprisingly well given their high capacity, overfitting remains an issue, especially in challenging situations where examples are scarce, or the labels are noisy.

While it is well understood what it means for a representation to generalize in the supervised setting, its corresponding meaning in the unsupervised learning setting is less clear. Unsupervised methods - by definition - can use any available amount of data, making overfitting to the training data a non-issue. Nonetheless, the learned features can still generalize poorly to vision tasks *other* than the pre-training task. Rather than overfitting to data, we thus run the risk of overfitting to learning tasks in SSL. Special care must be taken in the task design to avoid trivial solutions, so-called "shortcuts", where networks exploit unintended patterns in the data. These are often low-level, non-semantic features that networks converge to easily but are not useful in general. Therefore, the challenge is to design learning tasks that push models towards learning patterns relevant to a range of downstream vision problems.

Aside from SSL, generative adversarial learning has become a popular approach for unsupervised representation learning. Generative Adversarial Networks (GANs) are capable of photo-realistic image synthesis and found applications in various vision problems, including SSL tasks. The training of GANs is difficult, however, as it is susceptible to poor hyper-parameter choices. As a result, GANs often cannot generalize to the complete data distribution, *i.e.*, they cannot cover all the modes of the data distribution. The training instability stems from non-overlapping supports of the data and generator distributions. This, in turn, results in overconfident discriminators and either vanishing or exploding loss gradients, depending on the GAN objective.

1.4 Thesis Contributions

In this thesis, we study techniques to steer deep models towards learning generalizable visual data patterns, even without human supervision. The primary focus of this study lies in the design of self-supervised learning tasks. Besides enabling learning on vast amounts of unlabeled data, we demonstrate how SSL can steer deep models towards learning specific task-relevant patterns. For example, we develop SSL tasks that produce representations with induced biases towards shape, motion, and 3D pose features. Importantly, we base the design of these pretext tasks on a shared pattern: The recognition of data transformations. This design provides control over what patterns should be learned by manipulating selected visual data patterns through chosen transformations. We demonstrate this methodology on natural images, videos, and multiview data. The learned representations are evaluated on various vision tasks, including classification, detection, segmentation, action recognition, and 3D human pose estimation.

The thesis further addresses issues of generalization in supervised learning and the training of GANs. Generative adversarial training is an integral part of the proposed SSL tasks on images, where we use it to define image transformations. We address the training instability and improve the generalization of GANs through the use of noise. To address supervised learning issues in cases where training data is scarce, or labels are noisy, we describe a training algorithm that encapsulates the principle of cross-validation in every parameter update.

1.4.1 Chapter Outline

Chapter 2: Background. We provide a general overview of prior works in unsupervised visual representation learning, the central theme of this thesis. Chapter 2 discusses autoencoders, GANs, SSL via proxy tasks on images and videos, and recent advances based on the contrastive learning framework. More discussions of prior works specific to a given chapter are given in separate sections of the remaining chapters.

Chapter 3: Learning Visual Patterns by Damaging Them. In classic SSL tasks, some part of the data serves as a self-supervision signal. Some, for example, predict the spatial arrangement from image tiles or color channels from gray-scale images. In Chapter 3 we introduce a novel approach where we purposefully introduce non-trivial defects in images and train a neural network to recognize and localize them. Image defects are produced by corrupting an autoencoder's latent codes and then locally repairing them via adversarial training. The resulting images exhibit semantic defects such as missing object parts. By recognizing such defects, CNNs learn a representation that captures object semantics.

Chapter 4: Learning by Recognizing Image Transformations. Why do tasks that recognize image rotations or corruptions learn good representations? We investigate this question in Chapter 4. We observe that recognizing such data transformations is often not possible from local image statistics (e.g., textures) and requires global statistics (e.g., shape). Following this insight, we design image transformations that preserve local but distinctly alter global image statistics. Our experiments suggest that CNNs trained to recognize such transformations capture shape better than supervised networks.

Chapter 5: Learning Motion via Temporal Transformations. Supervised learning on video via action recognition is biased towards appearance because actions are often recognizable from a single frame. As a result, motion is not well represented, although it is essential to video understanding. In Chapter 5, we propose as SSL task the recognition of alterations of the natural frame order and changes of the playback speed. Since frame appearance is preserved and only motion is altered, recognizing such temporal transformations induces a bias towards motion in the learned representation. Representations with such a motion bias generalize well to action recognition and temporal reasoning tasks (*e.g.*, the ordering or the synchronization of frame sequences).

Chapter 6: Self-Supervised Learning of 3D Pose Features. SSL is most useful in situations where annotations are costly. One such example is 3D human pose estimation, where motion capture systems provide ground-truth. In Chapter 6 we describe an SSL task that exploits multi-view synchronized video data to learn 3D

pose features. As a task, we learn to recognize if two views of a scene represent a rigid or a non-rigid transformation of the 3D geometry. Rigid transformations can be obtained from synchronized views, whereas non-rigid ones from unsynchronized or flipped views. The learned representations generalize well to monocular 3D human pose estimation with few labeled examples.

Chapter 7: Stabilizing Generative Adversarial Training With Random Noise. Generative adversarial learning has proven helpful in many vision problems, including SSL. However, the training of GANs is highly sensitive to hyperparameters (*e.g.*, network architectures and learning rates). We explore the use of noise to stabilize GAN training in Chapter 7. In our method, both real and generated examples are augmented by adding samples of varying noise distributions. The result is a filtering of the underlying distributions. This reduces the instability by extending the supports of data and generator distributions. The method consistently improves standard GAN models and provides increased robustness to poor hyperparameter settings.

Chapter 8: Generalizing from Limited and Noisy Labels. Models pre-trained using SSL can transfer well, even with few labeled examples to downstream tasks. However, this supervised transfer can still lead to poor performance due to overfitting when labels are scarce or noisy. In Chapter 8 we describe a learning algorithm that exhibits better generalization than standard Stochastic Gradient Descent (SGD) in these circumstances. We introduce weights on training mini-batches in our approach and optimize those weights to minimize the loss on separate validation batches. The resulting training algorithm demonstrates increased robustness to label noise and better generalization on small labeled datasets.

The list of publications associated with each chapter:

1. Chapter 3 - "Self-supervised feature learning by learning to spot artifacts" [5], in CVPR 2018.
2. Chapter 4 - "Steering self-supervised feature learning beyond local pixel statistics" [6], in CVPR 2020.
3. Chapter 5 - "Video representation learning by recognizing temporal transformations" [7], in ECCV 2020.
4. Chapter 6 - "Self-supervised multi-view synchronization learning for 3d pose estimation" [8], in ACCV 2020.
5. Chapter 7 - "On stabilizing generative adversarial training with noise" [9], in CVPR 2019.
6. Chapter 8 - "Deep bilevel learning" [10], in ECCV 2018.

Chapter 2

Background

"If you would understand anything, observe its beginning and its development."

— Aristotle

This chapter provides an overview of prior works in unsupervised representation learning, which is the primary focus of this thesis. Before fully-supervised deep representation learning became feasible with the introduction of AlexNet [11], unsupervised pre-training was the de-facto standard approach for training deep neural networks. In fact, the unsupervised layer-wise pre-training in Deep Belief Nets (DBN) [12] sparked the renewed interest in neural networks. Each layer in a DBN is composed of a Restricted Boltzmann Machine (RBM) [13], each being trained in sequence (hence the term layer-wise pre-training). Efficient training of such stacks of RBMs was made possible through contrastive divergence learning [14]. Such unsupervised, layer-wise pre-training is not commonly used today since several optimization techniques have made the end-to-end training of deep networks feasible.

We will limit our further discussion of unsupervised techniques to deep learning approaches that remain in common usage today. After a brief review of autoencoders (an example of a classic unsupervised learning approach), we will focus our attention on generative adversarial learning, self-supervised learning via proxy tasks, and finally, discuss some recent contrastive learning approaches.

2.1 Autoencoders

The autoencoder is a classic model for unsupervised representation learning [15]. An autoencoder consists of two parts: an encoder and a decoder. Together these are tasked to faithfully reconstruct the training data, with reconstruction quality typically measured via mean-squared-error. Trivial learning (*e.g.*, by learning the identity function) is prevented by introducing constraints on the encoding, *e.g.*, by limiting the encodings' dimensionality. Such bottlenecks drive the learning of general input patterns (the structure of the data) in order to preserve as much information as possible. The simplest autoencoder consists of a single dense layer as the encoder and a single dense layer as the decoder. This model is, in fact, equivalent to classic Principal Component Analysis (PCA) if the hidden layer does not contain a non-linearity and the model is trained by minimizing the squared reconstruction error. Naturally, deep autoencoders (also called stacked autoencoders) with multiple layers learn better representations. Similar to DBNs, such deep autoencoders can be trained layer-wise [16]. For vision tasks, convolutional layers would typically be used [17].

There exist multiple extensions of the basic autoencoder model, two notable examples being the denoising autoencoder (DAE) [18] and the variational autoencoder (VAE) [19]. In a denoising autoencoder, the input to the encoder is corrupted (*e.g.*, by adding random noise), and the autoencoder is trained to reconstruct the uncorrupted input. In VAEs, the hidden state of the model is assumed to be multivariate Gaussian with a diagonal covariance matrix. Furthermore, the hidden state distribution is pushed towards a standard Normal prior via a KL-Divergence loss term. As a result, VAEs are generative models that allow a random sampling of the learned data distribution (by decoding samples of a standard Normal). It turns out that representations learned through VAEs can capture and disentangle the factors of variation reasonably well on some datasets [20]. Numerous extensions and variations of VAEs have been proposed. For example, a variant with discrete hidden states [21], and an approach using adversarial training to enforce the prior on the hidden variables [22].

Remark. We will often use the term autoencoder to refer to a general network design pattern (*i.e.*, encoder-decoder architectures) that is common in image restoration and translation tasks [23]. The goal in these cases is not typically to learn a good representation but rather to achieve some specific image processing.

2.2 Generative Adversarial Learning

The introduction of the generative adversarial network (GAN) model by Goodfellow *et al.* [24] sparked many exciting new developments in computer vision. The original GAN is an unsupervised generative model consisting of two components: 1) a generator network that generates samples similar to the training data from random noise (typically Standard Normal noise), and 2) a discriminator network that critiques samples from the generator on how similar they are to the training data. These two networks are trained in an adversarial two-player game, at the equilibrium of which the generator should faithfully model the data distribution.

Adversarial Feature Learning. Radford *et al.* [25] introduced a convolutional GAN model that allowed training at higher resolution and resulted in much-improved sample quality. Besides showing that GANs learn useful discriminative features in the discriminator network, they also showed that the latent generator representation has very desirable properties. For example, interpolations in the learned latent space result in seamless, natural interpolations in image space. Furthermore, the representation even allows for semantically meaningful arithmetic operations analogous to what has been observed for word representations in natural language processing [26]. Directions in the latent space thus tend to separate factors of variation. To leverage this representation, Donahue *et al.* [27] train a network to learn the inverse mapping of the generator. Learning this inverse mapping also improves training and mode coverage, resulting in state-of-the-art unconditional image generation on ImageNet [28].

Learnable Loss Functions. The principles of adversarial learning have found widespread adoption in image restoration and translation tasks. It turns out that the discriminator can act as a "learnable" loss function, quantifying the perceptual dissimilarity to some reference distribution. In this way, adversarial learning has been applied to classic image restoration tasks to enforce image realism, *e.g.*, in super-resolution [29], deblurring [30], or image inpainting [31]. More importantly,

this principle can also be applied to image translation tasks, *i.e.*, to translate corresponding images between two domains (sketch-to-image as an example). This methodology was first proposed on paired data [23], *i.e.*, when corresponding images in the two domains are available. Through cycle-consistency constraints, this idea can also be extended to the unpaired case [32].

Addressing Training Difficulties. While GANs have been shown to produce very high-quality samples, they are notoriously difficult to train. The community thus focused on addressing GAN training's inherent instability. Early works introduced a set of techniques and heuristics [33], or proposed improved architectural designs and hyper-parameter settings [25]. For example, [33] proposed using one-sided label smoothing and the injection of Gaussian noise into the layers of the discriminator. Arjovsky *et al.* [34] then provided a theoretical analysis of the unstable training and the vanishing gradients phenomena. They argued that the primary source of instability stems from the fact that the real and the generated distributions have disjoint supports or lie on low-dimensional manifolds. In the case of an optimal discriminator, this will result in zero gradients that stop the generator's training. They also proposed a way to reduce such difficulties by introducing noise.

Besides heuristics, a considerable number of alternative GAN objectives were proposed to address training difficulties. [35] built on the work of [34] and introduced the Wasserstein GAN (WGAN). The WGAN optimizes an integral probability metric that is dual to the Wasserstein distance. This formulation requires the discriminator to be Lipschitz-continuous, which is realized through weight-clipping. In [36] a better way to enforce the Lipschitz constraint via a gradient penalty over interpolations between real and generated data was introduced (WGAN-GP). These improvements enabled the usage of deep residual networks in GAN architectures, which helped further boost image quality. Several other loss functions and GAN models have been proposed, claiming superior stability and sample quality over a vanilla GAN (*e.g.*, [37], [38], [39], [40]). However, large-scale studies [41] found no clear empirical evidence in favor of alternative GAN models in terms of robustness to hyper-parameter settings and peak performance across different datasets.

Instead of novel GAN objectives, some authors focused on general regularization techniques. [42] introduced a stabilizing regularizer based on a gradient norm penalty similar to the approach by [36]. Another popular GAN regularization technique that bounds the Lipschitz constant of the discriminator is the spectral normalization introduced by [43]. This technique allowed the training of very large network architectures on ImageNet [44].

Another line of work promotes a progressive training of GANs, where the model architecture and its output resolution grow during training [45], [46]. This approach proved especially effective for very high-resolution image generation.

2.3 Self-Supervised Learning via Proxy Tasks

In this section, we discuss self-supervised learning approaches. These are unsupervised feature learning methods that rely on either naturally occurring or artificially introduced supervision as a means to learn visual representations. Self-supervised learning appeared in the machine learning literature more than two decades ago [47], [48]. We will first discuss methods for learning image representations in Section 2.3.1 and then discuss video representation learning methods in Section 2.3.2.

2.3.1 Methods for Learning Image Representations

Self-supervised learning methods on images have arguably garnered the majority of attention in the literature. Consequently, numerous sources of free supervision have been explored. We provide an overview of some of the most popular types of learning signals below.

Methods Exploiting Spatial Structure. The pioneering work by Doersch *et al.* [3] proposed the task of relative patch localization. A Siamese network architecture is fed two image patches and has to recognize their relative location among nine possible classes. [4] generalized this approach to a 3×3 grid of image patches by training a similar Siamese architecture to solve jigsaw puzzles. Several variations of these patch-based methods can be found (*e.g.*, [49], [50], [51]).

Another approach to learning image representations based on the spatial structure is to separate an image patch from its context (*i.e.*, the image region surrounding the patch) and train a network to predict the patch appearance from its context [31]. Image representations are learned in the context encoder network, which is trained using an adversarial loss. [52] showed that the discriminator in such an inpainting task also learns good image representations.

Spatial structure can also be exploited through consistency constraints as demonstrated in [53] where supposed visual primitives are counted. These counts are then assumed to be consistent, *i.e.*, the count on a large patch is assumed to be equal (or less than) the sum of counts on sub-patches.

A simple yet surprisingly effective learning signal is the orientation of an image, which has been used in the rotation prediction task of [54]. They train a network to recognize multiples of 90° rotations. This task exploits the photographer bias, *i.e.*, our tendency to capture images in an upright orientation that reflects how we observe the world most of the time. Much of this approach's effectiveness can also be attributed to the lack of shortcuts (low-level data artifacts that give away the correct label), which plague many other tasks [3], [4], [53].

Methods Exploiting Color. Aside from spatial self-supervision, color has also been a popular learning signal. The colorization of grey-scale images - an interesting vision problem by itself - has been shown to produce valuable features [55]. This task is posed as the prediction of color channels given the lightness channel for images represented in LAB color space. Because the learned representation can only access part of the data (only the color channels), [56] proposed to additionally predict the inverse mapping (lightness from color) and fusing the two learned representations. Further improvements on color proxy tasks are proposed in [57].

An interesting modification of this task can be found in [58], where the task is to colorize a grey-scale video given a single colored reference frame. It turns out that networks learn to track objects across multiple frames by solving this task.

Methods Exploiting Video. Videos provide unique possibilities for learning image representations. Natural changes in an object's appearance and pose occurring over time can be leveraged for representation learning. One approach is via tracking image-patches and then using matching (*i.e.*, tracked) image-patches for metric learning [59]. Three patches are used as input, where two patches are matched via tracking, and the third one is arbitrarily chosen. [60] extended this idea to using more than one negative example with the contrastive learning framework.

The approach by [61] used motion extracted from video clips to segment images and then used the resulting segmentation masks as the supervisory signal.

Methods Exploiting Audio. Another popular source of natural supervision can be found in the sound accompanying video frames. [62] trained a network to predict summary statistics of the sounds corresponding to a video frame. [63], [64] proposed to learn visual and aural representations jointly by training a two-stream architecture to recognize if an audio snippet and image correspond (*i.e.*, they are temporally aligned). They also demonstrated how the learned representations can be used to perform tasks like sound source localization. Similar ideas were explored in [65], [66].

Naturally, the above-listed works and learning signals are not exhaustive. There exist, for example, some methods that assume an interactive environment and exploit an agent's interaction with the environment for self-supervised learning [67], [68]. A detailed discussion of methods that exploit knowledge of unique training instances is deferred to Section 2.4. These methods learn to tell apart different training examples while learning invariance to data augmentations [69], [70]. Clustering methods [50], [71], [72] work in a similar fashion but aim at learning additional invariances among groups of examples (with groups formed based on the similarity in the evolving representation space). Several works also proposed to use combinations of the above learning signals, *e.g.*, in a multi-task learning setting [73], [74], [75].

2.3.2 Methods for Learning Video Representations

The self-supervised learning of video representations has attracted much attention in the vision community in recent years. Videos are attractive for SSL because human annotations are even more costly than on images, *e.g.*, action labels typically also require to specify the temporal extent of the action in the video. Dense annotations for localization tasks across multiple frames quickly become prohibitive, and methods that can leverage large amounts of unlabeled video for learning become very attractive. Besides the potential practical benefits, the added temporal dimension in videos also provides novel opportunities for self-supervised learning signals.

Several works have adapted SSL methods from domains such as images or natural language to videos, *e.g.*, rotation prediction [76], dense predictive coding [77], and language models [78]. However, here we are more interested in methods that were explicitly designed for videos and thus use temporal supervision or leverage auxiliary signals accompanying a video (*e.g.*, optical flow or audio).

The temporal ordering of video frames provides natural supervisory signals. For example, Misra *et al.* [79] proposed learning to distinguish a natural order of frames from a shuffled one. This idea has been extended to longer sequences for posture and behavior analysis using LSTM architectures [80]. The above approaches classify the correctness of a temporal order directly from one sequence. An alternative is to feed several sequences, some of which are modified, and ask the network to tell them apart [81]. Other works predict the permutation of a sequence of frames [82], or both the spatial and temporal ordering of frame patches [49], [83]. Rather than considering arbitrary reorderings of frames, [84] showed that solely predicting if a video is played forward or backward learns strong representations. Similarly, the prediction of playback speed was shown to be a good self-supervision signal [85], [86], [87].

Supervision can also come from additional signals obtained along with the video frames. For example, videos also come with audio. Similar to the image-based approaches [63], [64], Korbar *et al.* [88] learned audio and video representations by learning to synchronize audio and video signals. Other methods used optical flow to generate supervision signals. For example, [89] predicted motion and appearance statistics, [90] predicted future atomic 3D flows given an input sequence, and [91] used geometry in the form of flow fields and disparity maps on synthetic and 3D movies.

2.4 Contrastive Representation Learning

Current state-of-the-art methods in unsupervised image representation learning are primarily based on contrastive learning. The origin of these methods can be traced-back to instance discrimination tasks [69]. The goal is to recognize each distinct training example while also being invariant to common data augmentations. Similar to the supervised training of image classifiers, the set of data augmentations (*e.g.*, random cropping, horizontal flipping, or color-jittering) define desired invariances in the learned representation. These learned invariances likely explain much of the excellent performance in transfer to classification. However, properties of the contrastive loss function and the embedding space are also crucial [92].

Wu *et al.* [70] proposed a non-parametric formulation of this task based on a noise-contrastive estimation that allowed scaling instance discrimination to a large number of instances. [93] then suggested several improvements regarding architecture designs, stronger augmentations, and instance discrimination among large mini-batches. Some methods lessen the need for large batches by sampling negatives from a queue of past momentum-encoded examples [94]. Others remove the need for explicit negatives altogether [95], [96]. Several recent works proposed a contrastive learning beyond instance-level discrimination, *i.e.*, they learn a grouping or clustering of examples using the contrastive framework [97], [98]. While the learning of transformation invariance is fundamental to contrastve learning, [99] showed that it could be beneficial also to learn distinctiveness to image transformations depending on the downstream tasks.

Several authors have also explored the use of contrastive learning on videos. Some propose a natural extension of the above image-based methods to videos by adding spatially consistent temporal cropping to the set of augmentations [100]. Others proposed to treat temporally augmented clips as distinct instances [101], [102] to learn features sensitive to temporal input patterns (*i.e.*, video dynamics). The combination of contrastive learning with other SSL tasks via multi-task learning was also explored [103], [104], [105].

Another exciting line of work considers multi-modal data to perform contrastive learning across different modalities or "views". Such views can, for example, come from depth and surface normals [106]. On videos, [107] proposed a joint contrastive training between the raw RGB frames and corresponding optical flow. Other approaches rely on weak supervision in the form of text [108] or exploit the audio accompanying the video [109], [110]. Related methods extract different "views" of the data from different intermediate layers of the network [111], [112].

Chapter 3

Learning Visual Patterns by Damaging Them

"When patterns are broken, new worlds emerge."

— Tuli Kupferberg

Self-supervised learning methods have emerged as a viable alternative to the supervised learning paradigm [3], [4], [31], [55], [59]. These methods typically design a *pretext task* through the following strategy: one withholds some information about the input data and then trains a network to recover it. For example, some methods withhold image regions [31], color [55] or both grayscale and color values [56]; others withhold the location of patches [3], [4], or external information such as egomotion [113].

However, these task designs have their shortcomings. For one, by withholding part of the data, networks trained on these tasks do not have full access to all the information present in the data. Networks trained on a colorization task will thus not be able to exploit color information. Likewise, networks trained on patch-based learning tasks remain restricted to features at the patch level. Furthermore, these pretext tasks are often ambiguous as the desired mappings are one-to-many. A given gray-scale image, for example, can have many corresponding colorized versions. Similarly, a patch in the center of an image can be in-painted in several plausible ways, each solution varying in the exact textures and shapes used.

This chapter describes a novel self-supervised approach to learn features by classifying images as "real" or "with artifacts". As artifacts, we consider a non-trivial damaging of natural visual patterns, which could, for example, result in the removal of object parts or the distortion of shape and texture (see Figure 3.1). Note that all the information in real images is preserved in this task, and the objective is an unambiguous binary classification. We aim to create image artifacts such that a model capable of spotting them would require an accurate representation of objects and thus build features that could transfer well to tasks like object classification, detection, and segmentation. One approach to creating artifacts could be to use classic inpainting algorithms [114]. Besides being computationally inefficient on large inpainting regions, these methods are unsuitable for training because they may introduce low-level statistics that a neural network could quickly learn to detect. This would seriously limit what the network learns about objects. Therefore, instead of editing images at the pixel level, we tamper with their feature representation and

Material from: S. Jenni, and P. Favaro. "Self-Supervised Feature Learning by Learning to Spot Artifacts." In *IEEE Conference on Computer Vision and Pattern Recognition* 2018. © 2018 IEEE

FIGURE 3.1: **Images with and without artifacts.** We show a mixture of real images (green border) and images with synthetic artifacts (red border). Is a good object representation necessary to tell them apart?

create corrupt images so that the texture is locally unnoticeable but globally incorrect, as illustrated in Figure 3.1.

To generate artifacts, we first train an autoencoder to reproduce images. Then, we randomly drop entries from the encoded feature (at the bottleneck) so that information about the input image is lost. We then add a *repair* neural network to the decoder to help it render a realistic appearance. The repair network in-paints the feature representations at every layer of the decoder, but its limited capacity can not fully recover the missing information. In this way, a network cannot detect image artifacts through local analysis. We then train a discriminator to distinguish real from corrupt images.

Moreover, we also train the discriminator to predict a mask, which indicates the dropped feature entries. This mask prediction implicitly helps the discriminator focus on the essential details in the image. We also use the dropped entries' true mask to make sure that the repair network operates only on the features corresponding to the dropped entries. The repair network and the discriminator then learn in an adversarial fashion. However, in contrast to other adversarial schemes, we make sure that the repair network cannot confuse the discriminator. Furthermore, we transfer features from the discriminator since this is the model that learns an approximation of the distribution of images.

Contributions. Our contributions can be summarized as follows: 1) we introduce a novel feature learning framework based on detecting images with artifacts, which does not require human annotation; 2) we introduce a method to create images with non-trivial artifacts; 3) we demonstrate that these features achieve state-of-the-art performance on several transfer learning evaluations (ILSVRC2012 [115], Pascal VOC [116] and STL-10 [117]).

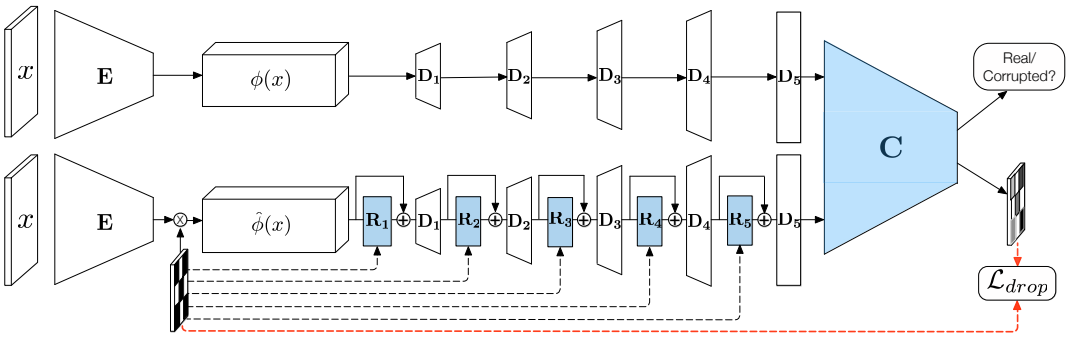


FIGURE 3.2: **Overview of the model architecture.** Two autoencoders $\{E, D_1, D_2, D_3, D_4, D_5\}$ output either real images (top row) or images with artifacts (bottom row). A discriminator C learns to distinguish them. The corrupted images are generated by masking the encoded feature $\phi(\mathbf{x})$ and then by using a repair network $\{R_1, R_2, R_3, R_4, R_5\}$ distributed across the layers of the decoder. The mask also restricts the repair network to change only the dropped entries of the feature (see Figure 3.3 for more details). The discriminator and the repair network (both shaded in blue) train in an adversarial fashion on the real/corrupt classification loss. The discriminator is also trained to output the mask used to drop feature entries so that it learns to localize all artifacts.

3.1 Background

This chapter relates to several machine learning topics: adversarial training, autoencoders, and self-supervised learning. We briefly discuss relevant prior works fusing these concepts here and refer to Chapter 2 for a broader exposition.

Our approach leverages adversarial training for self-supervised feature learning. Donahue *et al.* [27] also fuse those two techniques by extending the GAN model with an encoder network that learns the inverse mapping of the generator. They also show that transferring the discriminator features in the standard GAN setting leads to significantly worse performance. In contrast, the method we propose shows that by limiting the generator network's capabilities to local alterations, the discriminator network can learn significantly better visual representations.

The work of Pathak *et al.* [31] also shows similarities, as it combines autoencoders with an adversarial loss for the proxy task of inpainting (*i.e.*, to generate the content in an image region given its context). They, however, aim to learn image representations in the encoder of the autoencoder performing the inpainting. In contrast, our model's autoencoder component is solely used for training data generation and not for representation learning.

3.2 Architecture Overview

We briefly summarize the components of the architecture that we introduce in the next sections. Let $\mathbf{x}$ be a training image from a dataset and $\hat{\mathbf{x}}$ be its version with artifacts. As a model, we use a neural network consisting of the following components (see also Figure 3.2)

1. Two *autoencoder* networks $\{E, D_1, D_2, D_3, D_4, D_5\}$, where E is the encoder and $D = \{D_1, D_2, D_3, D_4, D_5\}$ is the decoder, pre-trained to reproduce high-fidelity real images $\mathbf{x}$; the output of the encoder E on an image $\mathbf{x}$ is denoted $\phi(\mathbf{x})$;

2. A spatial mask Ω to be applied to the feature output of E; the resulting masked feature is denoted $\hat{\phi}(\mathbf{x}) = \Omega \odot \phi(\mathbf{x}) + (1 - \Omega) \odot (u * \phi(\mathbf{x}))$, where u is some uniform spatial kernel so that $u * \phi(\mathbf{x})$ is a feature average, and $\odot$ denotes the element-wise product in the spatial domain (the mask is replicated along the channels);

3. A *discriminator* network C that learns to classify images $\mathbf{x}$ as real and images $\hat{\mathbf{x}}$ as fake (with artifacts); we also train the discriminator to output the mask Ω, so that it learns to localize all artifacts;

4. A *repair* network $\{R_1, R_2, R_3, R_4, R_5\}$ that attaches to the decoder layers of one of the autoencoder networks; the output of any layer R_i is masked by Ω so that it affects only masked features.

The repair network and the discriminator learn in an adversarial fashion on the real/corrupt classification loss.

3.3 Learning to Spot Artifacts

Our main objective is to train a classifying network (the discriminator) to learn an accurate distribution of real images. Prior work [25] showed that a discriminator trained to distinguish real from fake images develops features with interesting abstraction capabilities. In our work, we build on this observation and exploit a way to control the level of corruption of the fake images (see Section 3.3.1). Thus, we train a classifier to discriminate between real and corrupt images (see Section 3.3.3). As illustrated earlier, we hope that the classifier learns features suitable for other tasks such as object classification, detection, and segmentation by solving this task. In the next sections, we describe our model more in detail and present the design choices to avoid learning trivial features.

3.3.1 The Damage & Repair Network

In our approach, we would like to create corrupt images that are not too unrealistic; otherwise, a classifier could distinguish them from real images by detecting only low-level statistics (*e.g.*, unusual local texture patterns). At the same time, we would like to have as much variability as possible so that a classifier can build a robust model of real images.

To address the second concern, we randomly corrupt real images of an existing dataset. To address the first concern, instead of editing images at the pixel-level, we corrupt their feature representation and partly repair the corruption by fixing only the low-level details. We encode an image $\mathbf{x}$ in a feature $\phi(\mathbf{x}) \in \mathbf{R}^{M \times N \times L}$, where $\phi(\mathbf{x}) = E(\mathbf{x})$, and then at each spatial coordinate in the $M \times N$ domain we randomly drop all the L channels with a given probability $\theta \in (0, 1)$. This defines a binary mask matrix $\Omega \in \{0, 1\}^{M \times N}$ of what feature entries are dropped ($\Omega_{ij} = 0$) and which ones are preserved ($\Omega_{ij} = 1$). The dropped feature channels are replaced by the corresponding entries of an averaged feature computed by convolving $\phi(\mathbf{x})$ with a large uniform kernel u. As an alternative, we also replace the dropped feature channels with random noise. Experimentally, we find no significant difference in performance

(A) (B) (C)

FIGURE 3.3: **Influence of the repair network.** We show two examples of corrupt output images obtained from our Damage & Repair network. (A) shows two original images. At the bottom-left corner of those images, we display the masks applied to the encoded feature $\phi(\mathbf{x})$. These masks drop on average about 50% of the encoded feature. (B) shows the corrupt output images. The repair network assists the decoder in inpainting texture that is only locally unnoticeable. However, on the global scale, the objects are no longer recognizable as valid instances. (C) shows the output of the decoder when the repair network is not active. In this case, the artifacts are evident and easy to detect by exploiting low-level statistics.

between these two methods. The mask Ω is generated online during training, and thus the same image $\mathbf{x}$ is subject to a different mask at every epoch. If we fed the corrupt feature directly to the decoder D the output would be extremely unrealistic (see Figure 3.3 (c)). Thus, we introduce a *repair* network that partially compensates for the loss of information due to the mask. The repair network introduces repair layers R_i between layers of the decoder D_i. These layers receive as input the corrupt feature or the outputs of the decoder layers and are allowed to fix only entries that were dropped. More precisely, we define the input to the first decoder layer D_1 as

$$\hat{\phi}(\mathbf{x}) + (1 - \Omega) \odot R_1(\hat{\phi}(\mathbf{x})), \tag{3.1}$$

where $\hat{\phi}(\mathbf{x}) = \Omega \odot \phi(\mathbf{x}) + (1 - \Omega) \odot (u * \phi(\mathbf{x}))$ and u is a large uniform filter. At the later layers D_2, D_3, and D_4 we upsample the mask Ω with the nearest neighbor method and match the spatial dimension of the corresponding intermediate output, *i.e.*, we provide the following input to each layer D_i with $i = 2, 3, 4$

$$D_{i-1} + (1 - U_{i-1}(\Omega)) \odot R_i(D_{i-1}), \tag{3.2}$$

where U_{i-1} denotes the upsampling (with the nearest neighbor method) to the spatial dimensions of the output of D_{i-1}. Finally, we also design our encoder E so that it encodes features that are spatially localized and with limited overlap with one another. To do so, we define the encoder $E = \{E_1, E_2, E_3, E_4, E_5\}$ with five layers where E_1 uses 3×3 convolutional filters with stride 1 and the remaining four layers E_2, E_3, E_4, and E_5 use 2×2 convolutional filters with stride 2. As shown in Figure 3.4, this

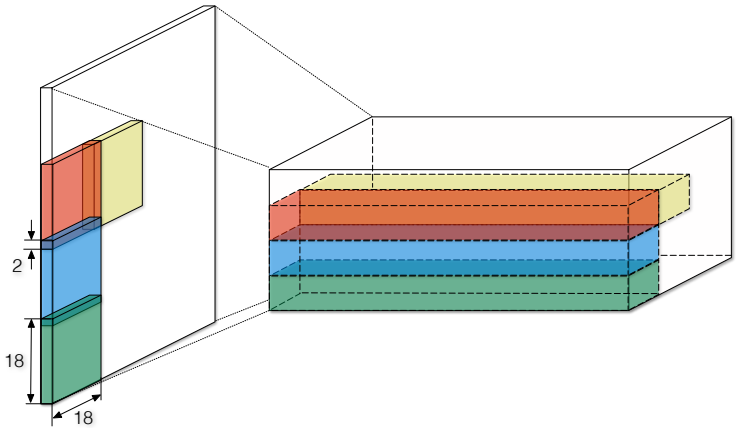


FIGURE 3.4: **Encoder design.** We show the input image on the left, and on the right, we show the corresponding feature encoding $\phi(\mathbf{x})$. We use 5 layers, where the first one uses 3×3 convolutional filters with stride 1 and the remaining 4 use 2×2 convolutional filters with stride 2. As can be observed, this results in almost separate receptive fields for each feature entry of $\phi(\mathbf{x})$. Each entry corresponds to an 18×18 pixels patch in the original input and overlaps 2 pixels with the neighboring patches (1 pixel each side). This encoding ensures a strong spatial locality to each entry of the encoded features.

design limits the receptive field of each encoded feature entry to an 18×18 pixels patch in the input image with a 2 pixels overlap with neighboring patches. Thus, dropping one of these feature entries is essentially equivalent to dropping one of the 18×18 pixels patches in the input image.

Figure 3.3 shows two examples of real images and corresponding corrupt images obtained under different conditions. Figure 3.3 (a) shows the original images and the mask Ω, where $M, N = 8$, inset on the bottom-left corner. The dropping probability is $\theta = 0.5$. Figure 3.3 (b) shows the resulting corrupt images obtained by our complete architecture. Notice that locally it is impossible to determine whether the image is real or corrupt. Only by looking at a larger region and exploiting prior knowledge of what real objects look like can we determine that the image is corrupt. For comparison purposes, we show in Figure 3.3 (c) and (d) the corresponding corrupt images obtained by disabling the repair network and by using the repair network without the mask restriction, respectively.

3.3.2 Replication of Real Images

Given the corrupt images generated as described in the previous section, we should be ready to train the discriminator C. The real images could indeed be just the original images from the same dataset used to create the corrupt ones. One potential issue with this scheme is that the discriminator may learn to distinguish real from corrupt images based on image processing patterns introduced by the decoder network. These patterns may be unnoticeable to the naked eye, but neural networks seem to be quite good at spotting them. For example, networks have learnt to detect chromatic aberration [3] or downsampling patterns [53].

To avoid this issue, we use the same autoencoder $\{E, D\}$ used to generate corrupt images to replicate real images. Since the same final decoder layer generates the real

and corrupt images, we expect both images to share the same processing patterns. This should discourage the discriminator from focusing on such patterns.

We therefore pre-train this autoencoder and make sure that it has a high capacity, so that images are replicated with high accuracy. The training of E and D is a standard optimization with the following least-squares objective

$$\mathcal{L}_{\text{auto}} = \sum_{\mathbf{x}\sim p(\mathbf{x})} |D(E(\mathbf{x})) - \mathbf{x}|^2. \tag{3.3}$$

3.3.3 Training the Discriminator

As just discussed in Section 3.3.2, to replicate real images we use an autoencoder $\{E, D\}$ and, as described in Section 3.3.1, to create corrupt images we use a *damage & repair* autoencoder $\{\Omega \odot E, \hat{D}\}$, where $\hat{D} = \{R_1, D_1, R_2, D_2, R_3, D_3, R_4, D_4, R_5, D_5\}$. Then, we train the discriminator and the repair subnetwork $R = \{R_1, R_2, R_3, R_4, R_5\}$ via adversarial training [24]. Our discriminator C has two outputs, a binary probability $C^{\text{class}} \in [0,1]$ for predicting real vs corrupt images and a prediction mask $C^{\text{mask}} \in [0,1]^{M\times N}$ to localize artifacts in the image. Given an image $\mathbf{x}$, training the discriminator C and the repair network R involves solving

$$\mathcal{L}_{\text{class}} = \min_R \max_C \sum_{\mathbf{x}\sim p(\mathbf{x})} \log C^{\text{class}}(D(\phi(\mathbf{x}))) + \log(1 - C^{\text{class}}(\hat{D}(\hat{\phi}(\mathbf{x})))). \tag{3.4}$$

We also train the discriminator to predict the mask Ω by minimizing

$$\mathcal{L}_{\text{mask}} = \min_C \sum_{\hat{\mathbf{x}}} \sum_{ij} \Omega_{ij} \log \sigma\left(C^{\text{mask}}_{ij}(\hat{\mathbf{x}})\right) + (1 - \Omega_{ij}) \log\left(1 - \sigma(C^{\text{mask}}_{ij}(\hat{\mathbf{x}}))\right)$$

where $\hat{\mathbf{x}} = \hat{D}(\hat{\phi}(\mathbf{x}))$ and $\sigma(z) = 1/(1+e^{-z})$ is the sigmoid function.

3.3.4 Implementation

Let `(64)3c2` denote a convolutional layer with 64 filters of size 3×3 at a stride of 2. The architecture of the encoder E is then defined by `(32)3c1-(64)2c2-(128)2c2-(256)2c2-(512)2c2`. The decoder network D is given by `(256)3rc2-(128)3rc2-(64)3rc2-(32)3rc2-(3)3c1` where *rc* denotes resize-convolutions (*i.e.*, bilinear resizing followed by standard convolution). Batch normalization [118] is applied at all layers of E and D except the last convolutional layer of D. All convolutional layers in E and D are followed by the leaky-ReLU activation $f(x) = \max(x/10, x)$. The filtering of $\phi(x)$ with u is realized using 2D average pooling with a kernel of size 3×3.

The discriminator network C is based on the standard AlexNet architecture [11] to allow for a fair comparison with other methods. The network is identical to the original up to `conv5`. We drop `pool5` and use a single 3×3 convolutional layer for the mask prediction. For the classification, we remove the second fully-connected layer. Batch normalization is only applied after `conv5` during unsupervised training and removed in transfer experiments. The standard ReLU activation is used throughout C. The repair layers follow a similar design as the residual blocks found in ResNet [119]. We illustrate their design in Figure 3.5.

Adam [120] with an initial learning rate of $3 \cdot 10^{-4}$ is used for the optimization. During training, we linearly decay the learning rate to $3 \cdot 10^{-6}$. We set $\beta_1 = 0.5$ and keep all other parameters at their default values. The autoencoder is pre-trained for

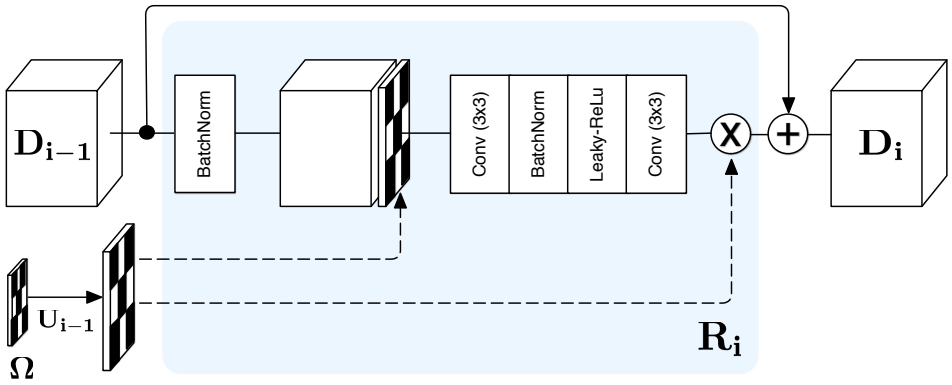


FIGURE 3.5: **The repair block.** We use a residual block design. We concatenate the drop mask to the input of the first convolutional layer. We also gate the output of the last convolutional layer with the same drop mask. This design ensures that the repair block can alter only corrupted regions.

80 epochs, and the damage & repair network is trained for 150 epochs on random 128×128 crops of the 1.3M ImageNet [115] training images.

3.4 Experiments

In this section we first report ablation experiments regarding our damage & repair approach and then compare to prior self-supervised methods in Section 3.4.2.

3.4.1 Ablation Analysis

We validate our design choices by performing transfer learning for classification on STL-10 [117]. All results were obtained with models trained for 400 epochs on the unlabeled training set and supervised transfer learning for 200 epochs on the labeled training set. We use the standard AlexNet architecture during transfer learning except that we drop the `pool5` layer to handle the smaller image size. The weights in the convolutional layers are transferred while the fully-connected layers are randomly initialized. We perform the following set of experiments:

(a) Input image as real: We show that it is better to use autoencoded images as real examples for discriminator training. The rationale here is that the discriminator could exploit the decoder network's pixel patterns to decide between real and corrupted images. We also observed common GAN artifacts in this setting (see Figure 3.6). In our model, the inputs to the discriminator pass through the same convolutional layer.

(b) Without mask prediction: This experiment demonstrates that the additional self-supervision signal provided through the drop mask improves performance.

(c) Distributed vs. local repair network: Here, we illustrate the importance of distributing the repair network throughout the decoder. In the local case, we apply the five repair layers consecutively before the first decoder layer. We observe more low-level artifacts in this setup (see Figure 3.6h).

TABLE 3.1: **Ablation experiments on STL-10.** Influence of different architectural choices on the classification accuracy on STL-10 [117]. Convolutional layers were pre-trained on the proposed self-supervised task and kept fixed during transfer for classification.

	Ablation experiment	Accuracy
(a)	Input image as real	74.99%
(b)	Without mask prediction	78.44%
(c)	Distributed vs. local repair network	77.51%
(d)	3×3 encoder convolutions	79.84%
(e)	No gating in repair layers	79.66%
(f)	No history of corrupted examples	79.76%
(g)	Dropping rate = 0.1	70.92%
(h)	Dropping rate = 0.3	76.26%
(i)	Dropping rate = 0.7	81.06%
(j)	Dropping rate = 0.9	79.60%
	Baseline	79.94%

(d) 3×3 **encoder convolutions:** In this experiment, we let the encoder features overlap. We replace the 2×2 convolutions with 3×3 convolutions. This change increases the receptive field from 18×18 to 33×33 and results in an overlap of 15 pixels. We observe a small decrease in transfer performance.

(e) No gating in repair layers: We demonstrate the influence of training without gating the repair network output with the drop mask. Without gating, the repair network can potentially affect all image regions.

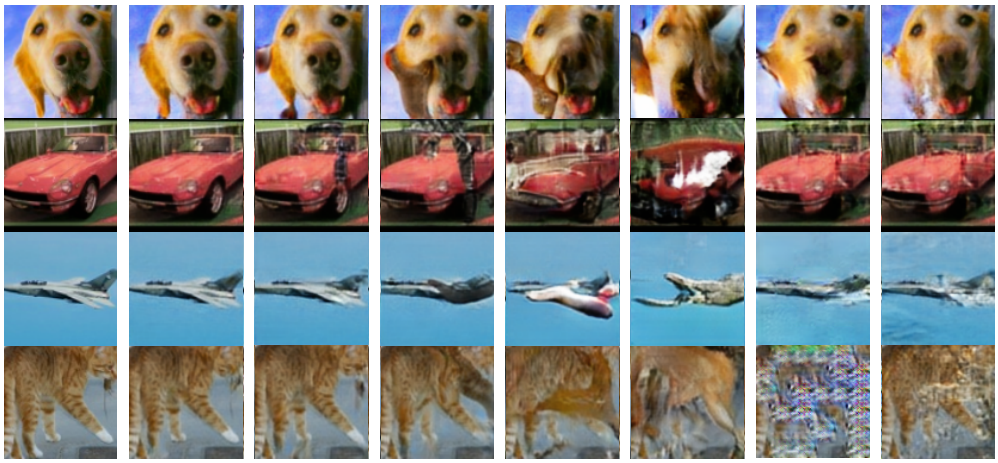
(f) No history of corrupted examples: Following [121] we keep a buffer of corrupted images and replace half of the corrupted examples in the mini-batch with samples from the buffer. Removing the history has a small negative effect on performance.

(g)-(j) Dropping rate: We show how different dropping rates influence the performance on classification. Five different dropping rates are considered: 0.1, 0.3, 0.5 (the baseline), 0.7 and 0.9. We observe that the dropping rate has a strong influence on the performance of the learned features. A value of around 0.7 gives the best results, and low drop rates lead to a significant decrease in performance. With low dropping rates, it is unlikely that object parts are corrupted. This makes the examples less valuable for learning semantic content.

The resulting transfer learning performance of the AlexNet discriminator on STL-10 is shown in Table 3.1. In Figure 3.6 we show renderings for some of the generative models.

3.4.2 Transfer Learning Experiments

We perform transfer learning experiments for classification, detection, and semantic segmentation on standard datasets with our pre-trained AlexNet discriminator weights. We use a dropping probability of $\theta = 0.7$ in all experiments.



(A) (B) (C) (D) (E) (F) (G) (H)

FIGURE 3.6: **Renderings with the damage and repair network.** Column (A) shows the input image. Columns (B)-(F) show results with dropping rates of 0.1, 0.3, 0.5, 0.7 and 0.9. Column (G) shows results when input images are used as real examples. Note that this introduces commonly observed GAN artifacts. Finally, column (H) shows results with the local instead of distributed repair network.

Classification on STL-10

The STL-10 dataset [117] is designed with unsupervised representation learning in mind and is a common baseline for comparison. The dataset contains 100,000 *unlabeled* training images, 5,000 *labeled* training images evenly distributed across 10 classes for supervised transfer and 8,000 test images. The data consists of 96×96 color images. We randomly resize the images and extract 96×96 crops. Unsupervised training is performed for 400 epochs and supervised transfer for an additional 200 epochs.

We follow the standard evaluation protocol and perform supervised training on the ten pre-defined folds. We compare the resulting average accuracy to state-of-the-art results in Table 3.2. We can observe an increase in performance over the other methods. Note that the models compared in Table 3.2 do not use the same network architecture, making it difficult to attribute the difference in performance to a specific cause. It nonetheless showcases the potential of the proposed self-supervised learning task and model.

Classification, Detection and Segmentation on PASCAL VOC

The Pascal VOC2007 and VOC2012 datasets consist of images coming from 20 object classes. They are relatively challenging benchmarks due to the high variability in size, pose, and objects' position in the images.

We transfer only the convolutional layers of the AlexNet and randomly initialize the fully-connected layers. The data-dependent rescaling proposed by Krähenbühl *et al.* [128] is used in all experiments, as is standard practice. The convolutional layers are fine-tuned, *i.e.*, not frozen. This experiment demonstrates the usefulness of the discriminator weights as initialization for other tasks. A comparison to the state-of-the-art feature learning methods is shown in Table 3.3.

TABLE 3.2: **Comparison to prior works on STL-10.** Following the guidelines in [117] the average accuracy from models trained on the ten pre-defined folds is reported. We train a linear classifier on top of `conv5` features for a fair comparison with the other methods.

Model	Accuracy	SD
Dosovitskiy *et al.*[69]	74.2%	±0.4
Dundar *et al.et al.*[122]	74.1%	-
Huang *et al.*[123]	76.8%	±0.3
Swersky *et al.*[124]	70.1%	±0.6
Zhao *et al.*[125]	74.3%	-
Ours (`conv1-conv5` frozen)	**76.9%**	±0.2
Ours (`conv1-conv5` finetuned)	**80.1%**	±0.3

Classification on VOC2007. For multilabel classification, we use the framework provided by Krähenbühl *et al.* [128]. The fine-tuning is performed on random crops of the *'trainval'* dataset. The final predictions are computed as the average prediction of 10 random crops per test image. With an mAP of 69.8%, we achieve state-of-the-art performance on this task.

Detection on VOC2007. The Fast-RCNN [129] framework is used for detection. We follow the guidelines in [128] and use multi-scale training and single-scale testing. All other settings are kept at their default values. With an mAP of 52.5%, we achieve the second-best result.

Semantic Segmentation on VOC2012. We use the standard FCN framework [130] with default settings. We train for 100,000 iterations using a fixed learning rate of 10^{-4}. Our discriminator weights achieve a state-of-the-art result with a mean intersection over union (mIU) of 38.1%.

Layerwise Performance on ImageNet & Places

We evaluate the quality of representations learned at different depths of the network with the evaluation framework introduced in [55]. We freeze all convolutional layers and train multinomial logistic regression classifiers on top of them. The convolutional layers' outputs are resized such that the flattened features are of similar size ($\approx$ 9200). A comparison to other models on ImageNet is given in Table 3.4. Our model outperforms all other approaches in this benchmark. Note also that our `conv1` features perform even slightly better than the supervised counterparts. To demonstrate that the learned representations generalize to other input data, we also performed a transfer to the Places [131] dataset. This dataset contains 2.4M images from 205 scene categories. As can be seen in Table 3.5 we outperform all the other methods for layers `conv2-conv5`. Note also that we achieve the highest overall accuracy with 37.3%.

3.4.3 Qualitative Analysis of the Features

To better understand what the discriminator has learned we use different network visualization techniques. We show the learnt `conv1` filters as well as maximally activating image-patches [132], [133] for some neurons of each convolutional layer in

TABLE 3.3: **Transfer learning experiments on Pascal VOC.** We report results for classification, detection, and segmentation on Pascal VOC2007 and VOC2012 compared to state-of-the-art feature learning methods.

Model	[Ref]	Classification (mAP)	Detection (mAP)	Segmentation (mIU)
Krizhevsky *et al.*[11]	[55]	79.9%	56.8%	48.0%
Random	[31]	53.3	43.4%	19.8%
Agrawal *et al.*[126]	[27]	54.2%	43.9%	-
Bojanowski *et al.*[127]	[127]	65.3%	49.4%	-
Doersch *et al.*[3]	[27]	65.3%	51.1%	-
Donahue *et al.*[27]	[27]	60.1%	46.9%	35.2%
Jayaraman & Grauman [113]	[113]	-	41.7%	-
Krähenbühl *et al.*[128]	[128]	56.6%	45.6%	32.6%
Larsson *et al.*[57]	[57]	65.9%	-	38.0%
Noroozi & Favaro [4]	[4]	67.6%	**53.2%**	37.6%
Noroozi *et al.*[53]	[53]	67.7%	51.4%	36.6%
Owens *et al.*[62]	[62]	61.3%	44.0%	-
Pathak *et al.*[31]	[31]	56.5%	44.5%	29.7%
Pathak *et al.*[61]	[61]	61.0%	52.2%	-
Wang & Gupta [59]	[128]	63.1%	47.4%	-
Zhang *et al.*[55]	[55]	65.9%	46.9%	35.6%
Zhang *et al.*[56]	[56]	67.1%	46.7%	36.0%
Ours	-	**69.8%**	52.5%	**38.1%**

Figure 3.7. We observe prominent edge-detectors in the `conv1` filters, much like what can be observed in a supervised AlexNet. Figure 3.8 shows nearest neighbor retrievals obtained with our `conv5` features.

We use Grad-CAM [134] in Figure 3.9 to illustrate what image regions the discriminator focuses on when deciding between real and with artifacts. We can observe that the discriminator often looks for missing or existing object parts.

3.5 Discussion

This chapter has demonstrated how to learn features by classifying images into 'real' or 'with artifacts.' The image artifacts are carefully constructed so that they require a high-level understanding of image semantics and are not recognizable from low-level features. This classification task is designed to use images without human annotation and thus can exploit large readily-available image datasets. Our approach to generating non-trivial artifacts combines autoencoders and an assistive network (the repair network) with adversarial networks. The transfer (via fine-tuning) of features learned by the classification network achieves state-of-the-art performance on several benchmarks on ILSVRC2012, Pascal VOC and STL-10, demonstrating the effectiveness of the proposed task.

Fundamentally, the task we introduced in this chapter is the recognition of a type of image manipulation. In the next chapter, we investigate other tasks where the goal

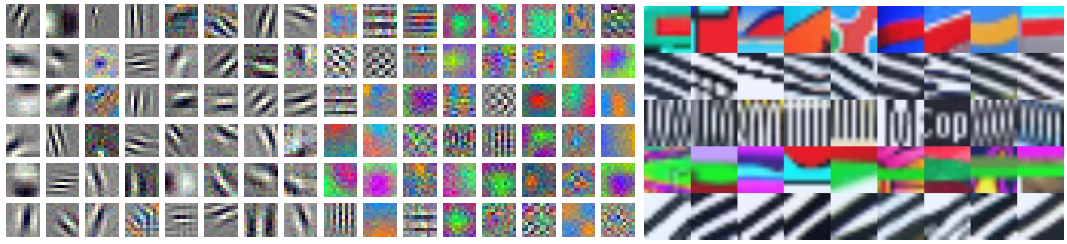
TABLE 3.4: **Linear classifier experiments on ImageNet.** We report validation set accuracy of linear classifiers on frozen convolutional features after unsupervised pre-training. Results for the other methods are taken from [53].

Model\Layer	`conv1`	`conv2`	`conv3`	`conv4`	`conv5`
Krizhevsky *et al.*[11]	19.3%	36.3%	44.2%	48.3%	50.5%
Random	11.6%	17.1%	16.9%	16.3%	14.1%
Doersch *et al.*[3]	16.2%	23.3%	30.2%	31.7%	29.6%
Donahue *et al.*[27]	17.7%	24.5%	31.0%	29.9%	28.0%
Krähenbühl *et al.*[128]	17.5%	23.0%	24.5%	23.2%	20.6%
Noroozi & Favaro [4]	18.2%	28.8%	34.0%	33.9%	27.1%
Noroozi *et al.*[53]	18.0%	30.6%	34.3%	32.5%	25.7%
Pathak *et al.*[31]	14.1%	20.7%	21.0%	19.8%	15.5%
Zhang *et al.*[55]	13.1%	24.8%	31.0%	32.6%	31.8%
Zhang *et al.*[56]	17.7%	29.3%	35.4%	35.2%	32.8%
Ours	**19.5%**	**33.3%**	**37.9%**	**38.9%**	**34.9%**

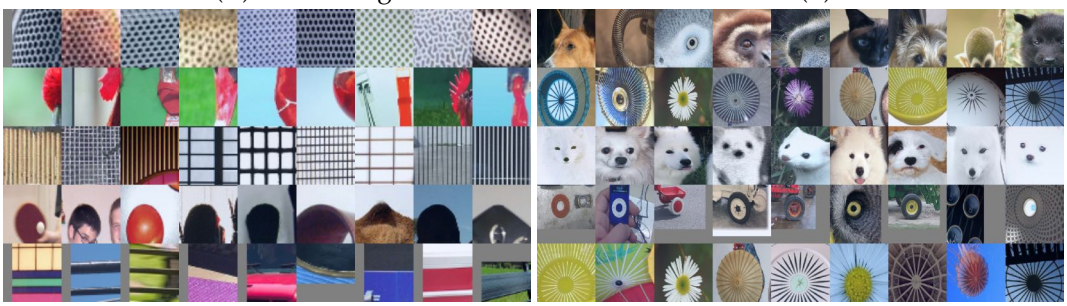
is to recognize classes of image transformations and provide insights into why and when such tasks learn useful features.

TABLE 3.5: **Linear classifier experiments on Places.** We report validation set accuracy of linear classifiers on frozen convolutional features after unsupervised pre-training. Results for the other methods are taken from [53].

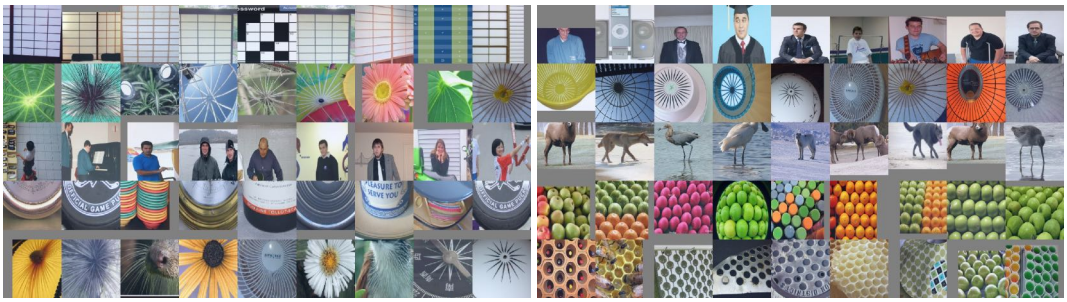
Model\Layer	`conv1`	`conv2`	`conv3`	`conv4`	`conv5`
Places-labels *et al.*[11]	22.1%	35.1%	40.2%	43.3%	44.6%
ImageNet-labels *et al.*[11]	22.7%	34.8%	38.4%	39.4%	38.7%
Random	15.7%	20.3%	19.8%	19.1%	17.5%
Doersch *et al.*[3]	19.7%	26.7%	31.9%	32.7%	30.9%
Donahue *et al.*[27]	22.0%	28.7%	31.8%	31.3%	29.7%
Krähenbühl *et al.*[128]	21.4%	26.2%	27.1%	26.1%	24.0%
Noroozi & Favaro [4]	23.0%	31.9%	35.0%	34.2%	29.3%
Noroozi *et al.*[53]	**23.3%**	33.9%	36.3%	34.7%	29.6%
Owens *et al.*[62]	19.9%	29.3%	32.1%	28.8%	29.8%
Pathak *et al.*[31]	18.2%	23.2%	23.4%	21.9%	18.4%
Wang & Gupta [59]	20.1%	28.5%	29.9%	29.7%	27.9%
Zhang *et al.*[55]	16.0%	25.7%	29.6%	30.3%	29.7%
Zhang *et al.*[56]	21.3%	30.7%	34.0%	34.1%	32.5%
Ours	**23.3%**	**34.3%**	**36.9%**	**37.3%**	**34.4%**



(A) conv1 weights (B) conv1



(C) conv2 (D) conv3



(E) conv4 (F) conv5

FIGURE 3.7: **Visualization of the learned features.** Learned features for neurons in different layers of the AlexNet after unsupervised training are visualized. We show `conv1` weights and maximally activating image-patches for five neurons at each layer.

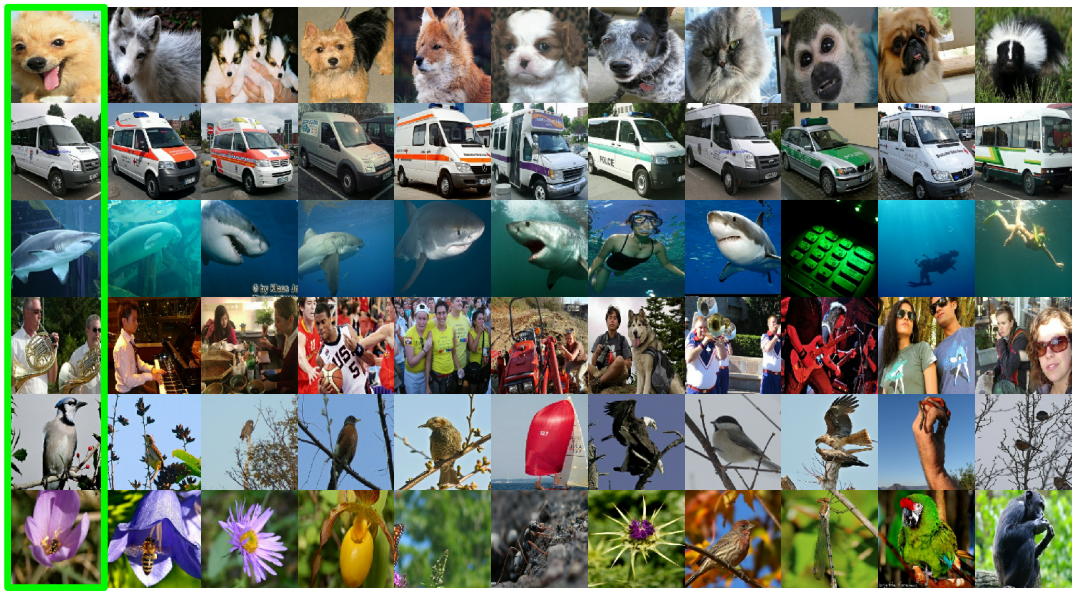


FIGURE 3.8: **Nearest-neighbor retrievals on the ImageNet validation set.** Nearest-neighbors were obtained using cosine-similarity on `conv5` features from the AlexNet discriminator.

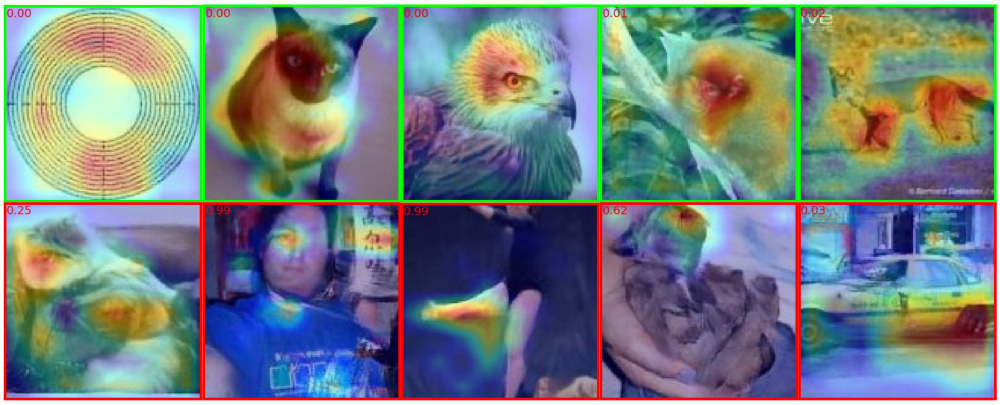


FIGURE 3.9: **Grad-CAM visualization.** We show a visualization of image features contributing to the decision between real images (green border) and images with artifacts (red border). The discriminator appears to focus on object parts such as legs and heads as well as geometric shapes.

Chapter 4

Learning by Recognizing Image Transformations

"If you just focus on the smallest details, you will never get the big picture right."

— Leroy Hood

The previous chapter demonstrated how the self-supervised classification of images into whether they were transformed through some corruption process or kept untouched could yield good image representations. This chapter discusses how self-supervised learning can address another fundamental limitation of learning through labels: Depending on the training procedure and dataset, it might yield features that describe mostly local statistics (*e.g.*, textures) and thus have limited generalization capabilities. An illustration of this issue is shown in Figure 4.1. On the bottom row, we show images that have been transformed such that local statistics of the corresponding image on the top row are preserved, but global statistics are not.[2] Experimentally, we find that features pre-trained with ImageNet labels [115] have difficulties in telling real images apart from the transformed ones. This simple test illustrates that focusing on local image statistics could mostly solve the classification task on ImageNet. We might not notice such a problem when evaluating these features on tasks and datasets that are solvable based on similar local statistics. However, more general classification settings would certainly expose such a limitation. [1] also pointed out this problem and showed that training supervised models to focus on the global statistics (which they refer to as *shape*) can improve the generalization and the robustness of the learned image representation.

Thus, to address this fundamental shortcoming and to limit the need for human annotation, this chapter describes a novel self-supervised learning (SSL) method. The training task in our method is to *discriminate variations of global image statistics*. To this end, we transform images so that local statistics are mostly unchanged while global statistics are clearly altered. This way, we make sure that the discrimination of such transformations is not possible by working on just local patches. Instead, it requires using the whole image, learning also the global image statistics. We illustrate this principle in Figure 4.3. Incidentally, several existing SSL tasks can be seen as learning from such transformations, *e.g.*, context prediction [31], rotation prediction [54], solving jigsaw puzzles [4], and the spotting artifacts task from Chapter 3.

Material from: S. Jenni, H. Jin, and P. Favaro. "Steering Self-Supervised Feature Learning Beyond Local Pixel Statistics." In *IEEE Conference on Computer Vision and Pattern Recognition* 2020.

[2] The transformed images are obtained by partitioning an image into a 4×4 grid, randomly permuting the tiles, and training a network to inpaint a band of pixels across the tiles through adversarial training [24].

FIGURE 4.1: **The importance of global image statistics**. Top row: Natural images. Bottom row: Images transformed such that local statistics are preserved while global statistics are significantly altered. An accurate image representation should be able to distinguish these two categories. A linear binary classifier trained to distinguish original versus transformed images on top of `conv5` features pre-trained on ImageNet labels yields an accuracy of 78%. If instead, we use features pre-trained with our proposed self-supervised learning task, the classifier achieves an accuracy of 85%. Notice that our self-supervised pre-training did not use this transformation and that the transformed images were built independently of either feature.

We cast the task of discriminating changes in the global image statistics as the self-supervised recognition of several image transformations (see Figure 4.2). As a novel image transformation, we introduce *Limited Context Inpainting* (LCI). LCI selects a random patch from a natural image, substitutes the center with noise (thus preserving a small outer boundary of pixels), and trains a network to inpaint a realistic center through adversarial training. While LCI can inpaint a natural center of the patch that seamlessly blends with the preserved boundaries, it is unlikely to provide a meaningful match with the rest of the original image. Hence, this mismatch can only be detected by learning the global statistics of the image. Our formulation is also highly scalable and allows us to incorporate more transformations as additional categories easily. In fact, we also include the classification of image warping and image rotations (see examples of such transformations in Figure 4.3). An illustration of the proposed training scheme is shown in Figure 4.2.

Contributions. Our proposed method has the following original contributions: 1) we introduce a novel self-supervised learning principle based on image transformations that only global observations can detect; 2) we introduce a novel transformation according to this principle and demonstrate its impact on feature learning experimentally; 3) we formulate the method so that it can easily scale with additional transformations; 4) our proposed method achieves a state of the art performance in transfer learning on several data sets; in particular, for the first time, we show that our trained features, when transferred to Places, achieve performance on par with features trained through supervised learning with ImageNet labels.

4.1 Background

We discuss how our approach relates to the most relevant prior works here and refer to Chapter 2 for more background discussion.

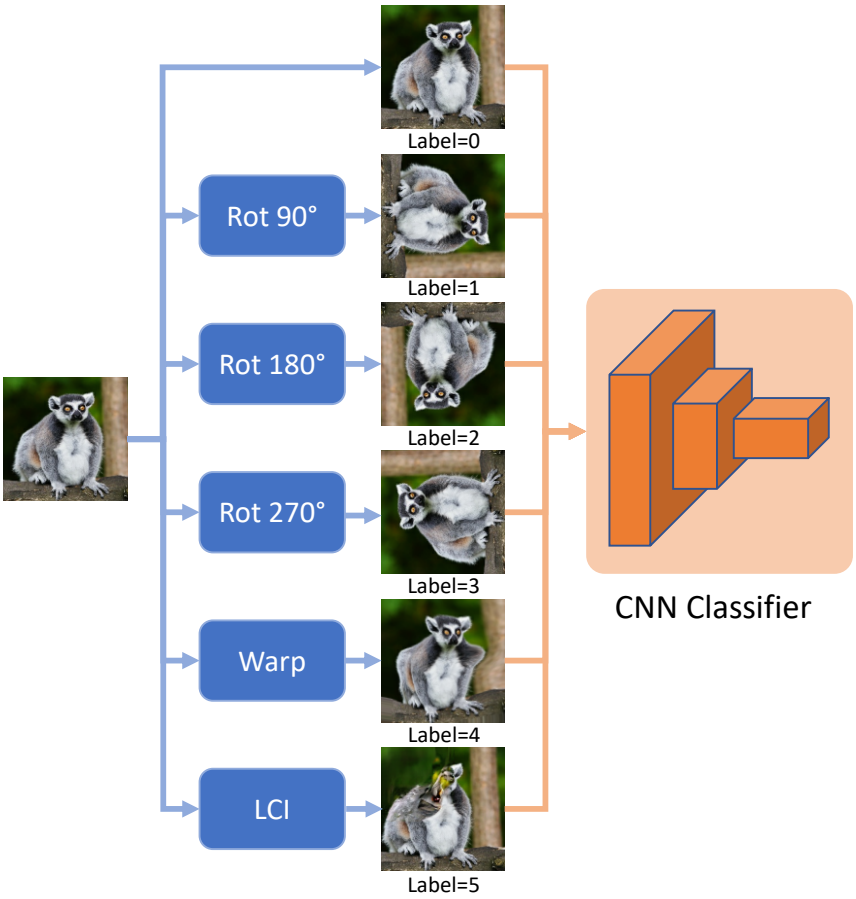


FIGURE 4.2: **Learning global statistics**. We propose to learn image representations by training a convolutional neural network to classify image transformations. The transformations are chosen such that local image statistics are preserved while global statistics are distinctly altered.

Connections to prior SSL approaches. Several self-supervised tasks can be seen as the prediction of some form of transformation applied to an image. Gidaris *et al.* [54] as an obvious example, predict the number of 90° rotations applied to an image. [3], [4] predict transformations at the level of image patches. Recently, Zhang *et al.* [135] proposed to predict the parameters of a relative projective transformation between two images using a Siamese architecture. Finally, the task introduced in Chapter 3 also fits this framework. This chapter shows that we can form new and more challenging learning tasks that learn better features by predicting a combination of novel and previously explored image transformations.

The damage&repair network from Chapter 3 has some similarities to the LCI transformation. In Chapter 3 we generated image artifacts by erasing and locally repairing hidden representations of an autoencoder. The LCI approach is different in two important ways. First, it more strongly limits the context of the inpainter and is by design a local operation. Second, a separate patch discriminator allows stable adversarial training independent of the feature learning component.

Recognizing image manipulations. Several works have considered the detection of image manipulations in the context of image forensics [136], [137], [138], [139]. For example, Wang *et al.* [137] predict subtle face image manipulations based on local

(a) (b) (c) (d)

FIGURE 4.3: **Selected image transformations**. Examples of local patches from images that were (a) warped, (b) locally inpainted, (c) rotated, or (d) not transformed. The bottom row shows the original images; the middle row shows the corresponding transformed images, and the top row shows a detail of the transformed image. By only observing a local patch (top row), is it possible in all of the above cases to tell if and how an image has been transformed, or is it instead necessary to observe the whole image (middle row), *i.e.*, the global pixel statistics?

warping. Zhou *et al.* [138] detect image tampering generated using semantic masks. In these cases, transformations are usually subtle and do not change the global image statistics in a predictable way (images are manipulated to appear realistic). The aim is therefore antithetical to ours.

4.2 Learning Features by Discriminating Global Image Transformations

We aim to learn image representations without human annotation by recognizing variations in global image statistics. We do so by distinguishing between natural images and images that underwent several different image transformations. Our principle is to choose image transformations that: 1) preserve local pixel statistics (*e.g.*, texture), but alter the global image statistics of an image and 2) can be recognized from a single transformed example in most cases. In this exposition, we choose the following transformations: limited context inpainting, warping, rotations, and the case of no transformation. We will introduce these transformations in detail in the next sections.

Formally, given a set of unlabeled training images $\{x_i\}_{i=1,\dots,N}$ and a set of image transformations $\{T_j\}_{j=0,\dots,K}$, we train a classifier C to predict the transformation-label j given a transformed example $T_j \circ x_i$. In our case we set $K = 5$. We include the identity (no-transformation) case by letting $T_0 \circ x \doteq x$. We train the network C by

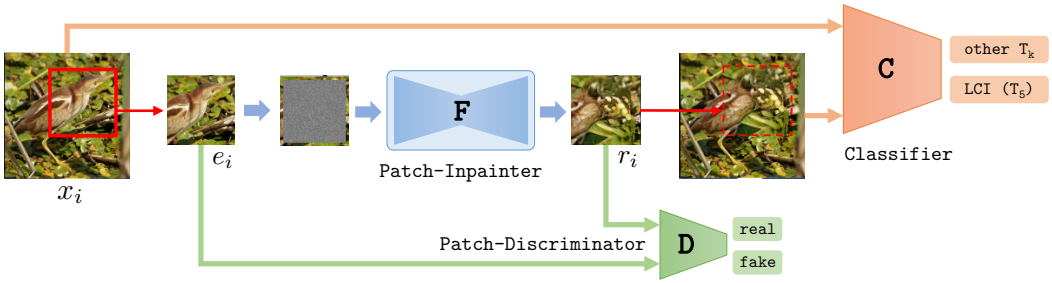


FIGURE 4.4: **Training of the Limited Context Inpainting (LCI) network**. A random patch is extracted from a training image x, and all but a thin border of pixels is replaced by random noise. The inpainter network F fills the patch with realistic textures conditioned on the remaining border pixels. The resulting patch is replaced back into the original image, generating an image with natural local image statistics but unnatural global statistics.

minimizing the following self-supervised objective

$$\mathcal{L}_{\text{SSL}}(T_0, \dots, T_5) \doteq \min_C \frac{1}{6N} \sum_{i=1}^{N} \sum_{y=0}^{5} \ell_{\text{cls}}\Big(C\big(T_y \circ x_i\big), y\Big), \tag{4.1}$$

where ℓ_{cls} is the standard cross-entropy loss for a multi-class classification problem.

4.2.1 Limited Context Inpainting

The first transformation that we propose to use in Equation (4.1) is the local inpainting operation LCI. The aim here is to modify images only locally, *i.e.*, at the scale of image patches. We train an inpainter network F conditioned only on a thin border of pixels of the patch (see Figure 4.4). The inpainted patch should be realistic on its own and blend in at the boundary with the surrounding image, but should not meaningfully match the content of the whole image (see an example in Figure 4.3 (b)). The inpainter F is trained using adversarial training against a patch discriminator D (which ensures that we match the local statistics) and the transformation classifier C. The patch to be inpainted is randomly selected at a uniformly sampled location $\Delta \in \Omega$, where Ω is the image domain. Then, $\mathcal{W}_\Delta \subset \Omega$ is a square region of pixels around Δ. We define e_i as the original patch of pixels at $\mathcal{W}_\Delta$ and r_i as the corresponding inpainted patch

$$e_i(p - \Delta) \doteq x_i(p), \quad \forall p \in \mathcal{W}_\Delta \tag{4.2}$$

$$r_i \doteq F(e_i \odot (1 - m) + z \odot m) \tag{4.3}$$

with m a mask that is 1 in the center of the patch and 0 at the boundary (2 to 4 pixels in our baseline), $z \sim \mathcal{N}(0, I)$ is a zero-mean Gaussian noise, and $\odot$ denotes the Hadamard (pixel-to-pixel) product. The LCI transformation T_5 is then defined as

$$(T_5 \circ x_i)(p) \doteq \begin{cases} x_i(p) & \text{if } p \notin \mathcal{W}_\Delta \\ r_i(p - \Delta) & \text{if } p \in \mathcal{W}_\Delta. \end{cases} \tag{4.4}$$

Finally, to train the inpainter F, we minimize the cost

$$\mathcal{L}_{\text{inp}} = \frac{1}{N}\sum_{i=1}^{N} \ell_{\text{GAN}}(r_i, e_i) + \lambda_{\text{border}} \left|(r_i - e_i) \circ (1 - m)\right|^2 - \mathcal{L}_{\text{SSL}}(T_0, \ldots, T_5),$$

where $\lambda_{\text{border}} = 50$ is a tuning parameter to regulate the importance of autoencoding the input boundary, and $\ell_{\text{GAN}}(\cdot, \cdot)$ is the hinge loss for adversarial training [140], which also includes the maximization in the discriminator D.

Remark. In contrast to prior SSL methods [5], [31], [52], here we do not take the features from the networks that we used to learn the transformation (*e.g.*, D or F). Instead, here we take features from a separate classifier C that has only a partial role in the training of F. This separation has several advantages: 1) a separate tuning of training parameters is possible; 2) GAN tricks can be applied without affecting the classifier C; (3) GAN training can be stable even when the classifier wins ($\mathcal{L}_{\text{SSL}}$ saturates w.r.t. F).

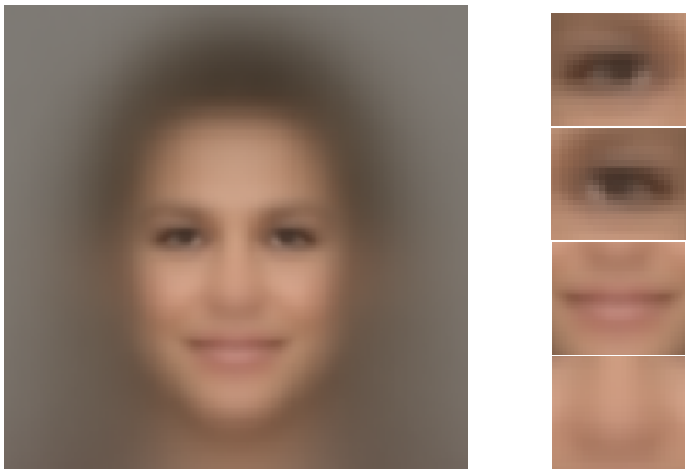
4.2.2 Random Warping

In addition to the LCI, a local image transformation, we consider random global warping as our T_4 transformation. A warping is a smooth deformation of the image coordinates defined by n pixel coordinates $\{(u_i, v_i)\}_{i=1,\ldots,n}$, which act as control points. We place the control points on a uniform grid of the image domain and then randomly offset each control point by sampling the shifts from a rectangular range $[-d, d] \times [-d, d]$, where d is typical $1/10$-th of the image size. The dense flow field for warping is then computed by interpolating between the offsets at the control points using a polyharmonic spline [141]. Warping affects the local image statistics only minimally: in general, it is difficult to distinguish a warped patch from a patch undergoing a change in perspective. Therefore, the classifier needs to learn global image statistics to detect image warping.

4.2.3 Image Rotations

Finally, we consider as T_1, T_2, and T_3 image rotations of 90°, 180°, and 270°. This choice is inspired by Gidaris *et al.* [54] who proposed RotNet, a network to predict image rotations by multiples of 90°. This was shown to be a simple yet effective SSL pretext task. These transformations are predictable because the photographer bias introduces a canonical reference orientation for many natural images. They also require global statistics as local patches of rotated images often do not indicate the image's orientation because similar patches can be found in the untransformed dataset.

Remark. There exist, however, several settings in which the prediction of image rotations does not result in useful features. Many natural images, for example, do not have a canonical image orientation. Thus, in these cases, the prediction of image rotations is an ill-posed task. There also exist entire data domains of interest, where the image orientation is ambiguous, such as satellite and cell imaging datasets. Even when a clear upright image orientation exists, this method alone can lead to non-optimal feature learning. As an example, we show that the prediction of image rotations on CelebA [142], a dataset of face images, leads to significantly worse features than can be learned through the prediction of other transformations (see Table 4.3).



(A) (B)

FIGURE 4.5: **Image statistics on CelebA**. (a) The mean image obtained from 8000 samples from CelebA. (b) Four local patches extracted from the mean image. Because these patterns always appear with the same orientation in the dataset, it is possible to distinguish rotated images using only these local statistics.

This limitation's main reason is that local patches can be found in the dataset always with the same orientation (see Figure 4.5). For instance, the classifier can easily distinguish rotated faces by only detecting one eye or the mouth.

4.2.4 Preventing Degenerate Learning

As was observed by Doersch *et al.* [3], networks trained to solve a self-supervised task might do so by using very local statistics (*e.g.*, localization by detecting the chromatic aberration). Such solutions are called *shortcuts* and are a form of degenerate learning as they yield features with poor generalization capabilities. When introducing artificial tasks, such as the discrimination of several image transformations, it is vital to ensure that the trained network cannot exploit (local) artifacts introduced by the transformations to solve the task. For example, the classifier could learn to recognize processing artifacts of the inpainter F in order to recognize LCI transformed images. Although adversarial training should help prevent this behavior, we find that it is not sufficient by itself. To further prevent such failure cases, we also train the network F to autoencode image patches by modifying the loss $\mathcal{L}_{\text{inp}}$ in Equation (5.1) as $\mathcal{L}_{\text{inp,AE}} = \mathcal{L}_{\text{inp}} + \lambda_{\text{AE}} \frac{1}{N} \sum_{i=1}^{N} |F(e_i) - e_i|^2$, where $\lambda_{\text{AE}} = 50$ is a tuning parameter to regulate the importance of autoencoding image patches. We also create artificial untransformed images by substituting a random patch with its autoencoded version. In each mini-batch to the classifier, we replace half of the untransformed images with these patch-autoencoded images. In this manner, the classifier will not focus on the small artifacts (which could even be invisible to the naked eye) to discriminate the transformations. We also replace half of the original images in a minibatch with these patch-autoencoded images before applying the rotation during training.

4.3 On the Choice of Transformations

Our goal is to learn features by discriminating images undergoing different transformations. We pointed out that this approach should use transformations that are

distinguishable only by observing large regions of pixels and is scalable, *i.e.*, it can be further refined by including more transformations. In this section, we would like to make these two aspects clearer.

Determining suitable transformations. We find that the choice of what transformations to use depends on the data distribution. An example of such dependency in the case of RotNet on CelebA is shown in Figure 4.5. Intuitively, *an ideal transformation is such that any transformed local patch should be found in the original dataset, but any transformed global patch should not be found in the dataset*. This is also the key idea behind the design of LCI.

Introducing additional transformations. As we will show in the Experiments section, adding more transformations (as specified above) can improve the performance. An important aspect is that the classifier must be able to distinguish the different transformations. Otherwise, its task is ambiguous and can lead to degenerate learning. Put in simple terms, *a transformed global patch should be different from any other global patch (including itself) transformed with a different transformation*. We verify that our chosen transformations satisfy this principle. LCI and image warping cannot produce rotated images, and warping is a global deformation, while LCI is a local one.

4.4 Experiments

We perform an extensive experimental evaluation of our formulation on several established unsupervised feature learning benchmarks. For a fair comparison with prior work, we implement the transformation classifier C with a standard AlexNet architecture [11]. Following previous work, we remove the local response normalization layers and add batch normalization [118] to all layers except for the final one. We made no other modifications to the original architecture (we preserve the two-stream architecture). We remove the max-pooling layer after `conv5` and use SAME padding throughout the network for experiments on lower resolution images. We used the standard data-augmentation strategies (random cropping and horizontal flipping). Self-supervised pre-training of the classifier was performed using the AdamW optimizer [143] with parameters $\beta_1 = 0.5$, $\beta_2 = 0.99$ and a weight decay of 10^{-4}. We decayed the learning rate from $3 \cdot 10^{-4}$ to $3 \cdot 10^{-7}$ over the course of training using cosine annealing [144]. The training of the inpainter network F and patch-discriminator D was done using the Adam optimizer [120] with a fixed learning rate of $2 \cdot 10^{-4}$ and $\beta_1 = 0.5$. The size of the patch boundary is set to 2 pixels in experiments on STL-10 and CelebA. On ImageNet, we use a 4-pixel boundary. Details for the network architectures and additional results are provided in the supplementary material.

4.4.1 Ablation Experiments

Limited Context Inpainting

We perform ablation experiments on STL-10 [117] to validate several design choices for the joint inpainter and classifier training. We also illustrate the effect of the patch-size on the performance of the learned features. We pre-train the transformation classifier for 200 epochs on 64×64 crops of the unlabeled training set. The mini-batch size was set to 64. We then transfer the frozen `conv5` features by training a

TABLE 4.1: **Ablation experiments for LCI on STL-10.** We perform ablations for different design choices of Limited Context Inpainting (LCI) on STL-10 [117]. We pre-train an AlexNet to predict if an image has been transformed with LCI or not and transfer the frozen `conv5` features for linear classification.

Ablation	**Accuracy**
(a) 32×32 patches	61.2%
(b) 40×40 patches	70.6%
(c) 56×56 patches	75.1%
(d) Pre-trained and frozen F	63.7%
(e) No adversarial loss w.r.t. C	68.0%
(f) No patch autoencoding	69.5%
Baseline (48×48 patches)	76.2%

TABLE 4.2: **Combinations of image transformations on STL-10.** We report the test set accuracy of linear classifiers trained on frozen features for models trained to predict different combinations of image transformations on STL-10.

Initialization	`conv1`	`conv2`	`conv3`	`conv4`	`conv5`
Random	48.4%	53.3%	51.1%	48.7%	47.9%
Warp	57.2%	64.2%	62.8%	58.8%	55.3%
LCI	58.8%	67.2%	67.4%	68.1%	68.0%
Rot	58.2%	67.3%	69.3%	69.9%	70.1%
Warp + LCI	**59.3%**	68.1%	69.5%	68.5%	67.2%
Rot + Warp	57.4%	69.2%	70.7%	70.5%	70.6%
Rot + LCI	58.5%	69.2%	71.3%	72.8%	72.3%
Rot + Warp + LCI	59.2%	**69.7%**	**71.9%**	**73.1%**	**73.7%**

linear classifier for 500 epochs on randomly cropped 96×96 images of the small labeled training set. Only LCI was used as a transformation in these experiments. The results of the following ablations are reported in Table 4.1:

(a)-(c) Varying patch-size: We vary the size of the inpainted patches. We observe that small patches lead to a significant drop in feature performance. Smaller patches are easy to inpaint, and the results often do not alter the global image statistics;

(d)-(f) preventing shortcuts: Following Section 4.2.4, we show how adversarial training of F is necessary to achieve good performance by removing the feedback of both D and C in (d) and only C in (e). We also demonstrate the importance of adding autoencoded patches to the non-transformed images in (f);

TABLE 4.3: **Combinations of image transformations on CelebA.** We report the average precision of linear classifiers trained to predict facial attributes on frozen features of models trained to predict different combinations of image transformations on CelebA.

Initialization	conv1	conv2	conv3	conv4	conv5
Random	68.9%	70.1%	66.7%	65.3%	63.2%
Warp	71.7%	73.4%	71.2%	68.8%	64.3%
LCI	71.3%	73.0%	72.0%	71.1%	68.0%
Rot	70.3%	70.9%	67.8%	65.6%	62.1%
Warp + LCI	**72.0%**	73.9%	73.3%	72.1%	69.0%
Rot + Warp	71.6%	73.6%	72.0%	70.1%	66.4%
Rot + LCI	71.3%	72.7%	71.9%	70.8%	66.7%
Rot + Warp + LCI	71.8%	**74.0%**	**73.5%**	**72.5%**	**69.2%**

Combination of Image Transformations

We perform additional ablation experiments on STL-10 and CelebA [142] where C is trained to predict different combinations of image transformations. These experiments illustrate how our formulation can scale with the number of considered transformations and how the transformations' effectiveness depends on the data domain.

We pre-train the AlexNet to predict image transformations for 200 epochs on 64×64 crops on STL-10 and 100 epochs on 96×96 crops on CelebA using the standard data augmentations. For transfer, we train linear classifiers on top of the frozen convolutional features (without resizing the feature-maps) to predict the ten object categories in the case of STL-10 and predict the 40 face attributes of CelebA. Transfer learning is performed for 700 epochs on 64×64 crops in the case of STL-10 and 100 epochs on 96×96 crops in the case of CelebA. We report results for STL-10 in Table 4.2 and for CelebA in Table 4.3.

We can observe that the discrimination of many image transformations generally leads to better feature performance on both datasets. When considering each of the transformations in isolation, we see that not all of them generalize equally well to different data domains. Rotation prediction especially performs significantly worse on CelebA than on STL-10. The performance of LCI, on the other hand, is good on both datasets.

4.4.2 Unsupervised Feature Learning Benchmarks

We compare our proposed model to state-of-the-art methods on the established feature learning benchmarks. We pre-train the transformation classifier for 200 epochs on the ImageNet training set. We randomly cropped images to 128×128, and removed the last max-pooling layer during pre-training to preserve the feature map's size before the fully-connected layers. We used a batch size of 96 and trained on 4 GPUs.

Pascal VOC. We finetune our transformation classifier features for multi-label classification, object detection, and semantic segmentation on the Pascal VOC dataset. We follow the established experimental setup and use the framework provided by

TABLE 4.4: **Transfer learning experiments on PASCAL VOC.** We report transfer learning results for classification, detection, and segmentation on PASCAL VOC compared to state-of-the-art feature learning methods (* use a bigger AlexNet).

Model	[Ref]	Classification (mAP)	Detection (mAP)	Segmentation (mIoU)
Krizhevsky *et al.*[11]	[55]	79.9%	59.1%	48.0%
Random	[31]	53.3%	43.4%	19.8%
Agrawal *et al.*[126]	[27]	54.2%	43.9%	-
Bojanowski *et al.*[127]	[127]	65.3%	49.4%	-
Donahue *et al.*[27]	[27]	60.1%	46.9%	35.2%
Feng *et al.*[75]	[75]	74.3%	**57.5%**	**45.3%**
Gidaris *et al.*[54]	[54]	73.0%	54.4%	39.1%
Jayaraman & Grauman [113]	[113]	-	41.7%	-
Jenni & Favaro [5]	[5]	69.8%	52.5%	38.1%
Krähenbühl *et al.*[128]	[128]	56.6%	45.6%	32.6%
Larsson *et al.*[57]	[57]	65.9%	-	38.0%
Noroozi & Favaro [4]	[4]	67.6%	53.2%	37.6%
Noroozi *et al.*[53]	[53]	67.7%	51.4%	36.6%
Noroozi *et al.*[50]	[50]	72.5%	56.5%	42.6%
Mahendran *et al.*[145]	[145]	64.4%	50.3%	41.4%
Mundhenk *et al.*[51]	[51]	69.6%	55.8%	41.4%
Owens *et al.*[62]	[62]	61.3%	44.0%	-
Pathak *et al.*[31]	[31]	56.5%	44.5%	29.7%
Pathak *et al.*[61]	[61]	61.0%	52.2%	-
Wang & Gupta [59]	[128]	63.1%	47.4%	-
Zhan *et al.*[146]	[146]	-	-	44.5%
Zhang *et al.*[55]	[55]	65.9%	46.9%	35.6%
Zhang *et al.*[56]	[56]	67.1%	46.7%	36.0%
Doersch *et al.*[3]*	[27]	65.3%	51.1%	-
Caron *et al.*[71]*	[71]	73.7%	55.4%	45.1
Ours	-	**74.5%**	56.8%	44.4

Krähenbühl *et al.* [128] for multilabel classification, the Fast-RCNN [129] framework for detection and the FCN [130] framework for semantic segmentation. We absorb the batch-normalization parameters into the associated layers' parameters in the AlexNet and apply the data-dependent rescaling by Krähenbühl *et al.* [128], as is standard practice. The results of these transfer learning experiments are reported in Table 4.4. We achieve state-of-the-art performance in classification and competitive results for detection and segmentation.

Linear Classifier Experiments on ImageNet and Places. To measure our SSL task's quality, we use the transformation classifier as a fixed feature extractor and train a linear classifier on top of each convolutional layer. These experiments are performed both on ImageNet (the dataset used for pre-training) and Places [131] (to measure how well the features generalize to new data). We follow the same setup as the state-of-the-art methods and report the accuracy achieved on a single crop. Results

TABLE 4.5: **Linear classifier experiments on ImageNet.** We report validation set accuracy on ImageNet with linear classifiers trained on frozen convolutional layers. † indicates multi-crop evaluation and * use a bigger AlexNet.

Model\Layer	conv1	conv2	conv3	conv4	conv5
ImageNet Labels	19.3%	36.3%	44.2%	48.3%	50.5%
Random	11.6%	17.1%	16.9%	16.3%	14.1%
Donahue *et al.*[27]	17.7%	24.5%	31.0%	29.9%	28.0%
Feng *et al.*[75]	19.3%	33.3%	**40.8%**	41.8%	**44.3%**
Gidaris *et al.*[54]	18.8%	31.7%	38.7%	38.2%	36.5%
Huang *et al.*[147]	15.6%	27.0%	35.9%	39.7%	37.9%
Jenni & Favaro [5]	19.5%	33.3%	37.9%	38.9%	34.9%
Noroozi & Favaro [4]	18.2%	28.8%	34.0%	33.9%	27.1%
Noroozi *et al.*[53]	18.0%	30.6%	34.3%	32.5%	25.7%
Noroozi *et al.*[50]	19.2%	32.0%	37.3%	37.1%	34.6%
Tian *et al.*[106]	18.4%	33.5%	38.1%	40.4%	42.6%
Wu *et al.*[70]	16.8%	26.5%	31.8%	34.1%	35.6%
Zhang *et al.*[55]	13.1%	24.8%	31.0%	32.6%	31.8%
Zhang *et al.*[56]	17.7%	29.3%	35.4%	35.2%	32.8%
Zhang *et al.*[135]	19.2%	32.8%	40.6%	39.7%	37.7%
Doersch *et al.*[3]*	16.2%	23.3%	30.2%	31.7%	29.6%
Caron *et al.*[71]*	12.9%	29.2%	38.2%	39.8%	36.1%
Zhuang *et al.*[72]*†	18.7%	32.7%	38.1%	42.3%	42.4%
Ours	**20.8%**	**34.5%**	40.2%	**43.1%**	41.4%
Ours†	22.4%	37.4%	43.1%	46.6%	46.0%

for ImageNet are shown in Table 4.5 and for Places in Table 4.6. Our learned features achieve state-of-the-art performance for conv1, conv2 and conv4 on ImageNet. On Places we achieve the best results on conv1, conv3 and conv4. Our results on conv4 in particular are the best overall and even slightly surpass the performance of an AlexNet trained on ImageNet using supervision.

ResNet Experiments on STL-10. We performed additional experiments with a more modern network architecture on STL-10. We followed the setup of [149] and trained a ResNet-34 [119] for 200 epochs on the 100K unlabeled training images of STL-10. We then fine-tuned the network for 300 epochs on the 5K labeled training images and evaluated using the 8K test images. The training parameters are the same as in our experiments with AlexNet. We used data augmentation and multi-crop evaluation similar to [149]. Results and comparison to prior work are shown in Table 4.7.

Nearest Neighbor Evaluation. Features learned in deep CNNs through supervised learning tend to distribute so that their Euclidean distance relates closely to the semantic *visual similarity* of the images they correspond to. We want to see if also our SSL features enjoy the same property. Thus, we compute the nearest neighbors of our SSL and SL features in conv5 features space on the validation set of ImageNet. Results are shown in Figure 4.7. We also show a quantitative comparison of k-nearest

TABLE 4.6: **Linear classifier experiments on Places.** We report validation set accuracy on Places with linear classifiers trained on frozen convolutional layers. † indicates multi-crop evaluation and * the use of a bigger AlexNet.

Model\Layer	conv1	conv2	conv3	conv4	conv5
Places Labels	22.1%	35.1%	40.2%	43.3%	44.6%
ImageNet Labels	22.7%	34.8%	38.4%	39.4%	38.7%
Random	15.7%	20.3%	19.8%	19.1%	17.5%
Donahue *et al.*[27]	22.0%	28.7%	31.8%	31.3%	29.7%
Feng *et al.*[75]	22.9%	32.4%	36.6%	37.3%	**38.6%**
Gidaris *et al.*[54]	21.5%	31.0%	35.1%	34.6%	33.7%
Jenni & Favaro [5]	23.3%	**34.3%**	36.9%	37.3%	34.4%
Noroozi & Favaro [4]	23.0%	31.9%	35.0%	34.2%	29.3%
Noroozi *et al.*[53]	23.3%	33.9%	36.3%	34.7%	29.6%
Noroozi *et al.*[50]	22.9%	34.2%	37.5%	37.1%	34.4%
Owens *et al.*[62]	19.9%	29.3%	32.1%	28.8%	29.8%
Pathak *et al.*[31]	18.2%	23.2%	23.4%	21.9%	18.4%
Wu *et al.*[70]	18.8%	24.3%	31.9%	34.5%	33.6%
Zhang *et al.*[55]	16.0%	25.7%	29.6%	30.3%	29.7%
Zhang *et al.*[56]	21.3%	30.7%	34.0%	34.1%	32.5%
Zhang *et al.*[135]	22.1%	32.9%	37.1%	36.2%	34.7%
Doersch *et al.*[3]*	19.7%	26.7%	31.9%	32.7%	30.9%
Caron *et al.*[71]*	18.6%	30.8%	37.0%	37.5%	33.1%
Zhuang *et al.*[72]*†	18.7%	32.7%	38.2%	40.3%	39.5%
Ours	**24.1%**	33.3%	**37.9%**	**39.5%**	37.7%
Ours†	25.5%	35.3%	40.1%	42.1%	40.3%

TABLE 4.7: **Comparison to prior works on STL-10.** We report test set accuracy obtained with a ResNet-34 on STL-10 and compare with other published results. Note that the methods do not all use the same network architecture.

Method	**Accuracy**
Dosovitskiy *et al.*[69]	74.2%
Dundar *et al.et al.*[122]	74.1%
Hjelm *et al.*[148]	77.0%
Huang *et al.*[123]	76.8%
Jenni & Favaro [5]	80.1%
Ji *et al.*[149]	88.8%
Oyallon *et al.*[150]	87.6%
Swersky *et al.*[124]	70.1%
Zhao *et al.*[125]	74.3%
Ours	**91.8%**

neighbor classification on the Places validation set in Figure 4.6. We report the leave-one-out cross-validation (LOOCV) accuracy for different values of k. This can be

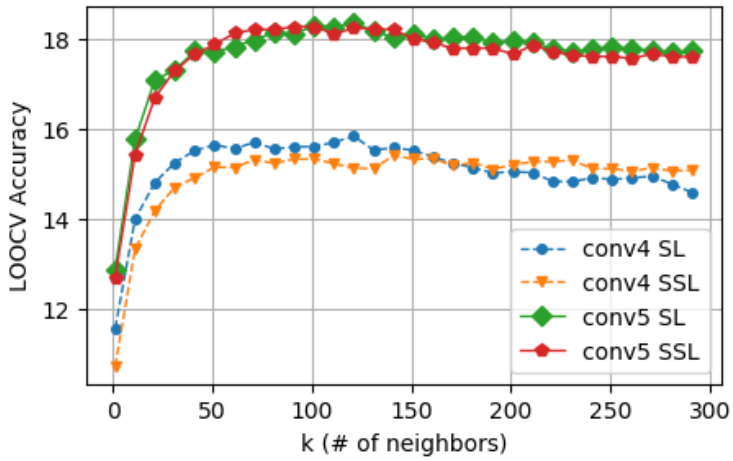


FIGURE 4.6: **Nearest-neighbor accuarcy on Places.** We report leave-one-out cross-validation (LOOCV) accuracy for k-nearest neighbor classifiers on the Places validation set. We compare our self-supervised transformation classifier's performance against features of a supervised network for different values of k. Both networks were pre-trained on ImageNet.

done efficiently by computing $(k+1)$-nearest neighbors using the complete dataset and by excluding the closest neighbor for each query. The concatenation of features from five 128×128 crops (extracted at the resolution we trained the networks on) is used for nearest neighbors. The features are standardized, and cosine similarity is used for nearest neighbor computation.

4.5 Discussion

This chapter introduced the self-supervised feature learning task of discriminating natural images from images transformed through local inpainting, image warping, and rotations. This approach is based on the principle that trained features generalize better when their task requires recognizing natural image's global pixel statistics. Substantial experimental evaluation supports this principle: trained features achieve state-of-the-art performance on several transfer learning benchmarks (Pascal VOC, STL-10, CelebA, and ImageNet) and even slightly outperform supervised training on Places. The recognition of distinct transformations of visual patterns provides a flexible framework for designing self-supervised learning tasks. In the next chapter, we explore a similar methodology on videos by learning to recognize transformations of the temporal domain.

FIGURE 4.7: **Comparison of nearest neighbor retrieval.** The leftmost column shows the query image. Odd rows: Retrievals with our features. Even rows: Retrievals with features learned using ImageNet labels. We computed nearest neighbors on the validation set of ImageNet with `conv5` features using cosine similarity.

Chapter 5

Learning Motion via Temporal Transformations

"A body in motion tends to stay in motion unless acted on by an outside force."

— Isaac Newton

In Chapters 3 and 4, we focused on self-supervised learning tasks that learn visual representations of single images. Single images carry crucial information about a scene, such as an object's appearance (*e.g.*, shape and texture). However, by observing only static images, we miss an essential aspect of the visual world: the temporal evolution of scenes. When we observe a temporal sequence of image frames, *i.e.*, a video, it is possible to understand much more about the objects and the scene. In fact, by moving, objects reveal their 3D shape (through a change in the occlusions), their behavior (how they move due to the laws of physics or their inner mechanisms), and their interaction with other objects (*e.g.*, how they deform, break, or clamp). However, learning such information is non-trivial. Even when labels related to motion categories are available (such as in action recognition), there is no guarantee that the trained model will learn the desired information. Instead, it might merely focus on a single iconic frame and recognize a key pose or some notable features strongly correlated to the action [2].

This chapter describes how to build representations of videos that capture more than the information contained in a single frame. To this end, we design tasks that learn an accurate model of motion by learning to distinguish an unprocessed video from a temporally-transformed one. Since similar frames are present in both the unprocessed and transformed sequence, the only piece of information that discriminates between them is their temporal evolution. This idea has been exploited in the past [79], [81], [82], [84], [151]. Furthermore, it is also related to work in time-series analysis, where dynamic time warping is used as a distance for temporal sequences [152].

We analyze different temporal transformations and evaluate how learning to distinguish them yields a useful representation for classifying videos into meaningful action categories. Our main finding is that the most effective temporal distortions require observing the largest number of frames to be identified. For instance, substituting the second half of a video with its first half in reverse order can be detected by comparing the three frames around the temporal symmetry. In contrast, distinguishing when a video is played backward from when it is played forward [84] may

Material from: S. Jenni, G. Meishvili, and P. Favaro. "Video Representation Learning by Recognizing Temporal Transformations." In *European Conference on Computer Vision 2020*.

FIGURE 5.1: **Learning from temporal transformations**. The frame number is indicated below each image. (a)-(d) **speed**: skip 0 frames (a), 1 frame (b), 3 frames (c), or 7 frames (d). (e) **random**: frame permutation. (f) **periodic**: forward-backward motion (at the selected speed). (g) **warp**: variable frame skipping while preserving the order.

require observing many frames. Thus, one can achieve powerful video representations through the pseudo-task of classifying temporal distortions that differ in their long-range motion dynamics. Towards this goal, we investigate four different temporal transformations of a video, which we illustrate in Figure 5.1:

1. **Speed**: Select a subset of frames with uniform sub-sampling (*i.e.*, with a fixed number of frames in between every pair of selected frames) while preserving the order in the original sequence;

2. **Random**: Select a random permutation of the frame sequence;

3. **Periodic**: Select a random subset of frames in their natural (forward) temporal order and then a random subset in the backward order;

4. **Warp**: Select a subset of frames with a random sub-sampling (*i.e.*, with a random number of frames in between every pair of selected frames) while preserving the natural (forward) order in the original sequence.

We use these transformations to verify and illustrate the hypothesis that learning to distinguish them from one another (and the original sequence) is useful to build a representation of videos for action recognition. For simplicity, we train a neural network that takes videos of the same duration as input and outputs two probabilities: one is about which one of the above temporal transformations the input sequence

is likely to belong to and the second is about identifying the correct speed of the chosen **speed** transformation.

In the experiments section, we transfer features of standard 3D-CNN architectures (C3D [153], 3D-ResNet [154], and R(2+1)D [155]) pre-trained through the above pseudo-task to standard action recognition data sets such as UCF101 and HMDB51, with improved performance compared to prior works. We also show that features learned through our proposed pseudo-task capture long-range motion better than features obtained through supervised learning.

Contributions. Our contributions can be summarized as follows: 1) we introduce a novel self-supervised learning task to learn video representations by distinguishing temporal transformations; 2) we study the discrimination of the following novel temporal transformations: **speed**, **periodic** and **warp**; 3) we show that our features are a better representation of motion than features obtained through supervised learning and achieve state-of-the-art transfer learning performance on action recognition benchmarks.

5.1 Background

Several prior works have explored the temporal ordering of video frames as a supervisory signal. Misra *et al.* [79] first explored this direction by recognizing shuffled sequences, which corresponds to recognizing the **random** transformation in our framework. Another related work focuses on predicting the arrow of time in videos [84], *i.e.*, if frames are played in forward or backward order. We also observed three concurrent publications that also exploit the playback speed as a self-supervision signal [85], [86], [87]. In contrast to these prior works, our work studies a broader range of temporal transformations. This encourages the learning of longer-range motion patterns since many of the temporal transformations we consider are not distinguishable based only on short-range motion patterns. As a result, our experiments show that our features' increased temporal extent (in frames) correlates to the transfer learning performance in action recognition. We refer to Chapter 2 for more discussion of prior unsupervised approaches on video.

5.2 Learning Video Dynamics

Recent work [89] illustrated how careful learning of motion statistics led to a video representation with good transfer performance on several tasks and data sets. The learning of motion statistics was made explicit by extracting optical flow between frame pairs, computing flow changes, and then identifying the region where some key attributes (*e.g.*, maximum magnitude and orientation) of the time-averaged flow-change occurred. In this chapter, we also aim to learn from motion statistics. However, we focus our attention entirely on the temporal evolution without specifying motion attributes of interest or defining a task based on appearance statistics. We hypothesize that these important aspects could be implicitly learned and exploited by the neural network to solve the lone task of discriminating temporal transformations of a video. Our objective is to encourage the neural network to capture long-range motion statistics well. To do so, we train the network to discriminate videos where the image content is preserved, but not the temporal evolution. For example, we ask the network to distinguish a video at the original frame rate from

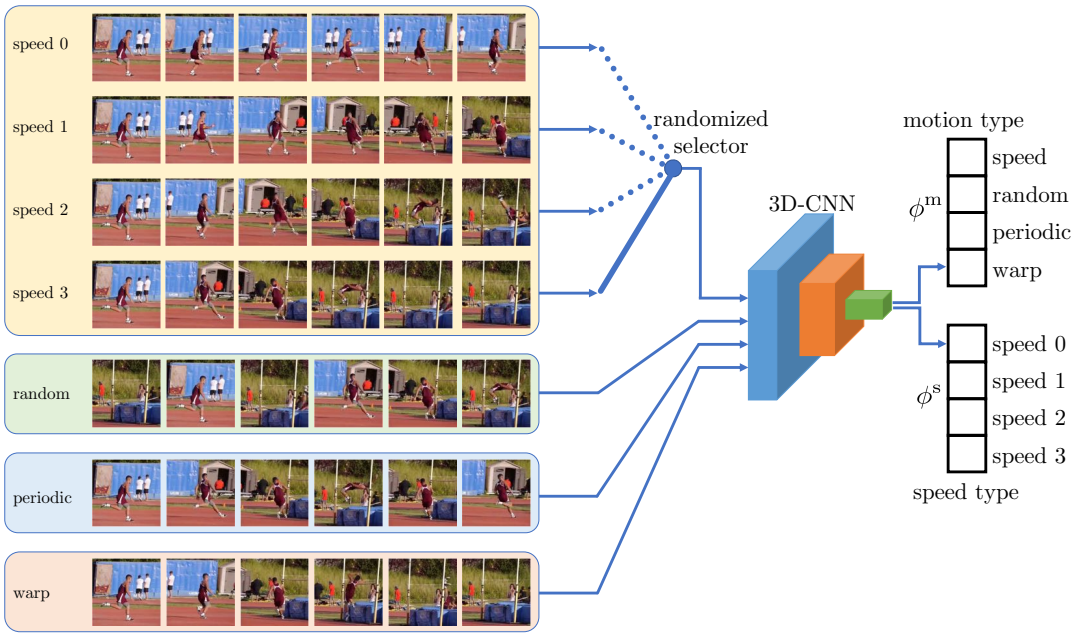


FIGURE 5.2: **Training a 3D-CNN to distinguish temporal transformations.** In each mini-batch, we select a video speed (out of 4 possible choices), *i.e.*, how many frames are skipped in the original video. Then, the 3D-CNN receives as input mini-batch a mixture of 4 possible transformed sequences: **speed** (with the chosen frame skipping), **random**, **periodic** and **warp**. The network outputs the probability of which motion type a sequence belongs to and the probability of which speed type the speed-transformed sequence has.

when it is played four times faster. Due to the laws of physics, *executing* the same task at different speeds leads to different motion dynamics compared to when a video is just *played* at different speeds (*e.g.*, compare marching vs. walking played at a higher speed). Capturing the subtleties of these motions' dynamics requires more than estimating motion between 2 or 3 frames. Moreover, these subtleties are specific to the moving object, and thus they require object detection and recognition.

In our approach, we transform videos by sampling frames according to different schemes, which we call *temporal transformations*. To support our learning hypothesis, we analyze transformations that require short- (*i.e.*, temporally local) and long-range (*i.e.*, temporally global) video understanding. As we will illustrate in the experiments section, short-range transformations yield representations that transfer to action recognition with lower performance than long-range ones.

5.2.1 Transformations of Time

Figure 5.2 illustrates how we train our neural network (a 3D-CNN [153]) to build a video representation (with 16 frames). In this section, we focus on the inputs to the network. As mentioned above, our approach is based on distinguishing different temporal transformations. We consider four fundamental types of transformations: Speed changes, random temporal permutations, periodic motions, and temporal warp changes. These transformations boil down to picking a sequence of temporal indices to sample the videos in our data set. $\mathcal{V}_\kappa^\tau \subset \{0, 1, 2, \dots\}$ denotes the chosen subset of indices of a video based on the transformation $\tau \in \{0, 1, 2, 3\}$ and with

speed κ.

Speed ($\tau = 0$): In this first type, we artificially change the video frame rate, *i.e.*, its playing speed. We achieve that by skipping a different number of frames. We consider 4 cases, **speed 0, 1, 2, 3** corresponding to $\kappa = 0,\ 1,\ 2,\ 3$ respectively, where we skip $2^\kappa - 1$ frames. The resulting playback speed of **speed κ** is therefore 2^κ times the original speed. In the generation of samples for the neural network training, we first uniformly sample $\kappa \in \{0, 1, 2, 3\}$, the playback speed, and then use this parameter to define other transformations. This sequence is used in all experiments as one of the categories against either other speeds or one of the other transformations below. The index sequence $\mathcal{V}_\kappa^0$ is thus $\rho + [0, 1 \cdot 2^\kappa, 2 \cdot 2^\kappa, \ldots, 15 \cdot 2^\kappa]$, where ρ is a random initial index.

Random ($\tau = 1$): In this second temporal transformation, we randomly permute the indices of a sequence without skipping frames. We fix $\kappa = 0$ to ensure that the maximum frame skip between two consecutive frames is not too dissimilar to other transformations. This case is used as a reference, as random permutations can often be detected by observing only a few nearby frames. Indeed, in the Experiments section, one can see that this transformation yields a low transfer performance. The index sequence $\mathcal{V}_0^1$ is thus $\rho + \text{permutation}([0, 1, 2, \ldots, 15])$. This transformation is similar to that of the pseudo-task of Misra *et al.* [79].

Periodic ($\tau = 2$): This transformation synthesizes motions that exhibit approximate periodicity. To create such artificial cases we first pick a point $2 \cdot 2^\kappa < s < 13 \cdot 2^\kappa$ where the playback direction switches. Then, we compose a sequence with the following index sequence: 0 to s and then from $s - 1$ to $2s - 15 \cdot 2^\kappa$. Finally, we sub-sample this sequence by skipping $2^\kappa - 1$ frames. Notice that the randomization of the midpoint s in the case of $\kappa > 0$ yields pseudo-periodic sequences, where the frames in the second half of the generated sequence often do not match the frames in the first half of the sequence. The index sequence $\mathcal{V}_\kappa^2$ is thus $\rho + [0, 1 \cdot 2^\kappa, 2 \cdot 2^\kappa, \ldots, \bar{s} \cdot 2^\kappa, (\bar{s} - 1) \cdot 2^\kappa + \delta, \ldots, (2\bar{s} - 15) \cdot 2^\kappa + \delta])$, where $\bar{s} = \lfloor s / 2^\kappa \rfloor$, $\delta = s - \bar{s} \cdot 2^\kappa$, and $\rho = \max(0, (15 - 2\bar{s}) \cdot 2^\kappa - \delta)$.

Warp ($\tau = 3$): In this transformation, we pick a set of 16 ordered indices with a non-uniform number of skipped frames between them (we consider sampling any frame, so we let $\kappa = 0$). In other words, between any of the frames in the generated sequence, we have a random number of skipped frames, each chosen independently from the set $\{0, \ldots, 7\}$. This transformation creates a warping of the temporal dimension by varying the playback speed from frame to frame. To construct the index sequence $\mathcal{V}_0^3$ we first sample the frame skips $s_j \in \{0, \ldots, 7\}$ for $j = 1, \ldots, 15$ and set $\mathcal{V}_0^3$ to $\rho + [0, s_1, s_1 + s_2, \ldots, \sum_{j=1}^{15} s_j]$.

5.2.2 Training

Let ϕ denote our network, and let us denote with ϕ^{m} (`motion`) and ϕ^{s} (`speed`) its two softmax outputs (see Figure 5.2). To train ϕ we optimize the following loss

$$
\begin{aligned}
&-\mathrm{E}_{\kappa \sim \mathcal{U}[0,3], p \in \mathcal{V}_\kappa^0, q \in \mathcal{V}_0^1, s \in \mathcal{V}_\kappa^2, t \in \mathcal{V}_0^3, x} \Big[\log \left(\phi_0^{\mathrm{m}}(x_p)\, \phi_1^{\mathrm{m}}(x_q)\, \phi_2^{\mathrm{m}}(x_s)\, \phi_3^{\mathrm{m}}(x_t) \right) \Big] \\
&-\mathrm{E}_{\kappa \sim \mathcal{U}[0,3], p \in \mathcal{V}_\kappa^0, x} \big[\log \left(\phi_\kappa^{\mathrm{s}}(x_p) \right) \big]
\end{aligned}
\tag{5.1}
$$

TABLE 5.1: **Ablation experiments.** We train a 3D-CNN to distinguish different sets of temporal transformations. The learned features' quality is evaluated through transfer learning for action recognition on UCF101 (with frozen convolutional layers) and HMDB51 (with fine-tuning of the whole network).

Pre-Training Signal	`speed` loss	**UCF101** (conv frozen)	**HMDB51** (conv fine-tuned)
Action Labels UCF101	-	60.7%	28.8%
Speed	YES	49.3%	32.5%
Speed + Random	NO	44.5%	31.7%
Speed + Periodic	NO	40.6%	29.5%
Speed + Warp	NO	43.5%	32.6%
Speed + Random	YES	55.1%	33.2%
Speed + Periodic	YES	56.5%	36.1%
Speed + Warp	YES	55.8%	36.9%
Speed + Random + Periodic	NO	47.4%	30.1%
Speed + Random + Warp	NO	54.8%	36.6%
Speed + Periodic + Warp	NO	50.6%	36.4%
Speed + Random + Periodic	YES	60.0%	37.1%
Speed + Random + Warp	YES	60.4%	39.2%
Speed + Periodic + Warp	YES	59.5%	39.0%
Speed + Random + Periodic + Warp	NO	54.2%	34.9%
Speed + Random + Periodic + Warp	YES	60.6%	38.0%

where x is a video sample, the sub-index denotes the set of frames. This loss is the cross entropy both for `motion` and `speed` classification (see Figure 5.2).

5.2.3 Implementation

Following prior work [89], we use the smaller variant of the C3D architecture [153] for the 3D-CNN transformation classifier in most of our experiments. Training was performed using the AdamW optimizer [143] with parameters $\beta_1 = 0.9, \beta_2 = 0.99$ and a weight decay of 10^{-4}. The initial learning rate was set to $3 \cdot 10^{-4}$ during pre-training and $5 \cdot 10^{-5}$ during transfer learning. We decayed the learning rate by a factor of 10^{-3} throughout training using cosine annealing [144] both during pre-training and transfer learning. We use batch-normalization [118] in all but the last layer. Mini-batches are constructed such that all the different coarse time warp types are included for each sampled training video. The batch size is set 28 examples (including all the transformed sequences). The speed type is uniformly sampled from all the considered speed types. Since not all the videos allow sampling of all speed types (due to their short video duration), we limit the speed type range to the maximal possible speed type in those examples. We use the standard pre-processing for the C3D network. In practice, video frames are first resized to 128×171 pixels, from which we extract random crops of size 112×112 pixels. We also apply random horizontal flipping of the video frames during training. We use only the raw, unfiltered RGB video frames as input to the motion classifier and do not use optical-flow or other auxiliary signals.

5.3 Experiments

Datasets and Evaluation. In our experiments, we consider three datasets. Kinetics [156] is a large human action dataset consisting of around 500K videos. Video clips are collected from YouTube and span 600 human action classes. We use the training split for self-supervised pre-training. UCF101 [157] contains around 13K video clips spanning 101 human action classes. HMDB51 [158] contains around 5K videos belonging to 51 action classes. Both UCF101 and HMDB51 come with three pre-defined train and test splits. We report the average performance over all splits for transfer learning experiments. We use UCF101 train split 1 for self-supervised pre-training. For transfer learning experiments, we skip three frames corresponding to transformation **speed 2**. For the evaluation of action recognition classifiers in transfer experiments, we use as prediction the maximum class probability averaged over all center-cropped sub-sequences for each test video. More details are provided in the supplementary material.

Understanding the Impact of the Temporal Transformations. We perform ablation experiments on UCF101 and HMDB51, where we vary the number of different temporal transformations the 3D-CNN is trained to distinguish. The 3D-CNN is pre-trained for 50 epochs on UCF101 with our self-supervised learning task. We then perform transfer learning for action recognition on UCF101 and HMDB51. On UCF101, we freeze the weights of the convolutional layers and train three randomly initialized fully-connected layers for action recognition. This experiment treats the transformation classifier as a fixed video feature extractor. On HMDB51, we fine-tune the whole network, including convolutional layers, on the target task. This experiment, therefore, measures the quality of the network initialization obtained through self-supervised pre-training. In both cases, we again train for 50 epochs on the action recognition task. The results of the ablations are summarized in Table 5.1. For reference, we also report the performance of network weights learned through supervised pre-training on UCF101.

We observe that when considering the impact of a single transformation across different cases, the types **warp** and **speed** achieve the best transfer performance. With the same analysis, the transformation **random** leads to the worst transfer performance on average. We observe that **random** is also the most straightforward transformation to detect (based on training performance – not reported). As can be seen in Figure 5.1 (e), this transformation can lead to drastic differences between consecutive frames. Such examples can therefore be easily detected by only comparing pairs of adjacent frames. In contrast, the motion type **warp** can not be distinguished based solely on two adjacent frames and requires modeling long-range dynamics. We also observe that distinguishing a larger number of transformations generally leads to increased transfer performance. The effect of the **speed** type classification is quite noticeable. It leads to a very significant transfer performance increase in all cases. This is also the most challenging pseudo task (based on the training performance – not reported). Recognizing the speed of actions is challenging since different action classes naturally exhibit widely different motion speeds (*e.g.*, "applying makeup" vs. "biking"). This task might often require a deeper understanding of the physics and objects involved in the video. Notice also that our pre-training strategy leads to a better transfer performance on HMDB51 than supervised pre-training using action labels. This suggests that the video dynamics learned through our pre-training generalize well to action recognition and that such dynamics are not well captured through the lone supervised action recognition.

TABLE 5.2: **Comparison to prior work on self-supervised video representation learning.** Whenever possible, we compare to results reported with the same data modality we used, *i.e.*, unprocessed RGB input frames. * are our reimplementations.

Method	Ref	Network	Train Dataset	UCF101	HMDB51
Shuffle&Learn [79]	[79]	AlexNet	UCF101	50.2%	18.1%
O3N [81]	[81]	AlexNet	UCF101	60.3%	32.5%
AoT [84]	[84]	VGG-16	UCF101	78.1%	-
OPN [82]	[82]	VGG-M-2048	UCF101	59.8%	23.8%
DPC [77]	[77]	3D-ResNet34	Kinetics	75.7%	35.7%
SpeedNet [86]	[86]	S3D-G	Kinetics	81.1%	48.8%
AVTS [88] (RGB+audio)	[88]	MC3	Kinetics	85.8%	56.9%
Shuffle&Learn [79]*	-	C3D	UCF101	55.8%	25.4%
3D-RotNet [76]*	-	C3D	UCF101	60.6%	27.3%
Clip Order [159]	[159]	C3D	UCF101	65.6%	28.4%
Spatio-Temp [89]	[89]	C3D	UCF101	58.8%	32.6%
Spatio-Temp [89]	[89]	C3D	Kinetics	61.2%	33.4%
3D ST-puzzle [83]	[83]	C3D	Kinetics	60.6%	28.3%
Ours	-	C3D	UCF101	68.3%	38.4%
Ours	-	C3D	Kinetics	**69.9%**	**39.6%**
3D ST-puzzle [83]	[83]	3D-ResNet18	Kinetics	65.8%	33.7%
3D RotNet [76]	[76]	3D-ResNet18	Kinetics	66.0%	37.1%
DPC [77]	[77]	3D-ResNet18	Kinetics	68.2%	34.5%
Ours	-	3D-ResNet18	UCF101	77.3%	47.5%
Ours	-	3D-ResNet18	Kinetics	**79.3%**	**49.8%**
Clip Order [159]	[159]	R(2+1)D	UCF101	72.4%	30.9%
PRP [87]	[87]	R(2+1)D	UCF101	72.1%	35.0%
Ours	-	R(2+1)D	UCF101	**81.6%**	**46.4%**

Transfer to UCF101 and HMDB51. We compare to prior work on self-supervised video representation learning in Table 5.2. A fair comparison to much of the prior work is difficult due to the use of very different network architectures and training and transfer settings. We opted to compare with some commonly used network architectures (*i.e.*, C3D, 3D-ResNet, and R(2+1)D) and re-implemented two prior works [79] and [76] using C3D. We performed self-supervised pre-training on UCF101 and Kinetics. C3D is pre-trained for 100 epochs on UCF101 and 15 epoch on Kinetics. 3D-ResNet and R(2+1)D are pre-trained for 200 epochs on UCF101 and 15 epochs on Kinetics. We fine-tune all the layers for action recognition. Fine-tuning is performed for 75 epochs using C3D and for 150 epochs with the other architectures. When pre-training on UCF101, our features outperform prior work on the same network architectures. Pre-training on Kinetics leads to improvements in all cases.

Long-Range vs Short-Range Temporal Statistics. To illustrate how well our video representations capture motion, we transfer them to other pseudo-tasks that focus on videos' temporal evolution. One task is the classification of the synchronization of video pairs, *i.e.*, how many frames one video is delayed from the other. A second task is the classification of two videos into which one comes first temporally. These two tasks are illustrated in Figure 5.3.

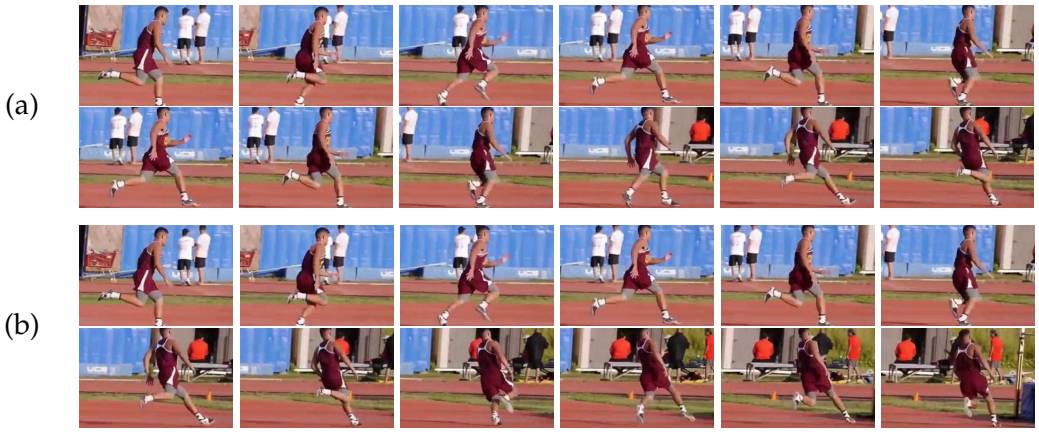


FIGURE 5.3: **Time-related pseudo-tasks**. (a) Synchronization problem: The network is given two sequences with a time delay (4 frames in the example), and a classifier is trained to determine the delay. (b) The before-after problem: The network is given two non-overlapping sequences, and it needs to determine which comes first (the bottom sequence after the top one in the example).

TABLE 5.3: **Time-related pseudo-tasks.** We examine how well features from different pre-training strategies can be transferred to time-related tasks on videos. As tasks, we consider synchronizing two overlapping videos and the temporal ordering of two non-overlapping videos. We report the accuracy for both tasks on the UCF101 test set and report Mean Absolute Error (MAE) for the synchronization task. * are our reimplementations.

	Sync.		Before-After
Method	Accuracy	MAE	Accuracy
Action Labels (UCF101)	36.7%	1.85	66.6%
3D-RotNet [76]*	28.0%	2.84	57.8%
Shuffle&Learn [79]*	39.0%	1.89	69.8%
Ours	**42.4**%	**1.61**	**76.9**%

For the synchronization task, two temporally overlapping video sequences x_1 and x_2 are separately fed to the pre-trained C3D network to extract features $\psi(v_1)$ and $\psi(v_2)$ at the `conv5` layer. These features are then fused through $\psi(v_1) - \psi(v_2)$ and fed as input to a randomly initialized classifier consisting of three fully-connected layers trained to classify the offset between the two sequences. We consider random offsets between the two video sequences in the range of -6 to +6. For the second task we construct a single input sequence by sampling two non-overlapping 8 frame sub-sequences x_{i1} and x_{i2}, where x_{i1} comes before x_{i2}. The network inputs are then either (x_{i1}, x_{i2}) for class "before" or (x_{i2}, x_{i1}) for the class "after". We reinitialize the fully-connected layers in this case as well.

In Table 5.3 we compare the performance of different pre-training strategies on the time-related pseudo-tasks. We see that our self-supervised features perform better at these tasks than supervised features and other self-supervised features, thus showing that they capture the temporal dynamics in the videos well.

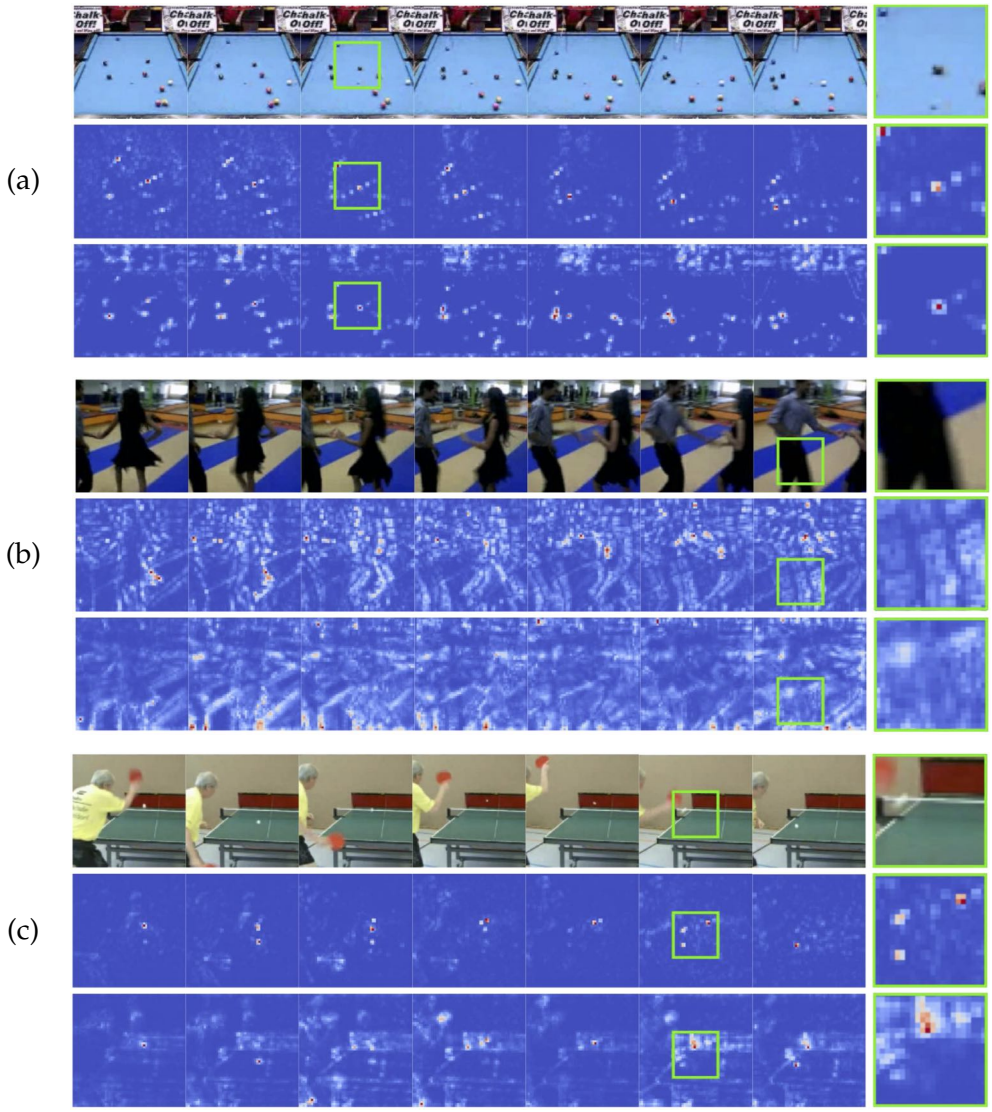


FIGURE 5.4: **Visualization of active pixels**. The first row in each block corresponds to the input video. Rows two and three show the output of our adaptation of Guided Backpropagation [160] when applied to a network trained through self-supervised learning and supervised learning, respectively. In all three cases, we observe that the self-supervised network focuses on image regions of moving objects or persons. In (a), we can also observe how the self-supervised model is detecting long-range dynamics. On the other hand, the supervised model focuses a lot on static frame features in the background.

Visualization. What are the attributes, factors, or features of the videos that self-supervised and supervised models are extracting to perform the final classification? To examine what the self-supervised and supervised models focus on, we apply Guided Backpropagation [160]. This method allows us to visualize which part of the input has the most impact on the model's final decision. We slightly modify the procedure by subtracting the median values from every frame of the gradient video and taking the result's absolute value. We visualize the pre-trained self-supervised, and supervised models on several test samples from UCF101. As one can see in Figure 5.4, a model pre-trained on our self-supervised task tends to ignore the background and focuses on persons performing an action and on moving objects.

Models trained with supervised learning, on the other hand, tend to focus more on the appearance of foreground and background. Another observation we make is that the self-supervised model identifies the location of moving objects/people in the past and future frames. This is visible in row number 2 of blocks *(a)* and *(c)* of Figure 5.4, where the network tracks the possible locations of the moving ping-pong and billiard balls, respectively. A possible explanation for this observation is that our self-supervised task only encourages the learning of dynamics. The appearance of non-moving objects or static backgrounds is not useful to solve the pretext task and is thus ignored.

Learning Dynamics vs. Frame Features. The visualizations in Figure 5.4 indicate that features learned through motion discrimination focus on the dynamics in videos and not so much on static content present in single frames (*e.g.*, background) when compared to supervised features. To further investigate how much the features learned through the two pre-training strategies rely on motion, we performed experiments where we remove all the dynamics from videos. To this end, we create input videos by replicating a single frame 16 times (resulting in a still video) and train the three fully-connected layers on `conv5` features for action classification on UCF101. Features obtained through supervised pre-training achieve an accuracy of 18.5% (vs. 56.5% with dynamics), and features from our self-supervised task achieve 1.0% (vs. 58.1%). Although this experiment's setup is somewhat contrived (since the input domain is altered), it still illustrates that our features rely almost exclusively on motion instead of features present in single frames. This can be advantageous since motion features might generalize better to variations in the background appearance in many cases.

Nearest-Neighbor Evaluation. We perform an additional quantitative evaluation of the learned video representations via the nearest-neighbor retrieval. The features are obtained by training a 3D-ResNet18 network on Kinetics with our pseudo-task and are chosen as the output of the global average pooling layer, which corresponds to a vector of size 512. For each video, we extract and average features of 10 temporal crops. To perform the nearest-neighbor retrieval, we first normalize the features using the training set statistics. Cosine similarity is used as the metric to determine the nearest neighbors. We follow the evaluation proposed by [49] on UCF101. Query videos are taken from test split 1, and all the videos of train split 1 are considered as retrieval targets. A query is considered correctly classified if the k-nearest neighbors contain at least one video of the correct class (*i.e.*, same class as the query). We report the mean accuracy for different values of k and compare to prior work in Table 5.4. Our features achieve state-of-the-art performance.

Qualitative Nearest-Neighbor Results We show some examples of nearest neighbor retrievals in Figure 5.5. Frames from the query test video are shown in the leftmost block of three columns. The second and third blocks of three columns show the top two nearest neighbors from the training set. We observe that the retrieved examples often capture the semantics of the query well. This is the case even when the action classes do not agree (*e.g.*, last row).

TABLE 5.4: **Video retrieval performance on UCF101.** We compare to prior work in terms of k-nearest neighbor retrieval accuracy. Query videos are taken from test split 1, and retrievals are computed on train split 1. A query is correctly classified if the query class is present in the top-k retrievals. We report mean retrieval accuracy for different values of k.

Method	Network	Top1	Top5	Top10	Top20	Top50
Jigsaw [4]	AlexNet	19.7	28.5	33.5	40.0	49.4
OPN [82]	AlexNet	19.9	28.7	34.0	40.6	51.6
Büchler *et al.* [49]	AlexNet	25.7	36.2	42.2	49.2	59.5
Clip Order [159]	R3D	14.1	30.3	40.0	51.1	66.5
SpeedNet [86]	S3D-G	13.0	28.1	37.5	49.5	65.0
PRP [87]	R3D	22.8	38.5	46.7	55.2	69.1
Ours	3D-ResNet18	**26.1**	**48.5**	**59.1**	**69.6**	**82.8**

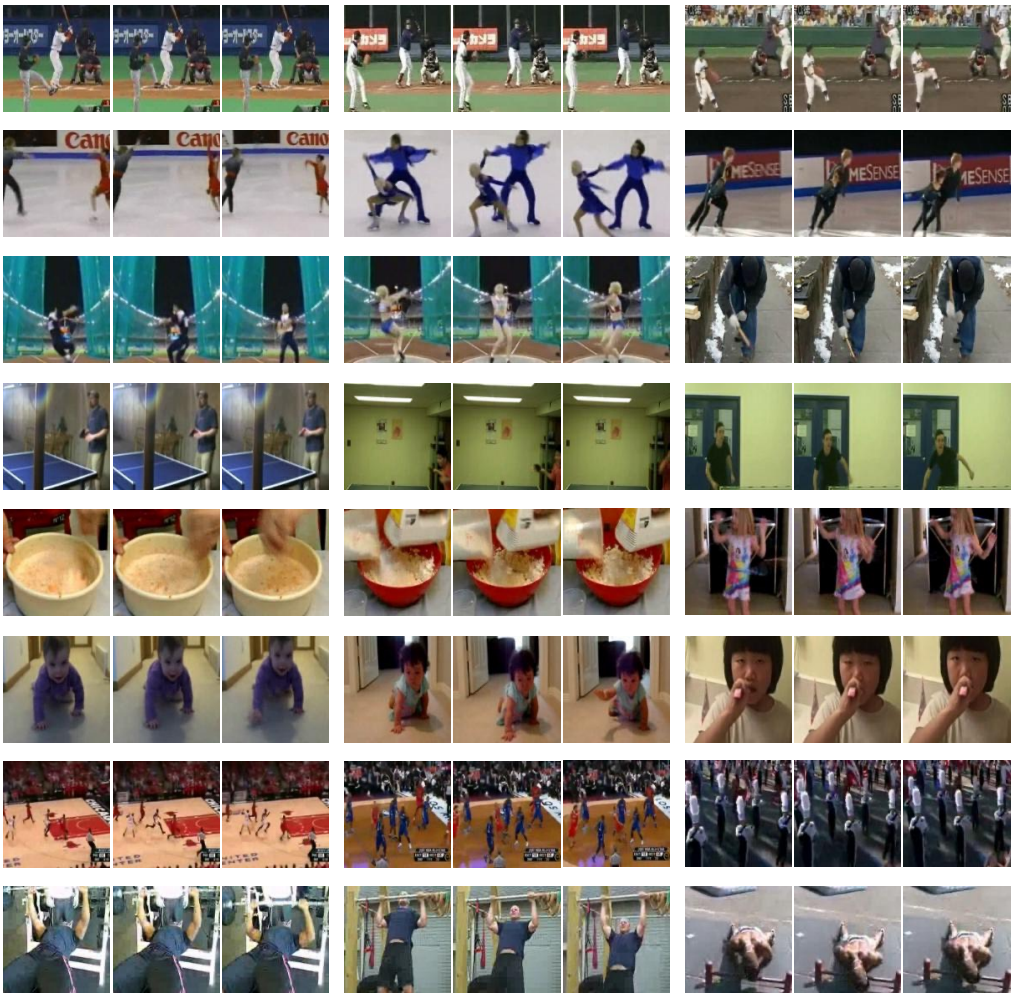


FIGURE 5.5: **Examples of retrievals in UCF101.** Leftmost block of 3 columns: Frames from the query sequences. Second and third blocks of 3 columns: Frames from the two nearest neighbors.

5.4 Discussion

This chapter has introduced a novel task for the self-supervised learning of video representations by distinguishing between different types of temporal transformations. This learning task is based on the principle that recognizing a transformation of time requires an accurate model of the underlying natural video dynamics. This idea is supported by experiments that demonstrate that features learned by distinguishing temporal transformations capture video dynamics better than supervised learning. Such features generalize well to classic vision tasks such as action recognition or time-related tasks such as video synchronization.

The approach in this chapter was based on single-view videos. In the following chapter, we will explore yet another data modality by learning image representations that capture 3D properties of the scene from multi-view video data.

Chapter 6

Self-Supervised Learning of 3D Pose Features

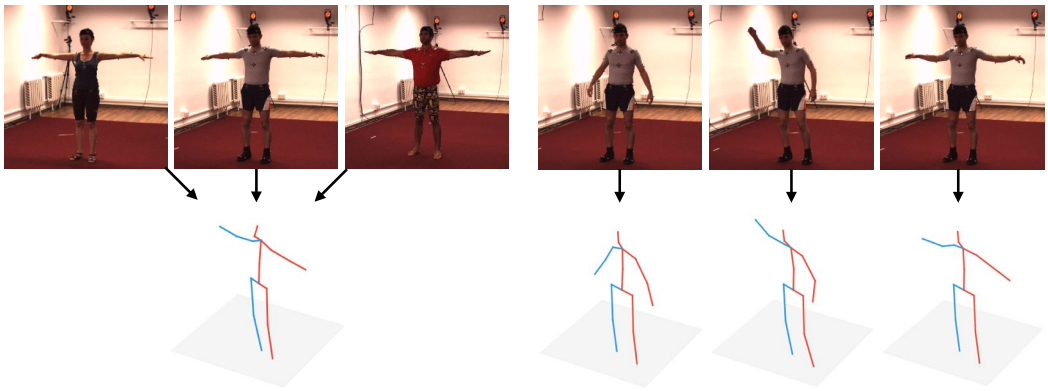
"Synchronicity is an ever present reality for those who have eyes to see."

— Carl Jung

The ability to accurately reconstruct the 3D human pose from a single real image opens a wide variety of applications, including computer graphics animation, postural analysis and rehabilitation, human-computer interaction, and image understanding [161]. State-of-the-art methods for monocular 3D human pose estimation employ neural networks. They require large training datasets where each sample is a pair consisting of an image and its corresponding 3D pose annotation [162]. The collection of such datasets is expensive, time-consuming, and because it requires controlled lab settings, the diversity of motions, viewpoints, subjects, appearance, and illumination, is limited (see Figure 6.1). Ideally, to maximize diversity, one should collect data in the wild. However, in this case, precise 3D annotation is challenging to obtain and might require costly human intervention.

In this chapter, we overcome the above limitations via *self-supervised learning*. Our objective is to build a latent representation of an image that can successfully transfer to 3D human pose estimation via fine-tuning. The performance of the transferred representations depends on how related the self-supervised pretext task is to the target task. Thus, to build latent representations relevant to 3D human poses, we propose a pretext task that implicitly learns 3D structures. To collect data suitable to this goal, examples from nature point towards multi-view imaging systems. In fact, the visual system in many animal species hinges on the presence of two or multiple eyes to achieve a 3D perception of the world [163], [164]. 3D perception is often exemplified by considering two views of the same scene captured simultaneously and by studying the correspondence problem. Thus, we take inspiration from this setting and pose the task of determining if two images have been captured at the same time. In general, the main difference between two views captured simultaneously and when they are not is that a rigid transformation always describes the former but the latter potentially not (*e.g.*, in the presence of articulated or deformable motion or multiple rigid motions). Therefore, we propose as pretext task the detection of *synchronized views*, which translates into a classification of *rigid* versus *non-rigid* motion.

Material from: S. Jenni, and P. Favaro. "Self-Supervised Multi-View Synchronization Learning for 3D Pose Estimation." In *Asian Conference on Computer Vision 2020*.



(A) Desired subject invariance. (B) Desired pose equivariance.

FIGURE 6.1: **Monocular 3D human pose estimation.** An ideal regressor would be able to generalize across subjects regardless of their appearance, as shown in (a), and be sensitive to small pose variations, such as those shown in (b).

As shown in Figure 6.1, we aim to learn a latent representation that can generalize across subjects and that is sensitive to small pose variations. Thus, we train our model on pairs of images where the subject identity is irrelevant to the pretext task. Instead, slight pose variations define the difference between the task categories. To do so, we use two views of the same subject as the synchronized (*i.e.*, rigid motion) pair and two views taken at different (but not too distant) time instants as the unsynchronized (*i.e.*, non-rigid motion) pair.[3] Since these pairs share the same subject (and appearance) in all images, the model cannot use this as a cue to learn the correct classification. Because the pair of unsynchronized images is close in time, the deformation is relatively small and forces the model to discriminate small pose variations. Furthermore, to make the representation sensitive to the human poses' left-right symmetries, we also introduce in the pretext task the classification of two synchronized views into *horizontally flipped* or not as a second goal. This formulation of the SSL task allows to potentially train a neural network on data captured in the wild by merely using a synchronized multi-camera system. As we show in our experiments, the learned representation successfully embeds 3D human poses, and it further improves if we also remove the background from the images. We train and evaluate our SSL pre-training on the Human3.6M dataset [162], and find that it yields state-of-the-art results when compared to other methods under the same training conditions. We demonstrate quantitatively and qualitatively that our trained model is sensitive to small pose variations and can generalize across subjects. Finally, we believe that this approach could easily be incorporated in other methods that exploit additional labeled data (*e.g.*, 2D poses).

Contributions. Our contributions are: 1) a novel self-supervised learning task for multi-view data to recognize when two views are synchronized and/or flipped; 2) extensive ablation experiments to demonstrate the importance of avoiding shortcuts

[3] If the subject does not move between the two chosen time instants, the unsynchronized pair would also be undergoing a rigid motion and thus create ambiguity in training. However, these cases can be easily spotted as they only require detecting no motion over time. Besides standing still, the probability that a moving subject performs a rigid motion is extremely low. In practice, we found experimentally that a simple temporal sub-sampling was sufficient to avoid these scenarios.

via the static background removal and the effect of different feature fusion strategies; 3) our method achieves state-of-the-art performance on 3D human pose estimation benchmarks.

6.1 Background

In this section, we briefly review literature in unsupervised learning, human pose estimation, and synchronization, that is relevant to our approach.

Unsupervised Learning of 3D. Recent progress in unsupervised learning has shown promising results for learning implicit and explicit generative 3D models from natural images [165], [166], [167]. The focus in these methods is on modeling 3D and not performance on downstream tasks. Our goal is to learn general-purpose 3D features that perform well on downstream tasks such as 3D pose estimation.

Synchronization. Learning the alignment of multiple frames taken from different views is an important component of many vision systems. Classical approaches are based on fitting local descriptors [168], [169], [170]. More recently, methods based on metric learning [171], or that exploit audio [172] have been proposed. We provide a simple learning-based approach by posing the synchronization problem as a binary classification task. Our aim is not to achieve synchronization for its own sake but to learn a useful image representation as a byproduct.

Monocular 3D Pose Estimation. State-of-the-art 3D pose estimation methods make use of large annotated in-the-wild 2D pose datasets [173] and datasets with ground truth 3D pose obtained in indoor lab environments. We identify two main categories of methods: 1) methods that learn the mapping to 3D pose directly from images [174], [175], [176], [177], [178], [179], [180], [181], [182] often trained jointly with 2D poses [183], [184], [185], [186], [187], and 2) methods that learn the mapping of images to 3D poses from predicted or ground truth 2D poses [188], [189], [190], [191], [192], [193], [194], [195], [196]. To deal with the limited amount of 3D annotations, some methods explored the use of synthetic training data [197], [198], [199]. In our transfer learning experiments, we follow the first category and predict 3D poses directly from images. However, we do not use any 2D annotations.

Weakly Supervised Methods. Much prior work has focused on reducing the need for 3D annotations. One approach is weak supervision, where only the 2D annotation is used. These methods are typically based on minimizing the re-projection error of a predicted 3D pose [200], [201], [202], [203], [204], [205], [206]. Some methods require multi-view data to resolve ambiguities and constrain the 3D [200], [201], [202]. Other methods rely on unpaired 3D data used via adversarial training [204], [205]. [203] solely relies on 2D annotation and uses an adversarial loss on random projections of the 3D pose. Our aim is not to rely on a weaker form of supervision (*i.e.*, 2D annotations). Instead, we leverage multi-view data to learn a representation that can transfer with only a few annotated examples to the 3D estimation tasks with a good performance.

Self-Supervised Methods for 3D Human Pose Estimation. Here, we consider approaches that do not make use of any additional supervision, *e.g.*, in the form of 2D pose annotation. These are methods that learn representations on unlabeled data

and can be transferred via fine-tuning using a limited amount of 3D annotation. Rhodin *et al.* [207] learn a 3D representation via novel view synthesis, *i.e.*, by reconstructing one view from another. Their method relies on synchronized multi-view data, knowledge of camera extrinsics, and background images. Mitra *et al.* [208] use metric learning on multi-view data. The distance in feature space for images with the same pose but a different viewpoint is minimized while the distance to hard negatives sampled from the mini-batch is maximized. By construction, the resulting representation is view-invariant (the local camera coordinate system is lost), and the transfer can only be performed in a canonical, rotation-invariant coordinate system. We also exploit multi-view data in our pre-training task. Unlike [207], our task does not rely on knowledge of camera parameters and unlike [208], we successfully transfer our features to pose prediction in the local camera system.

6.2 Unsupervised Learning of 3D Pose-Discriminative Features

Our goal is to build image features that allow us to discriminate different 3D human poses. We achieve this objective by training a network to detect if two views depict the same scene up to a rigid transformation. We consider three different cases: 1) the two views depict the same scene at the same time (rigid transformation); 2) the two views show the same scene, but at a different time (non-rigid transformation is highly likely); 3) the two views show the same scene, but one view is horizontally mirrored (a special case of a non-rigid transformation), which, for simplicity, we refer to as *flipped*.

To train such a network, we assume we have access to synchronized multi-view video data. The dataset consists of N examples (in the Human3.6M dataset $N = S \times A$, where S is the number of subjects and A the number of actions) $\{x^{(i)}_{\nu,t}\}_{i=1,\dots,N}$, where $\nu \in \mathcal{V}^{(i)} = \{1,\dots,\nu^{(i)}\}$ indicates the different views of the i-th example and $t \in \mathcal{T}^{(i)} = \{1,\dots,t^{(i)}\}$ indicates the time of the frame. Let F denote the neural network trained to solve the self-supervised task. We will now describe the different self-supervision signals.

6.2.1 Classifying Synchronized and Flipped Views

To train our neural network we define three types image pairs, each corresponding to a different 3D deformation:

Synchronized Pairs. In this case, the scenes in the two views are related by a rigid transformation. This is the case when the two images are captured at the same time instant, *i.e.*, they are synchronized. The input for this category is a sample pair $\mathbf{x}_p = (x^{(i)}_{\nu_1,t}, x^{(i)}_{\nu_2,t})$, where $i \in \{1,\dots,N\}$, $\nu_1 \neq \nu_2 \in \mathcal{V}^{(i)}$, and $t \in \mathcal{T}^{(i)}$, *i.e.*, a pair with different views taken at the same time.

Unsynchronized Pairs. Pairs of images that are captured at different times (*i.e.*, unsynchronized) are likely to undergo a non-rigid deformation (by assuming that objects are non static between these two time instants). To create such image pairs we sample $\mathbf{x}_n = (x^{(i)}_{\nu_1,t_1}, x^{(i)}_{\nu_2,t_2})$, where $i \in \{1,\dots,N\}$, $\nu_1 \neq \nu_2 \in \mathcal{V}^{(i)}$, and $t_1, t_2 \in \mathcal{T}^{(i)}$ such that $d_{min} < |t_2 - t_1| < d_{max}$, where d_{min} and d_{max} define the range in time for sampling unsynchronized pairs. In our experiments, we

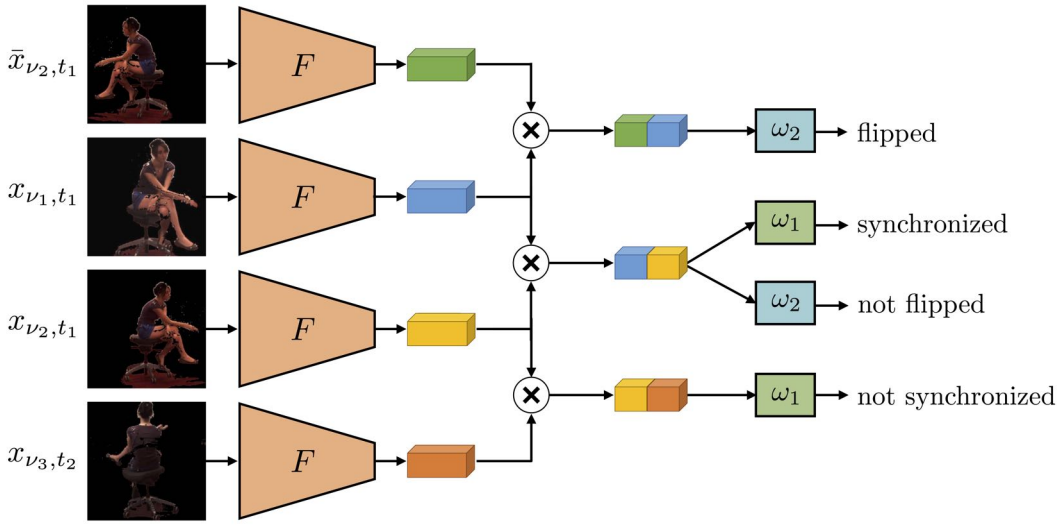


FIGURE 6.2: **An overview of the proposed self-supervised task.** We train a network F to learn image representations that transfer well to 3D pose estimation by learning to recognize if two views of the same scene are *synchronized* and/or *flipped*. Our model uses a siamese architecture. The inputs are pairs of frames of the same scene under different views. Frames are denoted by $x_{\nu,t}$, where ν indicates the viewpoint and t the time ($\bar{x}_{\nu,t}$ denotes a flipped frame). Pairs are classified into whether the frames are *synchronized* or *not synchronized*, and whether one of the frames was *flipped* or not.

set $d_{min} = 4$ and $d_{max} = 128$ and sample uniformly within this range.

Flipped Pairs. The last class of image pairs consists of images from two views which are synchronized, but where one of them has been horizontally mirrored. Let $\bar{x}$ denote the image obtained by applying horizontal mirroring to the sample x. Flipped pairs are defined as $\mathbf{x}_f = (x^{(i)}_{\nu_1,t}, \bar{x}^{(i)}_{\nu_2,t})$ or $\mathbf{x}_f = (\bar{x}^{(i)}_{\nu_1,t}, x^{(i)}_{\nu_2,t})$, where $i \in \{1,\dots,N\}, \nu_1 \neq \nu_2 \in \mathcal{V}^{(i)}$ and $t \in \mathcal{T}^{(i)}$. In the case of approximately symmetric objects (such as with human bodies) and in the absence of background cues (since we mirror the entire image), distinguishing this relative transformation between views requires accurate 3D pose discriminative features.

Although both **unsynchronized** and **flipped** pairs exhibit non-rigid deformations, they have distinct characteristics. In flipped pairs, the 3D pose in the second view is heavily constrained by the one in the first view. In contrast, given a non-negligible temporal gap between the frames, the 3D pose is much less constrained in the case of unsynchronized views.

We define our self-supervised learning task as a combination of the classification of image pairs into *synchronized vs. unsynchronized* and *flipped vs. not flipped*. Let $F_1(\mathbf{x})$ denote the predicted probability of synchronization in $\mathbf{x}$ and $F_2(\mathbf{x})$ the predicted probability of no flipping in $\mathbf{x}$ (both implemented via a sigmoid activation). The final loss function is then given by

$$\mathcal{L}_{SSL} = -\sum_j \log F_1(\mathbf{x}^{(j)}_p) F_2(\mathbf{x}^{(j)}_p) + \log\left(1 - F_1(\mathbf{x}^{(j)}_n)\right) + \log\left(1 - F_2(\mathbf{x}^{(j)}_f)\right). \quad (6.1)$$

An illustration of the training is shown in Figure 6.2. Note that although $\mathbf{x}_f$ is a synchronized pair, we do not optimize the synchronization head F_1 on these examples. The reason for leaving this term undefined is that we equate synchronization to rigid motion, which does not hold in the case of $\mathbf{x}_f$.

6.2.2 Static Backgrounds Introduce Shortcuts

A common issue with self-supervised learning is that neural networks exploit low-level cues to solve the pretext task when allowed to do so. In this case, they often do not learn semantically meaningful features. Such shortcuts have been observed in prior work, where, for instance, chromatic aberration [31] was used as a cue to solve a localization task.

In our self-supervised learning task, we observed that the dataset's shared image background could be used as a shortcut. This issue is most evident in the flipping task: Because the background is shared among all examples in the dataset, it is easier to use texture in the background to determine image flipping. As a result, the network focuses on the background (instead of the foreground object), which decreases its generalization performance. It is further possible that the network learns to detect and associate the person's absolute position by focusing on the background as a cue for the synchronization task.

We propose two solutions: one is *background removal* and the other is *background substitution*. Since the dataset consists of non-moving cameras, we compute the background as the median image per view. Background removal is then performed via simple per-pixel thresholding. Although the resulting separation is not of high quality, we found it sufficient for our purpose. Similarly, background substitution introduces an additional step where the background from a random view is used to replace the original (removed) one. Both approaches are evaluated in our experiments. We expect that background removal would not be necessary for data captured in the wild with synchronized moving cameras or with a large variety of backgrounds.

6.2.3 Implementation

We implement the network F using a Siamese architecture. As a shared backbone architecture Φ we use standard ResNet architectures [119]. Samples $(x^{(j)}_{v_1,t_1}, x^{(j)}_{v_2,t_2})$ are hence encoded into features vectors $(\Phi(x^{(j)}_{v_1,t_1}), \Phi(x^{(j)}_{v_2,t_2}))$. In our experiments we use the output of the global average pooling as the feature representation, *i.e.*, $\Phi(x) \in \mathbb{R}^{2048}$. To fuse the features we choose to use element-wise multiplication followed by a ReLU activation. We found this fusion mechanism to perform better than feature concatenation, which is often used in prior work [4], [31], [135]. The fused feature is fed as input to linear layers ω_1 and ω_2 to produce the final outputs. To summarize, we define $F_i(\mathbf{x}) = \omega_i(\text{ReLU}(\Phi(x^{(j)}_{v_1,t_1}) \odot \Phi(x^{(j)}_{v_2,t_2})))$, where $\odot$ denotes the element-wise product.

The training images use the standard image resolution for ResNet architectures of 224×224 pixels. For a fair comparison with prior work, we extract crops centered on the subject. We perform random horizontal flipping consistently to both frames in the input pair (this operation is performed before preparing the flipped pairs).

6.2.4 Transfer to 3D Human Pose Estimation

We transfer the learned image representation to 3D human pose regression via fine-tuning. The training set in this case consists of a set of image[4] and target pairs $\{(x^{(i)}, y^{(i)})\}_{i=1,\dots,N_p}$ with target 3D joint positions $y^{(i)} \in \mathbb{R}^{n_j \times 3}$ represented in the local camera coordinate system. The number of joints is set to $n_j = 17$ in our experiments on Human3.6M. As in prior work [201], [207], [208], we fix the root joint (at the pelvis) to zero. We effectively only regress the 16 remaining joints and correct this in evaluation metrics by scaling the per-joint errors by $\frac{17}{16}$. We normalize the target poses for training using the per-joint mean and standard deviation computed on the training set as proposed in [182]. During the evaluation, we un-normalize the predicted pose accordingly. To predict the 3D pose we simply add a single linear layer ω_p on the feature encoding $\Phi(x^{(i)})$ resulting in our normalized prediction $P(x^{(i)}) = \omega_p(\Phi(x^{(i)}))$. The network P is trained to minimize the mean-squared error on the training set, *i.e.*, $\mathcal{L}_{pose} = \frac{1}{N_p}\sum_{i=1}^{N_p} \|P(x^{(i)}) - y^{(i)}\|_2^2$.

We again extract crops centered on the subject as in prior work [201], [207], [208]. Since this corresponds to a change of the camera position, we correct for this by rotating the target pose accordingly, *i.e.*, to virtually center the camera on the root joint. We apply random horizontal flipping jointly to the input images, and the corresponding target 3D poses as a form of data augmentation.

6.3 Experiments

Dataset. We perform an extensive experimental evaluation on the Human3.6M dataset [162]. The dataset consists of 3.6 million 3D human poses with the corresponding images. The data is captured in an indoor setting with four synchronized and fixed cameras placed at each corner of the room. Seven subjects perform 17 different actions. As in prior work, we use the five subjects with the naming convention S1, S5, S6, S7, and S8 for training and test on subjects S9 and S11. We filter the training dataset by skipping frames without significant movement. On average, we skip every fourth frame in the training dataset. On the test set, we follow prior work and only evaluate our network on one every 64 frames.

Metrics. To evaluate our method on 3D human pose regression, we adopt the established metrics. We use the Mean Per Joint Prediction Error (MPJPE) and its variants Normalized MPJPE (NMPJPE) and Procrustes MPJPE (PMPJPE). In NMPJPE, the predicted joints are aligned to the ground truth in a least-squares sense with respect to the scale. PMPJE uses Procrustes alignment to align the poses both in terms of scale and rotation.

6.3.1 Ablations

We perform ablation experiments to validate several design choices for our self-supervised learning task. We illustrate the effect of background removal concerning the shortcuts in the case of non-varying backgrounds (as is the case for Human3.6M). We also demonstrate the impact of the two self-supervision signals (flipping and synchronization) by themselves and in combination and explore different feature fusion strategies.

[4] We drop the subscripts ν, t indicating viewpoint and time in this notation.

TABLE 6.1: **Ablation experiments.** We investigate the influence of scene background (a)-(d), the impact of the different self-supervision signals (e)-(g), the use of varying fusion strategies (h)-(k), and the importance of synchronized multiview data (l) and (m). The experiments were performed on Human3.6M using only subject S1 for transfer learning.

	Ablation	MPJPE	NMPJPE	PMPJPE
(a)	Random with background	167.8	147.1	124.5
(b)	Random without background	165.5	145.4	114.5
(c)	SSL with background	138.4	128.1	100.8
(d)	SSL without background	104.9	91.7	78.4
(e)	Only flip	129.9	117.6	101.1
(f)	Only sync	113.7	102.4	80.3
(g)	Both sync & flip	104.9	91.7	78.4
(h)	Fusion concat	180.0	162.5	130.5
(i)	Fusion add	169.4	158.4	122.8
(j)	Fusion diff	108.2	93.9	80.4
(k)	Fusion mult	104.9	91.7	78.4
(l)	Single-view SSL	158.3	142.0	106.9
(m)	Multi-view SSL	104.9	91.7	78.4

The baseline model uses background removal, the combination of both self-supervision signals, and element-wise multiplication for feature fusion. The networks are pre-trained for 200K iterations on all training subjects (S1, S5, S6, S7, and S8) using our SSL task and without using annotations. We use a ResNet-18 architecture [119] and initialize with random weights (*i.e.*, we do not rely on ImageNet pre-training). Transfer learning is performed for an additional 200K iterations using only subject S1 for supervision. We freeze the first three residual blocks and only finetune the remaining layers. Training of the networks was performed using the AdamW optimizer [143] with default parameters and a weight decay of 10^{-4}. We decayed the learning rate from 10^{-4} to 10^{-7} throughout training using cosine annealing [144]. The batch size is set to 16.

We perform the following set of ablation experiments and report transfer learning performance on the test set in Table 6.1:

(a)-(d) Influence of background removal: We explore the influence of background removal on the performance of randomly initialized networks and networks initialized with weights from our self-supervised pre-training. We observe that background removal provides only a relatively small gain in the case of training from scratch. The improvement in the case of our self-supervised learning pre-training is much more substantial. This suggests that the static background introduces shortcuts for the pretext task;

(e)-(f) Combination of self-supervision signals: We compare networks initialized by pre-training (e) only to detect flipping, (f) only to detect synchronization, and (g) on the combination of both. We observe that the synchronization by itself leads to better features compared to flipping alone. This correlates with the pretext task performance, where we observe that the accuracy on the flipping task is higher than for synchronization. Interestingly, the combination of

TABLE 6.2: **Comparison to prior work.** We compare to other prior work. All methods are pre-trained on all the training subjects of Human3.6M. Transfer learning is performed either using all the training subjects or only S1 for supervision. Methods using large amounts of data with 2D pose annotation are *italicized*. * indicates the use of an ImageNet pre-trained ResNet-50. † indicates a viewpoint invariant representation of the 3D pose.

Supervision	Method	MPJPE	NMPJPE	PMPJPE
All	*Chen et al.[202]**	*80.2*	-	*58.2*
	*Kocabas et al.[200]**	*51.8*	*51.6*	*45.0*
	*Rhodin et al.[201]**	*66.8*	*63.3*	*51.6*
	Rhodin *et al.*[201]*	-	95.4	-
	Rhodin (UNet) *et al.*[207]	-	127.0	-
	Rhodin *et al.*[207]*	-	115.0	-
	Mitra *et al.*[208]*†	94.3	92.6	72.5
	Ours	79.5	73.4	59.7
	Ours*	72.6	68.5	54.5
	Ours* (with background)	64.9	62.3	53.5
S1	*Chen et al.[202]**	*91.9*	-	*68.0*
	*Kocabas et al.[200]**	-	*67.0*	*60.2*
	*Rhodin et al.[201]**	-	*78.2*	-
	Rhodin *et al.*[201]*	-	153.3	128.6
	Rhodin (UNet) *et al.*[207]	149.5	135.9	106.4
	Rhodin *et al.*[207]*	131.7	122.6	98.2
	Mitra *et al.*[208]*†	121.0	111.9	90.8
	Ours	104.9	91.7	78.4
	Ours*	101.2	89.6	76.9
	Ours* (with background)	101.4	93.7	82.4

both signals gives a substantial boost. This result suggests that both signals learn complementary features;

(h)-(k) Fusion of features: Different designs for the fusion of features in the Siamese architecture are compared. We consider (h) concatenation, (i) addition, (j) subtraction, and (k) element-wise multiplication. We observe that multiplication performs the best. Multiplication allows a simple encoding of similarity or dissimilarity in feature space solely via the sign of each entry. The fused feature is zero (due to the ReLU) if the two input features do not agree on the sign;

(l)-(m) Necessity of multi-view data: We want to test how important synchronized multi-view data is for our approach. We compare to a variation of our task, where we do not make use of multi-view data. Instead, we create artificial synchronized views via data augmentation. Therefore, the images in the input pairs are from the same camera view but with different augmentations (*i.e.*, random cropping, scaling, and rotations). This setup corresponds to using a 2D equivalent of our self-supervised learning task. We observe drastically decreased feature performance in this case.

6.3.2 Comparison to Prior Work

We compare our method to prior work by Rhodin *et al.* [201], [207] and Mitra *et al.* [208] on Human3.6M. To the best of our knowledge, these are the only methods using the same training settings as in our approach. Other approaches, which we also report, train their networks with additional weak supervision in the form of 2D pose annotation (see Table 6.2).

We train the networks on our self-supervised learning task for 200K iterations on all training subjects (S1, S5, S6, S7, and S8). Our comparison uses two different protocols: 1) **All**, where transfer learning is performed on all the training subjects, and 2) **S1**, where only subject S1 is used for transfer learning. We use two different network architectures: 1) a ResNet-18 initialized with random weights, *i.e.*, no ImageNet pre-training, and 2) a ResNet-50 initialized with ImageNet pre-trained weights. Since prior work transfers to images with backgrounds, we include results with a ResNet-50 fine-tuned on images with backgrounds. To reduce the domain gap between pre-training and transfer and eliminate shortcuts, we pre-train on images with removed backgrounds and images with substituted backgrounds. We keep the ratio without and with substituted backgrounds to 50:50 during pre-training. We freeze the first 3 residual blocks during transfer learning and fine-tune the remaining layers for 200K iterations using a mini-batch size of 16. Training of the networks was again performed using the AdamW optimizer [143] with default parameters, an initial learning rate of 10^{-4} with cosine annealing, and a weight decay of 10^{-4}.

The results and the comparisons to prior work are provided in Table 6.2. We observe that our fine-tuned network generalizes well and outperforms the relevant prior work [201], [207], [208] by a considerable margin. Note that our ResNet-18 is the only method besides [207] that does not rely on any prior supervision. Interestingly, our model with substituted backgrounds performs better than our model with removed backgrounds in protocol **All** and worse in protocol **S1**. The difference is most severe in the scale-sensitive metric MPJPE. We hypothesize that the backgrounds might be useful to learn the correct absolute scale, given several different training subjects. We also list methods that make use of large amounts of additional labeled 2D pose data. Using all the training subjects for transfer learning, we even outperform the weakly supervised approach [202]. We show some qualitative predictions for the two test subjects in Figure 6.3.

6.3.3 Evaluation of the Synchronization Task

Generalization Across Subjects. We qualitatively evaluate how well our model generalizes to new subjects on the synchronization prediction task. The idea is to look for synchronized frames, where the two frames contain different subjects. A query image $x^{(q)}$ of a previously unseen test subject is chosen. We look for the training image $x^{(a)}$ of each training subject that maximizes the predicted probability of synchronization, *i.e.*, $a = \arg\max_i F_1((x^{(q)}, x^{(i)}))$. Note that we did not explicitly train our network for this task and that it only received pairs of frames as input containing the same subject during training.

We show example retrievals in Figure 6.6. We can observe that the retrieved images often show very similar poses to the query image. The retrieved images also cover different viewpoints. The method generalizes surprisingly well across subjects and manages to associate similar poses across subjects. These examples indicate a certain degree of invariance to the person's appearance. As discussed in the Introduction, such an invariance is welcome in 3D human pose estimation and

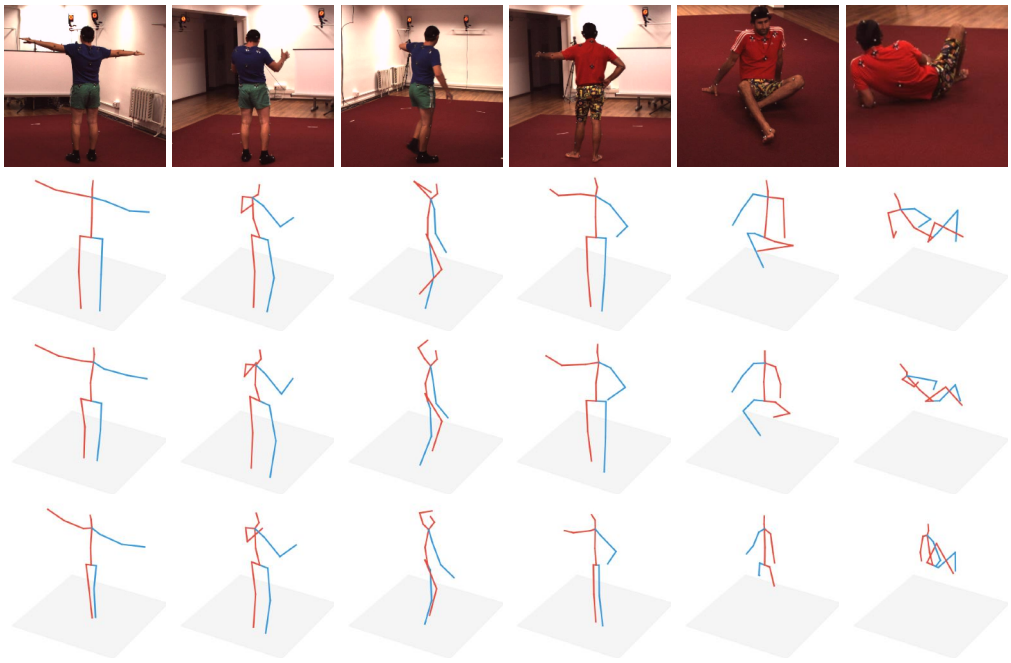


FIGURE 6.3: **Examples of predictions.** We show predictions on unseen test subjects (S9 and S11) of a model pre-trained using our self-supervised learning task and fine-tuned for 3D pose estimation. We provide the test images on the first row, the ground truth poses on the second row, the predictions of a model fine-tuned on all training subjects on the third row, and the predictions of a model fine-tuned only on the subject S1 on the last row.

could explain the good performance in the transfer experiments.

Synchronization Performance on Test Subjects. We evaluate the performance of the synchronization network also on the test subjects. To this end, we sample synchronized and unsynchronized pairs in equal proportion with varying time gaps. We plot the accuracy in Figure 6.4. Note that we use all the frames in the test examples in this case. As expected, we see an increase in performance with a more extensive temporal gap between the two frames. We also show some qualitative samples of synchronization retrievals in Figure 6.5.

6.4 Discussion

In this chapter, we described a novel self-supervised learning method to tackle mono-cular 3D human pose estimation. Our approach delivers high performance without requiring large manually labeled datasets with 2D or 3D pose annotations. To avoid such detailed annotation, we exploit a novel self-supervised task that leads to a representation that supports 3D pose estimation. Our task is to detect when two views have been captured simultaneously (and thus, the scene is related by a rigid transformation) and when they are horizontally flipped with respect to one another. We show on the well-known Human3.6M dataset that these two objectives build features that generalize across subjects and are highly sensitive to slight pose variations.

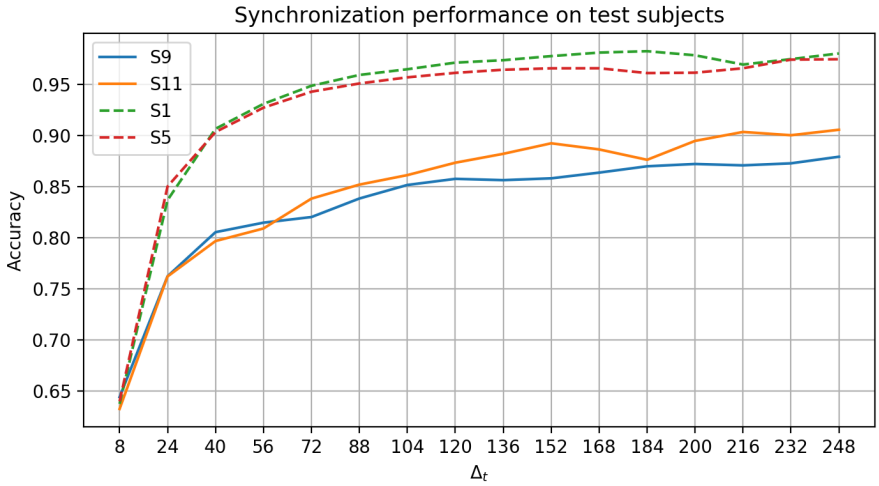


FIGURE 6.4: **Synchronization performance on unseen test subjects.** We evaluate the pre-trained synchronization network on test subjects S9 and S11. Unsynchronized pairs are created with a time difference sampled from the interval $[\Delta_t - 7, \Delta_t + 8]$. We also show the accuracy on two training subjects S1 and S5.

FIGURE 6.5: **Synchronization retrievals on test subjects.** Query images are shown on the top row, and the corresponding predicted synchronized frames are shown in the row below.

The task design proposed here, again, has some similarities to the tasks proposed in the previous chapters. We again pose the recognition of how samples were transformed. However, here we are considering a relative transformation between two views of the same scene, and the transformation concerns non-rigid deformations resulting from moving subjects.

FIGURE 6.6: **Generalization across subjects.** We investigate if a network trained on our self-supervised synchronization task generalizes across different subjects. The left-most column shows the query image from test subject S9. The remaining columns show images of training subjects S1, S5, S6, S7, and S8, respectively, for which our model predicts the highest probability of being synchronized.

Chapter 7

Stabilizing Generative Adversarial Training With Random Noise

"When some systems are stuck in a dangerous impasse, randomness and only randomness can unlock them and set them free."

— Nassim Nicholas Taleb

Since the seminal work of [24], generative adversarial networks (GAN) have been widely used and analyzed due to the quality of the samples they produce when applied to the space of natural images.

In Chapters 3 and 4, we have seen how one can use generative adversarial learning to define image transformations for self-supervised representation learning. Unfortunately, GANs still prove difficult to train. In fact, a vanilla implementation does not converge to a high-quality sample generator, and heuristics used to improve the generator often exhibit unstable behavior. These training issues have led to substantial work to understand GANs better (see, for instance, [34], [42], [209]). In particular, [34] points out how the unstable training of GANs is due to the (limited and low-dimensional) support of the data and model distributions. In the original GAN formulation, the generator trains against a discriminator in a minimax optimization problem. The discriminator learns to distinguish real from fake samples, while the generator learns to generate fake samples that fool the discriminator. When the support of the data and model distributions is disjoint, the generator stops improving as soon as the discriminator achieves perfect classification. These non-overlapping supports prevent the propagation of useful information to the generator through gradient descent (see Figure 7.1a).

The recent work by [34] proposes extending the distributions' support by adding noise to both generated and real images before they are fed as input to the discriminator. This procedure results in a smoothing of both data and model probability distributions which indeed increases their support extent (see Figure 7.1b). For simplicity, let us assume that the data's probability density function is well defined and let us denote it with p_d. Then, samples $\tilde{x} = x + \epsilon$, obtained by adding noise $\epsilon \sim p_\epsilon$ to the data samples $x \sim p_d$, are also instances of the probability density function $p_{d,\epsilon} = p_\epsilon * p_d$, where $*$ denotes the convolution operator. The support of $p_{d,\epsilon}$ is the Minkowski sum of the supports of p_ϵ and p_d and thus larger than the support of p_d. Similarly, adding noise to the samples from the generator probability density p_g leads to the smoothed probability density $p_{g,\epsilon} = p_\epsilon * p_g$. Adding noise is a relatively well-known technique that has been used in maximum likelihood methods but is considered undesirable as it yields approximate generative models that

Material from: S. Jenni, and P. Favaro. "On Stabilizing Generative Adversarial Training with Noise." In *IEEE Conference on Computer Vision and Pattern Recognition 2019*. © IEEE 2019

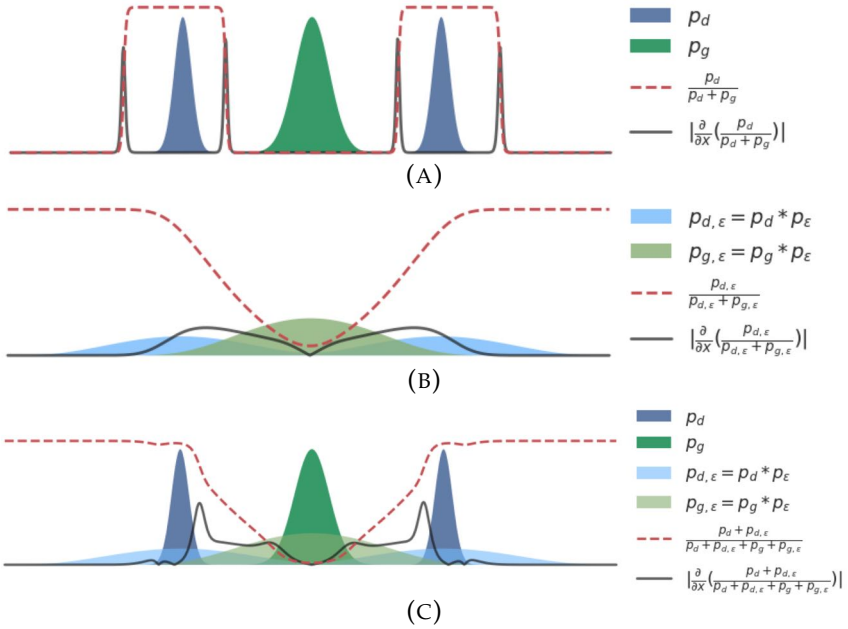


FIGURE 7.1: **Addressing the limited support of the data distribution in GANs.** (a) When the probability density functions of the real p_d and generated data p_g do not overlap, the discriminator can easily distinguish samples. The gradient of the discriminator with respect to its input is zero in these regions, which prevents any further improvement of the generator. (b) Adding samples from an arbitrary p_ϵ to those of the real and the generated data results in the filtered versions $p_d * p_\epsilon$ and $p_g * p_\epsilon$. Because the supports of the filtered distributions overlap, the gradient of the discriminator is not zero, and the generator can improve. However, the high-frequency content of the original distributions is missing. (c) By varying p_ϵ, the generator can learn to match the data distribution accurately thanks to the extended supports.

produce low-quality blurry samples. Indeed, most formulations with additive noise boil down to finding the model distribution p_g that best solves $p_{d,\epsilon} = p_{g,\epsilon}$. However, this usually results in a low quality estimate p_g because $p_d * p_\epsilon$ has lost the high frequency content of p_d. An immediate solution is to use a form of noise annealing. The noise variance is initially high and is then reduced gradually during the iterations so that the original distributions, rather than the smooth ones, are eventually matched. This results in improved training, but as the noise variance approaches zero, the optimization problem converges to the original formulation. As a result, the algorithm may be subject to the usual unstable behavior.

In this chapter, we design a novel adversarial training procedure that is stable and yields accurate results. We show that under some general assumptions, it is possible to modify both the data and generator probability densities with additional noise without affecting the original noise-free formulation's optimality conditions. As an alternative to the original formulation, with $z \sim \mathcal{N}(0, I_d)$ and $x \sim p_d$,a

$$\min_G \max_D \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z)))], \tag{7.1}$$

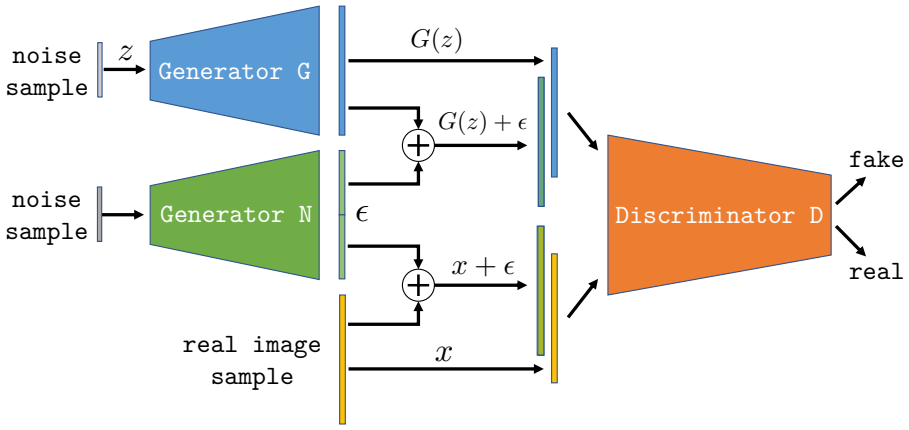


FIGURE 7.2: **Simplified scheme of the proposed GAN training.** We also show a noise generator N that is explained in detail in Section 7.2.1. The discriminator D needs to distinguish both noise-free and noisy real samples from fake ones.

where D denotes the discriminator, we propose to train a generative model G by solving the following optimization

$$\min_G \max_D \sum_{p_\epsilon \in \mathcal{S}} \mathbb{E}_{\epsilon \sim p_\epsilon} \left[\mathbb{E}_{x \sim p_d} [\log D(x+\epsilon)] \right] + \mathbb{E}_{\epsilon \sim p_\epsilon} \left[\mathbb{E}_{z \sim \mathcal{N}(0, I_d)} [\log(1 - D(G(z)+\epsilon))] \right], \tag{7.2}$$

where we introduced a set $\mathcal{S}$ of probability density functions. If we solve the innermost optimization problem in Problem (7.2), then we obtain the optimal discriminator

$$D(x) = \frac{\sum_{p_\epsilon \in \mathcal{S}} p_{d,\epsilon}(x)}{\sum_{p_\epsilon \in \mathcal{S}} p_{d,\epsilon}(x) + p_{g,\epsilon}(x)}, \tag{7.3}$$

where we have defined p_g as the probability density of $G(z)$, where $z \sim \mathcal{N}(0, I_d)$. If we substitute this in the problem above and simplify, we have

$$\min_G \mathrm{JSD} \left(\tfrac{1}{|\mathcal{S}|} \textstyle\sum_{p_\epsilon \in \mathcal{S}} p_{d,\epsilon}, \tfrac{1}{|\mathcal{S}|} \textstyle\sum_{p_\epsilon \in \mathcal{S}} p_{g,\epsilon} \right), \tag{7.4}$$

where JSD is the Jensen-Shannon divergence. We show that, under suitable assumptions, the optimal solution of Problem (7.4) is unique and $p_g = p_d$. Moreover, since $1/|\mathcal{S}| \sum_{p_\epsilon \in \mathcal{S}} p_{d,\epsilon}$ enjoys a larger support than p_d, the optimization via iterative methods based on gradient descent is more likely to achieve the global minimum, regardless of the support of p_d. Thus, our formulation enjoys the following properties: 1) It defines a fitting of probability densities that is not affected by their support; 2) It guarantees the exact matching of the data probability density function; 3) It can be easily applied to other GAN formulations. A simplified scheme of the proposed approach is shown in Figure 7.2.

In the next sections, we introduce our analysis in detail and then devise a computationally feasible approximation of the Problem (7.2). Our method is evaluated quantitatively on CIFAR-10 [210], STL-10 [117], and CelebA [142], and qualitatively on ImageNet [211] and LSUN bedrooms [212].

7.1 Background

We briefly review similar GAN stabilization approaches that also rely on transformations of the data here and refer to Chapter 2 for a general overview of the relevant GAN literature. Arjovsky *et al.* [34] first identified the disjoint supports of the real and fake data distributions as the culprit behind unstable GAN training. As a solution, they proposed the use of additive Gaussian noise. [209] made similar observations and proposed the use of "instance noise" which is a similar use of additive noise that is gradually reduced during training. Another approach that relies on transforming the discriminator inputs is given by [213]. Rather than using additive noise, they propose to learn a "lens" network that transforms real samples and is trained against the discriminator. As in [34] and [209], the strength of the transformation is reduced during training. Our approach is also based on additive noise. However, in contrast to the Gaussian noise used in [34] and [209], we additionally learn the optimal noise distribution. Furthermore, different from all the above approaches, we do not require decreasing the transformation magnitude. This leads to more stability throughout the complete training process.

7.2 Matching Filtered Distributions

We are interested in finding a formulation that yields as optimal generator G a sampler of the data probability density function (pdf) p_d, which we assume is well defined. The main difficulty in dealing with p_d is that it may be zero on some data space neighborhoods. An iterative optimization of Problem (7.1) based on gradient descent may yield a degenerate solution, *i.e.*, such that the model pdf p_g only partially overlaps with p_d (a scenario called *mode collapse*). It has been noticed that adding samples of an arbitrary distribution to both real and fake data samples during training helps reduce this issue. In fact, adding samples $\epsilon \sim p_\epsilon$ corresponds to blurring the original pdfs p_d and p_g, an operation known to increase their support and thus their likelihood to overlap. This increased overlap means that iterative methods can exploit useful gradient directions at more locations and are more likely to converge to the global solution. By building on this observation, we propose to solve Problem (7.2) instead and look for a way to increase the support of the data pdf p_d without losing the optimality conditions of the original formulation of Problem (7.1).

Our result below proves that this is the case for some choices of additive noise. We consider images of $m \times n$ pixels and with values in a compact domain $\Omega \subset \mathbb{R}^{m \times n}$ since image intensities are bounded from above and below. Then, also the support of the pdf p_d is bounded and contained in Ω. This implies that p_d is also $L^2(\Omega)$.

Theorem 1. *Let us choose $\mathcal{S}$ such that Problem (7.4) can be written as*

$$\min_{p_g} JSD\left(\frac{1}{2}(p_d + p_d * p_\epsilon), \frac{1}{2}(p_g + p_g * p_\epsilon)\right), \tag{7.5}$$

where p_ϵ is a zero-mean Gaussian function with a positive definite covariance Σ. Let us also assume that the domain of p_g is restricted to Ω (and thus $p_g \in L^2(\Omega)$). Then, the global optimum of Problem (7.5) is $p_g(x) = p_d(x), \forall x \in \Omega$.

Algorithm 1: Distribution Filtering GAN (DFGAN)

Input: Training set $\mathcal{D} \sim p_d$, number of discriminator updates n_{disc}, number of training iterations N, batch-size m, learning rate α, noise penalty λ
Output: Generator parameters θ
Initialize generator parameters θ, discriminator parameters ϕ and noise-generator parameters ω ;
for $1 \dots N$ **do**
  for $1 \dots n_{disc}$ **do**
    Sample $\{x_1, \dots, x_m\} \sim p_d$, $\{\tilde{x}_1, \dots, \tilde{x}_m\} \sim p_g$ and $\{\epsilon_1, \dots, \epsilon_m\} \sim p_\epsilon$;
    $L_D^r = \sum_{i=1}^m \ln(D(x_i)) + \ln(D(x_i + \epsilon_i))$;
    $L_D^f = \sum_{i=1}^m \ln(1 - D(\tilde{x}_i)) + \ln(1 - D(\tilde{x}_i + \epsilon_i))$;
    $L_\epsilon = \sum_{i=1}^m |\epsilon_i|^2$;
    $\phi \leftarrow \phi + \nabla_\phi L_D^r(\phi, \omega) + \nabla_\phi L_D^f(\phi, \omega)$;
    $\omega \leftarrow \omega - \nabla_\omega \big(L_D^r(\phi, \omega) + L_D^f(\phi, \omega) + \lambda L_\epsilon(\omega)\big)$;
  end
  Sample $\{\tilde{x}_1, \dots, \tilde{x}_m\} \sim p_g$ and $\{\epsilon_1, \dots, \epsilon_m\} \sim p_\epsilon$;
  $L_G^f = \sum_{i=1}^m \ln(D(\tilde{x}_i)) + \ln(D(\tilde{x}_i + \epsilon_i))$;
  $\theta \leftarrow \theta + \nabla_\theta L_G^f(\theta)$;
end

Proof. The global minimum of the Jensens-Shannon divergence is achieved if and only if

$$p_d + p_d * p_\epsilon = p_g + p_g * p_\epsilon. \tag{7.6}$$

Let $p_g = p_d + \Delta$. Then, we have $\int |\Delta(x)|^2 dx < \infty$. By substituting p_g in Equation (7.6) we obtain $\Delta * p_\epsilon = -\Delta$. Since Δ and p_ϵ are in $L^2(\Omega)$, we can take the Fourier transform of both sides, compute their absolute value and obtain

$$\left|\hat{\Delta}(\omega)\right| |\hat{p}_\epsilon(\omega)| = \left|\hat{\Delta}(\omega)\right|, \quad \forall \omega \in \hat{\Omega}. \tag{7.7}$$

Because p_g and p_d integrate to 1, $\int \Delta(x) dx = 0$ and $\hat{\Delta}(0) = 0$. Suppose $\exists \omega \neq 0$ such that $\hat{\Delta}(\omega) \neq 0$. Since p_ϵ is Gaussian, $|\hat{p}_\epsilon(\omega)| = \left|e^{-\frac{1}{2}\omega^\top \Sigma^{-1} \omega}\right| < 1$, which contradicts the optimality condition (7.7). Thus, $\Delta(x) = 0$, $\forall x \in \Omega$ and we can conclude that $p_g(x) = p_d(x)$, $\forall x \in \Omega$. □

7.2.1 Formulation

Based on the above theorem we consider two cases:

1. **Gaussian noise** with fixed/learned standard deviation σ: $p_\epsilon(\epsilon) = \mathcal{N}(\epsilon; 0, \sigma I_d)$;
2. **Learned noise** from a noise generator network N with parameters σ: $p_\epsilon(\epsilon)$ such that $\epsilon = N(w, \sigma)$, with $w \sim \mathcal{N}(0, I_d)$.

In both configurations we can learn the parameter(s) σ. We do so by minimizing the cost function after the maximization with respect to the discriminator. The minimization encourages large noise since this would make $p_{d,\epsilon}(\omega)$ more similar to $p_{g,\epsilon}(\omega)$ regardless of p_d and p_g. This would not be very useful to gradient descent. Therefore, to limit the noise magnitude we introduce as a regularization

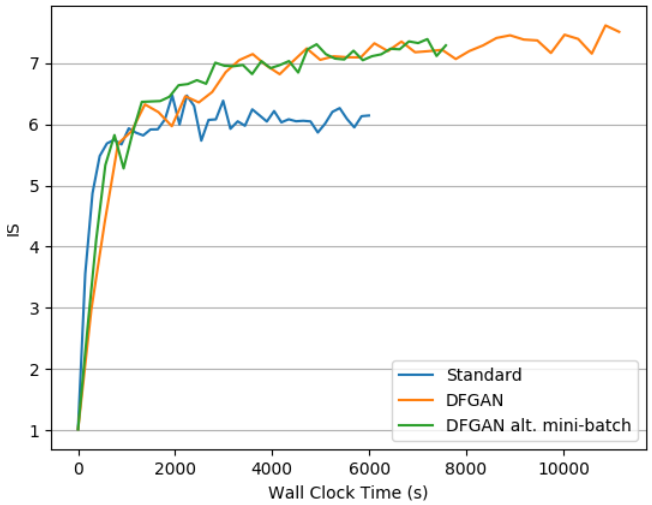


FIGURE 7.3: **Wall clock time vs IS for GANs with and without distribution filtering.** The models use the architecture specified in Table 7.2 and were trained on CIFAR-10. The computational overhead introduced by our method does not negatively affect the speed of convergence.

term the noise variance $\Gamma(\sigma) = \sigma^2$ or the Euclidean norm of the noise output image $\Gamma(\sigma) = \mathbb{E}_{w\sim\mathcal{N}(0,I_d)}|N(w,\sigma)|^2$, and multiply it by a positive scalar λ, which we tune.

The proposed formulations can then be written in a unified way as:

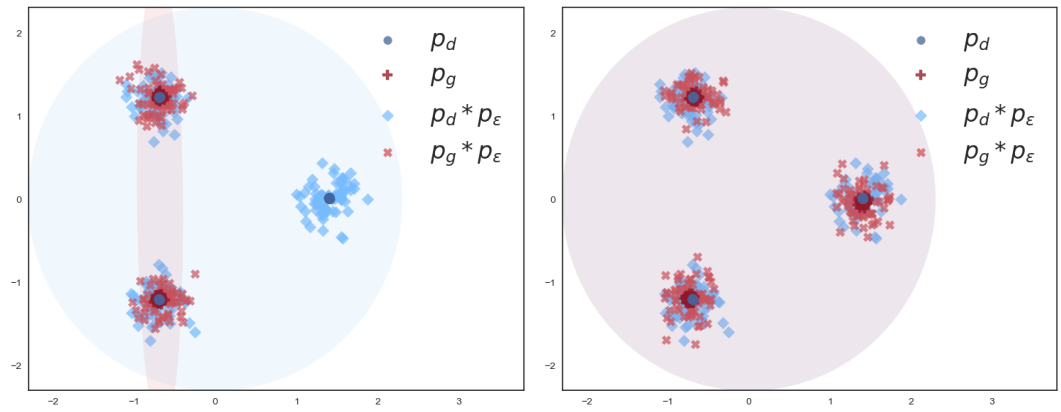
$$\begin{aligned}\min_G \min_\sigma \max_D \lambda\Gamma + \mathbb{E}_x\Big[\log D(x) + \mathbb{E}_\epsilon \log D(x+\epsilon)\Big] + \\ \mathbb{E}_z\Big[\log[1 - D(G(z))] + \mathbb{E}_\epsilon \log[1 - D(G(z)+\epsilon)]\Big].\end{aligned} \tag{7.8}$$

7.2.2 Implementation

Implementing our algorithm only requires a few minor modifications of the standard GAN framework. We perform the update for the noise-generator and the discriminator in the same iteration. Mini-batches for the discriminator are formed by collecting all the fake and real samples in two separate batches, *i.e.*, $\{x_1, \ldots, x_m, x_1 + \epsilon_1, \ldots, x_m + \epsilon_m\}$ is the batch with real examples and $\{\tilde{x}_1, \ldots, \tilde{x}_m, \tilde{x}_1 + \epsilon_1, \ldots, \tilde{x}_m + \epsilon_m\}$ the fake examples batch. The complete procedure is outlined in Algorithm 1. The noise-generator architecture is typically the same as the generators', but with a reduced number of convolutional filters. Since the inputs to the discriminator are doubled when compared to the standard GAN framework, the DFGAN framework can be 1.5 to 2 times slower. Similar and more severe performance drops are present in existing variants (*e.g.*, WGAN-GP). Note that by constructing the batches as $\{x_1, \ldots, x_{m/2}, x_{m/2+1} + \epsilon_1, \ldots, x_m + \epsilon_m\}$ the training time is instead comparable to the standard framework, but it is much more stable and yields an accurate generator. For a comparison of the runtimes, see Figure 7.3.

7.2.3 Batch-Normalization and Mode Collapse

The current best practice is to apply batch normalization to the discriminator separately on the real and fake mini-batches [214]. Indeed, this showed much better



(A) (B)

FIGURE 7.4: **How a separate normalization of fake and real mini-batches discourages mode collapse.** In (A), no normalization is applied, and mode collapse is observed. Since the covered modes are indistinguishable, the generator receives no signal that encourages better mode coverage. In (B), separate normalization of the real and fake data is applied. The mismatch in the batch statistics (mean and standard deviation) can now be detected by the discriminator, forcing the generator to improve.

results than feeding mini-batches with a 50/50 mix of real and fake examples in our experiments. The reason for this is that batch normalization implicitly considers the distribution of samples in each mini-batch. To see this, consider the example in Figure 7.4. In the case of no separate normalization of fake and real batches, we can observe mode-collapse. The modes covered by the generator are indistinguishable for the discriminator, which observes each example independently. There is no signal to the generator that leads to better mode coverage in this case. Since the first two moments of the fake and real batch distribution are not matching, a separate normalization will help the discriminator distinguish between real and fake examples and encourage better mode coverage by the generator.

Using batch normalization in this way turns out to be crucial for our method as well. Indeed, when no batch normalization is used in the discriminator, the generator will often produce noisy examples. This is not easy to detect by the discriminator since it judges each sample independently. To mitigate this issue, we apply separate normalization of the noisy real and fake examples before feeding them to the discriminator. We use this technique for models without batch normalization (*e.g.*, SNGAN).

7.3 Experiments

We compare and evaluate our model using two common GAN metrics: the Inception Score `IS` [33] and the Fréchet Inception Distance `FID` [215]. Throughout this section, we use 10K generated and real samples to compute `IS` and `FID`. To get a measure of the training's stability, we report the mean and standard deviation of the last five checkpoints for both metrics (obtained in the last 10% of training).

TABLE 7.1: **Ablation experiments on CIFAR-10 and STL-10.** We perform ablation experiments on CIFAR-10 and STL-10 to demonstrate the effectiveness of our proposed algorithm. The experiments (a)-(c) show results where only filtered examples are fed to the discriminator. Experiment (c) corresponds to the previously proposed noise-annealing and improves the standard GAN training. Our approach of feeding both filtered and clean samples to the discriminator shows a clear improvement over the baseline.

Experiment	CIFAR-10		STL-10	
	FID	IS	FID	IS
Standard GAN	46.1 ± 0.7	$6.12 \pm .09$	78.4 ± 6.7	$8.22 \pm .37$
(a) Noise only: $\epsilon \sim \mathcal{N}(0, I)$	94.9 ± 4.9	$4.68 \pm .12$	107.9 ± 2.3	$6.48 \pm .19$
(b) Noise only: ϵ learned	69.0 ± 3.4	$5.05 \pm .14$	107.2 ± 3.4	$6.39 \pm .22$
(c) Noise only: $\epsilon \sim \mathcal{N}(0, \sigma I), \sigma \to 0$	44.5 ± 3.2	$6.85 \pm .20$	75.9 ± 1.9	$8.49 \pm .19$
(d) Clean + noise: $\epsilon \sim \mathcal{N}(0, I)$	29.7 ± 0.6	$7.16 \pm .05$	66.5 ± 2.3	$8.64 \pm .17$
(e) Clean + noise: $\epsilon \sim \mathcal{N}(0, \sigma I)$ with learnt σ	28.8 ± 0.7	$7.23 \pm .14$	71.3 ± 1.7	$8.30 \pm .12$
(f) DFGAN ($\lambda = 0.1$)	27.7 ± 0.8	$7.31 \pm .06$	$\mathbf{63.9} \pm 1.7$	$\mathbf{8.81} \pm .07$
(g) DFGAN ($\lambda = 1$)	$\mathbf{26.5} \pm 0.6$	$\mathbf{7.49} \pm .04$	64.0 ± 1.4	$8.52 \pm .16$
(h) DFGAN ($\lambda = 10$)	29.8 ± 0.4	$6.55 \pm .08$	66.9 ± 3.2	$8.38 \pm .20$
(i) DFGAN alt. mini-batch ($\lambda = 1$)	28.7 ± 0.6	$7.3 \pm .05$	67.8 ± 3.2	$8.30 \pm .11$

7.3.1 Ablations

To verify our model we perform ablation experiments on two common image datasets: CIFAR-10 [210] and STL-10 [117]. For CIFAR-10 we train on the 50K 32×32 RGB training images and for STL-10 we resize the 100K 96×96 training images to 64×64. The network architectures resemble the DCGAN architectures of [25] and are detailed in Table 7.2. All the models are trained for 100K generator iterations using a mini-batch size of 64. We use the ADAM optimizer [120] with a learning rate of 10^{-4} and $\beta_1 = 0.5$. Results on the following ablations are reported in Table 7.1:

(a)-(c) Only noisy samples: In this set of experiments, we only feed noisy examples to the discriminator. In experiment **(a)** we add Gaussian noise and in **(b)** we add learned noise. In both cases, the noise level is not annealed. While this leads to stable training, the resulting samples are of poor quality, which is reflected by high FID and low IS. The generator will also tend to produce noisy samples since there is no incentive to remove the noise. Annealing the added noise during training as proposed by [34] and [209] leads to an improvement over the standard GAN. This is demonstrated in experiment **(c)**. The added Gaussian noise is linearly annealed during the 100K iterations in this case;

(d)-(i) Both noisy and clean samples: The second set of experiments consists of variants of our proposed model. Experiments **(d)** and **(e)** use a simple Gaussian noise model; in **(e)** the standard deviation of the noise σ is learned. We observe a drastic improvement in the quality of the generated examples even with this simple modification. The other experiments show the results of our full model with a separate noise-generator network. We vary the weight λ of the L^2 norm of the noise in experiments **(f)-(h)**. Ablation **(i)** uses the alternative mini-batch construction with faster runtime as described in Section 7.2.2;

TABLE 7.2: **Network architectures used for experiments on CIFAR-10 and STL-10.** Images are assumed to be of size 32×32 for CIFAR-10 and 64×64 for STL-10. We set $M = 512$ for CIFAR-10 and $M = 1024$ for STL-10. Layers in parentheses are only included for STL-10. The noise-generator network follows the generator architecture with the number of channels reduced by a factor of 8. BN indicates the use of batch-normalization [118].

Generator CIFAR-10/(STL-10)	**Discriminator CIFAR-10/(STL-10)**
$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$	conv 3×3 str.=1 lReLU 64
fully-conn. BN ReLU $4 \times 4 \times M$	conv 4×4 str.=2 BN lReLU 64
(deconv 4×4 str.=2 BN ReLU 512)	conv 4×4 str.=2 BN lReLU 128
deconv 4×4 str.=2 BN ReLU 256	conv 4×4 str.=2 BN lReLU 256
deconv 4×4 str.=2 BN ReLU 128	conv 4×4 str.=2 BN lReLU 512
deconv 4×4 str.=2 BN ReLU 64	(conv 4×4 str.=2 BN lReLU 1024)
deconv 3×3 str.=1 `tanh` 3	fully-connected `sigmoid` 1

7.3.2 Application to Different GAN Models

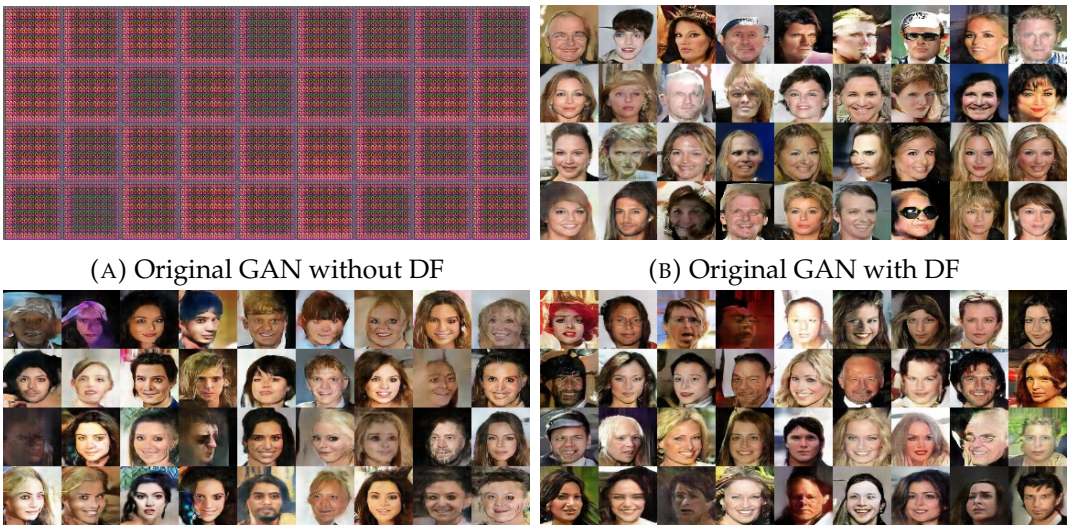
We investigate the possibility of applying our proposed training method to several standard GAN models. The network architectures are the same as proposed in the original works with only the necessary adjustments to the given image-resolutions of the datasets (*i.e.*, truncation of the network architectures). The only exception is SVM-GAN, where we use the architecture in Table 7.2. Note that for the GAN with minimax loss (MMGAN) and WGAN-GP, we use the architecture of DCGAN. Hyper-parameters are kept at their default values for each model. The models are evaluated on two common GAN benchmarks: CIFAR-10 [210] and CelebA [142]. The image resolution is 32×32 for CIFAR-10 and 64×64 for CelebA. All models are trained for 100K generator iterations. For the alternative objective function of LSGAN and SVM-GAN we set the noise generator's loss to be the negative of the discriminator loss, as is the case in our standard model. The results are shown in Table 7.3. We can observe that applying our training method improves performance in most cases and even enables the training with the original saturation-prone minimax GAN objective, which is very unstable otherwise. Note also that applying our method to SNGAN [43] (the current state-of-the-art) leads to an improvement on both datasets. We also evaluated SNGAN with and without our method on 64×64 images of STL-10 (same as in Table 7.1). Our method boosts the performance from an FID of 66.3 ± 1.1 to 58.3 ± 1.4. We show random CelebA reconstructions from models trained with and without our approach in Figure 7.5.

7.3.3 Robustness to Hyperparameters

We test the robustness of DFGANs to various hyperparameters by training on CIFAR-10 with the settings listed in Table 7.4. The network is the same as specified in Table 7.2. The noise penalty term is set to $\lambda = 0.1$. We compare to a model without our training method (Standard), a model with the gradient penalty regularization proposed by [42] (GAN+GP), and a model with spectral normalization (SNGAN). To the best of our knowledge, these methods are the current state-of-the-art in terms of GAN stabilization. Figure 7.6 shows that our method is stable and accurate across all settings.

TABLE 7.3: **Application to established GAN models on CIFAR-10 and CelebA.** We apply our proposed GAN training to various previous GAN models trained on CIFAR-10 and CelebA. The same network architectures and hyperparameters as in the original works are used (for SVM-GAN, we used the network in Table 7.2). We observe that our method increases performance in most cases, even with the suggested hyperparameter settings. Note that our approach also allows successful training with the original minimax MMGAN loss instead of the commonly used heuristic (*e.g.*, in DCGAN).

Model	CIFAR-10		CelebA
	FID	IS	FID
MMGAN [24]	> 450	~ 1	> 350
DCGAN [25]	33.4 ± 0.5	$6.73 \pm .07$	25.4 ± 2.6
WGAN-GP [36]	37.7 ± 0.4	$6.55 \pm .08$	15.5 ± 0.2
LSGAN [37]	38.7 ± 1.8	$6.73 \pm .12$	21.4 ± 1.1
SVM-GAN [140]	43.9 ± 1.0	$6.25 \pm .09$	26.5 ± 1.9
SNGAN ([43]	29.1 ± 0.4	$7.26 \pm .06$	13.2 ± 0.3
MMGAN +DF ($\lambda = 0.1$)	33.1 ± 0.7	$6.91 \pm .05$	16.6 ± 1.9
DCGAN + DF ($\lambda = 10$)	31.2 ± 0.3	$6.95 \pm .11$	14.7 ± 1.0
LSGAN + DF ($\lambda = 10$)	36.7 ± 1.2	$6.63 \pm .17$	19.9 ± 0.4
SVM-GAN + DF ($\lambda = 1$)	28.7 ± 1.1	$7.31 \pm .11$	12.7 ± 0.7
SNGAN + DF ($\lambda = 1$)	$\mathbf{25.9} \pm 0.3$	$\mathbf{7.47} \pm .08$	$\mathbf{10.5} \pm 0.4$



(A) Original GAN without DF (B) Original GAN with DF

(C) DCGAN without DF (D) DCGAN with DF

FIGURE 7.5: **Comparison of reconstructions with and without distribution filtering.** Left column: Random reconstructions from models trained on CelebA without distribution filtering (DF). Right column: Random reconstructions with our proposed method.

Robustness to Network Architectures. To test the robustness of DFGANs against non-optimal network architectures, we modified the networks in Table 7.2 by doubling the number of layers in both generator and discriminator. This leads to significantly worse performance in terms of FID in all cases: 46 to 135 (Standard), 33 to 111 (SNGAN), 28 to 36 (GAN+GP), and 27 to 60 (DFGAN). However, SNGAN+DF

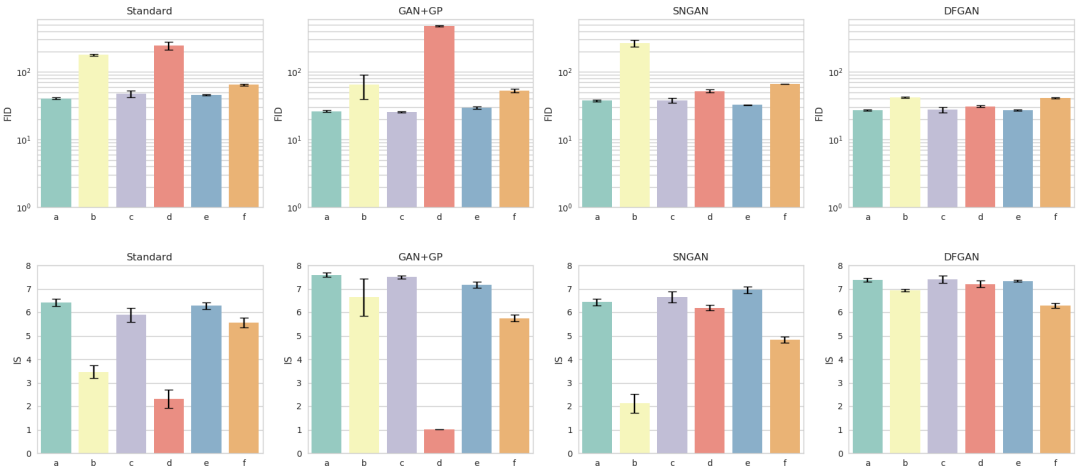


FIGURE 7.6: **Results of robustness experiments on CIFAR-10.** We compare the standard GAN (*1st column*), a GAN with gradient penalty (*2nd column*), a GAN with spectral normalization (*3rd column*) and a GAN with our proposed method (*4th column*). Results are reported in Fréchet Inception Distance `FID` (*top*) and Inception Score `IS` (*bottom*).

TABLE 7.4: **Hyperparameter settings for the robustness experiments.** We list the settings used to evaluate the robustness of our proposed GAN training method. We vary the learning rate α, the normalization in G, the optimizer, the activation functions, the number of discriminator iterations n_{disc} and the number of training examples n_{train}.

Exp.	**LR** α	**BN in** G	**Opt.**	**ActFn**	n_{disc}	n_{train}
a)	$2 \cdot 10^{-4}$	FALSE	ADAM	(l)ReLU	1	50K
b)	$2 \cdot 10^{-4}$	TRUE	ADAM	tanh	1	50K
c)	$1 \cdot 10^{-3}$	TRUE	ADAM	(l)ReLU	1	50K
d)	$1 \cdot 10^{-2}$	TRUE	SGD	(l)ReLU	1	50K
e)	$2 \cdot 10^{-4}$	TRUE	ADAM	(l)ReLU	5	50K
f)	$2 \cdot 10^{-4}$	TRUE	ADAM	(l)ReLU	1	5K

leads to good results with an FID of 27.6.

7.3.4 Qualitative Results

We trained DFGANs on 128×128 images from two large-scale datasets: ImageNet [211] and LSUN bedrooms [212]. The network architecture is similar to the one in Table 7.2 with one additional layer in both networks. We trained the models for 100K iterations on LSUN and 300K iterations on ImageNet. Random samples of the models are shown in Figure 7.8. Figure 7.7 shows some examples of the noise produced by the noise generator at different stages during training. These examples resemble the image patterns that typically appear when the generator diverges.

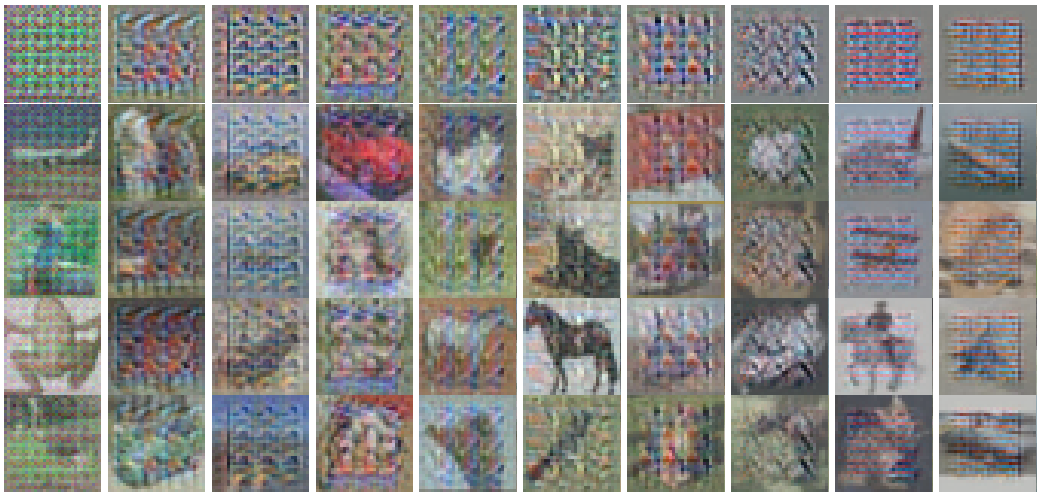


FIGURE 7.7: **Examples of the generated noise patterns.** We show samples of the generated noise (top row) and corresponding noisy training examples (rows 2 to 4). The columns correspond to different iterations. The noise varies over time to continually challenge the discriminator.

FIGURE 7.8: **Reconstructions from DFGANs trained on** 128×128 **images.** We show samples from DFGANs trained on 128×128 images from the LSUN bedrooms dataset (*top*) and ImageNet (*bottom*).

7.4 Discussion

We have introduced a novel method to stabilize generative adversarial training that results in more accurate generative models. Our approach is rather general and can be applied to existing GAN formulations with an average improvement in generated sample quality and variety and training stability. Since GAN training aims to match probability density distributions, we add random samples to both generated and real data to extend the densities' support and thus facilitate their matching through gradient descent. We demonstrated that the proposed training algorithm leads to more robustness to training parameters and achieves consistently improved FID and IS scores on several standard datasets and GAN models.

Chapter 8

Generalizing from Limited and Noisy Labels

"The growth of knowledge depends entirely upon disagreement."

— Karl Popper

A core objective in machine learning is to build models that generalize well, *i.e.*, that have the ability to perform well on new unseen data. A common strategy to achieve generalization is to employ regularization, which is a way to incorporate additional information about the space of suitable models. Regularization, in principle, prevents the estimated model from overfitting the training data. However, recent work [216] showed that current regularization methods applied to neural networks do not work according to conventional wisdom. In fact, it has been shown that neural networks can learn to map data samples to arbitrary labels despite using regularization techniques such as weight decay, dropout, and data augmentation. While the lone model architecture of a neural network seems to have an implicit regularizing effect [217], experiments suggest that it can overfit on any dataset, given enough training time. This poses a limitation to any trained neural network's performance, especially when labels are partially noisy.

In this chapter, we introduce a novel learning framework that reduces overfitting by formulating training as a *bilevel optimization* problem [218], [219]. Although the mathematical formulation of bilevel optimization is often involved, our final algorithm is a relatively straightforward modification of the current training methods. Bilevel optimization differs from the conventional one in that one of the constraints is also an optimization problem. The primary objective function is called the *upper-level* optimization task, and the optimization problem in the set of constraints is called the *lower-level* optimization task. In our formulation, the lower-level problem is a model parameter optimization on samples from the *training set*, while the upper-level problem works as a performance evaluation on samples from a separate *validation set*. The optimal model is thus the one that is trained on one dataset but performs well on a different one, a property that closely follows the definition of generalization.

In the optimization procedure, we introduce a scalar weight for each sample mini-batch. The purpose of these variables is to find the linear combination of a subset of mini-batches from the training set that can best approximate the validation set error. They can also be seen as a way to 1) discard noisy samples and 2) adjust the parameter optimization path. Finally, these weights can also be interpreted as

Material from: S. Jenni, and P. Favaro. "Deep Bilevel Learning." In *European Conference on Computer Vision 2018*.

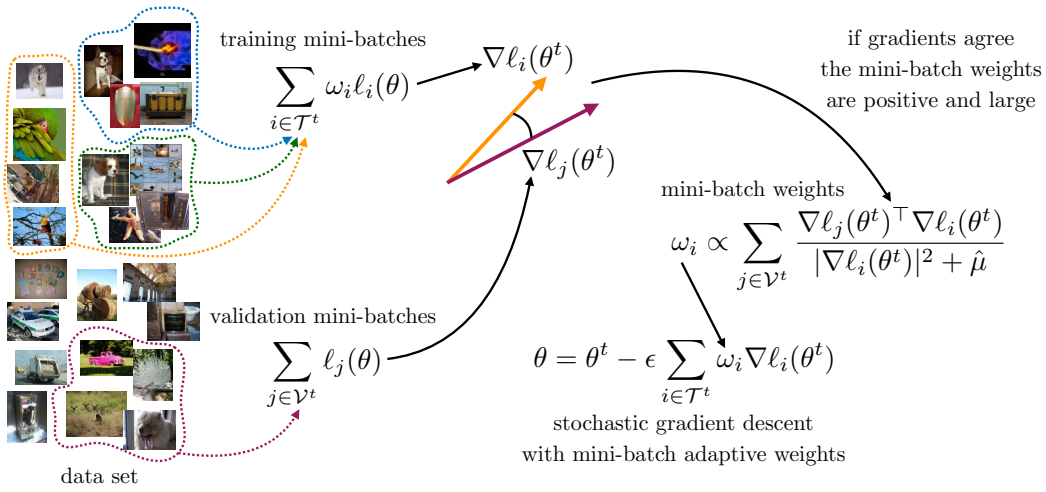


FIGURE 8.1: **The training procedure of our bilevel formulation.** We sample mini-batches from the data set at each iteration, which we split into validation and training batches. The validation batches define the weights of the loss gradient used in the stochastic gradient descent to update the model parameters. If the gradients of the training batches and those of the validation batches agree, then the weights are large and positive. Vice versa, if they disagree, the weights might be zero or negative.

hyper-parameters. Hence, bilevel optimization can be seen as an integrated way to continuously optimize for both the model parameters and the hyper-parameters, as done in cross-validation.

In its general form, bilevel optimization is known to present computational challenges. To address these challenges, we propose to approximate the loss objectives at every iteration with quadratic functions. These approximations result in closed-form solutions that resemble the well-known SGD update rules. Essentially, our bilevel optimization computes loss gradients on the training set and then prescribes adjustments to the learning rates of the SGD iteration so that the updated parameters perform well on the validation set. As we will show later, these adjustments depend on how well the gradients computed on the training set "agree" with the gradients computed on the validation set (see Figure 8.1). It is illustrative to consider how the agreement or disagreement between gradient directions can benefit the learning from noisy labels: we would expect gradients on correctly labeled examples to agree on average, while a disagreement could indicate the presence of label noise.

Our method can be easily integrated into current training procedures for neural networks, and our experiments show that it yields models with better generalization on several network architectures and datasets.

8.1 Background

We give an overview of prior work relating to three main aspects of the approach: 1) generalization properties of deep networks and how learning algorithms affect them; 2) memorization of corrupt labels as a particular case of overfitting; 3) bilevel optimization in the context of deep learning.

Understanding Generalization in Deep Learning. Although convolutional neural networks trained using stochastic gradient descent generalize well in practice, Zhang *et al.* [216] experimentally demonstrated that these models can fit random labelings of the training data. This is true even when using standard explicit regularization techniques. Several works gave possible explanations for the apparent paradox of good generalization despite the models' high capacity. Kawaguchi *et al.* [220] provided an explanation based on model-selection (*e.g.*, network architecture) via cross-validation. Their theoretical analysis also resulted in new generalization bounds and regularization strategies. Zhang *et al.* [221] attributed the generalization properties of CNNs to characteristics of the stochastic gradient descent optimizers. Their results showed that SGD favors flat minima, which in turn correspond to large (geometrical) margin classifiers. In contrast, we argue that current training schemes for neural networks can avoid overfitting altogether by exploiting cross-validation during the optimization.

Combating Memorization of Noisy Labels. The memorization of corrupted labels is a form of overfitting that is of practical importance since labels are often unreliable. Several authors have therefore addressed the problem of learning with noisy labels. Rolnick *et al.* [222] showed that neural networks can be robust to even high levels of noise provided good hyper-parameter choices. They specifically demonstrated that larger batch sizes are beneficial in the case of label noise. Patriani *et al.* [223] addressed label noise with a loss correction approach. Nataranjan *et al.* [224] performed a theoretical study of the binary classification problem under the presence of label noise and proposed approaches to modify the loss accordingly. Jindal and Chen [225] used dropout and augmented networks with a softmax layer that models the label noise and is trained jointly with the network. Sukhabar *et al.* [226] introduced an extra noise layer into the network that adapts the network output to match the noisy label distribution. Reed *et al.* [227] tackled the problem by augmenting the classification objective with a notion of consistency given similar percepts. Besides approaches that explicitly model the noise distribution, several regularization techniques have proven effective in this scenario. Jiang *et al.* [228] introduced a regularization technique to counter label noise. They train a network (MentorNet) to assign weights to each training example. Another recent regularization technique was introduced by Zhang *et al.* [229]. Their method is a form of data augmentation where two training examples are mixed (both images and labels) in a convex combination. Azadi *et al.* [230] proposed a regularization technique based on overlapping group norms. Their regularizer demonstrates good performance but relies on features trained on correctly labeled data. Our method differs from the above because we avoid memorization by encouraging only model parameter updates that reduce errors on shared sample patterns rather than example-specific details.

Bilevel Optimization. Various authors have considered bilevel optimization to solve for hyper-parameters based on the performance on a validation set [231], [232]. Domke [233] introduced a truncated bilevel optimization method where the lower-level is approximated by running an iterative algorithm for a given number of steps and subsequently computing the gradient on the validation loss via algorithmic differentiation. Our method uses the limiting case of using a single step in the lower-level problem. Ochs *et al.* [234] introduced a similar technique to the case of non-smooth lower-level problems by differentiating the iterations of a primal-dual algorithm. Maclaurin *et al.* [235] addressed the issue of expensive caching required for this kind of optimization by deriving an algorithm to exactly reverse SGD while

storing only a minimal amount of information. Kunish *et al.* [236] applied bilevel optimization to learn parameters of a variational image denoising model. We do not use bilevel optimization to solve for existing hyper-parameters, but rather introduce and solve for new hyper-parameters by assigning weights to stochastic gradient samples at each iteration.

Meta Learning. Our proposed algorithm has some similarity to the meta-learning literature [237], [238], [239]. Most notably, the MAML algorithm by Finn *et al.* [237] also incorporates gradient information of two datasets. However, it does so in different ways: their method uses second-order derivatives whereas we only use first-order derivatives. In general, our approach's purpose and data are quite different from the meta-learning setting: we have only one task, while in meta-learning, there are multiple tasks.

8.2 Learning to Generalize

We are given m sample pairs $(x^{(k)}, y^{(k)})_{k=1,\dots,m}$, where $x^{(k)} \in \mathcal{X}$ represents input data and $y^{(k)} \in \mathcal{Y}$ represents targets/labels. We denote with $\phi_\theta : \mathcal{X} \mapsto \mathcal{Y}$ a model that depends on parameters $\theta \in \mathbb{R}^d$ for some positive integer d. In all our experiments this model is a neural network and θ collects all its parameters. To measure the performance of the model, we introduce a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ per sample. Since we evaluate the loss $\mathcal{L}$ on b mini-batches $\mathcal{B}_i \subset \{1, \dots, m\}$, $i = 1, \dots, b$, where $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$ for $i \neq j$, we redefine the loss as

$$\ell_i(\theta) \triangleq \textstyle\sum_{k \in \mathcal{B}_i} \mathcal{L}\left(\phi_\theta\left(x^{(k)}\right), y^{(k)}\right). \tag{8.1}$$

At every iteration, we collect a subset of the mini-batches $\mathcal{U}^t \subset \{1, \dots, b\}$, which we partition into two separate sets: one for training $\mathcal{T}^t \subset \mathcal{U}^t$ and one for validation $\mathcal{V}^t \subset \mathcal{U}^t$, where $\mathcal{T}^t \cap \mathcal{V}^t = \emptyset$ and $\mathcal{T}^t \cup \mathcal{V}^t = \mathcal{U}^t$. Thus, mini-batches $\mathcal{B}_i$ in the training set have $i \in \mathcal{T}^t$ and those in the validation set have $i \in \mathcal{V}^t$. In all our experiments, the validation set $\mathcal{V}^t$ is always a singleton (one mini-batch).

8.2.1 Bilevel Learning

At the t-th iteration, SGD uses only one mini-batch to update the parameters via

$$\theta^{t+1} = \theta^t - \hat{\epsilon} \nabla \ell_i(\theta^t), \tag{8.2}$$

where $\hat{\epsilon} > 0$ is the SGD learning rate and $i \in \mathcal{U}^t$. Instead, we consider the subset $\mathcal{T}^t \subset \mathcal{U}^t$ of mini-batches and look for the linear combination of the losses that best approximates the validation error. We introduce an additional coefficient ω_i per mini-batch in $\mathcal{T}^t$, which we estimate during training. Our task is to find parameters θ of our model by using only mini-batches from the training set $\mathcal{T}^t \subset \mathcal{U}^t$, and to identify coefficients (hyper-parameters) ω_i so that the model performs well on the validation set $\mathcal{V}^t \subset \mathcal{U}^t$. We thus propose to optimize

$$\begin{aligned} \hat{\theta}, \hat{\boldsymbol{\omega}} = \quad & \arg\min_{\theta, \boldsymbol{\omega}} \quad \textstyle\sum_{j \in \mathcal{V}^t} \ell_j(\theta(\boldsymbol{\omega})) + \frac{\mu}{2} |\boldsymbol{\omega}|^2 \\ & \text{subj. to} \quad \theta(\boldsymbol{\omega}) = \arg\min_{\bar{\theta}} \textstyle\sum_{i \in \mathcal{T}^t} \omega_i \ell_i(\bar{\theta}) \\ & \qquad\quad |\boldsymbol{\omega}|_1 = 1, \end{aligned} \tag{8.3}$$

where $\boldsymbol{\omega}$ is the vector collecting all ω_i, $i \in \mathcal{T}^t$ and $\mu > 0$ is a parameter to regulate the distribution of the weights (large values would encourage a uniform distribution across mini-batches and small values would allow more sparsity). Notice that the lower-level problem's solution does not change if we multiply all the coefficients ω_i by the same strictly positive constant. Therefore, to fix the magnitude of $\boldsymbol{\omega}$ we introduced the L^1 normalization constraint $|\boldsymbol{\omega}|_1 = 1$.

A classical method to solve the above bilevel problem is to solve a linear system in the second-order derivatives of the lower-level problem, the so-called *implicit differentiation* [233]. This step leads to solving a very high-dimensional linear system. To avoid these computational challenges, in the next section, we introduce a proximal approximation. Notice that when we compare the bilevel formulation (8.3) with SGD in the experiments, we equalize computational complexity by using the same number of visits per sample.

8.2.2 A Proximal Formulation

To simplify the bilevel formulation (8.3) we propose to solve a sequence of approximated problems. The parameters estimated at the t-th approximated problem are denoted θ^{t+1}. Both the upper-level and the lower-level problems are approximated via a first-order Taylor expansion of the loss function based on the previous parameter estimate θ^t, *i.e.*, we let

$$\ell_i(\theta) \simeq \ell_i(\theta^t) + \nabla \ell_i(\theta^t)^\top (\theta - \theta^t). \tag{8.4}$$

Since the above Taylor expansion holds only in the proximity of the previous parameter estimates θ^t, we also introduce *proximal quadratic* terms $\left|\theta - \theta^t\right|^2$. By plugging the linear approximation (8.4) and the proximal terms in Problem (8.3) we obtain the following formulation

$$\begin{aligned} \theta^{t+1}, \hat{\boldsymbol{\omega}} = \quad & \underset{\theta, \boldsymbol{\omega}}{\arg\min} \quad \textstyle\sum_{j \in \mathcal{V}^t} \ell_j(\theta^t) + \nabla \ell_j(\theta^t)^\top (\theta(\boldsymbol{\omega}) - \theta^t) + \frac{\left|\theta(\boldsymbol{\omega}) - \theta^t\right|^2}{2\lambda} + \frac{\mu}{2} |\boldsymbol{\omega}|^2 \\ & \text{s.t.} \quad \theta(\boldsymbol{\omega}) = \arg\min_{\bar{\theta}} \textstyle\sum_{i \in \mathcal{T}^t} \omega_i \left[\ell_i(\theta^t) + \nabla \ell_i(\theta^t)^\top (\bar{\theta} - \theta^t) \right] + \frac{\left|\bar{\theta} - \theta^t\right|^2}{2\epsilon} \\ & \quad\;\; |\boldsymbol{\omega}|_1 = 1, \end{aligned} \tag{8.5}$$

where the coefficients $\lambda, \epsilon > 0$. The lower-level problem is now quadratic and can be solved in closed-form. This yields an update rule identical to the SGD step (8.2) when $\omega_i = 1$

$$\theta(\boldsymbol{\omega}) = \theta^t - \epsilon \textstyle\sum_{i \in \mathcal{T}^t} \omega_i \nabla \ell_i(\theta^t). \tag{8.6}$$

Now we can plug this solution in the upper-level problem and obtain

$$\begin{aligned} \hat{\boldsymbol{\omega}} = \quad & \underset{\theta, \boldsymbol{\omega}}{\arg\min} \quad \textstyle\sum_{j \in \mathcal{V}^t, i \in \mathcal{T}^t} -\omega_i \nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) + \frac{\left|\sum_{i \in \mathcal{T}^t} \omega_i \nabla \ell_i(\theta^t)\right|^2}{2\lambda/\epsilon} + \frac{\mu}{2\epsilon} |\boldsymbol{\omega}|^2 \\ & \text{s.t.} \quad |\boldsymbol{\omega}|_1 = 1. \end{aligned} \tag{8.7}$$

We simplify the notation by introducing $\hat{\lambda} = \lambda/\epsilon$ and $\hat{\mu} = \mu/\epsilon$. To find the optimal coefficients $\boldsymbol{\omega}$ we temporarily ignore the normalization constraint $|\boldsymbol{\omega}|_1 = 1$ and solve the unconstrained optimization. Afterward, we enforce the L^1 normalization to the solution. As a first step, we compute the derivative of the cost functional with respect to w_i and set it to zero, *i.e.*, $\forall i \in \mathcal{T}^t$

$$0 \quad = \textstyle\sum_{j \in \mathcal{V}^t} -\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) + \frac{1}{\hat{\lambda}} \sum_{k \in \mathcal{T}^t} \omega_k \nabla \ell_k(\theta^t)^\top \nabla \ell_i(\theta^t) + \hat{\mu} \omega_i. \tag{8.8}$$

We now approximate the second sum by ignoring all terms such that $k \neq i$, *i.e.*,

$$0 = \sum_{j \in \mathcal{V}^t} -\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) + \left(\tfrac{1}{\hat{\lambda}} |\nabla \ell_i(\theta^t)|^2 + \hat{\mu} \right) \omega_i \tag{8.9}$$

so that we can obtain the weight update rule

$$\forall i \in \mathcal{T}^t, \quad \omega_i \leftarrow \sum_{j \in \mathcal{V}^t} \frac{\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t)}{|\nabla \ell_i(\theta^t)|^2 / \hat{\lambda} + \hat{\mu}}, \qquad \hat{\boldsymbol{\omega}} = \boldsymbol{\omega} / |\boldsymbol{\omega}|_1. \tag{8.10}$$

Since eq. (8.8) describes a linear system, it could be solved exactly via several iterative methods, such as Gauss-Seidel or successive over-relaxation [240]. However, we found that using this level of accuracy does not substantially improve the model performance to justify the additional computational cost. We can then combine the update rule (8.10) with the update (8.6) of the parameters θ and obtain a new gradient descent step

$$\theta(\boldsymbol{\omega}) = \theta^t - \epsilon \sum_{i \in \mathcal{T}^t} \hat{\omega}_i \nabla \ell_i(\theta^t). \tag{8.11}$$

Notice that $\epsilon \hat{\omega}_i$ can be seen as a learning rate specific to each mini-batch. The update rule for the weights follows a very intuitive scheme: if the gradients of a mini-batch in the training set $\nabla \ell_i(\theta^t)$ agree with the gradients of a mini-batch in the validation set $\nabla \ell_j(\theta^t)$, then their inner product $\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) > 0$ and their corresponding weights are also positive and large. This means that we encourage updates of the parameters that also minimize the upper-level problem. When these two gradients disagree, that is, if they are orthogonal $\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) = 0$ or in the opposite directions $\nabla \ell_j(\theta^t)^\top \nabla \ell_i(\theta^t) < 0$, then the corresponding weights are also set to zero or a negative value, respectively (see Figure 8.1 for a general overview of the training procedure). Moreover, these inner products are scaled by the gradient magnitude of mini-batches from the training set, and division by zero is avoided when $\mu > 0$.

Remark 1. *Attention must be paid to the sample composition in each mini-batch since we aim to approximate the validation error with a linear combination of a few mini-batches. In fact, if samples in a mini-batch of the training set are relatively independent of samples in mini-batches of the validation set (for example, they belong to very different categories in a classification problem), then their inner product will tend to be very small on average. This would not allow any progress in the estimation of the parameters θ. We ensure that samples in each mini-batch from the training set have overlapping labels with samples in mini-batches from the validation set at each iteration.*

8.3 Implementation

To implement our method we modify SGD with momentum [241]. First, at each iteration t we sample k mini-batches $\mathcal{B}_i$ in such a way that the distributions of labels across the k mini-batches are identical (in the experiments, we consider $k \in \{2, 4, 8, 16, 32\}$). Next, we compute the gradients $\nabla \ell_i(\theta^t)$ of the loss function on each mini-batch $\mathcal{B}_i$. $\mathcal{V}^t$ contains only the index of one mini-batch and $\mathcal{T}^t$ all the remaining indices. We then use $\nabla \ell_j(\theta^t)$, $j \in \mathcal{V}^t$, as the *single* validation gradient and compute the weights ω_i of $\nabla \ell_i(\theta^t)$, $i \in \mathcal{T}^t$, using eq. (8.10). The re-weighted gradient $\sum_{i \in \mathcal{T}^t} \omega_i \nabla \ell_i(\theta^t)$ is then fed to the neural network optimizer.

8.4 Experiments

We perform extensive experiments on several common datasets used for training image classifiers. Section 8.4.1 shows ablations to verify several design choices. In Sections 8.4.2 and 8.4.3 we follow the experimental setup of Zhang *et al.* [216] to demonstrate that our method reduces sample memorization and improves performance on noisy labels at test time. In Section 8.4.4 we show improvements on small datasets. The datasets considered in this section are the following:

CIFAR-10 [210]: It contains 50K training and 10K test images of size 32×32 pixels, equally distributed among 10 classes.

CIFAR-100 [210]: It contains 50K training and 10K test images of size 32×32 pixels, equally distributed among 100 classes.

Pascal VOC 2007 [116]: It contains 5,011 training and 4,952 test images (the `trainval` set) of 20 object classes.

ImageNet [242]: It is a large dataset containing 1.28M training images of objects from 1K classes. We test on the validation set, which has 50K images.

We evaluate our method on several network architectures. On Pascal VOC and ImageNet, we use AlexNet [11]. Following Zhang *et al.* [216], we use CifarNet (an AlexNet-style network), and a small Inception architecture adapted to the smaller image sizes of CIFAR-10 and CIFAR-100. We refer the reader to [216] for a detailed description of those architectures. We also train variants of the ResNet architecture [119] to compare to other methods.

8.4.1 Ablations

We perform extensive ablation experiments on CIFAR-10 using the CifarNet and Inception network. The networks are trained on both clean labels and labels with 50% random noise. We report classification accuracy on the training labels (clean or noisy) and the accuracy on the *clean* test labels. The baseline in all the ablation experiments compares 8 mini-batches and uses $\mu = 0.01$ and $\lambda = 1$. Both networks have a single dropout layer, and the baseline configuration uses the same dropping in all the compared mini-batches. The networks are trained for 200 epochs on mini-batches of size 128. We do not use data augmentation for CifarNet, but we use standard augmentations for the Inception network (*i.e.*, random cropping and perturbation of brightness and contrast). Therefore, the case of the Inception network is closer to the common setup for training neural networks, and the absence of augmentation in the case of CifarNet makes overfitting more likely. We use SGD with a momentum of 0.9 and an initial learning rate of 0.01 in the case of CifarNet and 0.1 for Inception. The learning rate is reduced by a factor of 0.95 after every epoch. Although the validation and training sets split the selected mini-batches into two separate sets in our formulation, after one epoch, mini-batches used in the validation set could be used in the training set and vice versa. We test the case where we manually enforce that no examples (in mini-batches) used in the validation set are ever used for training and find no benefit. We explore different sizes of the separate validation and training sets. We define as *validation ratio* the fraction of samples from the dataset used for validation only. Figure 8.2 demonstrates the influence of the validation ratio (top row), the number of compared mini-batches (second row),

the size of the compared mini-batches (third row), and the hyper-parameter μ (bottom row). We can observe that the validation ratio has only a small influence on the performance. We see an overall negative trend in the test accuracy with the increasing size of the validation set. This is probably due to the corresponding reduction of the training set size. The number of mini-batches has a much more pronounced influence on the network's performance, especially in the case of CifarNet, where overfitting is very likely. Note that we keep the number of training steps constant in this experiment. Hence, the case with more mini-batches corresponds to smaller batch sizes. While the performance on noisy labels increases with the number of compared mini-batches, we observe a decrease in performance on clean data. We want to mention that the case of 2 mini-batches is rather interesting since it amounts to flipping (or not) the sign of the single training gradient based on the dot product with the single validation gradient. To test whether the performance with a growing number of batches is due to the batch sizes, we perform experiments where we vary the batch size while keeping the number of compared batches fixed at 8. Since this modification leads to more iterations, we adjust the learning rate schedule accordingly. Notice that all comparisons use the same overall number of times each sample is used. We can observe a behavior similar to the case of the varying number of mini-batches. This suggests that small mini-batch sizes lead to better generalization in the presence of label noise. Notice also the special case where the batch size is 1, which corresponds to per-example weights. Besides inferior performance, we found this choice to be computationally inefficient and interfering with batch-norm. Interestingly, the parameter μ does not seem to influence the performance of both networks significantly. Overall the performance on clean labels is quite robust to hyper-parameter choices except for the size of the mini-batches.

In Table 8.1, we also summarize the following set of ablation experiments:

a) No L^1-constraint on $\boldsymbol{\omega}$: We show that using the L^1 constraint $|\boldsymbol{\omega}|_1 = 1$ is beneficial for both clean and noisy labels. We set $\mu = 0.01$ and $\lambda = 1$ for this experiment in order for the magnitude of the weights ω_i to resemble the case with the L^1 constraint. While tuning of μ and λ might lead to an improvement, the use of the L^1 constraint allows plugging our optimization method without adjusting the learning rate schedule of existing models;

b) Weights per layer: In this experiment, we compute a separate $\omega_i^{(l)}$ for the gradients corresponding to each layer l. We then also apply L^1 normalization to the weights $\omega_i^{(l)}$ per layer. While the results on noisy data with CifarNet improve in this case, the performance of CifarNet on clean data and the Inception network on both datasets clearly degrades;

c) Mini-batch sampling: Here, we do not force the distribution of (noisy) labels in the compared mini-batches to be identical. The poor performance, in this case, highlights the importance of identically distributed labels in the mini-batches;

d) Dropout: We remove the restriction of equal dropping in all the compared mini-batches. Somewhat surprisingly, this improves performance in most cases. Note that unequal dropping lowers the influence of gradients in the deep fully-connected layers. Therefore this gives more weight to gradients of early convolutional layers in the dot-product. Also, dropout essentially amounts to having a different classifier at each iteration. Our method could encourage gradient updates that work well for different classifiers, possibly leading to a more general representation.

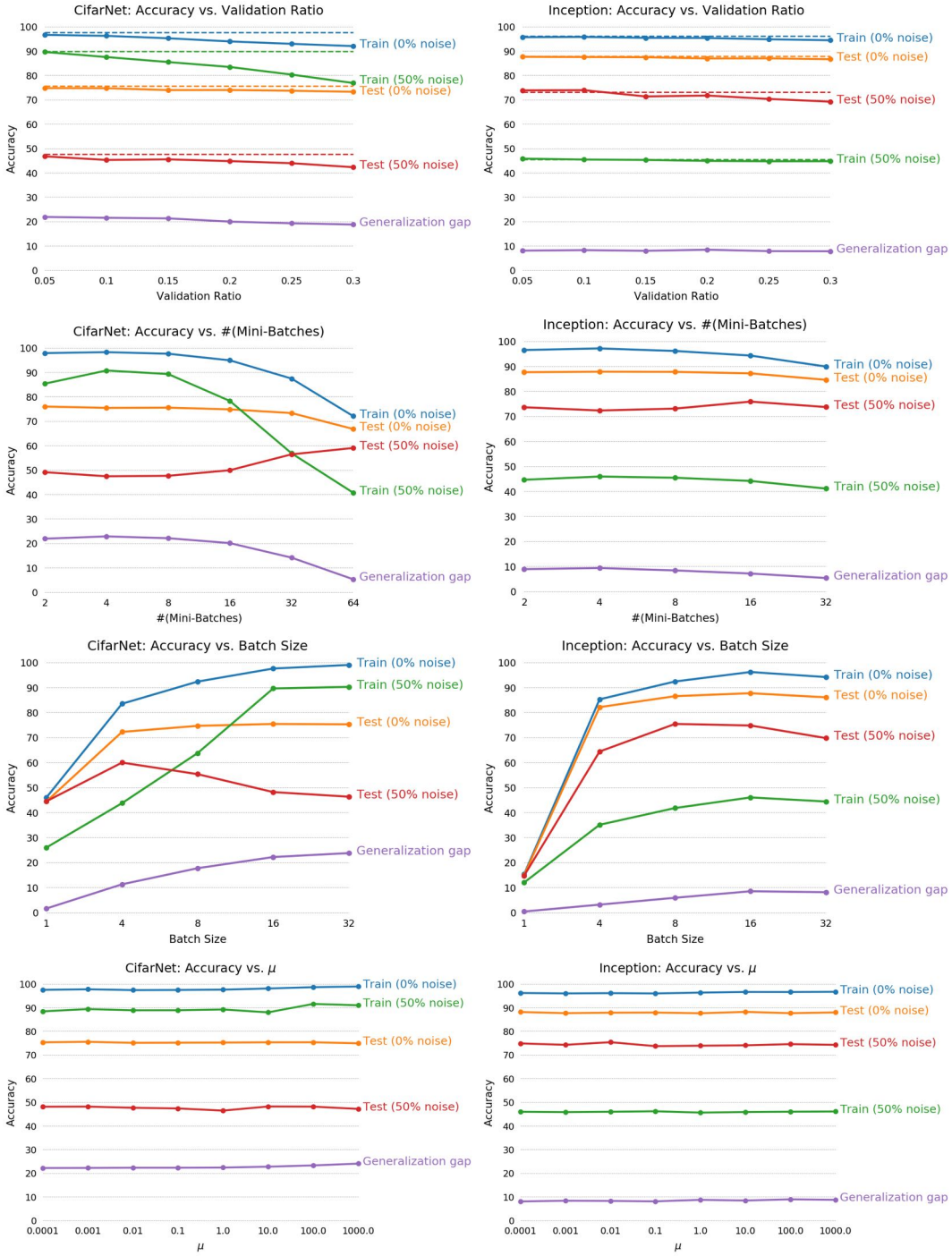


FIGURE 8.2: **Ablation experiments on CIFAR-10.** We report classification accuracy with CifarNet (a small AlexNet style network) (*left*) and a small Inception network (*right*). We vary the size of the validation set (*1st row*), the number of mini-batches being compared (*2nd row*), the mini-batch size (*3rd row*) and the hyper-parameter μ (*4th row*). The networks were trained on clean as well as 50% noisy labels. The amount of label noise during training is indicated in parentheses. We show the accuracy on the clean or noisy training data but always evaluate it on clean data. Note that the baseline of using the full training data as the validation set is indicated with dashed lines on the top row.

TABLE 8.1: **Results of ablation experiments on CIFAR-10.** Models were trained on clean labels and labels with 50% random noise. We report classification accuracy on the clean or noisy training labels and clean test labels. The generalization gap (the difference between training and test accuracy) on clean data is also included. We also show results of the baseline model and of a model trained with standard SGD.

Experiment	CifarNet					Inception				
	Clean			50% Random		Clean			50% Random	
	Train	Test	Gap	Train	Test	Train	Test	Gap	Train	Test
SGD	99.99	75.68	24.31	96.75	45.15	99.91	88.13	11.78	65.06	47.64
Baseline	97.60	75.52	22.08	89.28	47.62	96.13	87.78	8.35	45.43	73.08
a) L^1	96.44	74.32	22.12	95.50	45.79	79.46	77.07	2.39	33.86	62.16
b) ω **per Layer**	97.43	74.36	23.07	81.60	49.62	90.38	85.25	5.13	81.60	49.62
c) Sampling	72.69	68.19	4.50	16.13	23.93	79.78	78.25	1.53	17.71	27.20
d) Dropout	95.92	74.76	21.16	82.22	49.23	95.58	87.86	7.72	44.61	75.71

TABLE 8.2: **Experiments with random pixel permutations.** We report performance of the Inception network when trained on data with random pixel permutations (fixed per image). We observe much less overfitting using our method when compared to standard SGD.

Model	Train	Test	Gap
SGD	50.0	33.2	16.8
Bilevel	34.8	33.6	1.2

8.4.2 Fitting Random Pixel Permutations

Zhang *et al.* [216] demonstrated that CNNs could fit the training data even when images undergo random permutations of the pixels. Since object patterns are destroyed under such manipulations, learning should be minimal (restricted to simple statistics of pixel colors). We test our method with the Inception network trained for 200 epochs on images undergoing fixed random permutations of the pixels and report a comparison to standard SGD in Table 8.2. While both variants' test accuracy is similar, the network trained using our optimization shows a tiny generalization gap.

8.4.3 Memorization of Partially Corrupted Labels

The problem of label noise is of practical importance since the labeling process is often unreliable, and incorrect labels can be introduced in the process. Providing methods that are robust to noise in the training labels is therefore of interest. This section reports experiments on several datasets (CIFAR-10, CIFAR-100, ImageNet) with different forms and levels of label corruption and using different network architectures. We compare to other state-of-the-art regularization and label-noise methods on CIFAR-10 and CIFAR-100.

Random Label Corruptions on CIFAR-10 and CIFAR-100. We test our method under different levels of synthetic label noise. For a noise level $\pi \in [0, 1]$ and a dataset with c classes, we randomly choose a fraction of π examples per class and

TABLE 8.3: **Comparison to prior works.** We compare to state-of-the-art regularization techniques and methods for dealing with label noise on 40% corrupted labels.

Method	Ref.	Network	CIFAR-10	CIFAR-100
Reed *et al.* [227]	[228]	ResNet	62.3%	46.5%
Golderberger *et al.* [243]	[228]	ResNet	69.9%	45.8%
Azadi *et al.* [230]	[230]	AlexNet	75.0%	-
Jilang *et al.* [228]	[228]	ResNet	76.6%	56.9%
Zhang *et al.* [229]	-	PreAct ResNet-18	88.3%	56.4%
Standard SGD	-	PreAct ResNet-18	69.6%	44.9%
Dropout ($p = 0.3$) [244]	-	PreAct ResNet-18	84.5%	50.1%
Label Smoothing (0.1) [245]	-	PreAct ResNet-18	69.3%	46.1%
Bilevel	-	PreAct ResNet-18	87.0%	59.8%
Bilevel + [229]	-	PreAct ResNet-18	89.0%	61.6%

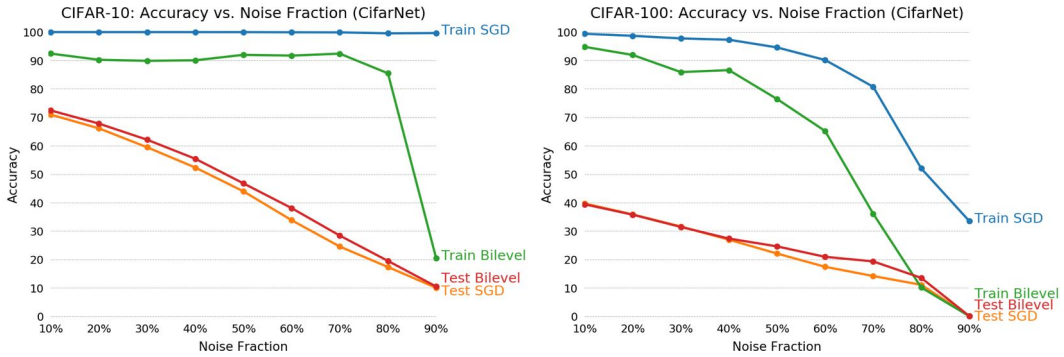


FIGURE 8.3: **CifarNet on CIFAR-10/100 with varying label noise.** We report performance with CifarNet trained on data with varying amounts of random label noise. We observe that our optimization leads to higher test accuracy and less overfitting in all cases when compared to standard SGD.

uniformly assign labels of the other $c - 1$ classes. Note that this leads to entirely random labeling in the case of 90% label noise on CIFAR-10. Networks are trained on datasets with varying amounts of label noise. We train the networks with our bilevel optimizer using eight mini-batches and using the training set for validation. The networks are trained for 100 epochs on mini-batches of size 64. Learning schedules, initial learning rates, and data augmentation are identical to those in sec. 8.4.1. The results using CifarNet are summarized in Figure 8.3 and the results for Inception in Figure 8.4. We observe a consistent improvement over standard SGD on CifarNet and significant gains for Inception on CIFAR-10 up to 70% noise. On CIFAR-100, our method leads to better results up to a noise level of 50%. We compare to state-of-the-art regularization methods and methods for dealing with label noise in Table 8.3. The networks used in the comparison are variants of the ResNet architecture [119] as specified in [228] and [229]. An exception is [230], which uses AlexNet, but relies on having a separate large dataset with clean labels for their model. We use the same architecture as the state-of-the-art method by Zhang *et al.* [229] for our results. We also explored the combination of our bilevel optimization with the data augmentation introduced by [229] in the last row. This results in the best performance on both CIFAR-10 and CIFAR-100. We also include results using Dropout

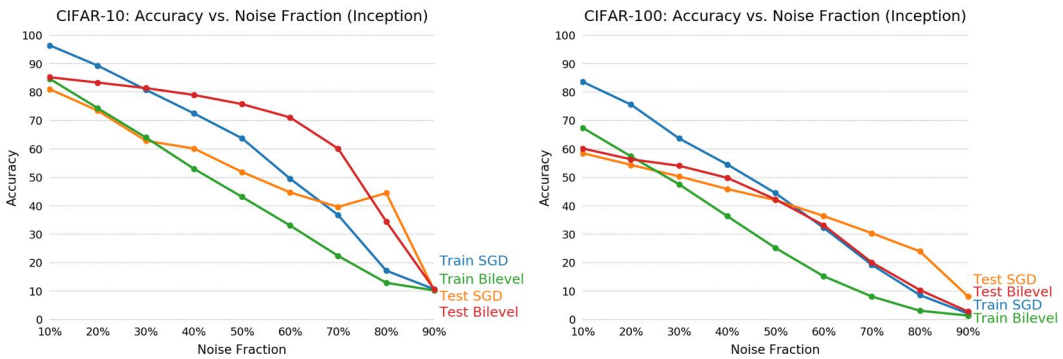


FIGURE 8.4: **Inception on CIFAR-10/100 with varying label noise.** We report performance with an Inception network trained on data with varying amounts of random label noise. On CIFAR-10, our optimization leads to substantially higher test accuracy in most cases when compared to standard SGD. Our method also shows more robustness to noise levels up to 50% on CIFAR-100.

TABLE 8.4: Experiments with more realistic label noise on ImageNet.

Method	44% Noise	Clean
SGD	50.75%	57.4%
Bilevel	52.69%	58.2%

[244] with a low keep-probability p as suggested by Arpit *et al.* [246] and results with label-smoothing as suggested by Szegedy *et al.* [245] .

Modelling Realistic Label Noise on ImageNet. In order to test the method on more realistic label noise, we perform the following experiment: We use the predicted labels of a pre-trained AlexNet to model realistic label noise. Our rationale here is that predictions of a neural network will make similar mistakes as a human annotator would. To obtain a high noise level, we leave dropout active when making the predictions on the training set. This results in approximately 44% label noise. We then retrain an AlexNet from scratch on those labels using standard SGD and our bilevel optimizer. This experiment's results and comparison on clean data are given in Table 8.4. The bilevel optimization leads to better performance in both cases, improving over standard SGD by nearly 2% in case of noisy labels.

Experiments on Real-World Data with Noisy Labels. We test our method on the Clothing1M dataset introduced by Xiao *et al.* [247]. The dataset consists of fashion images belonging to 14 classes. It contains 1M images with noisy labels and additional smaller sets with clean labels for training (50K), validation (14K), and testing (10K). We follow the same setup as the state-of-the-art by Patrini *et al.* [223] using an ImageNet pre-trained 50-layer ResNet. We achieve 69.9% after training only on the noisy data and 79.9% after fine-tuning on the clean training data. These results are comparable to [223] with 69.8% and 80.4% respectively.

8.4.4 Generalization on Small Datasets

Small datasets pose a challenge since deep networks will easily overfit in this case. We test our method under this scenario by training an AlexNet on the multi-label

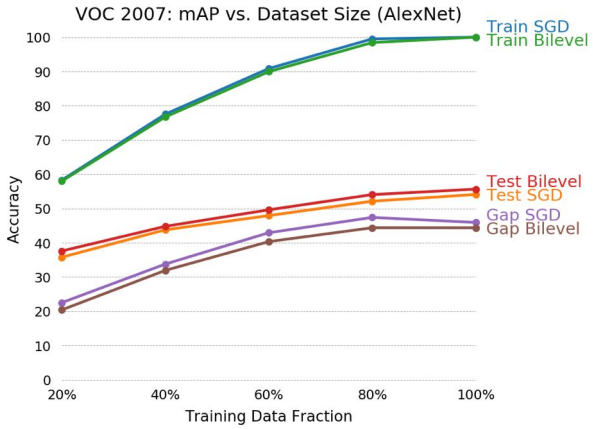


FIGURE 8.5: **Classification on varying fractions of Pascal VOC.** We train an AlexNet for multi-label classification on varying fractions of the Pascal VOC 2007 `trainval` set and report mAP on the test set and the complete `trainval` set. Our optimization technique leads to higher test performance and a smaller generalization gap in all cases.

classification task of Pascal VOC 2007. Training images are randomly cropped to an area between 30% to 100% of the original and then resized to 227×227. We linearly decay the learning rate from 0.01 to 0 and train for 1K epochs on mini-batches of size 64. We use the bilevel optimization method with four mini-batches and without a separate validation set. In Figure 8.5 we report the mAP obtained from the average prediction over ten random crops on varying fractions of the original dataset. We observe a small but consistent improvement over the baseline in all cases.

8.5 Discussion

Neural networks seem to benefit from additional regularization when compared to alternative models in machine learning. However, neural networks still suffer from overfitting, and current regularization methods have a limited impact. We introduce a novel regularization approach that implements the principles of cross-validation as a bilevel optimization problem. This formulation is computationally efficient and can be incorporated with other regularizations. It consistently improves the generalization of several neural network architectures on challenging datasets such as CIFAR10/100, Pascal VOC 2007, and ImageNet. In particular, we show that the proposed method is effective in avoiding overfitting with noisy labels.

Chapter 9

Conclusions

"What I cannot create, I do not understand."

— Richard P. Feynman

In this thesis, we studied techniques to learn representations of visual data, even without human supervision. Along the way, our aim was always to capture visual patterns that generalize to new data and new vision tasks. The primary tool towards this goal was the design of self-supervised learning tasks. In Chapter 3 we described a novel SSL approach based on the recognition and localization of corruptions in natural images. We then identified, as a general principle for SSL, the recognition of data transformations in Chapter 4. These transformations are chosen such that they alter the visual patterns we want the network to learn. To steer deep neural networks towards learning global image statistics we thus designed transformations that alter global statistics but preserve local statistics in Chapter 4. Following the same principle, we designed SSL tasks that learn motion from video in Chapter 5 by recognizing data transformations that affect the temporal domain. In Chapter 6 we then built representations of 3D shape by learning to recognize if the geometric transformation between two views of the same scene is rigid or non-rigid. All these instances of the proposed SSL design pattern illustrate its effectiveness for learning features that generalize to downstream vision tasks with few labeled examples.

Aside from self-supervised learning, this thesis also addressed generalization issues in supervised learning and generative adversarial training. In Chapter 7 we described how GAN training could be stabilized by adding random noise to the input images of the discriminator. Coincidentally, this again corresponds to a transformation of the data. In this case, however, the aim was not to make the discriminator focus on the transformation but rather to blur the boundary between fake and real examples. Finally, Chapter 8 addressed overfitting in the supervised learning setting when labels are scarce or unreliable. Our proposed algorithm improves generalization by modulating the magnitudes and signs of training gradients based on their agreement with a validation gradient.

Self-supervised learning has made tremendous progress recently and is fast approaching or surpassing the performance of supervised pre-training. Effective SSL techniques will allow us to tackle more complex vision tasks for which large supervised datasets are impractical. It is, therefore, our belief that self-supervised learning is an important stepping stone on the path towards artificial systems with visual perception and reasoning capabilities that match or surpass those of humans. With some luck, we might also further our understanding of human perception and learning on the way. Towards this goal, we outline below three possible directions that continue this work.

Learning from Video. SSL on images produces representations that capture well the semantics of images as measured by recognition or localization tasks. However, the visual understanding of current systems is still far from human capabilities, and mostly devoid of common sense reasoning (e.g., about human interactions) or an understanding of the laws of nature (i.e., Newtonian physics). Such understanding is crucial for artificial vision but difficult, if not impossible, to gain from static images. SSL on videos has the potential to evolve richer visual representations by observing the world in motion, similar to humans. The processing and learning from video poses computational challenges that might require novel architectures to achieve long-term reasoning. Indeed, current 3D convolutional networks typically only process a relatively small number of frames corresponding to a few seconds of video. We also largely lack benchmarks beyond action recognition to measure deeper visual understanding.

Directly Solving Vision Tasks via Self-Supervision. While self-supervised pre-training often drastically reduces the demand for supervision in transfer learning, large differences in the pre-training and target tasks can hamper its effectiveness. Furthermore, some tasks may require a level of dense supervision that is even prohibitive for transfer learning (*e.g.*, segmentation or pose-estimation on videos). One solution to these problems is the design SSL tasks that directly solve vision problems of interest. Such tasks could, for example, exploit motion cues in videos for segmentation, multi-view data for 3D correspondence learning, or audio-visual data to predict object interactions.

Going Beyond Hand-Crafted Self-Supervised Learning Tasks. The optimal visual representation for a given transfer learning task likely depends on the target task, *i.e.*, there might not be a universally optimal representation. In this case, the properties of features (*e.g.*, learned invariances and equivariances) will depend on the target task and require the design of targeted SSL tasks, which at present relies on intuition and trial and error. A more desirable system would adapt the parameters of a flexible SSL framework towards learning only those patterns relevant to the target task. This could be achieved through a meta-learning algorithm that optimizes feature performance in transfer learning. The design of such a framework could build on the learning of invariances and equivariances to predefined or learned data transformations, as explored in this thesis.

Bibliography

[1] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness", *Arxiv preprint arxiv:1811.12231*, 2018.

[2] K. Schindler and L. Van Gool, "Action snippets: How many frames does human action recognition require?", in *2008 ieee conference on computer vision and pattern recognition*, IEEE, 2008, pp. 1–8.

[3] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction", in *Proceedings of the ieee international conference on computer vision*, 2015, pp. 1422–1430.

[4] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles", in *European conference on computer vision*, Springer, 2016, pp. 69–84.

[5] S. Jenni and P. Favaro, "Self-supervised feature learning by learning to spot artifacts", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018, pp. 2733–2742.

[6] S. Jenni, H. Jin, and P. Favaro, "Steering self-supervised feature learning beyond local pixel statistics", in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 6408–6417.

[7] S. Jenni, G. Meishvili, and P. Favaro, "Video representation learning by recognizing temporal transformations", in *European conference on computer vision (ECCV)*, 2020.

[8] S. Jenni and P. Favaro, "Self-supervised multi-view synchronization learning for 3d pose estimation", in *Asian conference on computer vision (ACCV)*, 2020.

[9] ——, "On stabilizing generative adversarial training with noise", in *The ieee conference on computer vision and pattern recognition (cvpr)*, 2019.

[10] ——, "Deep bilevel learning", in *Proceedings of the european conference on computer vision (eccv)*, 2018, pp. 618–633.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets", *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[13] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory", Colorado Univ at Boulder Dept of Computer Science, Tech. Rep., 1986.

[14] M. A. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning.", in *Aistats*, Citeseer, vol. 10, 2005, pp. 33–40.

[15] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length, and helmholtz free energy", *Advances in neural information processing systems*, pp. 3–3, 1994.

[16] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, *et al.*, "Greedy layer-wise training of deep networks", *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[17] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction", in *International conference on artificial neural networks*, Springer, 2011, pp. 52–59.

[18] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion", *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", in *Iclr*, 2014.

[20] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "Beta-vae: Learning basic visual concepts with a constrained variational framework", in *Iclr*, 2016.

[21] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning", *Arxiv preprint arxiv:1711.00937*, 2017.

[22] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow, "Adversarial autoencoders", *Arxiv preprint arxiv:1511.05644*, 2015.

[23] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks", *Arxiv preprint arxiv:1611.07004*, 2016.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets", in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[25] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks", *Arxiv preprint arxiv:1511.06434*, 2015.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space", *Arxiv preprint arxiv:1301.3781*, 2013.

[27] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning", *International conference on learning representations*, 2017.

[28] J. Donahue and K. Simonyan, "Large scale adversarial representation learning", *Arxiv preprint arxiv:1907.02544*, 2019.

[29] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[30] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "Deblurgan: Blind motion deblurring using conditional adversarial networks", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018, pp. 8183–8192.

[31] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[32] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks", in *Proceedings of the ieee international conference on computer vision*, 2017, pp. 2223–2232.

[33] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans", in *Advances in neural information processing systems*, 2016, pp. 2226–2234.

[34] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks", *Arxiv preprint arxiv:1701.04862*, 2017.

[35] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan", *Arxiv:1701.07875*, 2017.

[36] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans", in *Advances in neural information processing systems*, 2017, pp. 5769–5779.

[37] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks", in *2017 ieee international conference on computer vision (iccv)*, IEEE, 2017, pp. 2813–2821.

[38] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network", *Arxiv preprint arxiv:1609.03126*, 2016.

[39] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks", *Arxiv preprint arxiv:1703.10717*, 2017.

[40] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "How to train your dragan", *Arxiv preprint arxiv:1705.07215*, 2017.

[41] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are gans created equal? a large-scale study", *Arxiv preprint arxiv:1711.10337*, 2017.

[42] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann, "Stabilizing training of generative adversarial networks through regularization", in *Advances in neural information processing systems*, 2017, pp. 2015–2025.

[43] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks", *Arxiv preprint arxiv:1802.05957*, 2018.

[44] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis", *Arxiv preprint arxiv:1809.11096*, 2018.

[45] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation", in *International conference on learning representations*, 2018.

[46] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks", in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[47] R. Caruana and V. R. de Sa, "Promoting poor features to supervisors: Some inputs work better as outputs", *Nips*, 1996.

[48] R. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data", *Jmlr*, 2005.

[49] U. Büchler, B. Brattoli, and B. Ommer, "Improving spatiotemporal self-supervision by deep reinforcement learning", *Arxiv preprint arxiv:1807.11293*, 2018.

[50] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash, "Boosting self-supervised learning via knowledge transfer", in *Cvpr*, 2018.

[51] Mundhenk *et al.*, "Improvements to context based self-supervised learning", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018.

[52] E. Denton, S. Gross, and R. Fergus, "Semi-supervised learning with context-conditional generative adversarial networks", *Arxiv preprint arxiv:1611.06430*, 2016.

[53] M. Noroozi, H. Pirsiavash, and P. Favaro, "Representation learning by learning to count", *Iccv*, 2017.

[54] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations", in *International conference on learning representations*, 2018.

[55] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization", in *European conference on computer vision*, Springer, 2016, pp. 649–666.

[56] ——, "Split-brain autoencoders: Unsupervised learning by cross-channel prediction", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 1058–1067.

[57] G. Larsson, M. Maire, and G. Shakhnarovich, "Colorization as a proxy task for visual understanding", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 6874–6883.

[58] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy, "Tracking emerges by colorizing videos", in *Proc. eccv*, 2018.

[59] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos", in *Proceedings of the ieee international conference on computer vision*, 2015, pp. 2794–2802.

[60] D. Gordon, K. Ehsani, D. Fox, and A. Farhadi, "Watching the world go by: Representation learning from unlabeled videos", *Arxiv preprint arxiv:2003.07990*, 2020.

[61] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan, "Learning features by watching objects move", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017.

[62] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba, "Ambient sound provides supervision for visual learning", in *European conference on computer vision*, Springer, 2016, pp. 801–816.

[63] A. Owens and A. A. Efros, "Audio-visual scene analysis with self-supervised multisensory features", in *The european conference on computer vision (eccv)*, 2018.

[64] R. Arandjelovic and A. Zisserman, "Look, listen and learn", in *2017 ieee international conference on computer vision (iccv)*, IEEE, 2017, pp. 609–617.

[65] H. Zhao, C. Gan, A. Rouditchenko, C. Vondrick, J. McDermott, and A. Torralba, "The sound of pixels", in *Proceedings of the european conference on computer vision*, 2018, pp. 570–586.

[66] R. Gao, R. Feris, and K. Grauman, "Learning to separate object sounds by watching unlabeled video", *Arxiv preprint arxiv:1804.01665*, 2018.

[67] P. Agrawal, J. Carreira, and J. Malik, "Learning to see by moving", *The ieee international conference on computer vision (iccv)*, 2015.

[68] L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta, "The curious robot: Learning visual representations via physical interactions", in *European conference on computer vision*, Springer, 2016, pp. 3–18.

[69] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks", in *Advances in neural information processing systems*, 2014, pp. 766–774.

[70] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018, pp. 3733–3742.

[71] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features", in *Proceedings of the european conference on computer vision*, 2018, pp. 132–149.

[72] C. Zhuang, A. L. Zhai, and D. Yamins, "Local aggregation for unsupervised learning of visual embeddings", in *Proceedings of the ieee international conference on computer vision*, 2019, pp. 6002–6012.

[73] Z. Ren and Y. Jae Lee, "Cross-domain self-supervised multi-task feature learning using synthetic imagery", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018, pp. 762–771.

[74] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning", in *Proceedings of the ieee international conference on computer vision*, 2017, pp. 2051–2060.

[75] Z. Feng, C. Xu, and D. Tao, "Self-supervised representation learning by rotation feature decoupling", in *The ieee conference on computer vision and pattern recognition*, 2019.

[76] L. Jing, X. Yang, J. Liu, and Y. Tian, "Self-supervised spatiotemporal feature learning via video rotation prediction", *Arxiv preprint arxiv:1811.11387*, 2018.

[77] T. Han, W. Xie, and A. Zisserman, "Video representation learning by dense predictive coding", in *Proceedings of the ieee international conference on computer vision workshops*, 2019, pp. 0–0.

[78] C. Sun, F. Baradel, K. Murphy, and C. Schmid, "Contrastive bidirectional transformer for temporal representation learning", *Arxiv preprint arxiv:1906.05743*, 2019.

[79] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and learn: Unsupervised learning using temporal order verification", in *European conference on computer vision*, Springer, 2016, pp. 527–544.

[80] B. Brattoli, U. Büchler, A.-S. Wahl, M. E. Schwab, and B. Ommer, "Lstm self-supervision for detailed behavior analysis", in *Proceedings of the ieee conference on computer vision and pattern recognition (cvpr)*, vol. 2, 2017.

[81] B. Fernando, H. Bilen, E. Gavves, and S. Gould, "Self-supervised video representation learning with odd-one-out networks", in *Computer vision and pattern recognition (cvpr), 2017 ieee conference on*, IEEE, 2017, pp. 5729–5738.

[82] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang, "Unsupervised representation learning by sorting sequences", in *Proceedings of the ieee international conference on computer vision*, 2017, pp. 667–676.

[83] D. Kim, D. Cho, and I. S. Kweon, "Self-supervised video representation learning with space-time cubic puzzles", in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 2019, pp. 8545–8552.

[84] D. Wei, J. Lim, A. Zisserman, and W. T. Freeman, "Learning and using the arrow of time", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018, pp. 8052–8060.

[85] D. Epstein, B. Chen, and C. Vondrick, "Oops! predicting unintentional action in video", in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 919–929.

[86] S. Benaim, A. Ephrat, O. Lang, I. Mosseri, W. T. Freeman, M. Rubinstein, M. Irani, and T. Dekel, "Speednet: Learning the speediness in videos", in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 9922–9931.

[87] Y. Yao, C. Liu, D. Luo, Y. Zhou, and Q. Ye, "Video playback rate perception for self-supervised spatio-temporal representation learning", in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 6548–6557.

[88] B. Korbar, D. Tran, and L. Torresani, "Cooperative learning of audio and video models from self-supervised synchronization", in *Advances in neural information processing systems*, 2018, pp. 7763–7774.

[89] J. Wang, J. Jiao, L. Bao, S. He, Y. Liu, and W. Liu, "Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2019, pp. 4006–4015.

[90] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei, "Unsupervised learning of long-term motion dynamics for videos", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 2203–2212.

[91] C. Gan, B. Gong, K. Liu, H. Su, and L. J. Guibas, "Geometry guided convolutional neural networks for self-supervised video representation learning", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018, pp. 5589–5597.

[92] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere", in *International conference on machine learning*, PMLR, 2020, pp. 9929–9939.

[93] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations", *Arxiv preprint arxiv:2002.05709*, 2020.

[94] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning", in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[95] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning", *Arxiv preprint arxiv:2006.07733*, 2020.

[96] X. Chen and K. He, "Exploring simple siamese representation learning", *Arxiv preprint arxiv:2011.10566*, 2020.

[97] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments", *Arxiv preprint arxiv:2006.09882*, 2020.

[98] X. Wang, Z. Liu, and S. X. Yu, "Unsupervised feature learning by cross-level discrimination between instances and groups", *Arxiv preprint arxiv:2008.03813*, 2020.

[99] T. Xiao, X. Wang, A. A. Efros, and T. Darrell, "What should not be contrastive in contrastive learning", *Arxiv preprint arxiv:2008.05659*, 2020.

[100] R. Qian, T. Meng, B. Gong, M.-H. Yang, H. Wang, S. Belongie, and Y. Cui, "Spatiotemporal contrastive video representation learning", *Arxiv preprint arxiv:2008.03800*, 2020.

[101] I. Dave, R. Gupta, M. N. Rizve, and M. Shah, "Tclr: Temporal contrastive learning for video representation", *Arxiv preprint arxiv:2101.07974*, 2021.

[102] M. Patrick, Y. M. Asano, R. Fong, J. F. Henriques, G. Zweig, and A. Vedaldi, "Multi-modal self-supervision from generalized data transformations", *Arxiv preprint arxiv:2003.04298*, 2020.

[103] Y. Bai, H. Fan, I. Misra, G. Venkatesh, Y. Lu, Y. Zhou, Q. Yu, V. Chandra, and A. Yuille, "Can temporal information help with contrastive self-supervised learning?", *Arxiv preprint arxiv:2011.13046*, 2020.

[104] J. Wang, J. Jiao, and Y.-H. Liu, "Self-supervised video representation learning by pace prediction", in *European conference on computer vision*, Springer, 2020, pp. 504–521.

[105] L. Tao, X. Wang, and T. Yamasaki, "Self-supervised video representation using pretext-contrastive learning", *Arxiv preprint arxiv:2010.15464*, 2020.

[106] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding", *Arxiv preprint arxiv:1906.05849*, 2019.

[107] T. Han, W. Xie, and A. Zisserman, "Self-supervised co-training for video representation learning", *Arxiv preprint arxiv:2010.09709*, 2020.

[108] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman, "End-to-end learning of visual representations from uncurated instructional videos", in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 9879–9889.

[109] T. Afouras, A. Owens, J. S. Chung, and A. Zisserman, "Self-supervised learning of audio-visual objects from video", *Arxiv preprint arxiv:2008.04237*, 2020.

[110] H. Alwassel, D. Mahajan, B. Korbar, L. Torresani, B. Ghanem, and D. Tran, "Self-supervised learning by cross-modal audio-video clustering", *Arxiv preprint arxiv:1911.12667*, 2019.

[111] R Devon *et al.*, "Representation learning with video deep infomax", *Arxiv preprint arxiv:2007.13278*, 2020.

[112] F. Xue, H. Ji, W. Zhang, and Y. Cao, "Self-supervised video representation learning by maximizing mutual information", *Signal processing: Image communication*, vol. 88, p. 115 967, 2020.

[113] D. Jayaraman and K. Grauman, "Learning image representations tied to egomotion", in *Proceedings of the ieee international conference on computer vision*, 2015, pp. 1413–1421.

[114] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting", in *Proceedings of the 27th annual conference on computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424.

[115] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in *Computer vision and pattern recognition, 2009. cvpr 2009. ieee conference on*, IEEE, 2009, pp. 248–255.

[116] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge", *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[117] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning", in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[118] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *International conference on machine learning*, 2015, pp. 448–456.

[119] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[120] D. Kingma and J. Ba, "Adam: A method for stochastic optimization", *Arxiv preprint arxiv:1412.6980*, 2014.

[121] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training", *Arxiv preprint arxiv:1612.07828*, 2016.

[122] A. Dundar, J. Jin, and E. Culurciello, "Convolutional clustering for unsupervised learning", *Arxiv preprint arxiv:1511.06241*, 2015.

[123] C. Huang, C. Change Loy, and X. Tang, "Unsupervised learning of discriminative attributes and visual representations", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 5175–5184.

[124] K. Swersky, J. Snoek, and R. P. Adams, "Multi-task bayesian optimization", in *Advances in neural information processing systems*, 2013, pp. 2004–2012.

[125] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun, "Stacked what-where auto-encoders", *Arxiv preprint arxiv:1506.02351*, 2015.

[126] P. Agrawal, J. Carreira, and J. Malik, "Learning to see by moving", in *Proceedings of the ieee international conference on computer vision*, 2015, pp. 37–45.

[127] P. Bojanowski and A. Joulin, "Unsupervised learning by predicting noise", in *Proceedings of the 34th international conference on machine learning-volume 70*, 2017, pp. 517–526.

[128] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell, "Data-dependent initializations of convolutional neural networks", *Arxiv preprint arxiv:1511.06856*, 2015.

[129] R. Girshick, "Fast r-cnn", in *Proceedings of the ieee international conference on computer vision*, 2015, pp. 1440–1448.

[130] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[131] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database", in *Advances in neural information processing systems 27*, Curran Associates, Inc., 2014, pp. 487–495.

[132] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[133] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization", in *Deep learning workshop, international conference on machine learning (icml)*, 2015.

[134] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization", *Arxiv preprint arxiv:1610.02391*, 2016.

[135] L. Zhang, G.-J. Qi, L. Wang, and J. Luo, "Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2019, pp. 2547–2555.

[136] M. Huh, A. Liu, A. Owens, and A. A. Efros, "Fighting fake news: Image splice detection via learned self-consistency", in *Proceedings of the european conference on computer vision*, 2018, pp. 101–117.

[137] S.-Y. Wang, O. Wang, A. Owens, R. Zhang, and A. A. Efros, "Detecting photoshopped faces by scripting photoshop", *Arxiv preprint arxiv:1906.05856*, 2019.

[138] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, "Learning rich features for image manipulation detection", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018, pp. 1053–1061.

[139] J. H. Bappy, A. K. Roy-Chowdhury, J. Bunk, L. Nataraj, and B. Manjunath, "Exploiting spatial structure for localizing manipulated image regions", in *Proceedings of the ieee international conference on computer vision*, 2017, pp. 4970–4979.

[140] J. H. Lim and J. C. Ye, "Geometric gan", *Arxiv preprint arxiv:1705.02894*, 2017.

[141] J. Duchon, "Splines minimizing rotation-invariant semi-norms in sobolev spaces", in *Constructive theory of functions of several variables*, Springer, 1977, pp. 85–100.

[142] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild", in *Proceedings of international conference on computer vision (iccv)*, 2015.

[143] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam", *Arxiv preprint arxiv:1711.05101*, 2017.

[144] ——, "Sgdr: Stochastic gradient descent with warm restarts", *Arxiv preprint arxiv:1608.03983*, 2016.

[145] A. Mahendran, J. Thewlis, and A. Vedaldi, "Cross pixel optical-flow similarity for self-supervised learning", in *Asian conference on computer vision*, Springer, 2018, pp. 99–116.

[146] Zhan *et al.*, "Self-supervised learning via conditional motion propagation", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2019.

[147] J. Huang, Q. Dong, S. Gong, and X. Zhu, "Unsupervised deep learning by neighbourhood discovery", *Arxiv preprint arxiv:1904.11567*, 2019.

[148] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization", *Arxiv preprint arxiv:1808.06670*, 2018.

[149] X. Ji, J. F. Henriques, and A. Vedaldi, "Invariant information clustering for unsupervised image classification and segmentation", in *Proceedings of the ieee international conference on computer vision*, 2019, pp. 9865–9874.

[150] E. Oyallon, E. Belilovsky, and S. Zagoruyko, "Scaling the scattering transform: Deep hybrid networks", in *Proceedings of the ieee international conference on computer vision*, 2017, pp. 5618–5627.

[151] X. Li, S. Liu, S. De Mello, X. Wang, J. Kautz, and M.-H. Yang, "Joint-task self-supervised learning for temporal correspondence", in *Advances in neural information processing systems*, 2019, pp. 317–327.

[152] B. K. Iwana and S. Uchida, "Time series classification using local distance-based features in multi-modal fusion networks", *Pattern recognition*, vol. 97, p. 107 024, 2020, ISSN: 0031-3203.

[153] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks", in *Proceedings of the ieee international conference on computer vision*, 2015, pp. 4489–4497.

[154] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018, pp. 6546–6555.

[155] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018, pp. 6450–6459.

[156] A. Zisserman, J. Carreira, K. Simonyan, W. Kay, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, *et al.*, "The kinetics human action video dataset", *Arxiv*, 2017.

[157] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild", *Arxiv preprint arxiv:1212.0402*, 2012.

[158] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition", in *Proceedings of the international conference on computer vision (iccv)*, 2011.

[159] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, and Y. Zhuang, "Self-supervised spatiotemporal learning via video clip order prediction", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2019, pp. 10 334–10 343.

[160] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, *Striving for simplicity: The all convolutional net*, 2014. arXiv: 1412.6806 [cs.LG].

[161] B. Rosenhahn, R. Klette, and D. Metaxas, *Human motion: Understanding, modeling, capture*. Springer, 2008.

[162] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments", *Ieee transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1325–1339, 2014.

[163] V. Nityananda and J. C. Read, "Stereopsis in animals: Evolution, function and mechanisms", *Journal of experimental biology*, vol. 220, no. 14, pp. 2502–2512, 2017.

[164] D. P. Harland, D. Li, and R. R. Jackson, "How jumping spiders see the world.", *Oxford university press*, 2012.

[165] S. Wu, C. Rupprecht, and A. Vedaldi, "Unsupervised learning of probably symmetric deformable 3d objects from images in the wild", in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 1–10.

[166] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang, "Hologan: Unsupervised learning of 3d representations from natural images", in *Proceedings of the ieee international conference on computer vision*, 2019, pp. 7588–7597.

[167] A. Szabó, G. Meishvili, and P. Favaro, "Unsupervised generative 3d shape learning from natural images", *Arxiv preprint arxiv:1910.00287*, 2019.

[168] A. Agarwala, K. C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski, "Panoramic video textures", in *Acm siggraph 2005 papers*, 2005, pp. 821–827.

[169] P. Sand and S. Teller, "Video matching", *Acm transactions on graphics (tog)*, vol. 23, no. 3, pp. 592–599, 2004.

[170] T. Tuytelaars and L. Van Gool, "Synchronizing video sequences", in *Proceedings of the 2004 ieee computer society conference on computer vision and pattern recognition, 2004. cvpr 2004.*, IEEE, vol. 1, 2004, pp. I–I.

[171] P. Wieschollek, I. Freeman, and H. P. Lensch, "Learning robust video synchronization without annotations", in *2017 16th ieee international conference on machine learning and applications (icmla)*, IEEE, 2017, pp. 92–100.

[172] J. Liang, P. Huang, J. Chen, and A. Hauptmann, "Synchronization for multi-perspective videos in the wild", in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, IEEE, 2017, pp. 1592–1596.

[173] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2014, pp. 3686–3693.

[174] S. Li and A. B. Chan, "3d human pose estimation from monocular images with deep convolutional neural network", in *Asian conference on computer vision*, Springer, 2014, pp. 332–347.

[175] S. Li, W. Zhang, and A. B. Chan, "Maximum-margin structured learning with deep networks for 3d human pose estimation", in *Proceedings of the ieee international conference on computer vision*, 2015, pp. 2848–2856.

[176] B. Tekin, I. Katircioglu, M. Salzmann, V. Lepetit, and P. Fua, "Structured prediction of 3d human pose with deep neural networks", *Arxiv preprint arxiv: 1605.05180*, 2016.

[177] X. Zhou, X. Sun, W. Zhang, S. Liang, and Y. Wei, "Deep kinematic pose regression", in *European conference on computer vision*, Springer, 2016, pp. 186–201.

[178] B. Tekin, P. Márquez-Neila, M. Salzmann, and P. Fua, "Learning to fuse 2d and 3d image cues for monocular body pose estimation", in *Proceedings of the ieee international conference on computer vision*, 2017, pp. 3941–3950.

[179] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Harvesting multiple views for marker-less 3d human pose annotations", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 6988–6997.

[180] ——, "Coarse-to-fine volumetric prediction for single-image 3d human pose", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 7025–7034.

[181] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3d human pose estimation in the wild using improved cnn supervision", in *2017 international conference on 3d vision (3dv)*, IEEE, 2017, pp. 506–516.

[182] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei, "Integral human pose regression", in *Proceedings of the european conference on computer vision (eccv)*, 2018, pp. 529–545.

[183] A.-I. Popa, M. Zanfir, and C. Sminchisescu, "Deep multitask architecture for integrated 2d and 3d human sensing", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 6289–6298.

[184] D. Tome, C. Russell, and L. Agapito, "Lifting from the deep: Convolutional 3d pose estimation from a single image", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 2500–2509.

[185] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: Real-time 3d human pose estimation with a single rgb camera", *Acm transactions on graphics (tog)*, vol. 36, no. 4, pp. 1–14, 2017.

[186] G. Rogez, P. Weinzaepfel, and C. Schmid, "Lcr-net: Localization-classification-regression for human pose", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 3433–3441.

[187] R. Dabral, A. Mundhada, U. Kusupati, S. Afaque, A. Sharma, and A. Jain, "Learning 3d human pose from structure and motion", in *Proceedings of the european conference on computer vision (eccv)*, 2018, pp. 668–683.

[188] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation", in *Proceedings of the ieee international conference on computer vision*, 2017, pp. 2640–2649.

[189] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Towards 3d human pose estimation in the wild: A weakly-supervised approach", in *Proceedings of the ieee international conference on computer vision*, 2017, pp. 398–407.

[190] F. Moreno-Noguer, "3d human pose estimation from a single image via distance matrix regression", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 2823–2832.

[191] M. Rayat Imtiaz Hossain and J. J. Little, "Exploiting temporal information for 3d human pose estimation", in *Proceedings of the european conference on computer vision (eccv)*, 2018, pp. 68–84.

[192] H.-S. Fang, Y. Xu, W. Wang, X. Liu, and S.-C. Zhu, "Learning pose grammar to encode human body configuration for 3d pose estimation", in *Thirty-second aaai conference on artificial intelligence*, 2018.

[193] C.-H. Chen and D. Ramanan, "3d human pose estimation= 2d pose estimation+ matching", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 7035–7043.

[194] L. Zhao, X. Peng, Y. Tian, M. Kapadia, and D. N. Metaxas, "Semantic graph convolutional networks for 3d human pose regression", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2019, pp. 3425–3435.

[195] S. Sharma, P. T. Varigonda, P. Bindal, A. Sharma, and A. Jain, "Monocular 3d human pose estimation by generation and ordinal ranking", in *Proceedings of the ieee international conference on computer vision*, 2019, pp. 2325–2334.

[196] K. Wang, L. Lin, C. Jiang, C. Qian, and P. Wei, "3d human pose machines with self-supervised learning", *Ieee transactions on pattern analysis and machine intelligence*, vol. 42, no. 5, pp. 1069–1082, 2019.

[197] W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-Or, and B. Chen, "Synthesizing training images for boosting human 3d pose estimation", in *2016 fourth international conference on 3d vision (3dv)*, IEEE, 2016, pp. 479–488.

[198] G. Rogez and C. Schmid, "Mocap-guided data augmentation for 3d pose estimation in the wild", in *Advances in neural information processing systems*, 2016, pp. 3108–3116.

[199] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 109–117.

[200] M. Kocabas, S. Karagoz, and E. Akbas, "Self-supervised learning of 3d human pose using multi-view geometry", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2019, pp. 1077–1086.

[201] H. Rhodin, J. Spörri, I. Katircioglu, V. Constantin, F. Meyer, E. Müller, M. Salzmann, and P. Fua, "Learning monocular 3d human pose estimation from multi-view images", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018, pp. 8437–8446.

[202] X. Chen, K.-Y. Lin, W. Liu, C. Qian, and L. Lin, "Weakly-supervised discovery of geometry-aware representation for 3d human pose estimation", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2019, pp. 10 895–10 904.

[203] C.-H. Chen, A. Tyagi, A. Agrawal, D. Drover, S. Stojanov, and J. M. Rehg, "Unsupervised 3d pose estimation with geometric self-supervision", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2019, pp. 5714–5724.

[204] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2018, pp. 7122–7131.

[205] B. Wandt and B. Rosenhahn, "Repnet: Weakly supervised training of an adversarial reprojection network for 3d human pose estimation", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2019, pp. 7782–7791.

[206] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, "3d human pose estimation in video with temporal convolutions and semi-supervised training", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2019, pp. 7753–7762.

[207] H. Rhodin, M. Salzmann, and P. Fua, "Unsupervised geometry-aware representation for 3d human pose estimation", in *Proceedings of the european conference on computer vision (eccv)*, 2018, pp. 750–767.

[208] R. Mitra, N. B. Gundavarapu, A. Sharma, and A. Jain, "Multiview-consistent semi-supervised learning for 3d human pose estimation", in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 6907–6916.

[209] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár, "Amortised map inference for image super-resolution", *Arxiv preprint arxiv:1610.04490*, 2016.

[210] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images", 2009.

[211] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge", *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[212] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop", *Corr*, vol. abs/1506.03365, 2015.

[213] M. S. Sajjadi and B. Schölkopf, "Tempered adversarial networks", *Arxiv preprint arxiv:1802.04374*, 2018.

[214] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu, *How to train a gan? tips and tricks to make gans work*, `https://github.com/soumith/ganhacks`, 2016.

[215] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a nash equilibrium", *Arxiv preprint arxiv:1706.08500*, 2017.

[216] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization", in *International conference on learning representations*, 2017.

[217] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior", in *The ieee conference on computer vision and pattern recognition (cvpr)*, 2018.

[218] J. Bracken and J. T. McGill, "Mathematical programs with optimization problems in the constraints", *Oper. res.*, vol. 21, no. 1, pp. 37–44, Feb. 1973, ISSN: 0030-364X.

[219] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization", *Annals of operations research*, vol. 153, no. 1, pp. 235–256, 2007, ISSN: 1572-9338.

[220] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio, "Generalization in deep learning", *Arxiv preprint arxiv:1710.05468*, 2017.

[221] C. Zhang, Q. Liao, A. Rakhlin, K. Sridharan, B. Miranda, N. Golowich, and T. Poggio, "Theory of deep learning iii: Generalization properties of sgd", Center for Brains, Minds and Machines (CBMM), Tech. Rep., 2017.

[222] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, "Deep learning is robust to massive label noise", *Arxiv preprint arxiv:1705.10694*, 2017.

[223] G. Patrini, A. Rozza, A. Menon, R. Nock, and L. Qu, "Making neural networks robust to label noise: A loss correction approach", in *Computer vision and pattern recognition*, 2017.

[224] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels", in *Advances in neural information processing systems*, 2013, pp. 1196 –1204.

[225] I. Jindal, M. Nokleby, and X. Chen, "Learning deep networks from noisy labels with dropout regularization", *Arxiv preprint arxiv:1705.03419*, 2017.

[226] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels", *Arxiv preprint arxiv:1406.2080*, 2014.

[227] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping", *Arxiv preprint arxiv:1412.6596*, 2014.

[228] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "Mentornet: Regularizing very deep neural networks on corrupted labels", *Arxiv preprint arxiv:1712.05055*, 2017.

[229] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization", in *International conference on learning representations*, 2017.

[230] S. Azadi, J. Feng, S. Jegelka, and T. Darrell, "Auxiliary image regularization for deep cnns with noisy labels", in *International conference on learning representations*, 2016.

[231] Y. Bengio, "Gradient-based optimization of hyperparameters", *Neural computation*, vol. 12, no. 8, pp. 1889–1900, 2000.

[232] A. G. Baydin and B. A. Pearlmutter, "Automatic differentiation of algorithms for machine learning", *Arxiv preprint arxiv:1404.7456*, 2014.

[233] J. Domke, "Generic methods for optimization-based modeling", in *Artificial intelligence and statistics*, 2012, pp. 318–326.

[234] P. Ochs, R. Ranftl, T. Brox, and T. Pock, "Bilevel optimization with nonsmooth lower level problems", in *International conference on scale space and variational methods in computer vision*, Springer, 2015, pp. 654–665.

[235] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning", in *International conference on machine learning*, 2015, pp. 2113–2122.

[236] K. Kunisch and T. Pock, "A bilevel optimization approach for parameter learning in variational models", *Siam journal on imaging sciences*, vol. 6, no. 2, pp. 938–983, 2013.

[237] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks", *Arxiv preprint arxiv:1703.03400*, 2017.

[238] A. Nichol and J. Schulman, "Reptile: A scalable metalearning algorithm", *Arxiv preprint arxiv:1803.02999*, 2018.

[239] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning", in *Advances in neural information processing systems*, 2016, pp. 3630–3638.

[240] A. Hadjidimos, "Successive overrelaxation (sor) and related methods", *J. comput. appl. math.*, vol. 123, no. 1-2, pp. 177–199, Nov. 2000, ISSN: 0377-0427.

[241] N. Qian, "On the momentum term in gradient descent learning algorithms", *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[242] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in *Computer vision and pattern recognition*, IEEE, 2009, pp. 248–255.

[243] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer", in *International conference on learning representations*, 2016.

[244] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting.", *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[245] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[246] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, *et al.*, "A closer look at memorization in deep networks", *Arxiv preprint arxiv:1706.05394*, 2017.

[247] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015, pp. 2691–2699.

Erklärung

gemäss Art. 18 PromR Phil.-nat. 2019

Name/Vorname: Jenni Simon

Matrikelnummer: 09-116-005

Studiengang: Informatik

Bachelor ☐ Master ☐ Dissertation ☑

Titel der Arbeit: Learning Generalizable Visual Patterns Without Human Supervision

LeiterIn der Arbeit: Prof. Dr. Paolo Favaro

Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quel-len benutzt habe. Alle Stellen, die wörtlich oder sinn-gemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andern-falls der Senat gemäss Artikel 36 Absatz 1 Buchstabe r des Gesetzes über die Universität vom 5. September 1996 und Artikel 69 des Universitätssta-tuts vom 7. Juni 2011 zum Entzug des Doktortitels be-rechtigt ist.
Für die Zwecke der Begutachtung und der Überprüfung der Einhaltung der Selbständigkeitserklärung bzw. der Reglemente betreffend Plagiate erteile ich der Univer-sität Bern das Recht, die dazu erforderlichen Perso-nendaten zu bearbeiten und Nutzungshandlungen vor-zunehmen, insbesondere die Doktorarbeit zu vervielfäl-tigen und dauerhaft in einer Datenbank zu speichern sowie diese zur Überprüfung von Arbeiten Dritter zu verwenden oder hierzu zur Verfügung zu stellen.

Wangen bei Olten, 29.03.2021

Ort/Datum

Simon Jenni

Unterschrift