Pragmatic Investigations of Applied Deep Learning in Computer Vision Applications

by

Alireza Shafaei

M.Sc., The University of British Columbia, 2015

B.Sc., Amirkabir University of Technology, 2013

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Computer Science)

The University of British Columbia

(Vancouver)

December 2020

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Pragmatic Investigations of Applied Deep Learning in Computer Vision Applications

submitted by **Alireza Shafaei** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Computer Science**.

Examining Committee:

James J. Little, Computer Science, UBC
Supervisor

Mark Schmidt, Computer Science, UBC
Co-supervisor

Leonid Sigal, Computer Science, UBC
Supervisory Committee Member

Michiel van de Panne, Computer Science, UBC
University Examiner

Rabab K. Ward, Electrical and Computer Engineering, UBC
University Examiner

Abstract

Deep neural networks have dominated performance benchmarks on numerous machine learning tasks. These models now power the core technology of a growing list of products such as Google Search, Google Translate, Apple Siri, and even Snapchat, to mention a few. We first address two challenges in the real-world applications of deep neural networks in computer vision: data scarcity and prediction reliability. We present a new approach to data collection through synthetic data via video games that is cost-effective and can produce high-quality labelled training data on a large scale. We validate the effectiveness of synthetic data on multiple problems through cross-dataset evaluation and simple adaptive techniques. We also examine the reliability of neural network predictions in computer vision problems and show that these models are fragile on out-of-distribution test data. Motivated by statistical learning theory, we argue that it is necessary to detect out-of-distribution samples before relying on the predictions. To facilitate the development of reliable out-of-distribution sample detectors, we present a less biased evaluation framework. Using our framework, we thoroughly evaluate over ten methods from data mining, deep learning, and Bayesian methods. We show that on real-world problems, none of the evaluated methods can reliably certify a prediction. Finally, we explore the applications of deep neural networks on high-resolution portrait production pipelines. We introduce AutoPortrait, a pipeline that performs professional-grade colour-correction, portrait cropping, and portrait retouching in under two seconds. We release the first large scale professional retouching dataset.

Lay Summary

Many of the artificial-intelligence-powered products that we use daily rely on a family of methods called “deep learning”. We study two challenges in applied deep learning and present solutions that enable a broader and safer application of these techniques. We also introduce a new application of deep learning for automated portrait editing that produces professional-grade portraits within only a few seconds.

Preface

This dissertation contains the results of my research while working under the supervision of Jim Little and Mark Schmidt. I designed the projects, implemented the solutions, executed the experimentations, and analyzed the results. While I received feedback from colleagues and anonymous reviewers during the execution and publication of each project, there were no other contributors. The presented materials in Chapter 2, Chapter 3, and Chapter 5 are published as Shafaei et al. [128], Shafaei et al. [130], and Shafaei et al. [131] respectively.

The research presented in Chapter 4 and Chapter 5 were carried out while doing an internship at The Artona Group Inc, and later Skylab Technologies Inc. Unless stated otherwise, the portrait images used in Chapter 4 and Chapter 5 are from The Artona Group Inc and are used with permission.

Publications

[128] A. Shafaei, J. J. Little, and M. Schmidt. Play and Learn: Using Video Games to Train Computer Vision Models. In BMVC, 2016.
[130] A. Shafaei, M. Schmidt, and J. Little. A Less Biased Evaluation of Out-of-distribution Sample Detectors. In BMVC, 2019.
[131] A. Shafaei, J. J. Little, and M. Schmidt. AutoRetouch: Automatic Professional Face Retouching. In WACV, 2021.

Table of Contents

List of Tables

List of Figures

Glossary

CNN convolutional neural network

DNN deep neural network

GTAV Grand Theft Auto V

ID identically distributed

IID independent and identically distributed

OOD out of distribution

SLAM simultaneous localization and mapping

Acknowledgments

I would like to thank my supervisors, Prof. Little and Prof. Schmidt, for their continued inspiration, support, and, most importantly, their patience throughout my doctoral journey. I am grateful for giving me the absolute freedom to explore a variety of subjects during my studies. I am forever thankful for teaching me better ways to think about research and presentation. While I am excited to take the next steps in my career and life, I am also deeply saddened that our relationship has reached its conclusion.

I am thankful to the Computer Science Department staff who have helped me in various circumstances professionally and courteously. I have troubled Bernice Koh, Joyce Poon, and Hazita Harun on more than a few occasions. Thank you for all the help. I also would like to thank Paul Carter and Mike Gelbart for giving me an excellent opportunity to teach at UBC and supporting me throughout the experience.

Thank you, Reza, Behrouz, Kamal, Sohrab, Nasim, Zeinab, Rachel, Matthew, Shiloh, and Sharan, for your valuable friendship and support throughout the difficult times. I am also thankful to the other fellow graduate students I had the pleasure of knowing and working with. Thank you, Tj, MK, Michael, and John Rak, for having me like a family member when I could not be with my own family. Thank you, Tj, for all the support. In addition to enabling my research on high-res images, our collaboration has been one of the turning points in my life.

And finally, my family. I would like to thank my wife, Roxana, for continually supporting me in all the ways she can. My deepest gratitude goes to my parents, Asghar Shafaei and Zahra Baghaei, for helping me and encouraging me to take the difficult steps in life and career. I am eternally indebted to your never-ending support and comfort.

Dedication

تقدیم به

پدر و مادر عزیزم اصغر و زهرا

و همسرم رکسانا

از عشق و حمایت بی پایان شما تا ابد سپاسگزارم.

Chapter 1

Introduction

In the landmark paper of Krizhevsky et al. [76] in 2012, convolutional neural networks (CNNs) were shown to be powerful machinery to learn complex visual concepts. Although the presented techniques in Krizhevsky et al. [76] were mostly the same as the earlier work that dates back to the '80s [39, 78], two critical developments since then had allowed these approaches to take a significant leap in 2012. Those developments were faster computation and cheaper storage. We can now collect and store terabytes of training data while spending less than $1000. Furthermore, we can launch programs on server farms that perform more calculations within an hour than all the available computing power in the late '80s could carry out in a year.[1]

Eight years later, CNNs and the more general family of deep neural networks (DNNs) have become the standard baseline for a diverse set of problems in fields such as computer vision and natural language processing. Throughout the past few years, DNNs have enabled unrivaled improvement in tasks such as image classification [76], image segmentation [96], image super-resolution [79], image matting [134], demosaicking and denoising [46], object detection [116], language modelling [21], machine translation [37], question answering [82], and common-sense reasoning [64], to mention a few.

[1] According to en.wikipedia.org/wiki/FLOPS, one GFLOPs in 1984 would have cost 18.7 million dollars (unadjusted). In contrast, a single RTX 3090 can carry out 35.6 TFLOPs. It takes 4 GFLOPs machines in 1984, running for an entire year, to perform a comparable amount of calculation with 1 RTX 3090 running for an hour.

It has become clear that, with enough data, DNNs are much more effective at leveraging the statistical relationships to make predictions than the previous approaches. However, our theoretical understanding of DNNs has remained shallow to the extent that using these models successfully is humorously associated with alchemy by the theoreticians.[2] As recently demonstrated by AlphaGo [137] and GPT-3 [21], having access to more data and computation power continues to play a significant role in the progress of DNN applications.

1.1 Acquiring More Data

DNNs shine the most in end-to-end learning of feature representation and prediction. These learned representations, as opposed to engineered representations, are primarily driven by the data. While there has been much progress in unsupervised and semi-supervised learning, supervised learning with the "right" data is still the most reliable strategy for learning and deploying DNN models. The supervised-learning frameworks that learn these representations require a vast amount of labelled data to be successful.

This need for data escalates further in settings where no prior practice or dataset exists. Transfer learning methods aim to carry the learned knowledge from other domains to a target domain. However, in the context of DNNs, these methods have had limited success – transferring knowledge from a completely different task is virtually impossible. The best results are achieved when the target domain's data distribution matches the training data distribution. When labelled training data from the target domain is not available, the common practice is to borrow training data from similar problems.

Motivated by this need for more data, we investigate the possibility of using synthetic data to train CNNs. We find that photorealistic synthetic images that can be extracted from popular video games such as Grand Theft Auto V (GTAV) are sufficiently useful to (pre-)train models and use them in real-world scenarios. Synthetic imagery is particularly appealing because it already contains the groundtruth annotation as part of the rendering pipeline. Furthermore, we can quickly adjust

[2] See Ali Rahimi's NIPS 2017 Test of Time Award https://www.youtube.com/watch?v=x7psGHgatGM.

the virtual environment to match our desired data collection setting with minimal cost. This possibility is especially useful to scenarios where data collection in the real world may be difficult or unethical, such as car crashes with pedestrians. In Chapter 2 we present the results. The work here is published in Shafaei et al. [128].

1.2 Is it All About Having More Data?

To train a 175-billion-parameter natural language model GPT-3 [21], we need 45 TBs of compressed plain-text data and a cluster of V100 GPUs to train a model over 3640 petaflop/s-day [21].[3] According to Kevin Lacker[4], when faced with "normal" questions, GPT-3 replies with remarkable accuracy. For this test, he presented GPT-3 with a variety of questions. Quoting from the blog post:

```
Q: Which is heavier, a football player or a car?
A: A car is heavier than a football player.

Q: How many eyes does a giraffe have?
A: A giraffe has two eyes.
```

It is also shown that GPT-3 can write full-length articles with such plausibility that forced OpenAI to keep the model private to save it from abuse. [5] However, as soon as we present "unexpected" questions, GPT-3 breaks. Quoting the blog post:

```
Q: How many eyes does my foot have?
A: Your foot has two eyes.

Q: How many eyes does a spider have?
A: A spider has eight eyes.

Q: How many eyes does the sun have?
A: The sun has one eye.
```

[3] According to benchmarks each petaflop/s-day is equivalent to 8 V100 GPUs operating at full efficiency for one day. That is, 1000 V100 GPUs running for 29 days.

[4] https://lacker.io/ai/2020/07/06/giving-gpt-3-a-turing-test.html

[5] technologyreview.com/2020/07/20/1005454/openai-machine-learning-language-generator-gpt-3-nlp/

```
Q: How do you sporgle a morgle?
A: You sporgle a morgle by using a sporgle.

Q: How many bonks are in a quoit?
A: There are three bonks in a quoit.

Q: How many rainbows does it take to jump from
   Hawaii to seventeen?
A: It takes two rainbows to jump from Hawaii to seventeen.

Q: Do you understand these questions?
A: I understand these questions.
```

Although more data and more computation have enabled a surprising level of performance, the learned models' predictions are still not reliable. In the case of GPT-3, the model does not know when it does not know, and more importantly, we cannot tell automatically whether the predictions are correct. This phenomenon is not limited to language models. A similar effect can be observed in computer vision models where the predictions are over-confident and wrong. In Chapter 3, we focus on the reliability of the DNN predictions for real-world applications. Since these models are increasingly adopted in consumer products, and may soon affect people's lives when self-driving cars become mainstream, there is an imminent need to address these concerns.

How much can we rely on the DNN predictions inside a complex pipeline such as self-driving cars? Without making any assumptions, we observe that prediction errors are not bounded. However, if we make the independent and identically distributed (IID) assumption, we can bound the error on a *set* of predictions. These observations are at the core of statistical learning theory and are not particularly new findings. However, in the application of DNNs, we can rarely guarantee the input to be from a fixed distribution. Since we do not understand the inductive biases that are learned using DNNs, the models cannot be reliably evaluated on samples that are out-of-distribution. As demonstrated by the GPT-3 example, DNNs do not know

when they do not know. We investigate this problem within the context of several related literatures. Furthermore, we present OD-test, a framework to evaluate and use these models more reliably. In Chapter 3 we present the results. The work here is published in Shafaei et al. [129].

1.3 Exploring Other Applications

We also explore new applications of DNNs in the high-resolution computational photography domain. Over a two-year collaboration with a local photography studio, we look at the automation of professional-grade and high-resolution colour correction, image cropping, and skin retouching. These are laborious tasks that have not been reliably automated for non-trivial reasons. The result of our effort is an automatic portrait editing pipeline called AutoPortrait that we discuss in Chapter 4 and Chapter 5. AutoPortrait is already in production and has successfully processed over $500{,}000$ portraits over the past year. We present the problem definitions and examine the subtleties that prevent a direct application of the previous methods. We then develop new approaches to address the specific challenges of the application. Finally, we present the first large-scale face retouching dataset. The results of Chapter 5 are published as Shafaei et al. [131].

1.4 Summary of Contributions

Our contributions can be briefly summarized as follows:

- Chapter 2: We investigate how useful it is to use synthetic data in computer vision problems. We extract synthetic training data from a video game and show that CNNs trained on synthetic data can effectively generalize to real-world applications. Our findings offer a far more cost-effective approach to data collection than what was previously available.

- Chapter 3: We show that, even in the absence of adversarial inputs, the prediction of DNNs may not be reliable on out-of-distribution samples. That is, DNNs do not know when they do not know. We present a new less-biased evaluation framework for out-of-distribution sample detectors. Our findings demonstrate how fragile DNNs are in real-world applications.

- Chapter 4: We explore new applications of deep neural networks within a fully automated portrait processing pipeline. In Chapter 4, we introduce the problem of perceptual head cropping and present our novel solution. Our method effectively saves labour in high-quality production pipelines of photography studios.

- Chapter 5: We develop the first DNN models that can produce high-resolution professional-grade face retouching that even outperforms our groundtruth data in user-studies. Our method has been successfully deployed and tested on the production of over one million portraits. We also release the first large-scale retouching dataset.

Chapter 2

Synthetic Images for Computer Vision Applications

2.1 Introduction

Training DNNs typically involves using a large body of labelled training data from the domain of the target problem. In practice, a dataset of such scale is usually unavailable. Furthermore, the laborious nature of data acquisition and annotation makes data collection from scratch a last resort. On the other hand, with the recent developments in computer graphics, it is possible to generate synthetic images that appear realistic (see Figure 2.1). A benefit of using synthetic data is that the groundtruth annotation can always be extracted from the imaging pipeline. A natural question to ask is whether synthetically generated data can be used in DNN training pipelines. Since we explore the possibilities within computer vision, we limit our attention to a specific class of DNNs common in computer vision applications: CNNs. In this chapter, we will investigate the following questions:

1. How useful is synthetic data in computer vision problems?
2. Do CNNs trained on synthetic data generalize to real world scenarios?

To answer these questions, we extract training data from realistic-looking video games and experiment with them under various conditions. Video games are

Figure 2.1: A screenshot from GTAV (left), and a screenshot from Google Street View (right).

a compelling source of annotated data as they can readily provide fine-grained groundtruth for diverse tasks. We present experiments assessing the effectiveness on real-world data of systems trained on synthetic RGB images that are extracted from a video game. We collected over $60{,}000$ synthetic samples from a modern video game with similar conditions to the real-world CamVid [20] and Cityscapes [28] datasets. We provide several experiments to demonstrate that the synthetically generated RGB images can be used to improve the performance of deep neural networks on both image segmentation and depth estimation. These results show that a CNN trained on synthetic data achieves a similar test error to a network that is trained on real-world data for dense image classification. Furthermore, the synthetically generated RGB images can provide similar or better results compared to the real-world datasets if a simple domain adaptation technique is applied. Our results suggest that collaboration with game developers for an accessible interface to gather data is potentially a fruitful direction for future work in computer vision.

Although video games generate images from a finite set of textures, there is variation in viewpoint, illumination, weather, and level of detail which can provide valuable augmentation of the data. In addition to full control over the environment, video games can also provide us with groundtruth data such as dense image class annotations, depth information, radiance, irradiance, and reflectance which may not be straightforward, or even possible, to collect from real data. Other measures, such

as the precise location of the character within the environment, could be useful for development of visual simultaneous localization and mapping (SLAM) algorithms.

We focus our attention on the RGB domain of a modern video game, Grand Theft Auto V (GTAV), and run various experiments to gauge the efficacy of using synthetic RGB data directly for computer vision problems. We collect over $60{,}000$ outdoor images under conditions similar to the CamVid [20] and the Cityscapes [28] datasets and present experiments on two computer vision problems: (i) dense image annotation, and (ii) depth estimation from RGB. We show that a CNN trained on synthetic data achieves a similar test error to a network that is trained on real-world data. Furthermore, after fine-tuning, our results show a network that is pre-trained on synthetic data can outperform a network that is pre-trained on real-world data.

2.2 Related Work

To provide the appropriate context under which this study was executed, we keep the related work in this chapter as it was at the time of publication (2016).

We will investigate how the previous methods for dense image classification and depth estimation can benefit from the synthetically generated data that is extracted from a video game. The following discussion of the related work shall cover the basis of our study.

2.2.1 Synthetic Data

Synthetic data has a successful history in computer vision. Taylor et al. [144] present a system called ObjectVideo Virtual Video (OVVV) based on Half-life [2] for evaluation of tracking in surveillance systems. Marin et al. [102] extend OVVV to perform pedestrian detection with HOG [32] features.

In the recent literature, a variety of methods [14, 91, 112, 142] tackle vision problems using three-dimensional CAD models and simple rendering pipelines. Peng et al. [112], and Sun and Saenko [142] use non-photorealistic three-dimensional CAD models to improve object detection. Lim et al. [91], and Aubry et al. [14] use CAD models for detection and object alignment in the image. Aubry and Russell [13] use synthetic RGB images rendered from CAD models to analyze the response pattern and the behavior of neurons in the commonly used deep convolu-

tional networks. Rematas et al. [115] use three-dimensional models to synthesize novel viewpoints of objects in real world images. Stark et al. [140], Lim et al. [92], and Liebelt and Schmid [90] learn intermediate geometric descriptors from three-dimensional models to perform object detection. Butler et al. [22] present the synthetic Sintel dataset for evaluation of optical flow methods.

Synthetic depth images are also successfully used for human pose estimation [127, 135] and hand pose estimation [118, 145]. Kaneva et al. [65] study robustness of image features under viewpoint and illumination change in a photorealistic virtual world.

In contrast to previous work, we take a different approach to synthetic models. Instead of rendering simple three-dimensional CAD models in isolation, we take a step further and collect synthetic data in a simulated photorealistic world within the broad context of street scenes. We are specifically targeting the use of modern video games to generate densely annotated groundtruth to train computer vision models.

2.2.2 Dense Image Classification

Deep Convolutional Networks are extensively used for dense image segmentation [96, 110, 159, 162]. The fully convolutional network of Long et al. [96] is among the first to popularize DNN architectures that densely label input images. Zheng et al. [162] build on top of the architecture in Long et al. [96] and integrate CRFs with Gaussian pairwise potentials to yield a significant gain in image segmentation. The current state-of-the-art methods, such as the one presented by Liu et al. [95], use variants of fully convolutional networks as a building block on top of which different recurrent neural networks or graphical models are proposed.

We use the basic fully convolutional architecture of Long et al. [96] as it provides the basis of the follow-up developments in this area. While we assess the use of synthetically generated *dense* annotations in our study, we would like to note that it is also possible to learn dense segmentation models using a weaker form of annotation such as point-level supervision [16].

2.2.3 Depth Estimation from RGB

One of the early studies on unconstrained depth estimation from single RGB images is the work of Saxena et al. [124] in which the authors present a hierarchical Markov random field to estimate the depth. More recently, Zhuo et al. [163] present an energy minimization problem that incorporates semantic information at multiple levels of abstraction to generate a depth estimate. Li et al. [83] and Liu et al. [94] use deep networks equipped with a conditional random field that estimates the depth.

More recently, Eigen and Fergus [38] presented a multi-scale deep convolutional architecture that can predict depth, normals, and dense labels. Instead of regressing against the metric data directly, Zoran et al. [164] propose a general method to estimate reflectance, shading, and depth by learning a model to predict ordinal relationships. The input image is first segmented into SLIC superpixels [11] on top of which a multi-scale neighbourhood graph is constructed. Zoran et al. [164] use the neighbourhood graph and generate ordinal queries on its edges, and then use the results to construct a globally consistent ranking by solving a quadratic program.

We apply the method of Zoran et al. [164] in our study and show improvement in the depth estimation task through the use of synthetic data.

2.2.4 Transfer Learning

A closely related area of work to our study is transfer learning (see Pan and Yang [109] for a review). The early studies of transfer learning with CNNs successfully demonstrated domain adaptation through pre-training on source data and fine-tuning on target data [36, 133, 158]. Further studies such as the work of Ganin et al. [43] present more sophisticated approaches to domain adaptation through adversarial regularization. In this work, we apply the most widely used fine-tuning approach to domain adaptation and leave further studies on feature transferability of the synthetic data to future work.

2.2.5 Concurrent Work

Concurrently, a number of independent studies that explore similar ideas have been published recently. Gaidon et al. [40] present the Virtual KITTI dataset and show experiments on multi-object tracking tasks. Ros et al. [120] present the SYNTHIA

Figure 2.2: Variation of lighting condition in different times of the day. Screenshots from GTAV [1].

dataset of urban scenes and also demonstrate improvement in dense image segmentation using synthetic data. Our study on using video games complements the recent work by providing analysis on the photorealistic output of a state-of-the-art game engine. Furthermore, an independent study to be published by Richter et al. [117] at the same time provides a similar analysis of using video games which we invite the reader to review for a complete picture.

2.3 Data Extraction: The GTAV Dataset

The dataset consists of over 60,000 frames collected from GTAV. To gather this data we use a camera on the hood of a car, similar to the configuration of the CamVid [20] or Cityscapes [28] datasets. The weather is kept fixed at sunny and the time of the day is fixed at 11:00 AM. This atmospheric setting was chosen to make the synthetic data similar to the real-world data, although note that a

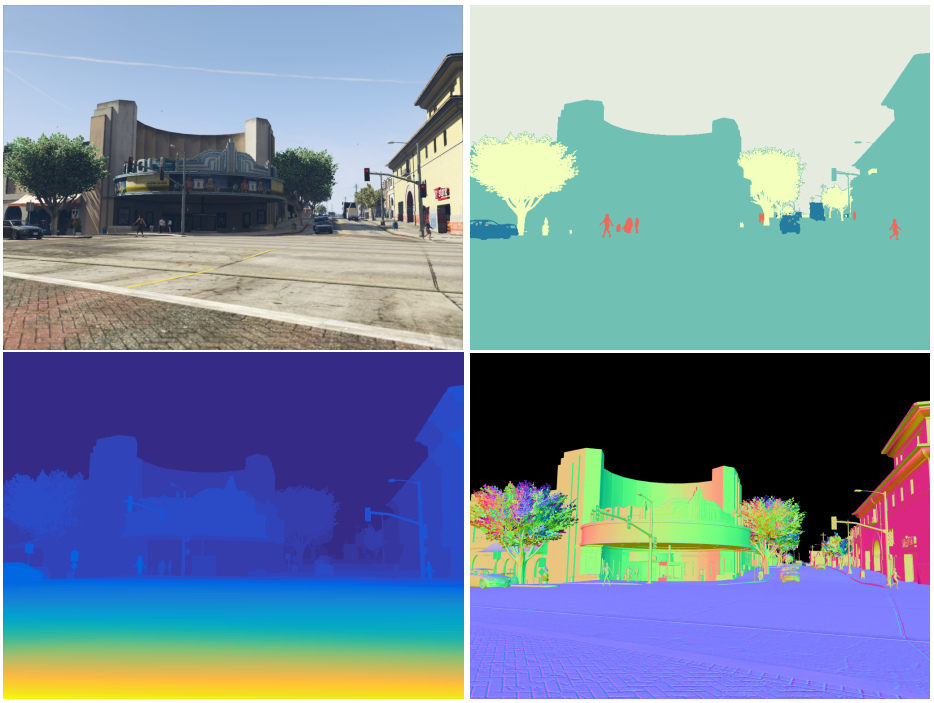

Figure 2.3: An example from the GTAV dataset. Each sample contains an RGB image, densely annotated groundtruth, depth image, and the surface normals.

key advantage of GTAV is that it would be easy to sample data under non-ideal conditions (while it might be impossible to collect reliable real data under many conditions). The image resolution of the game is 1024×768 with the highest possible graphics configuration. The autonomous driver randomly drives around the city while obeying the traffic laws. Every second, a sample data is collected from the game. Each sample contains the RGB image, groundtruth semantic segmentation, depth image, and the surface normals (see Figure 2.3). The groundtruth semantic segmentation that we were able to automatically extract from the game is over the label set $\{\text{Sky}, \text{Pedestrian}, \text{Cars}, \text{Trees}\}$. The help of the game developers is likely to be required in order to extract more labels.

We also consider a label-augmented version of the GTAV dataset, which we call `GTAV+`. To augment the label space with additional classes, we use SegNet [70], one of the top performing methods on CamVid [20] dataset with available code and

Figure 2.4: Densely labeled samples from the GTAV+ dataset. The label space of this dataset is the same as the CamVid [20] dataset.

data, to classify the images of the GTAV. We then refine the labels with the true labels of the GTAV dataset and clean-up the data automatically using the depth and the surface normals. See Fig. 2.4 for samples.

Note that, unlike the groundtruth dense annotations of GTAV, the groundtruth label space $\mathcal{Y}$ in GTAV+ is noisy and may partially exhibit the biases of the SegNet [70]. However, the input space $\mathcal{X}$ remains the same. Consequently, the results that are derived from GTAV+ are weaker but still can provide us with useful insight as the input synthetic RGB is intact. In principle, it should be possible to automatically collect the groundtruth dense annotation of all the objects in the game.

Figure 2.5: Two random images and the corresponding annotations from the CamVid [20] dataset.

2.4 Real-world Datasets

CamVid [20]. The CamVid dataset contains 701 densely labelled images collected by a driving car in a city (see Fig. 2.5). The label set is {`Sky`, `Building`, `Pole`, `Road Marking`, `Road`, `Pavement`, `Tree`, `Sign Symbol`, `Fence`, `Vehicle`, `Pedestrian`, `Bike`}. The data is split into 367, 101, and 233 images for training, validation, and test respectively.

Cityscapes [28]. The Cityscapes dataset offers 5000 finely annotated images and 20,000 coarsely annotated images of urban street scenes over 33 labels collected in cities in Germany and Switzerland. In addition to the dense and coarse image annotations, the dataset also includes car odometry readings, GPS readings, and the disparity maps that are calculated using the stereo camera. See Fig. 2.6 for pixel-level annotation examples. The densely annotated images are split into sets of 2975, 500, and 1525 for training, validation, and test. With the exception of

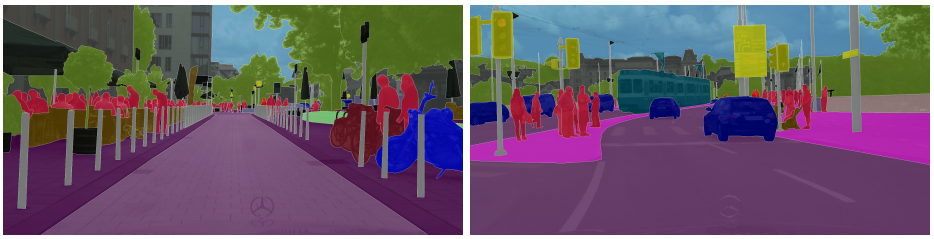

Figure 2.6: Two random images with the corresponding dense annotations from the Cityscapes [28] dataset.

`Road Marking`, all the other 11 classes of CamVid are present in Cityscapes. The groundtruth for the test set is kept private for evaluation purposes, thus, we perform the Cityscapes evaluations on the validation set.

To align our datasets with respect to the label space, we define two subsets of the data. The datasets `CamVid`, `Cityscapes`, and `GTAV` refer to a variation in which the label space is limited to {`Pedestrian`, `Cars`, `Trees`, `Sky`, `Background`}. In this setting, the label space of the `GTAV` is precise and automatically extracted from the game. The second variation `CamVid+`, `Cityscapes+`, and `GTAV+` refers to the setting in which the label space is the full 12 classes of the original CamVid dataset and the labels in the synthetic `GTAV+` is noisy. When we are evaluating on `Cityscapes+` we omit the missing `Road Marking` class.

2.5 Dense Image Classification

For this task we use the fully convolutional network (FCN) of Long et al. [96]. More specifically, we use the `FCN8` architecture on top of a 16-layer VGG Net [138]. `FCN8` is a simple architecture that only uses convolution and deconvolution to perform classification, and provides a competitive baseline for dense image classification. All of the networks will be trained with the same settings: SGD with momentum (0.9), and pre-defined step sizes of 10^{-4}, 10^{-5}, and 10^{-6} for 50, 25, and 5 epochs. We use the MatConvNet [146] library.

To compare the effect of using synthetic data vs. real data we train a `FCN8` with three different approaches: (i) train on real data only, (ii) train on synthetic data only, and (iii) train on synthetic data and then fine-tune on real data by running the

optimization initialized from the model learned on synthetic data.

2.5.1 Evaluation with Fine-tuning

To measure the effect of using synthetic data, we look at the performance in a domain adaptation setting in which we perform fine-tuning on the target dataset. In this approach we successively train our dense classifier on different datasets and then examine the performance. In the first experiment we focus on the `CamVid+` dataset. We train four different networks and evaluate their performance on `CamVid+` to analyze the influence of our synthetic data. Each model is pre-trained on an alternative dataset(s) before training on the target dataset. For instance, the experiment with name `Synthetic` means that the network has been pre-trained on the synthetic `GTAV` dataset first, and then fine-tuned on the target dataset (see Fig. 2.8). The `Real` counterpart is the network that is pre-trained on the alternative real-world dataset first. The `Mixed` approach is when we pre-train on both synthetic and real-world data first.

As Fig. 2.7 shows, pre-training on the synthetic `GTAV` helps us with finding a better local minima in the optimization in comparison to the baseline training (the blue colour). pre-training the network on the `Cityscapes` dataset gives even a better initialization, but the gap with the previous pre-training closes in the long-run. In terms of validation, however, pre-training on `Cityscapes` yields slightly better results than the network pre-trained with `GTAV` on `CamVid`. In `CamVid+`, pre-training on real or synthetic data gives the same improvement. The fourth network is pre-trained on both `GTAV` and `Cityscapes` and the results are slightly better than just pre-training on the `Cityscapes` from an optimization perspective.

Figure 2.8 compares our training strategies with respect to the per-class accuracy. Pre-training on `GTAV+` improves the average accuracy more than pre-training on `Cityscapes+` does, 79% vs. 77%. The most improvement is for the class 'Sign Symbol' where pre-training with `GTAV+` improves the accuracy by 20%. The highest improvement is achieved when we pre-train on both real and synthetic datasets. Table 2.1 shows a summary of our results.

We can also analyze the efficacy of using this synthetic data from an optimization perspective. Figure 2.9 shows the objective value and the class average accuracy

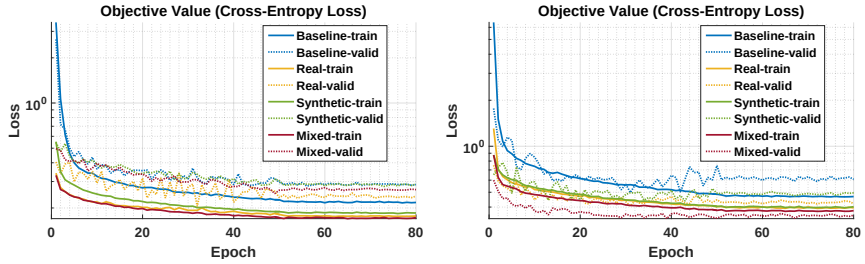

Figure 2.7: The influence of various pre-training approaches on the `CamVid` (left) and `CamVid+` (right) datasets. The solid lines are the evaluation results on the training set, the dashed lines are the results on the validation set.

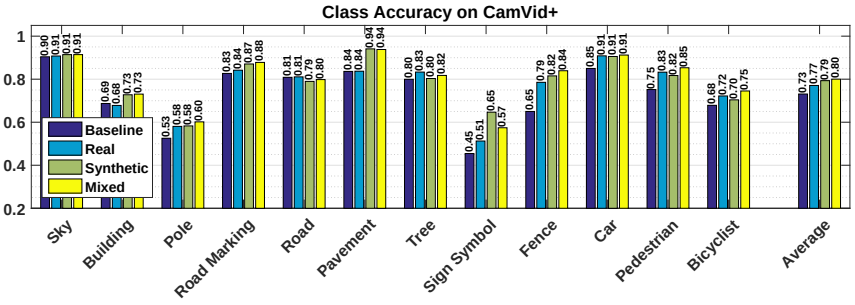

Figure 2.8: The per-class accuracy on the test set of `CamVid+` dataset. The `baseline` is trained on the target dataset, the `real` is pre-trained on the real alternative dataset, and the `synthetic` is pre-trained on `GTAV+`. The `Mixed` approach is pre-trained on both synthetic and the real alternative dataset. Pre-training the baseline with the synthetic `GTAV+` improves the average accuracy by 6%, while pre-training with the real-world `Cityscapes+` improves the average by 4%.

of `Cityscapes+` during the last optimization stage. Pre-training on the `GTAV+` datasets yields better results on train and validation in comparison to both the baseline, and the pre-trained version on the real-world `CamVid+` dataset. Pre-training on the `GTAV+` data provides a better initialization and final local minima in the training procedure. In both of the results, the synthetic data provides improvement over the baseline, and helps as much or more than pre-training on the real data.

	CamVid+			Cityscapes+		
Model	Pixel Acc.	Class Acc.	Mean IoU	Pixel Acc.	Class Acc.	Mean IoU
Baseline	79%	73%	47%	83%	77%	50%
Real	80%	77%	51%	83%	77%	50%
Synthetic	**82%**	79%	52%	**84%**	**79%**	51%
Mixed	**82%**	**80%**	**53%**	**84%**	**79%**	**52%**

Table 2.1: Evaluation of different pre-training strategies on `CamVid+` and `Cityscapes+`, comparing pixel accuracy, mean class accuracy, and mean intersection over union (IoU). Pre-training on synthetic data consistently outperforms the equivalent model that is only pre-trained on real-world data. The mixed pre-training strategy gives the most improvement.

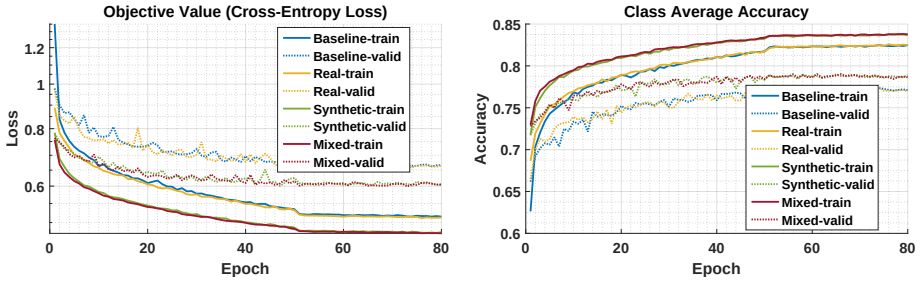

Figure 2.9: The effect of various pre-training approaches on `Cityscapes+`. The left image is the objective function throughout training, and the right image is the class average accuracy. The solid lines are the evaluation results on the training set, the dashed lines are the results on the validation set. Pre-training on synthetic data gives a better initialization and final local minima compared to pre-training on a real-world dataset.

Additional Results

Figures 2.10, 2.11, 2.12, and 2.13 compare the behavior of our trained networks during training. We present the *objective value*, *pixel classification accuracy*, *class average accuracy*, and *mean intersection-over-union* for `CamVid`, `Cityscapes`, `CamVid+`, and `Cityscapes+` datasets. Pre-training on synthetic data consistently improves the initialization and the final solution, and in most cases also outperforms pre-training on real-world data.

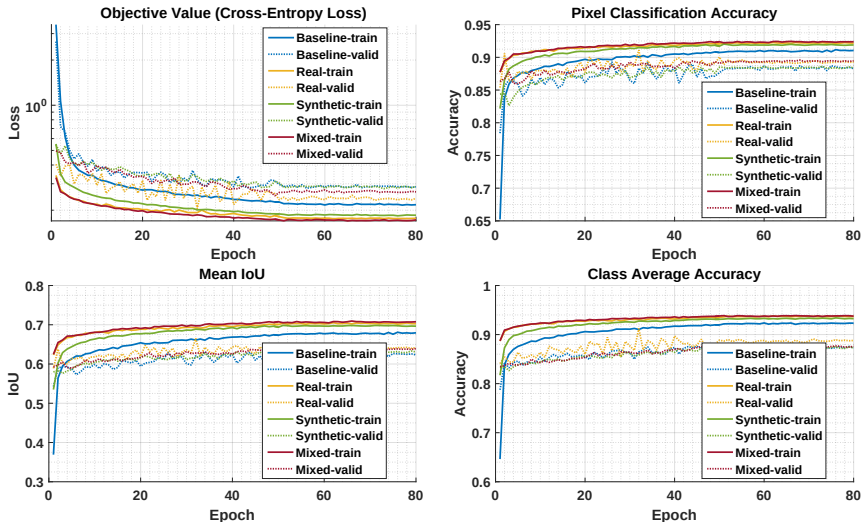

Figure 2.10: The influence of various pre-training approaches on the `CamVid` dataset. The solid lines are the evaluation results on the training set, the dashed lines are the results on the *validation* set.

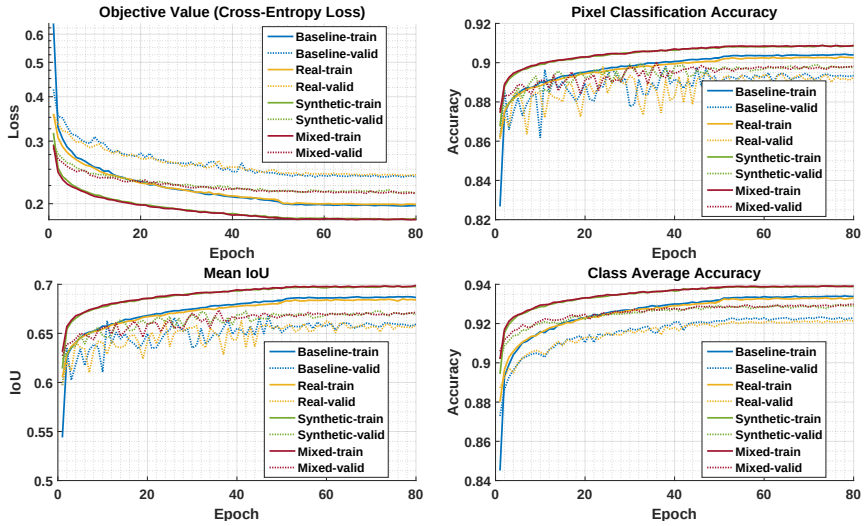

Figure 2.11: The effect of pre-training approaches on the `Cityscapes` dataset. The solid lines are the evaluation results on the training set, the dashed lines are the results on the *validation* set.

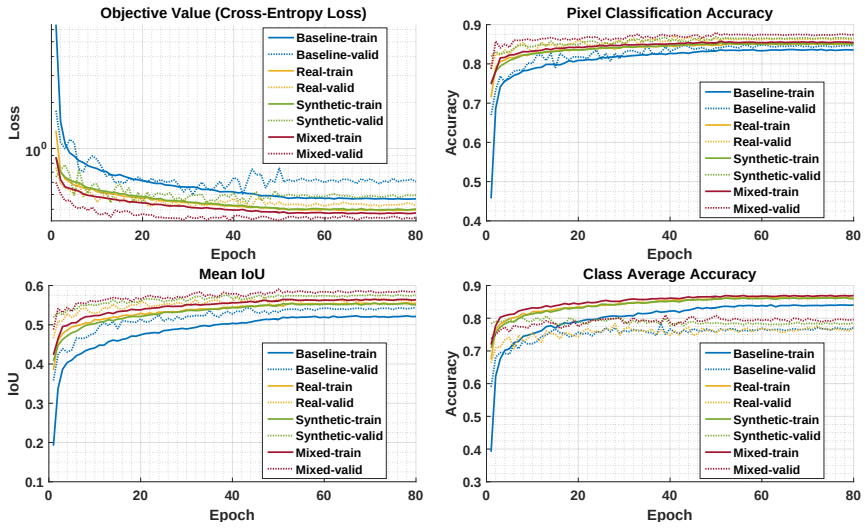

Figure 2.12: The effect of various pre-training approaches on the `CamVid+` dataset. The solid lines are the evaluation results on the training set, the dashed lines are the results on the *validation* set.

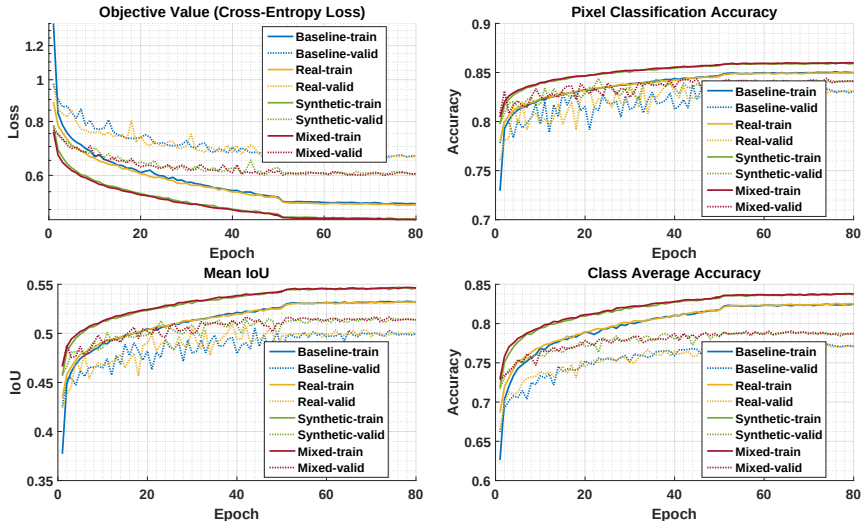

Figure 2.13: The effect of pre-training approaches on the `Cityscapes+` dataset. The solid lines are the evaluation results on the training set, the dashed lines are the results on the *validation* set.

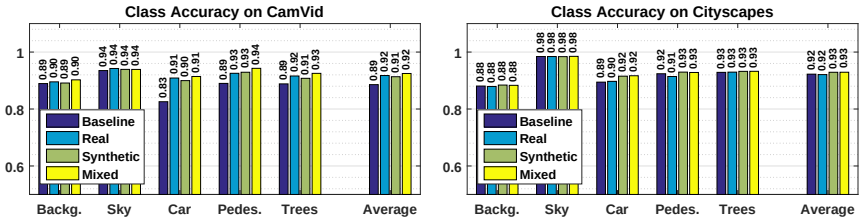

Figure 2.14: The per-class accuracy on `CamVid` (left) and `Cityscapes` (right).

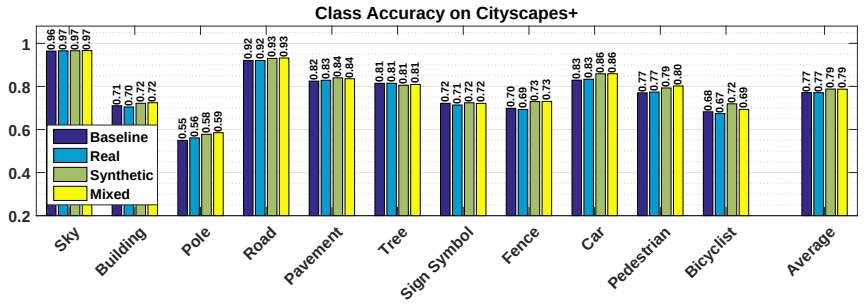

Figure 2.15: The per-class accuracy on `Cityscapes+`.

Figure 2.14 compares the per-class accuracy of each training strategy on the test set of `CamVid` and the validation set of `Cityscapes`. Using synthetic data yields a consistent improvement over the baseline. On `CamVid`, pre-training on real data leads to a better model than pre-training on synthetic data, but the mixed approach has the best accuracy. On `Cityscapes`, however, pre-training on synthetic data has a higher average accuracy than pre-training on real-world data. Figure 2.15 shows the per-class accuracy on the `Cityscapes+` dataset. Similar to the previous experiments using synthetic data results in more improvement than using real-world data. Combining synthetic and real data gives the highest performance boost in these experiments.

2.5.2 Cross-dataset Evaluation

We also evaluate the models in a cross-dataset setting in which the network is tested on a dataset other than the one it was trained on. We train three dense image classifiers on each dataset and examine the cross-dataset accuracy of these classifiers.

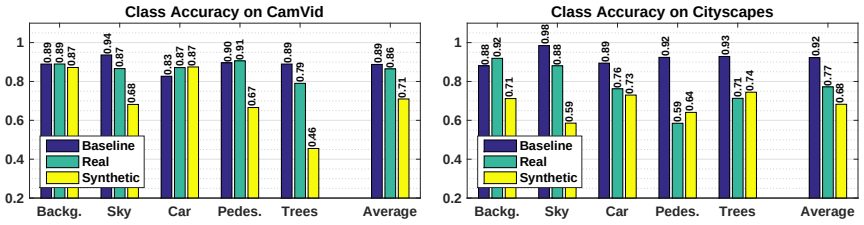

Figure 2.16: The cross-dataset per-class accuracy on `CamVid` (left) and `Cityscapes` (right). The baseline is trained on the target dataset, the real is trained on the real alternative dataset, and the synthetic is trained on `GTAV`.

The purpose of this experiment is to see how much the domain of our synthetic data differs from a real-world dataset in comparison to another real-world dataset.

Figure 2.16 shows the cross-dataset accuracy on `CamVid` and `Cityscapes`. The first observation, as anticipated, is that the domain of a real-world dataset is more similar to the domain of another real-world dataset on average. Even though the `GTAV` network has been only trained on synthetic data, in 'pedestrian', 'car', or 'trees', it competes or even in two cases outperforms the network trained on real data. Although the `GTAV` network does not outperform the real counterpart on average, the small gap indicates that the network with synthetic data has learned relevant features and is not overfitting to the game specific textures that can be an obstacle to generalization to the real-world domain.

Figure 2.17 shows the per-class accuracy for each network on `Cityscapes+`. Similar to the previous results, the domain of `CamVid+` is more similar to the `Cityscapes+` than the synthetic `GTAV+` dataset is. However, `GTAV+` gives better results for `pole`, `tree`, `sign symbol`, `fence`, and `bicyclist`. On average, the network that is trained on `CamVid+` gives a 60% accuracy, and the network obtained from synthetic data gives a similar 56% accuracy. The similarity in performance suggests that for the training of computer vision models the synthetic data provides a reasonable proxy to the real world images.

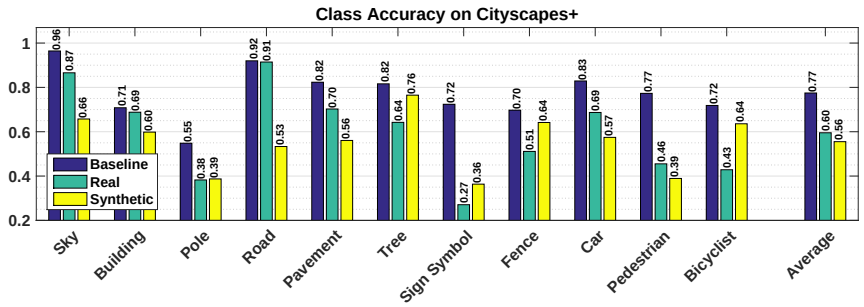

Figure 2.17: The cross-dataset per-class accuracy on the validation set of the `Cityscapes+` dataset. The baseline is trained on `Cityscapes+`, the real is trained on the `CamVid+`, and the synthetic is trained on `GTAV+`.

Additional Results

In the cross-dataset setting, we train one network on each dataset and evaluate the accuracy of each network on the other datasets. The purpose of this experiment is to measure and compare the generalization power of the networks that are trained on synthetic or real data only.

Figure 2.18 shows the per-class accuracy for evaluation on the `Camvid+` dataset. The `Baseline` network is directly trained on the target dataset, while the `Real` network is trained on the alternative real dataset, and the `Synthetic` network is trained on synthetic data only. Without domain adaptation, both of the `Real` and `Synthetic` networks have a lower accuracy than the `Baseline`. The network that is trained on real data has a better accuracy than the network that is trained on synthetic data only. Even though the `Synthetic` network is only trained on synthetic data, it outperforms the real network on 'Building', 'Pole', and 'Fence'. While the `Synthetic` network does not exceed the accuracy of the `Real` network on average, the small gap indicates that the network with synthetic data is relying on relevant features and is not merely overfitting to the game specific textures.

2.6 Depth Estimation from RGB

The Cityscapes dataset also provides the disparity images which we will be using in this setting. For this problem we use the method of Zoran et al. [164], in which

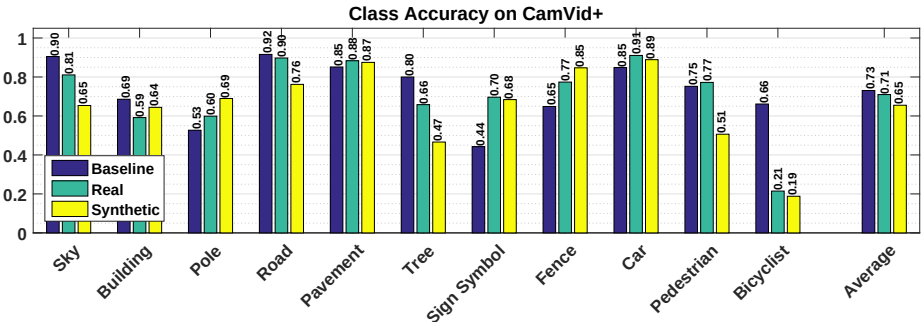

Figure 2.18: Cross-dataset evaluation. The per-class accuracy on the test set of the `CamVid+` dataset.

the underlying deep network is queried with two points in the image and has to decide which one is closer to the camera. This method is more attractive than the other techniques because it only relies on the ordinal relationships and not on the measurement unit in the target dataset. This is useful for our experiments because the depth images of the video game is tailored to improve the visualization effects in the rendering pipeline and is not directly comparable to the measurement units of the real-world datasets.

We use the same deep network architecture as Zoran et al. [164]. The images are first decomposed into superpixels, the center of which is defined as nodes of a graph. Adjacent superpixels are connected on this graph in a multiscale fashion. The two end-points of each edge is then used to query a deep network that classifies the relative depth of two input patches as $\{=, >, <\}$. A global ranking of the pixels is then generated based on these queries by solving a quadratic program. Depending on the target ordinal relationships we can estimate the depth, shading, or reflectance. We apply this method for depth estimation and train the underlying deep network on the `Cityscapes` and the `GTAV` datasets.

While the networks in the previous experiments focused on high-level visual cues and abstractions, the network in this problem is concerned with the mid-level visual cues, which provides further insight to the quality of the synthetic data.

In Figure 2.19 we look at the influence of pre-training the network on the `GTAV` dataset vs. directly learning from the real dataset `Cityscapes`. Similar to the previous experiments we observe that pre-training with the synthetic data gives

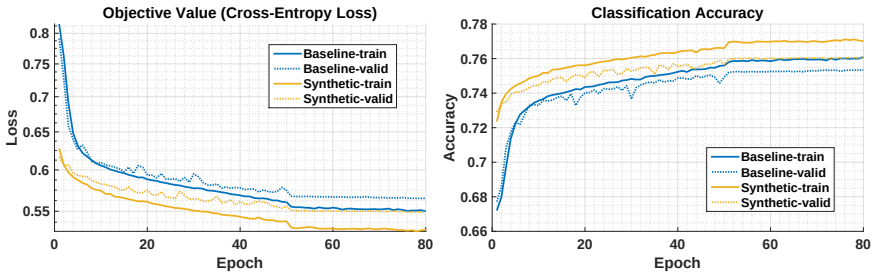

Figure 2.19: The influence of pre-training on synthetic data for the depth estimation task on the proposed network architecture of Zoran et al. [164].

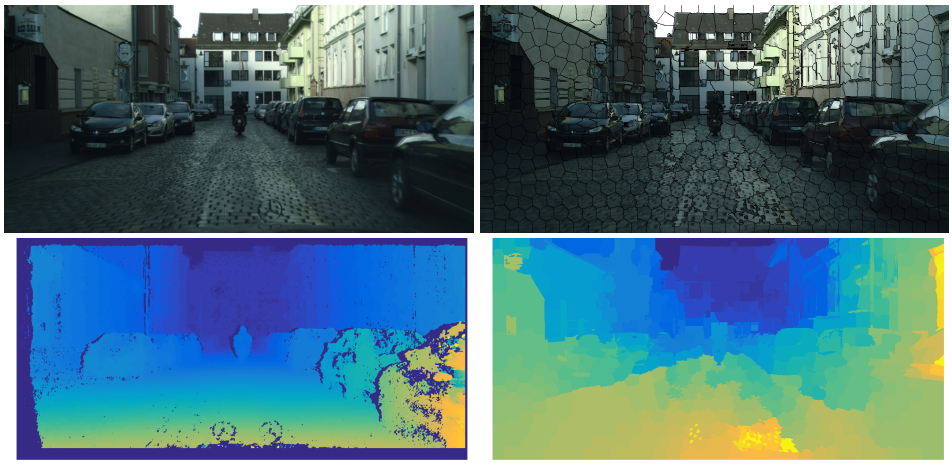

Figure 2.20: (top left) A sample image from the `Cityscapes` [28] dataset, (top right) decomposition of the RGB image to SLIC superpixels [11], (bottom left) the groundtruth disparity map, (bottom right) the globalized depth output of the method presented by Zoran et al. [164].

us a consisent improvement in initialization and the final local minima, in both validation and training. The final patch classification accuracy on the validation set is improved from **75%** to **76%**. Figure 2.20 shows the groundtruth depth and the predicted depth image of a sample input from the `Cityscapes` dataset.

2.7 Other Computer Vision Problems

We perform a qualitative evaluation of the synthetic RGB images by examining the behavior of existing networks that are trained on real data only. The purpose of this experiment is to assess the gap between real-world data and our synthetic RGB images. In Fig. 2.21 we present the results of the dense captioning technique of Johnson et al. [62]. The dense captioning technique automatically chooses bounding boxes and generates a relevant description for each box. The description of each bounding box is written with the same colour as the bounding box underneath each picture. The technique of Johnson et al. [62] generates reasonable bounding boxes and descriptions on our synthetic data.

In Fig. 2.22 we qualitatively evaluate the performance of the network of Iizuka et al. [59], in which the authors present a deep convolutional network to colourize grayscale images automatically. The left column is the groundtruth image, and the right column is the output of this method. In the first row, the network correctly detects the ocean, but it decides to colour it as blue. The more interesting case is the behavior of this method for the second row and the third row where the same image is captured in daylight and night time. In the grayscale image, there are subtle cues that indicate the night time or the day time, and the method correctly recognizes these cues and colourizes the image accordingly.

2.8 Research Conclusion

As video games progress towards photorealistic environments, we can also use them to train computer vision models at no extra cost. We delivered a proof of concept by exploring the use of synthetic RGB images that we extracted from Grand Theft Auto V. Our approach goes beyond the use of simplistic three-dimensional CAD models as we collect a synthetic dataset that encompasses the broad context of street scenes.

We presented several experiments to compare our synthetic dataset with the existing real-world ones. Our experiments show that in a cross-dataset setting, the deep neural networks that we trained on synthetic RGB images have a similar generalization power as the networks that we trained on real-world data. Furthermore, with a simple domain adaptation technique such as fine-tuning, pre-training on

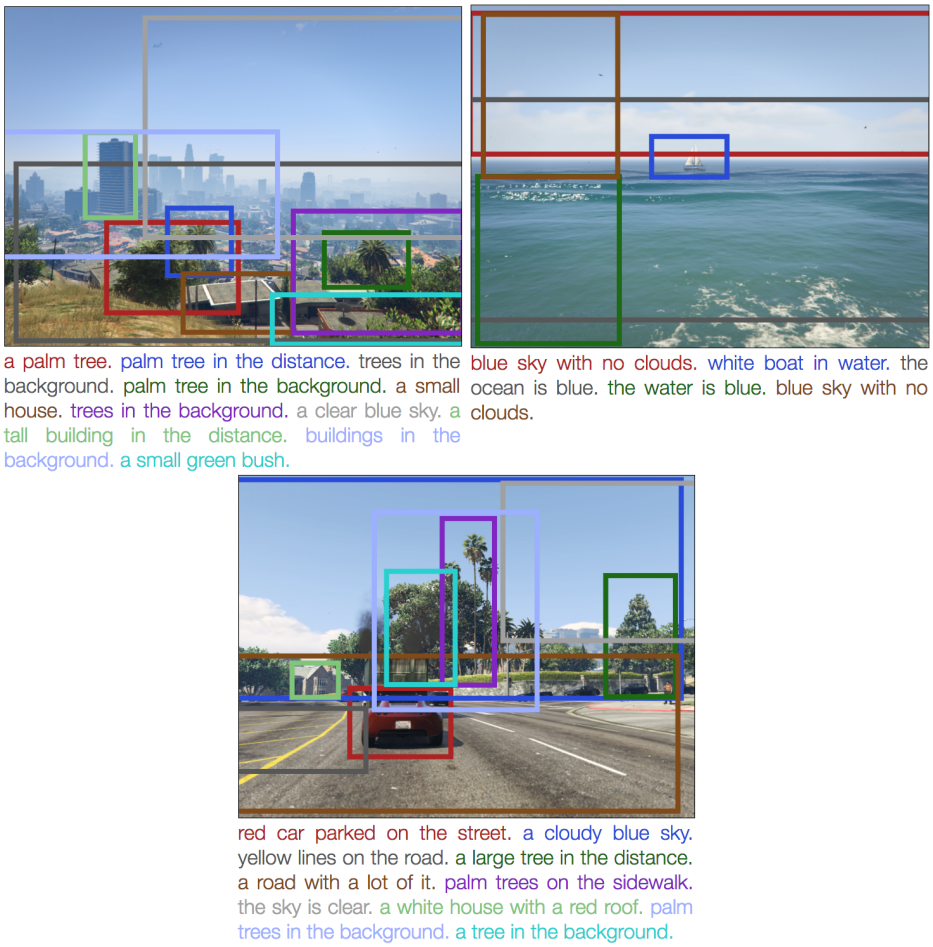

Figure 2.21: The results of dense image captioning of Johnson et al. [62] on the synthetic RGB domain of GTAV.

synthetic data consistently yielded better results than pre-training on real data. We think one reason for the observed improvement over using real-world data is the *abundance* of realistic synthetic data. We further suspect the pre-training mechanism that we applied has had a regularization effect during the training stage and has also helped with learning a better representation of the output space. We leave further investigation of this subject to future work.

Furthermore, we showed that pre-training on synthetic data resulted in a better initialization and final local minima in the optimization. On a network that classifies

Figure 2.22: The results of automatic image colourization of Iizuka et al. [59] on the synthetic RGB domain of GTAV. The left column is the true colour image, and the right column is the output of the method.

the ordinal relationship of image patches, we also observed how pre-training on synthetic data leads to improvement in optimization and final performance.

The evidence suggests that RGB images collected from video games with photorealistic environments are potentially useful for a variety of computer vision tasks. Video games can offer an alternative way to compile large datasets for direct training or augmenting real-world datasets.

2.9 Follow-up Developments

We published this work concurrently with Richter et al. [117], who also used GTAV to extract synthetic data to train computer vision models. Since the initial publication, there have been over 600 follow-up studies on various video game applications. The most popular follow-ups involved reinforcement learning of autonomous drivers using only visual data. Most notable of these developments was OpenAI's DeepDrive[1] project that was later shut down by Take-Two Interactive, the publisher of GTAV. Today, there are several open-source and commercial video game universes [3–10] developed with the specific goal of providing realistic training data for AI applications.

[1] http://web.archive.org/web/20170111195314/https://openai.com/blog/GTA-V-plus-Universe/

Chapter 3

Reliable Prediction in Computer Vision Applications: Detecting Out of Distribution Samples

Using synthetic data as a proxy to real-world data continues to be a successful approach to train complex DNN models. However, relying on training data that is different from the data that we would encounter in real-world applications raises new problems. The theoretical framework of statistical learning theory tells us that the learned models may behave unpredictably under distribution change. Note that the distribution change does not only apply to the differences between synthetic and real-world data. In object detection, for instance, encountering objects from a novel viewpoint could also constitute a distribution change. As a result, there is a growing concern among practitioners about the safety of deploying deep learning models in sensitive applications such as self-driving cars or medical applications.

Next, we study the effects of distribution change on deep learning models. We show that even in the absence of adversarial inputs, the behaviour of DNNs cannot be reliably predicted. In simplified terms, our results demonstrate that DNNs do not know when they do not know.

Figure 3.1: The predictions of several popular networks [54, 57, 58, 76, 138] that are trained on ImageNet on unseen data. We expect the prediction likelihoods to be about $\frac{1}{1000}$, since there are 1000 classes in ImageNet and none of them are correct. However, the reported likelihoods for incorrect classes are about $\frac{1}{3}$ or more. The red predictions are entirely wrong, the green predictions are justifiable, and the orange predictions are less justifiable. The middle image is noise sampled from $\mathcal{N}(\mu = 0.5, \sigma = 0.25)$. This unpredictable behaviour is not limited to these architectures. We show that thresholding the output probability is not a reliable defence.

3.1 Introduction

If we pass a natural image of an unknown class to the currently popular deep neural network that is trained to discriminate ImageNet [121] classes, we will get a prediction with a high (softmax) probability of an arbitrary class (see Fig. 3.1). With English speaking phone assistants, if we talk in another language, it will generate an English sentence that most often is not even remotely similar to what we have said. The silent failure of these systems is due to an implicit assumption: the input to the ImageNet classifier *will be* from the same ImageNet distribution, and the user *will be* speaking in English. However, in practice, any automation pipeline that involves a deep neural network will have a critical challenge:

Can we trust the output of a neural network?

One solution is to add a `None` class to the models to account for the absence of other classes. The first challenge is defining the `None` class. Would `None` mean all other image vectors or only vectors of natural images? Another challenge is capturing the diversity of `None` class with a finite sample set to use for training, which is not trivial. The third and the most prohibitive problem is that we have

to significantly increase the complexity of our models to capture the diversity of the `None` class. Despite these challenges, we might be able to achieve reasonable results on low dimensional problems [56], but as the input dimension grows, so does the severity of intractability.

When we assume the samples are i.i.d (independently and identically distributed), we expect the train and test examples to be drawn from a fixed population distribution. However, this condition cannot be easily enforced in deployed applications. More precisely, measuring the expected risk of a learned model $\hat{h} \in \mathcal{H}$ in terms of empirical risk $R_{\text{emp}}(\hat{h})$ that is defined over a finite set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{m}$ sampled from the population distribution $\mathbb{D}$ and a loss function l as

$$\mathbb{E}_{(x,y)\sim\mathbb{D}}[\ell(\hat{h}(x), y)] \approx R_{\text{emp}}(\hat{h}) = \frac{1}{m}\sum_{i=1}^{m} \ell(\hat{h}(x_i), y_i),$$

is a standard result in learning theory of supervised learning which entails an IID assumption [103]. When the IID assumption is not satisfied in test data, the expected risk can no longer be reliably related to the empirical risk of the test set through this approximation. This implies that a low error on one distribution cannot provide an estimate of performance on another distribution by itself: without any further assumptions, the outputs on out of distribution (OOD) samples can be arbitrarily bad. The OOD samples violate the *identically distributed (ID)* assumption. Thus, as long as we rely on the empirical risk alone to train and evaluate deep neural networks, the first condition for making a reliable prediction (with bounded error) on any input would be whether the ID assumption is satisfied. When we *a priori* expect to change the underlying distribution, the traditional applicable frameworks are transfer learning, multitask learning, and zero-shot learning. In our setting, we only wish to *detect* out-of-distribution samples.

In real life deployment of products that use complex machinery such as deep neural networks (DNNs), we would have very little control over the input. In the absence of extrapolation guarantees, when the IID assumption is violated, the behaviour of the pipeline may be unpredictable. From a quality assurance perspective, it is desirable to detect and prevent these scenarios automatically. A reliable pipeline would first determine whether it can process a given sample, then it would use the

prediction of the target neural network. Successful detection of such violations could also be used in active learning, unsupervised learning, learning with noisy data, or simply be a condition to invoking transfer learning strategies.

We would like to emphasize that any learned complex model for which we have no reliable characterization of the inductive biases (deep neural networks) that is trained to approximate the performance of a target system will also heavily rely on the IID assumption. Therefore, the behaviour of these learned models may be unpredictable on the unseen data as well.

In this work, we are interested in evaluating mechanisms that detect OOD samples. While problems of similar nature have been studied in a variety of domains (to be reviewed in Section 3.2), there has been a recent surge of interest in specifically OOD sample detection with deep neural networks [17, 55, 77, 89, 125]. Given that the problem is still in its infancy, there is an imminent need to standardize the problem and properly define a benchmark that is both realistic and principled to compare the previous and future approaches reliably.

Our contributions are as follows:

1. We introduce a new **o**utlier **d**etection test (OD-test), an abstract formulation of the task that offers a clear vision of applicable methods.
2. We establish a new benchmark to evaluate the existing techniques for OOD detection exhaustively and reliably.
3. We study the performance of several previously proposed methods under OD-test and highlight shortcomings and potential future directions for research. Furthermore, we demonstrate that the performance of current techniques quickly approaches the random prediction baseline as we make a transition to realistic high-dimensional images, highlighting the gap between the current state-of-art and what is required in practice.
4. We release our PyTorch [111] implementation to replicate all the results easily and also speed up progress on the presented problem. It is available at https://github.com/ashafaei/OD-test.

3.2 Related Work

The violation of the ID assumption is not the only way to wreak havoc on deep learning pipelines. Adversarial example [143] attacks are worst-case crafted signals in otherwise innocent-looking images that fool the neural networks into misclassification. These perturbations can be constructed in such a way that they break a variety of architectures trained on different datasets, with a high probability [104]. While the OOD detection is a model-independent problem, adversarial images exploit inductive bias in the model families. We limit our attention to the OOD detection problem.

To provide the appropriate context under which this study was executed, we keep the related work in this chapter as it was at the time of publication (2018). The previous work on similar problems can be categorized into: (i) uncertainty estimation, (ii) prediction with abstention, (iii) anomaly detection, and (iv) novelty detection.

The Uncertainty View An uncertainty measure could be directly applied to reject OOD samples as we would expect the uncertainty to be high on such inputs. The `MC-Dropout` [41] approach is a feasible uncertainty estimation method for a variety of applications [41, 42, 69]. Lakshminarayanan et al. [77] show an ensemble of five neural networks (`DeepEnsemble`) trained with an adversarial-sample-augmented loss is sufficient to provide a measure of predictive uncertainty. We evaluate `DeepEnsemble` and `MC-Dropout`.

The Abstention View If we allow our models to abstain from prediction at the cost of a penalty, we end up with the abstention view. The purpose of abstention is to incur a lower cost than the cost of misclassification when possible. The pair (h, r) consists of the predictive hypothesis $h \in \mathcal{H}$ and a reject function $r : \mathcal{X} \to \{0, 1\}$. The reject function makes the abstention choice and could be a threshold on the magnitude of the prediction [15] or chosen from a reject-hypothesis set $\mathcal{R}$ [30, 31]. The task is to find a pair (h, r) that minimizes an abstention-augmented loss. The reject function aims to detect when h would make a wrong prediction. The binary reject function gives us more flexibility on the choice of the reject class $\mathcal{R}$ compared

to the possible continuous formulations[1]. However, the reliability of r is still contingent on evaluation on a fixed distribution. If we encounter an OOD sample, we do not know *a priori* if r would reject it. Our formulation is inspired by the abstention view with key differences that we will discuss in Section 3.3. We show that all the prior work on OOD detection can be reduced to an abstract problem of learning a reject function within a specific class $\mathcal{R}$.

The Anomaly View *Density estimation*-based techniques assume that low measure samples are outliers. These approaches tend to work well mostly within low-dimensional or well-defined distributions. `PixelCNN++` [122] is an autoregressive model with a tractable likelihood that could be used within a density estimation scheme. Note that density estimation is *not equivalent* to the binary OOD detection: a perfect density estimator can solve the outlier detection problem, but a perfect outlier detector does not necessarily have the information needed to solve the density estimation problem. *Proximity*-based methods use a distance measure and the training data to flag anomalies. A simple strategy is to threshold the distances of the K-nearest neighbours for a given input – we learn the threshold with SVM and call it `K-NNSVM` (see Appendix A.2 for more information). Clustering methods reject points that do not conform to any of the identified clusters. The one-class SVM [126] with a radial basis function learns a conservative region that covers the train data. Goldstein and Uchida [48] show that the proximity-based approaches are empirically the most effective outlier detectors over a range of datasets. *Reconstruction*-based methods learn to reconstruct the train data, then try to reconstruct each given input. The samples that cannot be reconstructed well are then flagged as anomalies. We use an autoencoder with a reconstruction threshold to test this idea (`AEThreshold`).

The Novelty View Open-set recognition and novelty detection study the detection of anomalies at a *semantic* level. These methods are typically concerned with recognition of unseen classes, e.g., , new objects in the scene. This is a special case of OOD detection where the OOD samples explicitly differ by the semantic

[1] It subsumes the class of functions that can be used for continuous (uncertainty) estimation by adding a threshold.

content. However, the notion of OOD is more granular: an unseen viewpoint of a specific object violates the ID assumption, but it does not necessarily constitute a novelty. The notion of novelty is often underspecified in practice and results are limited to particular assumptions and problem definitions. Bendale and Boult [17] present `OpenMax`, a replacement for the softmax layer that detects unknown classes through evaluation against a representative neural activation of each class.

Deep Learning Literature The previous related work in deep learning can be categorized into two broad groups based on the underlying assumptions: (i) in-distribution techniques, and (ii) out-of-distribution techniques.

In-Distribution These methods focus primarily on the performance of the network on the in-distribution inputs to either calibrate the predictions or abstain from prediction. Guo et al. [52] observed that modern neural networks tend to be overconfident in their predictions. They show that temperature scaling in the softmax operator, also known as Platt scaling, can be used to calibrate the output probabilities of a neural network to empirically align the accuracy of a prediction with its probability. Their efforts fall under the uncertainty estimation approaches. Geifman and El-Yaniv [44] present a framework for selective classification with deep neural networks that follows the abstention view. A selection function decides whether to make a prediction or not. For the choice of selection function, they experiment with MC-Dropout and the softmax output. They provide an analytical trade-off between risk and coverage within their formulation.

Hendrycks and Gimpel [55] investigate OOD sample detection within computer vision, natural language processing, and speech recognition. They demonstrate that it is possible to detect OOD samples by simply thresholding the output softmax probabilities. We call this method `PbThreshold`. More recently, Liang et al. [89] present `ODIN`, a method based on (i) temperature rescaling and (ii) input perturbation to detect OOD samples. Temperature rescaling, in light of the previous work [52], provides the means of confidence calibration. They further posit that the predictive function, as represented by the deep neural network, would have a different behaviour around the in-distribution samples as opposed to OOD samples.

Therefore, the input perturbation serves as a way to assess how the network would behave nearby the given input. When the temperature is 1 and the perturbation step is 0 we simply recover the `PbThreshold` method. `ODIN`, the state-of-the-art at the time of this writing, is reported to outperform the previous work [55] by a significant margin. We also assess the performance of `ODIN` in our work. Further extensions rely on statistics of hidden representations [81, 113], construct classifier ensembles with subsets of data [147], or perform regression on word embeddings [132].

Wang et al. [149] present a "safe" classification paradigm based on generative adversarial networks (GANs) [49]. They train a generator per class, and during test, find the best approximate input to all the generators and use the distance of the generated samples from the input as a measure for prediction and uncertainty. There are also other ideas that rely on GANs [33, 74, 80, 125][2] to detect anomalies or novelty in the data. These methods provide an abstract idea which depends on the successful training of GANs. To the best of our knowledge, training GANs [49] is an active area of research, and it is not apparent what design decisions would be appropriate to implement these ideas in practice. Furthermore, some of these ideas are prohibitively expensive to execute at the time of this writing. We are therefore unable to evaluate these ideas fairly at this time.

All the previous studies primarily focus on low-dimensional MNIST [78], SVHN [107], and CIFAR [75] datasets. We evaluate several previously proposed solutions in controlled experiments on datasets with varying complexity. We show that, in such low-dimensional spaces, simple anomaly detection methods work as well, thus stressing that a more comprehensive evaluation is necessary for assessments of the current and future work.

3.3 OD-test: A Less Biased Evaluation of Outlier Detectors

In this section, we present a less biased evaluation scheme for outlier detectors. We then evaluate and compare the performance of the top outlier detectors under the traditional binary evaluation scheme and OD-test in Sec. 3.5.

[2]Note that some of the most recent work is not yet peer-reviewed.

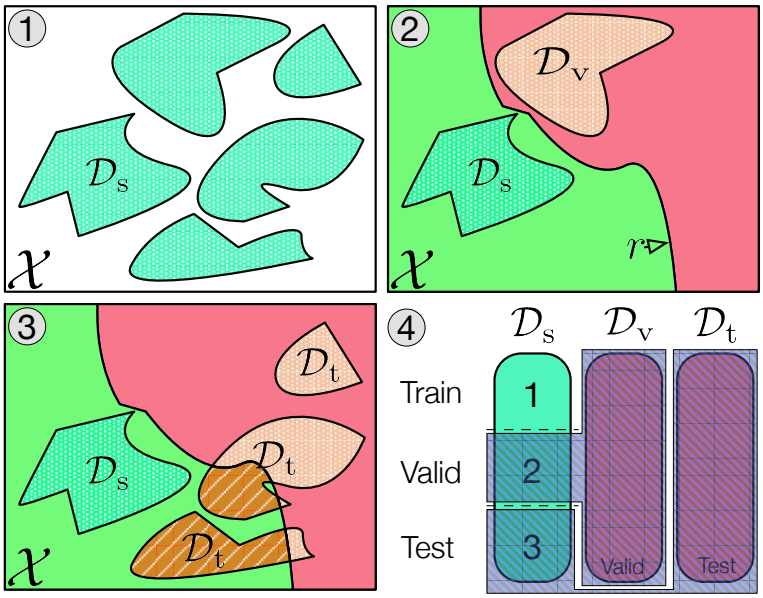

Figure 3.2: The OD-test. 1) A set of distributions visualized as shapes within $\mathcal{X}$. The source distribution $\mathcal{D}_\mathrm{s}$ is identified in the image; everything else is an outlier. 2) We pick one validation distribution $\mathcal{D}_\mathrm{v}$ and learn a binary reject function r that partitions the input space $\mathcal{X}$ based on $\mathcal{D}_\mathrm{s}$ and $\mathcal{D}_\mathrm{v}$ only. 3) We evaluate r on other distributions (as $\mathcal{D}_\mathrm{t}$) and measure the accuracy. 4) The dataset splits for each step.

Let us define the source distribution $\mathcal{D}_\mathrm{s}$ to be our input distribution. The objective is to decide whether a given sample belongs to $\mathcal{D}_\mathrm{s}$. We define a reject function $r : \mathcal{X} \to \{0, 1\}$ that makes this binary decision. Note that this decision can be made independently from the ultimate prediction task. While in the abstention view we reject the samples that the hypothesis h is likely to mislabel, here we reject the samples that do not belong to the source distribution $\mathcal{D}_\mathrm{s}$, hence decoupling the reject function r and the predictive hypothesis h.

If the reject function r flags an input, then the sample does not belong to the source distribution; thus, the output of the pipeline may not be reliable. On the other hand, if the function accepts an input, we can continue the pipeline with the ID assumption. This form of rejection is more relaxed than the previous formulations. In addition to the previous methods, we can study new approaches that operate in the input space directly (e.g., `K-NNSVM`, or `AEThreshold`).

The r function is a binary classifier, the classes are in-distribution vs. out-of-distribution. To learn the classifier, we might adopt the supervised learning assumptions and use a dataset $\mathcal{D}_{\text{v}}$ as a representative of the OOD samples (supervised outlier detection). Unfortunately, this approach may be misleading because the learned models can be biased by $\mathcal{D}_{\text{v}}$. A high accuracy in this scenario may not yield an accurate model in practice in many settings where the outliers may not look like samples from $\mathcal{D}_{\text{v}}$. The actual OOD samples are beyond our direct reach and our models can easily overfit in distinguishing $\mathcal{D}_{\text{s}}$ from $\mathcal{D}_{\text{v}}$ (we verify this empirically). We present a less optimistic evaluation framework that prevents scoring high through overfitting.

Specifically, we introduce a third "target" distribution $\mathcal{D}_{\text{t}}$ to measure whether a method can actually detect outliers that are not only outside of $\mathcal{D}_{\text{s}}$ but that also might be outside of $\mathcal{D}_{\text{v}}$. The idea is to treat the problem as a binary classification involving three different datasets. Similar to the supervised outlier detection, we begin by learning a reject function to distinguish $\mathcal{D}_{\text{s}}$ from $\mathcal{D}_{\text{v}}$. But, for evaluation, we use a third unseen distribution $\mathcal{D}_{\text{t}}$ instead of $\mathcal{D}_{\text{v}}$ (see Fig. 3.2). $\mathcal{D}_{\text{t}}$ represents OOD examples that were not encountered during training – a more realistic evaluation setting for uncontrolled scenarios. The spectrum of choices for $\mathcal{D}_{\text{s}}$, $\mathcal{D}_{\text{v}}$, and $\mathcal{D}_{\text{t}}$ allows for a rigorous evaluation of the r functions. The pseudocode of the evaluation procedure is outlined in Alg. 1.

When $\mathcal{D}_{\text{v}}{=}\mathcal{D}_{\text{t}}$, we recover the evaluation of the previous work in the deep learning literature. However, we specifically require $\mathcal{D}_{\text{v}}{\neq}\mathcal{D}_{\text{t}}$ to ensure the evaluation is not biased by $\mathcal{D}_{\text{v}}$. Analogous to supervised learning, $\mathcal{D}_{\text{v}}$ is a validation set (we do not care about our performance at this task) and $\mathcal{D}_{\text{t}}$ is the test set (we care about our performance at this task).

If the method $\mathcal{M}$ successfully learns a density function for $\mathcal{D}_s$, it would be capable of scoring very high in this evaluation. With a density function, we only would have to pick a single threshold to reject OOD samples. Methods that yield a confidence estimate can be used similarly. A typical binary classification method is also not sufficient to score high on this benchmark since we change the second distribution during the test stage. In Section 3.5 we show how a traditional binary classifier would fall short. The methods that learn conservative boundaries around $\mathcal{D}_{\text{s}}$ will have a higher chance of success. All the existing approaches can be

Algorithm 1: OD-test

input : $\mathcal{D}_{\mathrm{s}} = (\mathcal{D}_{\mathrm{s}}^{\mathrm{train}}, \mathcal{D}_{\mathrm{s}}^{\mathrm{valid}}, \mathcal{D}_{\mathrm{s}}^{\mathrm{test}})$ the source dataset.
input : $\mathrm{D} = \{\mathcal{D}_i\}$ a **set** of non-overlapping datasets with the source $\mathcal{D}_{\mathrm{s}}$.
input : $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ the method under evaluation.

begin
  $A \longleftarrow \{\}$
  `/* Generate a reject-hypothesis class` $\mathcal{R}$ `using` $\mathcal{D}_{\mathrm{s}}^{\mathrm{train}}$. `*/`
  $\mathcal{R} \longleftarrow \mathcal{M}(\mathcal{D}_{\mathrm{s}}^{\mathrm{train}})$
  for $\mathcal{D}_{\mathrm{v}} \in \mathrm{D}$ **do**
    `/* Find the best binary classifier in` $\mathcal{R}$. `*/`
    $r \longleftarrow \mathrm{train}(\mathcal{R},\ \{\mathcal{D}_{\mathrm{s}}^{\mathrm{valid}} : 0, \mathcal{D}_{\mathrm{v}} : 1\})$
    for $\mathcal{D}_{\mathrm{t}} \in \mathrm{D} \backslash \{\mathcal{D}_{\mathrm{v}}\}$ **do**
      `/* Evaluate accuracy of` r. `*/`
      acc $\longleftarrow \mathrm{eval}(r,\ \{\mathcal{D}_{\mathrm{s}}^{\mathrm{test}} : 0, \mathcal{D}_{\mathrm{t}} : 1\})$
      add acc to A
  return mean(A)

implemented within this framework (see Appendix A.2 for more information).

Example – `PbThreshold`

This method's idea is that if we threshold the maximum probability of a discriminative neural network, we should detect outliers. To test this idea, we need to train a deep neural network on a $\mathcal{D}_{\mathrm{s}}$ dataset and then find the probability threshold to reject outliers.

Input

$\mathcal{D}_{\mathrm{s}}$ Let us assume $\mathcal{D}_{\mathrm{s}}$ is the MNIST dataset, which is already split into a train, validation, and a test set.

$\{\mathcal{D}_i\}$ is the outlier set. Let us assume we have CIFAR 10 and FashionMNIST as the outliers. None of these outlier sets have any classes in common with $\mathcal{D}_{\mathrm{s}}$.

$\mathcal{M}$ method produces a reject hypothesis class $\mathcal{R}$ for outlier detection. In the case of `PbThreshold`, the class consists of a pre-trained neural network and a free threshold parameter τ. During the training of our outlier detection, we will choose τ, and therefore, pick a specific member $r \in \mathcal{R}$.

Evaluation

1. We first train a deep neural network on MNIST to discriminate the image classes (line 3). The hypothesis class $\mathcal{R}$ returned on line 3 will have a single free parameter τ, the threshold to use on the underlying classifier. However, at this point in the evaluation, we have not committed to a specific threshold yet. We will be using $\mathcal{R}$ in the next step to find the threshold. Note that as we change $\mathcal{D}_{\mathrm{s}}$ the corresponding reject class $\mathcal{R}$ will also change since the underlying neural network will be different.

2. Now we pick the first outlier dataset $\mathcal{D}_{\mathrm{v}}$ from the outlier dataset set. Let us assume we choose CIFAR10.

3. We must learn the optimal threshold τ in the next step. On line 5, we pick the best threshold τ to discriminate between MNIST and CIFAR10. OD-test does not specify the "best" threshold or how we should learn the best threshold. The practitioner must decide what measure and learning method are appropriate for the application. We can find the optimal threshold with a simple 1-dimensional search if we use classification accuracy as the evaluation measure. After picking the threshold τ, we have committed to a specific reject function r, which we chose from $\mathcal{R}$. Next, we evaluate how robust our reject function is to a new set of outliers.

4. After finding the best threshold τ on line 5, we evaluate the learned reject function on MNIST and the other outlier dataset FashionMNIST (on line 7).

5. In the next iteration of the main loop, we will choose FashionMNIST as the first outlier dataset $\mathcal{D}_{\mathrm{v}}$ and test the resulting threshold on CIFAR10 ($\mathcal{D}_{\mathrm{t}}$).

6. We average all the results and return a single value.

For unsupervised OOD detection methods the evaluation is a single loop over $\mathcal{D}_{\mathrm{t}}$ (see Appendix A.1).

In the density estimation problem the only important factor is the intrinsic complexity of $\mathcal{D}_{\mathrm{s}}$. However, our formulation of the problem is controlled by several factors. The similarity between $\mathcal{D}_{\mathrm{s}}$, $\mathcal{D}_{\mathrm{v}}$, and $\mathcal{D}_{\mathrm{t}}$ would control the difficulty of the problem. If $\mathcal{D}_{\mathrm{s}}$ and $\mathcal{D}_{\mathrm{v}}$, the source and the validation, are too similar, a high accuracy would require learning more complex boundaries; therefore it would be a more difficult problem. Using our scheme, we can gradually make the problem harder and harder and improve our methods until they are satisfyingly accurate. Note that we can empirically assess the difficulty of separation without having an explicit notion of similarity. What we empirically observe is that separating low-level statistics is much easier than separating high-level concepts. If $\mathcal{D}_{\mathrm{v}}$ and $\mathcal{D}_{\mathrm{t}}$, the validation and the target, are too similar, the learned reject function r could yield good results due to overfitting. We cycle through mutually exclusive datasets in our evaluation and aggregate the results to reduce the potential biases in the performance estimate.

3.4 Evaluation

There are several evaluation metrics used in the previous work [55, 89]. These metrics, in conjunction with the possible combination of the evaluation datasets, lead to large tables that make the interpretation and reliable comparison of the results difficult. Measuring progress under such circumstances could be misleading and overly optimistic. To simplify the evaluation procedure, we equalize the binary classes and only measure accuracy. Furthermore, we require the methods to pick the optimal parameters such as the threshold. This simplification allows straightforward aggregation and analysis of the results and provides a simple baseline of random prediction. We can meaningfully average over multiple experiments and robustly compare methods in a variety of conditions.

We can also incorporate the prior knowledge on abundance and the risk associated with the OOD samples into the evaluation by modifying the implicit objective on line 5 and 7 similar to the traditional supervised learning settings. We leave the proper choice of application-dependent schemes to the practitioner and focus

	$\mathcal{D}_\text{s}$ – Source			Outliers		
	Train	Valid	Test	All	$\dim(\mathcal{X})$	$\lvert\mathcal{Y}\rvert$
MNIST	(50 k	10 k)	10 k	70 k	784	10
FashionMNIST	(50 k	10 k)	10 k	70 k	784	10
NotMNIST				18.6 k	784	10
CIFAR10	(40 k	10 k)	10 k	60 k	3072	10
CIFAR100	(40 k	10 k)	10 k	60 k	3072	100
TinyImagenet	100 k	10 k	10 k	110 k	12,288	200
STL10	5 k	(4 k	4 k)	13 k	27,648	10

Table 3.1: A summary of the datasets. The datasets are split as indicated by the parentheses. When the dataset is used a source $\mathcal{D}_\text{s}$, we split according to the original dataset specification and the above table. When the dataset is used as an outlier, we use the entire set of samples.

on assessing the discriminative power of the methods in the fixed $50/50$ scenario *without* any prior knowledge. Through this constraint, we ensure that the methods that rely on the prior likelihood cannot perform better than random prediction.

We extend the previous work by evaluating over a broader set of datasets with varying levels of complexity. The variation in complexity allows for a fine-grained evaluation of the techniques. Since OOD detection is closely related to the problem of density estimation, the dimensionality of the input image will be of vital importance in practical assessments. As the input dimensionality increases, we expect the task to become much more difficult. Therefore, to provide a more accurate picture of performance, it is crucial to evaluate the methods on high dimensional data. Table 3.1 summarizes the datasets that we use. We also evaluate with uniform and Gaussian noise for outliers. For a given $\mathcal{D}_\text{s}$ we filter out the conflicting classes to ensure there is no overlap between the corresponding distributions.

We evaluate the following methods:

- `BinClass`. A traditional binary classifier that is directly trained on $\mathcal{D}_\text{s}$ vs. $\mathcal{D}_\text{v}$.
- `PbThreshold`. A threshold on the softmax.
- `ScoreSVM`. An SVM [29] classifier on the logits.
- `LogisticSVM`. Similar to `ScoreSVM`, but the underlying classifier is

trained with logistic loss.

- `ODIN`. A threshold on the scaled softmax outputs of the perturbed input.
- `K-NNSVM`. A linear SVM on the sorted Euclidean distance between the input and the k-nearest training samples. Note that a threshold on the average distance is a special case of `K-NNSVM`.
- `AEThreshold`. A threshold on the autoencoder reconstruction error of the given input. We train an autoencoder with binary cross-entropy (BCE) and mean squared error (MSE).
- `K-MNNSVM`, `K-BNNSVM`. A `K-NNSVM` applied on the learned hidden representations of an autoencoder with MSE or BCE.
- `K-VNNSVM`. Similar to the previous, except we use the learned representation of a variational autoencoder [73].
- `MC-Dropout`. A threshold on the entropy of average prediction of 7 evaluations per input.
- `DeepEnsemble`. Similar to `MC-Dropout`, except we average over the predictions of 5 networks that are trained independently with adversarial-augmented loss.
- `PixelCNN++`. A threshold on the log-likelihood of each input.
- `OpenMax`. Similar to `ScoreSVM`, but we use the calibrated output of the OpenMax module that also includes a probability for an unknown class.

We use two generic architectures: `VGG-16` [138] and `Resnet-50` [54] and reuse the same networks for all the methods to provide a fair and meaningful comparison. We train these architectures with cross-entropy loss (CE), and k-way logistic loss (KWL). CE loss enforces mutual exclusion in the predictions while KWL loss does not. We test these two loss functions to see if the exclusivity assumption of CE hurts the ability to predict OOD samples. CE loss cannot make a `None` prediction without an explicitly defined `None` class, but KWL loss can make `None` predictions through low activations of all the classes.

Note that our formulation of the problem separates the target task from the OOD sample detection. Thus, it is plausible to use, for OOD detection, a different predictive model from the actual predictive model of the target problem if there is an advantage. We tune the hyper-parameters of these methods following the best practices and the published guidelines in the respective articles. The implementation details, a discussion of evaluation cost, and the performance statistics of the above

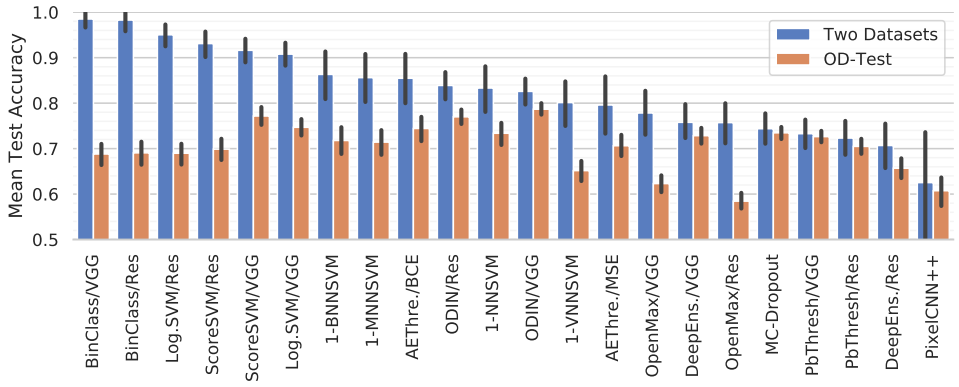

Figure 3.3: Evaluation with two datasets versus OD-test. Evaluating OOD detectors with only two distributions can be misleading for practical applications. The error bars are the 95% confidence level. The two-dataset evaluations are over all possible pairs of datasets ($n = 46$), whereas the OD-test evaluations are over all possible triplets ($n = 308$).

methods are in Appendix A. The PyTorch implementation with the pre-trained models is available on https://github.com/ashafaei/OD-test.

3.5 Results

We evaluate the methods in a controlled regime against the datasets in Tab. 3.1. We run over 10,000 experiments on all combinations of $\mathcal{D}_\text{s}$, $\mathcal{D}_\text{v}$, and $\mathcal{D}_\text{t}$. First we analyze the aggregated results, then we look at the breakdown of the accuracy per source dataset.

Figure 3.3 and Figure 3.4 show the average accuracy. Each method under OD-test is tested over the same set of 308 experiments consisting of *all non-overlapping triplet of datasets*. The two-dataset evaluations are averaged over 46 experiments (all possible pairs). See the project page for the list of experiments.

Figure 3.3 compares the mean test accuracy of methods within OD-test and the two-dataset setting. Methods that perform well in distinguishing two datasets fail when a third dataset is introduced. The gap in relative performance within each evaluation highlights the importance of having a more realistic assessment in practice. The general trend of the evaluation indicates that the methods that have

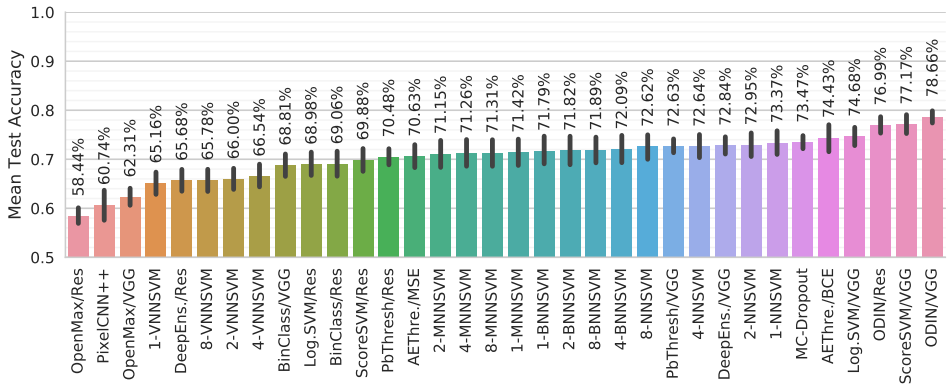

Figure 3.4: The average test accuracy of the OOD detection methods over 308 experiments per method. The error bars are 95% confidence level. `/VGG` or `/Res` indicates the backing network architecture. `#-NN./` is the number of nearest neighbours. A random prediction would have an accuracy of 0.5.

a higher degree of freedom in the space characterized by $\mathcal{R}$ (such as `BinClass`) are the most susceptible ones to overestimating accuracy within the traditional evaluation approach. This observation is consistent with the bias-variance tradeoff in learning theory.

`BinClass`, the direct binary classifier, achieves a near-perfect accuracy on the train and validation of the same datasets, however, once tested on a third dataset that it has not seen before, the average accuracy drops to 68% for both `VGG` and `Resnet`. This example demonstrates why we need to adopt a new evaluation scheme. The classifier overfits to the two distributions ($\mathcal{D}_{\text{s}}$ and $\mathcal{D}_{\text{v}}$) effortlessly, but it cannot distinguish a third distribution ($\mathcal{D}_{\text{t}}$). Because of the diversity in the OOD samples, we may always encounter new input that we have not seen before.

`MC-Dropout` and `DeepEnsemble`, the two uncertainty techniques, do not seem to provide a strong enough signal to distinguish the two classes compared to the simpler `ScoreSVM`. Interestingly, `MC-Dropout` has a higher accuracy than `DeepEnsemble`. Considering the training cost of `DeepEnsemble`, using `MC-Dropout` is a more favourable choice.

`VGG`-backed and `Resnet`-backed methods significantly differ in accuracy. The gap indicates the sensitivity of the methods to the underlying network architectures.

PbThreshold, ScoreSVM, and ODIN all prefer VGG over Resnet even though Resnet networks outperform the VGG variants in the underlying image classification on average. This means that the image classification accuracy may not be the only relevant factor in performance of these methods. ODIN is less sensitive to the underlying network. Furthermore, training the discriminator networks with KWL loss consistently reduces the accuracy of OOD detection methods on average. ScoreSVM/VGG and ScoreSVM/Res both outperform LogisticSVM/VGG, and LogisticSVM/Res respectively. Similarly, the autoencoders that were trained with BCE loss (AEThre./BCE) outperform the ones trained with MSE loss (AEThre./MSE). Note that we are comparing identical architectures.

Within the nearest-neighbour methods, #-NNSVM, #-BNNSVM, #-MNNSVM, and #-VNNSVM, the number of the nearest neighbours does not significantly impact the accuracy on average. However, performing the nearest-neighbour in the input space directly outperforms nearest-neighbour in the latent representations of autoencoders (BNNSVM, and MNNSVM) and VAE (VNNSVM). Interestingly, 1-NNSVM has a higher accuracy than thresholding the probability (PbThresh) and DeepEnsemble on average. For #-NNSVM, if the reference samples fit the GPU memory, a naive implementation could be faster than a forward pass on the neural networks of large datasets like TinyImagenet.

PixelCNN++, the method that estimates the log-likelihood, has a surprisingly low accuracy on this problem on average. We suspect the auto-regressive nature of the model, specifically when coupled with the IID assumption, may be the reason for its failure. The network approximates the likelihood only in the vicinity of the training data. If we evaluate the model on points that are far from the training data, the estimates are not reliable anymore.

Next, we break down the performance and study these methods on each $\mathcal{D}_{\text{s}}$. Figure 3.5 shows the average test accuracy across each source dataset $\mathcal{D}_{\text{s}}$. For the full figure, see appendices. Our quantification of performance shows that all the methods have a much lower accuracy on high-dimensional data than the low-dimensional data.

In low-dimensional datasets, K-NNSVM performs similarly or better than the other methods. In the high-dimensional case, however, the accuracy approaches the random baseline quickly. Interestingly, K-NNSVM performs better on STL10

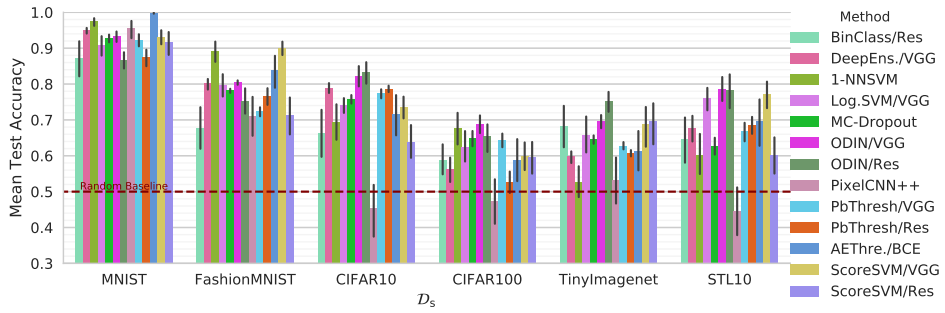

Figure 3.5: The test accuracy over 50 experiments per bar. The error bars are the 95% confidence level.

(96×96) than TinyImagenet (64×64) which might be due to the higher diversity in TinyImagenet compared to STL10. Given the high accuracy of `K-NNSVM` in 784 dimensions, it might be feasible to learn an embedding for high-dimensional data, or learn a kernel function, to replace the original image space and enjoy a high accuracy. Going from CIFAR10 to CIFAR100, the dimensionality and the dataset size remains the same, the only changing factor is the diversity of the data and that seems to make the problem as difficult as the higher dimensional datasets. Except for `K-NNSVM`, the accuracy of other methods drops significantly in this transition.

`AEThreshold` has a near perfect accuracy on MNIST, however, the performance drops quickly on complex datasets. `AEThreshold` also outperforms the density estimation method `PixelCNN++` on all datasets. We did not explore other autoencoder architectures – more research on better architectures or reconstruction constraints for `AEThreshold` may potentially have a high pay-off. The top-performing method, `ODIN`, is influenced by the number of classes in the dataset. Similar to `PbThreshold`, `ODIN` depends on the maximum signal in the class predictions, therefore the increased number of classes would directly affect both of the methods. Furthermore, neither of them consistently prefers `VGG` over `Resnet` within all datasets. Overall, `ODIN` consistently outperforms others in high-dimensional settings, but all the methods have a relatively low average accuracy in the 60%-78% range.

We can summarize the main observations as follows:

1. Outlier detection with two datasets yielded overly optimistic results with `VGG`

and `Resnet`.

2. The `MC-Dropout` and `DeepEnsemble` uncertainty methods were not reliable enough for OOD detection.
3. A more accurate image classifier did not lead to a more accurate outlier detector on average.
4. KWL loss did not yield a better-calibrated model than CE loss for OOD detection.
5. The autoencoders worked better with BCE loss than MSE loss for OOD detection.
6. The nearest neighbour methods were competitive in low-dimensional settings, both in computational cost and accuracy. However, the latent representations of vanilla (variational)-autoencoders were not useful for this task when combined with nearest neighbour methods.
7. The state-of-the-art auto-regressive density estimation method had a surprisingly low accuracy, performing worse than the random prediction baseline in some settings.
8. The number of the classes in a dataset (a proxy to diversity) affected the accuracy more than the dimensionality of the data in our experiments.
9. Although `ODIN` outperforms other methods in realistic high-dimensional settings, its average accuracy is still below 80%.

To perform supervised OOD sample detection in practice, we have to pick a method and choose a training outlier set $\mathcal{D}_{\text{v}}$. Assuming that $\mathcal{D}_{\text{v}}$ may not represent the full spectrum of anomalies, we should pick methods that do not overfit to $\mathcal{D}_{\text{v}}$. OD-test tells us which methods are less likely to overfit to a chosen $\mathcal{D}_{\text{v}}$ and therefore be more reliable in the face of an unseen OOD sample. Our results show that a two-dataset evaluation scheme can be too optimistic in identifying the best available method. OD-test is a more realistic evaluation of OOD sample detectors. In practice, the outlier set $\mathcal{D}_{\text{v}}$ should contain the largest variety of anomalies that we can use, and the method should be the one that is more accurate and less likely to overfit.

3.6 Research Conclusion

By detecting OOD samples, we can ensure the deep learning pipelines operate as expected. While similar problems are studied in various domains of artificial

intelligence, none of them adequately address this problem. We introduced OD-test, a new formulation of the problem that provides a realistic assessment of the OOD detection methods. We further presented a new benchmark for OOD sample detection within image classification pipelines. We showed that the traditional supervised learning approach to OOD detection does not always yield reliable results – the previous assessments of the OOD detectors are potentially too optimistic to be practical in many scenarios. We presented a comprehensive evaluation of a diverse set of approaches across a wide variety of datasets for the OOD detection problem. Furthermore, we showed that *none of the methods is suitable out-of-the-box for high-dimensional images*. We believe any progress on this problem could influence other areas such as active learning or unsupervised learning. We release the open-source PyTorch project with the pre-trained models to replicate the results of our study. Furthermore, we invite the machine learning community to tackle the outlined challenges outlined in this work.

3.7 Follow-up Developments

Ever since the initial publication [129, 130], the interest in out-of-distribution detection has grown significantly. According to Google Scholar, there are 172 papers on "out-of-distribution detection" published by the research community in 2018, 656 in 2019, and 810 by August of 2020. Our work is primarily cited in the literature for the results on the unreliability of the existing methods within practical applications.

As of August 2020, the published work on OOD detection within the major machine learning venues continues to evolve with the traditional overly-optimistic binary evaluation schemes.

Chapter 4

AutoPortrait: Automatic Portrait Enhancement from Studios to Mobile Phones – Portrait Cropping

In the previous chapters, we studied practical issues that arise in the application of DNNs. Next, we focus on the application of DNNs in computational photography within real-world workflows. During a two-year internship with a local photography studio, we studied three specific problems: portrait colour correction, portrait cropping, and portrait face retouching. The result of our work on these three problems created the AutoPortrait pipeline. AutoPortrait addresses all three problems. In this chapter, we present our novel approach to portrait cropping while Chapter 5 presents our novel approach to face retouching. For colour correction, we use off-the-shelf techniques with minor adjustments that we describe in Appendix B.2. The material presented in the following chapters is the result of the author's work at The Artona Group and Skylab Technologies. Unless stated otherwise, the images we present in Chapter 4 and Chapter 5 belong to Artona and are used with permission.

4.1 Introduction

Phone cameras have reached an unprecedented level of image quality and resolution. Huawei, Xiaomi, and Apple products now exceed a level of detail that was once exclusive to high-end single-lens reflex (SLR) cameras.[1] With the advance of these consumer-level products, demand for image enhancement software has also increased significantly. Software enhancement features such as DeepFusion, Night Sight, and artificial Bokeh now exist in virtually all high-end consumer phone cameras. Post-processing beautification products such as Meitu have surpassed a hundred million active users each month.[2] Further improvement of phone cameras will only increase the demand for high-quality software enhancement.

On the other side, photography studios process thousands of high-quality portraits a day. In the USA alone and within the niche market of school day photography, studios process over 56 million photos annually. A minimal production pipeline includes at least three laborious tasks: head cropping, colour correction, and face retouching. Surprisingly, these tasks are still not automated in the largest studios in North America that we could reach. Most studios that we surveyed were not happy with the quality of existing automation products, instead, relying on human labour. The most common comments we received expressed the following concerns:

1. Automatic colour-correction workflows are not adaptive to the studios' production style.

2. Automatic head cropping applications do not yield consistent head sizes.

3. Automatic retouching destroys the image with "plastic skin" outputs.

We show that the previous work does not adequately address the practical requirements. Furthermore, the lack of publicly-available datasets has stifled the development of practical solutions. We take a close look at these tasks, discuss our findings, present our novel solution inspired by the most recent developments in computer vision and machine learning, and show that average-consumers and professionals can benefit from our pipeline. Finally, we release the first large-scale face retouching dataset with our baseline.

[1] dxomark.com

[2] markets.businessinsider.com/news/stocks/meitu-inc-announces-2018-annual-results-1028045640

Figure 4.1 shows an overview of our AutoPortrait pipeline. Since we are developing a practical solution for real-world applications, we require the pipeline's output to be impeccable – the output image should not be deformed or degraded in any way. Furthermore, since we process high-resolution images, we need to design resource-sensitive models: a 24-megapixel image uncompressed in floating-point precision occupies over 250 MB memory. We first present the problems and discuss the subtleties that prevent a direct application of the existing methods. Then, we develop new strategies to address the specific challenges of the application. Our contributions are:

1. Chapter 4: We introduce the problem of perceptual head cropping. We describe a novel aesthetic-based head-cropping technique that is robust to noisy groundtruth. Our method efficiently finds the optimal cropping window with first-order optimization techniques. We generate perceptually consistent cropping and can incorporate custom cropping preferences while detecting failures automatically.

2. Chapter 5: We present a carefully-designed and validated fast face retouching neural architecture with a low memory footprint that preserves the skin's distinctive features while removing imperfections. Our solution enables quick and high-quality automated face retouching for the first time, producing output that is more aesthetically pleasant than the groundtruth data. We compare our method with previous related work qualitatively and quantitatively. We show that our method generalizes across datasets and is useful for mobile phone applications.

3. Chapter 5: We release the first large-scale professional face retouching dataset Flickr-Faces-HQ-Retouched (FFHQR) with our baseline to encourage more research on this problem.

4.2 Overview

Photography studios usually process a large volume of images during each season. Each image must go through a photo editing pipeline before it is delivered to the customers. A basic portrait editing pipeline consists of three steps:

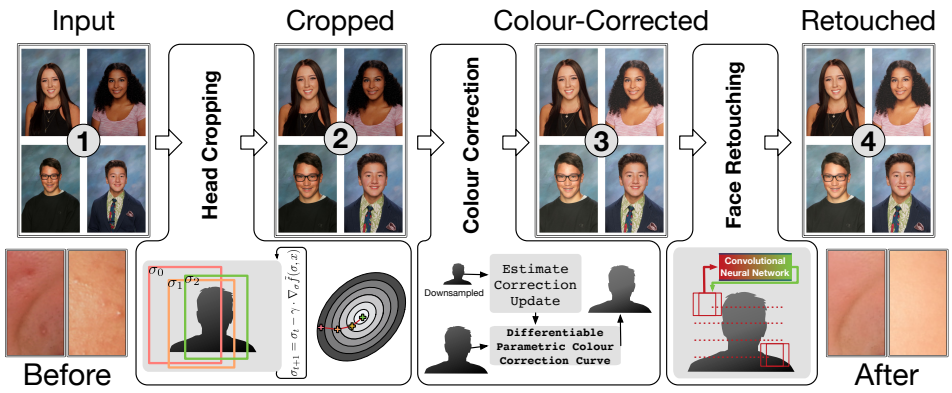

Figure 4.1: Overview of our AutoPortrait pipeline: (1) 24 MP input portraits, (2) portraits are cropped to a perceptually consistent size and location, (3) portraits are colour-corrected, (4) portraits are skin-retouched while preserving fine textures. Each block of our pipeline can be used individually and generalizes to mobile phone applications.

Head Cropping is adjusting the scale and translation of the head to ensure consistency among all the images within a batch. (Sec. 4.4)

Colour Correction is the process of adjusting the colour distribution within an image to ensure maximum *print* quality. There is no standard among the photography studios on how the image should be colour-corrected. We will need to learn the subjective preferences of each studio. (Appendix B.2)

Portrait Retouching is the process of removing imperfections (wrinkles, blemishes, eye-bags, etc.) to improve the general aesthetics of a portrait. (Chapter 5)

We will provide a more rigorous definition of each task later. Our pipeline (Fig. 4.1) mirrors the same steps. Currently, human professionals perform all of these tasks. As a result of human involvement, the pipelines do not scale and cannot handle a large volume of images. All of these tasks are laborious. Although some of these problems appear to be trivial to solve with off-the-shelf methods, we show that each task comes with unique requirements that the existing techniques fail to satisfy. Furthermore, for tasks such as skin retouching, there is no prior established

academic work. To address these problems, we present our novel approaches that successfully address (most of) the requirements and perform a careful comparative study on the alternative methods. In this chapter, we focus on the cropping problem.

4.3 Related Work

Image cropping has an extensive body of research for applications ranging from thumbnail generation for small displays [25, 26, 100, 141] to phone camera enhancement [99]. The relevant image cropping strategies are broadly (i) landmark-based, or (ii) aesthetic-based. Landmark-based methods extract target key-points in the image and perform geometric alignment to determine the cropping window. This approach can be useful when we target a specific and constrained domain. However, as we discuss later, the geometric approach does not adequately address our problem.

The aesthetic-based cropping uses an aesthetic function $\mathcal{A}(x)$ that measures the quality of the cropped image x [101, 148]. The early methods for aesthetic assessment used hand-picked features based on composition rules or image quality assessment methods [34, 68, 98, 156, 161]. The most recent techniques estimate the subjective aesthetic function directly from labelled training data. Learning the aesthetic function has been enabled by the availability of large-scale in-the-wild datasets such as AVA [105] and CUHKPQ [98].

Given a cropping transformation $\mathrm{T}_\sigma(x)$ parameterized by σ, we can assess the aesthetic value of a cropping window as $\mathcal{A}(\mathrm{T}_\sigma(x))$. Let us assume σ_x^* is the most aesthetically pleasant cropping window of an image x under $\mathcal{A}$, i.e., $\sigma_x^* \in \arg\max_\sigma \mathcal{A}(\mathrm{T}_\sigma(x))$. One way to approach this problem is to learn the best cropping window estimator $\mathrm{g}(x) \rightarrow \sigma_x^*$ directly as a regression problem. For instance, Guo et al. [53] use gradient boosting for regression. We will use direct regression as a baseline for our method.

Li et al. [85] frame the cropping task as a sequential decision-making process under a reinforcement learning framework within which an agent chooses to adjust the cropping proposal σ with the goal of maximizing the aesthetic value $\mathcal{A}(\mathrm{T}_\sigma(x))$. Zeng et al. [160] restrict the search space of cropping windows σ to speed up aesthetic cropping. Since we cannot easily get individuals to quantify quality, we usually do not have calibrated samples to learn $\mathcal{A}$. Furthermore, searching over σ to

find the highest scoring window [85, 108, 148, 160, 161] is a combinatorial search that is both expensive and does not guarantee an optimal result. Our approach is a novel aesthetic-based formulation for cropping that exploits the constraints of the portrait domain. Specifically, we construct a quadratic upperbound of $\mathcal{A}$, and find the optimal σ^* efficiently using first-order optimization techniques within only a few iterations.

4.4 Visually-consistent Portrait Cropping

Producing visually-consistent head cropping is desirable in several scenarios ranging from aesthetics improvement to quality assurance. Any software or online service that displays thumbnails of faces (e.g., Google Photos), can aesthetically improve with a visually consistent head cropping. Furthermore, within the production pipeline of professional portraits, we need to ensure all the photos are consistently cropped to maintain product consistency across the users. The inconsistency of previous head cropping strategies can be best demonstrated on a grid of portrait photos (Fig. 4.2). In real-world scenarios where the aesthetic aspect of a portrait product is important, at least two iterations of human labour is needed to crop and compare the final images.

At first glance, a geometrical facial landmark-based approach to image cropping seems sufficient [67, 161]. However, as shown in Figure 4.2, facial landmarks alone are not sufficient to create visually consistent portraits. A fixed pattern of landmark locations results in an inconsistent head sizing as variations between the individuals affect the scaling. In Fig. 4.2, we align the facial landmarks to the most similar landmarks in the training groundtruth data. The results do not appear harmonious, most noticeably in the scale. The perceptual judgment appears to dominate any geometrical arrangement when it comes to faces [139, 157]. Perceptual factors such as head shape, face tilt, shoulder size, perceived age and gender, clothing, and glasses all contribute to consistent-looking cropping.

We need to capture the subjective perception of head size to create consistent-looking portraits. We call this problem perceptual head-cropping (PHC) to emphasize the role of perception in the cropping problem. Although the general cropping problem is studied in the computer vision literature, to the best of our knowledge,

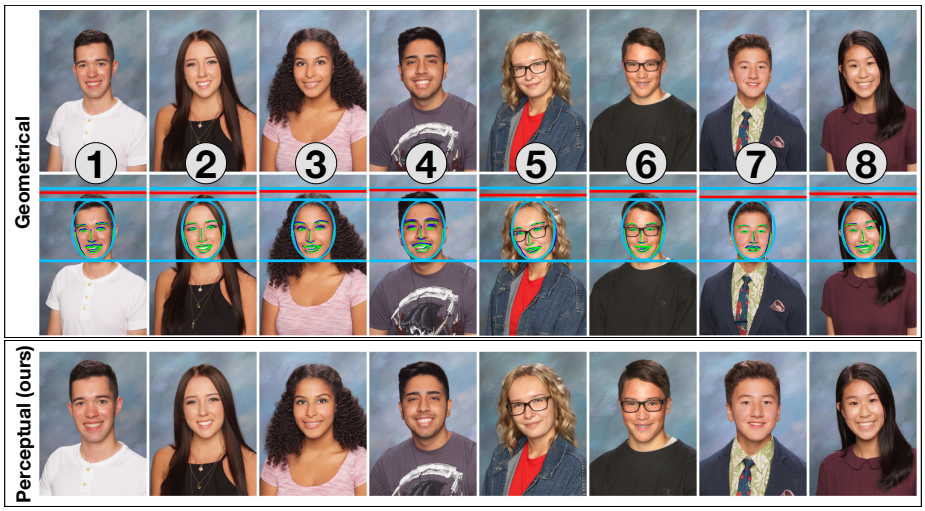

Figure 4.2: A comparison of geometrical landmark alignment to ours. Images (1), (5), (6), and (7) are relatively too small with geometrical alignment. Our perceptual approach creates more consistent head-sizes.

Figure 4.3: The red boxes are the reference cropping windows.

the perceptual head-cropping problem has been overlooked.

In the general cropping problem: (i) there could be several distinct cropping windows with equally high aesthetic value, and (ii) the problem is generally not too sensitive to the scaling of the window – a slightly larger or smaller window than the groundtruth is still acceptable. However, in the perceptual head cropping problem, there is only one globally optimal window at every scale. Unlike previous work, we do not learn an explicit aesthetic function. We exploit the uniqueness property of the portrait cropping problem and learn a gradient estimator of a proxy aesthetic function with respect to the cropping parameters (we will discuss this at length shortly). This modelling assumption allows a straightforward mechanism to find

the optimal σ^* with first-order optimization techniques.

Given the image x and the current cropping window σ, our model predicts how the cropping window can be aesthetically improved. To find the optimal cropping, we run gradient descent: we iteratively follow the predictions starting from an initial configuration. If the aesthetic function $\mathcal{A}$ is concave in σ, this approach yields the optimal cropping – the stationary point is the optimal point by definition. Although the function $\mathcal{A}$ may be quasi-concave in σ for portrait cropping, we can construct a concave lower-bound to find σ^*. Therefore, we can use convex optimization methods to efficiently find σ^* instead of expensive search strategies [86, 150, 152, 160].

Formulation

Let us define $f(\sigma, x) := -\mathcal{A}(\mathrm{T}_\sigma(x))$. f gives the negative aesthetic value of cropping x with a window parameterized by σ – we take this step so that the inference problem becomes a *minimization* of f with a cleaner, more familiar, notation. Recall that σ_x^* is the optimal cropping parameter for image x. We can construct a convex upper-bound proxy $\tilde{f}$ over σ as follows:

$$f(\sigma, x) \leq \tilde{f}(\sigma, x) := \frac{1}{2}\|\sigma - \sigma_x^*\|_2^2 + f(\sigma_x^*, x), \tag{4.1}$$

where the first term is an ℓ_2^2 error on the distance from the optimal σ_x^*, and the second term is the optimal aesthetic value. Note that at the optimal point σ_x^* the proxy and the true function touch: $\tilde{f}(\sigma_x^*, x) = f(\sigma_x^*, x)$. We chose ℓ_2^2 error to make $\tilde{f}$ a strongly-convex function with respect to σ. Since we will be using gradient descent for inference, a strongly-convex function can help us with faster convergence. The interesting aspect of this formulation is the simple structure of the gradient with respect to σ:

$$\nabla_\sigma \tilde{f}(\sigma, x) = \sigma - \sigma_x^* \tag{4.2}$$

The above gradient returns the negative direction and magnitude to improve a given σ on image x. To train a model, we learn to estimate $\nabla_\sigma \tilde{f}(\sigma, x)$ through regression using a deep neural network with groundtruth from Eq. 4.2. For inference, we rely on first-order convex optimization techniques.

Method	Inference	Inference with Constraint	Failure Detection	Require Aesthetic Func.
A2-RL [85]	Reinforcement Learning	✗	✗	✓
CCR [53]	Direct Regression	✗	✗	✓
EIC [160]	Grid Search	✓	✗	✓
MARS [87]	Direct Regression	✗	✗	✗
Ours	Gradient Descent	✓	✓	✗

Table 4.1: A comparison of aesthetic-based cropping algorithms.

Our approach can be viewed as a generalization of the previous work. If we had chosen an ℓ_1 error in Eq. 4.1 we would have arrived at the discretized reinforcement learning (RL) of Li et al. [85] as a special case. The gradient of an ℓ_1 error is the sign function, the possible values of which corresponds to the action space described in Li et al. [85]. Since we do not rely on reinforcement learning our results are easily reproducible. Furthermore, our approach is much faster in training (because we do simple regression) and inference (in Sec 4.5 we show that our model finds σ^* in fewer than 10 iterations).

Alternatively, our method can also be interpreted as a variation of the gradient boosting approach of Guo et al. [53], where we (re)use a single neural network for regression and run a variable number of iterations while incorporating cropping constraints. As a result, our method is faster in training (because it is not sequential), smaller in size (because we use a single model), and more flexible (because we can incorporate projection and a variable number of iterations without breaking the assumptions of the model). Furthermore, we *can automatically detect failures* (see Sec. 4.5). *None* of the previous work is able to detect failures automatically.

Table 4.1 and Tab. 4.2 show a comparison with other state-of-art methods for the general cropping problem. Since we are making specific assumptions about the optimal cropping (uniqueness and existence of quadratic upperbound), our method cannot be evaluated in the general aesthetic-based cropping domain. We also could not find any implementations of the previous work for comparison in our domain. For comparative evaluation, we will be comparing against optimized baselines of direct regression and landmark-alignment-based approaches.

Method	Inference	General Purpose	Sensitive to Scale	Multi-window Support
A2-RL [85]	Reinforcement Learning	✓	✗	✓
CCR [53]	Direct Regression	✓	✗	✗
EIC [160]	Grid Search	✓	✗	✓
MARS [87]	Direct Regression	✓	✗	✗
Ours	Gradient Descent	✗	✓	✗

Table 4.2: A comparison of aesthetic-based cropping algorithms.

Reference Cropping

We first crop the images into a perceptually-consistent reference window. Since all the portraits are consistent in the reference view, we can create arbitrarily desired cropping windows post-inference through an affine update of the cropping parameters (see appendices for more details). We define the reference cropping as a 120×160 image. Figure 4.3 shows the reference cropping window on input portraits. We chose this representation because it is large enough to contain the crucial information for perceptual cropping and small enough to hide away irrelevant patterns. We also experimented with 100×140 windows and achieved similar quantitative results. The exact window size of the reference view does not seem to matter as long as consistently-cropped training data is available.

Parametrization

We define the parameter space $\sigma = [\log(t_s), t_x, t_y]^{\mathrm{T}}$, where t_s is the scaling factor, and t_x, t_y are the top-left corner of the reference cropping window. Note that the reference cropping window has a fixed aspect ratio (100×140). Therefore, we can represent all the cropping configurations with three parameters. Since a neural network approximate of $\nabla_\sigma \tilde{f}(\sigma, x)$ is noisy, during the optimization we may walk into a region of the parameter space that is not feasible on the input image. Furthermore, the image may not have enough margin around the subject for the reference cropping window to be viable. In both of these cases, we need a projection step into the feasible σ set. The projection is implemented as a QP-program under a weighted ℓ_2 norm (see appendices). We use the MobileNetV2 [123] architecture to approximate $\nabla_\sigma \tilde{f}(\sigma, x)$.

Training

The data consists of $16{,}458$ training samples $\{(x_i, \sigma_i^*)\}$ provided by a photography studio (see Fig. 4.3). We split the dataset to $13{,}166$, 1645, and 1647 for train, validation, and test. To train the model on each iteration, we sample a cropping σ and use Eq. 4.2 to compute the groundtruth gradient vector. We then train the neural network in a supervised fashion using $(\mathrm{T}_\sigma(x), \nabla_\sigma \tilde{f}(\sigma, x))$. We use PyTorch [111], and Adam [72] on mean squared error (MSE) with learning rate 10^{-3} for optimization. We train for 2000 epochs with batch-size 128 in 4 hours on an RTX 2080Ti. We use the model with lowest validation error.

Inference

Now we have a neural network function that approximates $\nabla_\sigma \tilde{f}(\sigma, x)$. We can use this function to infer the optimal σ_x^*. For inference, we apply the projected gradient algorithm using our gradient estimator. In Sec. 4.5, we experiment with several initialization methods and learning-step strategies. We show that our formulation allows the detection of failures automatically. We terminate the procedure when the norm of the gradient mapping falls below a threshold τ (Def. 2.2.3 Nesterov [106]):

$$\frac{1}{\gamma_t}\|\sigma_t - \Pi_c[\sigma_t - \gamma_t \nabla_\sigma \tilde{f}(\sigma, x)]\|_2 < \tau, \tag{4.3}$$

where σ_t is the estimate at iteration t, Π_c is projection to the feasible set c derived from the feasibility constraints on the original image (see appendices), and γ_t is the step-size. When the optimal cropping window σ^* is feasible, this definition reduces to a threshold on the gradient norm.

Unlike previous work, we do not learn an explicit aesthetic function. We learn a gradient estimator of a proxy aesthetic function over the cropping parameter instead. This modelling assumption yields an efficient mechanism to find σ^* with first-order optimization techniques. In Sec. 4.5, we show that our method also converges quickly to a more accurate estimate than the baselines.

4.5 Evaluation

For evaluation we use the 1647 held-out test set for image cropping. Mean intersection over union (mIoU) and ℓ_2 distance are the common evaluation measures in image cropping. We use the ℓ_2 distance to the groundtruth to report our results. For initialization of inference in our experiments we (i) scale the image so that the smallest dimension matches a 120×160 window. This operation corresponds to initialization of $\sigma_0 = [\log(s), 0, 0]^{\mathrm{T}}$, where s is the downscaling factor (we call this simple initialization), and (ii) initialize σ_0 with uniformly random translation and a Gaussian random scale centred around $2 * s$ with a variance of $0.2 * s$ (we call this random initialization).

We experiment with three step-size strategies from the stochastic gradient descent literature: (i) constant step-size γ, (ii) decreasing step-size $\frac{\gamma}{\sqrt{t}}$, and (iii) $\frac{\gamma}{t}$, where t is the iteration and γ is a constant. Note that a constant step-size strategy will be sufficient with inexact gradient estimates from a theoretical standpoint. This experiment is meant to provide more insight into the behaviour of the model.

Figure 4.4 shows the results for $\gamma = 1$. The error is measured using the groundtruth values σ_i^*. Running the optimization from the same initial point, all three step-size strategies converge to a stationary point with the same error. The behaviour of the function described by our gradient estimator is consistent across the three paths. A constant step-size reaches close to the stationary point within five iterations on our test set on average.

Based on Fig. 4.4, the gradient-norm also drops below 1.0 whenever the optimization has converged, which indicates a natural termination criterion could be the norm of the estimated gradient. Note that a gradient norm of 1.0 indicates a sub-pixel translation update, which does not affect the output image because of spatial discretization. Both random initialization and simple initialization converge to the same average error. Although our approximation of $\nabla_\sigma \tilde{f}(\sigma, x)$ is noisy, the inference procedure makes progress in the correct direction on every iteration.

In Fig. 4.5 we examine the top-5 errors on the test set. The error appears to be the accumulation of groundtruth noise in the data. Based on the results, we use constant step-size $\gamma = 1$ and terminate the inference when the gradient mapping norm is less than one for the remainder of the experiments.

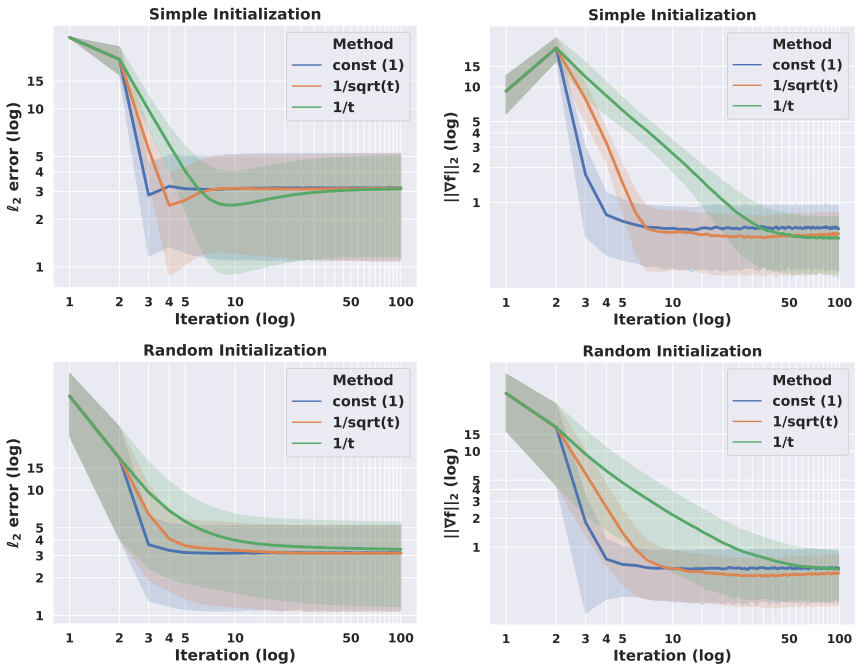

Figure 4.4: Cropping inference. Prediction error (left) and gradient norm (right) of our cropping algorithm using three step-size strategies with simple initialization (see text), and random initialization. The shaded area marks one standard deviation.

Method	Mean ℓ_2 Error
Landmark Alignment [67, 161]	11.71
DNN Regression	4.76
Ours	**3.10**

Table 4.3: Baseline Comparison. For DNN Regression we use the same MobileNetV2 [123] architecture.

Figure 4.5: Samples with the highest cropping error on the test set. The top row is the groundtruth, and the bottom row is the prediction of our model. Even though the groundtruth data is noisy, our model has learned to produce more consistent cropping than the data. Images are blurred for anonymization.

Table 4.3 compares the performance of our method with two baselines. Direct regression with the same network architecture yields a mean ℓ_2 error of $\mathbf{4.76}$, whereas our model reaches $\mathbf{3.10}$ mean error after only five iterations. A landmark alignment strategy where 68 facial key-points are aligned to the average location of groundtruth landmarks yields an error of $\mathbf{11.71}$.

To determine whether the model always converges to the same stationary point, we run 20 inferences with random initialization. For each image in the test set, we calculate the ℓ_2 distance between all 20 final predictions and take the maximum value. The average maximum prediction distance over the entire test set is 1.523. On average, all the predictions reach a stationary sphere with a radius of 0.76.

One of the current challenges in deploying neural-network-backed systems is automatic failure detection [35, 114, 130]. Our method allows automatic failure detection. To demonstrate this, we rotate half of the test set by randomly chosen $[90°, 180°, 270°]$ so that correct cropping may not be achievable. After 20 iterations, the *maximum* gradient norm of the correctly oriented images is 1.78, whereas the *minimum* gradient norm of the rotated images is 5.35 – we can detect failure in this

scenario with 100% accuracy. Running the inference longer does not reduce the gradient of the anomalous results. We detect the failure when the inference has failed to reach a stationary point after a fixed number of iterations.

4.6 Conclusion

We presented the perceptual cropping component of AutoPortrait. We reviewed the deficiencies of the previous work and presented a novel method to address those weaknesses. Our method generates perceptually consistent cropping of portraits within a few iterations and detects failures automatically. During the past year, our cropping method has been used successfully in the production pipeline of more than $500{,}000$ professional portrait photos.

Chapter 5

AutoPortrait: Automatic Portrait Enhancement from Studios to Mobile Phones – Portrait Retouching

In this chapter, we go through the face retouching problem and present our solution. Face retouching is one of the most time-consuming steps in professional photography pipelines. The existing automated approaches blindly smooth the skin, destroying the delicate texture of the face. We present the first automatic face retouching approach that produces high-quality professional-grade results in less than two seconds. Unlike previous work, we show that our method preserves textures and distinctive features while retouching the skin. We demonstrate that our trained models generalize across datasets and are suitable for low-resolution cellphone images. Finally, we release the first large-scale, professionally retouched dataset with our baseline to encourage further work on the presented problem.

5.1 Related Work

To the best of our knowledge, very little prior work exists for face retouching. Lin et al. [93] present an exemplar-based *freckle* retouching technique using a

small dataset of cosmetic laser therapy. In the commercial domain, there is Visage Lab, BeautyPlus, Meitu, and Facetune, to mention a few. All of the available products provide face retouching and beautification tools, but none of them are fully automatic. Furthermore, none of these tools can provide high-quality, high-resolution face retouching even when all the parameters are carefully tuned. The existing methods apply "blind" smoothing, where moles and freckles disappear, and fine skin-texture is destroyed (see Fig. 5.3). Our method preserves the identifying features of skin while removing imperfections at a high-resolution. We also release the first large-scale retouching dataset with our baseline to encourage further work on this problem (see Fig. 5.1).

A natural starting point for skin texture synthesis during retouching is through generative adversarial networks (GANs) [49]. GANs are a broad class of generative models within which two neural networks, a generator and a discriminator, compete against each other. During the training, the generator G produces samples from a *fake* distribution $\tilde{\mathcal{D}}$ while the discriminator D tries to detect whether the incoming samples are from the target distribution $\mathcal{D}$, i.e., , if the image is *real* or *fake*. The objective of GANs is to learn a generator G such that the *real* distribution and the *fake* distribution are approximately equal, i.e., , $\tilde{\mathcal{D}} \approx \mathcal{D}$. Over the previous years, the research community has focused on improving the training time, stability, quality, and analysis of samples generated by GANs [12, 18, 51], while simultaneously identifying new applications [60, 66, 136].

One of the major developments based on GANs is the image-to-image translation method Pix2Pix [60]. Pix2Pix uses a conditional GAN loss plus ℓ_1 loss to learn a mapping $x \rightarrow y$. The authors use a U-Net [119] generator and a PatchGAN [84] discriminator. The PatchGAN discriminator model assumes independence between pixels separated by more than the patch window width. The authors present several use cases of Pix2Pix applications with impressive results. Our approach is similar to Pix2Pix [60] in the way that we learn a mapping function from original images to retouched images. However, our (i) network architectures, (ii) training loss function, and (iii) data augmentation schemes are different and demonstrably necessary to perform professional face retouching. We will further discuss the differences in Sec. 5.3 and will provide comparisons in Sec. 5.4.

5.2 Flickr-Faces-HQ-Retouching (FFHQR) Dataset

Before discussing our method, we first introduce a new face retouching dataset based on the Flickr-Faces-HQ (FFHQ) dataset [66]. The original FFHQ dataset consists of $70{,}000$ $1\,\mathrm{MP}$ face-aligned images that are collected from Flickr. We chose FFHQ as the basis of our new dataset because of the variety of ages, ethnicity, lighting conditions, and the large number of images that could benefit from face retouching. To create the new dataset, we hired a team of professional image editors to retouch the images. See Fig. 5.1 for samples of our dataset. To the best of our knowledge, FFHQR is the first publicly available retouching dataset.

One of the key features of professional retouching is that the image updates are sparse – most of the pixels do not change. This attribute is unlike the result of commonly used blurring in the commercial applications where the entire skin is smoothed to remove blemishes. The FFHQR dataset reflects this professional retouching style. Figure 5.2 shows a scatter plot of the percentage of pixel change versus the mean absolute value of pixel updates (range $[0,\ 255]$). In the majority of images, less than 40% of the pixels are edited. Furthermore, most pixel value changes are in the $[100,\ 200]$ range. This observation has implications in model design: the retouching methods should facilitate learning a function that preserves the input values strictly.

We also experiment with studio data and measure cross-dataset performance. We present more information about the studio in the next section. The FFHQR dataset and the evaluation scripts are available at https://github.com/skylab-tech/ffhqr-dataset.

5.3 Texture-preserving Portrait Retouching

The final step of our AutoPortrait pipeline is skin retouching. Professionally retouched images must retain moles, freckles, and other distinctive features. Furthermore, the output must preserve the colour tone of the original image (colour-correction is a separate task in professional photography). The model must distinguish the features of the image that must be retained and the features that must be removed. Skin smoothing, which is the common algorithm in existing applications, will not preserve the identifying features. Furthermore, a blind smoothing alters

Figure 5.1: Three samples from the Flickr-Faces-HQ-Retouching (FFHQR) Dataset. The left image is the original image from FFHQ [66], and the right image is the retouched version in our dataset. The figure is best viewed on a screen.

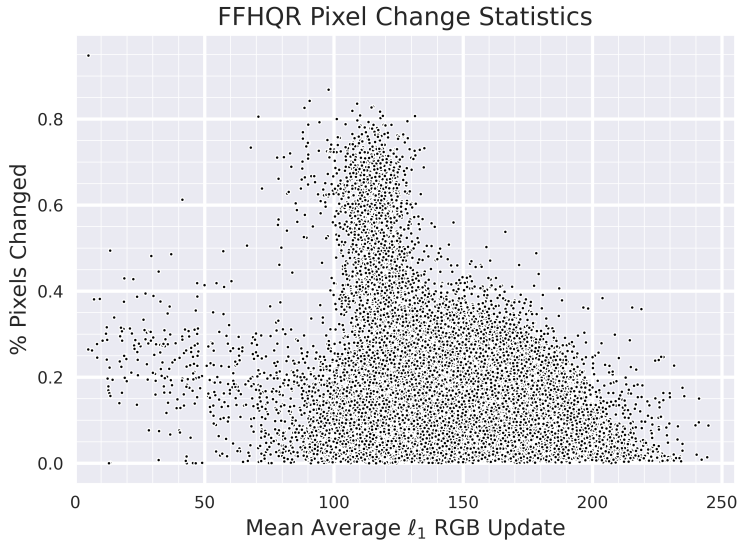

Figure 5.2: The scatter plot of FFHQR pixel update statistics. The y-axis is the percent of updated pixels per image. The x-axis is the mean absolute value of pixel updates. For the majority of images, less than 40% of pixels change.

the delicate texture of the original skin (see Fig. 5.3). Therefore, a desirable and practical retouching model needs to:

1. Preserve the image structure.
2. Preserve the image texture while also performing skin retouching.
3. Generalize to a variety of skin tones and lightings.

To address these concerns, we carefully build a fully convolutional neural network that filters the input image. For (2), a safe strategy is to train models that only retouch known imperfections. Since we need to process high-resolution images, even the slightest error will be visible. We empirically verify that our final model only removes the known skin imperfections and preserves everything else. Furthermore, we show that our method performs automatic face retouching while preserving the natural quality of the image. Interestingly, we present empirical results that indicate that our final model produces a more aesthetically pleasant skin retouching than the training data.

Figure 5.3: Automatic skin retouching. The left image is the original image, the middle image is the AI-assisted retouching of BeautyPlus, the right image is the automatically retouched image by our method.

Architecture

In our experiments, we observed that downscaling the image tensors to a factor smaller than half leads to slight pixel displacement and loss of the original texture in the output space. This observation eliminates most of the typical architectural choices such as U-Net [119] in Pix2Pix [60] since there are usually several levels of down-sampling in the architecture. We limited the downscaling in our network architecture search to preserve fine details.

Furthermore, we observed that incorporating additive skip-connection shortcuts decreases the random jitters in the output. Since we expect the majority of the pixels to remain unchanged, the skip connections also allow for more direct transfer of the input to the output.

Since we do not perform much downscaling, the depth of the architecture and the number of kernels would typically become limited by the available GPU memory.

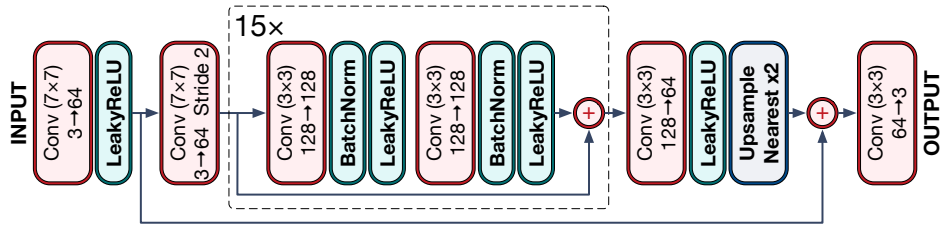

Figure 5.4: Our retouching network architecture.

However, face retouching is mostly a local operation – the correction of any pixel does not depend on far away pixels. This feature is unlike typical Pix2Pix-like applications, where the network takes the global context of the input image into account to generate the output. This problem structure enables a sliding-window strategy that we exploit with convolutional architectures [96]. Without loss of generality, we utilize more kernels and deeper architectures by limiting the input to a large-enough $w \times w$ sub-window.

Given that we limit our network's receptive field, the resulting models are not statistically dependent on the *entire image* – this provides more robustness to out-of-distribution test images, which enables a broader application than just the training image distribution. Combined with our data augmentation schemes, we show that our models can generalize across datasets with substantial domain shift. See Fig. 5.4 for our final generator architecture. We use the same discriminator architecture as Pix2Pix [60].

Training

The loss function that we use for training consists of (i) Relativistic Average GAN loss [49, 63], (ii) perceptual loss [61], and (iii) direct mean squared-error (MSE):

$$\mathcal{L}_{\mathcal{G}} = \alpha \cdot \mathcal{L}^{\mathcal{G}}_{\text{ragan}} + \beta \cdot \mathcal{L}_{\text{perceptual}} + \gamma \cdot \mathcal{L}_{\text{mse}} \tag{5.1}$$

We set $\alpha = 10^{-3}, \beta = 6 \cdot 10^{-3}, \gamma = 0.5$ in our experiments. For the Relativistic Average GAN, we use a running estimate of 300 previous predictions. For the perceptual loss, we use mean squared-error on the output features of the 14th layer

in the 19-layer VGG model [138].

We experimented with the original GAN [49] as well as other variants such as WGAN [12]. Overall, we were able to achieve the best results using RAGAN [63]. We also experimented with ℓ_1 loss, but we found that the model with mean-squared-error loss converges much faster. Similar to Pix2Pix [60] we combine traditional loss functions with GAN-type loss functions. However, we (i) specifically use RAGAN, we (ii) also include a *perceptual loss* function, and (iii) use mean-squared-error instead of ℓ_1 loss. In Sec. 5.4, we do an ablation study to show how much each term of the loss function contributes to the output retouching quality. Our results indicate that each term contributes to the overall performance, and the synergy of the loss functions creates a strong baseline for automated professional face retouching. We show that our final model yields retouching results that are qualitatively better than the groundtruth data.

We repeatedly perform one gradient descent step on D and one gradient descent step on G to train our models. We train the studio data model for approximately two weeks on three 2080 Ti GPUs and one 1080 Ti. We train a model on the FFHQR dataset in just five days. We run the training for 500 epochs and set $w = 150\,\mathrm{px}$. We set the batch-size to 75, the learning rate of Adam [72] to $2 \cdot 10^{-4}$ for our model and 10^{-4} for RAGAN discriminator.

Data Preprocessing

We use before-after professionally retouched image pairs for the training. We sample $w \times w$ image patches from the training data and perform mirror augmentation and colour perturbation during the training. To encourage scale-invariance, we randomly downscale the images before sampling. Since there are quadratically more $w \times w$ windows in larger images than the spatially smaller ones, uniformly random scaling under-represents the patches in the larger images. To fix the under-representation, we first take a random scaling factor $s \sim \mathrm{Unif}(s_{\min}, 1)$, where $s_{\min}$ is the smallest acceptable image scale cubed, and downscale the image to $s^{\frac{1}{3}}$. This change of variable ensures all windows in all image scales are equally likely (see appendices).

We evaluate our model using both the newly presented FFHQR dataset and also studio data. The FFHQR data consists of $70{,}000$ image pairs of original and

retouched images. We use $56{,}000$ images for training, 7000 images for validation, and 7000 images for testing. The studio data contains $203{,}725$ image pairs of original and retouched images. We split the studio data into $158{,}683$, $22{,}409$, and $22{,}633$ for train, validation, and test. Following the same procedure as Karras et al. [66] (FFHQ), we pre-process the studio data by cropping the images to the head. While FFHQR is free-form in-the-wild photos of faces, the studio data is captured in strictly controlled, well-lit conditions with DSLR cameras.

The code to reproduce our results is publicly available at https://github.com/skylab-tech/autoRetouch.

5.4 Evaluation

Figure 5.5 shows sample outputs of the AutoPortrait pipeline.

Portrait Retouching

Portrait retouching evaluation presents a similar set of challenges as the super-resolution (SR) literature. While the assessments in the SR literature primarily rely on reproducible measures such as PSNR and SSIM [151], Blau et al. [19] show that none of these metrics translate into a *better perceptual result* beyond a certain accuracy. In face retouching, this issue is more severe. Since professional retouching mostly produces sparse image updates, the before and the after images are already very similar before any processing. As a result, both of these metrics yield very high values, with only a slight variation across experiments. Our evaluation results on face retouching also confirm the findings of Blau et al. [19]: neither PSNR nor SSIM are reliable performance indicators for high-quality face retouching methods.

An alternative assessment in the SR literature is the subjective mean-opinion-score (MOS), where human subjects compare the perceptual quality of images generated with multiple algorithms. The MOS score is generally regarded as not reproducible and expensive to obtain. However, this approach can perceptually validate and compare different methods to produce reliable rankings.

To the best of our knowledge, there is no scalable and objective measure of quality for our application at the time of writing. We provide quantitative PSNR and SSIM comparisons as well as user-oriented studies. To perform the user study, we

Figure 5.5: Sample end-to-end results from AutoPortrait.

Figure 5.6: Outputs of Pix2Pix, FFHQR model, and the studio model on the FFHQR test set. The studio model generalizes well to FFHQR data even though there is a considerable domain shift. The Pix2Pix results often contain artifacts.

present professional retouchers with two methods' outputs and ask them to either choose the best output or skip if the images are too similar. We capture their choices and the time that it takes to make a decision. See appendices for more details of our setup.

For evaluation, we use a single RTX 2080Ti GPU. For Pix2Pix [60], we use the provided code and the recommended configuration by the authors. Figure 5.6 shows the output of our method when trained on FFHQR and the studio data. The studio model is only trained with the controlled studio data, whereas the FFHQR model is directly trained on FFHQR. Both of our models perform retouching well. However, Pix2Pix [60] fails to perform retouching and degrades the image. Figure 5.7 shows the test samples of retouched image patches based on our trained model on the studio data. Our model consistently preserves the moles and other identifying features while removing imperfections.

Figure 5.7: Sample outputs of our retouching model. The figure is best viewed on a screen.

Figure 5.8: The output of our model versus the groundtruth retouching. The left column is the input, the middle column is groundtruth retouching, and the right column is our output. Our model preserves the fine details more than the groundtruth. See appendices for more images.

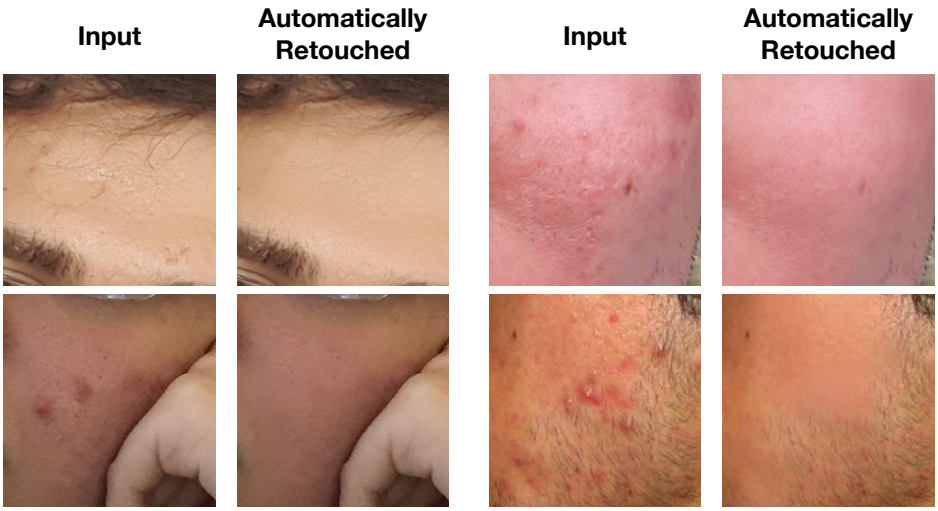

Figure 5.9: Sample input/output of retouched images captured with cellphones. The figure is best viewed on a screen. See appendices for more images.

Figure 5.8 compares the retouching output of our model with the groundtruth patches on the studio data. Notice that our model preserves the fine texture better than the groundtruth images. We suspect this happens because, in real-life retouching, the professionals may use large brushes for correction that inadvertently affects areas of the image that do not require retouching. Our model, however, operates at the pixel level and can preserve details at no extra cost. Furthermore, in Sec. 5.4, we perform ablation studies on the effect of each term in the loss function. We observe that adding the RAGAN loss term encourages the model to preserve the input as much as possible. The preservation of the details produces retouching models that produce better images than the groundtruth retouching data. We observe the same trends in our user study.

We also test our studio retouching model on lower-resolution smartphone camera images. The images are captured with iPhone 6 or iPhone X. Figure 5.9 shows sample outputs from our tests. Although the studio model is not trained on cellphone images, it generalizes to lower-resolution and noisy images.

In Fig. 5.10, we test if the retouching network corrects unfamiliar patterns in the image. We manually add brush strokes of different colours and brush sizes with

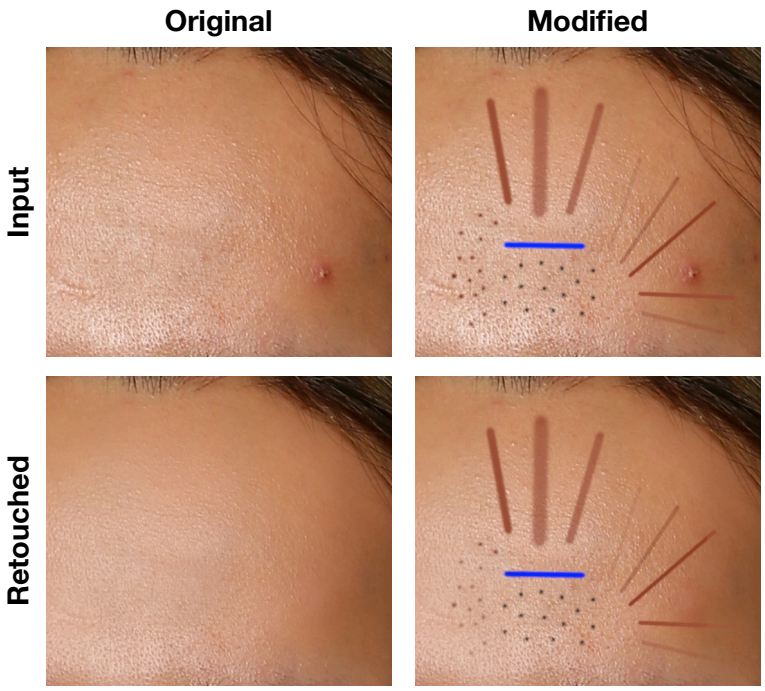

Figure 5.10: Our retouching model does not change unfamiliar patterns in the image. The model appears to be only responding to known skin imperfections. This behaviour is desirable in professional retouching to ensure distinctive features are preserved.

varying opacity to an image and process it using our model. Fig. 5.10 shows that our model did not change the added brush strokes. This behaviour is a desirable feature in professional face retouching since we wish to preserve the distinctive features as much as possible – it is safer for the model to retouch only known imperfections that require correction.

Figure 5.11 shows some of the failure cases of our retouching model. The images that our model fails to correct usually contain severe skin blemishes. Although our model improves the output image, it does not eliminate the blemishes.

Ablation Study

We perform a qualitative and quantitative ablation study to compare the effect of each term in Eq. 5.1. We run the following experiments on the loss function $\mathcal{L}$:

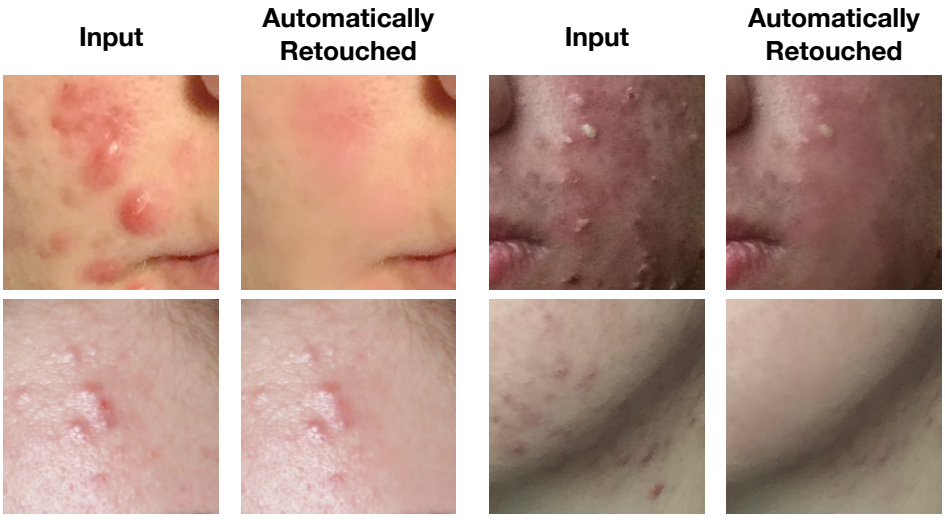

Figure 5.11: Failure cases. Our retouching model fails when blemishes are severe.

1. Only mean squared error (MSE) ($\alpha = \beta = 0$).
2. MSE and Perceptual ($\alpha = 0$).
3. MSE, Perceptual, and RAGAN.

For this evaluation, we run the experiment on a subset of 7905, 950, and 988 studio images for train, validation, and test, respectively. On FFHQR, we train on the entire training data. We run the training for 300 epochs and leave the other parameters as the original experiment. The training of each configuration takes 19 hours on studio data and three days on FFHQR. For each configuration, we use the model with the lowest validation loss.

Table 5.1 shows the quantitative results. Since the before and after images are almost identical (except for the sparsely retouched regions), both PSNR and SSIM metrics will be too high. While in the super-resolution literature the best PSNR ≈ 25, and SSIM ≈ 0.75 [97], the "no edit" baseline already achieves PSNR $= 45$ and SSIM $= 0.99$ on FFHQR. Except for one case, the numerical results indicate that adding more terms to the mean-squared error (MSE) hurts the performance in terms of PSNR and SSIM. However, our user study reveals that the addition of perceptual loss and RAGAN improves the quality of retouching by a large margin

		Studio Data		FFHQR	
		PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
	No edit (input → output)	39.12	0.9807	45.58	0.9938
(1)	MSE	39.58	0.9858	40.53	0.9876
	MSE + Perc.	39.41	**0.9875**	39.04	0.9889
	MSE + Perc. + RAGAN	**41.04**	0.9865	44.82	0.9944
	Pix2Pix [60]	28.12	0.8893	29.58	0.9224
(2)	MSE	40.01	0.9844	**45.88**	0.9949
	MSE + Perc.	39.88	0.9851	45.00	**0.9952**
	MSE + Perc. + RAGAN	39.65	0.9849	44.92	**0.9952**
	Pix2Pix [60]	30.29	0.9152	33.45	0.9585

Table 5.1: Quantitative results using PSNR and SSIM. (1) is trained on studio data, (2) is trained on FFHQR.

			Voted 1 ↑		Voted 2 ↑		Undecided		
#	Method 1	Method 2	%	Time (s)	%	Time (s)	%	Time (s)	n
(1)	Ours	MSE	**50.1%**	28.0	17.9%	24.0	32.0%	26.3	730
	Ours	MSE+Perc.	**28.3%**	30.9	22.9%	30.3	48.8%	26.1	791
	Ours	Pix2Pix [60]	**100.0%**	1.9	0%	-	0%	-	508
(2)	Ours	MSE	**94.6%**	12.2	4.4%	19.5	1.0%	19.6	802
	Ours	MSE+Perc.	**95.7%**	9.8	3.4%	17.5	0.9%	15.7	898
	Ours	Pix2Pix [60]	**99.9%**	1.3	0.1%	0.9	0%	-	1008
	Ours	Groundtruth	**76.3%**	8.4	18.7%	12.0	5.0%	19.0	672

Table 5.2: Human perceptual test. In each evaluation, we asked the subjects to chose their favourite method between two options. The participants could skip if the images were too similar. We measure how often a method is chosen and how long it took to make a decision. The time column shows the median time. (1) Models are trained and tested on FFHQR; (2) models are trained and tested on studio data.

(see Tab. 5.2 and Fig. 5.12). This observation is consistent with Blau et al. [19] – a higher PSNR or SSIM does not necessarily translate into higher perceptual quality.

Figure 5.12 shows the qualitative results of our ablation study. While MSE yields impressive results in retouching, it is easy to see the output of this network is smoothed, and sometimes fine textures (e.g., freckles) are removed. Although

Figure 5.12: The effect of loss function on retouching. Using MSE loss alone leads to smooth images, which is improved by adding a perceptual loss. However, a perceptual loss still does not preserve fine details. Adding an adversarial loss encourages the network to make as few changes on the image as possible. The figure is best viewed on a screen.

the images may appear pleasant at a low resolution, they do not possess the level of quality that is essential to professional high-resolution production. Adding perceptual loss improves the output by sharpening the image, but the network still does not preserve the input's delicate texture at all times. When we add the RAGAN loss, the network (seemingly) learns to minimize the output space changes, which results in minimal retouching that preserves the input details as much as possible. This quality leads to a better retouching than the groundtruth data (see Fig. 5.8).

Table 5.2 shows a summary of our user study results. We present the users with randomly ordered images of two test methods and ask them to pick the best photo. See appendices for a complete description of the setup. We run this experiment on both studio data and FFHQR, comparing against Pix2Pix [60] and models produced by our ablation study. Our proposed loss combination is favoured over the alternative more frequently across all the experiments. The preference gap between our method and the two alternatives is much larger on the studio data than for the FFHQR dataset. Furthermore, it took the users much longer to pick the best output on FFHQR than studio data. This gap is due to the higher quality imaging of the studio data compared to the in-the-wild FFHQR. While our proposed method still outperforms the alternatives, the difference is much more evident in the higher quality photos.

We also compare the output of our method with the groundtruth on the studio data. Since the studio images contain more details in perfect lighting, it is much easier to compare and perceive the difference. The skin-quality of our method is more frequently favored over the groundtruth data. Pix2Pix is seldom chosen, mainly because of the visible artifacts in the image. It took the users an average of two seconds to determine which output is better on both of the experiments that involved Pix2Pix models. Our proposed method perceptually outperforms alternative design choices despite having a slightly lower PSNR and SSIM.

We also have validated our automatic retouching model in real-world applications. Over the past year, our workflow has assisted professional retouching of over $1{,}000{,}000$ portraits. Typical failures that we have encountered ($< 2\%$) are when the image patches require a substantial amount of change, or the skin blemish is atypical. This usually happens when an extended region of the skin is covered in hard blemishes, and the natural colour of the skin is no longer apparent (see

Fig. 5.11). In these cases, the model removes the blemishes most often, but the area remains slightly red.

Our retouching model is only *20 MB* and requires *2580 MB* GPU RAM to process an input image of 1024×1024 in *1.1* s on a Titan RTX 2080Ti, or *1.9* s on a 1080Ti. The processing window size could be optimized according to the available resources down to a $w \times w$ sub-window at a time, while still producing an identical image. In contrast, the Pix2Pix [60] model is *208 MB* and occupies *1340 MB* memory while processing each image in *1.82* s on a 1080Ti.

5.5 Conclusion

We presented the AutoPortrait pipeline for automatic portrait enhancement. Our pipeline performs perceptual cropping, subjective colour correction, and face retouching. We explained the subtleties of each subproblem, discussed the shortcomings of the previous work, and introduced our novel solutions to address the specific challenges. Furthermore, we compared our cropping and retouching models with several baselines and highlighted the improvements. We release the first large-scale retouching dataset FFHQR and invite the community to improve upon the presented work.

Chapter 6

Conclusions and Future Work

Future Work

The research on synthetic data has been a fruitful path that has progressed much farther than our early work presented in Chapter 2. One aspect of synthetic data that might be worth investigating for future work is the possibility of controlled modification of rendering quality and dataset size while measuring the models' performance. The results could help us understand the factors that affect the predictions and the generalization power of DNNs.

The problem of making reliable predictions with DNNs that we presented in Chapter 3 is potentially one of the most exciting research directions in machine learning. Achieving high accuracy on OD-test will be an interesting challenge for the community to tackle. We outlined several directions for future work on this problem in Chapter 3. One simple additional problem that we can formulate for further work is this:

Is it possible to build a `cat` *vs.* `not-cat` *classifier?*

Note that this problem is different from `cat` vs. `dog` because both `cat` and `dog` are not as heavy-tailed in distribution as the class of `non-cats`. While a finite set of samples might roughly represent the class of `cats` or `dogs`, a finite set of samples for `non-cats` is unlikely to represent its class well. Therefore, a direct

supervised learning approach to this problem may not yield the desired results with proper prediction guarantees.

For future work of the research on automatic portrait editing that we presented in Chapter 4 and Chapter 5, we believe the following areas could be improved: (i) creating a more reliable/scalable evaluation metric for retouching, (ii) designing more accurate retouching models with even better generalization properties, (iii) designing models with smaller memory footprint, (iv) and building faster models.

Conclusion

In the first part of our work, we looked at two significant challenges that arise in the application of DNNs: data scarcity and prediction reliability. We showed that using photorealistic synthetic data to train computer vision models can be a viable strategy to address data scarcity. We then examined the prediction reliability of DNNs and showed that DNNs might be too fragile for sensitive applications. We presented OD-test, a less biased evaluation framework for out-of-distribution sample detectors. We evaluated methods from various domains and concluded that none could reliably certify a prediction from large-dimensional input spaces.

In the second part, we introduced AutoPortrait: a fully automated pipeline for colour-correction, portrait cropping, and portrait retouching. We presented the perceptual head cropping problem and discussed our solution. Furthermore, we showed that our retouching results exceed the quality of the training data. We showed that our models generalize to low-resolution images. Finally, we presented the first large-scale retouching dataset.

Final Words

I hope you enjoyed reading my dissertation as much as I enjoyed working on it.

Bibliography

[1] Grand Theft Auto V. URL https://www.rockstargames.com/V/. → pages xi, 12

[2] Half-Life 2. URL https://half-life.com/en/halflife2. → page 9

[3] Synthetic Datasets for ADAS and Autonomous Driving, . URL https://www.cognata.com/datasets/. → page 30

[4] Apollo Game Engine, . URL https://apollo.auto/gamesim.html.

[5] Precise Synthetic Image and LiDAR (PreSIL) Dataset for Autonomous Vehicle Perception, . URL https://uwaterloo.ca/waterloo-intelligent-systems-engineering-lab/projects/precise-synthetic-image-and-lidar-presil-dataset-autonomous.

[6] CVEDIA, . URL https://www.cvedia.com/.

[7] Deep Vision Data, . URL https://syntheticdatatraining.com/.

[8] Data Gen, . URL https://www.datagen.tech/.

[9] The Hive, . URL https://thehive.ai/.

[10] Lexset AI, . URL https://www.lexset.ai/. → page 30

[11] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *TPAMI*, 2012. → pages xiii, 11, 26

[12] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017. → pages 68, 74

[13] M. Aubry and B. C. Russell. Understanding Deep Features with Computer-generated Imagery. In *ICCV*, 2015. → page 9

[14] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3D Chairs: Exemplar Part-based 2D-3D Alignment using a Large Dataset of CAD Models. In *CVPR*, 2014. → page 9

[15] P. L. Bartlett and M. H. Wegkamp. Classification with a Reject Option using a Hinge Loss. *JMLR*, 2008. → page 35

[16] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. What's the Point: Semantic Segmentation with Point Supervision. In *ECCV*, 2016. → page 10

[17] A. Bendale and T. E. Boult. Towards Open Set Deep Networks. In *CVPR*, 2016. → pages 34, 37

[18] D. Berthelot, T. Schumm, and L. Metz. BeGAN: Boundary Equilibrium Generative Adversarial Networks. *arXiv preprint arXiv:1703.10717*, 2017. → page 68

[19] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor. 2018 PIRM Challenge on Perceptual Image Super-resolution. In *ECCV Workshops*, 2018. → pages 75, 83

[20] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic Object Classes in Video: A High-definition Ground Truth Database. *Pattern Recognition Letters*, 2009. → pages xi, 8, 9, 12, 13, 14, 15

[21] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, and Others. Language Models are Few-shot Learners. *arXiv preprint arXiv:2005.14165*, 2020. → pages 1, 2, 3

[22] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A Naturalistic Open Source Movie for Optical Flow Evaluation. In *ECCV*, 2012. → page 10

[23] J. Chen, S. Paris, and F. Durand. Real-time Edge-aware Image Processing with the Bilateral Grid. In *ACM Transactions on Graphics (TOG)*, 2007. → page 114

[24] J. Chen, A. Adams, N. Wadhwa, and S. W. Hasinoff. Bilateral Guided Upsampling. *ACM Transactions on Graphics (TOG)*, 2016. → page 114

[25] L.-Q. Chen, X. Xie, X. Fan, W.-Y. Ma, H.-J. Zhang, and H.-Q. Zhou. A Visual Attention Model for Adapting Images on Small Displays. *Multimedia systems*, 9(4):353–364, 2003. → page 56

[26] G. Ciocca, C. Cusano, F. Gasparini, and R. Schettini. Self-adaptive Image Cropping for Small Displays. *IEEE Transactions on Consumer Electronics*, 53(4):1622–1627, 2007. → page 56

[27] A. Coates, A. Ng, and H. Lee. An Analysis of Single-layer Networks in Unsupervised Feature Learning. In *AISTATS*, 2011. → page 105

[28] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *CVPR*, 2016. → pages xi, xiii, 8, 9, 12, 15, 16, 26

[29] C. Cortes and V. Vapnik. Support-vector Networks. *Machine learning*, 20 (3):273–297, 1995. → pages 44, 105

[30] C. Cortes, G. DeSalvo, and M. Mohri. Learning with Rejection. In *International Conference on Algorithmic Learning Theory (ALT)*, 2016. → page 35

[31] C. Cortes, G. DeSalvo, M. Mohri, and S. Yang. On-line Learning with Abstention. *ArXiv e-prints*, 2017. → page 35

[32] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005. → page 9

[33] L. Deecke, R. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft. Anomaly Detection with Generative Adversarial Networks, 2018. → page 38

[34] Y. Deng, C. C. Loy, and X. Tang. Image Aesthetic Assessment: An Experimental Survey. *IEEE Signal Processing Magazine*, 34(4):80–106, 2017. → page 56

[35] A. R. Dhamija, M. Günther, and T. Boult. Reducing Network Agnostophobia. In *NIPS*, 2018. → page 65

[36] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *ICML*, 2014. → page 11

[37] S. Edunov, M. Ott, M. Auli, and D. Grangier. Understanding Back-translation at Scale. *arXiv preprint arXiv:1808.09381*, 2018. → page 1

[38] D. Eigen and R. Fergus. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. In *ICCV*, 2015. → page 11

[39] K. Fukushima and S. Miyake. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Visual Pattern Recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982. → page 1

[40] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *CVPR*, 2016. → page 11

[41] Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *ICML*, 2016. → page 35

[42] Y. Gal and Z. Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *NIPS*, 2016. → page 35

[43] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-Adversarial Training of Neural Networks. *JMLR*, 2016. → page 11

[44] Y. Geifman and R. El-Yaniv. Selective Classification for Deep Neural Networks. In *NIPS*. 2017. → page 37

[45] M. Gharbi, Y. Shih, G. Chaurasia, J. Ragan-Kelley, S. Paris, and F. Durand. Transform Recipes for Efficient Cloud Photo Enhancement. *ACM Transactions on Graphics (TOG)*, 2015. → page 114

[46] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics (TOG)*, 2016. → page 1

[47] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand. Deep Bilateral Learning for Real-time Image Enhancement. *ACM Transactions on Graphics (TOG)*, 2017. → page 114

[48] M. Goldstein and S. Uchida. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PloS one*, 11(4), 2016. → pages 36, 114

[49] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *NIPS*, 2014. → pages 38, 68, 73, 74

[50] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, 2015. → page 107

[51] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved Training of Wasserstein GANs. In *NIPS*, 2017. → page 68

[52] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks. *ICML*, 2017. → page 37

[53] G. Guo, H. Wang, C. Shen, Y. Yan, and H.-Y. M. Liao. Automatic Image Cropping for Visual Aesthetic Enhancement using Deep Neural Networks and Cascaded Regression. *IEEE Transactions on Multimedia*, 20(8): 2073–2085, 2018. → pages 56, 60, 61

[54] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. → pages xiv, 32, 45

[55] D. Hendrycks and K. Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *ICLR*, 2017. → pages 34, 37, 38, 43, 105

[56] D. Hendrycks, M. Mazeika, and T. Dietterich. Deep Anomaly Detection with Outlier Exposure. *ICLR*, 2019. → page 33

[57] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely Connected Convolutional Networks. In *CVPR*, 2017. → pages xiv, 32

[58] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level Accuracy with 50x Fewer Parameters and $<$ 0.5 MB Model Size. *ArXiv e-prints*, 2016. → pages xiv, 32

[59] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (TOG)*, 2016. → pages xiii, 27, 29

[60] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image Translation with Conditional Adversarial Networks. In *CVPR*, 2017. → pages 68, 72, 73, 74, 77, 83, 85, 86, 116

[61] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual Losses for Real-time Style Transfer and Super-resolution. In *ECCV*, 2016. → page 73

[62] J. Johnson, A. Karpathy, and L. Fei-Fei. DenseCap: Fully Convolutional Localization Networks for Dense Captioning. In *CVPR*, 2016. → pages xiii, 27, 28

[63] A. Jolicoeur-Martineau. The Relativistic Discriminator: a Key Element Missing from Standard GAN. *ICLR*, 2019. → pages 73, 74

[64] Y. Ju, F. Zhao, S. Chen, B. Zheng, X. Yang, and Y. Liu. Technical Report on Conversational Question Answering. *arXiv preprint arXiv:1909.10772*, 2019. → page 1

[65] B. Kaneva, A. Torralba, and W. T. Freeman. Evaluation of Image Features using a Photorealistic Virtual World. In *ICCV*, 2011. → page 10

[66] T. Karras, S. Laine, and T. Aila. A style-based Generator Architecture for Generative Adversarial Networks. In *CVPR*, 2019. → pages xv, 68, 69, 70, 75

[67] V. Kazemi and J. Sullivan. One Millisecond Face Alignment with An Ensemble of Regression Trees. In *CVPR*, 2014. → pages 57, 64

[68] Y. Ke, X. Tang, and F. Jing. The Design of High-level Features for Photo Quality Assessment. In *CVPR*, 2006. → page 56

[69] A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *NIPS*, 2017. → pages 35, 106

[70] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding. *ArXiv e-prints*, 2015. → pages 13, 14

[71] B. Kim, J. Ponce, and B. Ham. Deformable Kernel Networks for Joint Image Filtering. *ArXiv e-prints*, 2018. → page 114

[72] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint 1412.6980*, 2014. → pages 62, 74

[73] D. P. Kingma and M. Welling. Auto-encoding Variational Bayes. In *ICLR*, 2014. → page 45

[74] M. Kliger and S. Fleishman. Novelty Detection with GAN. *ArXiv e-prints*, 2018. → page 38

[75] A. Krizhevsky and G. Hinton. Learning Multiple Layers of Features from Tiny Images. 2009. → pages 38, 105

[76] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012. → pages xiv, 1, 32

[77] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *NIPS*, 2017. → pages 34, 35, 106, 107

[78] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998. → pages 1, 38, 105

[79] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and Others. Photo-realistic Single Image Super-resolution using a Generative Adversarial Network. In *CVPR*, 2017. → page 1

[80] K. Lee, H. Lee, K. Lee, and J. Shin. Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples. In *ICLR*, 2018. → page 38

[81] K. Lee, K. Lee, H. Lee, and J. Shin. A Simple Unified Framework for Detecting Out-of-distribution Samples and Adversarial Attacks. In *NeurIPS*, 2018. → page 38

[82] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, and Others. Retrieval-augmented Generation for Knowledge-intensive NLP Tasks. *arXiv preprint arXiv:2005.11401*, 2020. → page 1

[83] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He. Depth and Surface Normal Estimation from Monocular Images using Regression on Deep Features and Hierarchical CRFs. In *CVPR*, 2015. → page 11

[84] C. Li and M. Wand. Precomputed Real-time Texture Synthesis with Markovian Generative Adversarial Networks. In *ECCV*, 2016. → page 68

[85] D. Li, H. Wu, J. Zhang, and K. Huang. A2-RL: Aesthetics Aware Reinforcement Learning for Image Cropping. In *CVPR*, 2018. → pages 56, 57, 60, 61

[86] D. Li, H. Wu, J. Zhang, and K. Huang. Fast A3RL: Aesthetics-Aware Adversarial Reinforcement Learning for Image Cropping. *IEEE Transactions on Image Processing*, 28(10):5105–5120, 2019. → page 59

[87] D. Li, J. Zhang, and K. Huang. Learning to Learn Cropping Models for Different Aspect Ratio Requirements. In *CVPR*, 2020. → pages 60, 61

[88] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep Joint Image Filtering. In *ECCV*, 2016. → page 114

[89] S. Liang, Y. Li, and R. Srikant. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. *ICLR*, 2018. → pages 34, 37, 43, 105

[90] J. Liebelt and C. Schmid. Multi-view Object Class Detection with a 3d Geometric Model. In *CVPR*, 2010. → page 10

[91] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing IKEA Objects: Fine Pose Estimation. In *ICCV*, 2013. → page 9

[92] J. J. Lim, A. Khosla, and A. Torralba. Fpm: Fine pose parts-based model with 3d cad models. In *ECCV*. 2014. → page 10

[93] T.-Y. Lin, Y.-T. Tsai, T.-S. Huang, W.-C. Lin, and J.-H. Chuang. Exemplar-based Freckle Retouching and Skin Tone Adjustment. *Computers & Graphics*, 78:54–63, 2019. → page 67

[94] F. Liu, C. Shen, and G. Lin. Deep Convolutional Neural Fields for Depth Estimation From a Single Image. In *CVPR*, 2015. → page 11

[95] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang. Semantic Image Segmentation via Deep Parsing Network. In *ICCV*, 2015. → page 10

[96] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*, 2015. → pages 1, 10, 16, 73

[97] A. Lugmayr, M. Danelljan, and R. Timofte. Ntire 2020 Challenge on Real-world Image Super-resolution: Methods and Results. In *CVPR Workshops*, 2020. → page 82

[98] W. Luo, X. Wang, and X. Tang. Content-based Photo Quality Assessment. In *ICCV*, 2011. → page 56

[99] M. Ma and J. K. Guo. Automatic Image Cropping for Mobile Device with Built-in Camera. In *First IEEE Consumer Communications and Networking Conference*, pages 710–711. IEEE, 2004. → page 56

[100] L. Marchesotti, C. Cifarelli, and G. Csurka. A Framework for Visual Saliency Detection with Applications to Image Thumbnailing. In *ICCV*, 2009. → page 56

[101] L. Marchesotti, F. Perronnin, D. Larlus, and G. Csurka. Assessing the Aesthetic Quality of Photographs using Generic Image Descriptors. In *ICCV*, 2011. → page 56

[102] J. Marin, D. Vázquez, D. Gerónimo, and A. M. López. Learning Appearance in Virtual Scenarios for Pedestrian Detection. In *CVPR*, 2010. → page 9

[103] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT press, 2012. → page 33

[104] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal Adversarial Perturbations. In *CVPR*, 2017. → page 35

[105] N. Murray, L. Marchesotti, and F. Perronnin. AVA: A Large-scale Database for Aesthetic Visual Analysis. In *CVPR*, 2012. → page 56

[106] Y. Nesterov. Introductory lectures on convex programming volume I: Basic course. *Lecture notes*, 3(4):5, 1998. → page 62

[107] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshops*, 2011. → page 38

[108] M. Nishiyama, T. Okabe, Y. Sato, and I. Sato. Sensation-based Photo Cropping. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 669–672. ACM, 2009. → page 57

[109] S. J. Pan and Q. Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010. → page 11

[110] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille. Weakly- and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation. In *ICCV*, 2015. → page 10

[111] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic Differentiation in PyTorch. *NIPS Workshops*, 2017. → pages 34, 62

[112] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning Deep Object Detectors from 3D Models. *ICCV*, 2015. → page 9

[113] I. M. Quintanilha, R. de M. E. Filho, J. Lezama, M. Delbracio, and L. O. Nunes. Detecting Out-Of-Distribution Samples Using Low-Order Deep Features Statistics, 2018. → page 38

[114] S. Rabanser, S. Günnemann, and Z. C. Lipton. Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift. *arXiv preprint arXiv:1810.11953*, 2018. → page 65

[115] K. Rematas, T. Ritschel, M. Fritz, and T. Tuytelaars. Image-based Synthesis and Re-synthesis of Viewpoints Guided by 3d Models. In *CVPR*, 2014. → page 10

[116] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*, 2015. → page 1

[117] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for Data: Ground Truth from Computer Games. In *ECCV*, 2016. → pages 12, 30

[118] G. Rogez, J. S. S. III, and D. Ramanan. First-Person Pose Recognition using Egocentric Workspaces. In *CVPR*, 2015. → page 10

[119] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*. Springer, 2015. → pages 68, 72

[120] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *CVPR*, 2016. → page 11

[121] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. → page 32

[122] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. *ICLR*, 2017. → page 36

[123] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobilenetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*, 2018. → pages ix, 61, 64, 116

[124] A. Saxena, S. H. Chung, and A. Y. Ng. 3-D Depth Reconstruction from a Single Still Image. *IJCV*, 2008. → page 11

[125] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. In *International Conference on Information Processing in Medical Imaging*, 2017. → pages 34, 38

[126] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support Vector Method for Novelty Detection. In *NIPS*, 2000. → page 36

[127] A. Shafaei and J. J. Little. Real-Time Human Motion Capture with Multiple Depth Cameras. In *CRV*, 2016. → page 10

[128] A. Shafaei, J. J. Little, and M. Schmidt. Play and Learn: Using Video Games to Train Computer Vision Models. In *BMVC*, 2016. → pages v, 3

[129] A. Shafaei, M. Schmidt, and J. J. Little. Does Your Model Know the Digit 6 Is Not a Cat? A Less Biased Evaluation of Outlier Detectors. *ArXiv e-prints*, 2018. → pages 5, 51

[130] A. Shafaei, M. Schmidt, and J. Little. A Less Biased Evaluation of Out-of-distribution Sample Detectors. In *BMVC*, 2019. → pages v, 51, 65, 114

[131] A. Shafaei, J. J. Little, and M. Schmidt. AutoRetouch: Automatic Professional Face Retouching. In *WACV*, 2021. → pages v, 5

[132] G. Shalev, Y. Adi, and J. Keshet. Out-of-distribution Detection using Multiple Semantic Label Representations. In *NeurIPS*, 2018. → page 38

[133] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. In *CVPR Workshops*, 2014. → page 11

[134] X. Shen, X. Tao, H. Gao, C. Zhou, and J. Jia. Deep Automatic Portrait Matting. In *ECCV*, 2016. → page 1

[135] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, and A. Kipman. Efficient Human Pose Estimation from Single Depth Images. *TPAMI*, 2013. → page 10

[136] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from Simulated and Unsupervised Images through Adversarial Training. In *CVPR*, 2017. → page 68

[137] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, and Others. Mastering the Game of Go with Deep Neural Networks and Tree Search. *nature*, 529(7587):484–489, 2016. → page 2

[138] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. *ArXiv e-prints*, 2014. → pages xiv, 16, 32, 45, 74

[139] V. Slaughter, V. E. Stone, and C. Reed. Perception of Faces and Bodies: Similar or Different? *Current directions in psychological science*, 13(6): 219–223, 2004. → page 57

[140] M. Stark, M. Goesele, and B. Schiele. Back to the Future: Learning Shape Models from 3D CAD Data. In *BMVC*, 2010. → page 10

[141] B. Suh, H. Ling, B. B. Bederson, and D. W. Jacobs. Automatic Thumbnail Cropping and its Effectiveness. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 95–104. ACM, 2003. → page 56

[142] B. Sun and K. Saenko. From Virtual to Reality: Fast Adaptation of Virtual Object Detectors to Real Domains. In *BMVC*, 2014. → page 9

[143] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing Properties of Neural Networks. In *ICLR*, 2014. → page 35

[144] G. R. Taylor, A. J. Chosak, and P. C. Brewer. OVVV: Using Virtual Worlds to Design and Evaluate Surveillance Systems. In *CVPR*, 2007. → page 9

[145] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time Continuous Pose Recovery of Human Hands using Convolutional Networks. *ACM Transactions on Graphics (TOG)*, 2014. → page 10

[146] A. Vedaldi and K. Lenc. MatConvNet – Convolutional Neural Networks for MATLAB. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015. → page 16

[147] A. Vyas, N. Jammalamadaka, X. Zhu, D. Das, B. Kaul, and T. L. Willke. Out-of-distribution Detection using an Ensemble of Self Supervised Leave-out Classifiers. In *ECCV*, 2018. → page 38

[148] W. Wang and J. Shen. Deep Cropping via Attention Box Prediction and Aesthetics Assessment. In *ICCV*, 2017. → pages 56, 57

[149] W. Wang, A. Wang, A. Tamar, X. Chen, and P. Abbeel. Safer Classification by Synthesis. *ArXiv e-prints*, 2017. → page 38

[150] W. Wang, J. Shen, and H. Ling. A Deep Network Solution for Attention and Aesthetics Aware Photo Cropping. *TPAMI*, 41(7):1531–1544, 2018. → page 59

[151] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: from Error Visibility to Structural Similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. → page 75

[152] Z. Wei, J. Zhang, X. Shen, Z. Lin, R. Mech, M. Hoai, and D. Samaras. Good View Hunting: Learning Photo Composition from Dense View Pairs. In *CVPR*, 2018. → page 59

[153] H. Wu, S. Zheng, J. Zhang, and K. Huang. Fast End-to-End Trainable Guided Filter. In *CVPR*, 2018. → page 114

[154] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *ArXiv e-prints*, 2017. → page 105

[155] L. Xu, J. Ren, Q. Yan, R. Liao, and J. Jia. Deep Edge-aware Filters. In *ICML*, 2015. → page 114

[156] J. Yan, S. Lin, S. Bing Kang, and X. Tang. Learning the Change for Automatic Image Cropping. In *CVPR*, 2013. → page 56

[157] R. K. Yin. Looking at Upside-down Faces. *Journal of experimental psychology*, 81(1):141, 1969. → page 57

[158] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How Transferable are Features in Deep Neural Networks? In *NIPS*. 2014. → page 11

[159] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *ICLR*, 2016. → page 10

[160] H. Zeng, L. Li, Z. Cao, and L. Zhang. Reliable and Efficient Image Cropping: A Grid Anchor based Approach. In *CVPR*, 2019. → pages 56, 57, 59, 60, 61

[161] M. Zhang, L. Zhang, Y. Sun, L. Feng, and W. Ma. Auto Cropping for Digital Photographs. In *ICME*, 2005. → pages 56, 57, 64

[162] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional Random Fields as Recurrent Neural Networks. In *ICCV*, 2015. → page 10

[163] W. Zhuo, M. Salzmann, X. He, and M. Liu. Indoor Scene Structure Analysis for Single Image Depth Estimation. In *CVPR*, 2015. → page 11

[164] D. Zoran, P. Isola, D. Krishnan, and W. T. Freeman. Learning Ordinal Relationships for Mid-Level Vision. In *ICCV*, 2015. → pages xiii, 11, 24, 25, 26

Appendix A

OOD Detection: Formulation and Evaluation

A.1 Evaluation of Unsupervised Techniques

Algorithm 2: OD-test – the evaluation procedure for an unsupervised method $\mathcal{M}$.

input : $\mathcal{D}_{\mathrm{s}} = (\mathcal{D}_{\mathrm{s}}^{\mathrm{train}}, \mathcal{D}_{\mathrm{s}}^{\mathrm{valid}}, \mathcal{D}_{\mathrm{s}}^{\mathrm{test}})$ the source dataset.
input : $\mathrm{D} = \{\mathcal{D}_i\}$ outlier **set**.
input : $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ the method under evaluation.

```
begin
    A ⟵ {}
    /* Generate a rejection hypothesis r using
       D_s^train.                                   */
    r ⟵ M(D_s^train)
    for D_t ∈ D do
        /* Evaluate the accuracy of r.              */
        acc ⟵ eval(r, {D_s^test : 0, D_t : 1})
        add acc to A
    return mean(A)
```

Algorithm 2 outlines the steps of evaluation for an unsupervised method $\mathcal{M}$. Note that in this setting $\mathcal{M}$ returns a single binary classifier r. To make the perfor-

mance of supervised and unsupervised methods comparable, we use the same splits of the datasets to guarantee fairness of evaluation. In this work, we do not evaluate any unsupervised method. This additional information is provided for clarity and completeness.

A.2 Implementation Details

For each training procedure, we randomly separate 80% and 20% of the (sub-)data for training and testing respectively. We return the model that has the highest performance on the test (sub-)subset. For classification tasks, we measure the performance by classification accuracy while for other tasks such as `AEThreshold` we measure the performance through the respective loss value on the test set.

`VGG`, `Resnet` We train two popular and generic classifier architectures `VGG-16` and `Resnet-50` on $\mathcal{D}_{\mathrm{s}}^{\mathrm{train}}$ to perform the corresponding classification task within the datasets. The network architectures slightly differ across datasets to account for the change in the spatial size or the number of classes. We apply our modifications to the reference implementations available in PyTorch's `torchvision` package. These trained networks are subsequently used in `PbThreshold`, `ScoreSVM`, `ODIN`, `Log.SVM`, and `DeepEnsemble`. For `MC-Dropout` we only use the `VGG` variant, as the `Resnet` variants do not have dropouts. Table A.1 shows the summary of the networks' classification accuracies on the entire $\mathcal{D}_{\mathrm{s}}^{\mathrm{train}}$ set.

Data Augmentation We only allow mirroring of the images for data augmentation. We apply data augmentation on all the datasets except `MNIST` for which mirror augmentation does not make sense. We explicitly instantiate mirrored samples, as opposed to implicit on-air augmentation, to ensure methods such as `K-NNSVM` are not at disadvantage. We do not apply any other data augmentation. We initially experimented with no data augmentation. Without any augmentation, the performance of all the methods reduces by $3-4\%$, but the relative ranking stays the same.

`BinClass` A binary classifier that is directly trained on $\mathcal{D}_{\mathrm{s}}$ and $\mathcal{D}_{\mathrm{v}}$. We use the reference `Resnet` or `VGG` architectures with an additional linear layer to transform

	VGG			Resnet		
	CE-Accuracy	KL-Accuracy	Size	CE-Accuracy	KL-Accuracy	Size
MNIST [78]	99.89%	99.91%	19 MB	99.89%	99.91%	70 MB
FashionMNIST [154]	98.82%	98.36%	19 MB	98.75%	98.73%	70 MB
CIFAR10 [75]	97.63%	97.34%	159.8 MB	97.75%	97.51%	94.3 MB
CIFAR100 [75]	91.40%	91.87%	161.3 MB	92.05%	91.82%	95.1 MB
TinyImagenet[a]	69.71%	72.23%	162.9 MB	89.59%	65.95%	95.9 MB
STL10 [27]	93.62%	95.18%	201.7 MB	92.32%	93.34%	94.3 MB
Mean	91.84%	92.48%		95.05%	91.21%	

Table A.1: The classification accuracy of the trained networks on $\mathcal{D}_{\mathrm{s}}^{\mathrm{train}}$ using cross-entropy (CE) and K-way Logistic (KL) loss functions. In both scenarios, the prediction is the maximum activation. Note that because of the difference in training data, this table is not comparable to the state-of-the-art performance on the respective datasets.

[a] https://tiny-imagenet.herokuapp.com/

the output of the network to a one-dimensional activation. We train the network with a binary cross-entropy loss on ($\mathcal{D}_{\mathrm{s}}^{\mathrm{train}}$+$\mathcal{D}_{\mathrm{s}}^{\mathrm{valid}}$:0, $\mathcal{D}_{\mathrm{v}}$:1) to ensure the method has access to the same data as the other methods. The networks typically achieve near-perfect accuracy after only a few epochs.

PbThreshold[55] One threshold parameter on top of the maximum of softmax output. The cost of the evaluation is a single forward pass on the network. We reuse the trained reference VGG or Resnet architectures for this.

ScoreSVM A natural generalization of PbThreshold is to train an SVM [29] classifier on the pre-softmax activations. The cost of evaluation is a single forward pass on the network with the additional SVM layer. We reuse the trained reference VGG or Resnet architectures for this. We set the weight-decay regularization to $\frac{1}{m}$, where m is the size of the training set.

ODIN[89] A threshold on the softmax outputs of the perturbed input. The cost of the evaluation is two forward passes and one backward pass. We do a grid search over the ϵ, the perturbation step size, and γ, the temperature of the softmax operation. The range for grid search is the same as the suggested range in [89]. We

reuse the trained reference `VGG` or `Resnet` architectures for this.

K-NNSVM A linear SVM on the sorted Euclidean distance between the input and the k-nearest training samples. Note that a threshold on the average distance is a special case of `K-NNSVM`. The cost of the evaluation is finding the k-nearest neighbours in the training data. We use the $\mathcal{D}_{\text{s}}^{\text{train}}$ as the reference set, and tune the parameters with ($\mathcal{D}_{\text{s}}^{\text{valid}}$, $\mathcal{D}_{\text{v}}$).

K-MNNSVM, K-BNNSVM, K-VNNSVM The same as `K-NNSVM`, except we use the low dimensional representations of an autoencoder trained with MSE, BCE, or the VAE.

AEThreshold A threshold on the autoencoder reconstruction error of the given input. The evaluation cost is a single forward pass on the autoencoder. We train the autoencoder on $\mathcal{D}_{\text{s}}^{\text{train}}$ and train the threshold parameters with ($\mathcal{D}_{\text{s}}^{\text{valid}}$,$\mathcal{D}_{\text{v}}$). We use the binary cross-entropy loss with continuous targets[1] or mean squared error to train and measure the reconstruction error of a given input. The bottleneck dimensionality varies between 32 and 1024. Our decision rule given a reconstruction error e_x for an input x is $r(x) = (e_x - \mu)^2 > \tau$, where τ is the threshold and μ is the center around which we are thresholding with τ. If we set $\mu = 0$, this decision function reduces to a basic threshold operator. We found that this simple decision rule improves the final accuracy of the model. The reconstruction errors of the in-distribution samples tend to stay more or less similar, whereas the reconstruction error for OOD samples could either be too low or too high. This decision rule is meant to utilize this observation. The network architectures are procedurally generated. See https://github.com/ashafaei/OD-test/blob/master/models/autoencoders.py for the models.

MC-Dropout A threshold on the entropy of average predictions of 7 evaluations per input. The dropout probability is $p = 0.5$. This approach follows the work of [77] and Kendall and Gal [69]. We did not evaluate this approach on `Resnet`

[1] See http://pytorch.org/docs/0.3.1/nn.html#bcewithlogitsloss.

because the original structure does not have a dropout; therefore, it is not trivial to identify where the dropouts should be located without sabotaging the performance of `Resnet`. We reuse the trained reference `VGG` architecture for this.

`DeepEnsemble` Similar to `MC-Dropout`, except we average over the predictions of 5 networks that are trained independently with the adversarial strategy of [77]. In this approach, we augment the original loss function with a similar loss function on the adversarially-generated examples of the same batch. The adversarially-generated samples are generated through the fast gradient-sign method (FGSM) [50].

`PixelCNN++` We use the implementation from https://github.com/pclucas14/pixel-cnn-pp. We train the models using $\mathcal{D}_{\text{s}}$ until plateau on the test (sub-)subset, then learn a threshold parameter with $\mathcal{D}_{\text{v}}$. Our models achieve a 0.89 BPD for MNIST, 2.65 BPD for FashionMNIST, 2.98 BPD for CIFAR10, 3.01 BPD for CIFAR100, 2.70 BPD for TinyImagenet, and 3.59 BPD for STL10 on the test (sub-)subset. Because of the auto-regressive nature, these models are prohibitively expensive to train. The `PixelCNN++` authors note that they have used 8 Titan X GPUs for five days to achieve state-of-the-art performance for CIFAR10[2] (2.92 BPD). For TinyImagenet, and STL10 we process a downsampled version to 32-pixel width to be able to train and evaluate the models. Our experiments with `AEThreshold` indicate that the downsampled versions of TinyImagenet, and STL10 are easier problems. However, even with this simplification, the `PixelCNN++` does not perform up to expectations.

`OpenMax` This method is a replacement for the softmax layer *after* the training has finished. It fits a Weibull distribution on the distances of logits from the representatives of each class to reweight the logits and provide probabilities for encountering an unknown class. The output of the OpenMax is similar to softmax, except with the addition of the probability for an unknown class. We learn the MAV vectors and the Weibull distribution on the $\mathcal{D}_{\text{s}}$. We use the $\mathcal{D}_{\text{v}}$ to learn the reject

[2] https://github.com/openai/pixel-cnn

function on the calibrated probability outputs.

You can access all the results on https://github.com/ashafaei/OD-test where you will find the full list of evaluations for OD-test ($n = 34 \times 308 = 10{,}472$) and the two-dataset evaluation scheme ($n = 22 \times 46 = 1012$).

A.3 More Results

Figure A.1 shows the average performance of all the methods per source dataset $\mathcal{D}_{\text{S}}$.

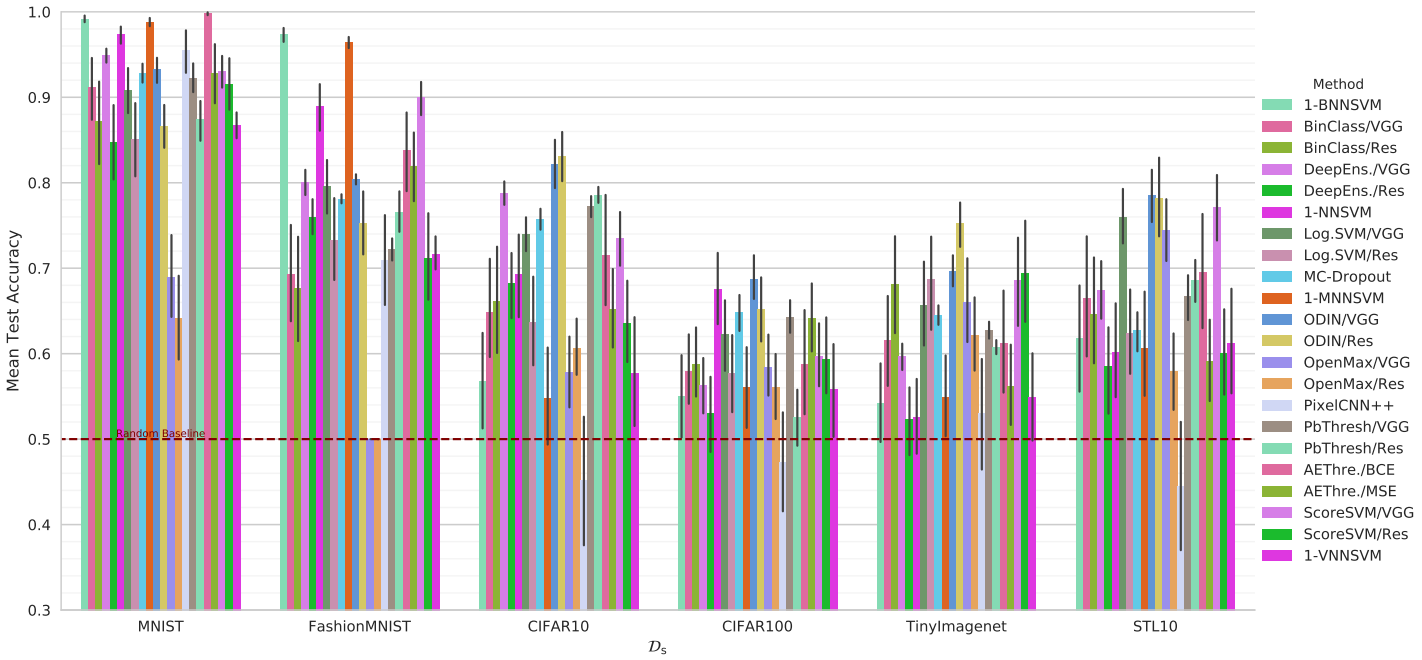

Figure A.1: The average test accuracy over 50 experiments per bar. The error bars indicate the 95% confidence level. The figure is best viewed in colour.

Appendix B

AutoPortrait Supplementary Material

B.1 Cropping

B.1.1 From the Reference Cropping to a Specific Cropping

We described how we crop a portrait into the reference cropping form where all input images are consistently cropped first. In the application, however, we wish to have the freedom to perform cropping to any desired configuration. To achieve this, we first find the σ^* cropping for a given image, then we perform a fixed transformation on the σ^* to generate a desired cropping $\hat{\sigma}$. Since we apply a fixed transformation on all the images, the resulting crop $\hat{\sigma}$ is also going to be a consistent cropping.

To specify the desired cropping we will need the output `width` and `height`, as well as the relative `scale` of the head and its coordinates `x`, and `y`. We define these parameters in relation to the reference cropping window. See Fig. B.1 for a visual description of these parameters. We first infer the reference cropping form (σ^*) using the described method. Let us assume $\sigma^* = [t_s, t_x, t_y]^{\mathrm{T}}$ is the configuration to achieve reference cropping. Recall that t_s is in the log-space. We can simply update

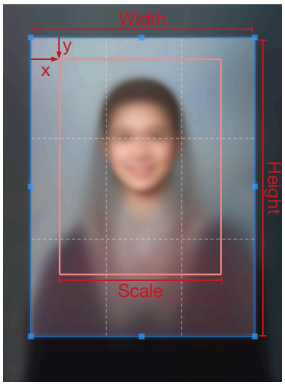

Figure B.1: The relationship between the reference cropping window and a specific cropping window. The red box is the reference cropping window, and the blue box is the user-specified cropping region. The `scale` parameter is a multiplier to the 120×160 canonical cropping window.

$\sigma^* \rightarrow \hat{\sigma} = [\hat{t_s}, \hat{t_x}, \hat{t_y}]^{\mathrm{T}}$ as follows:

$$\hat{t_x} := t_x - \frac{\mathrm{x}}{e^{t_s}} \tag{B.1}$$

$$\hat{t_y} := t_y - \frac{\mathrm{y}}{e^{t_s}} \tag{B.2}$$

$$\hat{t_s} := t_s + \log(\mathrm{scale}) \tag{B.3}$$

Once we have $\hat{\sigma}$ we resample the selected window into a `width×height` image.

B.1.2 Parameter Projection

We write the projection as a quadratic program (QP) under a weighed ℓ_2 norm. For this step, we transform t_s to the linear-space first. After the projection, we transform t_s to the log-space again. Assuming $\mathrm{t_w}$, $\mathrm{t_h}$ are the output width and height of a cropping window, and $\mathrm{I_w}$, $\mathrm{I_h}$ are the width and the height of the input image, the

projection constraints are:

$$t_s, t_x, t_y \geq 0 \tag{B.4}$$

$$t_x + t_s * \mathrm{t_w} \leq \mathrm{I_w} \tag{B.5}$$

$$t_y + t_s * \mathrm{t_h} \leq \mathrm{I_h} \tag{B.6}$$

Equation B.4 comes from the fact that the scale and the translation variables must remain positive. Equation B.5, and Equation B.6 are the upper-bound of the translation variables as controlled by the scaling variable t_s. These requirements directly translate into linear inequality constraints of our QP projection. When a target cropping is not feasible, we will have to adjust translation and scale variables. The following objective controls the amount of adjustment:

$$\frac{1}{2}(\tilde{\sigma} - \sigma)^{\mathrm{T}} W (\tilde{\sigma} - \sigma), \tag{B.7}$$

where σ is the point to be projected, and $\tilde{\sigma} = [t_s, t_x, t_y]^{\mathrm{T}}$ is the projection. W is the diagonal weight matrix that controls the rate at which the variables are adjusted to satisfy the constraints. We set W to $\mathrm{diag}([1000, 1, 10])$ to emphasize that we are less tolerant to change in t_s, and t_y, compared to t_x. This weighting scheme comes from our goal to generate consistently scaled images, even if it is at the expense of slight mistranslation.

B.1.3 Evaluation

Figure B.2 shows the evolution of cropping on nine sample images over six iterations. The estimates do not change significantly after the sixth iteration.

B.2 Colour Correction

During this step, colour analysts adjust the image's colour distribution according to the studio's production style. The analysts visually inspect the images on calibrated screens in dark rooms and tune parametric colour curves to adjust the image. Factors such as the amount of emitted light by the equipment or the camera's colour sensitivity can significantly affect the images. The amount of correction applied

Figure B.2: Cropping over six iterations.

also depends on several contextual factors, such as the clothing and the background.

It should be noted that:

- What constitutes a good looking image is entirely subjective.
- Not all the colour analysts tend to agree on the same final image.

Not only the goal of colour correction is not well-defined, but it also appears that there is no unique answer. To perform colour correction on each image, the colour analysts apply six global updates to the image:

Density The amount of brightness change in the image.

R, G, B The amount of shift in the red, green, and blue channels.

Contrast The amount of contrast change in the image.

Gamma The amount of gamma correction in the image.

While the classical mathematical models of these operations are linear, modern image processing toolboxes use non-linear parametric curves to update the image. The exact form of these parametric curves tends to differ from software to software and is usually not openly documented. The professional colour analysts are trained distinguish even the smallest change in the five major dimensions mentioned.

Following the successful previous work [23, 24, 45, 47, 47, 71, 88, 153, 155], we also predict image correction parameters based on a downsampled version of the image and apply the resulting correction function on the high-resolution image. However, instead of using an unrestricted high-dimensional bilateral grid of affine functions for each pixel [47], we (i) restrict the possible function space by introducing a parametric and differentiable colour correction model that is (ii) low-dimensional in degrees of freedom, yet adequate for the task, (iii) updates the image with explainable operations, and is (iv) more amenable to quality control for automatic failure detection considering that there are several effective methods for anomaly detection in low-dimensional spaces [48, 130].

Specifically, we approximate typical colour operations such as contrast adjustment, or brightness update, using polynomial transfer functions of this form:

$$f_{\mathbf{w}}(x,\, \theta) = [x^3\theta, x^2\theta^2, x^2\theta, x\theta^2, x\theta, \theta^2, \theta]^{\mathrm{T}}\mathbf{w} + x, \tag{B.8}$$

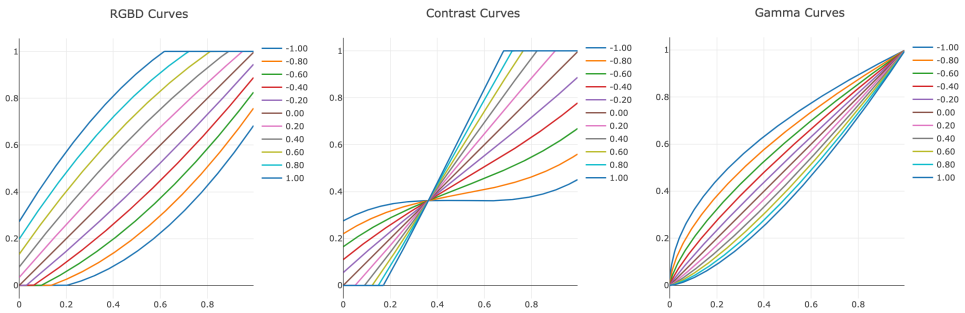

Figure B.3: The update curves with varying θ for each operation. $\theta = 0$ corresponds to identity function for all operations.

where $\mathbf{w}$ is a vector encoding the operation, and the function maps a pixel-channel value $x \in [0, 1]$ to a new value using the correction parameter $\theta \in \mathbb{R}$. Although we initially experimented with more complex multi-channel functions, we have found this simpler functional structure to be sufficient to approximate the typical colour adjustment operations up to the sRGB colour quantization levels. The function $f_{\mathbf{w}}$ is specifically structured so that θ controls the amount of operation with $\theta = 0$ amounting to a no-op.

In our experiments, we model the colour adjustment behaviour of Kodak Professional Digital Print that is popular for studio production pipelines. To learn the $\mathbf{w}$ of each operation, we first collect training data from the software by generating images with various adjustments. Then, we optimize $\mathbf{w}$ to generate transfer curves that match the output of the software. See Fig. B.3 for a visualization of our operations with varying θ.

We compose gamma correction, R, G, B, brightness, and contrast updates, followed by clamping to $[0, 1]$ into a single function $F(\mathbf{x}, \Theta)$, where Θ is the concatenation of all the intermediate correction parameters θ_i. F is smooth in $x \in (0, 1)$ and $\theta_i \in [-1, 1]$ allowing a properly initialized network to be trained with backpropagation. Given an image $\mathbf{x}$ and adjusted image $\mathbf{y}$, we can find the $\hat{\Theta}$ such that $F(\mathbf{x}, \hat{\Theta}) \approx \mathbf{y}$ by minimizing Θ over $\|F(\mathbf{x}, \Theta) - \mathbf{y}\|_2^2$ with backpropagation. The resulting $\hat{\Theta}$ is both explainable in terms of specific operations applied, and also transferable to our target software for further colour analysis.

Given an image $\mathbf{x}$, we estimate $g(\mathbf{x}) = \Theta$, and return the colour-corrected

image with $F(\mathbf{x}, \Theta)$. We estimate Θ from a downsampled image using deep neural networks and perform end-to-end learning by penalizing error on the high-resolution image output. We use the MobileNetV2 [123] architecture to predict Θ.

B.3 Skin Retouching

B.3.1 The Discriminator

For the discriminator we use the following sequential architecture from PatchGAN [60].

```
(0): Conv2d(3, 64, kernel_size=(4, 4),
            stride=(2, 2), padding=(2, 2))
(1): LeakyReLU(negative_slope=0.2, inplace)
(2): Conv2d(64, 128, kernel_size=(4, 4),
            stride=(2, 2), padding=(2, 2))
(3): InstanceNorm2d(128, eps=1e-05, momentum=0.1,
            affine=False)
(4): LeakyReLU(negative_slope=0.2, inplace)
(5): Conv2d(128, 256, kernel_size=(4, 4),
            stride=(2, 2), padding=(2, 2))
(6): InstanceNorm2d(256, eps=1e-05, momentum=0.1,
            affine=False)
(7): LeakyReLU(negative_slope=0.2, inplace)
(8): Conv2d(256, 512, kernel_size=(4, 4),
            stride=(1, 1), padding=(2, 2))
(9): InstanceNorm2d(512, eps=1e-05, momentum=0.1,
            affine=False)
(10): LeakyReLU(negative_slope=0.2, inplace)
(11): Conv2d(512, 1, kernel_size=(4, 4),
            stride=(1, 1), padding=(2, 2))
```

B.3.2 Multiscale Patch Sampling

The number of $w \times w$ patches in the image increases quadratically with respect to the scale of the image. We wish to augment the training data by downsampling the images. If we choose a down-scaling parameter at uniform, it will under-represent the $w \times w$ patches in the spatially larger images. Therefore we should be sampling higher-resolution images quadratically more often to make all the patches equally

likely. The cumulative distribution function of a quadratically increasing density function is cubic. Using the inverse transform sampling method we can simply use $s^{\frac{1}{3}}$ as the scaling factor, where $s \sim \mathrm{Unif}(0,1)$. This sampling procedure is approximate because we also have a minimum scale that truncates the density function. If we set a minimum scale $s_{\min}$, the scaling factor becomes $(s(1-s_{\min}^3))^{\frac{1}{3}}$; however, since we are using high-resolution images, the minimum scaling factor cubed $s_{\min}^3$ will become small enough ($< 10^{-3}$) that the difference is negligible. However, if the training data is not high-resolution, omitting the extra term could still produce unbalanced samples.

B.3.3 User Study

To perform the user study, we developed the UI in Fig. B.4. Each time, we show the user the output of two algorithms in random order and ask them to pick their favourite. At the bottom of the page, we show the user a zoomed-in version of the original image and the two outputs and an image highlighting the differences. As the users move the mouse cursor over the original image, they can visually inspect and compare a zoomed version of all the images. To conduct our user studies, we hired three professional retouchers. We run several experiments under this evaluation framework and present the results in the thesis.

For FFHQR evaluations, we pick 1000 images from the test set. More specifically, we use the images from `63000` to `63999`. Similarly, for the studio data, we pick 1000 images from the test set. Each user sees the 1000 evaluations in random order. We ran seven experiments, collected over 5400 votes, while spending 48 hrs in total.

B.3.4 Evaluation

Figure B.5 compares the retouching output of our model with the groundtruth patches. Our model preserves the fine texture better than the groundtruth data. In the thesis we perform ablation studies on the effect of each term in the loss function. We observed that adding RAGAN loss term encourages the model to preserve the input as much as possible. Preservation of the details produces retouching models that perform better than the groundtruth retouching data. We suspect this

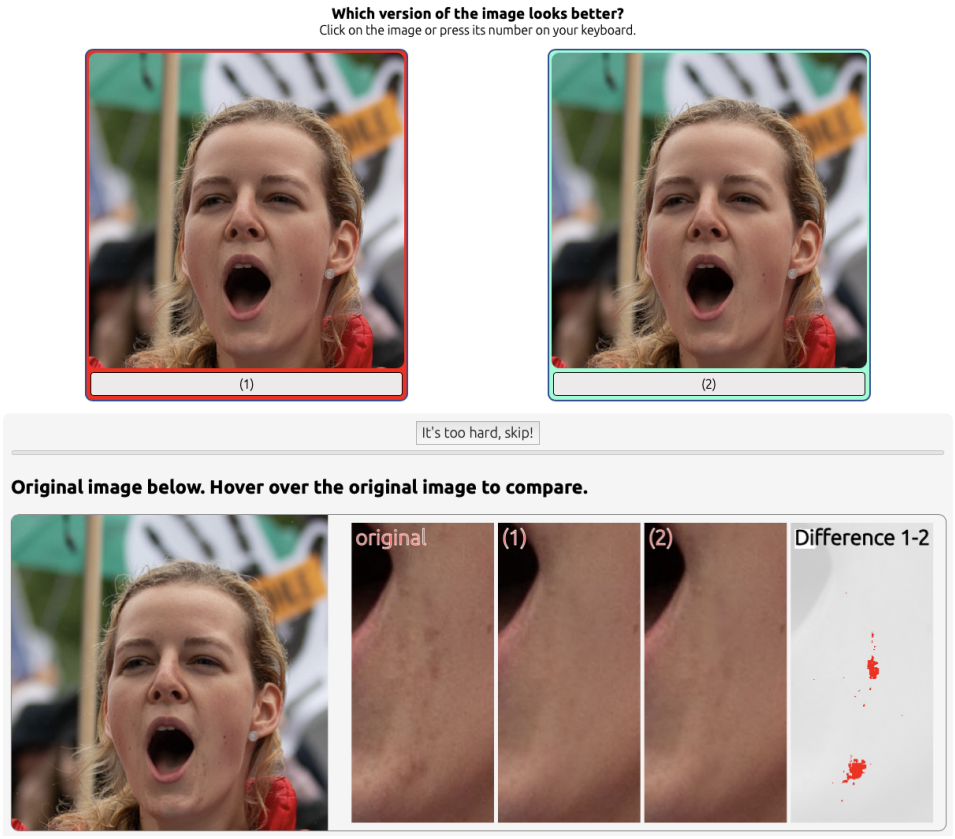

Figure B.4: The user study UI. The users are shown two images in random order, and they will decide which version they prefer. At the bottom of the page, dynamically changing figures allow easy comparison between the algorithms.

happens because, in real-life retouching, the professionals may use large brushes for correction that inadvertently affects areas of the image that do not require retouching. Our model, however, operates at the pixel level and can preserve details at no extra cost.

We also test our retouching model on lower-resolution smartphone camera images. The images are captured with iPhone 6 or iPhone X. Figure B.6 shows sample outputs from our tests.

Figure B.7 shows the failure cases of our retouching model. The images that our

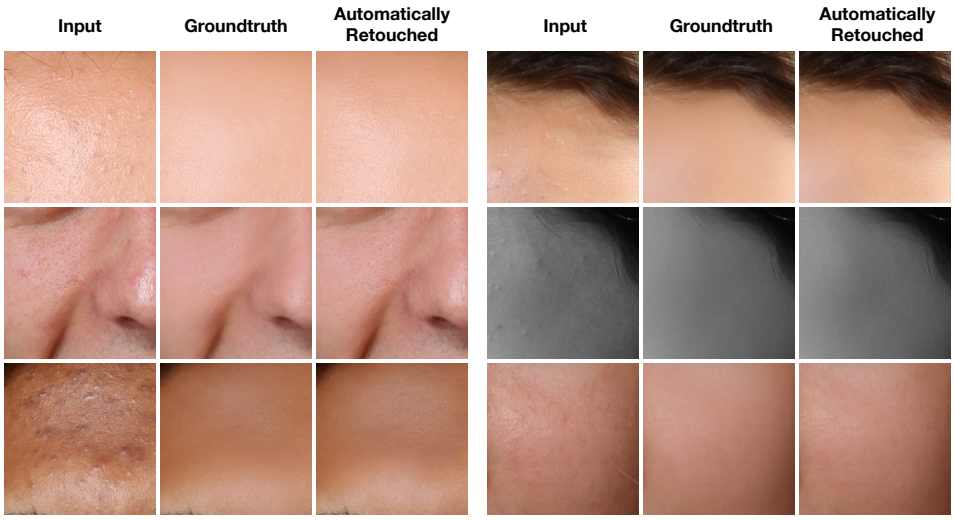

Figure B.5: The output of our model compared to the groundtruth retouching. The left column is the input, the middle column is groundtruth retouching, and the right column is our output. Our model preserves the fine details more than the groundtruth.

model fails to correct usually contain severe skin blemishes. Although our model improves the output image, it does not fully eliminate the blemishes.

Figure B.8 shows the output of our model on full-faces.

B.3.5 FFHQR

See Fig. B.9 for more samples of our new retouching dataset.

B.3.6 Studio Data

Figure B.10 shows the distribution of the head-crop images that we extracted from the studio training data to train our models.

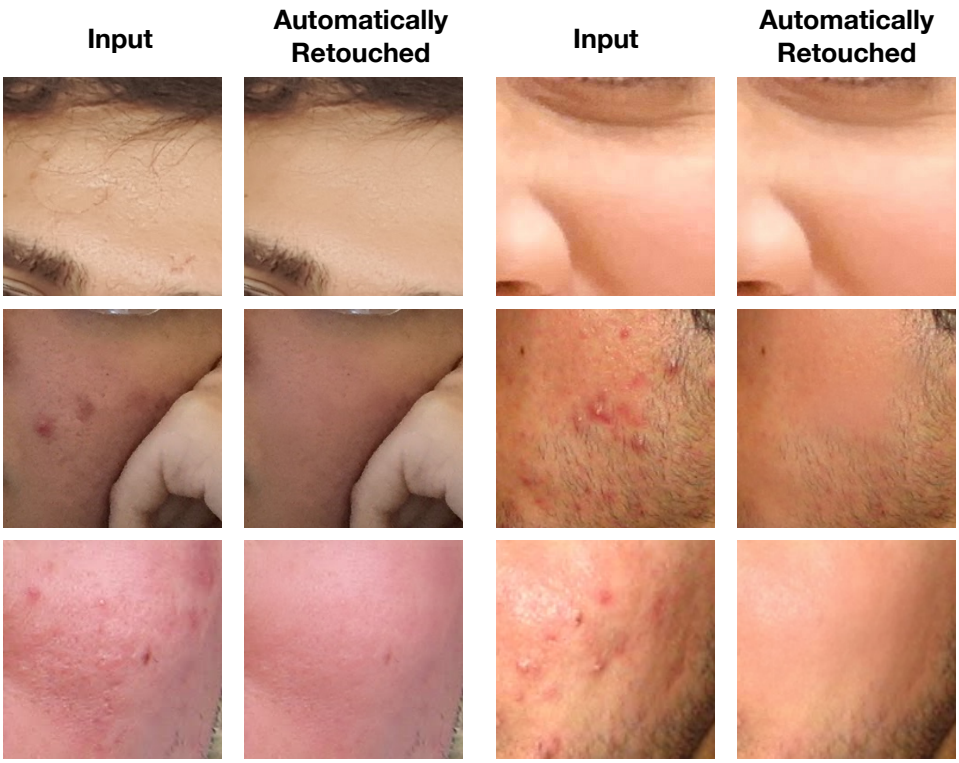

Figure B.6: Sample input/output of retouched images captured with cell-phones. The figure is best viewed on a screen.

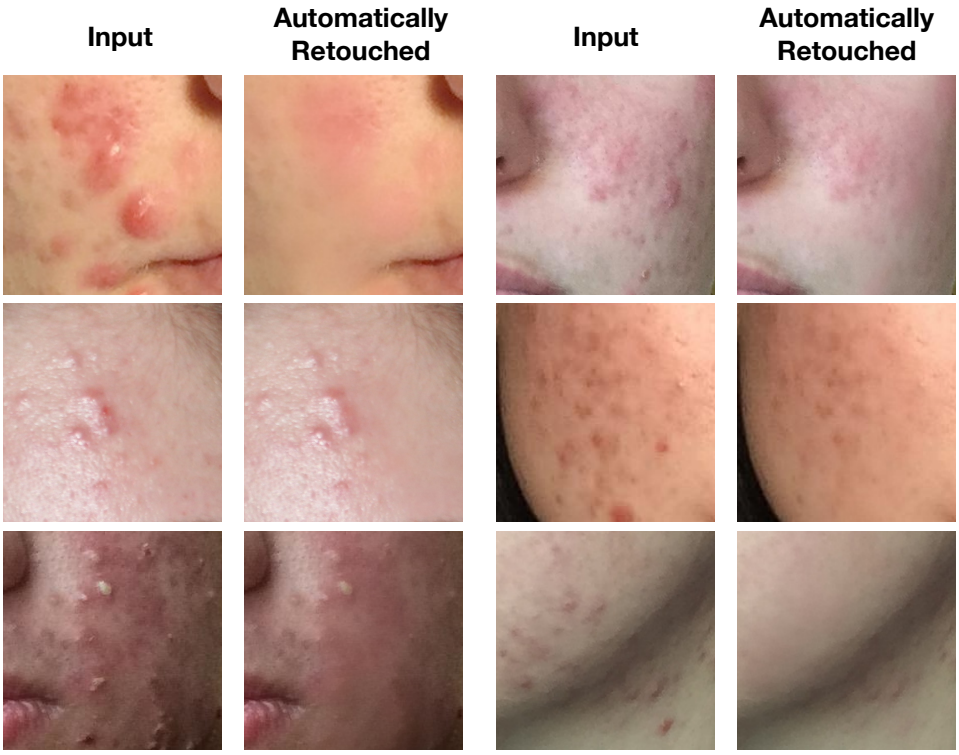

Figure B.7: Failure cases. Our retouching model fails when blemishes are severe.

Figure B.8: Original photo on the left, our automatic output on the right. Parts of the image are blurred to maintain anonymity. Skin retouching is usually the most time-consuming step of retouching.

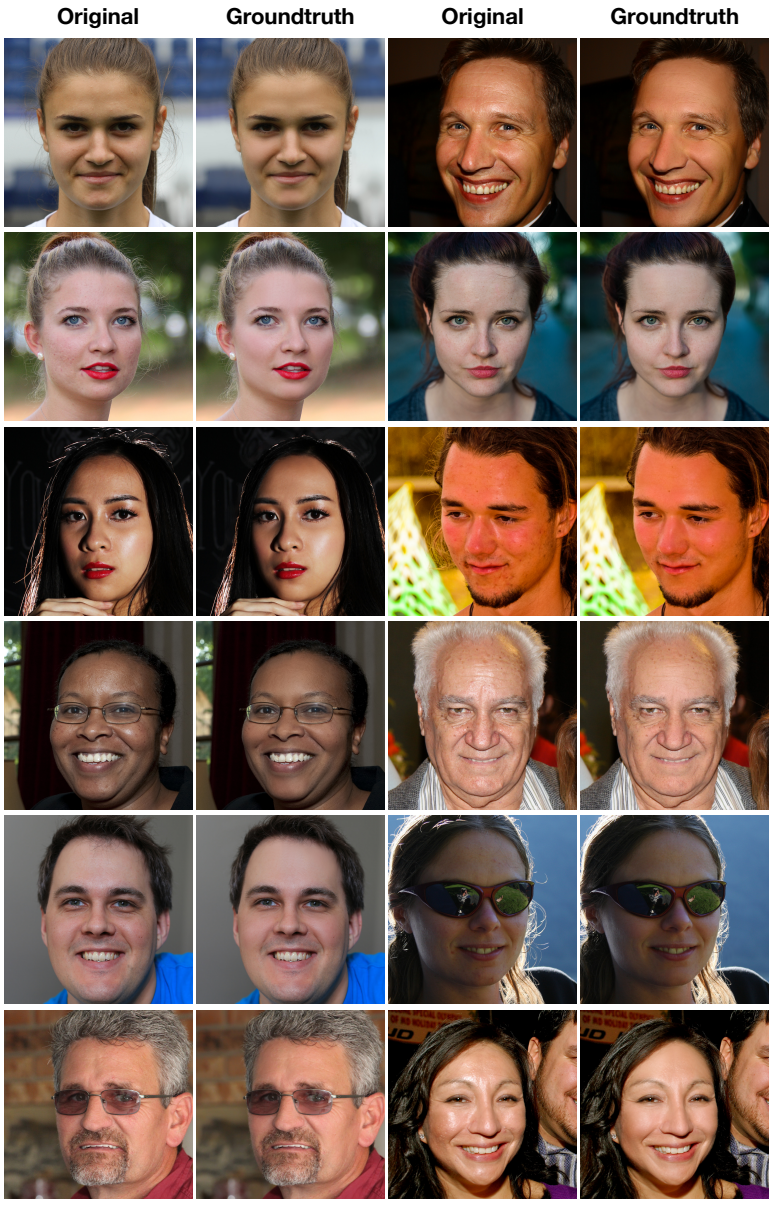

Figure B.9: More samples from our new retouching dataset FFHQR.

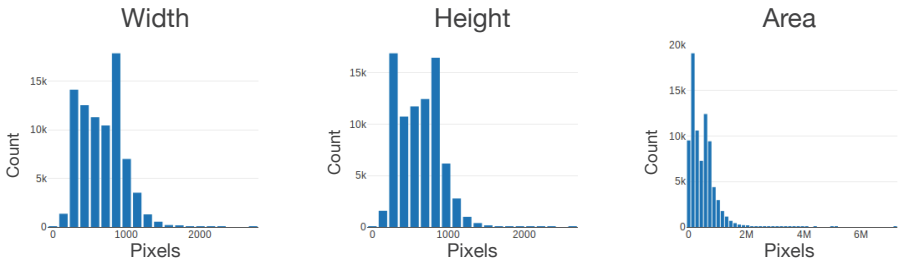

Figure B.10: The distribution of width, height, and area of the head-crops extracted from the studio retouching data. The data is similar to FFHQR in resolution.