IMAGE GENERATION WITH CONVOLUTIONAL NEURAL NETWORKS

Doctoral Thesis

by

DMITRY ULYANOV

DOCTORAL PROGRAM IN COMPUTATIONAL AND DATA SCIENCE AND ENGINEERING

Supervised by
Professor Victor Lempitsky, Skoltech
Professor Andrea Vedaldi, University of Oxford

Moscow

Abstract

Modern convolutional neural networks (ConvNets) recently found their application in image generation tasks. The advent of ConvNets led to significant improvement of data-driven image generation and allowed very complex scenarios like real-time face reenactment or animation of an arbitrary photograph. This thesis is based on a collection of papers that describe algorithmic advances and results obtained by convolutional neural networks. We first suggest a method for fast style transfer and texture synthesis and then improve it in several aspects. Next, we propose a novel formulation of a Generative Adversarial Networks (GAN) game, which allows learning a mapping from the image space to the latent space while maintaining the complexity of the system at the same level. Then, we equip GAN with a discriminator based on perceptual features and show its superior performance over the original GAN. We then create a GAN-based image generation system to synthesize images of a human, driven by the pose of an actor. Finally, we take a step towards explaining the "magic" of ConvNets, by decoupling the impact of the convolutional architecture from the learning on a dataset and show that an architecture itself imposes a strong image prior, especially helpful in image processing and generation tasks.

List of Publications

Works with the main contribution by the author:

1. Dmitry Ulyanov, Andrea Vedaldi and Victor Lempitsky. **Texture Networks: Feed-forward Synthesis of Textures and Stylized Images**. International Conference on Machine Learning (ICML), 2016; CORE rating = A*
2. Dmitry Ulyanov, Andrea Vedaldi and Victor Lempitsky. **Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis**. Computer Vision and Pattern Recognition (CVPR), 2017; CORE rating = A*
3. Dmitry Ulyanov, Andrea Vedaldi and Victor Lempitsky. **It Takes (Only) Two: Adversarial Generator-Encoder Networks**. AAAI Conference on Artificial Intelligence (AAAI), 2018; CORE rating = A*
4. Dmitry Ulyanov, Andrea Vedaldi and Victor Lempitsky. **Deep Image Prior**. Computer Vision and Pattern Recognition (CVPR), 2018; CORE rating = A*

In the following works the author contributed to specific parts:

1. Diana Sungatullina, Egor Zakharov, Dmitry Ulyanov and Victor Lempitsky. **Image Manipulation with Perceptual Discriminators**. European Conference on Computer Vision (ECCV), 2018; CORE rating = A

   Contributions:

   (a) Designed the main experiments and prepared the baselines

   (b) Performed C2ST evaluation of the proposed method and the baselines

   (c) Executed experiments with a part of baselines

2. Aliaksandra Shysheya, Egor Zakharov, Kara-Ali Aliev, Renat Bashirov, Egor Burkov, K Iskakov, A Ivakhnenko, Y Malkov, Igor Pasechnik, Dmitry Ulyanov, A Vakhitov, V Lempitsky **Textured Neural Avatars**. Computer Vision and Pattern Recognition (CVPR), 2019; CORE rating = A*

   Contributions:

   (a) Co-designed the proposed method and evaluation methodology

   (b) Evaluated different ways to represent a 3D pose ("stickman")

   (c) Contributed to paper writing, and designed the main explanatory figure

Acknowledgements

I am grateful to Skolkovo Institute for gathering the best professors and students and creating a unique environment for research. It is unbelievable how much the institute has grown within just four years. Thanks to Yandex for supporting and initiating the Yandex-Skoltech-Oxford collaboration that I became a part of.

Victor, Andrea, it is genuinely impossible to overrate your significance for my life and for this thesis. I am impressed by you and very happy that I had a chance to collaborate and learn from you. Thanks for teaching me despite my stubbornness and challenging me with exciting problems every day.

Thanks to my collaborators, who have contributed to the projects described in this thesis. I am thankful to Dmitry Vetrov, whose dedication to science inspired me to step into the PhD program in the first place.

Finally, this work could not be completed without my loving wife Ekaterina and my parents, who supported me every day (and nights before deadlines) of this four-year-long journey.

Contents

List of Abbreviations

3D Three dimensional

AGE Adversarial Generator-Encoder networks, a GAN model proposed in chapter 4

AlexNet A popular neural network architecture named after Alex Krizhevsky [Krizhevsky et al., 2012]

CNN Convolutional Neural Network

CPU Central processing unit

GAN Generative Adversarial Networks, a generative model [Goodfellow et al., 2014]

GPU Graphics processing unit

ReLU Rectified Linear Unit, a popular activation function [Maas et al.]

ResNet Residual Network, a moder neural network architecture [He et al., 2016]

RGB Red-green-blue, channels in a digital representation of an image

VAE Variational Autoencoder, a generative model [Kingma and Welling, 2014]

VGG A popular neural network architecture, proposed by Visual Geometry Group, University of Oxford [Simonyan and Zisserman, 2014]

Chapter 1

Introduction

This thesis is based on a collection of papers that describe algorithmic advances and results obtained by convolutional neural networks.

1.1 Motivation

Originally, neural networks were designed for important discriminative tasks, such as classification and regression [Rumelhart and McClelland, 1987]. Neural networks were mainly used to work with unstructured data, where each object is represented with a vector of independent features. Neural networks were later applied to image data thanks to a proposed convolutional layer [LeCun et al., 1989]. Since then *convolutional neural networks* have been shown to be successful on computer vision problems, such as image classification [Krizhevsky et al., 2012, He et al., 2016], detection [Girshick et al., 2014], and segmentation [Long et al., 2015b, Gong et al., 2018]. In mentioned tasks, a neural network is trained to map input objects to some representation useful in practice: image label for classification, semantic map for segmentation, or bounding box for detection. Convolutional neural networks were also successfully applied to image restoration problems, such as image super-resolution [Dong et al., 2014b, Ledig et al., 2017b], denoising [Burger et al., 2012, Yang and Sun, 2018], or inpainting [Iizuka et al., 2017b, Köhler et al., 2014], where the task is essentially *to generate* an image, rather than to predict an attribute of a given image.

While some image generation problems such as super-resolution can be solved discriminatively by learning a neural network on a huge dataset, many practical problems still cannot be formulated as a discriminative task and thus cannot be solved using standard machine learning techniques and neural networks models. For example, given a dataset,

sampled from an unknown distribution P, how can we generate more samples from the distribution P? How to determine if a given test sample is likely to be generated from the distribution P or it is coming from a different distribution? Answering these questions would be very useful for many real-world applications, including fraud and outlier detection. Another example: how can we perform style transfer? That is, for a given input image and style, how do we synthesize a new image, with the same content as on the input image and with a feel of a given style image? For example, how can we stylize an input image as if it was created by Vincent Van Gogh? Or is it possible, given a single instance of texture, generate more pictures with the same texture or extrapolate original texture image? To tackle these problems, one needs to be able to generate objects (images in this case), rather than simply returning an answer for existing objects as discriminative models do. Having a generation mechanism, we either directly solve some of the problems (e.g., texture synthesis, where the aim is essentially to generate an image), or make it possible to approach downstream tasks (e.g., outliers detection). The first part of this thesis focuses on the development of algorithms for fast, diverse, and high-quality texture synthesis and style transfer (chapters 2 and 3).

A big step in image generation has been made with the introduction of Variational Autoencoders (VAE) [Kingma and Welling, 2014] and Generative Adversarial Networks (GAN) [Goodfellow et al., 2014]. These models enabled a large number of applications that were not possible before. For example, an improved version of a GAN [Karras et al., 2018b, 2017b] can generate high-detailed faces, that a person would struggle to distinguish from real ones. A GAN-based model *pix2pix* [Isola et al., 2017a] lifted the quality in the image-to-image problems to a new level. With *pix2pix*, one can turn a pencil drawing into a colorful realistic image or turn a layout of a building into a rendering of that building. CycleGAN [Zhu et al., 2017a] provides a mechanism for conditional image generation, while it does not require a dataset with pairs for training as *pix2pix* does. Having two independent sets of images, for example, a set of horses and zebras, CycleGAN can learn to turn horses into zebras and back while maintaining background. Amazingly, CycleGAN learns to do so without any input from a human – that is, no semantic information or pairs before/after are presented.

GAN and VAE allow to efficiently model the data distribution and sample from it. In particular, both GAN and VAE learn a mapping that transforms easy-to-draw latent distribution into the data distribution, while only VAE learns to project data to the latent space. At the same time, GAN recommended itself as a much better model for image generation – it can generate crisp and good-looking samples, while VAE usually ends up with blurry ones. The second part of the thesis concentrates on a model called Adversarial Generative-Encoder Networks (AGE nets) (chapter 4). AGE uses the same learning principles as GAN but constructs both forward and backward mappings similar

to VAE. AGE allows latent code inference while maintaining the synthesis quality of a GAN.

In the GAN framework, Goodfellow *et al.* use an additional network called *descriminator* to train the generator network. The discriminator takes the images from the dataset and the ones created by the generator, compares them, and guides the generator learning towards better sampling. Thus, the discriminator should be designed carefully to enable efficient learning. In chapter 5, we describe a novel architecture of the discriminator network based on features of a pretrained network [Simonyan and Zisserman, 2014, Johnson et al., 2016], which results in better generator quality.

While the convolutional neural network (ConvNets) is the most popular tool for both image generation and recognition, it is still unclear why this particular architecture works well in practice. Is it because of a specific property of the convolutional architecture or due to learning on a large dataset? In chapter 7, we are taking a step towards answering this question. We separate the learning process from the prior induced by the architecture of the network and show that the convolutional structure of the neural network itself is sufficient to achieve good results at the image restoration tasks without learning on a dataset.

A great value is found by injecting convolutional neural networks into the pipelines, that did not change for ages and became a gold standard. For example, to render an object from the real world, conventionally, one would first reconstruct the geometry and the texture of the object, set up virtual lighting, and use a rendering engine to get the resulting image. The process takes a lot of human labour, including cleansing the reconstructed geometry and setting up the parameters of the scene so that the object looks natural when rendered. Moreover, for a photo-realistic result, a sophisticated physically-based rendering engine is needed. These engines are very compute-intensive and are not ready or even close to being ready for real-time applications. Interestingly, neural networks can be used to replace the reconstruction pipeline partially or even entirely. In chapter 6, we use neural networks for real-time, photo-realistic rendering of a human avatar, driven by an actor. Our approach does not require manual 3D modeling and almost entirely relies on convolutional neural networks.

1.2 Overview

1.2.1 Texture Networks: Feed-forward Synthesis of Textures and Stylized Images

In this work, we propose an image stylization and texture synthesis method that improves state-of-the-art methods of Gatys et al. [2015b,c]. It reaches comparable quality, yet we reduce the processing time by 500x. We found that an iterative generation process of Gatys et al. [2015b,c] requires repeated evaluation of a complex deep neural network-based function and its derivative, resulting in a long generation time. With a total number of iterations about 1000 the entire process takes several minutes on a high-end graphics processing unit. Differently to their method, our method requires only a single forward pass of a small convolutional neural network to synthesize a texture (stylized image). An improved inference time comes at the cost of spending resources on training a generator network. Yet, the training process takes only several hours per texture or style and performed only once.

For texture synthesis, we train a neural network to transform samples from an easy-to-draw distribution to instances of a given texture using the loss proposed in [Gatys et al., 2015b]. For style transfer, the network takes the content image as input and learns to output a stylized version of the input image. We use the same loss as in [Gatys et al., 2015c] to guide the training. The learning process takes several hours but executed once per style image. With a trained network, the generation is almost instant and can even be performed on a mobile device, enabling a wide range of applications.

We performed a quantitative and qualitative comparison of our results to [Gatys et al., 2015c]. We found a similar qualitative and quantitative performance in terms of stylization and texture synthesis quality while improving the runtime by 500x.

1.2.2 Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis

We base on our previous work, "Texture networks," and improve it in several aspects. We first note that in "Texture networks," the *diversity* in texture sampling is delivered only thanks to a specific architecture of the generator. Such a generator is a "shallow" one, and the neurons at the end of it have a limited receptive field (the end neurons do not "observe" the whole input). This property allows the network to generate different textures if fed with different data. Interestingly, we show that, in general, a sufficiently deep network is prone to collapsing any input into a single texture image.

While the direct optimization of the texture loss [Gatys et al., 2015b] with an optimization-based method allows generating diverse textures because of random initialization, the use of a similar loss in "Texture networks" for training a generator creates an issue. Using such loss for training permits the generator to map any input to a single image that delivers the global minima of the texture loss. That is, the generator is only required to sample images with high-quality texture (low texture loss), and it is not penalized for the lack of sample diversity. To alleviate this issue, we step back and revisit the foundation of the loss functional. We formulate the problem as a task of textures probability distribution approximation and derive a novel loss for training the generator. Our loss consists of two terms: the first one turns out to be equal to the texture loss used solely in "Texture Networks," and the second term is equal to negative entropy of the generator's output distribution. Minimization of such two-term loss pushes the generator to synthesize diverse samples, each being a high-quality texture instance. While we do not scale individual terms, the scaling can be used to find the desired trade-off between quality and diversity.

Our second contribution is targeting the *quality* of the "Texture Networks." We note that the texture/style loss essentially requires a generated image to have a predefined set of statistics extracted from the target texture or style image. We found that a conventional neural network with convolutional, pooling, ReLU blocks fails to approximate the desired statistics matching mapping efficiently. We introduce an *Instance Normalization* block, that normalizes the statistics of the features all across the generator, making it much easier for a generator to learn a mapping such that the statistics of an output image are decorrelated from the input statistics. We show huge qualitative and quantitative improvement over the stylization quality if compared to the previous methods and "Texture Networks," yet maintaining real-time performance.

1.2.3 It Takes (Only) Two: Adversarial Generator-Encoder Networks

Gatys et al. [2015b] proposed an energy surface for texture synthesis and style transfer, which can be converted into a texture distribution and evaluated at any point (for any image). Even though this distribution is not normalized, one can still efficiently sample from it, that is – generate textures or stylize images. But what if we want to generate other types of objects? The generation of generic objects is much more challenging since no probability distribution model exists for such a general case. In 2014, Goodfellow et al. [2014] proposed a framework called Generative Adversarial Networks (GAN), that can efficiently learn a data distribution automatically using only a dataset of samples from that distribution. They use a *generator* network to transform a latent distribution to the data distribution and use an additional *discriminator* network to generate a

learning signal to the generator by comparing the generation result with the images from the given dataset.

While being powerful at generating images, GAN lacks a mechanism for latent code inference. Several works [Donahue et al., 2017, Dumoulin et al., 2017] attempt to solve the issue by adding this mapping to the original GAN model. Nevertheless, they achieve the goal, their system involves three networks: a generator that maps a latent code to an image, an inference network that maps images to their codes, and a discriminator, used to train the first two.

In our work, we combine the discriminator with the inference network in a single network and propose a model called AGE. Differently to a standard GAN, where the discriminator essentially learns to distinguish the real images from the generated ones, our discriminator learns to transform both real distribution and the distribution of fake images into a fixed distribution P, for example, into a multi-dimensional uniform distribution on sphere. We show that if such transformed distribution made to coincide with P, and cycle consistency losses added, the discriminator would learn to map the input images to their latent codes.

We obtain a comparable generation quality to the baselines while using only two networks instead of three. Our system is less complicated while providing the same functionality, which facilitates further development of the methods in this area. Our model can learn any distribution directly from data extending the previously explored case of learning texture distribution.

1.2.4 Image Manipulation with Perceptual Discriminators

The *discriminator*, being an essential part of almost any GAN system, produces feedback to the generator and drives the generator's learning. It is crucial to design the discriminator in a way, that its feedback helps the generator to improve samples' aspects, essential to a human. For example, a deep discriminator might catch a very slight difference between fake and real images that a human eye would never notice. At the same time, poorly designed discriminator may completely ignore the discrepancy that a human would immediately recognize, for example, the difference in color.

We propose to use *perceptual features* [Johnson et al., 2016] instead of the images as a discriminator's input. A simple L2 metric defined in this feature space is proved [Zhang et al., 2018] to closely approximate perception of a human, and thus, those features are a good choice for image comparison. Technically, we employ a pretrained VGG-19 model [Simonyan and Zisserman, 2014] for feature extraction. The discriminator takes

deep features extracted from a preselected set of intermediate layers of VGG-19 network as input to compute the output during a forward pass. In a backward pass, the gradients w.r.t discriminator's input are propagated back through the VGG-19 to the generator, while not updating the gradient of VGG-19 weights and keeping the feature extractor network fixed. We show that this simple idea improves many of the existing pipelines both qualitatively and quantitatively.

1.2.5 Textured Neural Avatars

We revisit the image generation pipeline for the task of human rendering. We aim to render an avatar of a person from an *arbitrary standpoint*, with motions driven by the movements of a human from a video-sequence, shot from a *fixed position*. That is, for each frame, our system first extracts 3D pose of the actor, reprojects it to the desired view, and transforms it into a photo of the avatar, resulting in an image that looks as if it was shot from that camera. The system enables virtual/augmented reality telepresence applications and can be used to connect people all over the world in 3D compared to the existing conferencing systems based on 2D video.

The rending of a 3D human pose is done using a convolutional neural network. This network takes a representation of the 3D pose as input and renders the avatar in a given pose. The network is trained on a dataset of videos of moving people annotated with body feature points and background masks. Although it is possible to design the neural network to produce an RGB image from the input directly, we found this approach not to generalize well. Therefore, we employ a different concept, widely used in the classical rendering pipeline. Instead of estimating the colors in each pixel, our network generates an explicit 2D texture map of the human body surface and then warps a texture image of the body using the map. The texture images are learned simultaneously with the network and fixed in test time.

1.2.6 Deep Image prior

In this work, we show that a convolutional neural network's structure possesses a strong image prior. This property partially explains why the convolutional neural networks have become very successful models for image generation. Moreover, this prior can be used as a plug-in replacement for the existing image priors, such as total-variation prior and improve single-image image restoration performance. The exploration of image prior is also important for style transfer methods that strongly rely on the concept of image naturality. In particular, most of the existing style transfer methods, including the ones

proposed by the author, use total-variation image prior and seek better ways of imposing such prior.

In image restoration problems, the goal is to recover original image x having a corrupted image x_0. Such problems are often formulated as an optimization task:

$$\min_x E(x; x_0) + R(x)\,, \tag{1.1}$$

where $E(x; x_0)$ is a *data term* and $R(x)$ is an *image prior*. The data term $E(x; x_0)$ is usually easy to design for a wide range of problems, such as super-resolution, denoising, inpainting, while image prior $R(x)$ is a challenging one. Today's trend is to capture the prior $R(x)$ with a ConvNet by training it using large number of examples.

We first notice, that for a surjective $g : \theta \mapsto x$ the following procedure, in theory, is equivalent to (1.1):

$$\min_\theta E(g(\theta); x_0) + R(g(\theta))\,. \tag{1.2}$$

In practice, g dramatically changes how the image space is searched by an optimization method. Furthermore, by selecting a "good" (possibly injective) mapping g, we could get rid of the prior term. We define $g(\theta)$ as $f_\theta(z)$, where f is a deep ConvNet with parameters θ and z is a fixed input, leading to the formulation

$$\min_\theta E(f_\theta(z); x_0)\,. \tag{1.3}$$

Here, the network f_θ is initialized randomly, and input z is filled with noise and fixed.

In other words, instead of searching for the optima in the image space, we now search for it in the space of the neural network's parameters. We emphasize that we never use a pretrained network or an image database. Only a corrupted image x_0 is used in the restoration process.

From a practical perspective, our method can be used for image processing tasks. Compared to other single-image models, our method obtains superior quality on both synthetic and real-world data. We test our approach on image denoising, super-resolution, image inpainting benchmarks, and achieve state-of-the-art quantitative and qualitative results.

Chapter 2

Texture Networks: Feed-forward Synthesis of Textures and Stylized Images

Abstract

Gatys *et al.* recently demonstrated that deep networks can generate beautiful textures and stylized images from a single texture example. However, their methods requires a slow and memory-consuming optimization process. We propose here an alternative approach that moves the computational burden to a learning stage. Given a single example of a texture, our approach trains compact feed-forward convolutional networks to generate multiple samples of the same texture of arbitrary size and to transfer artistic style from a given image to any other image. The resulting networks are remarkably light-weight and can generate textures of quality comparable to Gatys *et al.*, but hundreds of times faster. More generally, our approach highlights the power and flexibility of generative feed-forward models trained with complex and expressive loss functions.

This work was published as: Dmitry Ulyanov, Andrea Vedaldi and Victor Lempitsky. *Texture Networks: Feed-forward Synthesis of Textures and Stylized Images.* International Conference on Machine Learning (ICML), 2016

FIGURE 2.1: Texture networks proposed in this work are feed-forward architectures capable of learning to synthesize complex textures based on a single training example. The perceptual quality of the feed-forwardly generated textures is similar to the results of the closely related method suggested in [Gatys et al., 2015b], which use slow optimization process.

2.1 Introduction

Several recent works demonstrated the power of deep neural networks in the challenging problem of *generating images*. Most of these proposed **generative networks** that produce images as output, using feed-forward calculations from a random seed; however, very impressive results were obtained by Gatys et al. [2015b,c] by using **networks descriptively**, as image statistics. Their idea is to reduce image generation to the problem of sampling at random from the set of images that match a certain statistics. In **texture synthesis** [Gatys et al., 2015b], the reference statistics is extracted from a single example of a visual texture, and the goal is to generate further examples of that texture. In **style transfer** [Gatys et al., 2015c], the goal is to match simultaneously the visual style of a first image, captured using some low-level statistics, and the visual content of a second image, captured using higher-level statistics. In this manner, the style of an image can be replaced with the one of another without altering the overall semantic content of the image.

Matching statistics works well in practice, is conceptually simple, and demonstrates that off-the-shelf neural networks trained for generic tasks such as image classification can be re-used for image generation. However, the approach of Gatys et al. [2015b,c] has certain shortcomings too. Being based on an *iterative optimization procedure*, it requires backpropagation to gradually change the values of the pixels until the desired statistics

is matched. This iterative procedure requires several seconds in order to generate a relatively small image using a high-end GPU, while scaling to large images is problematic because of high memory requirements. By contrast, feed-forward generation networks can be expected to be much more efficient because they require a single evaluation of the network and do not incur in the cost of backpropagation.

In this work we look at the problem of achieving the synthesis and stylization capability of descriptive networks using feed-forward generation networks. Our contribution is threefold. First, we show for the first time that a generative approach can produce textures of the quality and diversity comparable to the descriptive method. Second, we propose a generative method that is two orders of magnitude faster and one order of magnitude more memory efficient than the descriptive one. Using a single forward pass in networks that are remarkably compact make our approach suitable for video-related and possibly mobile applications. Third, we devise a new type of multi-scale generative architecture that is particularly suitable for the tasks we consider.

The resulting fully-convolutional networks (that we call *texture networks*) can generate textures and process images of arbitrary size. Our approach also represents an interesting showcase of training conceptually-simple feed-forward architectures while using complex and expressive loss functions. We believe that other interesting results can be obtained using this principle.

The rest of the work provides the overview of the most related approaches to image and texture generation (Section 2.2), describes our approach (Section 2.3), and provides extensive extensive qualitative comparisons on challenging textures and images (Section 2.4).

2.2 Background and related work

2.2.1 Image generation using neural networks

In general, one may look at the process of generating an image $\mathbf{x}$ as the problem of drawing a sample from a certain distribution $p(\mathbf{x})$. In texture synthesis, the distribution is induced by an example texture instance $\mathbf{x_0}$ (*e.g.* a polka dots image), such that we can write $\mathbf{x} \sim p(\mathbf{x}|\mathbf{x_0})$. In style transfer, the distribution is induced by an image $\mathbf{x_0}$ representative of the visual style (*e.g.* an impressionist painting) and a second image $\mathbf{x}_1$ representative of the visual content (*e.g.* a boat), such that $\mathbf{x} \sim p(\mathbf{x}|\mathbf{x_0}, \mathbf{x}_1)$.

Mahendran and Vedaldi [2015], Gatys et al. [2015b,c] reduce this problem to the one of finding a *pre-image* of a certain image statistics $\Phi(\mathbf{x}) \in \mathbb{R}^d$ and pose the latter as an

optimization problem. In particular, in order to synthesize a texture from an example image $\mathbf{x_0}$, the pre-image problem is:

$$\underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmin}} \|\Phi(\mathbf{x}) - \Phi(\mathbf{x_0})\|_2^2. \tag{2.1}$$

Importantly, the pre-image $\mathbf{x} : \Phi(\mathbf{x}) \approx \Phi(\mathbf{x_0})$ is usually not unique, and sampling pre-images achieves diversity. In practice, samples are extracted using a local optimization algorithm $\mathcal{A}$ starting from a random initialization $\mathbf{z}$. Therefore, the generated image is the output of the function

$$\underset{\mathbf{x}\in\mathcal{X}}{\operatorname{localopt}}(\|\Phi(\mathbf{x}) - \Phi(\mathbf{x_0})\|_2^2; \mathcal{A}, \mathbf{z}), \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \Sigma). \tag{2.2}$$

This results in a distribution $p(\mathbf{x}|\mathbf{x_0})$ which is difficult to characterise, but is easy to sample and, for good statistics Φ, produces visually pleasing and diverse images. Both Mahendran and Vedaldi [2015] and Gatys et al. [2015b,c] base their statistics on the response that $\mathbf{x}$ induces in deep neural network layers. Our approach reuses in particular the statistics based on correlations of convolutional maps proposed by Gatys et al. [2015b,c].

2.2.2 Descriptive texture modelling

The approach described above has strong links to many well-known models of visual textures. For texture, it is common to assume that $p(\mathbf{x})$ is a stationary *Markov random field* (MRF). In this case, the texture is ergodic and one may considers local spatially-invariant statistics $\psi \circ F(\mathbf{x}; i), i \in \Omega$, where i denotes a spatial coordinate. Often F is the output of a bank of linear filters and ψ an histogramming operator. Then the spatial average of this local statistics on the prototype texture $\mathbf{x_0}$ approximates its sample average

$$\phi(\mathbf{x_0}) = \frac{1}{|\Omega|} \sum_{i=1}^{|\Omega|} \psi \circ F(\mathbf{x_0}; i) \approx \underset{\mathbf{x}\sim p(\mathbf{x})}{E} [\psi \circ F_l(\mathbf{x}; 0)]. \tag{2.3}$$

The FRAME model of Zhu et al. [1998] uses this fact to induce the maximum-entropy distribution over textures $p(\mathbf{x}) \propto \exp(-\langle \lambda, \phi(\mathbf{x}) \rangle)$, where λ is a parameter chosen so that the marginals match their empirical estimate, i.e. $E_{\mathbf{x}\sim p(\mathbf{x})}[\phi(\mathbf{x})] = \phi(\mathbf{x_0})$.

A shortcoming of FRAME is the difficulty of sampling from the maxent distribution. Portilla and Simoncelli [2000] addresses this limitation by proposing to directly find images $\mathbf{x}$ that match the desired statistics $\Phi(\mathbf{x}) \approx \Phi(\mathbf{x_0})$, pioneering the pre-image method of (2.1).

Where Zhu et al. [1998], Portilla and Simoncelli [2000] use linear filters, wavelets, and histograms to build their texture statistics, Mahendran and Vedaldi [2015], Gatys et al. [2015b,b] extract statistics from pre-trained deep neural networks. [Gatys et al., 2015c] differs also in that it considers the style transfer problem instead of the texture synthesis one.

2.2.3 Generator deep networks

An alternative to using a neural networks as descriptors is to construct generator networks $\mathbf{x} = \mathbf{g}(\mathbf{z})$ that produce *directly* an image $\mathbf{x}$ starting from a vector of random or deterministic parameters $\mathbf{z}$.

Approaches such as [Dosovitskiy et al., 2015] learn a mapping from deterministic parameters $\mathbf{z}$ (*e.g.* the type of object imaged and the viewpoint) to an image $\mathbf{x}$. This is done by fitting a neural network to minimize the discrepancy $\|\mathbf{x}_i - \mathbf{g}(\mathbf{z}_i)\|$ for known image-parameter pairs $(\mathbf{x}_i, \mathbf{z}_i)$. While this may produce visually appealing results, it requires to know the relation $(\mathbf{x}, \mathbf{z})$ beforehand and cannot express any diversity beyond the one captured by the parameters.

An alternative is to consider a function $\mathbf{g}(\mathbf{z})$ where the parameters $\mathbf{z}$ are unknown and are sampled from a (simple) random distribution. The goal of the network is to map these random values to plausible images $\mathbf{x} = \mathbf{g}(\mathbf{z})$. This requires measuring the quality of the sample, which is usually expressed as a distance between $\mathbf{x}$ and a set of example images $\mathbf{x}_1, \ldots, \mathbf{x}_n$. The key challenge is that the distance must be able to generalize significantly from the available examples in order to avoid penalizing sample diversity.

Generative Adversarial Networks (GAN; [Goodfellow et al., 2014]) address this problem by training, together with the generator network $\mathbf{g}(\mathbf{z})$, a second *adversarial* network $f(\mathbf{x})$ that attempts to distinguish between samples $\mathbf{g}(\mathbf{z})$ and natural image samples. Then f can be used as a measure of quality of the samples and $\mathbf{g}$ can be trained to optimize it. LAPGAN [Denton et al., 2015] applies GAN to a Laplacian pyramid of convolutional networks and DCGAN [Radford et al., 2016] further optimizes GAN and learn is from very large .sets.

2.2.4 Moment matching networks

The maximum entropy model of Zhu et al. [1998] is closely related to the idea of *Maximum Mean Discrepancy* (MMD) introduced in [Gretton et al., 2006]. Their key observation the expected value $\mu_p = E_{\mathbf{x}\sim p(\mathbf{x})}[\phi(\mathbf{x})]$ of certain statistics $\phi(\mathbf{x})$ uniquely identifies

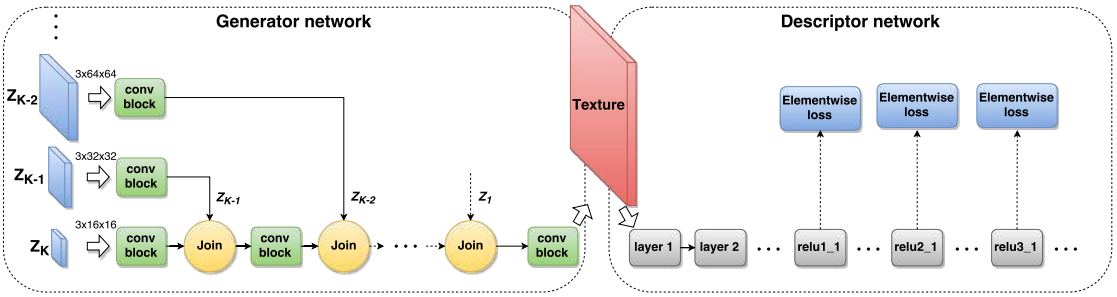


FIGURE 2.2: Overview of the proposed architecture (texture networks). We train a *generator network* (left) using a powerful loss based on the correlation statistics inside a fixed pre-trained *descriptor network* (right). Of the two networks, only the generator is updated and later used for texture or image synthesis. The `conv` block contains multiple convolutional layers and non-linear activations and the `join` block upsampling and channel-wise concatenation. Different branches of the generator network operate at different resolutions and are excited by noise tensors z_i of different sizes.

the distribution p. Li et al. [2015], Dziugaite et al. [2015] derive from it a loss function alternative to GAN by comparing the statistics averaged over network samples $\frac{1}{m}\sum_{i=1}^{m}\phi\circ\mathbf{g}(\mathbf{z}_i)$ to the statistics averaged over empirical samples $\frac{1}{m}\sum_{i=1}^{m}\phi(\mathbf{x}_i)$. They use it to train a *Moment Matching Network* (MMN) and apply it to generate small images such as MNIST digits. Our networks are similar to moment matching networks, but use very specific statistics and applications quite different from the considered in [Li et al., 2015, Dziugaite et al., 2015].

2.3 Texture networks

We now describe the proposed method in detail. At a high-level (see Figure 2.2), our approach is to train a feed-forward **generator network** $\mathbf{g}$ which takes a noise sample $\mathbf{z}$ as input and produces a texture sample $\mathbf{g}(\mathbf{z})$ as output. For style transfer, we extend this *texture network* to take both a noise sample $\mathbf{z}$ and a content image $\mathbf{y}$ and then output a new image $\mathbf{g}(\mathbf{y},\mathbf{z})$ where the texture has been applied to $\mathbf{y}$ as a visual style. A separate generator network is trained for each texture or style and, once trained, it can synthesize an arbitrary number of images of arbitrary size in an efficient, feed-forward manner.

A key challenge in training the generator network $\mathbf{g}$ is to construct a loss function that can assess automatically the quality of the generated images. For example, the key idea of GAN is to *learn* such a loss along with the generator network. We show in Sect. 2.3.1 that a very powerful loss can be derived from pre-trained and fixed **descriptor networks** using the statistics introduced in [Gatys et al., 2015b,c]. Given the loss, we then discuss the architecture of the generator network for texture synthesis (Sect. 2.3.2) and then generalize it to style transfer (Sect 2.3.3).

2.3.1 Texture and content loss functions

Our loss function is derived from [Gatys et al., 2015b,c] and compares image statistics extracted from a fixed pre-trained descriptor CNN (usually one of the VGG CNN [Simonyan and Zisserman, 2014, Chatfield et al., 2014] which are pre-trained for image classification on the ImageNet ILSVRC 2012 .). The descriptor CNN is used to measure the mismatch between the *prototype* texture $\mathbf{x_0}$ and the generated image $\mathbf{x}$. Denote by $F_i^l(\mathbf{x})$ the i-th map (feature channel) computed by the l-th convolutional layer by the descriptor CNN applied to image $\mathbf{x}$. The *Gram matrix* $G^l(\mathbf{x})$ is defined as the matrix of scalar (inner) products between such feature maps:

$$G_{ij}^l(\mathbf{x}) = \langle F_i^l(\mathbf{x}), F_j^l(\mathbf{x}) \rangle \,. \tag{2.4}$$

Given that the network is convolutional, each inner product implicitly sums the products of the activations of feature i and j at all spatial locations, computing their (unnormalized) empirical correlation. Hence $G_{ij}^l(\mathbf{x})$ has the same general form as (2.3) and, being an orderless statistics of local stationary features, can be used as a texture descriptor.

In practice, Gatys et al. [2015b,c] use as texture descriptor the combination of several Gram matrices $G^l, l \in L_T$, where L_T contains selected indices of convolutional layer in the descriptor CNN. This induces the following *texture loss* between images $\mathbf{x}$ and $\mathbf{x_0}$:

$$\mathcal{L}_T(\mathbf{x}; \mathbf{x_0}) = \sum_{l \in L_T} \|G^l(\mathbf{x}) - G^l(\mathbf{x_0})\|_2^2 \,. \tag{2.5}$$

In addition to the texture loss (2.5), [Gatys et al., 2015c] propose to use as *content loss* the one introduced by [Mahendran and Vedaldi, 2015], which compares images based on the output $F_i^l(\mathbf{x})$ of certain convolutional layers $l \in L_C$ (without computing further statistics such as the Gram matrices). In formulas

$$\mathcal{L}_C(\mathbf{x}; \mathbf{y}) = \sum_{l \in L_C} \sum_{i=1}^{N_l} \|F_i^l(\mathbf{x}) - F_i^l(\mathbf{y})\|_2^2 \,, \tag{2.6}$$

where N_l is the number of maps (feature channels) in layer l of the descriptor CNN. The key difference with the texture loss (2.5) is that the content loss compares feature activations at corresponding spatial locations, and therefore preserves spatial information. Thus this loss is suitable for content information, but not for texture information.

Analogously to [Gatys et al., 2015b], we use the texture loss (2.5) alone when training a generator network for texture synthesis, and we use a weighted combination of the texture loss (2.5) and the content loss (2.6) when training a generator network for

FIGURE 2.3: Our approach can also train feed-forward networks to transfer style from artistic images (left). After training, a network can transfer the style to any new image (e.g. right) while preserving semantic content. For some styles (bottom row), the perceptual quality of the result of our feed-forward transfer is comparable with the optimization-based method [Gatys et al., 2015c], though for others the results are not as impressive (top row).

stylization. In the latter case, the set L_C does not includes layers as shallow as the set L_T as only the high-level content should be preserved.

2.3.2 Generator network for texture synthesis

We now discuss the architecture and the training procedure for the generator network $\mathbf{g}$ for the task of texture synthesis. We denote the parameters of the generator network as θ. The network is trained to transform a noise vector $\mathbf{z}$ sampled from a certain distribution $\mathcal{Z}$ (which we set to be uniform i.i.d.) into texture samples that match, according to the texture loss (2.5), a certain prototype texture $\mathbf{x_0}$:

$$\theta_{\mathbf{x_0}} = \underset{\theta}{\operatorname{argmin}} \; E_{\mathbf{z}\sim\mathcal{Z}} \left[\mathcal{L}_T \left(\mathbf{g}(\mathbf{z};\theta), \, \mathbf{x_0} \right) \right] . \tag{2.7}$$

2.3.2.1 Network architecture

We experimented with several architectures for the generator network $\mathbf{g}$. The simplest are chains of convolutional, non-linear activation, and upsampling layers that start from a noise sample $\mathbf{z}$ in the form of a small feature map and terminate by producing an image. While models of this type produce reasonable results, we found that *multi-scale architectures* result in images with smaller texture loss and better perceptual quality while using fewer parameters and training faster. Figure 2.2 contains a high-level representation of our reference multi-scale architecture, which we describe next.

The reference texture $\mathbf{x_0}$ is a tensor $\mathbb{R}^{M\times M\times 3}$ containing three color channels. For simplicity, assume that the spatial resolution M is a power of two. The input noise $\mathbf{z}$ comprises K random tensors $\mathbf{z}_i \in \mathbb{R}^{\frac{M}{2^i}\times\frac{M}{2^i}}$, $i = 1, 2, \ldots, K$ (we use $M = 256$ and $K = 5$) whose entries are i.i.d. sampled from a uniform distribution. Each random noise tensor is first processed by a sequence of convolutional and non-linear activation layers, then upsampled by a factor of two, and finally concatenated as additional feature channels to the partially processed tensor from the scale below. The last full-resolution tensor is ultimately mapped to an RGB image $\mathbf{x}$ by a bank of 1×1 filters.

Each convolution block in Figure 2.2 contains three convolutional layers, each of which is followed by a ReLU activation layer. The convolutional layers contain respectively 3×3, 3×3 and 1×1 filters. Filers are computed densely (stride one) and applied using circular convolution to remove boundary effects, which is appropriate for textures. The number of feature channels, which equals the number of filters in the preceding bank, grows from a minimum of 8 to a maximum of 40.

Upsampling layers use simple nearest-neighbour interpolation (we also experimented strided full-convolution [Long et al., 2015a, Radford et al., 2016], but the results were not satisfying). We found that training benefited significantly from inserting batch normalization layers [Ioffe and Szegedy, 2015] right after each convolutional layer and, most importantly, right before the concatenation layers, since this balances gradients travelling along different branches of the network.

2.3.2.2 Learning

Learning optimizes the objective (2.7) using stochastic gradient descent (SGD). At each iteration, SGD draws a mini-batch of noise vectors $\mathbf{z}_k, k = 1, \ldots, B$, performs forward evaluation of the generator network to obtained the corresponding images $\mathbf{x}_k = \mathbf{g}(\mathbf{z}_k, \theta)$, performs forward evaluation of the descriptor network to obtain Gram matrices $G^l(\mathbf{x}_k), l \in L_T$, and finally computes the loss (2.5) (note that the corresponding terms $G^l(\mathbf{x_0})$ for the reference texture are constant). After that, the gradient of the texture loss with respect to the generator network parameters θ is computed using backpropagation, and the gradient is used to update the parameters. Note that LAPGAN [Denton et al., 2015] also performs multi-scale processing, but uses layer-wise training, whereas our generator is trained end-to-end.

Input | Gatys *et al.* | Texture nets (ours) | Portilla, Simoncelli | DCGAN

FIGURE 2.4: Further comparison of textures generated with several methods including the original statistics matching method [Portilla and Simoncelli, 2000] and the DCGAN [Radford et al., 2016] approach. Overall, our method and [Gatys et al., 2015b] provide better results, our method being hundreds times faster.

2.3.3 Style transfer

In order to extend the method to the task of image stylization, we make several changes. Firstly, the generator network $\mathbf{x} = \mathbf{g}(\mathbf{y}, \mathbf{z}; \theta)$ is modified to take as input, in addition to the noise variable $\mathbf{z}$, the image $\mathbf{y}$ to which the noise should be applied. The generator network is then trained to output an image $\mathbf{x}$ that is close in content to $\mathbf{y}$ and in texture/style to a reference texture $\mathbf{x_0}$. For example, $\mathbf{y}$ could be a photo of a person, and $\mathbf{x_0}$ an impressionist painting.

2.3.3.1 Network architecture

The architecture is the same as the one used for texture synthesis with the important difference that now the noise tensors $\mathbf{z}_i, i = 1, \ldots, K$ at the K scales are concatenated (as additional feature channels) with downsampled versions of the input image $\mathbf{y}$. For this application, we found beneficial to increased the number of scales from $K = 5$ to $K = 6$.

2.3.3.2 Learning

Learning proceeds by sampling noise vectors $\mathbf{z}_i \sim \mathcal{Z}$ and natural images $\mathbf{y}_i \sim \mathcal{Y}$ and then adjusting the parameters θ of the generator $\mathbf{g}(\mathbf{y}_i, \mathbf{z}_i; \theta)$ in order to minimize the

$k = 0.01$ $k = 0.1$ $k = 1$ $k = 10$

FIGURE 2.5: Our architecture for image stylization takes the content image and the noise vector as inputs. By scaling the input noise by different factors k we can affect the balance of style and content in the output image without retraining the network.

combination of content and texture loss:

$$\theta_{\mathbf{x_0}} = \underset{\theta}{\operatorname{argmin}} \; E_{\mathbf{z}\sim\mathcal{Z};\,\mathbf{y}\sim\mathcal{Y}} \left[\mathcal{L}_T \left(\mathbf{g}(\mathbf{y}, \mathbf{z}; \theta), \, \mathbf{x_0} \right) + \alpha \, \mathcal{L}_C \left(\mathbf{g}(\mathbf{y}, \mathbf{z}; \theta), \, \mathbf{y} \right) \right].$$

Here $\mathcal{Z}$ is the same noise distribution as for texture synthesis, $\mathcal{Y}$ empirical distribution on naturals image (obtained from any image collection), and α a parameter that trades off preserving texture/style and content. In practice, we found that learning is surprisingly resilient to overfitting and that it suffices to approximate the distribution on natural images $\mathcal{Y}$ with a very small pool of images (e.g. 16). In fact, our qualitative results degraded using too many example images. We impute this to the fact that stylization by a convolutional architecture uses local operations; since the same local structures exist in different combinations and proportions in different natural images $\mathbf{y}$, it is difficult for local operators to match in all cases the overall statistics of the reference texture $\mathbf{x_0}$, where structures exist in a fixed arbitrary proportion. Despite this limitation, the perceptual quality of the generated stylized images is usually very good, although for some styles we could not match the quality of the original stylization by optimization of [Gatys et al., 2015c].

2.4 Experiments

2.4.1 Further technical details

The generator network weights were initialized using Xavier's method. Training used *Torch7*'s implementation of Adam [Kingma and Ba, 2015], running it for 2000 iteration. The initial learning rate of 0.1 was reduced by a factor 0.7 at iteration 1000 and then again every 200 iterations. The batch size was set to 16. Similar to [Gatys et al., 2015b], the texture loss uses the layers $L_T = \{\texttt{relu1\_1}, \texttt{relu2\_1}, \texttt{relu3\_1}, \texttt{relu4\_1}, \texttt{relu5\_1}\}$ of VGG-19 and the content loss the layer $L_C = \{\texttt{relu4\_2}\}$. Fully training a single model

required just two hours on an NVIDIA Tesla K40, and visually appealing results could be obtained much faster, after just a few epochs.

2.4.2 Texture synthesis

We compare our method to [Gatys et al., 2015b,c] using the popular implementation of [Johnson, 2015], which produces comparable if not better results than the implementation eventually released by the authors. We also compare to the DCGAN [Radford et al., 2016] version of adversarial networks [Goodfellow et al., 2014]. Since DCGAN training requires multiple example images for training, we extract those as sliding 64×64 patches from the 256×256 reference texture $\mathbf{x_0}$; then, since DCGAN is fully convolutional, we use it to generate larger 256×256 images simply by inputting a larger noise tensor. Finally, we compare to [Portilla and Simoncelli, 2000].

Figure 2.4 shows the results obtained by the four methods on two challenging textures of [Portilla and Simoncelli, 2000]. Qualitatively, our generator CNN and [Gatys et al., 2015b]'s results are comparable and superior to the other methods; however, the generator CNN is much more efficient (see Sect. 2.4.4). Figure 2.1 includes further comparisons between the generator network and [Gatys et al., 2015b].

2.4.3 Style transfer

For training, example natural images were extracted at random from the ImageNet ILSVRC 2012 .. As for the original method of [Gatys et al., 2015c], we found that style transfer is sensitive to the trade-off parameter α between texture and content loss in (2.6). At test time this parameter is not available in our method, but we found that the trade-off can still be adjusted by changing the magnitude of the input noise $\mathbf{z}$ (see Figure 2.5).

We compared our method to the one of [Gatys et al., 2015c, Johnson, 2015] using numerous style and content images, including the ones in [Gatys et al., 2015c], and found that results are qualitatively comparable. Representative comparisons (using a fixed parameter α) are included in Figure 2.3. Other qualitative results are reported in Figure 2.7.

2.4.4 Speed and memory

We compare quantitatively the speed of our method and of the iterative optimization of [Gatys et al., 2015b] by measuring how much time it takes for the latter and for our generator network to reach a given value of the loss $\mathcal{L}_T(\mathbf{x}, \mathbf{x_0})$. Figure 2.6 shows

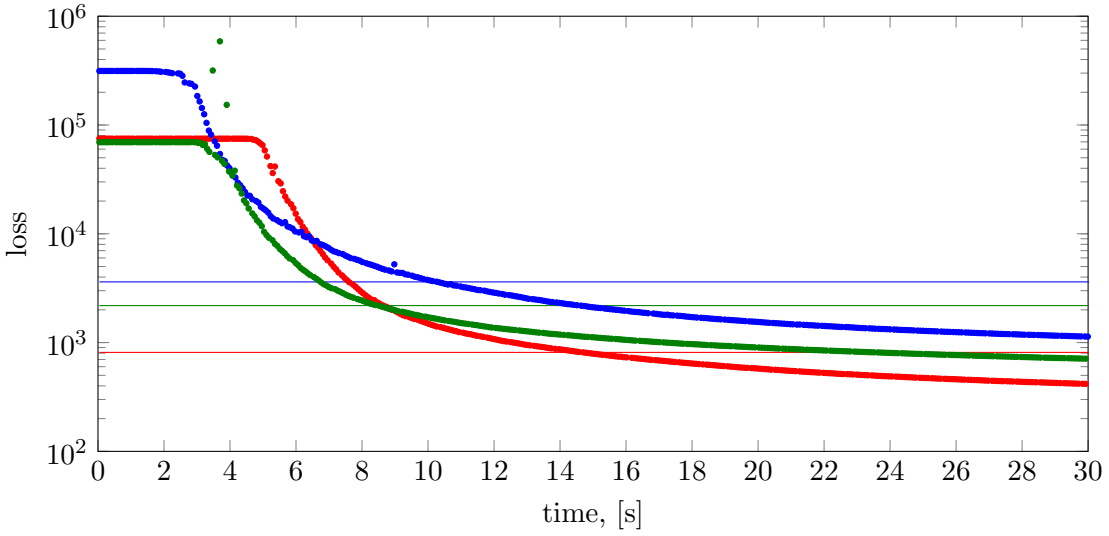


FIGURE 2.6: The objective values (log-scale) within the optimization-based method [Gatys et al., 2015b] for three randomly chosen textures are plotted as functions of time. Horizontal lines show the style loss achieved by our feedforward algorithm (mean over several samples) for the same textures. It takes the optimization within [Gatys et al., 2015b] around 10 seconds (500x slower than feedforward generation) to produce samples with comparable loss/objective.

that iterative optimization requires about 10 seconds to generate a sample $\mathbf{x}$ that has a loss comparable to the output $\mathbf{x} = \mathbf{g}(\mathbf{z})$ of our generator network. Since an evaluation of the latter requires ∼20ms, we achieve a 500× speed-up, which is sufficient for *real-time applications* such as video processing. There are two reasons for this significant difference: the generator network is much smaller than the VGG-19 model evaluated at each iteration of [Gatys et al., 2015b], and our method requires a single network evaluation. By avoiding backpropagation, our method also uses significantly less memory (170 MB to generate a 256×256 sample, vs 1100 MB of [Gatys et al., 2015b]).

2.5 Discussion

We have presented a new deep learning approach for texture synthesis and image stylization. Remarkably, the approach is able to generate complex textures and images in a purely feed-forward way, while matching the texture synthesis capability of [Gatys et al., 2015b], which is based on multiple forward-backward iterations. In the same vein as [Goodfellow et al., 2014, Dziugaite et al., 2015, Li et al., 2015], the success of this approach highlights the suitability of feed-forward networks for complex . generation and for solving complex tasks in general. The key to this success is the use of complex loss functions that involve different feed-forward architectures serving as "experts" assessing the performance of the feed-forward generator.

FIGURE 2.7: Stylization results for various styles and inputs (one network per row). Our approach can handle a variety of styles. The generated images are of `256x256` resolution and are computed in about 20 milliseconds each.

While our method generally obtains very good result for texture synthesis, going forward we plan to investigate better stylization losses to achieve a stylization quality comparable to [Gatys et al., 2015c] even for those cases (*e.g.* Figure 2.3.top) where our current method achieves less impressive results.

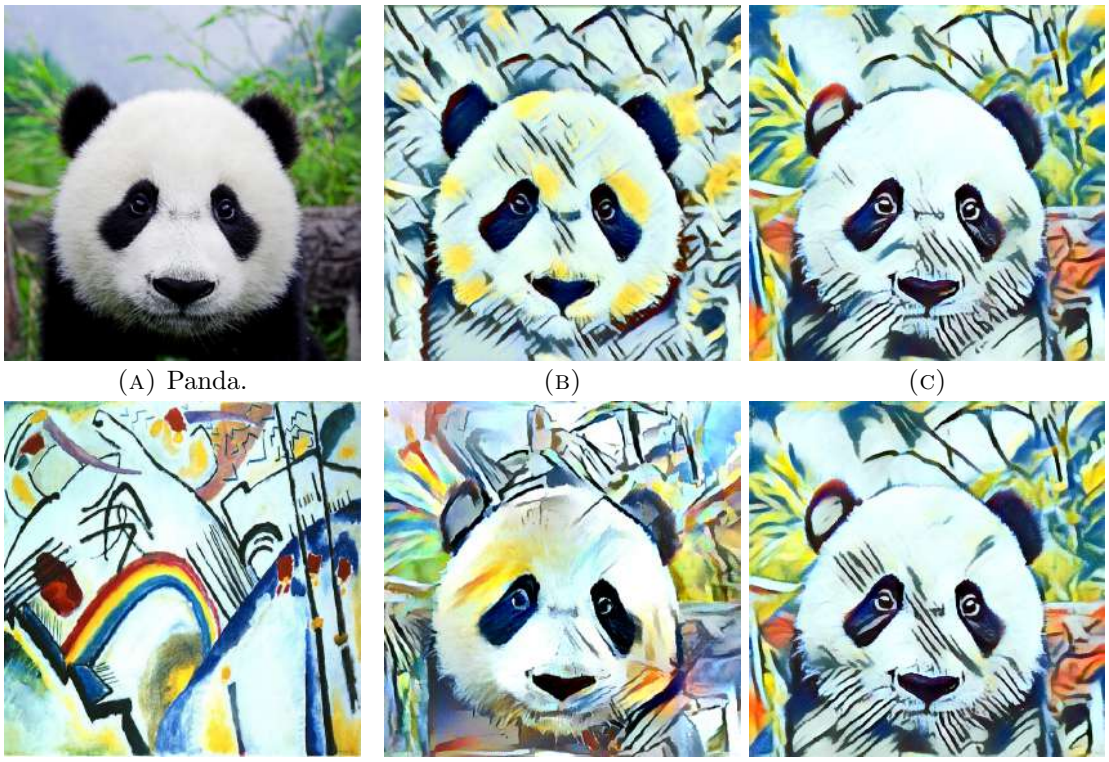
Chapter 3

Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis

Abstract

The recent work of Gatys *et al.*, who characterized the style of an image by the statistics of convolutional neural network filters, ignited a renewed interest in the texture generation and image stylization problems. While their image generation technique uses a slow optimization process, recently several authors have proposed to learn generator neural networks that can produce similar outputs in one quick forward pass. While generator networks are promising, they are still inferior in visual quality and diversity compared to generation-by-optimization. In this work, we advance them in two significant ways. First, we introduce an instance normalization module to replace batch normalization with significant improvements to the quality of image stylization. Second, we improve diversity by introducing a new learning formulation that encourages generators to sample unbiasedly from the Julesz texture ensemble, which is the equivalence class of all images characterized by certain filter responses. Together, these two improvements take feed forward texture synthesis and image stylization much closer to the quality of generation-via-optimization, while retaining the speed advantage.

This work was published as: Dmitry Ulyanov, Andrea Vedaldi and Victor Lempitsky. *Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis.* Computer Vision and Pattern Recognition (CVPR), 2017



(A) Panda. (B) (C)

(D) Style. (E) (F)

FIGURE 3.1: Which panda stylization seems the best to you? Definitely not the variant (b), which has been produced by a state-of-the-art algorithm among methods that take no longer than a second. The (e) picture took several minutes to generate using an optimization process, but the quality is worth it, isn't it? We would be particularly happy if you chose one from the rightmost two examples, which are computed with our new method that aspires to combine the quality of the optimization-based method and the speed of the fast one. Moreover, our method is able to produce diverse stylizations using a single network.

3.1 Introduction

The recent work of Gatys *et al.* [Gatys et al., 2015a,d], which used deep neural networks for texture synthesis and image stylization to a great effect, has created a surge of interest in this area. Following an eariler work by Portilla and Simoncelli [Portilla and Simoncelli, 2000], they generate an image by matching the second order moments of the response of certain filters applied to a reference texture image. The innovation of Gatys *et al.* is to use non-linear convolutional neural network filters for this purpose. Despite the excellent results, however, the matching process is based on local optimization, and generally requires a considerable amount of time (tens of seconds to minutes) in order to generate a single textures or stylized image.

In order to address this shortcoming, Ulyanov *et al.* [Ulyanov et al., 2016] and Johnson *et al.* [Johnson et al., 2016] suggested to replace the optimization process with feed-forward

generative convolutional networks. In particular, [Ulyanov et al., 2016] introduced *texture networks* to generate textures of a certain kind, as in [Gatys et al., 2015a], or to apply a certain texture style to an arbitrary image, as in [Gatys et al., 2015d]. Once trained, such texture networks operate in a feed-forward manner, three orders of magnitude faster than the optimization methods of [Gatys et al., 2015a,d].

The price to pay for such speed is a reduced performance. For texture synthesis, the neural network of [Ulyanov et al., 2016] generates good-quality samples, but these are not as diverse as the ones obtained from the iterative optimization method of [Gatys et al., 2015a]. For image stylization, the feed-forward results of [Ulyanov et al., 2016, Johnson et al., 2016] are qualitatively and quantitatively worse than iterative optimization. In this work, we address both limitations by means of two contributions, both of which extend beyond the applications considered in this work.

Our first contribution (section 3.4) is an architectural change that significantly improves the generator networks. The change is the introduction of an **instance-normalization layer** which, particularly for the stylization problem, greatly improves the performance of the deep network generators. This advance significantly reduces the gap in stylisation quality between the feed-forward models and the original iterative optimization method of Gatys *et al.*, both quantitatively and qualitatively.

Our second contribution (section 3.3) addresses the limited diversity of the samples generated by texture networks. In order to do so, we introduce a new formulation that learns generators that **uniformly sample the Julesz ensemble** [Zhu et al., 2000]. The latter is the equivalence class of images that match certain filter statistics. Uniformly sampling this set guarantees diverse results, but traditionally doing so required slow Monte Carlo methods [Zhu et al., 2000]; Portilla and Simoncelli, and hence Gatys *et al.*, cannot sample from this set, but only find individual points in it, and possibly just one point. Our formulation minimizes the Kullback-Leibler divergence between the generated distribution and a quasi-uniform distribution on the Julesz ensemble. The learning objective decomposes into a loss term similar to Gatys *et al.* minus the entropy of the generated texture samples, which we estimate in a differentiable manner using a non-parametric estimator [Kozachenko and Leonenko, 1987].

We validate our contributions by means of extensive quantitative and qualitative experiments, including comparing the feed-forward results with the gold-standard optimization-based ones (section 3.5). We show that, combined, these ideas dramatically improve the quality of feed-forward texture synthesis and image stylization, bringing them to a level comparable to the optimization-based approaches.

3.2 Background and related work

3.2.1 Julesz ensemble

Informally, a *texture* is a family of visual patterns, such as checkerboards or slabs of concrete, that share certain local statistical regularities. The concept was first studied by Julesz [Julesz, 1981], who suggested that the visual system discriminates between different textures based on the average responses of certain image filters.

The work of [Zhu et al., 2000] formalized Julesz' ideas by introducing the concept of *Julesz ensemble*. There, an image is a real function $x : \Omega \to \mathbb{R}^3$ defined on a discrete lattice $\Omega = \{1, \dots, H\} \times \{1, \dots, W\}$ and a texture is a distribution $p(x)$ over such images. The local statistics of an image are captured by a bank of (non-linear) filters $F_l : \mathcal{X} \times \Omega \to \mathbb{R}$, $l = 1, \dots, L$, where $F_l(x, u)$ denotes the response of filter F_l at location u on image x. The image x is characterized by the spatial average of the filter responses $\mu_l(x) = \sum_{u \in \Omega} F_l(x, u)/|\Omega|$. The image is perceived as a particular texture if these responses match certain characteristic values $\bar{\mu}_l$. Formally, given the loss function,

$$\mathcal{L}(x) = \sum_{l=1}^{L} (\mu_l(x) - \bar{\mu}_l)^2 \tag{3.1}$$

the Julesz ensemble is the set of all texture images

$$\mathcal{T}_\epsilon = \{x \in \mathcal{X} : \mathcal{L}(x) \leq \epsilon\}$$

that approximately satisfy such constraints. Since all textures in the Julesz ensemble are perceptually equivalent, it is natural to require the texture distribution $p(x)$ to be uniform over this set. In practice, it is more convenient to consider the exponential distribution

$$p(x) = \frac{e^{-\mathcal{L}(x)/T}}{\int e^{-\mathcal{L}(y)/T}\, dy}, \tag{3.2}$$

where $T > 0$ is a temperature parameter. This choice is motivated as follows [Zhu et al., 2000]: since statistics are computed from spatial averages of filter responses, one can show that, in the limit of infinitely large lattices, the distribution $p(x)$ is zero outside the Julesz ensemble and uniform inside. In this manner, eq. (3.2) can be though as a uniform distribution over images that have a certain characteristic filter responses $\bar{\mu} = (\bar{\mu}_1, \dots, \bar{\mu}_L)$.

Note also that the texture is completely described by the filter bank $F = (F_1, \dots, F_L)$ and their characteristic responses $\bar{\mu}$. As discussed below, the filter bank is generally fixed, so in this framework different textures are given by different characteristics $\bar{\mu}$.

3.2.2 Generation-by-minimization

For any interesting choice of the filter bank F, sampling from eq. (3.2) is rather challenging and classically addressed by Monte Carlo methods [Zhu et al., 2000]. In order to make this framework more practical, Portilla and Simoncelli [Portilla and Simoncelli, 2000] proposed instead to heuristically sample from the Julesz ensemble by the optimization process

$$x^* = \operatorname*{argmin}_{x \in \mathcal{X}} \mathcal{L}(x). \tag{3.3}$$

If this optimization problem can be solved, the minimizer x^* is by definition a texture image. However, there is no reason why this process should generate fair samples from the distribution $p(x)$. In fact, the only reason why eq. (3.3) may not simply return *always the same* image is that the optimization algorithm is randomly initialized, the loss function is highly non-convex, and search is local. Only because of this eq. (3.3) may land on different samples x^* on different runs.

3.2.3 Deep filter banks

Constructing a Julesz ensemble requires choosing a filter bank F. Originally, researchers considered the obvious candidates: Gaussian derivative filters, Gabor filters, wavelets, histograms, and similar [Zhu et al., 2000, Portilla and Simoncelli, 2000, Zhu et al., 1998]. More recently, the work of Gatys *et al.* [Gatys et al., 2015a,d] demonstrated that much superior filters are automatically learned by deep convolutional neural networks (CNNs) even when trained for apparently unrelated problems, such as image classification. In this work, in particular, we choose for $\mathcal{L}(x)$ the *style loss* proposed by [Gatys et al., 2015a]. The latter is the distance between the empirical correlation matrices of deep filter responses in a CNN.[1]

3.2.4 Stylization

The texture generation method of Gatys *et al.* [Gatys et al., 2015a] can be considered as a direct extension of the texture generation-by-minimization technique (3.3) of Portilla and Simoncelli [Portilla and Simoncelli, 2000]. Later, Gatys *et al.* [Gatys et al., 2015d] demonstrated that the same technique can be used to generate an image that mixes the statistics of two other images, one used as a texture template and one used as a content template. Content is captured by introducing a second loss $\mathcal{L}_{\text{cont.}}(x, x_0)$ that compares

[1] Note that such matrices are obtained by averaging local non-linear filters: these are the outer products of filters in a certain layer of the nerual network. Hence, the style loss of Gatys *et al.*. is in the same form as eq. (3.1).

the responses of deep CNN filters extracted from the generated image x and a content image x_0. Minimizing the combined loss $\mathcal{L}(x) + \alpha\mathcal{L}_{\text{cont.}}(x, x_0)$ yields impressive *artistic images*, where a texture $\bar{\mu}$, defining the artistic style, is fused with the content image x_0.

3.2.5 Feed-forward generator networks

For all its simplicity and efficiency compared to Markov sampling techniques, generation-by-optimization (3.3) is still relatively slow, and certainly too slow for real-time applications. Therefore, in the past few months several authors [Johnson et al., 2016, Ulyanov et al., 2016] have proposed to *learn generator neural networks* $g(z)$ that can directly map random noise samples $z \sim p_z = \mathcal{N}(0, I)$ to a local minimizer of eq. (3.3). Learning the neural network g amounts to minimizing the objective

$$g^* = \underset{g}{\operatorname{argmin}} \mathop{\mathbb{E}}_{p_z} \mathcal{L}(g(z)). \tag{3.4}$$

While this approach works well in practice, it shares the same important limitation as the original work of Portilla and Simoncelli: there is no guarantee that samples generated by g^* would be fair samples of the texture distribution (3.2). In practice, as we show in this work, such samples tend in fact to be not diverse enough.

Both [Johnson et al., 2016, Ulyanov et al., 2016] have also shown that similar generator networks work also for stylization. In this case, the generator $g(x_0, z)$ is a function of the content image x_0 and of the random noise z. The network g is learned to minimize the sum of texture loss and the content loss:

$$g^* = \underset{g}{\operatorname{argmin}} \mathop{\mathbb{E}}_{p_{x_0}, p_z} [\mathcal{L}(g(x_0, z)) + \alpha\mathcal{L}_{\text{cont.}}(g(x_0, z), x_0)]. \tag{3.5}$$

3.2.6 Alternative neural generator methods

There are many other techniques for image generation using deep neural networks.

The Julesz distribution is closely related to the FRAME maximum entropy model of [Zhu et al., 1998], as well as to the concept of *Maximum Mean Discrepancy* (MMD) introduced in [Gretton et al., 2006]. Both FRAME and MMD make the observation that a probability distribution $p(x)$ can be described by the expected values $\mu_\alpha = \mathbb{E}_{x\sim p(x)}[\phi_\alpha(x)]$ of a sufficiently rich set of statistics $\phi_\alpha(x)$. Building on these ideas, [Li et al., 2015, Dziugaite et al., 2015] construct generator neural networks g with the goal of minimizing the discrepancy between the statistics averaged over a batch of generated images

$\sum_{i=1}^{N} \phi_\alpha(g(z_i))/N$ and the statistics averaged over a traning set $\sum_{i=1}^{M} \phi_\alpha(x_i)/M$. The resulting networks g are called *Moment Matching Networks* (MMN).

An important alternative methodology is based on the concept of Generative Adversarial Networks (GAN; [Goodfellow et al., 2014]). This approach trains, together with the generator network $g(z)$, a second *adversarial* network $f(\cdot)$ that attempts to distinguish between generated samples $g(z), z \sim \mathcal{N}(0, I)$ and real samples $x \sim p_{\text{data}}(x)$. The adversarial model f can be used as a measure of quality of the generated samples and used to learn a better generator g. GAN are powerful but notoriously difficult to train. A lot of research is has recently focused on improving GAN or extending it. For instance, LAPGAN [Denton et al., 2015] combines GAN with a Laplacian pyramid and DCGAN [Radford et al., 2016] optimizes GAN for large datasets.

3.3 Julesz generator networks

This section describes our first contribution, namely a method to learn networks that draw samples from the Julesz ensemble modelling a texture (section 3.2), which is an intractable problem usually addressed by slow Monte Carlo methods [Zhu et al., 1998, 2000]. Generation-by-optimization, popularized by Portilla and Simoncelli and Gatys *et al.*, is faster, but can only find one point in the ensemble, not sample from it, with scarce sample diversity, particularly when used to train feed-forward generator networks [Johnson et al., 2016, Ulyanov et al., 2016].

Here, we propose a new formulation that allows to train generator networks that *sample the Julesz ensemble*, generating images with high visual fidelity as well as high diversity.

A generator network [Goodfellow et al., 2014] maps an i.i.d. noise vector $z \sim \mathcal{N}(0, I)$ to an image $x = g(z)$ in such a way that x is ideally a sample from the desired distribution $p(x)$. Such generators have been adopted for texture synthesis in [Ulyanov et al., 2016], but without guarantees that the learned generator $g(z)$ would indeed sample a particular distribution.

Here, we would like to sample from the Gibbs distribution (3.2) defined over the Julesz ensemble. This distribution can be written compactly as $p(x) = Z^{-1} e^{-\mathcal{L}(x)/T}$, where $Z = \int e^{-\mathcal{L}(x)/T}\, dx$ is an intractable normalization constant.

Denote by $q(x)$ the distribution induced by a generator network g. The goal is to make the target distribution p and the generator distribution q as close as possible by

minimizing their Kullback-Leibler (KL) divergence:

$$\begin{aligned} KL(q||p) &= \int q(x) \ln \frac{q(x)Z}{p(x)} \, dx \\ &= \frac{1}{T} \mathop{\mathbb{E}}_{x \sim q(x)} \mathcal{L}(x) + \mathop{\mathbb{E}}_{x \sim q(x)} \ln q(x) + \ln(Z) \\ &= \frac{1}{T} \mathop{\mathbb{E}}_{x \sim q(x)} \mathcal{L}(x) - H(q) + \text{const.} \end{aligned} \tag{3.6}$$

Hence, the KL divergence is the sum of the expected value of the style loss $\mathcal{L}$ and the negative entropy of the generated distribution q.

The first term can be estimated by taking the expectation over generated samples:

$$\mathop{\mathbb{E}}_{x \sim q(x)} \mathcal{L}(x) = \mathop{\mathbb{E}}_{z \sim \mathcal{N}(0,I)} \mathcal{L}(g(z)). \tag{3.7}$$

This is similar to the reparametrization trick of [Kingma and Welling, 2014] and is also used in [Johnson et al., 2016, Ulyanov et al., 2016] to construct their learning objectives.

The second term, the negative entropy, is harder to estimate accurately, but simple estimators exist. One which is particularly appealing in our scenario is the Kozachenko-Leonenko estimator [Kozachenko and Leonenko, 1987]. This estimator considers a *batch* of N samples $x_1, \ldots, x_n \sim q(x)$. Then, for each sample x_i, it computes the distance ρ_i to its nearest neighbour in the batch:

$$\rho_i = \min_{j \neq i} \|x_i - x_j\|. \tag{3.8}$$

The distances ρ_i can be used to approximate the entropy as follows:

$$H(q) \approx \frac{D}{N} \sum_{i=1}^{N} \ln \rho_i + \text{const.} \tag{3.9}$$

where $D = 3WH$ is the number of components of the images $x \in \mathbb{R}^{3 \times W \times H}$.

An energy term similar to (3.6) was recently proposed in [Kim and Bengio, 2016] for improving the diversity of a generator network in a adversarial learning scheme. While the idea is superficially similar, the application (sampling the Julesz ensemble) and instantiation (the way the entropy term is implemented) are very different.

3.3.1 Learning objective

We are now ready to define an objective function $E(g)$ to learn the generator network g. This is given by substituting the expected loss (3.7) and the entropy estimator (3.9), computed over a batch of N generated images, in the KL divergence (3.6):

$$E(g) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{T}\mathcal{L}(g(z_i)) - \lambda \ln \min_{j\neq i}\|g(z_i) - g(z_j)\|\right] \tag{3.10}$$

The batch itself is obtained by drawing N samples $z_1, \ldots, z_n \sim \mathcal{N}(0, I)$ from the noise distribution of the generator. The first term in eq. (3.10) measures how closely the generated images $g(z_i)$ are to the Julesz ensemble. The second term quantifies the lack of diversity in the batch by mutually comparing the generated images.

3.3.2 Learning

The loss function (3.10) is in a form that allows optimization by means of Stochastic Gradient Descent (SGD). The algorithm samples a batch $z_1, \ldots, z_n$ at a time and then descends the gradient:

$$\frac{1}{N}\sum_{i=1}^{N}\left[\frac{d\mathcal{L}}{dx^\top}\frac{dg(z_i)}{d\theta^\top} - \frac{\lambda}{\rho_i}(g(z_i) - g(z_{j_i^*}))^\top\left(\frac{dg(z_i)}{d\theta^\top} - \frac{dg(z_{j_i^*})}{d\theta^\top}\right)\right] \tag{3.11}$$

where θ is the vector of parameters of the neural network g, the tensor image x has been implicitly vectorized and j_i^* is the index of the nearest neighbour of image i in the batch.

3.4 Stylization with instance normalization

The work of [Ulyanov et al., 2016] showed that it is possible to learn high-quality texture networks $g(z)$ that generate images in a Julesz ensemble. They also showed that it is possible to learn good quality stylization networks $g(x_0, z)$ that apply the style of a fixed texture to an arbitrary content image x_0.

Nevertheless, the stylization problem was found to be harder than the texture generation one. For the stylization task, they found that learning the model from too many example content images x_0, say more than 16, yielded poorer *qualitative* results than using a smaller number of such examples. Some of the most significant errors appeared along the border of the generated images, probably due to padding and other boundary effects

in the generator network. We conjectured that these are symptoms of a learning problem too difficult for their choice of neural network architecture.

A simple observation that may make learning simpler is that the result of stylization should not, in general, depend on the contrast of the content image but rather should match the contrast of the texture that is being applied to it. Thus, the generator network should discard contrast information in the content image x_0. We argue that learning to discard contrast information by using standard CNN building block is unnecessarily difficult, and is best done by adding a suitable layer to the architecture.

To see why, let $x \in \mathbb{R}^{N\times C\times W\times H}$ be an input tensor containing a batch of N images. Let x_{nijk} denote its $nijk$-th element, where k and j span spatial dimensions, i is the feature channel (i.e. the color channel if the tensor is an RGB image), and n is the index of the image in the batch. Then, contrast normalization is given by:

$$\begin{aligned} y_{nijk} &= \frac{x_{nijk} - \mu_{ni}}{\sqrt{\sigma_{ni}^2 + \epsilon}}, \\ \mu_{ni} &= \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{nilm}, \\ \sigma_{ni}^2 &= \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} (x_{nilm} - \mu_{ni})^2. \end{aligned} \tag{3.12}$$

It is unclear how such as function could be implemented as a sequence of standard operators such as ReLU and convolution.

On the other hand, the generator network of [Ulyanov et al., 2016] does contain a normalization layers, and precisely *batch normalization* (BN) ones. The key difference between eq. (3.12) and batch normalization is that the latter applies the normalization to a whole batch of images instead of single ones:

$$\begin{aligned} y_{nijk} &= \frac{x_{nijk} - \mu_{i}}{\sqrt{\sigma_{i}^2 + \epsilon}}, \\ \mu_{i} &= \frac{1}{HWN}\sum_{n=1}^{N}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{nilm}, \\ \sigma_{i}^2 &= \frac{1}{HWN}\sum_{n=1}^{N}\sum_{l=1}^{W}\sum_{m=1}^{H} (x_{nilm} - \mu_{i})^2. \end{aligned} \tag{3.13}$$

We argue that, for the purpose of stylization, the normalization operator of eq. (3.12) is preferable as it can normalize each individual content image x_0.

FIGURE 3.2: Comparison of normalization techniques in image stylization. From left to right: BN, cross-channel LRN at the first layer, IN at the first layer, IN throughout.

While some authors call layer eq. (3.12) contrast normalization, here we refer to it as *instance normalization* (IN) since we use it as a drop-in replacement for batch normalization operating on individual instances instead of the batch as a whole. Note in particular that this means that instance normalization is applied throughout the architecture, not just at the input image—fig. 3.2 shows the benefit of doing so.

Another similarity with BN is that each IN layer is followed by a scaling and bias operator $s \odot \mathbf{x} + b$. A difference is that the IN layer is applied at test time as well, unchanged, whereas BN is usually switched to use accumulated mean and variance instead of computing them over the batch.

IN appears to be similar to the layer normalization method introduced in [Ba et al., 2016] for recurrent networks, although it is not clear how they handle spatial data. Like theirs, IN is a generic layer, so we tested it in classification problems as well. In such cases, it still work surprisingly well, but not as well as batch normalization (e.g. AlexNet [Krizhevsky et al., 2012] IN has 2-3% worse top-1 accuracy on ILSVRC [Russakovsky et al., 2015a] than AlexNet BN).

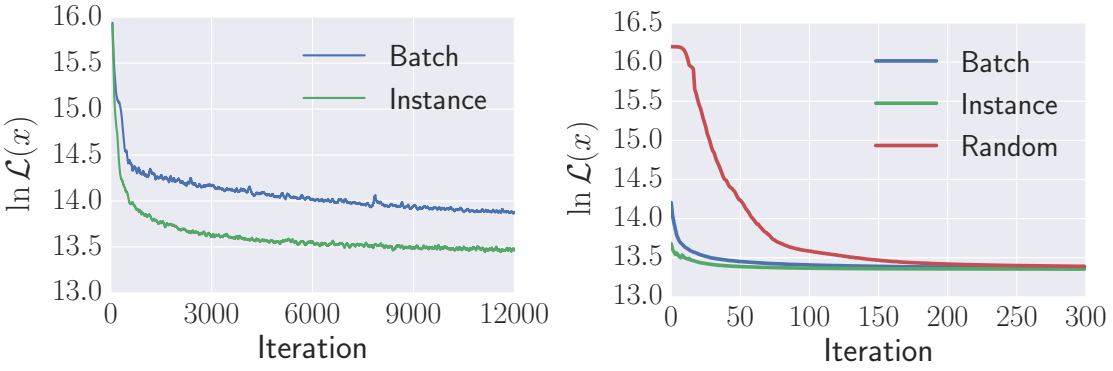
3.5 Experiments

In this section, after discussing the technical details of the method, we evaluate our new texture network architectures using instance normalization, and then investigate the ability of the new formulation to learn diverse generators.

3.5.1 Technical details

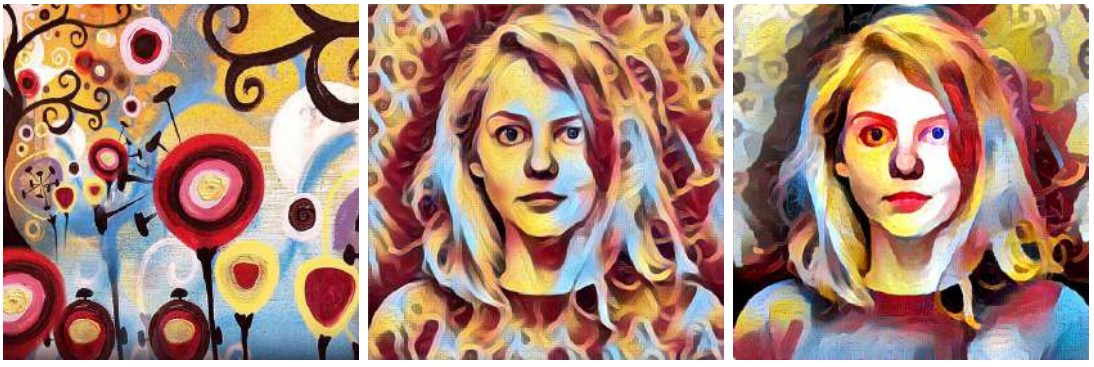
3.5.1.1 Network architecture

Among two generator network architectures, proposed previously in [Ulyanov et al., 2016, Johnson et al., 2016], we choose the residual architecture from [Johnson et al.,



(A) Feed-forward history (B) Finetuning history

(C) Content (D) StyleNet IN (E) IN finetuned



(F) Style (G) StyleNet BN (H) BN finetuned

FIGURE 3.3: (a) learning objective as a function of SGD iterations for StyleNet IN and BN. (b) Direct optimization of the Gatys *et al.* for this example image starting from the result of StyleNet IN and BN. (d,g) Result of StyleNet with instance (d) and batch normalization (g). (e,h) Result of finetuning the Gatys *et al.* energy.

Content StyleNet IN (ours) StyleNet BN Gatys *et al.* Style

FIGURE 3.4: Stylization results obtained by applying different textures (rightmost column) to different content images (leftmost column). Three methods are compared: StyleNet IN, StyleNet BN, and iterative optimization. StyleNet BN is similar to [Ulyanov et al., 2016] and [Johnson et al., 2016] but trained on larger images (`512x` compared to `256x` in [Ulyanov et al., 2016, Johnson et al., 2016]) for a fair comparison with StyleNet IN. We compare to original [Ulyanov et al., 2016, Johnson et al., 2016] in supmat.

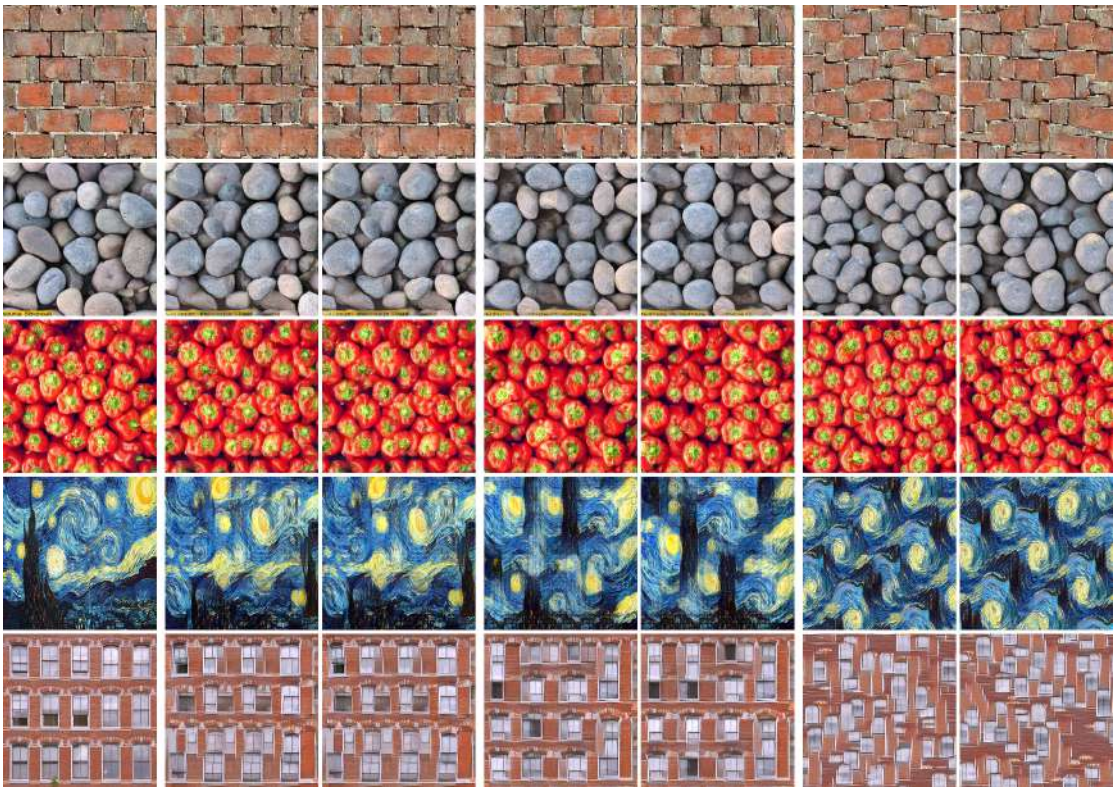


FIGURE 3.5: The textures generated by the high capacity Texture Net V2 without diversity term ($\lambda = 0$ in eq. (3.10)) are nearly identical. The low capacity TextureNet V1 of [Ulyanov et al., 2016] achieves diversity, but has sometimes poor results. TextureNet V2 with diversity is the best of both worlds.

2016] for all our style transfer experiments. We also experimented with architecture from [Ulyanov et al., 2016] and observed a similar improvement with our method, but use the one from [Johnson et al., 2016] for convenience. We call it *StyleNet* with a postfix BN if it is equipped with batch normalization or IN for instance normalization.

For texture synthesis we compare two architectures: the multiscale fully-convolutional architecture from [Ulyanov et al., 2016] (TextureNetV1) and the one we design to have a very large receptive field (TextureNetV2). TextureNetV2 takes a noise vector of size 256 and first transforms it with two fully-connected layers. The output is then reshaped to a 4×4 image and repeatedly upsampled with fractionally-strided convolutions similar to [Radford et al., 2016].

3.5.1.2 Weight parameters

In practice, for the case of $\lambda > 0$, entropy loss and texture loss in eq. (3.10) should be weighted properly. As only the value of $T\lambda$ is important for optimization we assume $\lambda = 1$ and choose T from the set of three values $(5, 10, 20)$ for texture synthesis (we pick

the higher value among those not leading to artifacts – see our discussion below). We fix $T = 10000$ for style transfer experiments. For texture synthesis, similarly to [Ulyanov et al., 2016], we found useful to normalize gradient of the texture loss as it passes back through the VGG-19 network. This allows rapid convergence for stochastic optimization but implicitly alters the objective function and requires temperature to be adjusted. We observe that for textures with flat lightning high entropy weight results in brightness variations over the image fig. 3.7. We hypothesize this issue can be solved if either more clever distance for entropy estimation is used or an image prior introduced.

3.5.2 Effect of instance normalization

In order to evaluate the impact of replacing batch normalization with instance normalization, we consider first the problem of *stylization*, where the goal is to learn a generator $x = g(x_0, z)$ that applies a certain texture style to the content image x_0 using noise z as "random seed". We set $\lambda = 0$ for which generator is most likely to discard the noise.

The StyleNet IN and StyleNet BN are compared in fig. 3.3. Panel fig. 3.3.a shows the training objective (3.5) of the networks as a function of the SGD training iteration. The objective function is the same, but StyleNet IN converges much faster, suggesting that it can solve the stylization problem more easily. This is confirmed by the stark difference in the qualitative results in panels (d) end (g). Since the StyleNets are trained to minimize in one shot the same objective as the iterative optimization of Gatys *et al.*., they can be used to *initialize* the latter algorithm. Panel (b) shows the result of applying the Gatys *et al.* optimization starting from their random initialization and the output of the two StyleNets. Clearly both networks start much closer to an optimum than random noise, and IN closer than BN. The difference is qualitatively large: panels (e) and (h) show the change in the StyleNets output after finetuning by iterative optimization of the loss, which has a small effect for the IN variant, and a much larger one for the BN one.

Similar results apply in general. Other examples are shown in fig. 3.4, where the IN variant is far superior to BN and much closer to the results obtained by the much slower iterative method of Gatys *et al.*. StyleNets are trained on images of a fixed sized, but since they are convolutional, they can be applied to arbitrary sizes. In the figure, the top tree images are processed at 512×512 resolution and the bottom two at 1024×1024. In general, we found that higher resolution images yield visually better stylization results.

While instance normalization works much better than batch normalization for stylization, for texture synthesis the two normalization methods perform equally well. This is consistent with our intuition that IN helps in normalizing the information coming from

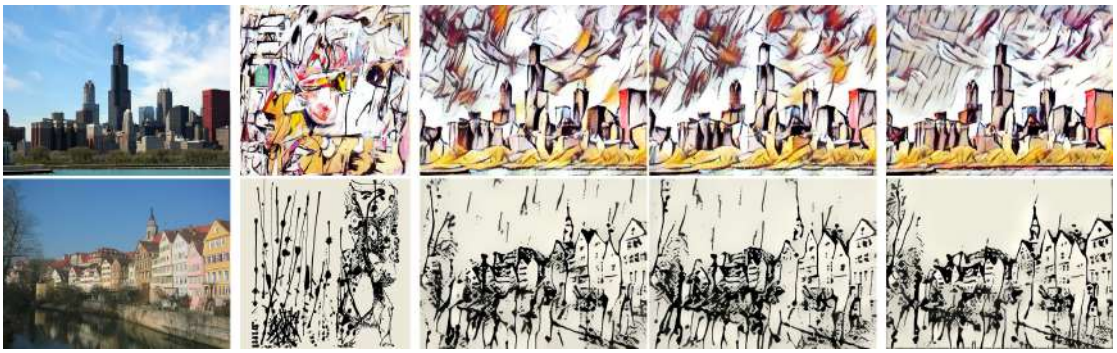


FIGURE 3.6: The StyleNetV2 $g(x_0, z)$, trained with diversity $\lambda > 0$, generates substantially different stylizations for different values of the input noise z. With $\lambda = 0$ generator tends to ignore noise channels when trained for sufficiently long time thus producing almost the same stylization for different noise z.

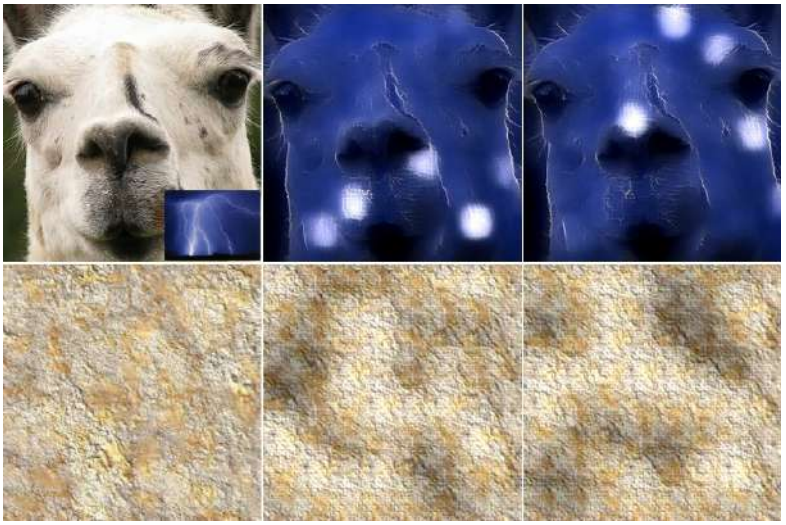


FIGURE 3.7: Negative examples. If the diversity term λ is too high for the learned style, the generator tends to generate artifacts in which brightness is changed locally (spotting) instead of (or as well as) changing the structure.

content image x_0, which is highly variable, whereas it is not important to normalize the texture information, as each model learns only one texture style.

3.5.3 Effect of the diversity term

Having validated the IN-based architecture, we evaluate now the effect of the entropy-based diversity term in the objective function (3.10).

The experiment in fig. 3.5 starts by considering the problem of texture generation. We compare the new high-capacity TextureNetV2 and the low-capacity TextureNetsV1 texture synthesis networks. The low-capacity model is the same as [Ulyanov et al.,

2016]. This network was used there in order to force the network to learn a non-trivial dependency on the input noise, thus generating diverse outputs even though the learning objective of [Ulyanov et al., 2016], which is the same as eq. (3.10) with diversity coefficient $\lambda = 0$, tends to suppress diversity. The results in fig. 3.5 are indeed diverse, but sometimes of low quality. This should be contrasted with TextureNetV2, the high-capacity model: its visual fidelity is much higher, but, by using the same objective function [Ulyanov et al., 2016], the network learns to generate a single image, as expected. TextureNetV2 with the new diversity-inducing objective ($\lambda > 0$) is the best of both worlds, being both high-quality and diverse.

The experiment in fig. 3.6 assesses the effect of the diversity term in the stylization problem. The results are similar to the ones for texture synthesis and the diversity term effectively encourages the network to learn to produce different results based on the input noise.

One difficultly with texture and stylization networks is that the entropy loss weight λ must be tuned for each learned texture model. Choosing λ too small may fail to learn a diverse generator, and setting it too high may create artifacts, as shown in fig. 3.7.

3.6 Summary

This work advances feed-forward texture synthesis and stylization networks in two significant ways. It introduces instance normalization, an architectural change that makes training stylization networks easier and allows the training process to achieve much lower loss levels. It also introduces a new learning formulation for training generator networks to sample uniformly from the Julesz ensemble, thus explicitly encouraging diversity in the generated outputs. We show that both improvements lead to noticeable improvements of the generated stylized images and textures, while keeping the generation runtimes intact.

Chapter 4

It Takes (Only) Two: Adversarial Generator-Encoder Networks

Abstract

We present a new autoencoder-type architecture that is trainable in an unsupervised mode, sustains both generation and inference, and has the quality of conditional and unconditional samples boosted by adversarial learning. Unlike previous hybrids of autoencoders and adversarial networks, the adversarial game in our approach is set up directly between the encoder and the generator, and no external mappings are trained in the process of learning. The game objective compares the divergences of each of the real and the generated data distributions with the prior distribution in the latent space. We show that direct generator-vs-encoder game leads to a tight coupling of the two components, resulting in samples and reconstructions of a comparable quality to some recently-proposed more complex architectures.

This work was published as: Dmitry Ulyanov, Andrea Vedaldi and Victor Lempitsky. *It Takes (Only) Two: Adversarial Generator-Encoder Networks.* AAAI Conference on Artificial Intelligence (AAAI), 2018.

4.1 Introduction

Deep (Variational) Auto Encoders (AEs [Bengio, 2009] and VAEs [Kingma and Welling, 2014, Rezende et al., 2014]) and deep Generative Adversarial Networks (GANs [Goodfellow et al., 2014]) are two of the most popular approaches to generative learning. These methods have complementary strengths and weaknesses. VAEs can learn a *bidirectional* mapping between a complex data distribution and a much simpler prior distribution, allowing both generation and inference; on the contrary, the original formulation of GAN learns a *unidirectional* mapping that only allows sampling the data distribution. On the other hand, GANs use more complex loss functions compared to the simplistic data-fitting losses in (V)AEs and can usually generate more realistic samples.

Several recent works have looked for hybrid approaches to support, in a principled way, both sampling and inference like AEs, while producing samples of quality comparable to GANs. Typically this is achieved by training a AE jointly with one or more adversarial discriminators whose purpose is to improve the alignment of distributions in the latent space [Brock et al., 2017, Makhzani et al., 2016], the data space [Che et al., 2017, Larsen et al., 2015] or in the joint (product) latent-data space [Donahue et al., 2017, Dumoulin et al., 2017]. Alternatively, the method of [Zhu et al., 2016] starts by learning a unidirectional GAN, and then learns a corresponding inverse mapping (the encoder) post-hoc.

While compounding autoencoding and adversarial discrimination does improve GANs and VAEs, it does so at the cost of added complexity. In particular, each of these systems involves at least three deep mappings: an encoder, a decoder/generator, and a discriminator. In this work, we show that this is unnecessary and that the advantages of autoencoders and adversarial training can be combined without increasing the complexity of the model.

In order to do so, we propose a new architecture, called an *Adversarial Generator-Encoder (AGE) Network*, that contains only two feed-forward mappings, the encoder and the generator, operating in opposite directions. As in VAEs, the generator maps a simple prior distribution in latent space to the data space, while the encoder is used to move both the real and generated data samples into the latent space. In this manner, the encoder induces two latent distributions, corresponding respectively to the *encoded real data* and the *encoded generated data*. The AGE learning process then considers the divergence of each of these two distributions to the original prior distribution.

There are two advantages of this approach. First, due to the simplicity of the prior distribution, computing its divergence to the latent data distributions reduces to the calculation of simple statistics over small batches of images. Second, unlike GAN-like

approaches, real and generated distributions are never compared directly, thus bypassing the need for discriminator networks as used by GANs. Instead, the adversarial signal in AGE comes from learning the encoder to increase the divergence between the latent distribution of the generated data and the prior, which works against the generator, which tries to decrease the same divergence (Figure 4.1). Optionally, AGE training may include reconstruction losses typical of AEs.

The AGE approach is evaluated on a number of standard image datasets, where we show that the quality of generated samples is comparable to that of GANs [Goodfellow et al., 2014, Radford et al., 2016], and the quality of reconstructions is comparable or better to that of the more complex Adversarially-Learned Inference (ALI) approach of [Dumoulin et al., 2017], while training faster. We further evaluate the AGE approach in the conditional setting, where we show that it can successfully tackle the colorization problem that is known to be difficult for GAN-based approaches.

Other related work. Apart from the above-mentioned approaches, AGE networks can be related to several other recent GAN-based systems. Thus, they are related to improved GANs [Salimans et al., 2016] that proposed to use batch-level information in order to prevent mode collapse. The divergences within AGE training are also computed as batch-level statistics.

Another avenue for improving the stability of GANs has been the replacement of the classifying discriminator with the regression-based one as in energy-based GANs [Zhao et al., 2017] and Wasserstein GANs [Arjovsky et al., 2017]. Our statistics (the divergence from the prior distribution) can be seen as a very special form of regression. In this way, the encoder in the AGE architecture can be (with some reservations) seen as a discriminator computing a single number similarly to how it is done in [Arjovsky et al., 2017, Zhao et al., 2017].

4.2 Adversarial Generator-Encoder Networks

This section introduces our Adversarial Generator-Encoder (AGE) networks. An AGE is composed of two parametric mappings: the *encoder* $e_\psi(\mathbf{x})$, with the learnable parameters ψ, that maps the data space $\mathcal{X}$ to the latent space $\mathcal{Z}$, and the *generator* $g_\theta(\mathbf{z})$, with the learnable parameters θ, which runs in the opposite direction. We will use the shorthand notation $f(Y)$ to denote the distribution of the random variable $f(\mathbf{y})$, $\mathbf{y} \sim Y$.

The reference distribution Z is chosen so that it is easy to sample from it, which in turns allow to sample $g_\theta(Z)$ unconditionally be first sampling $\mathbf{z} \sim Z$ and then by feed-forward evaluation of $\mathbf{x} = g_\theta(\mathbf{z})$, exactly as it is done in GANs. In our experiments, we pick

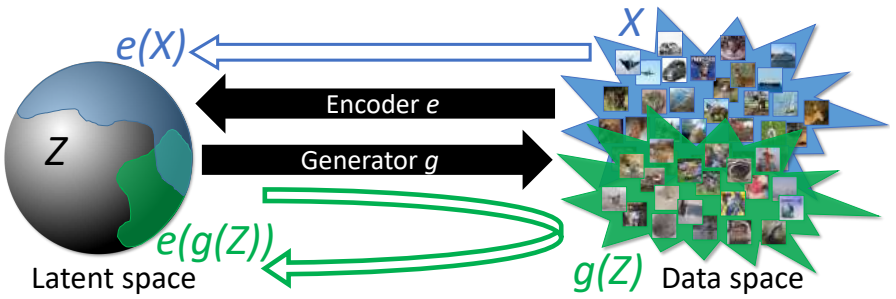


FIGURE 4.1: Our model (AGE network) has only two components: the generator g and the encoder e. The learning process adjusts their parameters in order to align a simple prior distribution Z in the latent space and the data distribution X. This is done by adversarial training, as updates for the generator aim to minimize the divergence between $e(g(Z))$ and Z (aligning green with gray), while updates for the encoder aim to minimize the divergence between $e(X)$ (aligning blue with gray) and to *maximize* the divergence between $e(g(Z))$ and Z (shrink green "away" from gray). We demonstrate that such adversarial learning gives rise to high-quality generators that result in the close match between the real distribution X and the generated distribution $g(Z)$. Our learning can also incorporate reconstruction losses to ensure that encoder-generator acts as autoencoder.

the latent space $\mathcal{Z}$ to be an M-dimensional sphere $\mathbb{S}^M$, and the latent distribution to be a uniform distribution on that sphere $Z = \text{Uniform}(\mathbb{S}^M)$. We have also conducted some experiments with the unit Gaussian distribution in the Euclidean space and have obtained results comparable in quality.

The goal of learning an AGE is to align the real data distribution X to the generated distribution $g_\theta(Z)$ while establishing a correspondence between data and latent samples $\mathbf{x}$ and $\mathbf{z}$. The real data distribution X is empirical and represented by a large number N of data samples $\{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_N\}$. Learning amounts to tuning the parameter ψ and θ to optimize the AGE criterion, discussed in the next section. This criterion is based on an adversarial game whose saddle points correspond to networks that align real and generated data distribution ($g(Z) = X$). The criterion is augmented with additional terms that encourage the reciprocity of the encoder e and the generator g.

4.2.1 Adversarial distribution alignment

The GAN approach to aligning two distributions is to define an adversarial game based on a ratio of probabilities [Goodfellow et al., 2014]. The ratio is estimated by repeatedly fitting a binary classifier that distinguishes between samples obtained from the real and generated data distributions. Here, we propose an alternative adversarial setup with some advantages with respect to GAN's, including avoiding generator collapse [Goodfellow, 2017].

The goal of AGE is to generate a distribution $g(Z)$ in data space that is close to the true data distribution X. However, direct matching of the distributions in the high-dimensional data space, as done in GAN, can be challenging. We propose instead to move this comparison *to the simpler latent space*. This is done by introducing a divergence measure $\Delta(P\|Q)$ between distributions defined in the latent space $\mathcal{Z}$. We only require this divergence to be non-negative and zero if, and only if, the distributions are identical ($\Delta(P\|Q) = 0 \iff P = Q$).[1] The encoder function e maps the distributions X and $g(Z)$ defined in data space to corresponding distributions $e(X)$ and $e(g(Z))$ in the latent space. Below, we show how to design an adversarial criterion such that minimizing the divergence $\Delta(e(X), e(g(Z)))$ in latent space induces the distributions X and $g(Z)$ to align in data space as well.

In the theoretical analysis below, we assume that encoders and decoders span the class of all measurable mappings between the corresponding spaces. This assumption, often referred to as *non-parametric limit*, is justified by the universality of neural networks [Hornik et al., 1989]. We further make the **assumption** that there exists at least one "perfect" generator that matches the data distribution, i.e. $\exists g_0 : g_0(Z) = X$.

We start by considering a simple game with objective defined as:

$$\max_e \min_g V_1(g, e) = \max_e \min_g \Delta(\, e(g(Z)) \| e(X)\,)\,. \tag{4.1}$$

As the following theorem shows, perfect generators form saddle points (Nash equilibria) of the game (4.1) and all saddle points of the game (4.1) are based on perfect generators.

Theorem 4.1. *A pair (g^*, e^*) forms a saddle point of the game (4.1) if and only if the generator g^* matches the data distribution, i.e. $g^*(Z) = X$.*

The proofs of this and the following theorems are given in the section 4.5 While the game (4.1) is sufficient for aligning distributions in the data space, finding such saddle points is difficult due to the need of comparing two empirical (hence non-parametric) distributions $e(X)$ and $e(g(Z))$. We can avoid this issue by introducing an intermediate reference distribution Y and comparing the distributions to that instead, resulting in the game:

$$\max_e \min_g V_2(g, e) = \max_e \min_g \Delta(e(g(Z))\|Y) - \Delta(e(X)\|Y). \tag{4.2}$$

Importantly, (4.2) still induces alignment of real and generated distributions in data space:

[1] We do not require the divergence to be a distance.

Theorem 4.2. *If a pair (g^*, e^*) is a saddle point of game (4.2) then the generator g^* matches the data distribution, i.e. $g^*(Z) = X$. Conversely, if the generator g^* matches the data distribution, then for some e^* the pair (g^*, e^*) is a saddle point of (4.2).*

The important benefit of formulation (4.2) is that, if Y is selected in a suitable manner, it is simple to compute the divergence of Y to the empirical distributions $e(g(Z))$ and $e(X)$. For convenience, in particular, we choose Y to coincide with the "canonical" (prior) distribution Z. By substituting $Y = Z$ in objective (4.2), the loss can be extended to include reconstruction terms that can improve the quality of the result. It can also be optimized by using stochastic approximations as described in section 4.2.3.

Given a distribution Q in data space, the encoder e and divergence $\Delta(\cdot\|Y)$ can be interpreted as extracting statistics $F(Q) = \Delta(e(Q)\|Y)$ from Q. Hence, game (4.2) can be though of as comparing certain statistics of the real and generated data distributions. Similarly to GANs, these statistics are not fixed but evolve during learning.

We also note that, even away from the saddle point, the minimization $\min_g V_2(g, e)$ for a fixed e does not tend to collapse for many reasonable choice of divergence (e.g. KL-divergence). In fact, any collapsed distribution would inevitably lead to a very high value of the first term in (4.2). Thus, unlike GANs, our approach can optimize the generator for a fixed adversary till convergence and obtain a non-degenerate solution. On the other hand, the maximization $\max_e V_2(g, e)$ for some fixed g can lead to $+\infty$ score for some divergences.

4.2.2 Encoder-generator reciprocity and reconstruction losses

In the previous section we have demonstrated that finding a saddle point of (4.2) is sufficient to align real and generated data distributions X and $g(Z)$ and thus generate realistically-looking data samples. At the same time, this by itself does not necessarily imply that mappings e and g are reciprocal. Reciprocity, however, can be desirable if one wishes to reconstruct samples $\mathbf{x} = g(\mathbf{z})$ from their codes $\mathbf{z} = e(\mathbf{x})$.

In this section, we introduce losses that encourage encoder and generator to be reciprocal. Reciprocity can be measured either in the latent space or in the data space, resulting in the loss functions based on reconstruction errors, e.g.:

$$L_{\mathcal{X}}(g_\theta, e_\psi) = \mathbb{E}_{\mathbf{x}\sim X}\|\mathbf{x} - g_\theta\left(e_\psi(\mathbf{x})\right)\|_1\,, \tag{4.3}$$

$$L_{\mathcal{Z}}(g_\theta, e_\psi) = \mathbb{E}_{\mathbf{z}\sim Z}\|\mathbf{z} - e_\psi\left(g_\theta(\mathbf{z})\right)\|_2^2\,. \tag{4.4}$$

Both losses (4.3) and (4.4) thus encourage the reciprocity of the two mappings. Note also that (4.3) is the traditional pixelwise loss used within AEs (L1-loss was preferred, as it is known to perform better in image synthesis tasks with deep architectures).

A natural question then is whether it is helpful to minimize both losses (4.3) and (4.4) at the same time or whether considering only one is sufficient. The answer is given by the following statement:

Theorem 4.3. *Let the two distributions W and Q be aligned by the mapping f (i.e. $f(W) = Q$) and let $\mathbb{E}_{\mathbf{w}\sim W}\|\mathbf{w} - h\Big(f(\mathbf{w})\Big)\|_2^2 = 0$. Then, for $\mathbf{w} \sim W$ and $\mathbf{q} \sim Q$, we have $\mathbf{w} = h(f(\mathbf{w}))$ and $\mathbf{q} = f(h(\mathbf{q}))$ almost certainly, i.e. the mappings f and h invert each other almost everywhere on the supports of W and Q. Furthermore, Q is aligned with W by h, i.e. $h(Q) = W$.*

Recall that Theorem 4.2 establishes that the solution (saddle point) of game (4.2) aligns distributions in the data space. Then Theorem 4.3 shows that when augmented with the latent space loss (4.4), the objective (4.2) is sufficient to ensure reciprocity.

4.2.3 Training AGE networks

Based on the theoretical analysis derived in the previous subsections, we now suggest the approach to the joint training of the generator in the encoder within the AGE networks. As in the case of GAN training, we set up the learning process for an AGE network as a game with the iterative updates over the parameters θ and ψ that are driven by the optimization of different objectives. In general, the optimization process combines the maximin game for the functional (4.2) with the optimization of the reciprocity losses (4.3) and (4.4).

In particular, we use the following game objectives for the generator and the encoder:

$$\hat{\theta} = \arg\min_{\theta} \left(V_2(g_\theta, e_{\bar{\psi}}) + \lambda L_{\mathcal{Z}}(g_\theta, e_{\bar{\psi}})\right) , \tag{4.5}$$

$$\hat{\psi} = \arg\max_{\psi} \left(V_2(g_{\bar{\theta}}, e_{\psi}) - \mu L_{\mathcal{X}}(g_{\bar{\theta}}, e_{\psi})\right) , \tag{4.6}$$

where $\bar{\psi}$ and $\bar{\theta}$ denote the value of the encoder and generator parameters at the moment of the optimization and λ, μ is a user-defined parameter. Note that both objectives (4.5), (4.6) include only one of the reconstruction losses. Specifically, the generator objective includes only the latent space reconstruction loss. In the experiments, we found that the omission of the other reconstruction loss (in the data space) is important to avoid possible blurring of the generator outputs that is characteristic to autoencoders.

Similarly to GANs, in (4.5), (4.6) we perform only several steps toward optimum at each iteration, thus alternating between generator and encoder updates.

By maximizing the difference between $\Delta(e_\psi(g_{\bar{\theta}}(Z))\|Z)$ and $\Delta(e_\psi(X)\|Z)$, the optimization process (4.6) focuses on the maximization of the mismatch between the real data distribution X and the distribution of the samples from the generator $g_{\bar{\theta}}(Z)$. Informally speaking, the optimization (4.6) forces the encoder to find the mapping that aligns real data distribution X with the target distribution Z, while mapping non-real (synthesized data) $g_{\bar{\theta}}(Z)$ away from Z. When Z is a uniform distribution on a sphere, the goal of the encoder would be to uniformly spread the real data over the sphere, while cramping as much of synthesized data as possible together assuring non-uniformity of the distribution $e_\psi\left(g_{\bar{\theta}}(Z)\right)$.

Any differences (misalignment) between the two distributions are thus amplified by the optimization process (4.6) and force the optimization process (4.5) to focus specifically on removing these differences. Since the misalignment between X and $g(Z)$ is measured after projecting the two distributions into the latent space, the maximization of this misalignment makes the encoder to compute features that distinguish the two distributions.

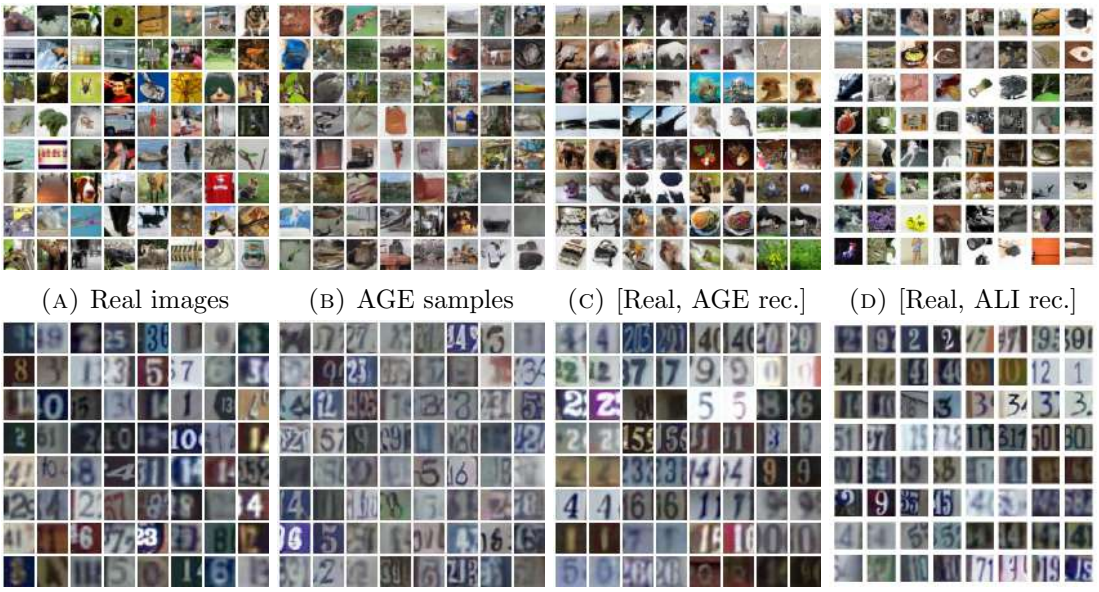
4.3 Experiments

We have validated AGE networks in two settings. A more traditional setting involves *unconditional* generation and reconstruction, where we consider a number of standard image datasets. We have also evaluated AGE networks in the *conditional* setting. Here, we tackle the problem of image colorization, which is hard for GANs. In this setting, we condition both the generator and the encoder on the grayscale image. Taken together, our experiments demonstrate the versatility of the AGE approach.

4.3.1 Unconditionally-trained AGE networks

Network architectures: In our experiments, the generator and the encoder networks have a similar structure to the generator and the discriminator in DCGAN [Radford et al., 2016]. To turn the discriminator into the encoder, we have modified it to output an M-dimensional vector and replaced the final sigmoid layer with the normalization layer that projects the points onto the sphere.

Divergence measure: As we need to measure the divergence between the empirical distribution and the prior distribution in the latent space, we have used the following



(A) Real images (B) AGE samples (C) [Real, AGE rec.] (D) [Real, ALI rec.]

FIGURE 4.2: Samples (b) and reconstructions (c) for Tiny ImageNet dataset (top) and SVHN dataset (bottom). The results of ALI [Dumoulin et al., 2017] on the same datasets are shown in (d). In (c,d) odd columns show real examples and even columns show their reconstructions. Qualitatively, our method seems to obtain more accurate reconstructions than ALI [Dumoulin et al., 2017], especially on the Tiny ImageNet dataset, while having samples of similar visual quality.

measure. Given a set of samples on the M-dimensional sphere, we fit the Gaussian Normal distribution with diagonal covariance matrix in the embedding M-dimensional space and we compute the KL-divergence of such Gaussian with the unit Gaussian as

$$\mathrm{KL}(Q\|\mathcal{N}(0,I)) = -\frac{1}{2} + \frac{1}{M}\sum_{j=1}^{M}\frac{s_j^2 + m_j^2}{2} - \log(s_j) \tag{4.7}$$

where m_j and s_j are the means and the standard deviations of the fitted Gaussians along various dimensions. Since the uniform distribution on the sphere will entail the lowest possible divergence with the unit Gaussian in the embedding space among all distributions on the unit sphere, such divergence measure is valid for our analysis above. We have also tried to measure the same divergence non-parametrically using Kozachenko-Leonenko estimator [Kozachenko and Leonenko, 1987]. In our initial experiments, both versions worked equally well, and we used a simpler parametric estimator in the presented experiments.

Hyper-parameters: We use ADAM [Kingma and Ba, 2015] optimizer with the learning rate of 0.0002. We perform two generator updates per one encoder update for all datasets. For each dataset we tried $\lambda \in \{500, 1000, 2000\}$ and picked the best one. We ended up using $\mu = 10$ for all datasets. The dimensionality M of the latent space was manually set according to the complexity of the dataset. We thus used $M = 64$

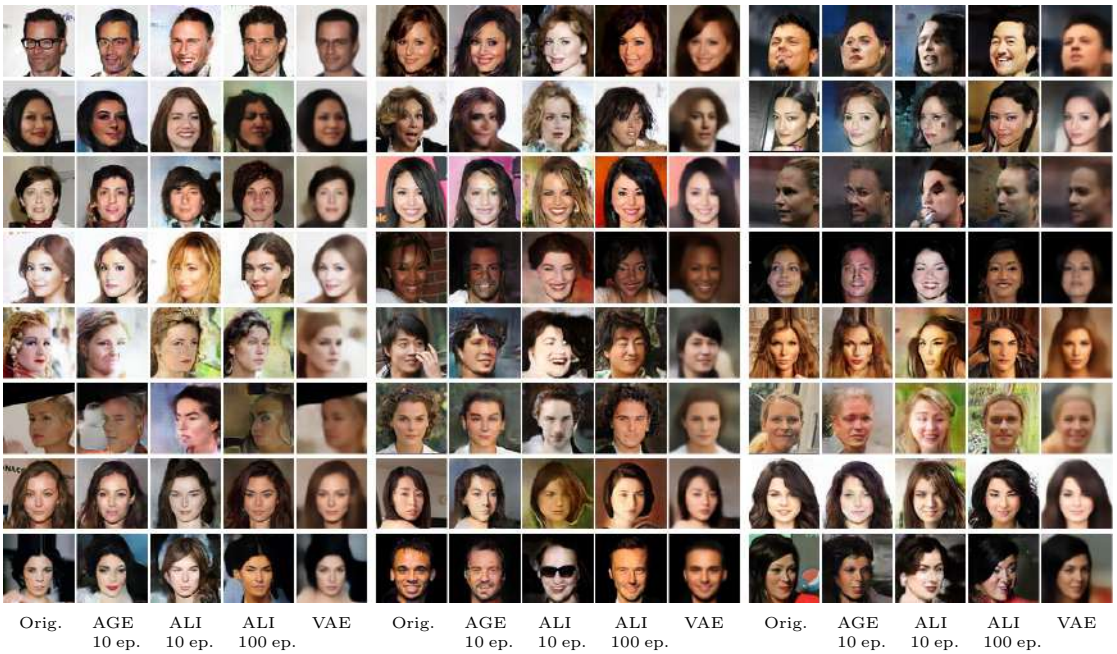


FIGURE 4.3: Reconstruction quality comparison of our method with ALI [Dumoulin et al., 2017] and VAE [Kingma and Welling, 2014]. The first column in each set shows examples from the test set of CelebA dataset. In the other columns the reconstructions for different methods are presented: column two for ours method, three and four for ALI and five for VAE. We train our model for 10 epochs and compare to ALI, trained for the same number of epochs (column three). Importantly one epoch for our method takes 3 times less time than for ALI. For a fair comparison we also present the results of ALI, trained till convergence.

for CelebA and SVHN datasets, and $M = 128$ for the more complex datasets of Tiny ImageNet and CIFAR-10.

Results: We evaluate unconditional AGE networks on several standard datasets, while treating the system [Dumoulin et al., 2017] as the most natural reference for comparison (as the closest three-component counterpart to our two-component system). The results for [Dumoulin et al., 2017] are either reproduced with the author's code or copied from [Dumoulin et al., 2017].

In Figure 4.2, we present the results on the challenging Tiny ImageNet dataset [Russakovsky et al., 2015b] and the SVHN dataset [Netzer et al., 2011]. We show both samples $g(\mathbf{z})$ obtained for $\mathbf{z} \sim Z$ as well as the reconstructions $g\Big(e(\mathbf{x})\Big)$ alongside the real data samples $\mathbf{x}$. We also show the reconstructions obtained by [Dumoulin et al., 2017] for comparison. Inspection reveals that the fidelity of [Dumoulin et al., 2017] is considerably lower for Tiny ImageNet dataset.

In Figure 4.3, we further compare the reconstructions of CelebA [Liu et al., 2015a] images obtained by the AGE network, ALI [Dumoulin et al., 2017], and VAE [Kingma and Welling, 2014]. Overall, the fidelity and the visual quality of AGE reconstructions

are roughly comparable or better than ALI. Furthermore, ALI takes notoriously longer time to converge than our method, and the reconstructions of ALI after 10 epochs (which take six hours) of training look considerably worse than AGE reconstructions after 10 epochs (which take only two hours), thus attesting to the benefits of having a simpler two-component system.

Next we evaluate our method quantitatively. For the model trained on CIFAR-10 dataset we compute Inception score [Salimans et al., 2016]. The AGE score is 5.90 ± 0.04, which is higher than the ALI [Dumoulin et al., 2017] score of 5.34 ± 0.05 (as reported in [Warde-Farley and Bengio, 2017]) and than the score of 4.36 ± 0.04 from [Salimans et al., 2016]. The state-of-the-art from [Warde-Farley and Bengio, 2017] is higher still (7.72 ± 0.13).

We also computed log likelihood for AGE and ALI on the MNIST dataset using the method of [Wu et al., 2016] with latent space of size 10 using authours source code. ALIs score is 721 while AGEs score is 746. The AGE model is also superior than both VAE and GAN, which scores are 705.375 and 328.772 respectively as evaluated by [Wu et al., 2016].

Finally, similarly to [Dumoulin et al., 2017, Donahue et al., 2017, Radford et al., 2016] we investigated whether the learned features are useful for discriminative tasks. We reproduced the evaluation pipeline from [Dumoulin et al., 2017] for SVHN dataset and obtained 23.7% error rate in the unsupervised feature learning protocol with our model, while their result is 19.14%. At the moment, it is unclear to us why AGE networks underperform ALI at this task.

4.3.2 Conditional AGE network experiments

Recently, several GAN-based systems have achieved very impressive results in the conditional setting, where the latent space is augmented or replaced with a second data space corresponding to different modality [Isola et al., 2017a, Zhu et al., 2017a]. Arguably, it is in the conditional setting where the bi-directionality lacking in conventional GANs is most needed. In fact, by allowing to switch back-and-forth between the data space and the latent space, bi-directionality allows powerful neural image editing interfaces [Zhu et al., 2016, Brock et al., 2017].

Here, we demonstate that AGE networks perform well in the conditional setting. To show that, we have picked the image colorization problem, which is known to be hard for GANs. To the best of our knowledge, while the idea of applying GANs to the colorization task seems very natural, the only successful GAN-based colorization results were presented in [Isola et al., 2017a], and we compare to the authors' implementation

of their pix2pix system. We are also aware of several unsuccessful efforts to use GANs for colorization.

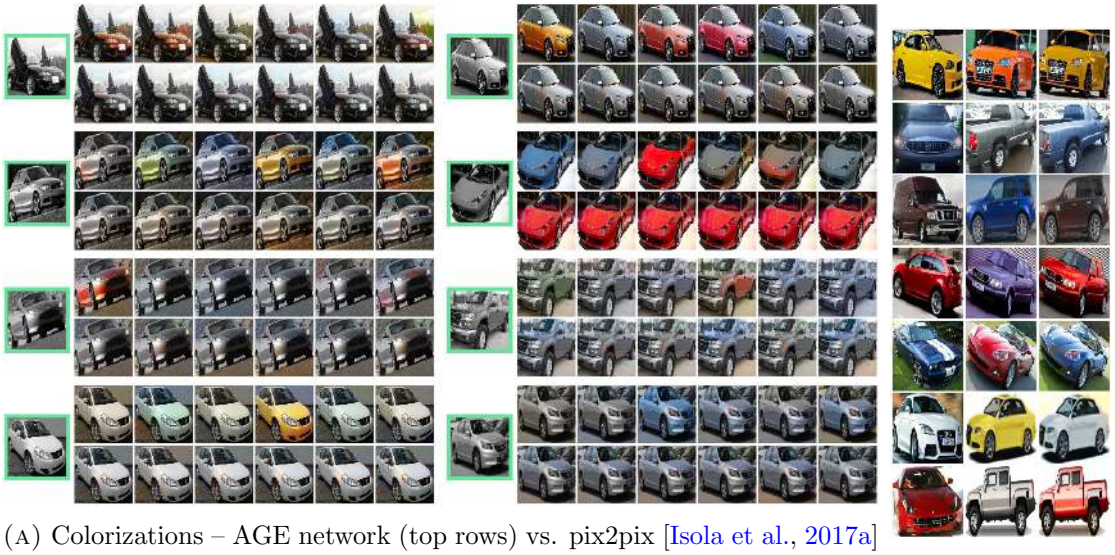
To use AGE for colorization, we work with images in the *Lab* color space, and we treat the *ab* color channels of an image as a data sample $\mathbf{x}$. We then use the lightness channel L of the image as an input to both the encoder $e_\psi(\mathbf{x}|L)$ and the generator $g_\theta(\mathbf{z}|L)$, effectively conditioning the encoder and the generator on it. Thus, different latent variables $\mathbf{z}$ will result in different colorizations $\mathbf{x}$ for the same grayscale image L. The latent space in these experiments is taken to be three-dimensional.

The particular architecture of the generator takes the input image L, augments it with $\mathbf{z}$ variables expanded to constant maps of the same spatial dimensions as L, and then applies the ResNet type architecture [He et al., 2016, Johnson et al., 2016] that computes $\mathbf{x}$ (i.e. the *ab*-channels). The encoder architecture is a convolutional network that maps the concatenation of L and $\mathbf{x}$ (essentially, an image in the Lab-space) to the latent space. The divergence measure is the same as in the unconditional AGE experiments and is computed "unconditionally" (i.e. each minibatch passed through the encoder combines multiple images with different L).

We perform the colorization experiments on Stanford Cars dataset [Krause et al., 2013] with 16,000 training images of 196 car models, since cars have inherently ambiguous colors and hence their colorization is particularly prone to the regression-to-mean effect. The images were downsampled to 64×64.

We present colorization results in Figure 4.4. Crucially, AGE generator is often able to produce plausible and diverse colorizations for different latent vector inputs. As we wanted to enable pix2pix GAN-based system of [Isola et al., 2017a] to produce diverse colorizations, we augmented the input to their generator architecture with three constant-valued maps (same as in our method). We however found that their system effectively learns to ignore such input augmentation and the diversity of colorizations was very low (Figure 4.4a).

To demonstrate the meaningfulness of the latent space learned by the conditional AGE training, we also demonstrate the color transfer examples, where the latent vector $\mathbf{z}_1 = e_\psi(\mathbf{x}_1|L_1)$ obtained by encoding the image $[x_1, L_1]$ is then used to colorize the grayscale image L_2, i.e. $\mathbf{x}_2 = g_\theta(\mathbf{z}_1|L_2)$ (Figure 4.4b).



(A) Colorizations – AGE network (top rows) vs. pix2pix [Isola et al., 2017a] (bottom rows)

(B) Color transfer

FIGURE 4.4: (a) Each pane shows colorizations of the input grayscale image (emphasized) using conditional AGE networks (top rows) and pix2pix [Isola et al., 2017a] with added noise maps (bottom rows). AGE networks produce diverse colorizations, which are hard to obtain using pix2pix. (b) In each row we show the result of color transfer using the conditional AGE network. The color scheme from the first image is transferred onto the second image.

4.4 Conclusion

We have introduced a new approach for simultaneous learning of generation and inference networks. We have demonstrated how to set up such learning as an adversarial game between generation and inference, which has a different type of objective from traditional GAN approaches. In particular the objective of the game considers divergences between distributions rather than discrimination at the level of individual samples. As a consequence, our approach does not require training a discriminator network and enjoys relatively quick convergence.

We demonstrate that on a range of standard datasets, the generators obtained by our approach provides high-quality samples, and that the reconstrunctions of real data samples passed subsequently through the encoder and the generator are of better fidelity than in [Dumoulin et al., 2017]. We have also shown that our approach is able to generate plausible and diverse colorizations, which is not possible with the GAN-based system [Isola et al., 2017a].

Our approach leaves a lot of room for further experiments. In particular, a more complex latent space distribution can be chosen as in [Makhzani et al., 2016], and other divergence measures between distributions can be easily tried.

4.5 Proofs

Let X and Z be distributions defined in the data and the latent spaces $\mathcal{X}$, $\mathcal{Z}$ correspondingly. We assume X and Z are such, that there exists an invertible almost everywhere function e which transforms the latent distribution into the data one $g(Z) = X$. This assumption is weak, since for every atomless (i.e. no single point carries a positive mass) distributions X, Z such invertible function exists. For a detailed discussion on this topic please refer to Villani [2008], Marzouk et al. [2016]. Since Z is up to our choice simply setting it to Gaussian distribution (for $\mathcal{Z} = \mathbb{R}^M$) or uniform on sphere for ($\mathcal{Z} = \mathbb{S}^M$) is good enough.

Lemma 4.4. *Let X and Y to be two distributions defined in the same space. The distributions are equal $X = Y$ if and only if $e(X) = e(Y)$ holds for for any measurable function $e : \mathcal{X} \to \mathcal{Z}$.*

Proof. It is obvious, that if $X = Y$ then $e(X) = e(Y)$ for any measurable function e.

Now let $e(X) = e(Y)$ for any measurable e. To show that $X = Y$ we will assume converse: $X \neq Y$. Then there exists a set $B \in \mathcal{F}_X$, such that $0 < \mathbb{P}_X(B) \neq \mathbb{P}_Y(B)$ and a function e, such that corresponding set $C = e(B)$ has B as its preimage $B = e^{-1}(C)$. Then we have $\mathbb{P}_X(B) = \mathbb{P}_{e(X)}(C) = \mathbb{P}_{e(Y)}(C) = \mathbb{P}_Y(B)$, which contradicts with the previous assumption. □

Lemma 4.5. *Let (g', e') and (g^*, e^*) to be two different Nash equilibria in a game $\min_g \max_e V(g, e)$. Then $V(g, e) = V(g', e')$.*

Proof. See chapter 2 of Owen [1982]. □

Theorem 4.1. For a game

$$\max_e \min_g V_1(g, e) = \max_e \min_g \Delta(\, e(g(Z)) \| e(X)\,) \tag{4.8}$$

(g^*, e^*) is a saddle point of (4.8) if and only if g^* is such that $g^*(Z) = X$.

Proof. First note that $V_1(g, e) \geq 0$. Consider g such that $g(Z) = X$, then for any e: $V_1(g, e) = 0$. We conclude that (g, e) is a saddle point since $V_1(g, e) = 0$ is a maximum over e and minimum over g.

Using lemma 4.5 for saddle point (g^*, e^*): $V_1(g^*, e^*) = 0 = \max_e V_1(g^*, e)$, which is only possible if for all e: $V_1(g^*, e) = 0$ from which immediately follows $g(Z) = X$ by lemma 4.4. □

Lemma 4.6. *Let function $e : \mathcal{X} \to \mathcal{Z}$ be X-almost everywhere invertible, i.e. $\exists e^{-1}$: $\mathbb{P}_X(\{\mathbf{x} \neq e^{-1}(e(\mathbf{x}))\}) = 0$. Then if for a mapping $g : \mathcal{Z} \to \mathcal{X}$ holds $e(g(Z)) = e(X)$, then $g(Z) = X$.*

Proof. From definition of X-almost everywhere invertibility follows $\mathbb{P}_X(A) = \mathbb{P}_X(e^{-1}(e(A)))$ for any set $A \in \mathcal{F}_X$. Then:

$$\begin{aligned}\mathbb{P}_X(A) = \mathbb{P}_X(e^{-1}(e(A))) = \mathbb{P}_{e(X)}(e(A)) = \\ = \mathbb{P}_{e(g(Z))}(e(A)) = \mathbb{P}_{g(Z)}(A).\end{aligned}$$

Comparing the expressions on the sides we conclude $g(Z) = X$.

□

Theorem 4.2. Let Y to be any fixed distribution in the latent space. Consider a game

$$\max_e \min_g V_2(g, e) = \max_e \min_g \Delta(e(g(Z))\|Y) - \Delta(e(X)\|Y)\,. \tag{4.9}$$

If the pair (g^*, e^*) is a Nash equilibrium of game (4.9) then $g^*(Z) \sim X$. Conversely, if the fake and real distributions are aligned $g^*(Z) \sim X$ then (g^*, e^*) is a saddle point for *some* e^*.

Proof.

- As for a generator which aligns distributions $g(Z) = X$: $V_2(g, e) = 0$ for any e we conclude by 4.5 that the optimal game value is $V_2(g^*, e^*) = 0$. For an optimal pair (g^*, e^*) and arbitrary e' from the definition of equilibrium:

$$\begin{aligned}0 = \Delta(e^*(g^*(Z))\|Y) - \Delta(e^*(X)\|Y) \geq \\ \geq \Delta(e'(g^*(Z))\|Y) - \Delta(e'(X)\|Y)\,.\end{aligned} \tag{4.10}$$

  For invertible almost everywhere encoder e' such that $\Delta(e'(X)\|Y) = 0$ the first term is zero $\Delta(e'(g^*(Z))\|Y) = 0$ since inequality (4.10) and then $e'(g^*(Z)) = e'(X) = Y$. Using result of the lemma 4.6 we conclude, that $g^*(Z) = X$.

- If $g^*(Z) = X$ then $\forall e : e(g^*(Z)) = e(X)$ and $V_2(g^*, e^*) = V_2(g^*, e) = 0 = \max_{e'} V_2(g^*, e')$.

  The corresponding optimal encoder e^* is such that $g^* \in \arg\min_g \Delta(e^*(g(Z))\|Y)$.

□

Note that not for every optimal encoder e^* the distributions $e^*(X)$ and $e^*(g^*(Z))$ are aligned with Y. For example if e^* collapses $\mathcal{X}$ into two points then for any distribution X: $e^*(X) = e^*(g^*(Z)) = Bernoulli(p)$. For the optimal generator g^* the parameter p is such, that for all other generators g' such that $e^*(g'(Z)) \sim Bernoulli(p')$: $\Delta(e^*(g^*(Z))\|Y) \leq \Delta(e^*(g'(Z))\|Y)$.

Theorem 4.3. Let the two distributions W and Q be aligned by the mapping f (i.e. $f(W) = Q$) and let $\mathbb{E}_{\mathbf{w}\sim W}\|\mathbf{w} - h\left(f(\mathbf{w})\right)\|^2 = 0$. Then, for $\mathbf{w} \sim W$ and $\mathbf{q} \sim Q$, we have $\mathbf{w} = h(f(\mathbf{w}))$ and $\mathbf{q} = f(h(\mathbf{q}))$ almost certainly, i.e. the mappings f and h invert each other almost everywhere on the supports of W and Q. More, Q is aligned with W by h: $h(Q) = W$.

Proof. Since $\mathbb{E}_{\mathbf{w}\sim W}\|\mathbf{w} - h\left(f(\mathbf{w})\right)\|^2 = 0$, we have $\mathbf{w} = h(f(\mathbf{w}))$ almost certainly for $\mathbf{w} \sim W$. We can substitute $h(f(\mathbf{w}))$ with $\mathbf{w}$ under an expectation over W. Using this and the fact that $f(\mathbf{w}) \sim Q$ for $\mathbf{w} \sim W$ we derive:

$$\mathbb{E}_{\mathbf{q}\sim Q}\|\mathbf{q} - f\left(h(\mathbf{q})\right)\|^2 = \mathbb{E}_{\mathbf{w}\sim W}\|f(\mathbf{w}) - f\left(h(f(\mathbf{w}))\right)\|^2 = \\ = \mathbb{E}_{\mathbf{w}\sim W}\|f(\mathbf{w}) - f(\mathbf{w})\|^2 = 0\,.$$

Thus $\mathbf{q} = f(h(\mathbf{q}))$ almost certainly for $\mathbf{q} \sim Q$.

To show alignment $h(Q) = W$ first recall the definition of alignment. Distributions are aligned $f(W) = Q$ iff $\forall \bar{Q} \in \mathcal{F}_Q$: $\mathbb{P}_Q(\bar{Q}) = \mathbb{P}_W(f^{-1}(\bar{Q}))$. Then $\forall \bar{W} \in \mathcal{F}_W$ we have

$$\mathbb{P}_W(\bar{W}) = \mathbb{P}_W(h(f(\bar{W}))) = \mathbb{P}_W(f^{-1}(f(\bar{W}))) = \\ = \mathbb{P}_Q(f(\bar{W})) = \mathbb{P}_Q(h^{-1}(\bar{W}))\,.$$

Comparing the expressions on the sides we conclude $h(Q) = W$. □

Chapter 5

Image Manipulation with Perceptual Discriminators

Abstract

Systems that perform image manipulation using deep convolutional networks have achieved remarkable realism. Perceptual losses and losses based on adversarial discriminators are the two main classes of learning objectives behind these advances. In this work, we show how these two ideas can be combined in a principled and non-additive manner for unaligned image translation tasks. This is accomplished through a special architecture of the discriminator network inside generative adversarial learning framework. The new architecture, that we call a *perceptual discriminator*, embeds the convolutional parts of a pre-trained deep classification network inside the discriminator network. The resulting architecture can be trained on unaligned image datasets, while benefiting from the robustness and efficiency of perceptual losses. We demonstrate the merits of the new architecture in a series of qualitative and quantitative comparisons with baseline approaches and state-of-the-art frameworks for unaligned image translation.

This work was published as: Diana Sungatullina, Egor Zakharov, Dmitry Ulyanov and Victor Lempitsky. *Image Manipulation with Perceptual Discriminators*. European Conference on Computer Vision (ECCV), 2018.

5.1 Introduction

Generative convolutional neural networks have achieved remarkable success in image manipulation tasks both due to their ability to train on large amount of data [Jain and Seung, 2009, Kim et al., 2016, Dosovitskiy et al., 2015] and due to natural image priors associated with such architectures [Ulyanov et al., 2018]. Recently, the ability to train image manipulation ConvNets has been shown in the *unaligned* training scenario [Zhu et al., 2017b,c, Benaim and Wolf, 2017], where the training is based on sets of annotated with the presence/absence of a certain attribute, rather than based on *aligned* datasets containing {input,output} image pairs. The ability to train from unaligned data provides considerable flexibility in dataset collection and in learning new manipulation effects, yet poses additional algorithmic challenges.

Generally, the realism of the deep image manipulation methods is known to depend strongly on the choice of the loss functions that are used to train generative ConvNets. In particular, simplistic pixelwise losses (e.g. the squared distance loss) are known to limit the realism and are also non-trivial to apply in the unaligned training scenario. The rapid improvement of realism of deep image generation and processing is thus associated with two classes of loss functions that go beyond pixel-wise losses. The first group (so-called *perceptual losses*) is based on matching activations inside pre-trained deep convolutional networks (the VGG architecture trained for ILSVRC image classification is by far the most popular choice [Simonyan and Zisserman, 2014]). The second group consists of *adversarial losses*, where the loss function is defined implicitly using a separate *discriminator* network that is trained adversarially in parallel with the main generative network.

The two groups (perceptual losses and adversarial losses) are known to have largely complementary strengths and weaknesses. Thus, perceptual losses are easy to incorporate and are easy to scale to high-resolution images; however, their use in unaligned training scenario is difficult, as these loss terms require a concrete target image to match the activations to. Adversarial losses have the potential to achieve higher realism and can be used naturally in the unaligned scenarios, yet adversarial training is known to be hard to set up properly, often suffers from mode collapse, and is hard to scale to high-resolution images. Combining perceptual and adversarial losses in an additive way has been popular [Dosovitskiy and Brox, 2016b, Wang et al., 2017, Ledig et al., 2017a, Sajjadi et al., 2017a]. Thus, a generative ConvNet can be trained by minimizing a linear combination of an adversarial and a perceptual (and potentially some other) losses. Yet such additive combination includes not only strengths but also weaknesses of the two approaches. In particular, the use of a perceptual loss still incurs the use of aligned datasets for training.

In this work we present an architecture for realistic image manipulation, which combines perceptual and adversarial losses in a natural *non-additive* way. Importantly, the architecture keeps the ability of adversarial losses to train on unaligned datasets, while also benefits from the stability of perceptual losses. Our idea is very simple and concerned with the particular design of the discriminator network for adversarial training. The design encapsulates a pretrained classification network as the initial part of the discriminator. During adversarial training, the generator network is effectively learned to match the activations inside several layers of this reference network, just like the perceptual losses do. We show that the incorporation of the pretrained network into the discriminator stabilizes the training and scales well to higher resolution images, as is common with perceptual losses. At the same time, the use of adversarial training allows to avoid the need for aligned training data

Generally, we have found that the suggested architecture can be trained with little tuning to impose complex image manipulations, such as adding to and removing smile from human faces, face ageing and rejuvenation, gender change, hair style change, etc. In the experiments, we show that our architecture can be used to perform complex manipulations at medium and high resolutions, and compare the proposed architecture with several adversarial learning-based baselines and recent methods for learning-based image manipulation.

5.2 Related work

5.2.1 Generative ConvNets

Our approach is related to a rapidly growing body of works on ConvNets for image generation and editing. Some of the earlier important papers on ConvNet image generation [Dosovitskiy et al., 2015] and image processing [Jain and Seung, 2009, Dong et al., 2014a, Kim et al., 2016] used per-pixel loss functions and fully supervised setting, so that at test time the target image is known for each input. While this demonstrated the capability of ConvNets to generate realistic images, the proposed systems all had to be trained on aligned datasets and the amount of high-frequency details in the output images was limited due to deficiencies of pixel-wise loss functions.

5.2.2 Perceptual Losses

The work of Mahendran and Vedaldi [Mahendran and Vedaldi, 2015] has demonstrated that the activations invoked by an image within a pre-trained convolutional network

can be used to recover the original image. Gatys et al. [Gatys et al., 2015d] showed that such activations can serve as content descriptors or texture descriptors of the input image, while Dosovitsky and Brox [Dosovitskiy and Brox, 2016b], Ulyanov et al. [Ulyanov et al., 2016], Johnson et al. [Johnson et al., 2016] have shown that the mismatches between the produced and the target activations can be used as so-called *perceptual losses* for a generative ConvNet. The recent work of [Chen and Koltun, 2017] pushed the spatial resolution and the realism of images produced by a feed-forward ConvNet with perceptual losses to megapixel resolution. Generally, in all the above-mentioned works [Chen and Koltun, 2017, Ulyanov et al., 2016, Johnson et al., 2016, Dosovitskiy and Brox, 2016b], the perceptual loss is applied in a fully supervised manner as for each training example the specific target deep activations (or the Gram matrix thereof) are given explicitly. Finally, [Upchurch et al., 2017] proposed a method that manipulates carefully aligned face images at high resolution by compositing desired activations of a deep pretrained network and finding an image that matches such activations using the non-feedforward optimization process similar to [Mahendran and Vedaldi, 2015, Gatys et al., 2015d].

5.2.3 Adversarial Training

The most impressive results of generative ConvNets were obtained within generative adversarial networks (GANs) framework proposed originally by Goodfellow et al. [Goodfellow et al., 2014]. The idea of adversarial training is to implement the loss function as a separate trainable network (the *discriminator*), which is trained in parallel and in adversarial way with the generative ConvNet (the *generator*). Multiple follow-up works including [Radford et al., 2016, Salimans et al., 2016, Arjovsky et al., 2017, Karras et al., 2017a] investigated the choice of convolutional architectures for the generator and for the discriminator. Achieving reliable and robust convergence of generator-discriminator pairs remains challenging [Goodfellow, 2017, Chintala et al., 2017, Lucic et al., 2017], and in particular requires considerably more efforts than training with perceptual loss functions.

5.2.4 Unaligned Adversarial Training

While a lot of the original interest to GANs was associated with unconditional image generation, recently the emphasis has shifted to the conditional image synthesis. Most relevant to our work are adversarially-trained networks that perform image translation, i.e. generate output images conditioned on input images. While initial methods used

aligned datasets for training [Zhang et al., 2016, Isola et al., 2017b], recently some impressive results have been obtained using unaligned training data, where only empirical distributions of the input and the output images are provided [Zhu et al., 2017b, Benaim and Wolf, 2017, Zhu et al., 2017c]. For face image manipulation, systems using adversarial training on unaligned data have been proposed in [Brock et al., 2017, Choi et al., 2018]. While we also make an emphasis on face manipulation, our contribution is orthogonal to [Brock et al., 2017, Choi et al., 2018] as perceptual discriminators can be introduced into their systems.

5.2.5 Combining Perceptual and Adversarial Losses

A growing number of works [Dosovitskiy and Brox, 2016b, Ledig et al., 2017a, Wang et al., 2017] use the combination of perceptual and adversarial loss functions to accomplish more stable training and to achieve convincing image manipulation at high resolution. Most recently, [Sajjadi et al., 2017a] showed that augmenting perceptual loss with the adversarial loss improves over the baseline system [Chen and Koltun, 2017] (that has already achieved very impressive results) in the task of megapixel-sized conditional image synthesis. Invariably, the combination of perceptual and adversarial losses is performed in an additive manner, i.e. the two loss functions are weighted and added to each other (and potentially to some other terms). While such additive combination is simple and often very efficient, it limits learning to the aligned scenario, as perceptual terms still require to specify target activations for each training example. In this work, we propose a natural non-additive combination of perceptual losses and adversarial training that avoids the need for aligned data during training.

5.3 Perceptual discriminators

5.3.1 Background and motivation

Generative adversarial networks have shown impressive results in photorealistic image synthesis. The model includes a generative network G, that is trained to match the target distribution $p_{\text{target}}(\mathbf{y})$ in the data space $\mathcal{Y}$, and a discriminator network D that is trained to distinguish whether the input is real or generated by G. In the simplest form, the two networks optimize the policy function $V(D, G)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{y} \sim p_{\text{target}}(\mathbf{y})} \log D(\mathbf{y}) + \mathbb{E}_{\mathbf{x} \sim p_{\text{source}}(\mathbf{x})} [\log(1 - D(G(\mathbf{x}))], \tag{5.1}$$

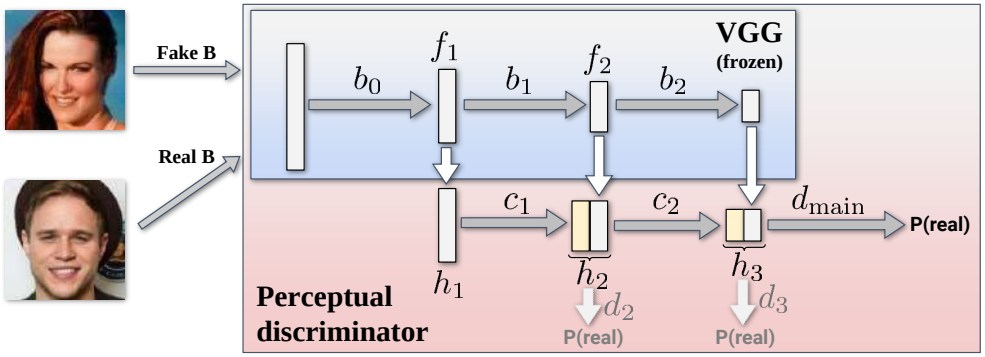


FIGURE 5.1: The perceptual discriminator is composed of a pre-trained image classification network (such as VGG), split into blocks b_i. The parameters of those blocks are not changed during training, thus the discriminator retains access to so-called perceptual features. The outputs of these blocks are processed using learnable blocks of convolutional operations c_i and the outputs of those are used to predict the probability of an image being real or manipulated (the simpler version uses a single discriminator d_{main}, while additional path discriminators are used in the full version).

In (5.1), the source distribution $p_{\text{source}}(\mathbf{x})$ may correspond to a simple parametric distribution in a latent space such as the unit Gaussian, so that after training unconditional samples from the learned approximation to $p_{\text{target}}(\mathbf{y})$ can be drawn. Alternatively, $p_{\text{source}}(\mathbf{x})$ may correspond to another empirical distribution in the image space $\mathcal{X}$. In this case, the generator learns to *translate* images from $\mathcal{X}$ to $\mathcal{Y}$, or to *manipulate* images in the space $\mathcal{X}$ (when it coincides with $\mathcal{Y}$). Although our contribution (perceptual discriminators) is applicable to both unconditional synthesis and image manipulation/translation, we focus our evaluation on the latter scenario. For the low resolution datasets, we use the standard non-saturating GAN modification, where the generator maximizes the log-likelihood of the discriminator instead of minimizing the objective (5.1) [Goodfellow et al., 2014]. For high-resolution images, following CycleGAN [Zhu et al., 2017b], we use the LSGAN formulation [Mao et al., 2016].

Converging to good equilibria for any of the proposed GAN games is known to be hard [Goodfellow, 2017, Chintala et al., 2017, Lucic et al., 2017]. In general, the performance of the trained generator network crucially depends on the architecture of the discriminator network, that needs to learn meaningful statistics, which are good for matching the target distribution p_{target}. The typical failure mode of GAN training is when the discriminator does not manage to learn such statistics before being "overpowered" by the generator.

5.3.2 Perceptual Discriminator Architecture

Multiple approaches have suggested to use activations invoked by an image $\mathbf{y}$ inside a deep pre-trained classification network $F(\mathbf{y})$ as statistics for such tasks as retrieval

[Babenko et al., 2014] or few-shot classification [Razavian et al., 2014]. Mahendran and Vedaldi [Mahendran and Vedaldi, 2015] have shown that activations computed after the convolutional part of such network retain most of the information about the input $\mathbf{y}$, i.e. are essentially invertable. Subsequent works such as [Gatys et al., 2015d, Ulyanov et al., 2016, Johnson et al., 2016, Dosovitskiy and Brox, 2016b] all used such "perceptual" statistics to match low-level details such as texture content, certain image resolution, or particular artistic style.

Following this line of work, we suggest to base the GAN discriminator $D(\mathbf{y})$ on the perceptual statistics computed by the reference network F on the input image $\mathbf{y}$, which can be either real (coming from p_{target}) or fake (produced by the generator). Our motivation is that a discriminator that uses perceptual features has a better chance to learn good statistics than a discriminator initialized to a random network. For simplicity, we assume that the network F has a chain structure, e.g. F can be the VGGNet of [Simonyan and Zisserman, 2014].

Consider the subsequent blocks of the convolutional part of the reference network F, and denote them as $b_0, b_1, \ldots, b_{K-1}$. Each block may include one or more convolutional layers interleaved with non-linearities and pooling operations. Then, the perceptual statistics $\{f_1(\mathbf{y}), \ldots, f_K(\mathbf{y})\}$ are computed as:

$$f_1(\mathbf{y}) = b_0(\mathbf{y}) \tag{5.2}$$

$$f_i(\mathbf{y}) = b_{i-1}(f_{i-1}(\mathbf{y})), \quad i = 2, \ldots, K\,, \tag{5.3}$$

so that each $f_i(\mathbf{y})$ is a stack of convolutional maps of the spatial dimensions $W_i \times W_i$. The dimension W_i is determined by the preceeding size W_{i-1} as well as by the presence of strides and pooling operations inside b_i. In our experiments we use features from consecutive blocks, i.e. $W_i = W_{i-1}/2$.

The overall structure of our discriminator is shown in Figure 5.1. The key novelty of our discriminator is the in-built perceptual statistics f_i (top of the image), which are known to be good at assessing image realism [Gatys et al., 2015d, Johnson et al., 2016, Upchurch et al., 2017]. During the backpropagation, the gradients to the generator flow through the perceptual statistics extractors b_i, but the parameters of b_i are frozen and inherited from the network pretrained for large-scale classification. This stabilizes the training, and ensures that at each moment of time the discriminator has access to "good" features, and therefore cannot be overpowered by the generator easily.

In more detail, the proposed discriminator architecture combines together perceptual statistics using the following computations:

$$h_1(\mathbf{y}) = f_1(\mathbf{y}) \tag{5.4}$$

$$h_i(\mathbf{y}) = \texttt{stack}\left[c_{i-1}(h_{i-1}(\mathbf{y}), \phi_{i-1}),\, f_i(\mathbf{y})\right], \quad i = 2, \ldots, K, \tag{5.5}$$

where `stack` denotes stacking operation, and the convolutional blocks c_j with learnable parameters ϕ_j (for $j = 1, \ldots, K-1$) are composed of convolutions, leaky ReLU non-linearities, and average pooling operations. Each of the c_j blocks thus transforms map stacks of the spatial size $W_j \times W_j$ to map stacks of the spatial size $W_{j+1} \times W_{j+1}$. Thus, the strides and pooling operations inside c_j match the strides and/or pooling operations inside b_j.

Using a series of convolutional and fully-connected layers with learnable parameters ψ_{main} applied to the representation $h_K(\mathbf{y})$, the discriminator outputs the probability d_{main} of the whole image $\mathbf{y}$ being real. For low- to medium- resolution images we perform experiments using only this probability. For high-resolution, we found that additional outputs from the discriminator resulted in better outcomes. Using the "patch discriminator" idea [Isola et al., 2017b, Zhu et al., 2017b], to several feature representations h_j we apply a convolution+LeakyReLU block d_j with learnable parameters ψ_j that outputs probabilities $d_{j,p}$ at every spatial locations p. We then replace the regular log probability $\log D(\mathbf{y}) \equiv \log d_{\text{main}}$ of an image being real with:

$$\log D(\mathbf{y}) = \log d_{\text{main}}(\mathbf{y}) + \sum_j \sum_{p \in \text{Grid}(W_j \times W_j)} \log d_{j,p}(\mathbf{y}) \tag{5.6}$$

Note, that this makes our discriminator "multi-scale", since spatial resolution W_j varies for different j. The idea of multiple classifiers inside the discriminator have also been proposed recently in [Wang et al., 2017, Iizuka et al., 2017a]. Unlike [Wang et al., 2017, Iizuka et al., 2017a] where these classifiers are disjoint, in our architecture all such classifiers are different branches of the same network that has perceptual features underneath.

During training, the parameters of the c blocks inside the feature network F remain fixed, while the parameters ϕ_i of feature extractors c_i and the parameters ψ_i of the discriminators d_i are updated during the adversarial learning, which forces the "perceptual" alignment between the output of the generator and p_{target}. Thus, wrapping perceptual loss terms into additional layers c_i and d_i and putting them together into the adversarial discriminator allows us to use such perceptual terms in the unaligned training scenario. Such unaligned training was, in general, not possible with the "traditional" perceptual losses.

5.3.3 Architecture Details

5.3.3.1 Reference Network

Following multiple previous works [Gatys et al., 2015d, Ulyanov et al., 2016, Johnson et al., 2016], we consider the so-called *VGG network* from [Simonyan and Zisserman, 2014] trained on ILSVRC2012 [Russakovsky et al., 2014] as the reference network F. In particular, we pick the VGG-19 variant, to which we simply refer to as VGG. While the perceptual features from VGG already work well, the original VGG architecture can be further improved. Radford et. al [Radford et al., 2016] reported that as far as leaky ReLU avoids sparse gradients, replacing ReLUs with leaky ReLUs [He et al., 2015b] in the discriminator stabilizes the training process of GANs. For the same reasons, changing max pooling layers to average pooling removes unwanted sparseness in the backpropagated gradients. Following these observations, we construct the *VGG** network, which is particularly suitable for the adversarial game. We thus took the VGG-19 network pretrained on ILSVRC dataset, replaced all max pooling layers by average poolings, ReLU nonlinearities by leaky ReLUs with a negative slope 0.2 and then trained on the ILSVRC dataset for further two days. We compare the variants of our approach based on VGG and VGG* features below.

5.3.3.2 Generator Architecture

For the image manipulation experiments, we used transformer network proposed by Johnson et al. [Johnson et al., 2016]. It consists of M convolutional layers with stride size 2, N residual blocks [He et al., 2015a] and M upsampling layers, each one increases resolution by a factor of 2. We set M and N in a way that allows outputs of the last residual block to have large enough receptive field, but at the same time for generator and discriminator to have similar number of parameters. We provide detailed descriptions of architectures in [sup, 2018].

5.3.3.3 Stabilizing the Generator

We have also used two additional methods to improve the generator learning and to prevent its collapse. First, we have added the *identity loss* [Taigman et al., 2016, Zhu et al., 2017b] that ensures that the generator does not change the input, when it comes from the p_{target}. Thus, the following term is added to the maximization objective of the generator:

$$J_{\text{id}}^{G} = -\lambda_{\text{id}}\, \mathbb{E}_{\mathbf{y}\sim p_{\text{target}}} \lambda \big\| \mathbf{y} - G(\mathbf{y}) \big\|_{L_1}, \tag{5.7}$$

where λ_{id} is a meta-parameter that controls the contribution of the weight, and $\|\cdot\|_{L_1}$ denotes pixel-wise L1-metric.

To achieve the best results for the hardest translation tasks, we have found the cycle idea from the CycleGAN [Zhu et al., 2017b] needed. We thus train two generators $G_{\mathbf{x}\to\mathbf{y}}$ and $G_{\mathbf{y}\to\mathbf{x}}$ operating in opposite directions in parallel (and jointly with two discriminators), while adding reciprocity terms ensuring that mappings $G_{\mathbf{x}\to\mathbf{y}} \circ G_{\mathbf{y}\to\mathbf{x}}$ and $G_{\mathbf{y}\to\mathbf{x}} \circ G_{\mathbf{x}\to\mathbf{y}}$ are close to identity mappings.

Moreover, we notice that usage of external features as inputs for the discriminator leads to fast convergence of the discriminator loss to zero. Even though this is expected, since our method essentially corresponds to pretraining of the discriminator, this behavior is one of the GAN failure cases [Chintala et al., 2017] and on practice leads to bad results in harder tasks. Therefore we find pretraining of the generator to be required for increased stability. For image translation task we pretrain generator as autoencoder. Moreover, the necessity to pretrain the generator makes our approach fail to operate in DCGAN setting with unconditional generator.

After an additional stabilization through the pretraining and the identity and/or cycle losses, the generator becomes less prone to collapse. Overall, in the resulting approach it is neither easy for the discriminator to overpower the generator (this is prevented by the identity and/or cycle losses), nor is it easy for the generator to overpower the discriminator (as the latter always has access to perceptual features, which are good at judging the realism of the output).

5.4 Experiments

The goal of the experimental validation is two-fold. The primary goal is to validate the effect of perceptual discriminators as compared to baseline architectures which use traditional discriminators that do not have access to perceptual features. The secondary goal is to validate the ability of our full system based on perceptual discriminators to handle harder image translation/manipulation task with higher resolution and with less data. Extensive additional results are available on our project page [sup, 2018]. We perform the bulk of our experiments on CelebA dataset [Liu et al., 2015b], due to its large size, popularity and the availability of the attribute annotations (the dataset comprises over 200k of roughly-aligned images with 40 binary attributes; we use 160×160 central crops of the images). As harder image translation task, we use CelebA-HQ [Karras et al., 2017a] dataset, which consists of high resolution versions of images from CelebA

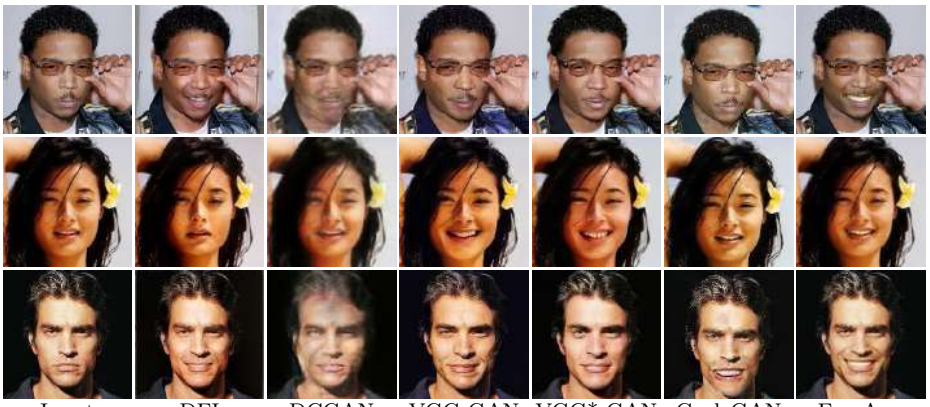


FIGURE 5.2: Qualitative comparison of the proposed systems as well as baselines for neutral→smile image manipulation. As baselines, we show the results of DFI (perceptual features, no adversarial training) and DCGAN (same generator, no perceptual features in the discriminator). Systems with perceptual discriminators output more plausible manipulations.

and is smaller in size. Lastly, we evaluate our model on problems with non-face datasets like apples to oranges and photo to Monet texture transfer tasks.

Experiments were carried out on NVIDIA DGX-2 server.

5.4.1 Qualitative Comparison on CelebA

Even though our contribution is orthogonal to a particular GAN-based image translation method, we chose one of them, provided modifications we proposed and compared it with the following important baselines in an attribute manipulation task:

- *DCGAN* [Radford et al., 2016]: in this baseline GAN system we used image translation model with generator and discriminator trained only with adversarial loss.
- *CycleGAN* [Zhu et al., 2017b]: this GAN-based method learns two reciprocal transforms in parallel with two discriminators in two domains. We have used the authors' code (PyTorch version).
- *DFI* [Upchurch et al., 2017]: to transform an image, this approach first determines target VGG feature representation by adding the feature vector corresponding to input image and the shift vector calculated using nearest neighbours in both domains. Then the resulting image is produced using optimization-based feature inversion as in [Mahendran and Vedaldi, 2015]. We have used the authors' code.
- *FaceApp* [Fac, 2018]: is a very popular closed-source app that is known for the quality of its filters (transforms), although the exact algorithmic details are unknown.

	(a) User study		(b) C2ST, $\times 10^{-2}$			(c) Classification loss		
	Smile	Age	Smile	Gender	Hair color	Smile	Gender	Hair color
DFI [Upchurch et al., 2017]	0.16	0.4	< 0.1	< 0.01	< 0.01	1.3	0.5	1.14
FaceApp [Fac, 2018]	0.45	0.41	–	–	–	–	–	–
DCGAN [Radford et al., 2016]	–	–	0.6	0.03	0.06	0.6	1.5	2.33
CycleGAN [Zhu et al., 2017b]	0.03	0.04	5.3	0.35	0.49	1.2	0.8	2.41
VGG-GAN	–	–	8.6	0.21	0.96	0.4	0.1	1.3
VGG*-GAN	0.36	0.15	5.2	0.24	1.29	0.7	0.1	1.24
Real data	–	–	–	–	–	0.1	0.01	0.56

TABLE 5.1: Quantitative comparison: (a) Photorealism user study. We show the fraction of times each method has been chosen as "the best" among all in terms of photorealism and identity preservation (the higher the better). (b) C2ST results (cross-entropy, the higher the better). (c) Log-loss of classifier trained on real data for each class (the lower the better). See main text for details.

Our model is represented by two basic variants.

- *VGG-GAN*: we use DCGAN as our base model. The discriminator has a single classifier and no generator pretraining or regularization is applied, other than identity loss mentioned in the previous section.
- *VGG*-GAN*: same as the previous model, but we use a finetuned VGG network variant with dense gradients.

The comparison with state-of-the-art image transformation systems is performed to verify the competitiveness of the proposed architecture (Figure 5.2). In general, we observe that VGG*-GAN and VGG-GAN models consistently outperformed DCGAN variant, achieving higher effective resolution and obtaining more plausible high-frequency details in the resulting images. While a more complex CycleGAN system is also capable of generating crisp images, we found that the synthesized smile often does not look plausible and does not match the face. DFI turns out to be successful in attribute manipulation, yet often produces undesirable artifacts, while FaceApp shows photorealistic results, but with low attribute diversity. Here we also evaluate the contribution of dense gradients idea for VGG encoder and find it providing minor quality improvements.

5.4.2 User Photorealism Study on CelebA

We have also performed an informal user study of the photorealism. The study enrolled 30 subjects unrelated to computer vision and evaluated the photorealism of VGG*-GAN, DFI, CycleGAN and FaceApp on smile and aging/rejuvenation transforms. To assess the photorealism, the subjects were presented quintuplets of photographs unseen during training. In each quintuplet the center photo was an image without the target attribute

FIGURE 5.3: Results for VGG*-MS-CycleGAN attribute editing at 256×256 resolution on Celeba-HQ dataset. Networks have been trained to perform pairwise domain translation between the values of hair color, gender and smile attributes. Digital zoom-in recommended. See [sup, 2018] for more manipulation examples.

(e. g. real photo of neutral expression), while the other four pictures were manipulated by one of the methods and presented in random order. The subjects were then asked to pick one of the four manipulations that they found most plausible (both in terms of realism and identity preservation). While there was no hard time limit, the users were asked to make the pick as quickly as possible. Each subject was presented overall 30 quintuplets with 15 quantuplets allocated for each of the considered attribute. The results in Table 6.1a show that VGG*-GAN is competitive and in particular considerably better than the other feed-forward method in the comparison (CycleGAN), but FaceApp being the winner overall. This comes with the caveat that the training set of FaceApp is likely to be bigger than CelebA. We also speculate that the diversity of smiles in FaceApp seems to be lower (Figure 5.2), which is the deficiency that is not reflected in this user study.

5.4.3 Quantitative Results on CelebA

To get objective performance measure, we have used the classifier two-sample test (C2ST) [Lopez-Paz and Oquab, 2016] to quantitatively compare GANs with the proposed discriminators to other methods. For each method, we have thus learned a separate classifier to discriminate between hold-out set of real images from target distribution and synthesized images, produced by each of the methods. We split both hold-out set and the set of fake images into training and testing parts, fit the classifier to the training set and report the log-loss over the testing set in the Table section 5.4.1b. The results comply with the qualitative observations: artifacts, produced by DCGAN and DFI are being easily detected by the classifier resulting in a very low log-loss. The proposed system stays on par with a more complex CycleGAN (better on two transforms out of three), proving that a perceptual discriminator can remove the need in two additional networks and cycle losses. Additionally, we evaluated attribute translation performance in a similar fashion to StarGAN [Choi et al., 2018]. We have trained a model for attribute classification on CelebA and measured average log-likelihood for the synthetic and real data to belong to the target class. Our method achieved lower log-loss than other methods on two out of three face attributes (see Table section 5.4.1c).

5.4.4 Higher Resolution

We further evaluate our model on CelebA-HQ dataset. Here in order to obtain high quality results we use all proposed regularization methods. We refer to our best model as VGG*-MS-CycleGAN, which corresponds to the usage of VGG* network with dense gradients as an encoder, multi-scale perceptual discriminator based on VGG* network, CycleGAN regularization and pretraining of the generator. Following CycleGAN, we use LSGAN [Mao et al., 2016] as an adversarial objective for that model. We trained on 256×256 version of CelebA-HQ dataset and present attribute manipulation results in Figure 5.3. As we can see, our model provides photorealistic samples while capturing differences between the attributes even for smaller amount of training samples (few thousands per domain) and higher resolution compared to our previous tests.

In order to ensure that each of our individual contributions affects the quality of these results, we consider three variations of our discriminator architecture and compare them to the alternative multi-scale discriminator proposed in Wang et al. [Wang et al., 2017]. While Wang et al. used multiple identical discriminators operating at different scales, we argue that this architecture has redundancy in terms of number of parameters and can be reduced to our architecture by combining these discriminators into a single network with shared trunk and separate multi-scale output branches (as is done in our method).

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

FIGURE 5.4: We compare different architectures for the discriminator on CelebA-HQ 256×256 male $\leftrightarrow$ female problem. We train all architectures in CycleGAN manner with LSGAN objective and compare different discriminator architectures. (a) Input, (b) VGG*-MS-CycleGAN: multi-scale perceptual discriminator with pretrained VGG* as a feature network F, (c) Rand-MS-CycleGAN: multi-scale perceptual discriminator with a feature network F having VGG* architecture with randomly-initialized weights, (d) MS-CycleGAN: multi-scale discriminator with the trunk shared across scales (as in our framework), where images serve as a direct input, (e) separate multi-scale discriminators similar to Wang et al. [Wang et al., 2017]. Digital zoom-in recommended.

Input CycleGAN VGG*-CycleGAN Input CycleGAN VGG*-CycleGAN

FIGURE 5.5: Comparison between CycleGAN and VGG*-MS-CycleGAN on painting$\leftrightarrow$photo translation task. It demonstrates the applicability of our approach beyond face image manipulation. See [sup, 2018] for more examples.

Both variants are included into the comparison in Figure 5.4. Also we consider *Rand-MS-CycleGAN* baseline that uses random weights in the feature extractor in order to tease apart the contribution of VGG* architecture as a feature network F and the effect of also having its weights pretrained on the success of the adversarial training. While the weights inside the VGG part were not frozen, so that adversarial training process could theoretically evolve good features in the discriminator, we were unable to make this baseline produce reasonable results. For high weight of the identity loss λ_{id} the resulting generator network produced near-identical results to the inputs, while decreasing λ_{id} lead to severe generator collapse. We conclude that the architecture alone cannot explain the good performance of perceptual discriminators (which is validated below) and that having pretrained weights in the feature network is important.

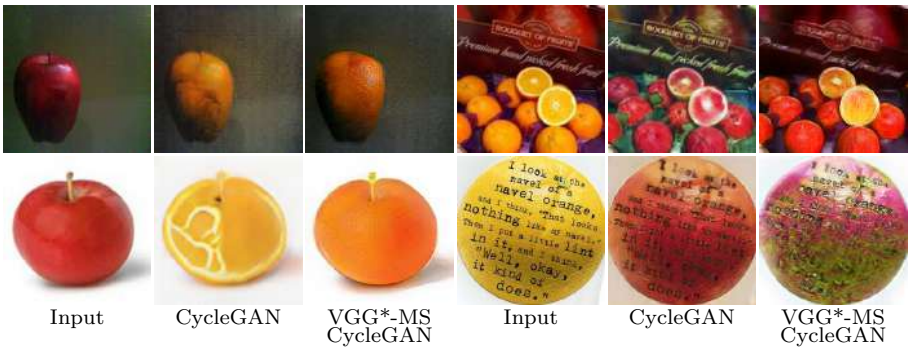


FIGURE 5.6: Apple↔orange translation samples with CycleGAN and VGG*-MS-CycleGAN are shown. Zoom-in recommended. See [sup, 2018] for more examples.

5.4.5 Non-face Datasets

While the focus of our evaluation was on face attribute modification tasks, our contribution applies to other translation tasks, as we verify in this section by performing qualitative comparison with the CycleGAN and VGG*-MS-CycleGAN architectures on two non-face domains on which CycleGAN was originally evaluated: an artistic style transfer task (Monet-photographs) in Figure 5.5 and an apple-orange conversion in Figure 5.6 (the figures show representative results). To achieve fair comparison, we use the same amount of residual blocks and channels in the generator and the same number of downsampling layers and initial amount of channels in discriminator both in our model and in the original CycleGAN. We used the authors' implementation of CycleGAN with default parameters. While the results on the style transfer task are inconclusive, for the harder apple-to-orange task we generally observe the performance of perceptual discriminators to be better.

5.4.6 Other Learning Formulations

Above, we have provided the evaluation of the perceptual discriminator idea to unaligned image translation tasks. In principle, perceptual discriminators can be used for other tasks, e.g. for unconditional generation and aligned image translation. In our preliminary experiments, we however were not able to achieve improvement over properly tuned baselines. In particular, for aligned image translation (including image superresolution) an additive combination of standard discriminator architectures and perceptual losses performs just as well as our method. This is not surprising, since the presence of alignment means that perceptual losses can be computed straight-forwardly, while they also stabilize the GAN learning in this case. For unconditional image generation, a naive application of our idea leads to discriminators that quickly overpower generators in the initial stages of the game leading to learning collapse.

5.5 Summary

We have presented a new discriminator architecture for adversarial training that incorporates perceptual loss ideas with adversarial training. We have demonstrated its usefulness for unaligned image translation tasks, where the direct application of perceptual losses is infeasible. Our approach can be regarded as an instance of a more general idea of using transfer learning, so that easier discriminative learning formulations can be used to stabilize and improve GANs and other generative learning formulations.

Chapter 6

Textured Neural Avatars

Abstract

We present a system for learning full-body *neural avatars*, i.e. deep networks that produce full-body renderings of a person for varying body pose and camera position. Our system takes the middle path between the classical graphics pipeline and the recent deep learning approaches that generate images of humans using image-to-image translation. In particular, our system estimates an explicit two-dimensional texture map of the model surface. At the same time, it abstains from explicit shape modeling in 3D. Instead, at test time, the system uses a fully-convolutional network to directly map the configuration of body feature points w.r.t. the camera to the 2D texture coordinates of individual pixels in the image frame. We show that such a system is capable of learning to generate realistic renderings while being trained on videos annotated with 3D poses and foreground masks. We also demonstrate that maintaining an explicit texture representation helps our system to achieve better generalization compared to systems that use direct image-to-image translation.

This work was published as: Aliaksandra Shysheya, Egor Zakharov, Kara-Ali Aliev, Renat Bashirov, Egor Burkov, K Iskakov, A Ivakhnenko, Y Malkov, Igor Pasechnik, Dmitry Ulyanov, A Vakhitov, V Lempitsky *Textured Neural Avatars*. Computer Vision and Pattern Recognition (CVPR), 2019.

6.1 Introduction

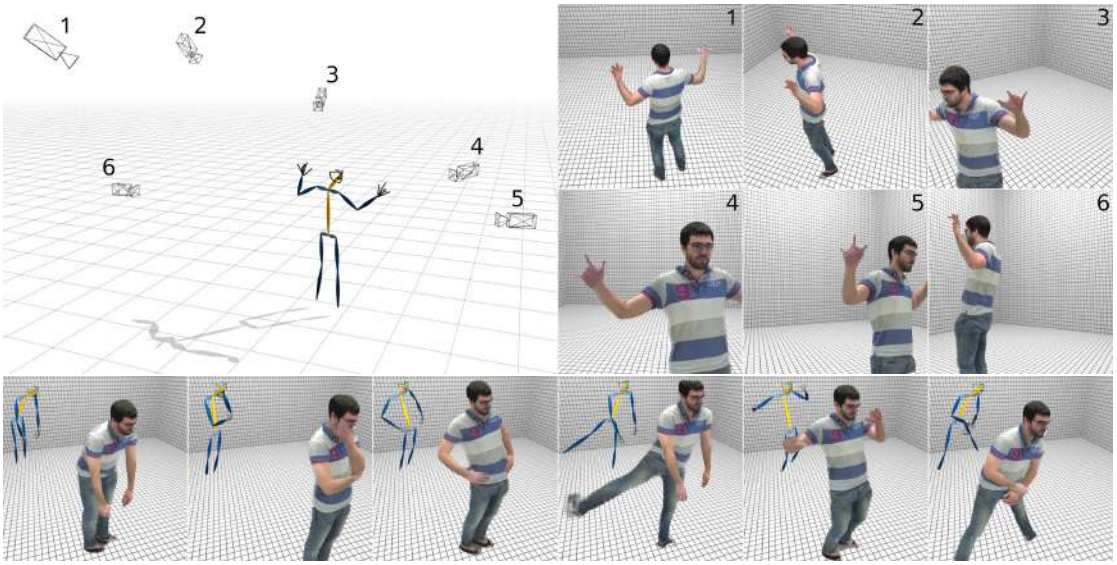


FIGURE 6.1: We propose a new model for neural rendering of humans. The model is trained for a single person and can produce renderings of this person from novel viewpoints (top) or in the new body pose (bottom) unseen during training. To improve generalization, our model retains explicit texture representation, which is learned alongside the rendering neural network.

Capturing and rendering human body in all of its complexity under varying pose and imaging conditions is one of the core problems of both computer vision and computer graphics. Recently, there is a surge of interest that involves deep convolutional networks (ConvNets) as an alternative to traditional computer graphics means. Realistic *neural rendering* of body fragments e.g. faces [Kim et al., 2018, Lombardi et al., 2018, Suwajanakorn et al., 2017], eyes [Ganin et al., 2016], hands [Mueller et al., 2018] is now possible. Very recent works have shown the abilities of such networks to generate views of a person with a varying body pose but with a fixed camera position, and using an excessive amount of training data [Aberman et al., 2018, Chan et al., 2018, Liu et al., 2018, Wang et al., 2018]. In this work, we focus on the learning of *neural avatars*, i.e. generative deep networks that are capable of rendering views of individual people under varying body pose defined by a set of 3D positions of the body joints and under varying camera positions (Figure 6.1). We prefer to use body joint positions to represent the human pose, as joint positions are often easier to capture using marker-based or marker-less motion capture systems.

Generally, neural avatars can serve as an alternative to classical ("neural-free") avatars based on a standard computer graphics pipeline that estimates a user-personalized body mesh in a neutral position, performs skinning (deformation of the neutral pose), and projects the resulting 3D surface onto the image coordinates, while superimposing

person-specific 2D texture. Neural avatars attempt to shortcut the multiple stages of the classical pipeline and to replace them with a single network that learns the mapping from the input (the location of body joints) to the output (the 2D image). As a part of our contribution, we demonstrate that, however appealing for its conceptual simplicity, existing pose-to-image translation networks generalize poorly to new camera views, and therefore new architectures for neural avatars are required.

Towards this end, we present a neural avatar system that does full-body rendering and combines the ideas from the classical computer graphics, namely the decoupling of geometry and texture, with the use of deep convolutional neural networks. In particular, similarly to the classic pipeline, our system explicitly estimates the 2D textures of body parts. The 2D texture within the classical pipeline effectively transfers the appearance of the body fragments across camera transformations and body articulations. Keeping this component within the neural pipeline boosts generalization across such transforms. The role of the convolutional network in our approach is then confined to predicting the texture coordinates of individual pixels in the output 2D image given the body pose and the camera parameters (Figure 6.2). Additionally, the network predicts the body foreground/background mask.

In our experiments, we compare the performance of our *textured neural avatar* with a direct video-to-video translation approach [Wang et al., 2018], and show that explicit estimation of textures brings additional generalization capability and improves the realism of the generated images for new views and/or when the amount of training data is limited.

6.2 Related work

Our approach is closely related to a vast number of previous works, and below we discuss a small subset of these connections.

Building **full-body avatars** from image data has long been one of the main topics of computer vision research. Traditionally, an avatar is defined by a 3D geometric mesh of a certain neutral pose, a texture, and a skinning mechanism that transforms the mesh vertices according to pose changes. A large group of works has been devoted to body modeling from 3D scanners [Pons-Moll et al., 2015], registered multi-view sequences [Robertini et al., 2017] as well as from depth and RGB-D sequences [Bogo et al., 2015, Weiss et al., 2011, Yu et al., 2018]. On the other extreme are methods that fit skinned parametric body models to single images [Bălan and Black, 2008, Bogo et al., 2016, Hasler et al., 2010, Kanazawa et al., 2018, Omran et al., 2018, Pavlakos et al., 2018,

Starck and Hilton, 2003]. Finally, research on building full-body avatars from monocular videos has started [Alldieck et al., 2018b,a]. Similarly to the last group of works, our work builds an avatar from a video or a set of unregistered monocular videos. The classical (computer graphics) approach to modeling human avatars requires explicit physically-plausible modeling of human skin, hair, sclera, clothing surface, as well as motion under pose changes. Despite considerable progress in reflectivity modeling [Alexander et al., 2010, Donner et al., 2008, Klehm et al., 2015, Weyrich et al., 2006, Wood et al., 2015] and better skinning/dynamic surface modeling [Feng et al., 2015, Loper et al., 2015, Stavness et al., 2014], the computer graphics approach still requires considerable "manual" effort of designers to achieve high realism [Alexander et al., 2010] and to pass the so-called uncanny valley [Mori, 1970], especially if real-time rendering of avatars is required.

Image synthesis using deep convolutional neural networks is a thriving area of research [Dosovitskiy et al., 2015, Goodfellow et al., 2014] and a lot of recent effort has been directed onto synthesis of realistic human faces [Choi et al., 2018, Karras et al., 2018a, Sungatullina et al., 2018]. Compared to traditional computer graphics representations, deep ConvNets model data by fitting an excessive number of learnable weights to training data. Such ConvNets avoid explicit modeling of the surface geometry, surface reflectivity, or surface motion under pose changes, and therefore do not suffer from the lack of realism of the corresponding components. On the flipside, the lack of ingrained geometric or photometric models in this approach means that generalizing to new poses and in particular to new camera views may be problematic. Still a lot of progress has been made over the last several years for the neural modeling of personalized talking head models [Kim et al., 2018, Lombardi et al., 2018, Suwajanakorn et al., 2017], hair [Wei et al., 2018], hands [Mueller et al., 2018]. Notably, the recent system [Lombardi et al., 2018] has achieved very impressive results for neural face rendering, while decomposing view-dependent texture and 3D shape modeling.

Over the last several months, several groups have presented results of neural modeling of full bodies [Aberman et al., 2018, Chan et al., 2018, Liu et al., 2018, Wang et al., 2018]. While the presented results are very impressive, the approaches still require a large amount of training data. They also assume that the test images are rendered with the same camera views as the training data, which in our experience makes the task considerably simpler than modeling body appearance from an arbitrary viewpoint. In this work, we aim to expand the neural body modeling approach to tackle the latter, harder task. The work [Martin-Brualla et al., 2018] uses a combination of classical and neural rendering to render human body from new viewpoints, but does so based on depth scans and therefore with a rather different algorithmic approach.

A number of recent works **warp a photo of a person** to a new photorealistic image with modified gaze direction [Ganin et al., 2016], modified facial expression/pose [Cao et al., 2018, Shu et al., 2018, Tulyakov et al., 2018, Wiles et al., 2018], or modified body pose [Balakrishnan et al., 2018, Neverova et al., 2018, Siarohin et al., 2018, Tulyakov et al., 2018], whereas the warping field is estimated using a deep convolutional network (while the original photo effectively serves as a texture). These approaches are however limited in their realism and/or the amount of change they can model, due to their reliance on a single photo of a given person for its input. Our approach also disentangles texture from surface geometry/motion modeling but trains from videos, therefore being able to handle harder problem (full body multi-view setting) and to achieve higher realism.

Our system relies on the **DensePose** body surface parameterization (UV parameterization) similar to the one used in the classical graphics-based representation. Part of our system performs a mapping from the body pose to the surface parameters (UV coordinates) of image pixels. This makes our approach related to the DensePose approach [Güler et al., 2018] and the earlier works [Güler et al., 2017, Taylor et al., 2012] that predict UV coordinates of image pixels from the input photograph. Furthermore, our approach uses DensePose results [Güler et al., 2018] for pretraining.

Our system is related to approaches that extract **textures from multi-view image collections** [Goldlücke and Cremers, 2009, Lempitsky and Ivanov, 2007] or multi-view video collections [Volino et al., 2014] or a single video [Rav-Acha et al., 2008]. Our approach is also related to free-viewpoint video compression and rendering systems, e.g. [Casas et al., 2014, Collet et al., 2015, Dou et al., 2017, Volino et al., 2014]. Unlike those works, ours is restricted to scenes containing a single human. At the same time, our approach aims to generalize not only to new camera views but also to new user poses unseen in the training videos. The work of [Xu et al., 2011] is the most related to ours in this group, as they warp the individual frames of the multi-view video dataset according to the target pose to generate new sequences. The poses that they can handle, however, are limited by the need to have a close match in the training set, which is a strong limitation given the combinatorial nature of the human pose configuration space.

6.3 Methods

6.3.1 Notation

We use the lower index i to denote objects that are specific to the i-th training or test image. We use uppercase notation, e.g. B_i to denote a stack of maps (a third-order tensor/three-dimensional array) corresponding to the i-th training or test image. We use

the upper index to denote a specific map (channel) in the stack, e.g. B_i^j. Furthermore, we use square brackets to denote elements corresponding to a specific image location, e.g. $B_i^j[x, y]$ denotes the scalar element in the j-th map of the stack B_i located at location (x, y), and $B_i[x, y]$ denotes the vector of elements corresponding to all maps sampled at location (x, y).

6.3.2 Input and output

In general, we are interested in synthesizing images of a certain person given her/his pose. We assume that the pose for the i-th image comes in the form of 3D joint positions defined in the camera coordinate frame. As an input to the network, we then consider a map stack B_i, where each map B_i^j contains the rasterized j-th segment (bone) of the "stickman" (skeleton) projected on the camera plane. To retain the information about the third coordinate of the joints, we linearly interpolate the depth value between the joints defining the segments, and use the interpolated values to define the values in the map B_i^j corresponding to the bone pixels (the pixels not covered by the j-th bone are set to zero). Overall, the stack B_i incorporates the information about the person and the camera pose.

As an output of the whole system, we expect an RGB image (a three-channel stack) I_i and a single channel mask M_i, defining the pixels that are covered by the avatar. Below, we consider two approaches: the *direct translation* baseline, which directly maps B_i into $\{I_i, M_i\}$ and the *textured neural avatar* approach that performs such mapping indirectly using texture mapping.

In both cases, at training time, we assume that for each input frame i, the input joint locations and the "ground truth" foreground mask are estimated, and we use 3D body pose estimation and human semantic segmentation to extract them from raw video frames. At test time, given a real or synthetic background image $\tilde{I}_i$, we generate the final view by first predicting M_i and I_i from the body pose and then linearly blending the resulting avatar into an image: $\hat{I}_i = I_i \odot M_i + \tilde{I}_i \odot (1 - M_i)$ (where $\odot$ defines a "location-wise" product, i.e. the RGB values at each location are multiplied by the mask value at this location).

6.3.3 Direct translation baseline

The direct approach that we consider as a baseline to ours is to learn an image translation network that maps the map stack B_i^k to the map stacks I_i and M_i (usually the two output stacks are produced within two branches that share the initial stage of the

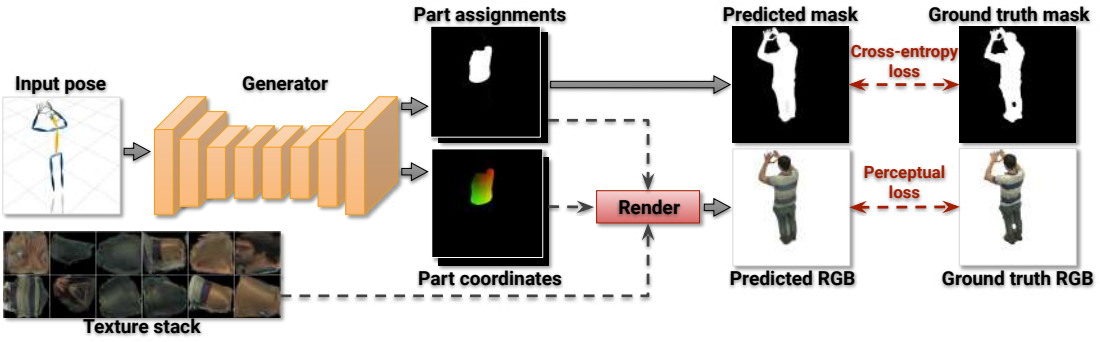


FIGURE 6.2: The overview of the textured neural avatar system. The input pose is defined as a stack of "bone" rasterizations (one bone per channel; here we show it as a skeleton image). The input is processed by the fully-convolutional network (generator) to produce the body part assignment map stack and the body part coordinate map stack. These stacks are then used to sample the body texture maps at the locations prescribed by the part coordinate stack with the weights prescribed by the part assignment stack to produce the RGB image. In addition, the last body assignment stack map corresponds to the background probability. During learning, the mask and the RGB image are compared with ground-truth and the resulting losses are backpropagated through the sampling operation into the fully-convolutional network and onto the texture, resulting in their updates.

processing [Dosovitskiy et al., 2015]). Generally, mappings between stacks of maps can be implemented using fully-convolutional architectures. Exact architectures and losses for such networks is an active area of research [Chen and Koltun, 2017, Isola et al., 2017b, Johnson et al., 2016, Ulyanov et al., 2016]. Very recent works [Aberman et al., 2018, Chan et al., 2018, Liu et al., 2018, Wang et al., 2018] have used direct translation (with various modifications) to synthesize the view of a person for a fixed camera. We use the video-to-video variant of this approach [Wang et al., 2018] as a baseline for our method.

6.3.4 Textured neural avatar

The direct translation approach relies on the generalization ability of ConvNets and incorporates very little domain-specific knowledge into the system. As an alternative, we suggest the textured avatar approach, that explicitly estimates the textures of body parts, thus ensuring the similarity of the body surface appearance under varying pose and cameras.

Following the DensePose approach [Güler et al., 2018], we subdivide the body into n=24 parts, where each part has a 2D parameterization. Each body part also has the texture map T^k, which is a color image of a fixed pre-defined size (256×256 in our implementation). The training process for the textured neural avatar estimates personalized part parameterizations and textures.

Again, following the DensePose approach, we assume that each pixel in an image of a person is (soft)-assigned to one of n parts or to the background and with a specific location on the texture of that part (body part coordinates). Unlike DensePose, where part assignments and body part coordinates are induced from the image, our approach at test time aims to predict them based solely on the pose B_i.

The introduction of the body surface parameterization outlined above changes the translation problem. For a given pose defined by B_i, the translation network now has to predict the stack P_i of body part assignments and the stack C_i of body part coordinates, where P_i contains $n{+}1$ maps of non-negative numbers that sum to identity (i.e. $\sum_{k=0}^{n} P_i^k[x,y] = 1$ for any position (x,y)), and C_i contains $2n$ maps of real numbers between 0 and w, where w is the spatial size (width and height) of the texture maps T^k.

The map channel P_i^k for $k = 0,\ldots,n{-}1$ is then interpreted as the probability of the pixel to belong to the k-th body part, and the map channel P_i^n corresponds to the probability of the background. The coordinate maps C_i^{2k} and C_i^{2k+1} correspond to the pixel coordinates on the k-th body part. Specifically, once the part assignments P_i and body part coordinates C_i are predicted, the image I_i at each pixel (x,y) is reconstructed as a weighted combination of texture elements, where the weights and texture coordinates are prescribed by the part assignment maps and the coordinate maps correspondingly:

$$s(P_i, C_i, T)[x,y] = \sum_{k=0}^{n-1} P_i^k[x,y] \cdot T^k\left[C_i^{2k}[x,y], C_i^{2k+1}[x,y]\right], \tag{6.1}$$

where $s(\cdot,\cdot,\cdot)$ is the sampling function (layer) that outputs the RGB map stack given the three input arguments. In (6.1), the texture maps T^k are sampled at non-integer locations $(C_i^{2k}[x,y], C_i^{2k+1}[x,y])$ in a piecewise-differentiable manner using bilinear interpolation [Jaderberg et al., 2015].

When training the neural textured avatar, we learn a convolutional network g_ϕ with learnable parameters ϕ to translate the input map stacks B_i into the body part assignments and the body part coordinates. As g_ϕ has two branches ("heads"), we denote with g_ϕ^P the branch that produces the body part assignments stack, and with g_ϕ^C the branch that produces the body part coordinates. To learn the parameters of the textured neural avatar, we optimize the loss between the generated image and the ground truth image $\bar{I}_i$:

$$\mathcal{L}_{\text{image}}(\phi, T) = d_{\text{Image}}\left(\bar{I}_i,\ s\left(g_\phi^P(B_i), g_\phi^C(B_i), T\right)\right) \tag{6.2}$$

where $d_{\text{Image}}(\cdot,\cdot)$ is a loss used to compare two images. In our current implementation we use a simple perceptual loss [Gatys et al., 2015d, Johnson et al., 2016, Ulyanov et al., 2016], which computes the maps of activations within pretrained fixed VGG network [Simonyan and Zisserman, 2014] for both images and evaluates the L1-norm between the resulting maps (`Conv1,6,11,20,29` of VGG19 were used). More advanced adversarial losses [Goodfellow et al., 2014] popular in image translation [Dosovitskiy and Brox, 2016b, Isola et al., 2017b] can also be used here.

During the stochastic optimization, the gradient of the loss (6.2) is backpropagated through (6.1) both into the translation network g_ϕ and onto the texture maps T^k, so that minimizing this loss updates not only the network parameters but also the textures themselves. As an addition, the learning also optimizes the mask loss that measures the discrepancy between the ground truth background mask $1-\bar{M}_i$ and the background mask prediction:

$$\mathcal{L}_{\text{mask}}(\phi, T) = d_{\text{BCE}}\left(\bar{1} - M_i,\ g_\phi^P(B_i)^n\right) \tag{6.3}$$

where d_{BCE} is the binary cross-entropy loss, and $g_\phi^P(B_i)^n$ corresponds to the n-th (i.e. background) channel of the predicted part assignment map stack. After backpropagation of the weighted combination of (6.2) and (6.3), the network parameters ϕ and the textures maps T^k are updated. As the training progresses, the texture maps change (Figure 6.2), and so does the body part coordinate predictions, so that the learning is free to choose the appropriate parameterization of body part surfaces.

6.3.5 Initialization of textured neural avatar

The success of our network depends on the initialization strategy. When training from multiple video sequences, we use the DensePose system [Güler et al., 2018] to initialize the textured neural avatar. Specifically, we run DensePose on the training data and pretrain g_ϕ as a translation network between the pose stacks B_i and the DensePose outputs.

An alternative way that is particularly attractive when training data is scarce is to initialize the avatar is through transfer learning. In this case, we simply take g_ϕ from another avatar trained on abundant data. The explicit decoupling of geometry from appearance in our method facilitates transfer learning, as the geometrical mapping provided by the network g_ϕ usually does not need to change much between two people, especially if the body types are not too dissimilar.

Once the mapping g_ϕ has been initialized, the texture maps T^k are initialized as follows. Each pixel in the training image is assigned to a single body part (according to the

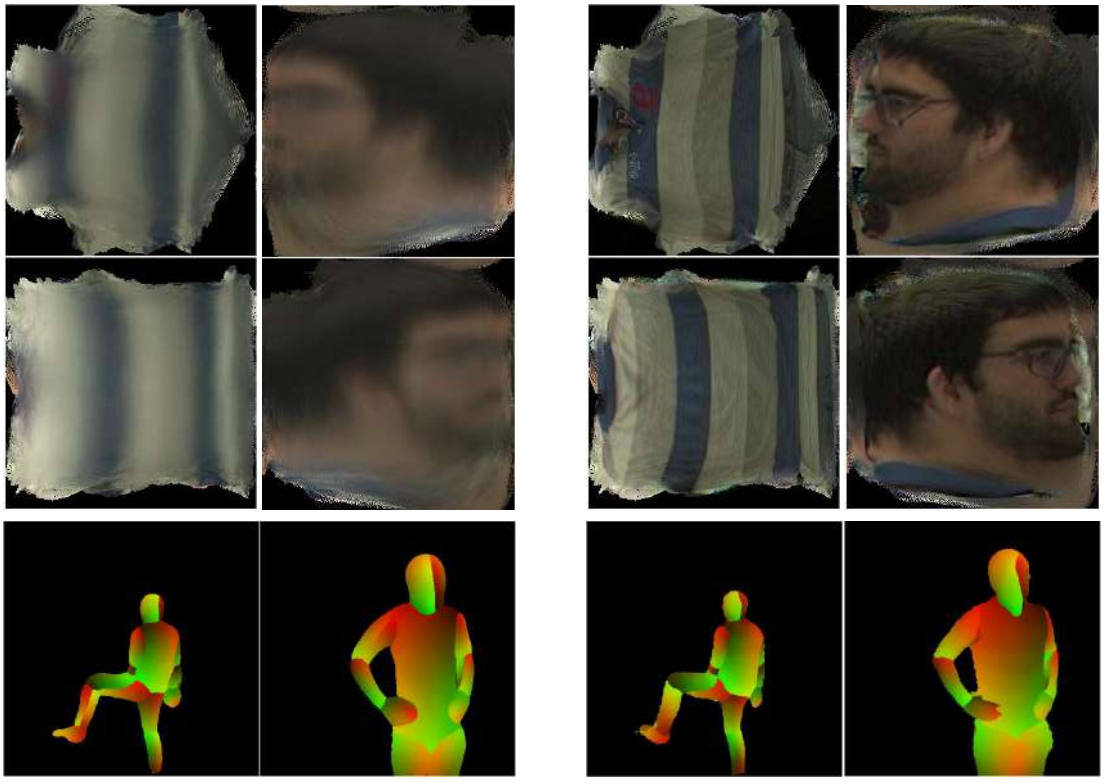


FIGURE 6.3: The impact of the learning on the texture (top, shown for the same subset of maps T^k) and on the convolutional network g_ϕ^C predictions (bottom, shown for the same pair of input poses). Left part shows the starting state (after initialization), while the right part shows the final state, which is considerably different from the start.

	(a) User study		(b) SSIM score			(c) Frechet distance		
	Ours-v-V2V	Ours-v-Direct	V2V	Direct	Ours	V2V	Direct	Ours
CMU1-16	0.56	0.75	0.908	0.899	0.919	6.7	7.3	8.8
CMU2-16	0.54	0.74	0.916	0.907	0.922	7.0	8.8	10.7
CMU1-6	0.50	0.92	0.905	0.896	0.914	7.7	10.7	8.9
CMU2-6	0.53	0.71	0.918	0.907	0.920	7.0	9.7	10.4

TABLE 6.1: Quantitative comparison of the three models operating on different datasets (see text for discussion).

prediction of the pretrained g_ϕ^P) and to a particular texture pixel on the texture of the corresponding part (according to the prediction of the pretrained g_ϕ^C). Then, the value of each texture pixel is initialized to the mean of all image pixels assigned to it (the texture pixels assigned zero pixels are initialized to black). The initialized texture T and g_ϕ usually produce images that are only coarsely reminding the person, and they change significantly during the end-to-end learning (Figure 6.3).

FIGURE 6.4: Renderings produced by multiple textured neural avatars (for all people in our study). All renderings are produced from the new viewpoints unseen during training.

6.4 Experiments

Below, we discuss the details of the experimental validation, provide comparison with baseline approaches, and show qualitative results. The project webpage[1] also contains videos of the learned avatars.

6.4.1 Architecture

We input 3D pose via bone rasterizations, where each bone, hand and face are drawn in separate channels. We then use standard image translation architecture [Johnson et al., 2016] to perform a mapping from these bones' rasterizations to texture assignments and coordinates. This architecture consists of downsampling layers, stack of residual blocks, operating at low dimensional feature representations, and upsampling layers. We then split the network into two roughly equal parts: encoder and decoder, with texture assignments and coordinates having separate decoders. We use 4 downsampling and upsampling layers with initial 32 channels in the convolutions and 256 channels in the residual blocks. The ConvNet g_ϕ has 17 million parameters.

6.4.2 Datasets

We train neural avatars on two types of datasets. First, we consider collections of multi-view videos registered in time and space, where 3D pose estimates can be obtained via triangulation of 2D poses. We use two subsets (corresponding to two persons from the 171026_pose2 scene) from the CMU Panoptic dataset collection [Joo et al., 2017], referring to them as `CMU1` and `CMU2` (both subsets have approximately four minutes / 7,200 frames in each camera view). We consider two regimes: training on 16 cameras (`CMU1-16`

[1] https://saic-violet.github.io/texturedavatar/

and `CMU2-16`) or six cameras (`CMU1-6` and `CMU2-6`). The evaluation is done on the hold-out cameras and hold-out parts of the sequence (no overlap between train and test in terms of the cameras or body motion).

We have also captured our own multi-view sequences of three subjects using a rig of seven cameras, spanning approximately 30°. In one scenario, the training sets included six out of seven cameras, where the duration of each video was approximately six minutes (11,000 frames). We show qualitative results for the hold-out camera as well as from new viewpoints. In the other scenario described below, training was done based on a video from a single camera.

Finally, we evaluate on two short monocular sequences from [Alldieck et al., 2018a] and a Youtube video in Figure 6.7.

6.4.3 Pre-processing

Our system expects 3D human pose as input. For non-CMU datasets, we used the OpenPose-compatible [Cao et al., 2017, Simon et al., 2017] 3D pose formats, represented by 25 body joints, 21 joints for each hand and 70 facial landmarks. For the CMU Panoptic datasets, we use the available 3D pose annotation as input (which has 19 rather than 25 body joints). To get a 3D pose for non-CMU sequences we first apply the OpenPose 2D pose estimation engine to five consecutive frames of the monocular RGB image sequence. Then we concatenate and lift the estimated 2D poses to infer the 3D pose of the last frame by using a multi-layer perceptron model. The perceptron is trained on the CMU 3D pose annotations (augmented with position of the feet joints by triangulating the output of OpenPose) in orthogonal projection.

For foreground segmentation we use DeepLabv3+ with Xception-65 backbone [Chen et al., 2018] initially trained on PASCAL VOC 2012 [Everingham et al., 2015] and fine-tuned on HumanParsing dataset [Liang et al., 2015a,b] to predict initial human body segmentation masks. We additionally employ GrabCut [Rother et al., 2004] with background/foreground model initialized by the masks to refine object boundaries on the high-resolution images. Pixels covered by the skeleton rasterization were always added to the foreground mask.

6.4.4 Baselines

We consider two other systems, against which ours is compared. First, we take the video-to-video ($V2V$) system [Wang et al., 2018], using the authors' code with minimal modifications that lead to improved performance. We provide it with the same input as

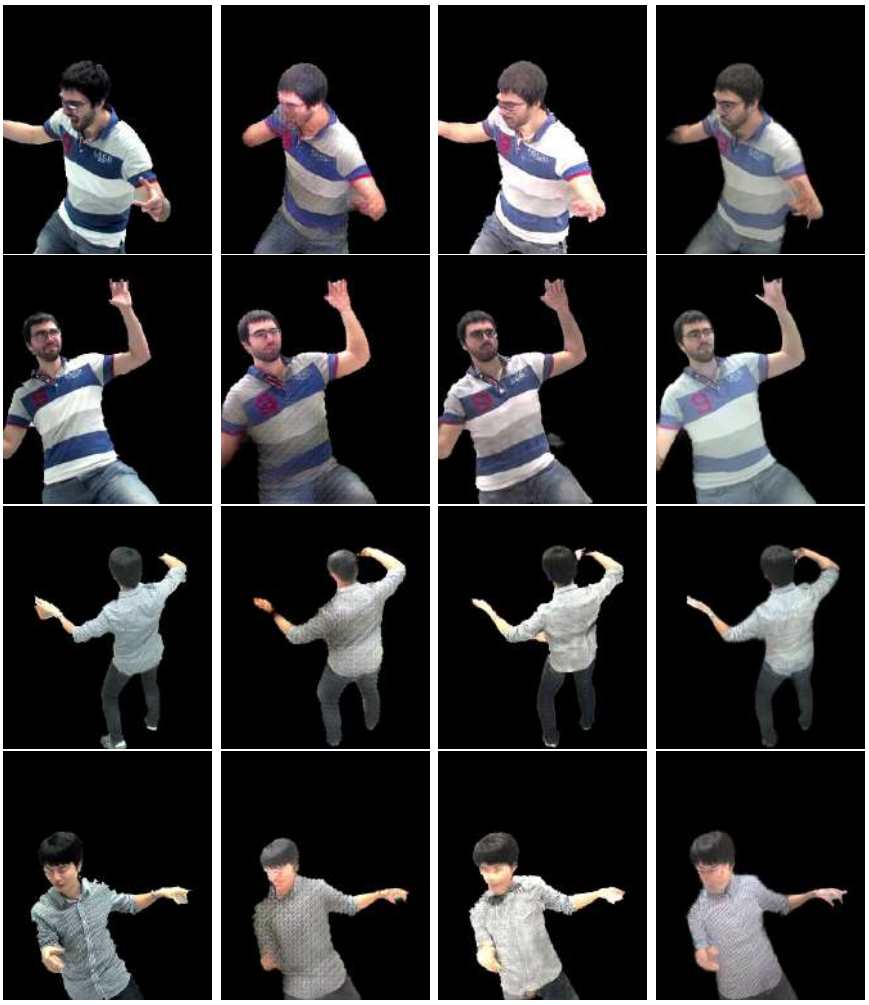


FIGURE 6.5: Comparison of the rendering quality for the Direct, V2V and proposed methods on the `CMU1-6` and `CMU2-6` sequences. Images from six arbitrarily chosen cameras were used for training. We generate the views onto the hold-out cameras which were not used during training. The pose and camera in the lower right corner are in particular difficult for all the systems.

ours, and we use images with blacked-out background (according to our segmentation) as desired output. On the `CMU1-6` task, we have also evaluated a model with DensePose results computed on the target frame given as input (alongside keypoints). Despite much stronger (oracle-type) conditioning, the performance of this model in terms of considered metrics has not improved in comparison with V2V that uses only body joints as input.

The video-to-video system employs several adversarial losses and an architecture different from ours. Therefore we consider a more direct ablation (*Direct*), which has the same network architecture that predicts RGB color and mask directly, rather than via body part assignments/coordinates. The Direct system is trained using the same losses and in the same protocol as ours.

6.4.5 Multi-video comparison

We compare the three systems (*ours*, *V2V*, *Direct*) in `CMU1-16`, `CMU2-16`, `CMU1-6`, `CMU2-6`. Using the hold-out sequences/motions, we then evaluated two popular metrics, namely structured self-similarity (SSIM) and Frechet Inception Distance (FID) between the results of each system and the hold-out frames (with background removed using our segmentation algorithm). Our method outperforms the other two in terms of SSIM and underperforms V2V in terms of FID. Representative examples are shown in Figure 6.5.

We have also performed user study using a crowd-sourcing website, where the users were shown the results of ours and one of the other two systems on either side of the ground truth image and were asked to pick a better match to the middle image. In the side-by-side comparison, the results of our method were always preferred by the majority of crowd-sourcing users. We note that our method suffers from a disadvantage both in the quantitative metrics and in the user comparison, since it averages out lighting from different viewpoints. The more detailed quantitative comparison is presented in Table 6.1.

We show more qualitative examples of our method for a variety of models in Figure 6.4 and some qualitative comparisons with baselines in Figure 6.6.

6.4.6 Single video comparisons

We also evaluate our system in a single video case. We consider the scenario, where we train the model and transfer it to a new person by fitting it to a single video. We use single-camera videos from one of the cameras in our rig. We then evaluate the model (and V2V baseline) on a hold-out set of poses projected onto the camera from the other side of the rig (around 30° away). We thus demonstrate that new models can be obtained using a single monocular video. For our models, we consider transferring from `CMU1-16`.

We thus pretrain V2V and our system on `CMU1-16` and use the obtained weights of g_ϕ as initialization for fine-tuning to the single video in our dataset. The texture maps are initialized from scratch as described above. Evaluating on hold-out camera and motion highlighted strong advantage of our method. In the user study on two subjects, the result of our method has been preferred to V2V in 55% and 65% of the cases. We further compare our method and the system of [Alldieck et al., 2018a] on the sequences from [Alldieck et al., 2018a]. The qualitative comparison is shown in Figure 6.7. In addition, we generate an avatar from a YouTube video. In this set of experiments, the avatars were obtained by fine-tuning from the same avatar (shown in Figure 6.6–left). Except for the considerable artefacts on hand parts, our system has generated avatars

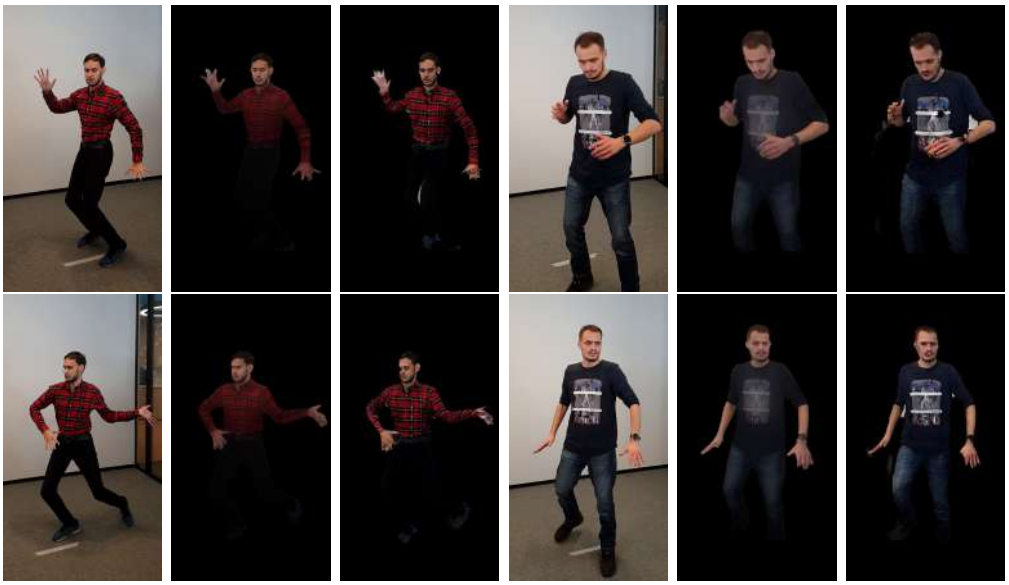


FIGURE 6.6: Results comparison for our multi-view sequences using a hold-out camera. Textured Neural Avatars and the images produced by the video-to-video (V2V) system correspond to the same viewpoint. Both systems use a video from a single viewpoint for training. *Electronic zoom-in recommended.*

that can generalize to new pose despite very short video input (300 frames in the case of [Alldieck et al., 2018a]).

6.5 Summary and Discussion

We have presented textured neural avatar approach to model the appearance of humans for new camera views and new body poses. Our system takes the middle path between the recent generation of methods that use ConvNets to map the pose to the image directly, and the traditional approach that uses geometric modeling of the surface and superimpose the personalized texture maps. This is achieved by learning a ConvNet that predicts texture coordinates of pixels in the new view jointly with the texture within the end-to-end learning process. We demonstrate that retaining an explicit shape and texture separation helps to achieve better generalization than direct mapping approaches.

Our method suffers from certain limitations. The generalization ability is still limited, as it does not generalize well when a person is rendered at a scale that is considerably different from the training set (which can be partially addressed by rescaling prior to rendering followed by cropping/padding postprocessing). Furthermore, textured avatars exhibit strong artefacts in the presence of pose estimation errors on hands and faces. Finally, our method assumes constancy of the surface color and ignores lighting effects. This can

FIGURE 6.7: Results on external monocular sequences. Rows 1-2: avatars for sequences from [Alldieck et al., 2018a] in an unseen pose (left – ours, right – [Alldieck et al., 2018a]). Row 3 – the textured avatar computed from a popular YouTube video ('PUMPED UP KICKS DUBSTEP'). In general, our system is capable of learning avatars from monocular videos.

be potentially addressed by making our textures view- and lighting-dependent [Debevec et al., 1998, Lombardi et al., 2018].

Chapter 7

Deep Image Prior

Abstract

Deep convolutional networks have become a popular tool for image generation and restoration. Generally, their excellent performance is imputed to their ability to learn realistic image priors from a large number of example images. In this work, we show that, on the contrary, the *structure* of a generator network is sufficient to capture a great deal of low-level image statistics *prior to any learning*. In order to do so, we show that a randomly-initialized neural network can be used as a handcrafted prior with excellent results in standard inverse problems such as denoising, super-resolution, and inpainting. Furthermore, the same prior can be used to invert deep neural representations to diagnose them, and to restore images based on flash-no flash input pairs.

Apart from its diverse applications, our approach highlights the inductive bias captured by standard generator network architectures. It also bridges the gap between two very popular families of image restoration methods: learning-based methods using deep convolutional networks and learning-free methods based on handcrafted image priors such as self-similarity.

This work was published as: Dmitry Ulyanov, Andrea Vedaldi and Victor Lempitsky. *Deep Image Prior*. Computer Vision and Pattern Recognition (CVPR), 2018."

7.1 Introduction

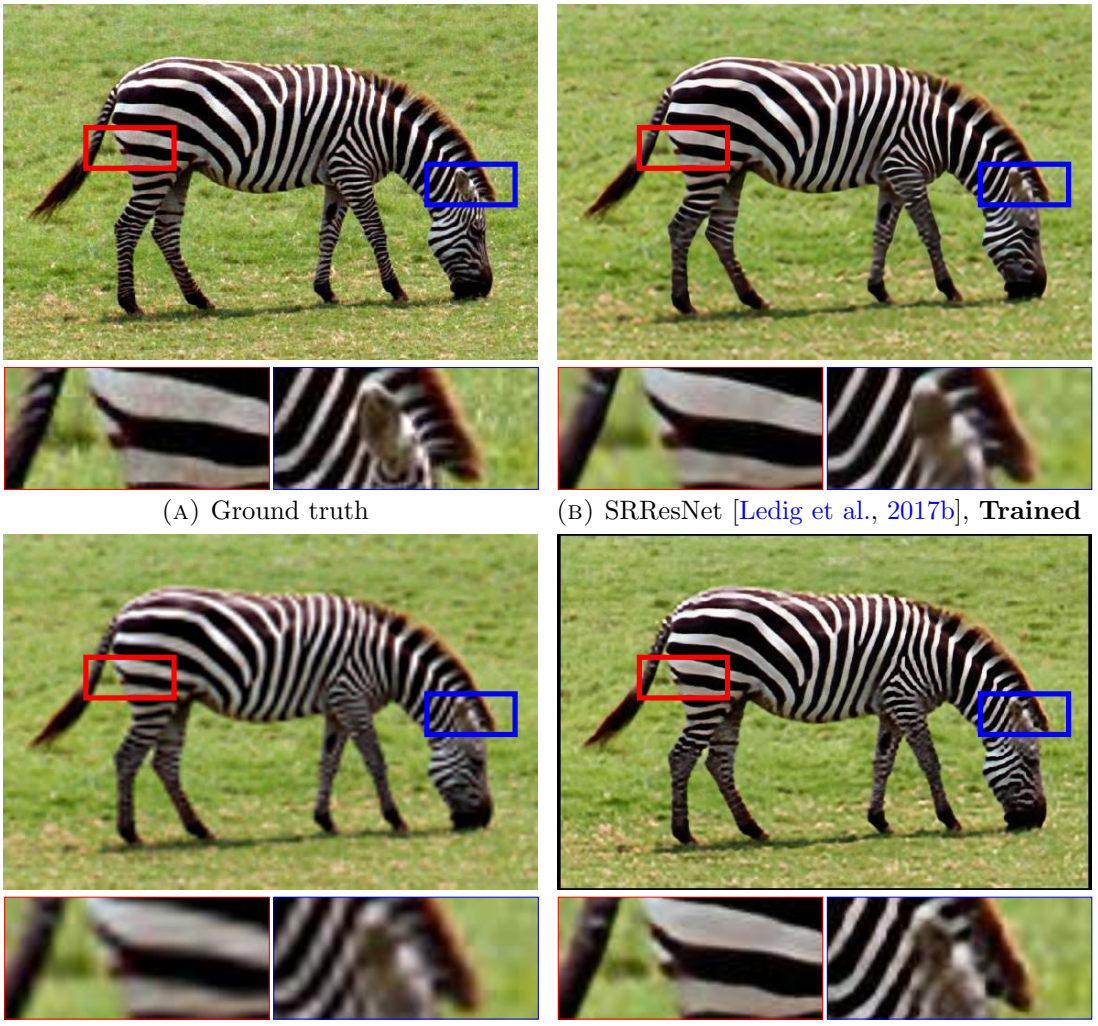
Deep convolutional neural networks (ConvNets) currently set the state-of-the-art in inverse image reconstruction problems such as denoising [Burger et al., 2012, Lefkimmiatis, 2016] or single-image super-resolution [Ledig et al., 2017b, Tai et al., 2017, Lai et al., 2017]. ConvNets have also been used with great success in more "exotic" problems such as reconstructing an image from its activations within certain deep networks or from its HOG descriptor [Dosovitskiy and Brox, 2016a]. More generally, ConvNets with similar architectures are nowadays used to generate images using such approaches as generative adversarial networks [Goodfellow et al., 2014], variational autoencoders [Kingma and Welling, 2014], and direct pixelwise error minimization [Dosovitskiy et al., 2015, Bojanowski et al., 2017].

State-of-the-art ConvNets for image restoration and generation are almost invariably trained on large datasets of images. One may thus assume that their excellent performance is due to their ability to learn realistic image priors from data. However, learning alone is insufficient to explain the good performance of deep networks. For instance, the authors of [Zhang et al., 2017] recently showed that the same image classification network that generalizes well when trained on genuine data can *also* overfit when presented with random labels. Thus, generalization requires the *structure* of the network to "resonate" with the structure of the data. However, the nature of this interaction remains unclear, particularly in the context of image generation.

In this work, we show that, contrary to the belief that learning is necessary for building good image priors, a great deal of image statistics are captured by the *structure* of a convolutional image generator independent of learning. This is particularly true for the statistics required to solve various image restoration problems, where the image prior is required to integrate information lost in the degradation processes.

To show this, we apply *untrained* ConvNets to the solution of several such problems. Instead of following the common paradigm of training a ConvNet on a large dataset of example images, we fit a generator network to a single degraded image. In this scheme, the network weights serve as a parametrization of the restored image. The weights are randomly initialized and fitted to maximize their likelihood given a specific degraded image and a task-dependent observation model.

Stated in a different way, we cast reconstruction as a *conditional* image generation problem and show that the only information required to solve it is contained in the single degraded input image *and* the handcrafted structure of the network used for reconstruction.



(A) Ground truth (B) SRResNet [Ledig et al., 2017b], **Trained**

(C) Bicubic, **Not trained** (D) Deep prior, **Not trained**

FIGURE 7.1: **Super-resolution using the deep image prior.** Our method uses a randomly-initialized ConvNet to upsample an image, using its structure as an image prior; similar to bicubic upsampling, this method does not require learning, but produces much cleaner results with sharper edges. In fact, our results are quite close to state-of-the-art super-resolution methods that use ConvNets learned from large datasets. The deep image prior works well for all inverse problems we could test.

We show that this very simple formulation is very competitive for standard image processing problems such as denoising, inpainting and super-resolution. This is particularly remarkable because *no aspect of the network is learned from data*; instead, the weights of the network are always randomly initialized, so that the only prior information is in the structure of the network itself. To the best of our knowledge, this is the first study that directly investigates the prior captured by deep convolutional generative networks independently of learning the network parameters from images.

In addition to standard image restoration tasks, we show an application of our technique to understanding the information contained within the activations of deep neural networks. For this, we consider the "natural pre-image" technique of [Mahendran and

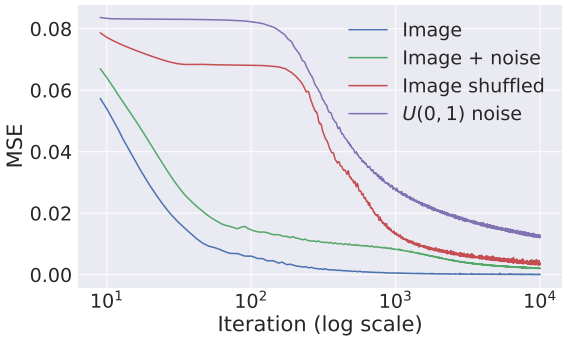


FIGURE 7.2: Learning curves for the reconstruction task using: a natural image, the same plus i.i.d. noise, the same randomly scrambled, and white noise. Naturally-looking images result in much faster convergence, whereas noise is rejected.

Vedaldi, 2015], whose goal is to characterize the invariants learned by a deep network by inverting it on the set of natural images. We show that an untrained deep convolutional generator can be used to replace the surrogate natural prior used in [Mahendran and Vedaldi, 2015] (the TV norm) with dramatically improved results. Since the new regularizer, like the TV norm, is not learned from data but is entirely handcrafted, the resulting visualizations avoid potential biases arising form the use of powerful learned regularizers [Dosovitskiy and Brox, 2016a].

7.2 Method

Deep networks are applied to image generation by learning generator/decoder networks $x = f_\theta(z)$ that map a random code vector z to an image x. This approach can be used to sample realistic images from a random distribution [Goodfellow et al., 2014]. Here we focus on the case where the distribution is *conditioned* on a corrupted observation x_0 to solve inverse problems such as denoising [Burger et al., 2012] and super-resolution [Dong et al., 2014b].

Our aim is to investigate the prior implicitly captured by the choice of a particular generator network structure, *before* any of its parameters are learned. We do so by interpreting the neural network as a *parameterization* $x = f_\theta(z)$ of an image $x \in \mathbb{R}^{3\times H\times W}$. Here $z \in \mathbb{R}^{C'\times H'\times W'}$ is a code tensor/vector and θ are the network parameters. The network itself alternates filtering operations such as convolution, upsampling and non-linear activation. In particular, most of our experiments are performed using a U-Net type "hourglass" architecture with skip-connections, where z and x have the same spatial size. Our default architecture has two million parameters θ.

To demonstrate the power of this parametrization, we consider inverse tasks such as denoising, super-resolution and inpainting. These can be expressed as energy minimization

problems of the type

$$x^* = \min_x E(x; x_0) + R(x), \tag{7.1}$$

where $E(x; x_0)$ is a task-dependent data term, x_0 the noisy/low-resolution/occluded image, and $R(x)$ a regularizer.

The choice of data term $E(x; x_0)$ is dictated by the application and will be discussed later. The choice of regularizer, which usually captures a generic prior on natural images, is more difficult and is the subject of much research. As a simple example, $R(x)$ can be the Total Variation (TV) of the image, which encourages solutions to contain uniform regions. In this work, we *replace* the explicit regularizer $R(x)$ with the implicit prior captured by the neural network, by reparameterizing the inverse problem as follows:

$$\theta^* = \operatorname*{argmin}_\theta E(f_\theta(z); x_0), \qquad x^* = f_{\theta^*}(z)\,. \tag{7.2}$$

Here, the minimizer θ^* is obtained using an optimizer such as gradient descent starting from a *random initialization* of the parameters. Given a (local) minimizer θ^*, the result of the restoration process is obtained as $x^* = f_{\theta^*}(z)$. Note that while it is also possible to optimize over the code z, in our experiments we do not do that. Thus, unless noted otherwise, z is a fixed $3D$ tensor with 32 feature maps and of the same spatial size as x filled with uniform noise. We found that additionally perturbing z randomly at every iteration lead to better results in some experiments.

In terms of (7.1), the prior $R(x)$ defined by (7.2) is an indicator function $R(x) = 0$ for all images that can be produced from z by a deep ConvNet of a certain architecture, and $R(x) = +\infty$ for all other signals. Since no aspect of the network is pre-trained from data, such *deep image prior* is effectively handcrafted, just like the TV norm. We show that this hand-crafted prior works very well for various image restoration tasks.

7.2.1 A parametrization with high noise impedance

One may wonder why a high-capacity network f_θ can be used as a prior at all. In fact, given the very high number of parameters in the network f_θ, one may expect to be able to find parameters θ recovering any possible image x, including random noise, so that the network should not impose any restriction on the generated image. We now show that, while indeed almost any image can be fitted, the choice of network architecture has a major effect on how the solution space is searched by methods such as gradient descent. In particular, we show that the network resists "bad" solutions and descends much more quickly towards naturally-looking images. The result is that minimizing (7.2)

either results in a good-looking local optimum, or, at least, the optimization trajectory passes near one.

In order to study this effect quantitatively, we consider the most basic reconstruction problem: given a target image x_0, we want to find the value of the parameters θ^* that reproduce that image. This can be setup as the optimization of (7.2) using a data term comparing the generated image to x_0:

$$E(x; x_0) = \|x - x_0\|^2 \tag{7.3}$$

Plugging this in eq. (7.2) leads us to the optimization problem

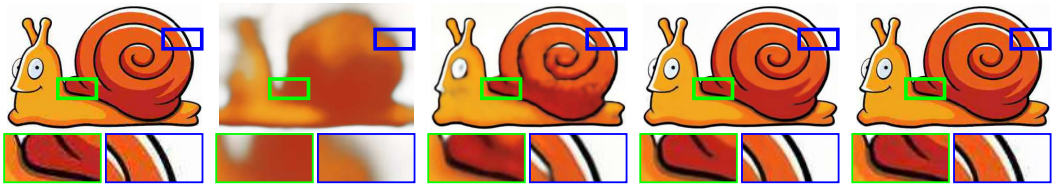
$$\theta^* = \operatorname*{argmin}_{\theta} \|f_\theta(z) - x_0\|^2 \tag{7.4}$$

Figure 7.2 shows the value of the energy $E(x; x_0)$ as a function of the gradient descent iterations for four different choices for the image x_0: 1) a natural image, 2) the same image plus additive noise, 3) the same image after randomly permuting the pixels, and 4) white noise. It is apparent from the figure that optimization is much faster for cases 1) and 2), whereas the parametrization presents significant "inertia" for cases 3) and 4).

Thus, although in the limit the parametrization *can* fit unstructured noise, it does so very reluctantly. In other words, the parametrization offers high impedance to noise and low impedance to signal. Therefore for most applications, we restrict the number of iterations in the optimization process (7.2) to a certain number of iterations. The resulting prior then corresponds to projection onto a reduced set of images that can be produced from z by ConvNets with parameters θ that are not too far from the random initialization θ_0.

7.3 Applications

We now show experimentally how deep image prior performs within a range of diverse image reconstruction problems. In each case, we present a few qualitative examples and also include quantitative evaluation numbers where possible. The project webpage [Ulyanov et al.] contains more qualitative results and also facilitates interactive comparisons.



Corrupted 100 iterations 600 iterations 2400 iterations 50K iterations

FIGURE 7.3: **Blind restoration of a JPEG-compressed image.** (*electronic zoom-in recommended*) Our approach can restore an image with a complex degradation (JPEG compression in this case). As the optimization process progresses, the deep image prior allows to recover most of the signal while getting rid of halos and blockiness (after 2400 iterations) before eventually overfitting to the input (at 50K iterations).

(A) GT (B) Input (C) Ours (D) CBM3D

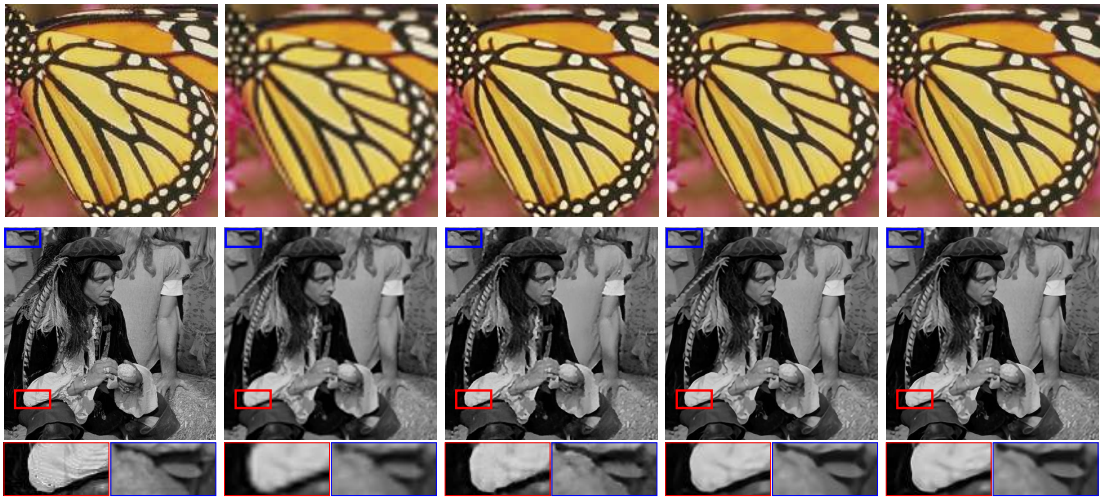
FIGURE 7.4: **Blind image denoising.** The deep image prior is successful at recovering both man-made and natural patterns. For reference, the result of a state-of-the-art non-learned denoising approach [Dabov et al., 2007] is shown.

7.3.1 Denoising and generic reconstruction

As our parametrization presents high impedance to image noise, it can be naturally used to filter out noise from an image. The aim of denoising is to recover a clean image x from a noisy observation x_0. Sometimes the degradation model is known: $x_0 = x + \epsilon$ where ϵ follows a particular distribution. However, more often in *blind denoising* the noise model is unknown.

Here we work under the blindness assumption, but the method can be easily modified to incorporate information about noise model. We use the same exact formulation as eqs. (7.3) and (7.4) and, given a noisy image x_0, recover a clean image $x^* = f_{\theta^*}(z)$ after substituting the minimizer θ^* of eq. (7.4).

Our approach does not require a model for the image degradation process that it needs to revert. This allows it to be applied in a "plug-and-play" fashion to image restoration tasks, where the degradation process is complex and/or unknown and where obtaining realistic data for supervised training is difficult. We demonstrate this capability by several qualitative examples in fig. 7.4, where our approach uses the quadratic energy (7.3) leading to formulation (7.4) to restore images degraded by complex and unknown



(A) Original image (B) Bicubic, **Not trained** (C) Ours, **Not trained** (D) LapSRN, **Trained** (E) SRResNet, **Trained**

FIGURE 7.5: **4x image super-resolution.** Similarly to e.g. bicubic upsampling, our method never has access to any data other than a single low-resolution image, and yet it produces much cleaner results with sharp edges close to state-of-the-art super-resolution methods (LapSRN [Lai et al., 2017], SRResNet [Ledig et al., 2017b]) which utilize networks trained from large datasets.

compression artifacts. Figure 7.3 (top row) also demonstrates the applicability of the method beyond natural images (a cartoon in this case).

We evaluate our denoising approach on the standard dataset[1], consisting of 9 colored images with noise strength of $\sigma = 25$. We achieve a PSNR of 29.22 after 1800 optimization steps. The score is improved up to 30.43 if we additionally average the restored images obtained in the last iterations (using exponential sliding window). If averaged over two optimization runs our method further improves up to 31.00 PSNR. For reference, the scores for the two popular approaches CMB3D [Dabov et al., 2007] and Non-local means [Buades et al., 2005] that do not require pretraining are 31.42 and 30.26 respectively.

7.3.2 Super-resolution

The goal of super-resolution is to take a low resolution (LR) image $x_0 \in \mathbb{R}^{3\times H\times W}$ and upsampling factor t, and generate a corresponding high resolution (HR) version $x \in \mathbb{R}^{3\times tH\times tW}$. To solve this inverse problem, the data term in (7.2) is set to:

$$E(x; x_0) = \|d(x) - x_0\|^2 , \tag{7.5}$$

[1] http://www.cs.tut.fi/~foi/GCF-BM3D/index.html#ref_results

	Barbara	Boat	House	Lena	Peppers	C.man	Couple	Finger	Hill	Man	Montage
Papyan et al.	28.14	31.44	34.58	35.04	31.11	27.90	31.18	31.34	32.35	31.92	28.05
Ours	**32.22**	**33.06**	**39.16**	**36.16**	**33.05**	**29.8**	**32.52**	**32.84**	**32.77**	**32.20**	**34.54**

TABLE 7.1: Comparison between our method and the algorithm in [Papyan et al., 2017b]. See fig. 7.7 bottom row for visual comparison.

where $d(\cdot) : \mathbb{R}^{3\times tH\times tW} \to \mathbb{R}^{3\times H\times W}$ is a *downsampling operator* that resizes an image by a factor t. Hence, the problem is to find the HR image x that, when downsampled, is the same as the LR image x_0. Super-resolution is an ill-posed problem because there are infinitely many HR images x that reduce to the same LR image x_0 (i.e. the operator d is far from surjective). Regularization is required in order to select, among the infinite minimizers of (7.5), the most plausible ones.

Following eq. (7.2), we regularize the problem by considering the reparametrization $x = f_\theta(z)$ and optimizing the resulting energy w.r.t. θ. Optimization still uses gradient descent, exploiting the fact that both the neural network and the most common downsampling operators, such as Lanczos, are differentiable.

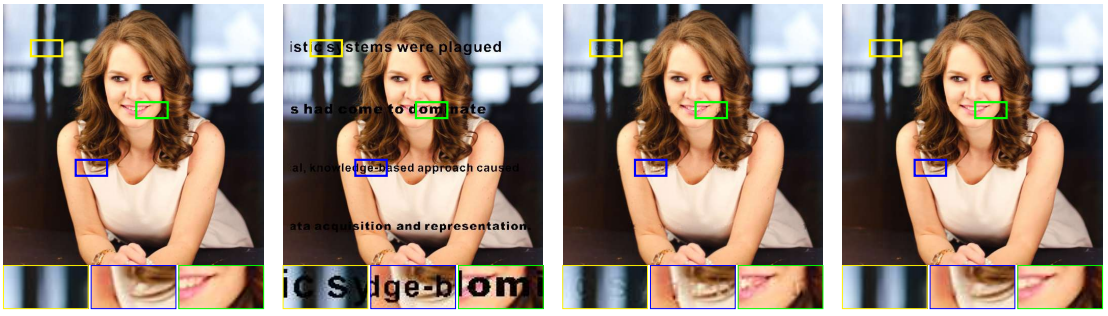
We evaluate super-resolution ability of our approach using Set5 [Bevilacqua et al., 2012] and Set14 [Zeyde et al., 2010] datasets. We use a scaling factor of 4 to compare to other works. We fix the number of optimization steps to be 2000 for every image.

Qualitative comparison with bicubic upsampling and state-of-the art learning-based methods SRResNet [Ledig et al., 2017b], LapSRN [Tai et al., 2017] is presented in fig. 7.5. Our method can be fairly compared to bicubic, as both methods never use other data than a given low-resolution image. Visually, we approach the quality of learning-based methods that use the MSE loss. GAN-based [Goodfellow et al., 2014] methods SRGAN [Ledig et al., 2017b] and EnhanceNet [Sajjadi et al., 2017b] (not shown in the comparison) intelligently hallucinate fine details of the image, which is impossible with our method that uses absolutely no information about the world of HR images.

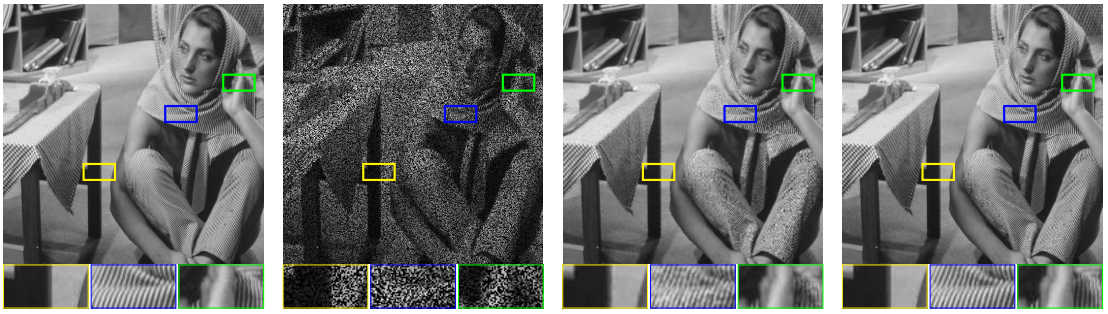
We compute PSNRs using center crops of the generated images. Our method achieves 29.90 and 27.00 PSNR on Set5 and Set14 datasets respectively. Bicubic upsampling gets a lower score of 28.43 and 26.05, while SRResNet has PSNR of 32.10 and 28.53. While our method is still outperformed by learning-based approaches, it does considerably better than bicubic upsampling. Visually, it seems to close most of the gap between bicubic and state-of-the-art trained ConvNets (c.f. fig. 7.1,fig. 7.5).

(A) Corrupted image (B) Global-Local GAN (C) Ours, LR = 0.01 (D) Ours, LR = 10^{-4}

FIGURE 7.6: **Region inpainting.** In many cases, deep image prior is sufficient to successfully inpaint large regions. Despite using no learning, the results may be comparable to [Iizuka et al., 2017b] which does. The choice of hyper-parameters is important (for example (d) demonstrates sensitivity to the learning rate), but a good setting works well for most images we tried.



(A) Original image (B) Corrupted image (C) Shepard networks (D) Deep Image Prior



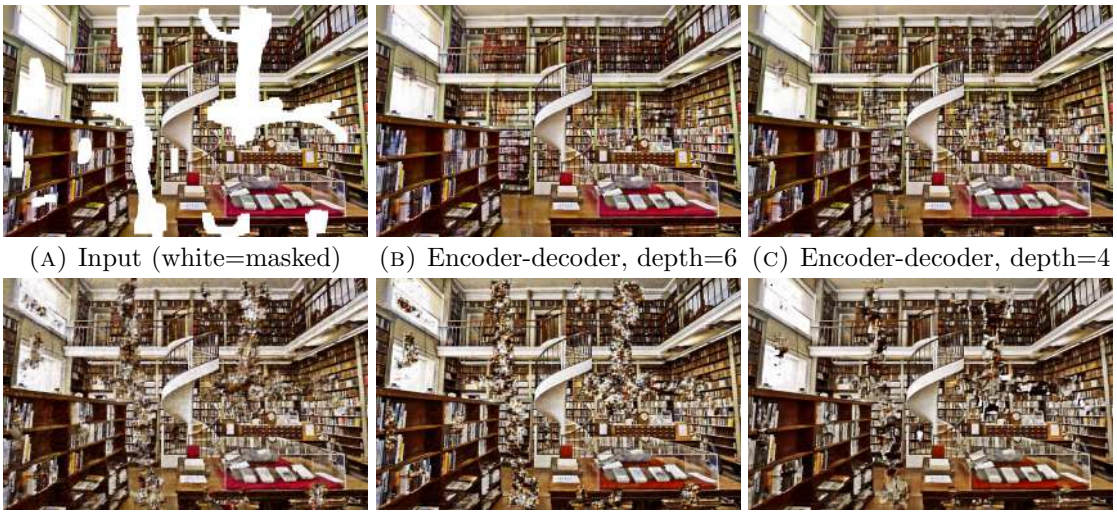
(E) Original image (F) Corrupted image (G) Papyan et al. (H) Deep Imgage Prior

FIGURE 7.7: **Comparison with two recent inpainting approaches**. Top – comparison with Shepard networks [Ren et al., 2015] on text inpainting example. Bottom – comparison with convolutional sparse coding [Papyan et al., 2017b] on inpainting 50% of missing pixels. In both cases, our approach performs better on the images used in the respective papers.

7.3.3 Inpainting

In image inpainting, one is given an image x_0 with missing pixels in correspondence of a binary mask $m \in \{0,1\}^{H \times W}$; the goal is to reconstruct the missing data. The corresponding data term is given by

$$E(x; x_0) = \|(x - x_0) \odot m\|^2 , \tag{7.6}$$



(A) Input (white=masked) (B) Encoder-decoder, depth=6 (C) Encoder-decoder, depth=4

(D) Encoder-decoder, depth=2 (E) ResNet, depth=8 (F) U-net, depth=5

FIGURE 7.8: **Inpainting using different depths and architectures.** The figure shows that much better inpainting results can be obtained by using deeper random networks. However, adding skip connections to ResNet in U-Net is highly detrimental.

where $\odot$ is Hadamard's product. The necessity of a data prior is obvious as this energy is independent of the values of the missing pixels, which would therefore never change after initialization if the objective was optimized directly over pixel values x. As before, the prior is introduced by optimizing the data term w.r.t. the reparametrization (7.2).

In the first example (fig. 7.7, top row) inpainting is used to remove text overlaid on an image. Our approach is compared to the method of [Ren et al., 2015] specifically designed for inpainting. Our approach leads to an almost perfect results with virtually no artifacts, while for [Ren et al., 2015] the text mask remains visible in some regions.

Next, fig. 7.7 (bottom) considers inpainting with masks randomly sampled according to a binary Bernoulli distribution. First, a mask is sampled to drop 50% of pixels at random. We compare our approach to a method of [Papyan et al., 2017b] based on convolutional sparse coding. To obtain results for [Papyan et al., 2017b] we first decompose the corrupted image x_0 into low and high frequency components similarly to [Gu et al., 2015] and run their method on the high frequency part. For a fair comparison we use the version of their method, where a dictionary is built using the input image (shown to perform better in [Papyan et al., 2017b]). The quantitative comparison on the standard data set [Heide et al., 2015] for our method is given in table 7.1, showing a strong quantitative advantage of the proposed approach compared to convolutional sparse coding. In fig. 7.7 (bottom) we present a representative qualitative visual comparison with [Papyan et al., 2017b].

We also apply our method to inpainting of large holes. Being non-trainable, our method is not expected to work correctly for "highly-semantical" large-hole inpainting (e.g. face

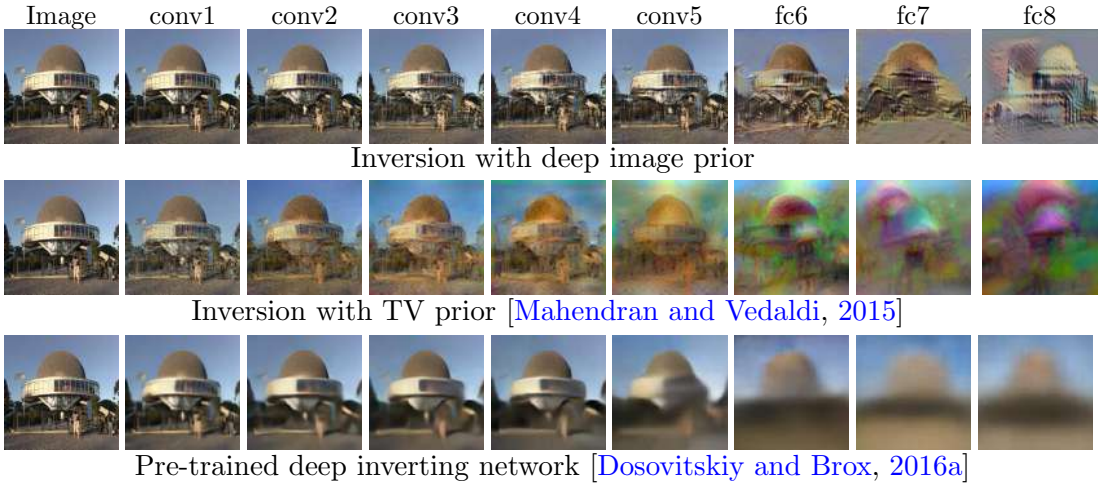


FIGURE 7.9: **AlexNet inversion.** Given the image on the left, we show the natural pre-image obtained by inverting different layers of AlexNet (trained for classification on ImageNet ISLVRC) using three different regularizers: the Deep Image prior, the TV norm prior of [Mahendran and Vedaldi, 2015], and the network trained to invert representations on a hold-out set [Dosovitskiy and Brox, 2016a]. The reconstructions obtained with the deep image prior are in many ways at least as natural as [Dosovitskiy and Brox, 2016a], yet they are not biased by the learning process.

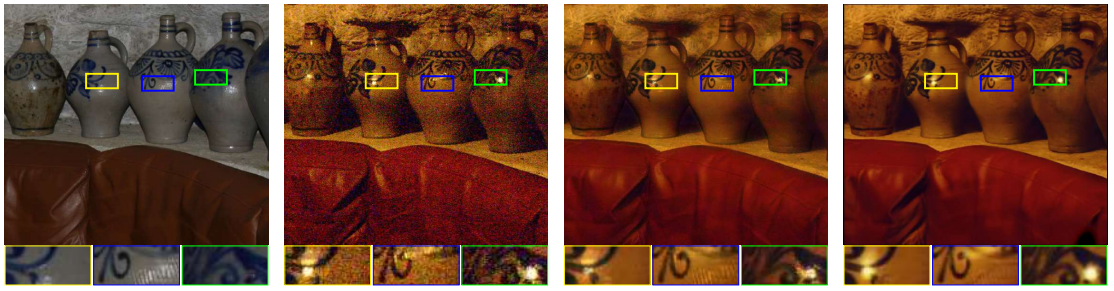
inpainting). Yet, it works surprisingly well for other situations. We compare to a learning-based method of [Iizuka et al., 2017b] in fig. 7.6. The deep image prior utilizes context of the image and interpolates the unknown region with textures from the known part. Such behaviour highlights the relation between the deep image prior and traditional self-similarity priors.

In fig. 7.8, we compare deep priors corresponding to several architectures. Our findings here (and in other similar comparisons) seem to suggest that having deeper architecture is beneficial, and that having skip-connections that work so well for recognition tasks (such as semantic segmentation) is highly detrimental.

7.3.4 Natural pre-image

The natural pre-image method of [Mahendran and Vedaldi, 2015] is a *diagnostic* tool to study the invariances of a lossy function, such as a deep network, that operates on natural images. Let Φ be the first several layers of a neural network trained to perform, say, image classification. The pre-image is the set $\Phi^{-1}(\Phi(x_0)) = \{x \in \mathcal{X} : \Phi(x) = \Phi(x_0)\}$ of images that result in the *same representation* $\Phi(x_0)$. Looking at this set reveals which information is lost by the network, and which invariances are gained.

Finding pre-image points can be formulated as minimizing the data term $E(x; x_0) = \|\Phi(x) - \Phi(x_0)\|^2$. However, optimizing this function directly may find "artifacts", i.e.



(A) Flash (B) No flash (C) Joint bilateral (D) Deep image prior

FIGURE 7.10: **Reconstruction based on flash and no-flash image pair.** The deep image prior allows to obtain low-noise reconstruction with the lighting very close to the no-flash image. It is more successful at avoiding "leaks" of the lighting patterns from the flash pair than joint bilateral filtering [Petschnigg et al., 2004] (c.f. blue inset).

non-natural images for which the behavior of the network Φ is in principle unspecified and that can thus drive it arbitrarily. More meaningful visualization can be obtained by restricting the pre-image to a set $\mathcal{X}$ of natural images, called a *natural pre-image* in [Mahendran and Vedaldi, 2015].

In practice, finding points in the natural pre-image can be done by regularizing the data term similarly to the other inverse problems seen above. The authors of [Mahendran and Vedaldi, 2015] prefer to use the TV norm, which is a weak natural image prior, but is relatively unbiased. On the contrary, papers such as [Dosovitskiy and Brox, 2016a] learn to invert a neural network from examples, resulting in better looking reconstructions, which however may be biased towards learning data-driven inversion prior. Here, we propose to use the deep image prior (7.2) instead. As this is handcrafted like the TV-norm, it is not biased towards a particular training set. On the other hand, it results in inversions at least as interpretable as the ones of [Dosovitskiy and Brox, 2016a].

For evaluation, our method is compared to the ones of [Mahendran and Vedaldi, 2016] and [Dosovitskiy and Brox, 2016a]. Figure 7.9 shows the results of inverting representations Φ obtained by considering progressively deeper subsets of AlexNet [Krizhevsky et al., 2012]: `conv1`, `conv2`, ..., `conv5`, `fc6`, `fc7`, and `fc8`. Pre-images are found either by optimizing (7.2) using a structured prior.

As seen in fig. 7.9, our method results in improved image clarity compared to the simple TV-norm. The difference is particularly large for deeper layers such as `fc6` and `fc7`, where the TV norm still produces noisy images, whereas the structured regularizer produces images that are often interpretable. Our approach also produces more informative inversions than a learned prior of [Dosovitskiy and Brox, 2016a], which have a tendency to regress to the mean.

7.3.5 Flash-no flash reconstruction

While in this work we focus on single image restoration, the proposed approach can be extended to the tasks of the restoration of multiple images, e.g. for the task of video restoration. We therefore conclude the set of application examples with a qualitative example demonstrating how the method can be applied to perform restoration based on pairs of images. In particular, we consider flash-no flash image pair-based restoration [Petschnigg et al., 2004], where the goal is to obtain an image of a scene with the lighting similar to a no-flash image, while using the flash image as a guide to reduce the noise level.

In general, extending the method to more than one image is likely to involve some coordinated optimization over the input codes z that for single-image tasks in our approach was most often kept fixed and random. In the case of flash-no-flash restoration, we found that good restorations were obtained by using the denoising formulation (7.4), while using flash image as an input (in place of the random vector z). The resulting approach can be seen as a non-linear generalization of guided image filtering [He et al., 2013]. The results of the restoration are given in the fig. 7.10.

7.4 Related work

Our method is obviously related to image restoration and synthesis methods based on learnable ConvNets and referenced above. At the same time, it is as much related to an alternative group of restoration methods that avoid training on the hold-out set. This group includes methods based on joint modeling of groups of similar patches inside corrupted image [Buades et al., 2005, Dabov et al., 2007, Glasner et al., 2009], which are particularly useful when the corruption process is complex and highly variable (e.g. spatially-varying blur [Bahat et al., 2017]). Also in this group are methods based on fitting dictionaries to the patches of the corrupted image [Mairal et al., 2010, Zeyde et al., 2010] as well as methods based on convolutional sparse coding [Zeiler et al., 2010], which can also fit statistical models similar to shallow ConvNets to the reconstructed image [Papyan et al., 2017b]. The work [Lefkimmiatis, 2016] investigates the model that combines ConvNet with a self-similarity based denoising and thus also bridges the two groups of methods, but still requires training on a hold-out set.

Overall, the prior imposed by deep ConvNets and investigated in this work seems to be highly related to self-similarity-based and dictionary-based priors. Indeed, as the weights of the convolutional filters are shared across the entire spatial extent of the image this ensures a degree of self-similarity of individual patches that a generative ConvNet

can potentially produce. The connections between ConvNets and convolutional sparse coding run even deeper and are investigated in [Papyan et al., 2017a] in the context of recognition networks, and more recently in [Papyan et al., 2017b], where a single-layer convolutional sparse coding is proposed for reconstruction tasks. The comparison of our approach with [Papyan et al., 2017b] (fig. 7.7 and table 7.1) however suggests that using deep ConvNet architectures popular in modern deep learning-based approaches may lead to more accurate restoration results at least in some circumstances.

Our approach is also related to inverse scale space denoising Scherzer and Groetsch [2001], Burger et al. [2005], Marquina [2009]. In this group of "non-deep" image processing methods, a sequence of solutions (a flow) that gradually progresses from a uniform image to the noisy image, while progressively finer scale details are recovered so that early stopping yields a denoised image. The inverse scale space approaches are however still driven by a simple total variation (TV) prior, which does not model self-similarity of images, and limits the ability to denoise parts of images with textures and gradual transitions. Note that our approach can also use the simple stopping criterion proposed in Burger et al. [2005], when the level of noise is known.

Since the publication of the preliminary version of our approach, it has also been used by other groups in different ways. Thus, Veen et al. [2018] proposes a novel method for compressed sensing recovery using deep image prior. The work Athar et al. [2018] learns a latent variable model, where the latent space is parametrized by a convolutional neural network. The approach Shedligeri et al. [2018] aims to reconstruct an image from an event-based camera and utilizes deep image prior framework to estimate sensor's ego-motion. The method Ilyas et al. [2017] successively applies deep image prior to defend against adversarial attacks. Deep image prior is also used in Boominathan et al. [2018] to perform phase retrieval for Fourier ptychography.

7.5 Discussion

We have investigated the success of recent image generator neural networks, teasing apart the contribution of the prior imposed by the choice of architecture from the contribution of the information transferred from external images through learning. As a byproduct, we have shown that fitting a randomly-initialized ConvNet to corrupted images works as a "Swiss knife" for restoration problems. While practically slow (taking several minutes of GPU computation per image), this approach does not require modeling of the degradation process or pre-training.

Our results go against the common narrative that explain the success of deep learning in image restoration to the ability to learn rather than hand-craft priors; instead, random networks *are better hand-crafted priors*, and learning builds on this basis. This also validates the importance of developing new deep learning architectures.

Chapter 8

Conclusion

The goal of this thesis is to advance the image generation methods using neural networks. The first application we have explored was texture synthesis and style transfer. For texture synthesis, we proposed a neural network, that transforms a sample from Normal distribution into a texture instance with a single forward pass of a neural network. Such network works almost two orders of magnitudes faster than the baseline method.

We then discussed Generative Adversarial Networks and proposed an enhanced model called AGE (Adversarial Generator-Encoder Networks). This model exhibits a comparable image generation quality to GAN, yet it also learns to map images to their latent codes. Compared to analogous methods, it utilizes only two networks, compared to three networks for competitors, making our method preferable following Occam's Razor concept. Next, we proposed a plug-in discriminator network replacement for any GAN-based model. The proposed discriminator compares perceptual features, extracted from the real and fake images while not "seeing" the images themselves.

Next, we described the system that converts a representation of a 3D pose of an actor into a rendering of a specific human. This system uses the classical idea of geometry texturing via UV mapping, yet it never explicitly builds the geometry of the human. The generator network maps a given 3D pose into camera-projected UV coordinates, and the texture applied. The system can generate temporarily consistent human renderings and is suitable for real-time applications.

Finally, we explored the prior, imposed by the structure of a convolutional neural network. We described how an untrained convolutional neural network could successfully compete with the methods that require learning and thus prove that the convolutional structure of the network itself carries a strong low-level prior towards natural images.

A further research direction could be to apply the knowledge about the generation of two-dimensional images to learn to generate 3D objects and explore if 3D convolutions provide as useful prior for 3D tasks as 2D convolutions for 2D problems. Another important stream of research could be to explore better quantitative metrics for evaluation of image generation methods as currenly qulalitative evaluation is mostly adopted.

Bibliography

Faceapp. https://www.faceapp.com/, 2018.

Project webpage. http://egorzakharov.github.io/perceptual_gan, 2018.

Kfir Aberman, Mingyi Shi, Jing Liao, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Deep video-based performance cloning. *arXiv preprint arXiv:1808.06847*, 2018.

Oleg Alexander, Mike Rogers, William Lambeth, Jen-Yuan Chiang, Wan-Chun Ma, Chuan-Chang Wang, and Paul Debevec. The Digital Emily project: Achieving a photorealistic digital actor. *IEEE Computer Graphics and Applications*, 30(4):20–31, 2010.

Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Video based reconstruction of 3d people models. In *Proc. CVPR*, June 2018a.

Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Detailed human avatars from monocular video. In *2018 International Conference on 3D Vision (3DV)*, pages 98–109. IEEE, 2018b.

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *Proc. ICLR*, 2017.

ShahRukh Athar, Evgeniy Burnaev, and Victor S. Lempitsky. Latent convolutional models. *CoRR*, 2018.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

Artem Babenko, Anton Slesarev, Alexander Chigorin, and Victor S. Lempitsky. Neural codes for image retrieval. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, pages 584–599, 2014.

Yuval Bahat, Netalee Efrat, and Michal Irani. Non-uniform blind deblurring by reblurring. In *Proc. CVPR*, pages 3286–3294, 2017.

Guha Balakrishnan, Amy Zhao, Adrian V. Dalca, Frédo Durand, and John V. Guttag. Synthesizing images of humans in unseen poses. In *Proc. CVPR*, pages 8340–8348, 2018.

Alexandru O Bălan and Michael J Black. The naked truth: Estimating body shape under clothing. In *Proc. ECCV*, pages 15–29. Springer, 2008.

Sagie Benaim and Lior Wolf. One-sided unsupervised domain mapping. In *Proc. NIPS*, pages 752–762, 2017.

Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie-Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *Proc. BMVC*, pages 1–10, 2012.

Federica Bogo, Michael J Black, Matthew Loper, and Javier Romero. Detailed full-body reconstructions of moving people from monocular RGB-D sequences. In *Proc. ICCV*, pages 2300–2308, 2015.

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *Proc. ECCV*, pages 561–578. Springer, 2016.

Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *CoRR*, abs/1707.05776, 2017.

Lokesh Boominathan, Mayug Maniparambil, Honey Gupta, Rahul Baburajan, and Kaushik Mitra. Phase retrieval for fourier ptychography under varying amount of measurements. *CoRR*, 2018.

Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *Proc. ICLR*, 2017.

Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *Proc. CVPR*, volume 2, pages 60–65. IEEE, 2005.

Harold C Burger, Christian J Schuler, and Stefan Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *2012 IEEE conference on computer vision and pattern recognition*, pages 2392–2399. IEEE, 2012.

Martin Burger, Stanley J. Osher, Jinjun Xu, and Guy Gilboa. Nonlinear inverse scale space methods for image restoration. In *Variational, Geometric, and Level Set Methods in Computer Vision, Third International Workshop, VLSM*, pages 25–36, 2005.

Jie Cao, Yibo Hu, Hongwen Zhang, Ran He, and Zhenan Sun. Learning a high fidelity pose invariant model for high-resolution face frontalization. *arXiv preprint arXiv:1806.08472*, 2018.

Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proc. CVPR*, 2017.

Dan Casas, Marco Volino, John Collomosse, and Adrian Hilton. 4d video textures for interactive character appearance. In *Computer Graphics Forum*, volume 33, pages 371–380. Wiley Online Library, 2014.

Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. *arXiv preprint arXiv:1808.07371*, 2018.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *Proc. ICLR*, 2017.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. ECCV*, 2018.

Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proc. ICCV*, pages 1520–1529, 2017.

Soumith Chintala, Emily Denton, Martin Arjovsky, and Michael Mathieu. How to train a GAN? Tips and tricks to make GANs work. https://github.com/soumith/ganhacks, 2017.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proc. CVPR*, June 2018.

Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (TOG)*, 34(4):69, 2015.

Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

Paul E. Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering Techniques '98, Proceedings of the Eurographics Workshop in Vienna, Austria, June 29 - July 1, 1998*, pages 105–116, 1998.

Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR*, abs/1506.05751, 2015.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *Proc. ICLR*, 2017.

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *Proc. ECCV*, pages 184–199, 2014a.

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014b.

Craig Donner, Tim Weyrich, Eugene d'Eon, Ravi Ramamoorthi, and Szymon Rusinkiewicz. A layered, heterogeneous reflectance model for acquiring and rendering human skin. In *ACM Transactions on Graphics (TOG)*, volume 27, page 140. ACM, 2008.

A. Dosovitskiy and T. Brox. Inverting convolutional networks with convolutional networks. In *Proc. CVPR*, 2016a.

Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Proc. NIPS*, pages 658–666, 2016b.

Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proc. Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1538–1546, 2015.

Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2fusion: real-time volumetric performance capture. *ACM Transactions on Graphics (TOG)*, 36(6): 246, 2017.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially learned inference. *Proc. ICLR*, 2017.

Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *CoRR*, abs/1505.03906, 2015.

M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

Andrew Feng, Dan Casas, and Ari Shapiro. Avatar reshaping and automatic rigging using a deformable model. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, pages 57–64. ACM, 2015.

Yaroslav Ganin, Daniil Kononenko, Diana Sungatullina, and Victor Lempitsky. Deepwarp: Photorealistic image resynthesis for gaze manipulation. In *Proc. ECCV*, pages 311–326. Springer, 2016.

Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems, NIPS*, pages 262–270, 2015a.

Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems, NIPS*, pages 262–270, 2015b.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015c.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015d.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *Proc. ICCV*, pages 349–356, 2009.

Bastian Goldlücke and Daniel Cremers. Superresolution texture maps for multiview reconstruction. In *Proc. ICCV*, pages 1677–1684, 2009.

Ke Gong, Xiaodan Liang, Yicheng Li, Yimin Chen, Ming Yang, and Liang Lin. Instance-level human parsing via part grouping network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 770–785, 2018.

Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NIPS*, pages 2672–2680, 2014.

Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems,NIPS*, pages 513–520, 2006.

Shuhang Gu, Wangmeng Zuo, Qi Xie, Deyu Meng, Xiangchu Feng, and Lei Zhang. Convolutional sparse coding for image super-resolution. In *ICCV*, pages 1823–1831. IEEE Computer Society, 2015.

Riza Alp Güler, George Trigeorgis, Epameinondas Antonakos, Patrick Snape, Stefanos Zafeiriou, and Iasonas Kokkinos. DenseReg: Fully convolutional dense shape regression in-the-wild. In *Proc. CVPR*, volume 2, page 5, 2017.

Riza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. DensePose: Dense human pose estimation in the wild. In *Proc. CVPR*, June 2018.

Nils Hasler, Hanno Ackermann, Bodo Rosenhahn, Thorsten Thormählen, and Hans-Peter Seidel. Multilinear pose and body shape estimation of dressed subjects from image sets. In *Proc. CVPR*, pages 1823–1830. IEEE, 2010.

Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *T-PAMI*, 35(6): 1397–1409, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015a. URL http://arxiv.org/abs/1512.03385.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034, 2015b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778, 2016.

Felix Heide, Wolfgang Heidrich, and Gordon Wetzstein. Fast and flexible convolutional sparse coding. In *Proc. CVPR*, pages 5135–5143, 2015.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/0893-6080(89)90020-8. URL http://www.sciencedirect.com/science/article/pii/0893608089900208.

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Trans. Graph.*, 36(4):107:1–107:14, 2017a.

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and Locally Consistent Image Completion. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 36(4): 107:1–107:14, 2017b.

Andrew Ilyas, Ajil Jalal, Eirini Asteri, Constantinos Daskalakis, and Alexandros G. Dimakis. The robust manifold defense: Adversarial training using generative models. *CoRR*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. International Conference on Machine Learning, ICML*, pages 448–456, 2015.

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

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proc. CVPR*, pages 5967–5976, 2017b.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Proc. NIPS*, pages 2017–2025, 2015.

Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. In *Proc. NIPS*, pages 769–776, 2009.

Justin Johnson. neural-style. https://github.com/jcjohnson/neural-style, 2015.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proc. CVPR*, 2018.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017a.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017b.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018a. URL https://openreview.net/forum?id=Hk99zCeAb.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018b.

Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Nießner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *arXiv preprint arXiv:1805.11714*, 2018.

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proc. CVPR*, pages 1646–1654, 2016.

Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proc. ICLR*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *Proc. ICLR*, 2014.

Oliver Klehm, Fabrice Rousselle, Marios Papas, Derek Bradley, Christophe Hery, Bernd Bickel, Wojciech Jarosz, and Thabo Beeler. Recent advances in facial appearance capture. In *Computer Graphics Forum*, volume 34, pages 709–733. Wiley Online Library, 2015.

Rolf Köhler, Christian Schuler, Bernhard Schölkopf, and Stefan Harmeling. Mask-specific inpainting with deep neural networks. In *German Conference on Pattern Recognition*, pages 523–534. Springer, 2014.

L. F. Kozachenko and N. N. Leonenko. Sample estimate of the entropy of a random vector. *Probl. Inf. Transm.*, 23(1-2):95–101, 1987.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proc.ICCV 3DRR Workshop*, pages 554–561, 2013.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *CoRR*, abs/1512.09300, 2015.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *Proc. CVPR*, 2017a.

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017b.

Stamatios Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *Proc. CVPR*, 2016.

Victor S. Lempitsky and Denis V. Ivanov. Seamless mosaicing of image-based texture maps. In *Proc. CVPR*, 2007.

Yujia Li, Kevin Swersky, and Richard S. Zemel. Generative moment matching networks. In *Proc. International Conference on Machine Learning, ICML*, pages 1718–1727, 2015.

Xiaodan Liang, Si Liu, Xiaohui Shen, Jianchao Yang, Luoqi Liu, Jian Dong, Liang Lin, and Shuicheng Yan. Deep human parsing with active template regression. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(12):2402–2414, Dec 2015a. ISSN 0162-8828. doi: 10.1109/TPAMI.2015.2408360.

Xiaodan Liang, Chunyan Xu, Xiaohui Shen, Jianchao Yang, Si Liu, Jinhui Tang, Liang Lin, and Shuicheng Yan. Iccv. 2015b.

Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. Neural animation and reenactment of human actor videos. *arXiv preprint arXiv:1809.03658*, 2018.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738. IEEE Computer Society, 2015a.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proc. ICCV*, 2015b.

Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Transactions on Graphics (TOG)*, 37(4):68, 2018.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,CVPR*, pages 3431–3440, 2015a.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015b.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):248, 2015.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs created equal? A large-scale study. *CoRR*, abs/1711.10337, 2017.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proc. CVPR*, 2015.

Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *IJCV*, 2016.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11 (Jan):19–60, 2010.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *Proc. ICLR*, 2016.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016.

Antonio Marquina. Nonlinear inverse scale space methods for total variation blind deconvolution. *SIAM J. Imaging Sciences*, 2(1):64–83, 2009.

Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien P. C. Valentin, Sameh Khamis, Philip L. Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B. Goldman, Cem Keskin, Steven M. Seitz, Shahram Izadi, and Sean Ryan Fanello. *LookinGood*: enhancing performance capture with real-time neural re-rendering. *ACM Trans. Graph.*, 37(6): 255:1–255:14, 2018.

Youssef Marzouk, Tarek Moselhy, Matthew Parno, and Alessio Spantini. An introduction to sampling via measure transport. *arXiv preprint arXiv:1602.05023*, 2016.

Masahiro Mori. The uncanny valley. *Energy*, 7(4):33–35, 1970.

Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. GANerated hands for real-time 3d hand tracking from monocular RGB. In *Proc. CVPR*, June 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Natalia Neverova, Riza Alp Güler, and Iasonas Kokkinos. Dense pose transfer. In *Proc. ECCV*, September 2018.

Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter V. Gehler, and Bernt Schiele. Neural body fitting: Unifying deep learning and model-based human pose and shape estimation. Verona, Italy, 2018.

G. Owen. *Game Theory*. Academic Press, 1982. ISBN 9780125311502. URL https://books.google.ru/books?id=pusfAQAAIAAJ.

Vardan Papyan, Yaniv Romano, and Michael Elad. Convolutional neural networks analyzed via convolutional sparse coding. *Journal of Machine Learning Research*, 18(83): 1–52, 2017a.

Vardan Papyan, Yaniv Romano, Jeremias Sulam, and Michael Elad. Convolutional dictionary learning via local processing. In *Proc. ICCV*, 2017b.

Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *Proc. CVPR*, June 2018.

Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael F. Cohen, Hugues Hoppe, and Kentaro Toyama. Digital photography with flash and no-flash image pairs. *ACM Trans. Graph.*, 23(3):664–672, 2004.

Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J Black. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics (TOG)*, 34(4):120, 2015.

J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, 40(1):49–70, 2000.

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

Alex Rav-Acha, Pushmeet Kohli, Carsten Rother, and Andrew W. Fitzgibbon. Unwrap mosaics: a new representation for video editing. *ACM Trans. Graph.*, 27(3):17:1–17:11, 2008.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2014, Columbus, OH, USA, June 23-28, 2014*, pages 512–519, 2014.

Jimmy S. J. Ren, Li Xu, Qiong Yan, and Wenxiu Sun. Shepard convolutional neural networks. In *Proc. NIPS*, pages 901–909, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Nadia Robertini, Dan Casas, Edilson De Aguiar, and Christian Theobalt. Multi-view performance capture of surface details. *International Journal of Computer Vision*, 124(1):96–113, 2017.

Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

D. E. Rumelhart and J. L. McClelland. *Learning Internal Representations by Error Propagation*. MITP, 1987. ISBN 9780262291408. URL https://ieeexplore.ieee.org/document/6302929.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL http://arxiv.org/abs/1409.0575.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015a. doi: 10.1007/s11263-015-0816-y.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015b.

Mehdi S. M. Sajjadi, Bernhard Scholkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *Proc. ICCV*, 2017a.

Mehdi S. M. Sajjadi, Bernhard Scholkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017b.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2226–2234, 2016.

Otmar Scherzer and Charles W. Groetsch. Inverse scale space theory for inverse problems. In *Scale-Space and Morphology in Computer Vision, Third International Conference*, pages 317–325, 2001.

Prasan A. Shedligeri, Ketul Shah, Dhruv Kumar, and Kaushik Mitra. Photorealistic image reconstruction from hybrid intensity and event based sensor. *CoRR*, 2018.

Zhixin Shu, Mihir Sahasrabudhe, Riza Alp Guler, Dimitris Samaras, Nikos Paragios, and Iasonas Kokkinos. Deforming autoencoders: Unsupervised disentangling of shape and appearance. In *Proc. ECCV*, September 2018.

Aliaksandr Siarohin, Enver Sangineto, Stphane Lathuilire, and Nicu Sebe. Deformable gans for pose-based human image generation. In *Proc. CVPR*, June 2018.

Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

J Starck and A Hilton. Model-based multiple view reconstruction of people. In *Proc. ICCV*, pages 915–922, 2003.

Ian Stavness, C Antonio Sánchez, John Lloyd, Andrew Ho, Johnty Wang, Sidney Fels, and Danny Huang. Unified skinning of rigid and deformable models for anatomical simulations. In *SIGGRAPH Asia 2014 Technical Briefs*, page 9. ACM, 2014.

Diana Sungatullina, Egor Zakharov, Dmitry Ulyanov, and Victor Lempitsky. Image manipulation with perceptual discriminators. In *Proc. ECCV*, September 2018.

Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Synthesizing Obama: learning lip sync from audio. *ACM Transactions on Graphics (TOG)*, 36 (4):95, 2017.

Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *CoRR*, abs/1611.02200, 2016. URL http://arxiv.org/abs/1611.02200.

Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Proc. CVPR*, pages 103–110. IEEE, 2012.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proc. CVPR*, June 2018.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep Image Prior webpage. https://dmitryulyanov.github.io/deep_image_prior.

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1349–1357, 2016.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. In *Proc. CVPR*, 2018.

Paul Upchurch, Jacob R. Gardner, Geoff Pleiss, Robert Pless, Noah Snavely, Kavita Bala, and Kilian Q. Weinberger. Deep feature interpolation for image content changes. In *Proc. CVPR*, pages 6090–6099, 2017.

David Van Veen, Ajil Jalal, Eric Price, Sriram Vishwanath, and Alexandros G. Dimakis. Compressed sensing with deep image prior and learned regularization. *CoRR*, 2018.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Marco Volino, Dan Casas, John P Collomosse, and Adrian Hilton. Optimal representation of multi-view video. In *Proc. BMVC*, 2014.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *arXiv preprint arXiv:1711.11585*, 2017.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018.

David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. In *Proc. ICLR*, 2017.

Lingyu Wei, Liwen Hu, Vladimir Kim, Ersin Yumer, and Hao Li. Real-time hair rendering using sequential adversarial networks. In *Proc. ECCV*, September 2018.

Alexander Weiss, David Hirshberg, and Michael J Black. Home 3d body scans from noisy image and range data. In *Proc. ICCV*, pages 1951–1958. IEEE, 2011.

Tim Weyrich, Wojciech Matusik, Hanspeter Pfister, Bernd Bickel, Craig Donner, Chien Tu, Janet McAndless, Jinho Lee, Addy Ngan, Henrik Wann Jensen, et al. Analysis of human faces using a measurement-based skin reflectance model. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1013–1024. ACM, 2006.

Olivia Wiles, A. Sophia Koepke, and Andrew Zisserman. X2face: A network for controlling face generation using images, audio, and pose codes. In *Proc. ECCV*, September 2018.

Erroll Wood, Tadas Baltrusaitis, Xucong Zhang, Yusuke Sugano, Peter Robinson, and Andreas Bulling. Rendering of eyes for eye-shape registration and gaze estimation. In *Proc. ICCV*, pages 3756–3764, 2015.

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger B. Grosse. On the quantitative analysis of decoder-based generative models. *CoRR*, abs/1611.04273, 2016.

Feng Xu, Yebin Liu, Carsten Stoll, James Tompkin, Gaurav Bharaj, Qionghai Dai, Hans-Peter Seidel, Jan Kautz, and Christian Theobalt. Video-based characters: creating new human performances from a multi-view video database. *ACM Transactions on Graphics (TOG)*, 30(4):32, 2011.

Dong Yang and Jian Sun. Bm3d-net: A convolutional neural network for transform-domain collaborative filtering. *IEEE Signal Processing Letters*, 25(1):55–59, 2018.

Tao Yu, Zerong Zheng, Kaiwen Guo, Jianhui Zhao, Qionghai Dai, Hao Li, Gerard Pons-Moll, and Yebin Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In *Proc. CVPR*, pages 7287–7296. IEEE Computer Society, 2018.

Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Proc. CVPR*, pages 2528–2535, 2010.

Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, volume 6920 of *Lecture Notes in Computer Science*, pages 711–730. Springer, 2010.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *Proc. ICLR*, 2017.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *CoRR*, abs/1612.03242, 2016.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. Energy-based generative adversarial network. *Proc. ICLR*, 2017.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *Proc. ECCV*, 2016.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017a.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. ICCV*, pages 2242–2251, 2017b.

Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Proc. NIPS*, pages 465–476, 2017c.

S. C. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *IJCV*, 27(2), 1998.

S. C. Zhu, X. W. Liu, and Y. N. Wu. Exploring texture ensembles by efficient markov chain monte carlotoward a atrichromacyo theory of texture. *PAMI*, 2000.