UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Application of Generative Neural Models for Style Transfer Learning in Fashion

Author:
Mykola MYKHAILYCH

Supervisor:
Dr. Rostyslav HRYNIV

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2017

Declaration of Authorship

I, Mykola MYKHAILYCH, declare that this thesis titled, "Application of Generative Neural Models for Style Transfer Learning in Fashion" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Abstract

Faculty of Applied Sciences

Master of Science

Application of Generative Neural Models for Style Transfer Learning in Fashion

by Mykola MYKHAILYCH

The purpose of this thesis is to analyze different generative adversarial networks for application in fashion. Research of "mode collapse" problem of generative adversarial networks. We studied the theoretical part of the "mode collapse" and conducted experiments on a synthetic toy dataset, and a dataset containing real data from fashion. Due to the developed method, it was possible to achieve visible results of improving the quality of garment generation by solving the problem of collapse.

Acknowledgements

I would like to thank Artem Chernodub, who super-vised my research and directed me during thesis writing. Also, I want to grateful Wix.com Ltd, Arvilab for computational resources and consultation.

Contents

List of Figures

List of Tables

List of Abbreviations

GAN	**G**enerative **A**dvertisal **N**entwork.
CGAN	**C**conditional **G**enerative **A**dvertisal **N**entwork.
WGAN	**W**asserstein **G**enerative **A**dvertisal **N**entwork.
KL	**K**ullback-**L**eibler.
JS	**J**ensen-**S**hannon.
EM	**E**arth-**M**over.
TV	**T**otal **V**ariation.
PDF	**P**robability **d**ensity **f**unction.
VAE	**Variational** **A**uto**e**ncoders.
CNN	**C**onvolutional **N**eural **N**etwork.

List of Symbols

A	Samples from real data.
B	Samples from approximated distribution.
y	Labels, condition.
$p_{\cdot}$	Distribution.
$\mathbb{P}_{\cdot}$	PDF.
$\mathcal{D}$	Space of all discriminators.
$\mathcal{G}$	Space of all generators.
$G_{\mathcal{G}}$	Some particular generator from space of all generators.
D_{D}	Some particular discriminator from space of all discriminators.
$\mathbb{R}^{d}$	Real space.
$\mathbb{E}$	Mathematical expectation

To my patient and loving wife

Chapter 1

Introduction

1

Essentially, there are several kinds of models in machine learning — generative models and discriminative models. Suppose you have a supervised learning task, where x_i are the given features of the data points, and y_i are the corresponding labels. One way to predict y on future x is to learn a function f from (x_i, y_i) that takes in x and outputs the most likely y. Such models fall into the category of discriminative models since you are learning how to discriminate between x's from different classes. Methods like SVMs, neural networks fall into this category. Even if you're able to classify the data very accurately, you have no notion of how the data might have been generated. The second approach is to model how the data might have been generated, and learn a function $f(x, y)$ that gives a score to the configuration determined by x and y together. Then you can predict y for a new x by finding the y for which the score $f(x, y)$ is maximum. A canonical example of this is Gaussian mixture models.

In terms of probability, we can formulate that discriminative models try to learn $\mathrm{p(y|x)}$ conditional probability, on the other hand, generative models try to approximate $\mathrm{p(x,y)}$. If we learn joint distribution, we would be able to sample from this distribution.

Generally, research community allocates this types of generative models:

- Gaussian mixture model (and other types of mixture model).
- Hidden Markov model.
- Probabilistic context-free grammar.

- Naive Bayes.
- Latent Dirichlet allocation.
- Restricted Boltzmann machine.
- Generative adversarial networks.
- etc.

Figure 1.1 visualize family of generative models. In my work, I'll review generative adversarial nets for the image-to-image translation task.

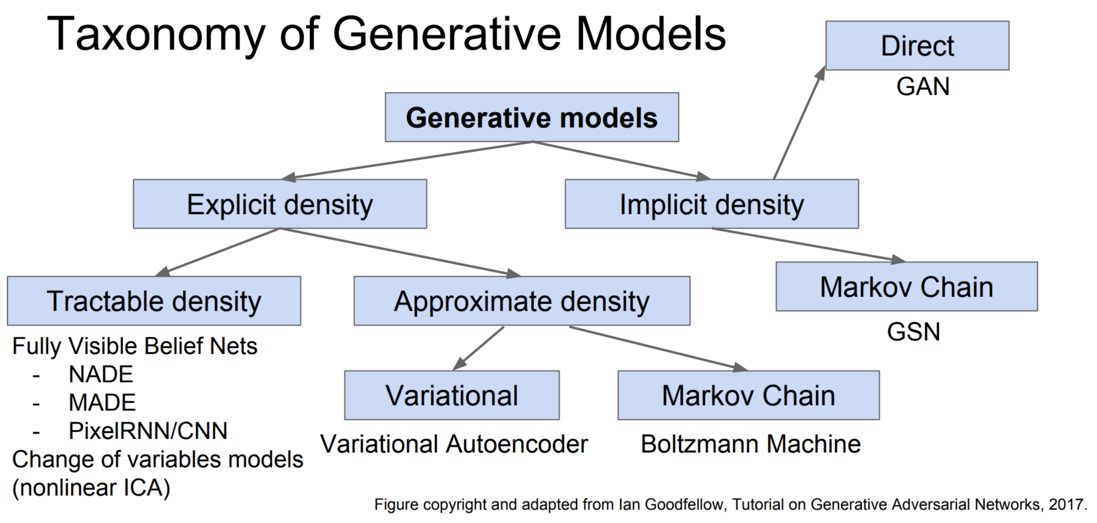


FIGURE 1.1: Family of generative models.

Neural networks have received a lot of attention in the past decade. Research in this area is very active. In 2014, was published paper "Generative Adversarial Nets"(Goodfellow et al., 2014). In this paper, authors proposed a new framework for estimating generative models via an adversarial process. This work gave impetus to the development of this approach. During the last 3 years, community published different variations of GANs.

After paper about adversarial nets was published, we saw different applications of this technology. People started using it for:

- Super resolution task (Ledig et al., 2016).
- Image generation (Liu et al., 2017).

- Style transfer (Zhu et al., 2017).
- Reinforcement learning (Ho and Ermon, 2016).
- Natural Language Processing (Rajeswar et al., 2017).
- Next frame generation for video (Ghosh et al., 2016).
- etc.

Although introduced in 2014 by Ian Goodfellow, it is in 2016 that GANs have started to show their real potential. Improved techniques for helping training and better architectures (Deep Convolutional GAN, Radford, Metz, and Chintala, 2015) have fixed some of the previous limitations, and new applications are revealing how powerful and flexible they can be.

In the proposed, by Goodfellow, adversarial nets framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles.

1.1 Goals

In this master's thesis, we want to explore generative adversarial nets, show it's potential and application in the fashion industry. We'll try out current state-of-the-art models and show it's results for the custom dataset. Also, we'll bring our conjectures, about how they can be improved. We will conduct experiments based on my guesses. And present results. Hence, the main goal of this work to study principles of generative adversarial networks, it's advantages and disadvantages, highlight main method's ill problems and propose a novel solution that improves methods in terms of the fashion industry.

FIGURE 1.2: Generated images using GAN model (dataset Cifar-10)

In particular, we aim to solve the following problem:

> Given input image of a person dressed in a t-shirt. We want to extract mask of t-shirt from a person. After, our goal to generate a new one and change person's t-shirt. I should mention that there is no such t-shirt in the world. It's very important to generate t-shirt in such way, that it looks like the real one on person. Also in case, we have a big dataset of t-shirts, we can recommend real t-shirt, which is similar to generated. We can change t-shirt to any other close. But in our case, we'll use just t-shirt. Try to make the generation of t-shirts more creative.

We want to show, how to build such pipeline. We want to check if current state-of-the-art models able to satisfy fashion industry. And create some useful service for people. Current work can be used as an overview of generative adversarial nets.

1.2 Research Questions and Limitations

In this section, we list down the main challenges, limitations and research questions involved in finding the best approach for generating items inside some figure.

- There is no open source dataset of t-shirts (at the time of writing this thesis). There are a lot of datasets online, such as Fashionist (Dong et al., 2015). But it doesn't have enough masks of t-shirts. So, for our research, we need to create our own dataset. It'll include just t-shirts, we need to avoid images which have something else on it, accept t-shirt.
- It's complicated to train adversarial networks, because, in reality, you train two neural networks, with two different objective functions Salimans et al., 2016.
- Very important to train model in such way, that it will be a good approximation of real data. Good edge detector will help us generate new t-shirt with some special style.

1.3 Deep learning

Deep learning is very popular nowadays, and strong research topic. The most impressive thing, that supervised learning has demonstrated human performance for some discriminative tasks (Simonyan and Zisserman, 2014).

These models were implemented using a huge amount of data, apply a series of matrix multiplication and non-linear operations, that allow to draw decision boundary in high-dimensional space and classify samples in 1000 classes (Russakovsky et al., 2014) or help to define if a person has a cancer.

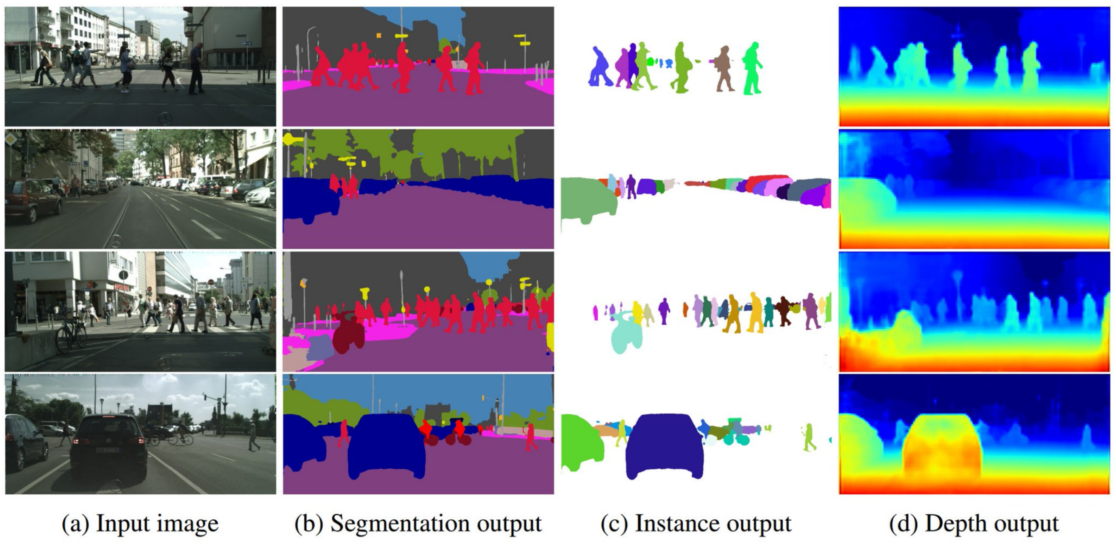


FIGURE 1.3: Example of DL case study. Here we can see results of training SegNet Badrinarayanan, Kendall, and Cipolla, 2017

But artificial intelligence NN have some cons, in order to achieve a state-of-the-art result, a NN needs a huge amount of labeled data, in some cases, could represent the laborious task. One of the way to overcome this problem - is to use unsupervised approaches instead (Salimans et al., 2016, Perarnau et al., 2016).

1.4 Generative models

An important subfield of unsupervised learning is generative modeling. It allows us to approximate the unknown distribution from its joint distribution. For example, it is possible to sample of high-dimensional image space, where each sample represents a generated image, unlike discriminative models, where the sampling is done over the distribution of class labels given an input image. In other words, given enough data from some domain (e.g. clothes), we can train a model which is able to generate more data from the given space. Probabilistic generative models can be used for texture synthesis (Raad et al., 2017), inpainting (Yeh et al., 2016), denoising (Divakar and Venkatesh Babu, 2017), compression (Santurkar, Budden, and Shavit, 2017), semi-supervised learning, unsupervised feature learning, etc. Given this wide list of application, gives us high heterogeneity in the way these models are formulated, trained and evaluated.

Many objective functions and training procedures have been proposed, for optimizing generative models (Nowozin, Cseke, and Tomioka, 2016). The motivation for introducing new training methods is typically the wish to fit probabilistic models with computationally intractable likelihoods, rendering direct maximum likelihood learning impractical. Most of the available training procedures are consistent in the sense that if the data is drawn from a model distribution, then this model distribution will be optimal under the training objective in the limit of an infinite number of training examples. That is if the model is correct, and for extremely large amounts of data, all of these methods will produce the same result. However, when there is a mismatch between the data distribution and the model, different objective functions can lead to very different results (Theis, van den Oord, and Bethge, 2015).

Generative models have many applications and can be evaluated in many ways. For density estimation and related tasks, log-likelihood has been the defacto standard for training and evaluating generative models. In our work, we'll make an overview of different objective functions for generative models, and their comparison.

Being able to generate data is the key feature of unsupervised learning and artificial intelligence in general. Generative models grants excellent position to know every aspect of this data. However, learning, and understanding of true distribution is challenging task. Also, the loss function is not clearly defined for generative models, as for discriminative tasks, the important disadvantage that there is no defined function for evaluation. For example, in case of image restoration, there is no well-working no inference image quality loss. The reason for that is that objective function and evaluation function depends on the many factors, such as the target of application, the complexity of the data and the type of generative model.

1.5 Image generation

As we mentioned in the introduction, in this thesis we cover a subset of generative modeling focused on image generation. Image generation has huge scope of application. For instance, it can be applied for deblurring (Kupyn et al., 2017), where a generator corrects image by adding more information on top of it. In fact, any image could be generated, if it's enough data to train properly generator. Another application could be to augment a dataset in order to use naive classification algorithms for classification, instead of other computational expensive algorithms. Similarly, a NN used as a generative model can be used to learn in an unsupervised manner the representations of the data and, later, use the first layers of the NN as a feature extractor in a semi-supervised setting. Then, this approach would not require to hand-label as much data as it would for directly training a supervised model from scratch.

There are different approaches to image generation. The most popular ones are:

- Probabilistic graphical models with latent variables, such as Restricted Boltzmann Machines (Decelle, Fissore, and Furtlehner, 2017), Deep Belief Networks (Turner et al., 2017) and Deep Boltzmann Machines (Srivastava, Salakhutdinov, and Hinton, 2013). These models need to rely on Markov chain Monte Carlo methods to avoid the cost of an expensive partition function, which necessary for applying a global normalization.

- Variational AutoEncoders (VAEs Kingma and Welling, 2013). They are based on the probabilistic graphical model framework and trained with gradient-based methods. Autoencoders are used to learn the representation of the data using a pixel-wise loss function (e.g. mean square error). On the other hand, variational methods are applied to sample from a tractable distribution Q (e.g. normal distribution) instead of the feature vector distribution P.

- Autoregressive models, for example, Pixel Recurrent Neural Networks (van den Oord, Kalchbrenner, and Kavukcuoglu, 2016) and Conditional PixelCNN (van den Oord et al., 2016). The idea is to model each pixel of the image considering the previous pixels along two spatial dimensions. Their training is very stable and generates plausible samples with a high log-likelihood. However, the sampling process is applied on one pixel at a time, which is quite inefficient.

- Generative Adversarial Networks. It is the approach that will be applied in this thesis. This approach is using two NN (generator, discriminator), solving minimax problem. Adversarial nets have the advantages that Markov chains are never needed, only backpropagation is used to obtain gradients, no inference is required during learning, and a wide variety of factors and interactions can easily be incorporated into the model. Also, generative adversarial networks allow us to generate new images from learned distribution in the one-shot, not like in Pixel Recurrent Neural Networks (van den Oord, Kalchbrenner, and Kavukcuoglu, 2016), where image generating pixel-by-pixel process. Also, unlike fully observed models, GANs do not allow to have access to the approximated final generative distribution, but to just sample from it, which makes the marginal log-likelihood harder to obtain for evaluation purposes. In an unconditioned generative model, there is no control on modes of the data being generated. However, by conditioning the model on additional information it is possible to direct the data generation process. Such conditioning could be based on class labels, on some part of data for inpainting like, or even on data from different modality. We will use this approach in this thesis.

1.6 Thesis Structure

In section 2 we made the detailed overview of Generative adversarial networks. We discuss problems of GANs, and ways of solving it. Also, we tried to describe all GANs framework modification which led us to write this thesis. We pay much attention to discuss problems from which GANs framework suffers. We did it, because in chapter 3, we will describe our experiments, the problems we faced. After, we introduce our novel idea how we see to solve this problem and results. In the beginning of chapter 3 we give the detailed description of datasets we created and used.

Chapter 2

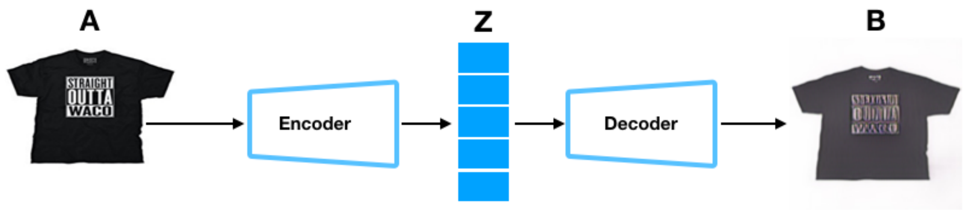
Overview of existing methods and problem's analysis

2

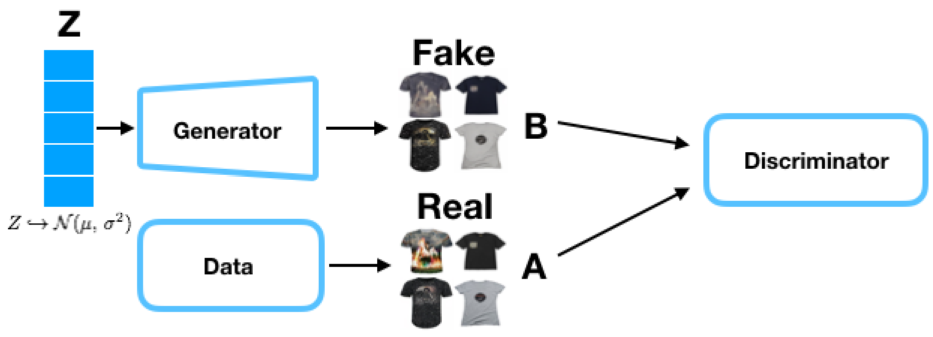
In this master thesis, we overview generative adversarial networks for image generation. Generative models are very important for the generation of some sample from some distribution. The main challenge for it, it generates realistic examples, in such way that even person couldn't recognize real this sample or not. This can be useful for training models, reinforcement learning. It'll be great to generate realistic samples with some conditions. In this chapter 2 we present the main idea of GAN framework, conditional GAN, Wasserstein GAN will be presented in the form we used them for our experiments, analysis, problem formulation.

2.1 Generative modeling

The most advanced results in generative modeling have been obtained using the different type of artificial neural networks. The reason for that is work of Krizhevsky, Sutskever, and Hinton, 2012. To this day, NNs have also achieved state-of-the-art, and even human-competitive results, not only in pattern recognition task. A significant contribution to NN is convolutional neural networks (CNN), which are very popular in signal processing tasks. The main idea of CNN's is to reduce a total number of parameters needed to train a NN, due to the fact that each point is not connected with every other point. With fewer



(A) VAE



(B) GAN

FIGURE 2.1: Main idea of VAE (a) and GAN (b) generate data.

parameters, overfitting is easier to avoid and the model is easier to train. Generative Adversarial Networks are strongly based on CNNs, as we will see in this section. I should notice that GANs can be formulated without using CNNs.

Natural image generation has been a strong research topic for many years, but it gave promising results with a combination of Deep learning and generative modeling (Goodfellow et al., 2014). Image generation models can be split into two main types: nonparametric and parametric. The first one does not generate an image from scratch, but often combine different patches between matched images in as database, which can be used, for example, inpainting technique. In this thesis, we are going to focus on the parametric model, which are able to generate natural images once a model is learned.

In my opinion, there are two promising approaches in building parametric models. First one is Variational Autoencoders (VAEs) (Kingma and Welling, 2013, Figure 2.1a). VAEs tries to make B image as similar as A with compressed representation Z. However, VAEs main limitation is that they tend to have a low performance with datasets with high variation. In order to reduce the variance of the gradient estimations, secondary networks (Mnih and Gregor, 2014) or conditional data (Kulkarni et al., 2015, Mansimov et al., 2015) are required. Another problem is the pixel-wise reconstruction error (e.g. mean square error) used as a loss function, which makes the model non-translation invariant and causes the output images to look blurry, as it generates the mean image of the distribution. Finally, we introduce below the state-of-the-art of the second approach of parametric models: Generative Adversarial Nets (GANs) (Figure 2.1b).

2.2 GAN framework

In this section, we review GAN framework in terms of distribution learning. Also, we will view how images are modeled as probability distributions. From a statistical point of view, generating images can be seen as sampling from a probability density function (PDF, $\mathbb{P}_X(x)$). PDF defines the likelihood of a random variable given a distribution. There are a lot of types of distributions, one of the most popular is the normal distribution. The main goal of PDF to model real-world data. However, the more complex the data, the more complex distribution tends to be. That means that there rare cases where it is not possible to formulate true distribution analytically. The only option you have it's to sample from this distribution. Image distribution represented in high-dimensional space, where each pixel gets its value according to this model. More concretely, when we make the photo with a camera, we sample from this distribution: $\mathbb{P}_X(x_1, x_2, x_3, ..., x_N)$, where $N = WxHx3$ (W, H - width, and height of image respectively), also we should add the spatial dependencies between pixels.

It's obvious that it's impossible to know the analytical true distribution of images. As this is not a case, we can approximate it as accurately as possible, it's challenging task. As we mentioned in section 2.1, the only possible approach (if we don't have labeled data) is generative models. Generative Adversarial nets where specifically designed to target this problem. in this following subsection, we explain the ability of GANs to approximate an unknown distribution.

TABLE 2.1: Notion overview of the GAN Framework

Samples	**Meaning**	**Distribution**	**Meaning**
A	Real data.	p_{data}	Real, complex distribution, $A \sim p_{data}$.
B	Fake data.	p_G	Generator distribution, $p_G = G_{\mathcal{G}}$, $B \sim p_G$.
z	Random noize.	p_z	Simple distribution (e.g. $\mathcal{N}(\mu, \sigma^2)$), $z \sim p_z$.
y	Label information.	p_y	Known label distribution, $p \sim p_y$.

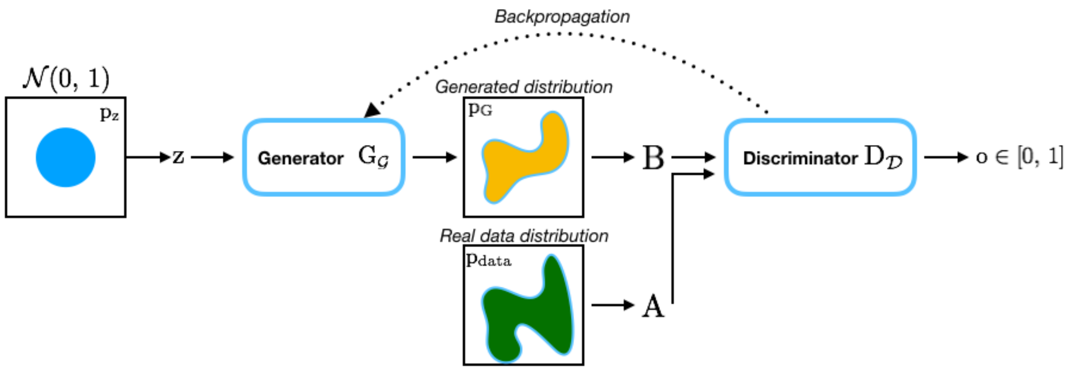


FIGURE 2.2: Pipeline for GAN training. Generator samples from z to generate B. The discriminator $D_{\mathcal{D}}$ trains with both fake B and real A data. Ideally we would have $p_G = p_{data}$.

2.2.1 Generative Adversarial Networks

This overview is based on paper the Goodfellow et al., 2014. A GAN is composed of two neural networks, a generator and discriminator. Let us define notations. Let's define $\mathcal{G}$ - space of all generator networks, $\mathcal{D}$ - space of all discriminators., $G_{\mathcal{G}}$ - sample from space of all generators, $D_{\mathcal{D}}$. In order to train both networks we need data A so we are able to generate fake data B from $G_{\mathcal{G}}$. To be more precisely, A is sampled from p_{data} and B from the generator $G_{\mathcal{G}}$ distribution p_G. The main goal of GANs is to accurately approximate p_G to p_{data}. In case of vanilla GAN, there is a third known distribution $z \sim p_z$ (e.g. a normal distribution) which is used as input for the generator $G_{\mathcal{G}}(z)$. Table 2.1 shows a summary of notations.

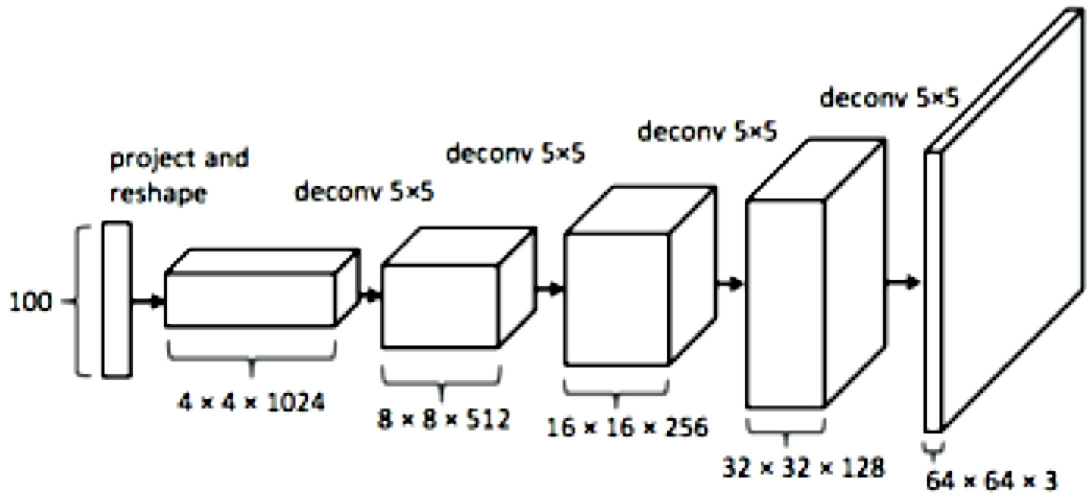


FIGURE 2.3: Example of generator $G_{\mathcal{G}}$.

The main idea of GAN is to iteratively train a discriminator $D_{\mathcal{D}}$ and generator $G_{\mathcal{G}}$, which are playing minimax game. The goal of $G_{\mathcal{G}}$ to approximate p_{data}, in such way, to fool $D_{\mathcal{D}}$, in the same time $D_{\mathcal{D}}$ iteratively becomes, better and better in distinguishing, where is generated sample and where is real. In other words, we want $G_{\mathcal{G}}$ to learn how to generate sampled from real data distribution p_{data}. On convergence, we expect $D_{\mathcal{D}}(A) = 0.5$, $D_{\mathcal{D}}(B) = 0.5$, which means that guesses of $D_{\mathcal{D}}(A) = 0.5$ absolutely random. All process and elements you can see on Figure 2.2.

We should mention, that $G_{\mathcal{G}}$ just tries to approximate p_{data}. The interesting fact is that $G_{\mathcal{G}}$ don't use any real data A from p_{data}, everything it has feedback from $D_{\mathcal{D}}$. Actually, this feedback is the gradients of a backpropagation step: $G_{\mathcal{G}}$ uses the gradients of the backpropagation error from $D_{\mathcal{D}}$ to change $G_{\mathcal{G}}$ parameters and slightly approximate p_{G} to p_{data}. Also, GANs have a dynamic function, that means, that at some point GAN can stop to train, there are a lot of reasons for that, for example, a weak $D_{\mathcal{D}}$, which output only 1. But that, allows us to create realistic data, in our case images.

More formally, let's introduce a function $V(\theta_{\mathrm{G}}, \theta_{\mathrm{D}})$, where G and D are the parameters of the generator and discriminator respectively, GAN training was introduced as optimizing of:

$$\min_{G} \max_{D} V(\theta_G, \theta_D) \tag{2.1}$$

where V is defined as,

$$V(\theta_G, \theta_D) = \mathbb{E}_{A \sim p_{data}}[\log D_{\mathcal{D}}(A)] + \mathbb{E}_{z \sim p_z}[\log(1 - D_{\mathcal{D}}(G_{\mathcal{G}}(z)))] \tag{2.2}$$

Goodfellow et al., 2014 showed very important fact in original GAN paper if considering an optimal discriminator, the generator minimizes the Jensen-Shannon divergence of an unknown distribution p_{data} and an approximate distribution p_G. This fact allows to avoid sampling from the mean of the distribution p_G, and as we know it causes generation of blurry results, as it is a common issue for generative models that minimizes the Kullback-Leibler divergence (Kullback and Leibler, 1951).

As we mentioned before, the generator $G_{\mathcal{G}}$ uses $z \sim p_z$ as input and outputs a generated image $B \sim p_G$. We can understand it as mapping between p_z and p_G. We call it decoding process because it decodes a compressed latent representation of vector z to a full image B. In the same time $D_{\mathcal{D}}$ receives as inputs an image A and as output return $o \in [0, 1]$. Then, o is compared to a binary binary label in origin of an image: real (1), or generated (0). More formally we can say that discriminator $D_{\mathcal{D}}$ evaluates the conditional probability $\mathbb{P}(\text{label}|\text{image})$ and the generator $G_{\mathcal{G}}$ evaluates $\mathbb{P}(B|z)$.

The architectures of $G_{\mathcal{G}}$ and $D_{\mathcal{D}}$ are opposed. On place of $D_{\mathcal{D}}$ can be any standard CNN for image classification (net with constitutional layers, activation functions, non-linearities). The most interesting is $G_{\mathcal{G}}$ architecture, it takes random noise vector z and convert it into an image B, we achieve it using transpose convolution (Long, Shelhamer, and Darrell, 2014). It allows us to upsample the input. In Figure 2.3 we can see an example of the architecture of a generator.

Let's talk about disadvantages of "vanilla" GAN, we introduced recently. It still remains unclear whether or not convergence in GANs is guaranteed, both theoretically and empirically. Future research is focusing on this direction. But until, research of GANs have two direction: improve of GANs framework, for better distribution approximation (Salimans et al., 2016, Arjovsky, Chintala, and Bottou, 2017), loss functions (Nowozin, Cseke, and Tomioka, 2016, Arora and Zhang, 2017), and more practical direction, GANs framework, modification for application it in real life (Tolstikhin et al., 2017, Mirza and Osindero, 2014, Isola et al., 2016). Recent applications of GANs have shown that they can produce excellent samples. However, the main idea of optimization of GANs in finding Nash equilibrium of a non-convex game with continuous, high dimensional parameters. Typically GANs trained using gradient descent methods, that designed to find a minimum of the loss function, rather than to find the Nash equilibrium of a game. When used to seek for a Nash equilibrium, these algorithms may fail to converge (Salimans et al., 2016). It still remains unclear whether or not convergence in GANs is guaranteed, both theoretically and empirically. Future research is focusing on this direction. But until, research of GANs have two direction: improve of GANs framework, for better distribution approximation (Salimans et al., 2016, Arjovsky, Chintala, and Bottou, 2017), loss functions (Nowozin, Cseke, and Tomioka, 2016, Arora and Zhang, 2017), and more practical direction, GANs framework, modification for application it in real life (Tolstikhin et al., 2017, Mirza and Osindero, 2014, Isola et al., 2016). Recent applications of GANs have shown that they can produce excellent samples. However, the main idea of optimization of GANs in finding Nash equilibrium of a non-convex game with continuous, high dimensional parameters. Typically GANs trained using gradient descent methods, that designed to find a minimum of the loss function, rather than to find the Nash equilibrium of a game. When used to seek for a Nash equilibrium, these algorithms may fail to converge (Goodfellow, 2014).

2.2.2 Conditional GAN

Goodfellow et al., 2014, in his paper mentioned about extensions of GANs, one of it, was Conditional generative adversarial networks and Mirza and Osindero, 2014 introduced it in his work. Conditional GANs are very similar to "vanilla" GANs, the only difference is that, in this case, we have extra information y (e. g. class label). This label strictly depends from the sample A. In case we can model a density model p_y in order to sample labels y for generated data

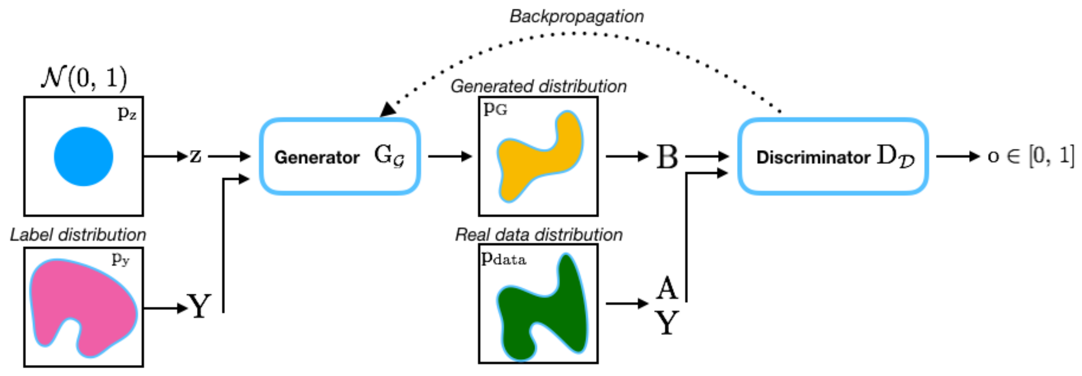


FIGURE 2.4: Conditional GAN.

B. On Figure 2.4, you can see updated pipeline for conditional GAN extension, as you can see generator $G_\mathcal{G}$ and discriminator $D_\mathcal{D}$ takes y. In the generator, y fed in the first layer. In discriminator, y we can introduce it in any layer (Mirza and Osindero, 2014). More formally, we can reformulate as:

$$V(\theta_G, \theta_D) = \mathbb{E}_{A,y \sim p_{data}}[\log D_\mathcal{D}(A, y)] + \mathbb{E}_{z \sim p_z, y \sim p_y}[\log (1 - D_\mathcal{D}(G_\mathcal{G}(z, y), y))] \tag{2.3}$$

Generally, we can say that conditional GAN can convergent more rapidly, than "vanilla" GAN, because both generator $G_\mathcal{G}$ and discriminator $D_\mathcal{D}$ takes additional information. For example, if you want to learn generator $G_\mathcal{G}$ to generate samples from MNIST dataset (LeCun and Cortes, 2010), in the same time we want control $G_\mathcal{G}$ to generate digit we want. In this case we fed vector of length 10 to both, generator $G_\mathcal{G}$ and discriminator $D_\mathcal{D}$. Discriminator $D_\mathcal{D}$ will adapt weights θ_D to understand meaning of vector y, and through backpropogation update θ_G in such way that $G_\mathcal{G}$ will learn how to interpret y. After publication of this paper (Mirza and Osindero, 2014), started another view for image generation models, more detailed overview, I will present in Chapter 3.

We can see conditional GAN from probabilistic point of view. $G_\mathcal{G}(z, y)$, as we mentioned, modeling distribution p_{data}, given z, y, that is our data is generated with this scheme $B \sim G_{\mathcal{G}}(B|z, y)$. Likewise for the discriminator, now it tries to find discriminating label for A and B, that are modeled with $o \sim D_{\mathcal{D}}(o|A, y)$. Hence, we could see that both $G_{\mathcal{G}}$ and $G_{\mathcal{G}}$ is jointly conditioned to two variables z or A and y.

2.3 WGAN

As we said in last section, there are two types of GAN researches, one that applies GAN in interesting problems and one that attempts to stabilize the training. Stabilizing GAN training is a very big deal in the field. The "vanilla" GAN suffers from several difficulties.

Firstly, it's mode collapse. It is usually referred to a problem when all the generator $G_{\mathcal{G}}$ outputs are identical (all of them or most of the samples are equal). In real world, distributions are complicated and multi modal. For example data distribution with two "peaks", where different sub-groups of samples are concentrated. In such a case a generator $G_{\mathcal{G}}$ can learn to yield images only from one of the sub-groups, causing mode collapse. This problem was actively tried to solve in Salimans et al., 2016, Tolstikhin et al., 2017, Arjovsky, Chintala, and Bottou, 2017, Gulrajani et al., 2017. My work is based on solving this problem, in the next chapter, I will overview this problem in more details.

Secondly, it's metric in GAN training, that tells us about the convergence. The generator $G_{\mathcal{G}}$ and discriminator $D_{\mathcal{D}}$ loss do not tell us anything about this. Of course we could monitor the training progress by looking at the data generated from generator every now and then. However, it is a strictly manual process. So, it would be great to have an interpretable metric that tells us about the training progress. That's why I make overview of Wasserstein GAN (Arjovsky, Chintala, and Bottou, 2017).

Classical approach for approximation of a probability distribution is to learn a probability density. This is often done by defining a parametric family of densities $(\mathrm{P}_\theta)_{\theta\in\mathbb{R}^\mathrm{d}}$ and finding the one, that maximized on our data:

$$\max_{\theta\in\mathbb{R}^\mathrm{d}} \frac{1}{m}\sum_{i=1}^{m} \log \mathrm{P}_\theta(\mathrm{x}^{(i)}) \tag{2.4}$$

If the real data distribution p_{data} admits a density and p_θ is the distribution of the parametrized density P_θ, then, asymptotically, this amounts to minimizing the Kullback-Leibler divergence $\mathrm{KL}(\mathrm{p}_{data}||\mathrm{p}_\theta)$.

For this to make sense, we need the model density p_θ to exist. This is not the case in the rather common situation where we are dealing with distributions supported by low dimensional manifolds. It is then unlikely that the model manifold and the true distribution's support have a non-negligible intersection, and this means that the KL distance is not defined (or simply infinite). The simple idea to solve this problem was proposed in Kaae Sønderby et al., 2016 work. They proposed to add a noise term to the model distribution. This is why virtually all generative models described in the classical machine learning literature include a noise component. But for image generation, this approach degrades the quality of the samples and makes them blurry.

In Wasserstein GAN paper was proposed method to solve this optimization problems. But for explanation, we need to introduce some notations and theoretical background. Let $\mathcal{X}$ be a compact metric set (such as the space of images $[0,1]^\mathrm{d}$) and let $\sum$ denote the set of all the Borel subsets of $\mathcal{X}$. Let $\mathrm{p}(\mathcal{X})$ denote space of probability measures defined on $\mathcal{X}$. We can now define elementary distances and divergences between two distributions $\mathrm{p}_{data}, \mathrm{p}_\mathrm{G} \in \mathrm{p}(\mathcal{X})$:

- The Total Variation distance:

$$\delta(\mathrm{p}_{data}, \mathrm{p}_\mathrm{G}) = \sup_{\mathrm{A}\in\sum} |\mathrm{p}_{data} - \mathrm{p}_\mathrm{G}|. \tag{2.5}$$

- The Kullback-Leibler (KL) divergence:

$$\mathrm{KL}(\mathrm{p}_{data}||\mathrm{p}_\mathrm{G}) = \int \log(\frac{\mathrm{p}_{data}(x)}{\mathrm{p}_\mathrm{G}(x)})\mathrm{p}_{data}(x)d\mu(x). \tag{2.6}$$

- The e Jensen-Shannon (JS) divergence:

$$JS(p_{data}, p_G) = KL(p_{data}||p_m) + KL(p_G||p_m). \tag{2.7}$$

where p_m is the mixture $\frac{p_{data}+p_G}{2}$. Optimization of "vanilla" GAN is the same as optimization of JS divergence, that have been shown in this work Goodfellow et al., 2014.

- The Earth-Mover (EM) distance or Wasserstein-1:

$$W(p_{data}, p_G) = \inf_{\gamma \in \prod(p_{data}, p_G)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||]. \tag{2.8}$$

where $\prod(p_{data}, p_G)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively p_{data} and p_G. Intuitively, $\gamma(x, y)$ indicates how much “mass” must be transported from x to y in order to transform the distribution p_G. The EM distance then is the “cost” of the optimal transport plan.

In Wasserstein GAN paper authors proofed the fact that $W(p_{data}, p_G)$ have nicer properties when optimized than $JS(p_{data}, p_G)$ (vanilla GAN loss optimization). However, equation 2.8 can be reformulated in terms of Kantorovich-Rubinstein duality (Villani, 2008):

$$W(p_{data}, p_G) = \sup_{||f||_L \le 1} \mathbb{E}_{x\sim p_{data}}[f(x)] - \mathbb{E}_{x\sim p_G}[f(x)] \tag{2.9}$$

where the supremum is over all the 1-Lipschitz functions $f : \mathbb{X} \to \mathbb{R}$. Note that if we replace $||f||_L \le 1$ for $||f||_L \le K$, then we end up with $KW(p_{data}, p_G)$. And finally, we can formulate optimization function for Wasserstein GAN as:

$$\min_G \max_{D\in\mathcal{D}} \mathbb{E}_{B\sim p_{data}}[D(B)] - \mathbb{E}_{A\sim p_G}[D(A)] \tag{2.10}$$

Where$\mathcal{D}$ is the set of 1-Lipschitz functions.

The WGAN value function results in a critic function whose gradient with respect to its input is better behaved than its GAN counterpart, making optimization of the generator easier. Additionally, WGAN has the desirable property that its value function correlates with sample quality, which is not the case for GANs.

In practice, authors of Wasserstein GAN proposed to optimize WGAN through optimization of neural network with weights w in compact space $\mathcal{W}$. Note that the fact that $\mathcal{W}$ is compact implies that all the functions f_{w} will be K-Lipschitz for some K that only depends on $\mathcal{W}$ and not the individual weights, therefore approximating up to an irrelevant scaling factor and the capacity of the f_{w}. In order to have parameters w lie in a compact space, something simple we can do is clamp the weights to a fixed box $\mathrm{w} = [-0.01, 0.01]$ after each gradient update.

Weight clipping is a clearly terrible way to enforce a Lipschitz constraint. If the clipping parameter is large, then it can take a long time for any weights to reach their limit, thereby making it harder to train the critic till optimality. If the clipping is small, this can easily lead to vanishing gradients when the number of layers is big, or batch normalization is not used.

Later, community proposed another way to enforce the Lipschitz constraint. A differentiable function is 1-Lipschtiz if and only if it has gradients with norm at most 1 everywhere, so they consider directly constraining the gradient norm of the discriminator's output with respect to its input. Formally we can present it as:

$$\min_{G} \max_{D \in \mathcal{D}} \mathbb{E}_{B \sim p_{data}}[D(B)] - \mathbb{E}_{A \sim p_G}[D(A)] + \lambda \mathbb{E}_{x \sim p_x}[(||\nabla_x D(x)||_2 - 1)^2] \quad (2.11)$$

Where p_x sampling uniformly along straight lines between pairs of points sampled from the data distribution p_{data} and the generator distribution p_G. This is motivated by the fact that the graph of the optimal critic consists of straight lines connecting points from p_{data} and p_G. Given that enforcing the unit gradient norm constraint everywhere is intractable, enforcing it only along these straight lines seems sufficient and experimentally results in good performance. This researches partially solved problems of "vanilla" GANs. Theoretically, WGAN improved stability and quality of GAN training. Also on practice, WGAN get better results in terms of "mode collapse" problem.

Chapter 3

Experiments

3

3.1 Datasets

We use two image datasets of different complexity and variation. For our experiments we created two datasets. One we call Shapes dataset. It's synthetic dataset. Second one we call T-shirts dataset. It's quite big dataset, parsed from different Internet resources, where it's allowed by the policy of resource.

3.1.1 Synthetic generated dataset

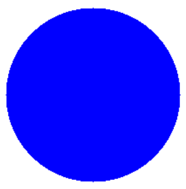
This dataset was synthetically generated. Shapes dataset consist of different geometric shapes on white background, shapes have different colors, sizes and randomly located on the canvas.

Generally speaking, we generated three similar dataset. The main difference was proportion of color in dataset (50%/50%, 75/25, 25/75), for our purposes. Shapes dataset consist of circles, squares, ellipses and rectangles. Number of samples in train subset is 7K and 3K in test subset. Examples you can see on Figure 3.1

(I) Blue rectangle.

(II) Blue ellipse.

(III) Blue square.


(IV) Blue circle.

(V) Red rectangle.

(VI) Red ellipse.

(VII) Red square.

(VIII) Red circle.

FIGURE 3.1: Samples from Figures dataset.

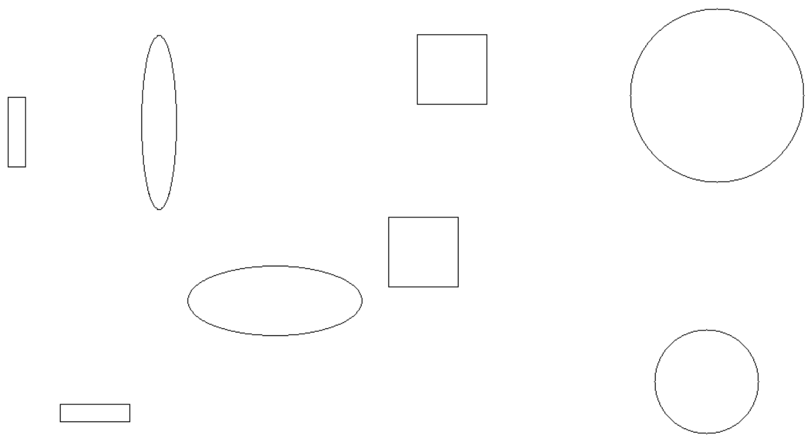


FIGURE 3.2: Samples of edges from Figures dataset.

Also for full pipeline we generated edges for each sample of geometrical figure. Examples you can see on Figure 3.2

3.1.2 T-shirts dataset

This dataset was parsed from Internet resources. I should mention, that dataset was parsed from resources where the policy resource allows it. The idea to create such dataset appeared after hackathon "Hackathon Expert Group" (Odessa, summer 2017), where I with my team got the 1st palace. The idea was in the extraction of cloth from a photo of a person. And change the color of this cloth using HSV format's channels. After this hackathon, I decided to build a pipeline in such way that we can change cloth in another way. In this way, we decided to build a pipeline, which would be proof of concept, that it is possible to build an on-line fitting room.

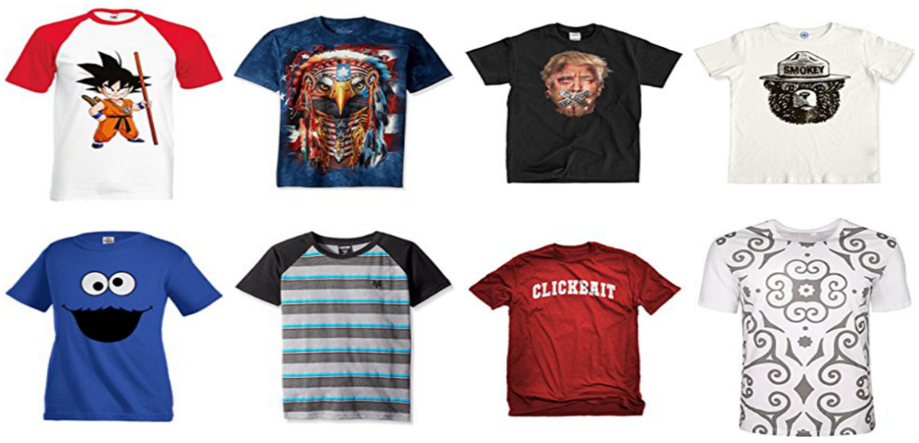


FIGURE 3.3: Samples from T-shirts dataset.

As you can see in Figure 3.3, we have different t-shirts in our dataset. It was important for us to have images with a complex structure on the t-shirt. Such as striped t-shirt, and colorful images on a t-shirt. When we parsed this dataset, we parsed not relevant images too. The main issues we had:

- Some images had several t-shirts on one image;
- Some t-shirts was folded;
- Some images was with as person on it;
- Some images were with watermarks;
- A lot of duplicates;

We did data cleaning by hands. Because we didn't find out any automatic way. I should mention that, we did a lot of work, initially, we had 45K of images after parsing, after data cleaning and preparation we left 13K (without duplicates, etc.). T-shirts dataset was split in train 12K and test 1K.

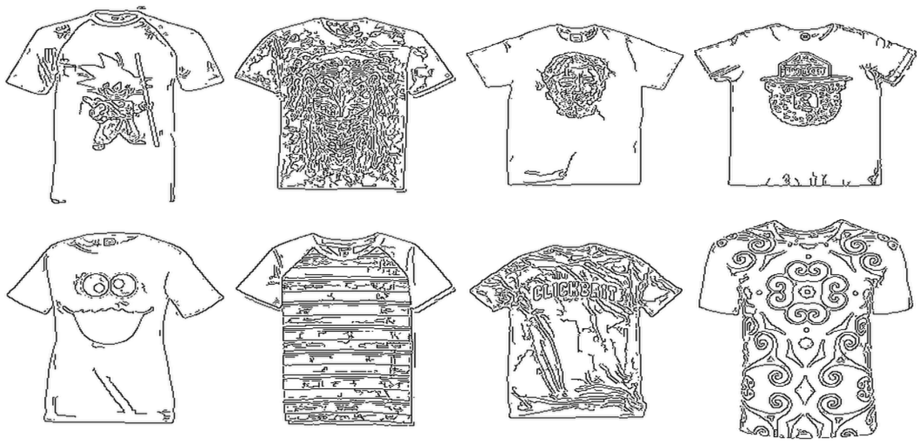


FIGURE 3.4: Samples from T-shirts dataset.

Also for full pipeline we generated edges for each sample of geometrical figure. Examples you can see on Figure 3.4. For generating edges for t-shirts, we used Canny edge detector (Bassil, 2012). All images had different size, some of them was 96x96x3, this scale is small for generating detailed edges. Empirically we decided to run edge detector on original image and then upscale both images. For upscaling an downscaling we used Bicubic interpolation. It helps to save details of edges. Also for us was important to save wrinkles on the t-shirt, because the idea was to generate real t-shirt, which look like person wear it Figure 3.5. All dataset have 256x256x3 size. Generated edges have 256x256x3 size too. Both datasets T-shirt and Synthetic have same size.

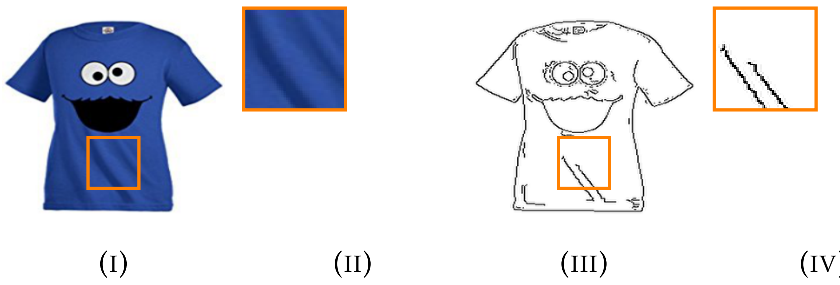


FIGURE 3.5: Wrinkles of t-shirt. It's important to save them.

3.2 Problem Statement

In this work, I highlight two problems of GAN training. But first of all, I want to explain why do we use GAN for our task. A lot of algorithms for image to image translation (e.g. Kingma and Welling, 2013) was developed using functions from L^p space, mostly L_2 norm, for algorithm optimization. Such optimization gives blurry results, and I want to show why (I should mention, that next arguments do not pretend to be strictly mathematical. The following arguments have intuitive character).

It is common situation in vision, using generative modeling, optimizing of L_2 loss function yielding blurry images (Mathieu, Couprie, and LeCun, 2015, Pathak et al., 2016). Let's define L_2 loss as:

$$L = \frac{1}{2M} \sum_{i}^{M} \sum_{j}^{N} (x^{'}_{ij} - x_{ij})^2 = \sum_{i}^{M} ||x^{'}_{ij} - x_{ij}||_2^2 \tag{3.1}$$

where M is number of all samples, $x^{'}$ is output image and x is input image.

Let's take look at Gaussian distribution. It's defined as follows:

$$\mathrm{p}(x|\mu, \sigma^2) = \frac{1}{Z} \exp \frac{||\mu - x||_2^2}{2\sigma^2} \tag{3.2}$$

where μ is the mean, σ^2 is the variance. Let's set $\mu = x^{'}$ and $\sigma^2 = 1$ then:

$$\mathrm{p}(x|x^{'}) \propto \exp \frac{1}{2} ||x^{'} - x||_2^2 \tag{3.3}$$

And apply log to both sides.

$$\log \mathrm{p}(x|x^{'}) \propto \frac{1}{2} ||x^{'} - x||_2^2 \tag{3.4}$$

so we can say that minimizing MSE is the same as maximizing the log-likelihood of Gaussian, so from this prospective we assume that x is from Gaussian distribution.

As we know that Gaussian distribution is unimodal distribution, but images in real world have a lot of "peaks" in their multi-dimensional distribution. For example, if you want to train NN to generate t-shirts, but there a lot of ways to generate t-shirt. So we can say that distribution of all t-shirt images is multimodal. So here we can see a problem, if you try to train model which will fit unimodal distribution to look alike multimodal you will got something like this:

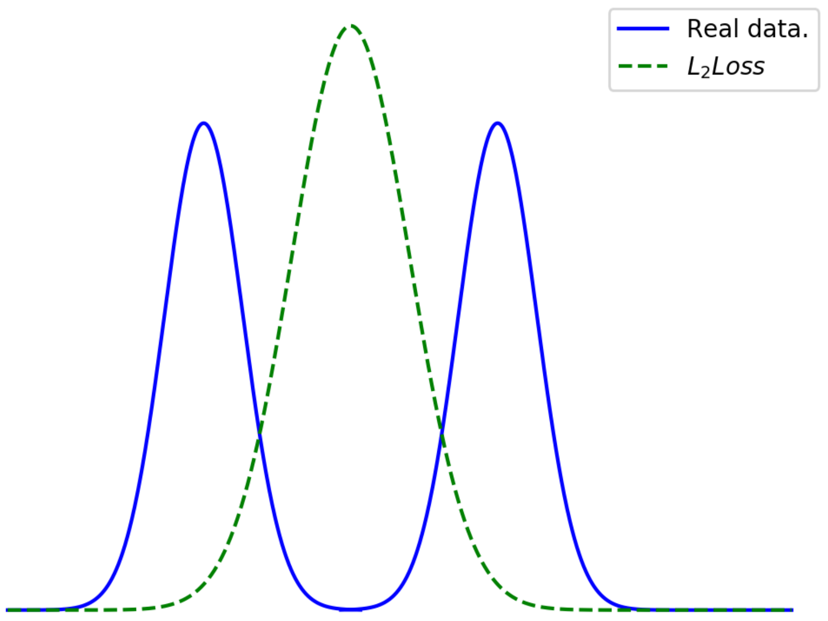


FIGURE 3.6: If we would have bimodal distribution, then optimizing of L_2 loss will finish with green line distribution. Our model will try satisfy both modes.

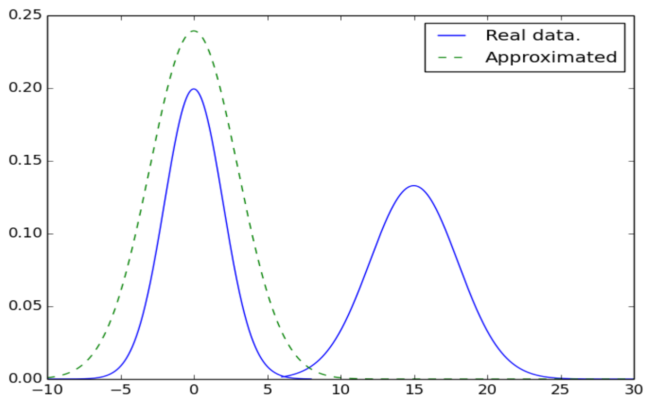
For this reason we use GAN framework based models for our task. The reason for that Jensen-Shannon divergence 2.2 tries to optimize our network in such way that $p_g \sim p_{data}$, in other words approximate an unknown distribution. In section 2.3 we started discussion of "mode collapse" problem let's analyze reason of it. To understand intuition behind "mode collapse" problem, we'll use KL divergence and reverse KL divergence.

Let's introduce KL divergence one more time:

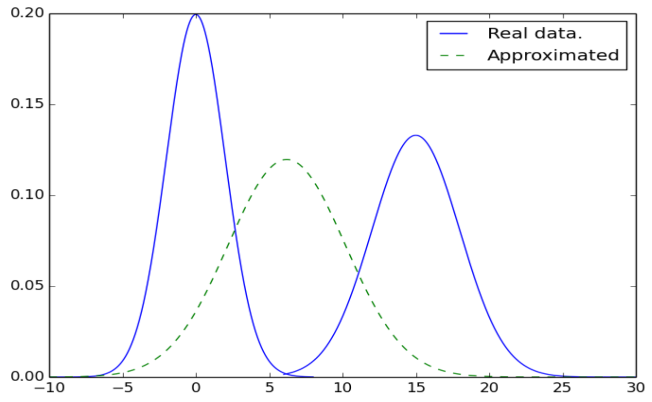
$$\mathrm{KL}(\mathrm{p}_{data}(x)||\mathrm{p}_G(x)) = \sum_{x \in X} \mathrm{p}_{data}(x) \log \frac{\mathrm{p}_{data}(x)}{\mathrm{p}_G(x)} \tag{3.5}$$

that is, for all random variable $x \in X$, KL Divergence calculates the weighted average on the difference between those distributions at x. Just like any other distance functions, we can use KL divergence as a loss function in an optimization setting, especially in a probabilistic setting. However, we have to note this important property about KL divergence: it is not symmetric. Formally, $\mathrm{KL}(\mathrm{p}_{data}(x)||\mathrm{p}_G(x)) \neq \mathrm{KL}(\mathrm{p}_G(x)||\mathrm{p}_{data}(x))$. $(\mathrm{p}_G(x)||\mathrm{p}_{data}(x))$ called reverse divergence.

In forward KL, the difference between $\mathrm{p}_{data}(x)$ and $\mathrm{p}_G(x)$ is weighted by $\mathrm{p}_{data}(x)$. Now let's ponder on that statement for a while. Consider $\mathrm{p}_{data}(x) = 0$ for a particular x. As $\mathrm{p}_{data}(x)$ is the weight, then it doesn't really matter what's the value of the other term. In other words, if $\mathrm{p}_{data}(x) = 0$, there is no consequence at all to have very big difference between $\mathrm{p}_{data}(x)$ and $\mathrm{p}_G(x)$. In this case, the total KL Divergence will not be affected when $\mathrm{p}_{data}(x) = 0$, as the minimum value for KL Divergence is 0. During the optimization process then, whenever $\mathrm{p}_{data}(x) = 0$, $\mathrm{p}_G(x)$ would be ignored. Reversely, if $\mathrm{p}_{data}(x) > 0$, then the $\log \frac{\mathrm{p}_{data}(x)}{\mathrm{p}_G(x)}$ term will contribute to the overall KL Divergence. This is not good if our objective is to minimize KL Divergence. Hence, during the optimization, the difference between $\mathrm{p}_{data}(x)$ and $\mathrm{p}_G(x)$ will be minimized if $\mathrm{p}_{data}(x) > 0$. Let's see some visual examples:



In the example above, the right hand side mode is not covered by $p_G(x)$, but it is obviously the case that $p_{data}(x) > 0$. The consequence for this scenario is that the KL divergence would be big. The optimization algorithm then would force $p_{data}(x)$ to take different form: In the above example, $p_G(x)$ is now more



spread out, covering all $p_{data}(x) > 0$. Now, there is no $p_{data}(x) > 0$ that are not covered by $p_G(x)$.Although there are still some area that are wrongly covered by $p_G(x)$, this is the desired optimization result as in this form of $p_G(x)$, the KL Divergence is low. Those are the reason why, Forward KL is known as zero avoiding, as it is avoiding $p_G(x) = 0$ whenever $p_{data}(x) > 0$. This thoughts gives some intuition of "mode collapse" emergence.

3.3 Experiments

We investigated that conditional GAN is general purpose of image-to-image translation. Based on paper Isola et al., 2016 we have seen that it's works well and can pretend for state-of-the-art approach. So for our task, this approach was starting point. As I mentioned our main task to investigate "mode collapse" problem and try to solve it in terms of fashion industry or understand how this "mode collapse" problem can be controlled for improvement of our task.

3.3.1 Brief overview of pix2pix framework

This subsection is based on Isola et al., 2016 work. Here I will give quick overview of idea their proposed. As we already said, that a naive approach to ask the CNN to minimize Euclidean distance between predicted and ground truth pixels, it will tend to produce blurry results. This is because Euclidean distance is minimized by averaging all plausible outputs, which causes blurring. Coming up with loss functions that force the CNN to do what we really want (e.g., output sharp, realistic images) is an open problem and generally requires expert knowledge. It will be grate to have goal like "produce realistic images", and that is what GAN framework is doing.

Conditional GANs instead learn a structured loss. Structured losses penalize the joint configuration of the output. In our work we tried different combinations of architecture of generator $G_{\mathcal{G}}$ and discriminator $D_{\mathcal{D}}$. For generator we tried architectures build with ResNet blocks (He et al., 2015) of size 6, 9, also we tried to use U-net (Ronneberger, Fischer, and Brox, 2015) like architectures. And for discriminator we used blocks of [convolution, batch-normalization (Ioffe and Szegedy, 2015), leaky relu (Xu et al., 2015)]. In empirical way defined best architecture for us. So let's repeat conditional GAN objective function one more time:

$$\min_{G}\max_{D} V(\theta_G, \theta_D) = \mathbb{E}_{A,y \sim p_{data}}[\log D_{\mathcal{D}}(A, y)] + \mathbb{E}_{z \sim p_z, y \sim p_y}[\log(1 - D_{\mathcal{D}}(G_{\mathcal{G}}(z, y), y))] \tag{3.6}$$

In case of image to image translation we can formulate in such image: In case

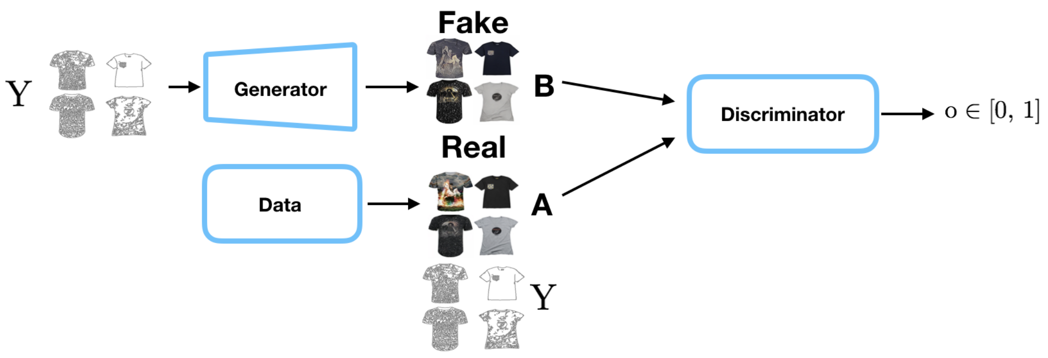


FIGURE 3.7: The main idea of pix2pix framework.

of image to image translation on place of Y can be any valid input, such as in this works inpainting (Pathak et al., 2016), future state prediction (Zhou and Berg, 2016), image manipulation guided by user constraints (Zhu et al., 2016), style transfer (Li and Wand, 2016), and superresolution (Ledig et al., 2016). All this applications of conditional GAN was proposed before pix2pix framework (Isola et al., 2016). But full full pipeline was proposed in this work.

Previous approaches have found it beneficial to mix the GAN objective with a more traditional loss, such as L_2 distance (Pathak et al., 2016). The discriminator's job remains unchanged, but the generator is tasked to not only fool the discriminator but also to be near the ground truth output in an L_2 sense. We also explore this option, using L_1 distance rather than L_2 as L_1 encourages less blurring.

$$L_1 = \mathbb{E}_{A,Y}[||A - G_{\mathcal{G}}(Y)||_1] \tag{3.7}$$

So full loss will look like:

$$\min_{G} \max_{D} V(\theta_G, \theta_D) + \lambda L_1(G) \tag{3.8}$$

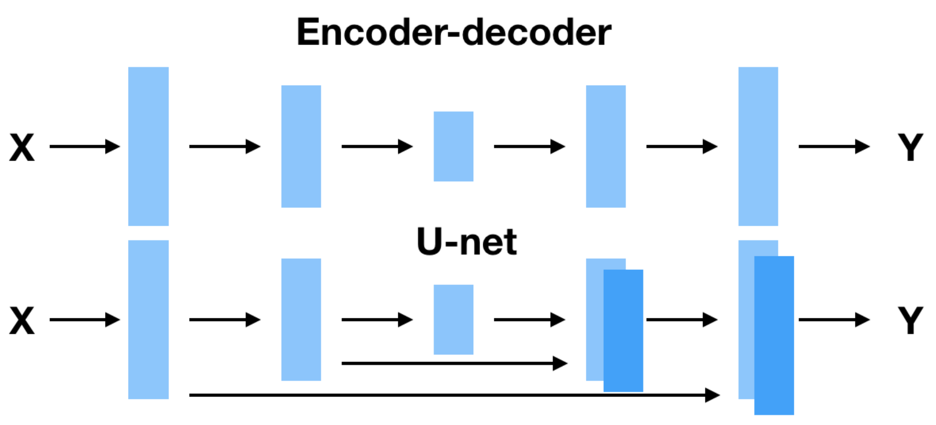


FIGURE 3.8: Simplified encoder-decoder based, U-net architectures.

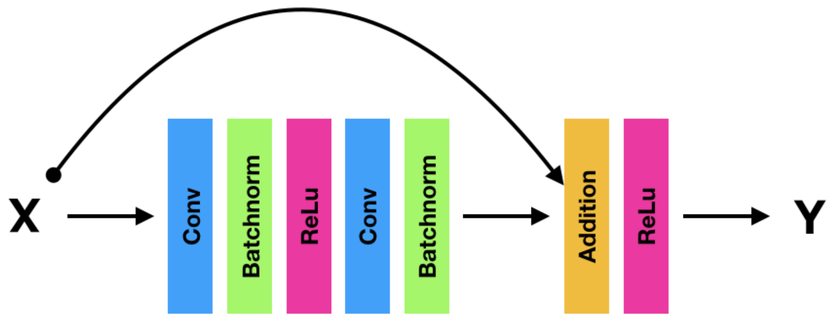


FIGURE 3.9: ResNet block.

For generator we had two pretenders. On Figure 3.8 we can see the simplified idea of encoder-decoder (Hinton and Salakhutdinov, 2006) and U-net (Ronneberger, Fischer, and Brox, 2015) networks. Many last approaches (Pathak et al., 2016, Wang and Gupta, 2016, Zhou and Berg, 2016, Yoo et al., 2016) are based on encoder-decoder architecture. Such architecture run image through bunch of layers which decreasing it to some small latent representation and then network do reverse process and recover information from latent space.

But Ronneberger, Fischer, and Brox, 2015 presented modification of encoder-decoder architecture. This architecture is better for image-to-image translation, because we can now transport some "low-level" information about image to output layers of "high-level" representation. This approach was proposed in Isola et al., 2016 paper. Analyzing this arguments we can propose another architecture that using skip connection for transferring information (in parallel with main network) and combine it. It's nets with stacked ResNet blocks (He et al., 2015). We tried both in our work, and describe full architecture where it's worked.

For discriminator $D_{\mathcal{D}}$ we have used just several stacked blocks of [convolution, batch-normalization (Ioffe and Szegedy, 2015), leaky relu (Xu et al., 2015)]

3.3.2 Results

In first experiment we have used synthetic dataset 3.1.1 to show "mode collapse" from experiment prospective. As we said in 3.1.1 we generated dataset of geometric shapes of two colors (blue and red) on white background. Also we generated corresponding edges of geometric shapes. Number of blue and red shapes will be 50X50. So the full pipeline looked like this:

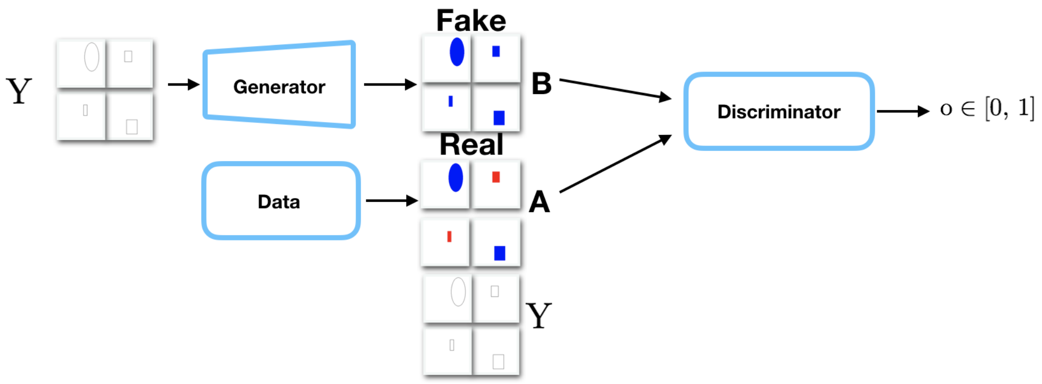


FIGURE 3.10: pix2pix training on Synthetic dataset 3.1.1.

In this experiment we saw "mode collapse" problem implicitly. If we initialize both generator $G_{\mathcal{G}}$ and discriminator $D_{\mathcal{D}}$ several times and train networks, then pix2pix model will learn only one "mode" of distribution of colors. Each time model will learn how to generate inside edges some color, but each time it'll learn to generate only one color, blue or red, depends on random initialization of networks.

On table 3.1 we can see results of one such training on test data. As we said, if you run experiment one more time, you will got reverse situation, model will generate only blue examples.

On table 3.2 we can see results of training same model but on synthetic dataset 3.1.1 but with proportion of blue/red color 75x25. As we can see model learned to generate only blue shapes into edges. So model learned only one "mode" of color distribution.

This behavior is not a surprise since we optimize 2.7 Jenson-Shannon divergence. As we discussed in 3.2 JS divergence is "mode-seeking" if concentrated distributions whose support may not overlap. So we can assume that in some

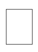
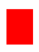
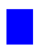
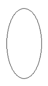
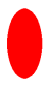
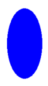
Edges	Generated	Ground truth
		
		

TABLE 3.1: Example of results by the training pix2pix on synthetic dataset. In this case network was initialized in such way, that it learned to generate only red shapes into edges.

high dimensional space of blue/red shapes are not overlapping in condition of color.

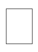
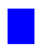
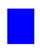
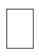
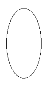
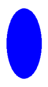
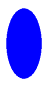
Edges	Generated	Ground truth
		
		
		

TABLE 3.2: Example of results by the training pix2pix on synthetic dataset 75x50. In this case network was random initialized in such way, that it learned to generate only blue shapes into edges.

All models was trained using U-net 3.8 based for generator $G_{\mathcal{G}}$, and for discriminator $D_{\mathcal{D}}$ with three blocks stacked blocks of convolution, batch normalization, and ReLu activation.

The next part of experiment was to apply pix2pix model for T-shirts dataset 3.1.2. As we said pix2pix model's loss consist of two losses conditional GAN loss and L_1 loss. For generator $G_{\mathcal{G}}$ we have used U-net 3.8 based networks of different number of layers, stacked ResNet blocks 3.9 with 6 and 9 blocks. But for our pix2pix model generator $G_{\mathcal{G}}$ with ResNet blocks didn't work well, so we present result of U-net based model.

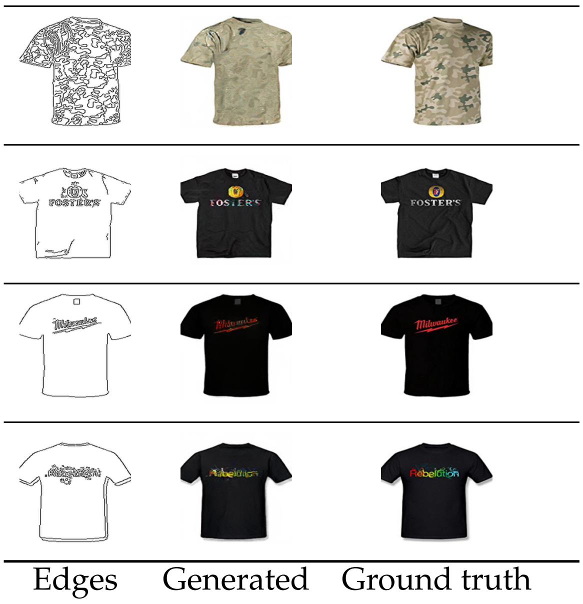


TABLE 3.3: Example of results by the training pix2pix on T-shirts dataset. This is cases of good generated t-shirts. It's defiantly looking realistic.

On table 3.3 we can see goof results of pix2pix model for generating t-shirts using edges as input. We can see that network generating t-shirt directly inside edges, it can generate complex edges of text. Also model can generate wrinkles on t-shirt, you can see it on Figure 3.11. So all our requirements are satisfied. But in our experiments we did a lot of manual work, and checking. The reason for that is no any metric which will say if generated images are looking realistic or not. Such methods as SSIM (Wang et al., 2004) and PSNR (Hore and Ziou, 2010) don't fit for such task, as they show improvements when we optimizing

L_2 loss. And we don't want generated samples by generator $G_{\mathcal{G}}$ look directly like real one.

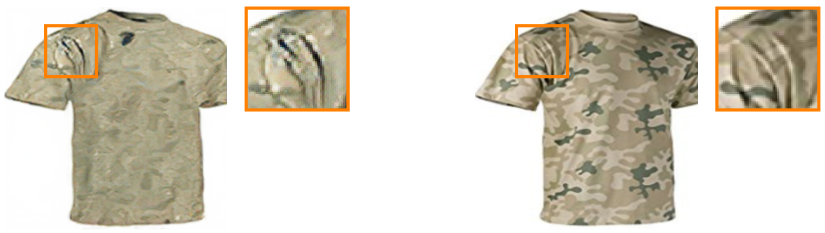


FIGURE 3.11: Wrinkles of t-shirt. It was important to save them.

It's obvious that most t-shirts has some main color and small pattern on it. Such manual approach helped us to see that our trained model learned how to generate simple t-shirts with short color pallet. But in case of complicated t-shirt with a lot of colors and figures on it, generator generates t-shirt with one color pallet. Examples you can see on table 3.4 Actually, we don't give any infor-

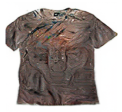
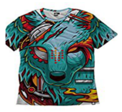
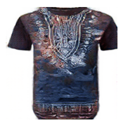
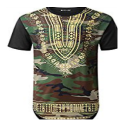
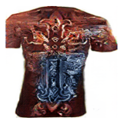
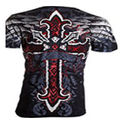
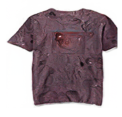
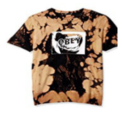
Edges	Generated	Ground truth
		
		
		
		

TABLE 3.4: Example of results by the training pix2pix on T-shirts dataset. This is cases of bad generated t-shirts. This what we called mode collapse, when generator learned to fool discriminator by generating the same color pallet.

mation for generator about color. But current model have other disadvantages, our T-shirts dataset 3.1.2 initially had different size parsed images. 5% of it had images of size 96x96x3, we had to upscale it to 256x256x3, we used bicubic interpolation which produce blurry results. Also we had small amount of images with striped pattern. And our model produce very bad output on this samples. Examples you can see on table 3.5.

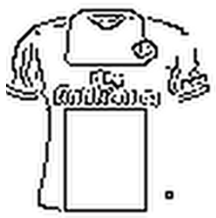
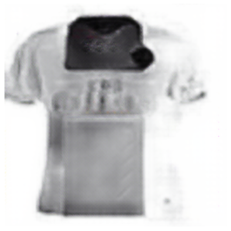
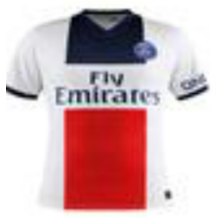
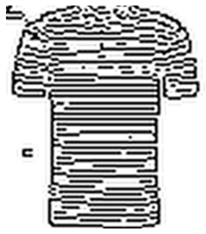
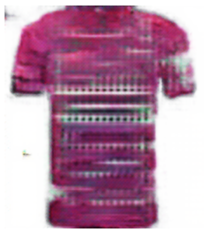
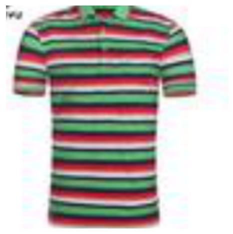
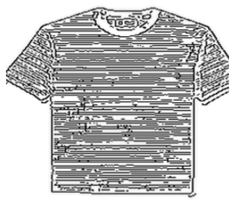
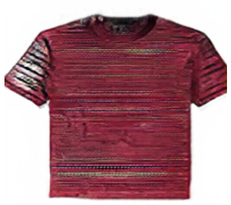
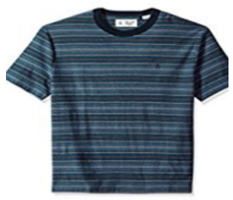
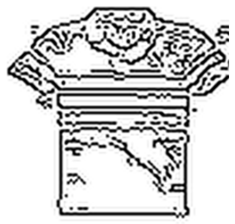
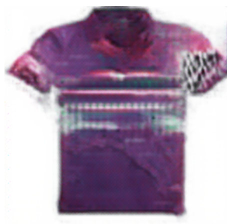
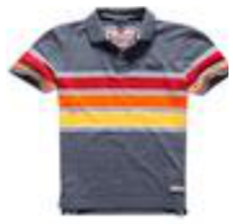
Edges	Generated	Ground truth
		
		
		
		

TABLE 3.5: Example of results by the training pix2pix on T-shirts dataset. This is cases of bad generated t-shirts. This what we called mode collapse, when generator didn't learn to generate some samples which not similar to other.

So we can see that pix2pix model produce very good results, but "mode collapse" exclude some corner cases, such as t-shirts with striped pattern. So in next subsection, we propose our approach to learn several modes of t-shirt or control generation from some point of view.

3.3.3 The proposed boosting-like method

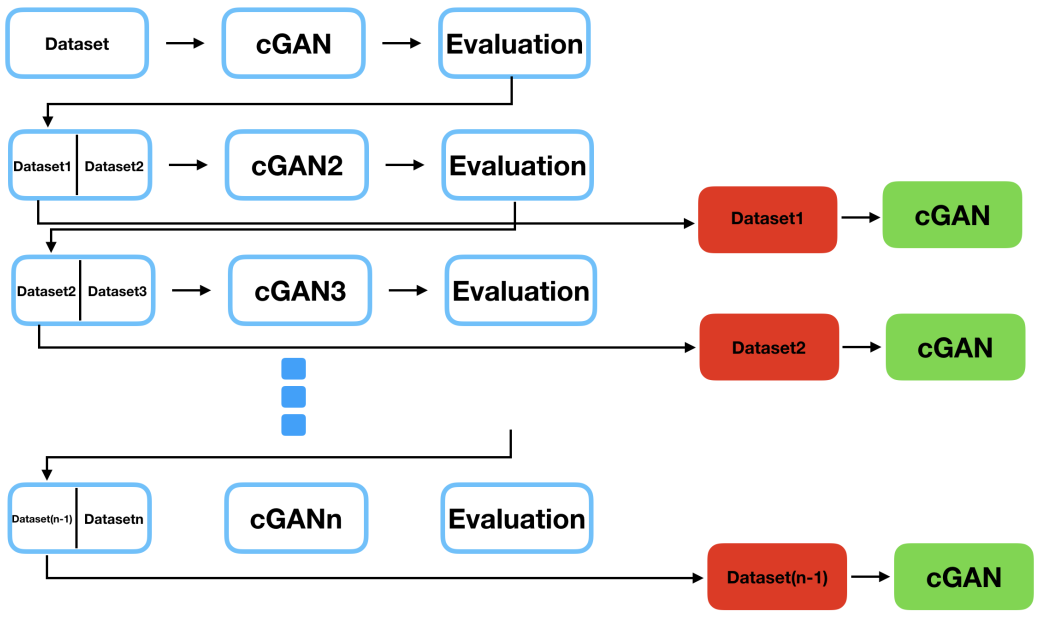


FIGURE 3.12: Our proposed pipeline for creating model which will cover more modes.

In this subsection, we present our approach for training image-to-image translation for cases, when we have multi-modal distribution and we need to generate images with more diversity.We will use the same pix2pix architecture as in previous subsection 3.3.2. In this case, we will manipulate with a dataset and add some process we called evaluation. Scheme of such approach you can see in Figure 3.12. We will explain our approach to an example of T-shirts dataset 3.1.2.

This algorithm is iterative, at first iteration we have the full dataset of T-shirts. We train pix2pix model on this dataset and generating images on all training data edges of t-shirts. Then we take ground truth t-shirt, generated and calculating feature reconstruction loss (Johnson, Alahi, and Fei-Fei, 2016). Rather than comparing the pixels of the output image B $= G(y)$ to exactly match the pixels of the target image A, we instead encourage them to have similar feature representations as computed by the loss network ϕ. Let $\phi_j(x)$ be the activations of the jth layer of the network ϕ when processing the image x; if j is a convolutional layer then $\phi_j(x)$ will be a feature map of shape $C_j X H_j X W_j$. The feature reconstruction loss is the (squared, normalized) Euclidean distance between feature representations:

$$l_{feat}^{\phi,j}(B,A) = \frac{1}{C_j W_j H_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} ||\hat{\phi}(G(y)) - \phi(A)||_2^2 \tag{3.9}$$

Where ϕ in our case is ResNet50, to calculate such loss we took outputs of second ResNet block 3.9. After we calculate loss for all dataset, we splitting it into two parts - samples which have an error less than median of losses on all dataset, and more than median. You can see the process on the first iteration when we took a dataset of T-shirts 3.1.2 and trained GAN as before and calculated feature reconstruction loss for the training set. And then split train dataset into two parts - dataset1 and iteration one dataset. I should say that there are not any strict mathematic reasons to do it in such way, but empirically it showed interesting observations and resolved our problems we formulated in the beginning. Partition operation splited T-shirts dataset 3.1.2 in proportion 3843/7545 dataset1 and iteration one dataset respectively.

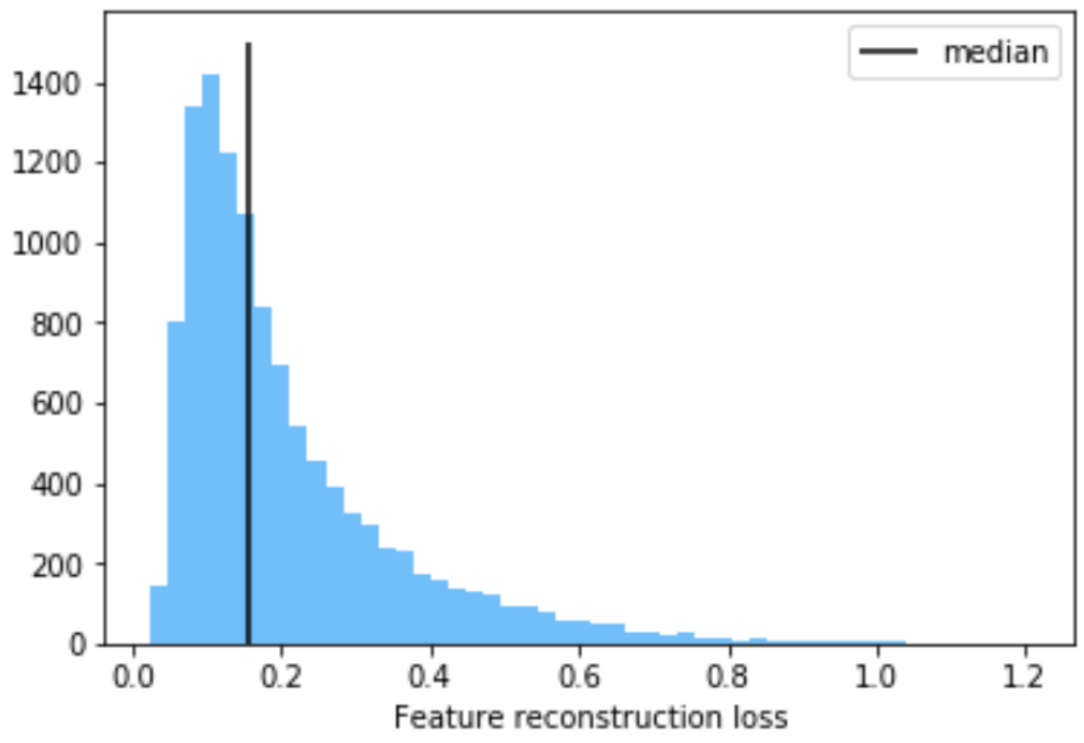


FIGURE 3.13: First iteration of our algorithm. We calculated loss on all dataset and split it on two parts using median.

After the first iteration is done, we take iteration one dataset and repeat the process. We train new pix2pix model on iteration one dataset, then we calculate loss on all iteration one dataset with generated images with the new model. And make partition by the median again. In our case it looks like Figure 3.14. So now we will have two more datasets partitioned on iteration one dataset

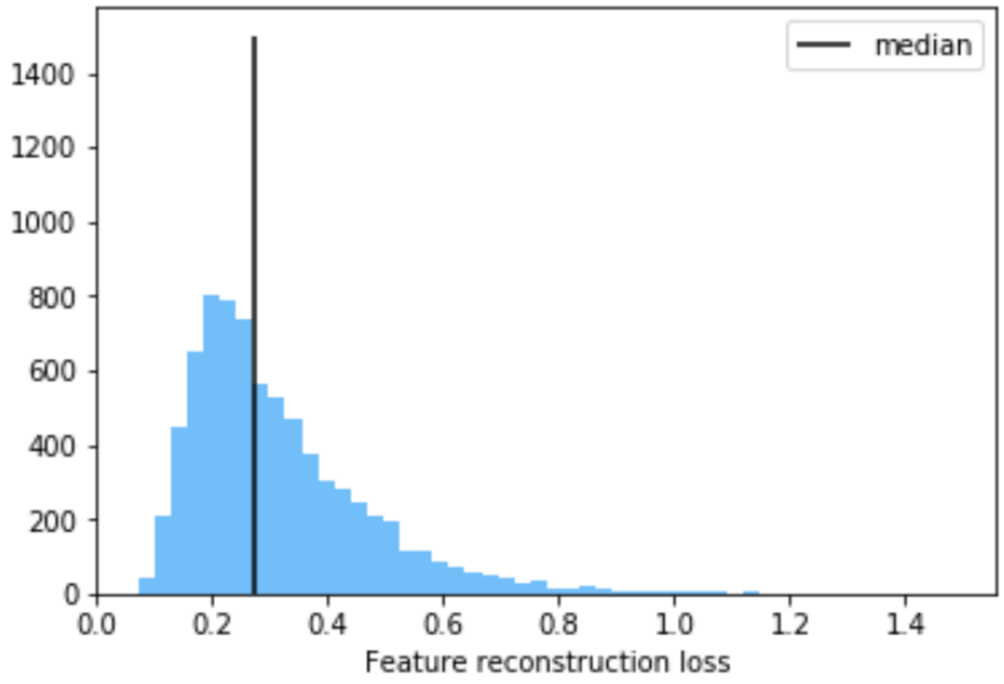


FIGURE 3.14: Second iteration of our algorithm. We calculated loss on all dataset and split it on two parts using median.

- dataset2 and iteration two datasets. But before next iteration let's analyze what we got through training two pix2pix models. So on table 3.6 we can see that most of the samples, which we pick out from an evaluation of first pix2pix model, got better results in terms of feature reconstruction loss. Partition operation split iteration one dataset in proportion 3772/3772 dataset2 and iteration three dataset respectively.

TABLE 3.6: Iteration one dataset after training second pix2pix model.

Iteration one dataset	**Error got better**	**Error got worse**
7545	5216	2329

After iteration two is done, we can train new GAN on iteration two dataset. And in our case, we call it final dataset3 and stop iteration process. Because in our case such partition as we did make for us 3 datasets in proportion 3843/3772/3773, dataset1, dataset2, dataset3 respectively and it's the minimum amount of data to train pix2pix model if we do not want change generator $G_{\mathcal{G}}$ and discriminator $D_{\mathcal{D}}$ architectures at the next step. The next step in our proposed method 3.12 is to take datasets produced at each step, in our case, it's dataset1, dataset2, dataset3 and train for each of them own pix2pix model. In such way, we believe that at each iteration, according to our optimization function, will cover their own mode of distribution. In such way, we will get 2 models, each of them are covering their own mode from p_{data}.

So let's call our new models as pix2pix1, pix2pix2, pix2pix3, for dataset1, dataset2, dataset3 respectively. As we mentioned in previous subsection 3.5 our initial model trained on full T-shirts dataset (3.1.2) is suffering from mode collapse in the way it's didn't learn some rare samples with similar patterns such as striped pattern and bad quality images.

As we see in table 3.7, our method for bad cases of pix2pix model trained on full T-shirts dataset, generates 3 different images by 3 different models. And we can see that some models built using our approach generating better results. That example proofs our assumption that in such way we can build the bunch of models which will cover more modes of data distribution p_{data}. Of course, we have examples of images that none of the models generate proper results, but in such cases, as a striped pattern and upscaled images, our model is able to solve. Manually we checked dataset1, dataset2, dataset3 and visually we can see some dependents of partition.

According to our visual assessment in the first dataset, we got images from T-shirt dataset 3.1.2 which have some main plane color and small figure on it. This is expected, because dataset was parsed from real commercial web resources, and it's obvious that most close has limited color pallet. In dataset2 we got t-shirts which have more colors, not standard colors (e.g. gray, white, black) and complicated figures on it. The most interesting for us was dataset3 as far as images from this dataset showed big error on both iterations. And we have seen

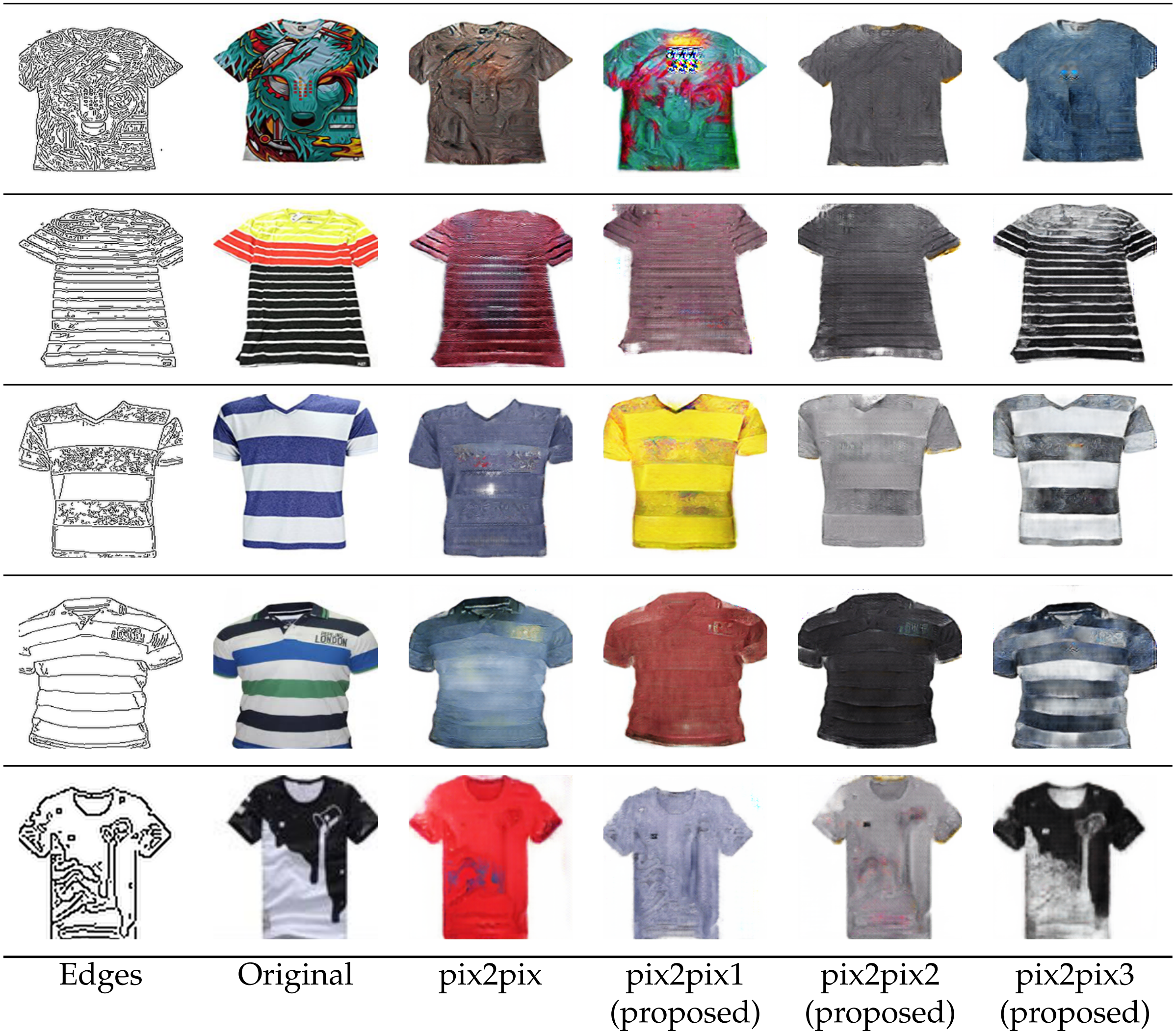


TABLE 3.7: Example of results of our approach in comparison with initial pix2pix model and ground truth.

that in dataset3 we mostly have images of bad quality, which were upscaled from 96X96X3 to 256X256X3, and images with a striped pattern on it. I don't know why but for networks, it was complicated such pattern. But as we see on table 3.7 such images succeed with a pix2pix3 model. So we can say that using our approach we can train such model which will succeed with such corner cases, and I should mention that we don't need any manual work for that.

The second task for us was to build such pipeline, that for one input we could generate images with the different color pallet. Under different color pallet, we mean that we pursued a goal to build a pipeline in a way that it outputs realistic images for the same condition, but it's not similar to each other, and to the ground truth. But, and in this requirement, we got some satisfying results. And this is important for the main goal an application for the fashion industry. In other words, we can say, that we increased the variety of image-to-image translation task. Example you can see on table 3.8

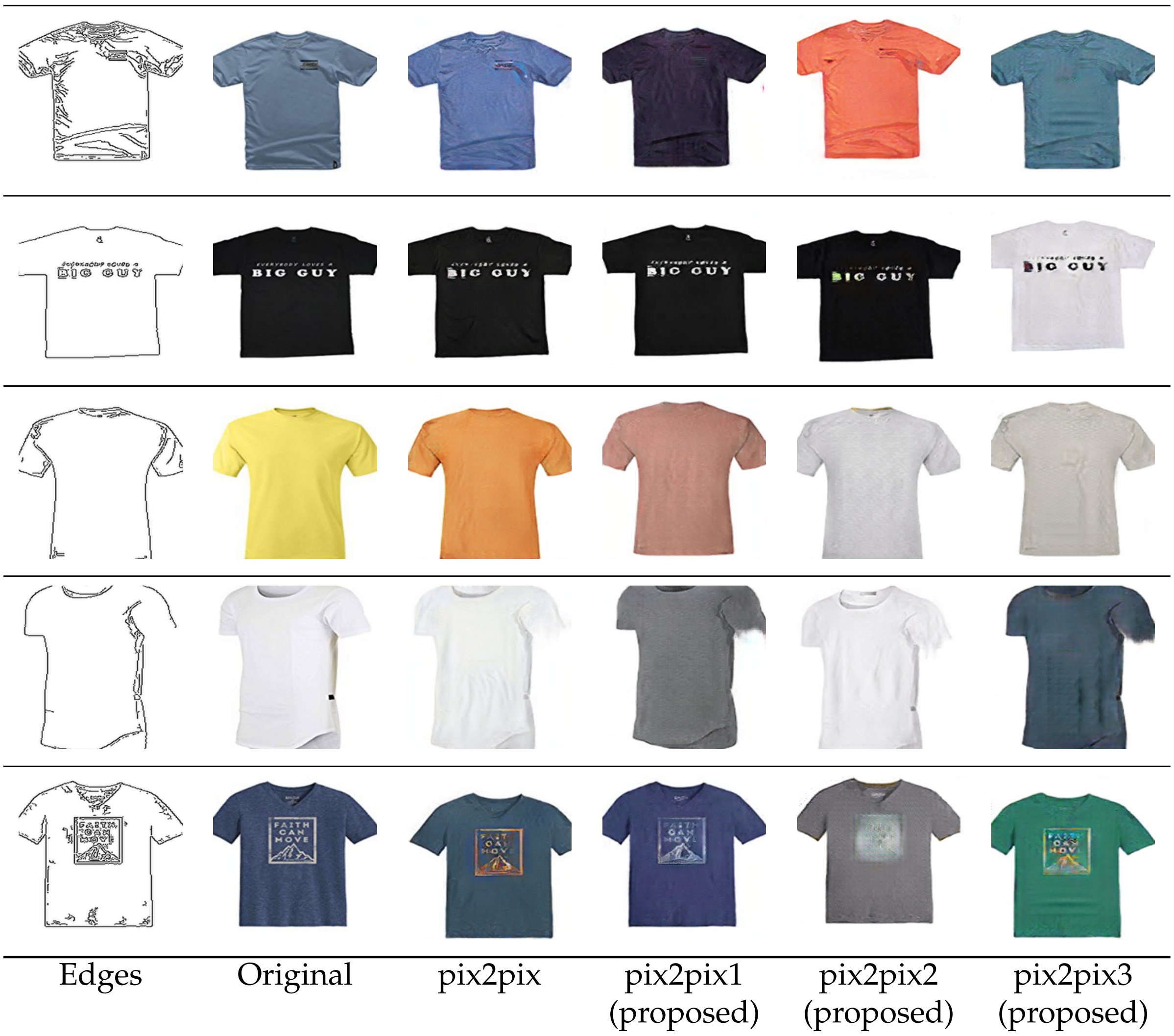


TABLE 3.8: Example of results of our approach in comparison with initial pix2pix model and ground truth.

Let's more formally defined our proposed algorithm:

Algorithm 1 Our proposed algorithm to construct a “strong” set of K individual GANs, trained sequentially.

```
Require: Dataset D = $X_1$, $X_2$, ... $X_N$, Y = $y_1$, $y_2$, ... $y_N$
  $DATA\_PARTITIONS \leftarrow [\quad]$
  $D_p \leftarrow D$
  while $K \neq 0$ do
    $GAN \leftarrow pix2pix.intialize()$
    $GAN.train(D_p)$
    $dataset_k, D_p \leftarrow SPLIT[l_{feat}(GAN.predict(D_p), Y_k), median(GAN.predict(D_p)]$

    $DATA\_PARTITIONS.append(dataset_k)$
  end while
  $GANs \leftarrow set()$
  for data in DATA_PARTITIONS do
    $GAN \leftarrow pix2pix.intialize()$
    $GAN.train(datas)$
    $GANs.append(GAN)$
  end for
  return $GANs$
```

Chapter 4

Conclusions

In this thesis, we made an overview of image-to-image translation problem from the generative adversarial networks point of you. We did a detailed overview of GAN framework introduced by Goodfellow et al., 2014, also in chapter 2 we made an overview of GAN based extensions. We discussed problems of this methods and looked at different solutions developed at that time. We covered such extension of GAN as GAN framework, conditional GAN, Wasserstein GAN, f-divergence GAN. Also, we gave some overview of generative modeling generally from an image-to-image translation problem.

At chapter 3, we made detailed overview of current state-of-the-art solution - pix2pix. For experiments with the current state-of-the-art model, we created two datasets. One of them is synthetically generated it consist of different geometric shapes on white background, and the second one was parsed from Internet resources, where it's allowed, in Dataset section 3.1 we gave a detailed overview of data cleaning, preprocessing. And explained why the creation of dataset need a lot of manual work and why it should be clean. We should mention that there is no such T-shirts dataset 3.1.2 presented yet, as we know, and we give interesting tips and tricks for dataset creation for image-to-image translation task.

Using pix2pix (Isola et al., 2016) model with synthetic dataset 3.1.1 we discovered that conditional GAN application for image-to-image translation suffering from "mode collapse" too. Our synthetic dataset 3.1.1 showed on a more low-level dataset and made this problem explicit. This fact pushed us on some idea, that resulted novel approach for image-to-image translation based on the pix2pix model. It was very useful for variety problem-solving.

We developed a conditional GAN boosting-like algorithm, which solves in a special way "mode collapse" problem. And allows us to generate for one input several outputs with different "style". Also, we proposed our own solution for "mode collapse" problem. Our algorithm outputs set of different GANs, each of them covers different modes of distribution. For image-to-image translation where distribution is very high-dimensional and have a lot of "peaks", this method solves a problem well, but if we talk in terms of fashion industry it works very well. Now we can generate several images for one input, because we learned several modes of dataset distribution, in other words, we solved the problem of variety in some sense.

Despite this, there are a lot of research can be done in this area, not only for image-to-image translation but in general in direction of complex distribution approximation. Generally, We can make an important conclusion that it's very useful to start research from simple points, as we did with a synthetic dataset. This experiment allowed us to see real problems of current approaches, and develop our own approach.

I want to say thank you to my supervisor Artem Chernodub, who directed me in my research and helped me to highlight important problems and the ways of solving it.

Bibliography

Arjovsky, M., S. Chintala, and L. Bottou (2017). “Wasserstein GAN”. In: *ArXiv e-prints*. arXiv: 1701.07875 [stat.ML].

Arora, S. and Y. Zhang (2017). “Do GANs actually learn the distribution? An empirical study”. In: *ArXiv e-prints*. arXiv: 1706.08224 [cs.LG].

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2017). “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Bassil, Y. (2012). “Image Steganography based on a Parameterized Canny Edge Detection Algorithm”. In: *ArXiv e-prints*. arXiv: 1212.6259 [cs.CR].

Decelle, A., G. Fissore, and C. Furtlehner (2017). “Spectral Learning of Restricted Boltzmann Machines”. In: *ArXiv e-prints*. arXiv: 1708.02917 [cond-mat.dis-nn].

Divakar, N. and R. Venkatesh Babu (2017). “Image Denoising via CNNs: An Adversarial Approach”. In: *ArXiv e-prints*. arXiv: 1708.00159 [cs.CV].

Dong, X. et al. (2015). “Ground-truth dataset and baseline evaluations for image base-detail separation algorithms”. In: *ArXiv e-prints*. arXiv: 1511.06830 [cs.CV].

Ghosh, A. et al. (2016). “Contextual RNN-GANs for Abstract Reasoning Diagram Generation”. In: *ArXiv e-prints*. arXiv: 1609.09444 [cs.CV].

Goodfellow, I. J. (2014). “On distinguishability criteria for estimating generative models”. In: *ArXiv e-prints*. arXiv: 1412.6515 [stat.ML].

Goodfellow, I. J. et al. (2014). “Generative Adversarial Networks”. In: *ArXiv e-prints*. arXiv: 1406.2661 [stat.ML].

Gulrajani, I. et al. (2017). “Improved Training of Wasserstein GANs”. In: *ArXiv e-prints*. arXiv: 1704.00028 [cs.LG].

He, K. et al. (2015). “Deep Residual Learning for Image Recognition”. In: *ArXiv e-prints*. arXiv: 1512.03385 [cs.CV].

Hinton, Geoffrey and Ruslan Salakhutdinov (2006). “Reducing the Dimensionality of Data with Neural Networks”. In: *Science* 313.5786, pp. 504 –507.

Ho, J. and S. Ermon (2016). "Generative Adversarial Imitation Learning". In: *ArXiv e-prints*. arXiv: 1606.03476 [cs.LG].

Hore, Alain and Djemel Ziou (2010). "Image Quality Metrics: PSNR vs. SSIM". In: *Proceedings of the 2010 20th International Conference on Pattern Recognition*. ICPR '10. Washington, DC, USA: IEEE Computer Society, pp. 2366–2369. ISBN: 978-0-7695-4109-9. DOI: 10.1109/ICPR.2010.579. URL: http://dx.doi.org/10.1109/ICPR.2010.579.

Ioffe, S. and C. Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *ArXiv e-prints*. arXiv: 1502.03167 [cs.LG].

Isola, P. et al. (2016). "Image-to-Image Translation with Conditional Adversarial Networks". In: *ArXiv e-prints*. arXiv: 1611.07004 [cs.CV].

Johnson, J., A. Alahi, and L. Fei-Fei (2016). "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *ArXiv e-prints*. arXiv: 1603.08155 [cs.CV].

Kaae Sønderby, C. et al. (2016). "Amortised MAP Inference for Image Super-resolution". In: *ArXiv e-prints*. arXiv: 1610.04490 [cs.CV].

Kingma, D. P and M. Welling (2013). "Auto-Encoding Variational Bayes". In: *ArXiv e-prints*. arXiv: 1312.6114 [stat.ML].

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Kulkarni, T. D. et al. (2015). "Deep Convolutional Inverse Graphics Network". In: *ArXiv e-prints*. arXiv: 1503.03167 [cs.CV].

Kullback, S. and R. A. Leibler (1951). "On Information and Sufficiency". In: *Ann. Math. Statist.* 22.1, pp. 79–86. DOI: 10.1214/aoms/1177729694. URL: https://doi.org/10.1214/aoms/1177729694.

Kupyn, O. et al. (2017). "DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks". In: *ArXiv e-prints*. arXiv: 1711.07064 [cs.CV].

LeCun, Yann and Corinna Cortes (2010). "MNIST handwritten digit database". In: URL: http://yann.lecun.com/exdb/mnist/.

Ledig, C. et al. (2016). "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *ArXiv e-prints*. arXiv: 1609.04802 [cs.CV].

Li, C. and M. Wand (2016). “Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks”. In: *ArXiv e-prints*. arXiv: 1604.04382 [cs.CV].

Liu, Y. et al. (2017). “Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks”. In: *ArXiv e-prints*. arXiv: 1705.01908 [cs.CV].

Long, J., E. Shelhamer, and T. Darrell (2014). “Fully Convolutional Networks for Semantic Segmentation”. In: *ArXiv e-prints*. arXiv: 1411.4038 [cs.CV].

Mansimov, E. et al. (2015). “Generating Images from Captions with Attention”. In: *ArXiv e-prints*. arXiv: 1511.02793 [cs.LG].

Mathieu, M., C. Couprie, and Y. LeCun (2015). “Deep multi-scale video prediction beyond mean square error”. In: *ArXiv e-prints*. arXiv: 1511.05440 [cs.LG].

Mirza, M. and S. Osindero (2014). “Conditional Generative Adversarial Nets”. In: *ArXiv e-prints*. arXiv: 1411.1784 [cs.LG].

Mnih, A. and K. Gregor (2014). “Neural Variational Inference and Learning in Belief Networks”. In: *ArXiv e-prints*. arXiv: 1402.0030 [cs.LG].

Nowozin, S., B. Cseke, and R. Tomioka (2016). “f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization”. In: *ArXiv e-prints*. arXiv: 1606.00709 [stat.ML].

Pathak, D. et al. (2016). “Context Encoders: Feature Learning by Inpainting”. In: *ArXiv e-prints*. arXiv: 1604.07379 [cs.CV].

Perarnau, G. et al. (2016). “Invertible Conditional GANs for image editing”. In: *ArXiv e-prints*. arXiv: 1611.06355 [cs.CV].

Raad, L. et al. (2017). “A survey of exemplar-based texture synthesis”. In: *ArXiv e-prints*. arXiv: 1707.07184 [cs.CV].

Radford, A., L. Metz, and S. Chintala (2015). “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *ArXiv e-prints*. arXiv: 1511.06434 [cs.LG].

Rajeswar, S. et al. (2017). “Adversarial Generation of Natural Language”. In: *ArXiv e-prints*. arXiv: 1705.10929 [cs.CL].

Ronneberger, O., P. Fischer, and T. Brox (2015). “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *ArXiv e-prints*. arXiv: 1505.04597 [cs.CV].

Russakovsky, O. et al. (2014). “ImageNet Large Scale Visual Recognition Challenge”. In: *ArXiv e-prints*. arXiv: 1409.0575 [cs.CV].

Salimans, T. et al. (2016). "Improved Techniques for Training GANs". In: *ArXiv e-prints*. arXiv: 1606.03498 [cs.LG].

Santurkar, S., D. Budden, and N. Shavit (2017). "Generative Compression". In: *ArXiv e-prints*. arXiv: 1703.01467 [cs.CV].

Simonyan, K. and A. Zisserman (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *ArXiv e-prints*. arXiv: 1409.1556 [cs.CV].

Srivastava, N., R. R Salakhutdinov, and G. E. Hinton (2013). "Modeling Documents with Deep Boltzmann Machines". In: *ArXiv e-prints*. arXiv: 1309.6865 [cs.LG].

Theis, L., A. van den Oord, and M. Bethge (2015). "A note on the evaluation of generative models". In: *ArXiv e-prints*. arXiv: 1511.01844 [stat.ML].

Tolstikhin, I. et al. (2017). "AdaGAN: Boosting Generative Models". In: *ArXiv e-prints*. arXiv: 1701.02386 [stat.ML].

Turner, J. et al. (2017). "Deep Belief Networks used on High Resolution Multichannel Electroencephalography Data for Seizure Detection". In: *ArXiv e-prints*. arXiv: 1708.08430 [cs.CV].

van den Oord, A., N. Kalchbrenner, and K. Kavukcuoglu (2016). "Pixel Recurrent Neural Networks". In: *ArXiv e-prints*. arXiv: 1601.06759 [cs.CV].

van den Oord, A. et al. (2016). "Conditional Image Generation with PixelCNN Decoders". In: *ArXiv e-prints*. arXiv: 1606.05328 [cs.CV].

Villani, C. (2008). *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg. ISBN: 9783540710509. URL: https://books.google.com.ua/books?id=hV8o5R7_5tkC.

Wang, X. and A. Gupta (2016). "Generative Image Modeling using Style and Structure Adversarial Networks". In: *ArXiv e-prints*. arXiv: 1603.05631 [cs.CV].

Wang, Zhou et al. (2004). "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *Trans. Img. Proc.* 13.4, pp. 600–612. ISSN: 1057-7149. DOI: 10.1109/TIP.2003.819861. URL: http://dx.doi.org/10.1109/TIP.2003.819861.

Xu, B. et al. (2015). "Empirical Evaluation of Rectified Activations in Convolutional Network". In: *ArXiv e-prints*. arXiv: 1505.00853 [cs.LG].

Yeh, R. A. et al. (2016). "Semantic Image Inpainting with Deep Generative Models". In: *ArXiv e-prints*. arXiv: 1607.07539 [cs.CV].

Yoo, D. et al. (2016). "Pixel-Level Domain Transfer". In: *ArXiv e-prints*. arXiv: 1603.07442 [cs.CV].

Zhou, Y. and T. L. Berg (2016). "Learning Temporal Transformations From Time-Lapse Videos". In: *ArXiv e-prints*. arXiv: 1608.07724 [cs.CV].

Zhu, J.-Y. et al. (2017). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *ArXiv e-prints*. arXiv: `1703.10593 [cs.CV]`.

Zhu, Jun-Yan et al. (2016). "Generative Visual Manipulation on the Natural Image Manifold". In: *Proceedings of European Conference on Computer Vision (ECCV)*.