Sections and Chapters

Gubert Farnsworth

Contents

1 Introduction

2 Background

2.1 Convolutional Neural Networks

Convolutional Neural Networks, also known as ConvNets or CNNs, are the state-of-the-art neural networks for most computer vision tasks, such as image classification and object detection or face recognition.

Unlike regular feed-forward neural networks based on fully-connected layers of neurons, ConvNets are trained to learn relatively small filter kernels, that can extract valuable information from different areas of an image. Therefore, they reduce the amount of training parameters significantly and are able to train also on large images.

2.1.1 Convolution

The cornerstone of ConvNets are the convolutions, represented in Figure 1. Convolutions are based on small filters, which run through the image and sum the multiplications of each input value with the corresponding filter value. Each filter must have the same amount of channels as the input, the width and height are optional.

2.1.2 Convolutional Layers

In the first layers of a network, each filter usually extracts some low-level feature, such as horizontal or vertical edges. The output of the convolution, are the outputs of each filter stacked in the channel dimension. As the convolutions get deeper into the network, they start to combine these low-level features, to higher level features, such as an eye or a mouth.

2.2 Generative Adversarial Networks

Introduced in 2014 [3], generative adversarial networks, so-called GANs, have been the focus of countless research papers and creative implementations. GANs have learned how to generate an image of a cat from a simple drawing [4], create new artworks [20], generate high-resolution celebrity faces [9], colorize grayscale images and much more.

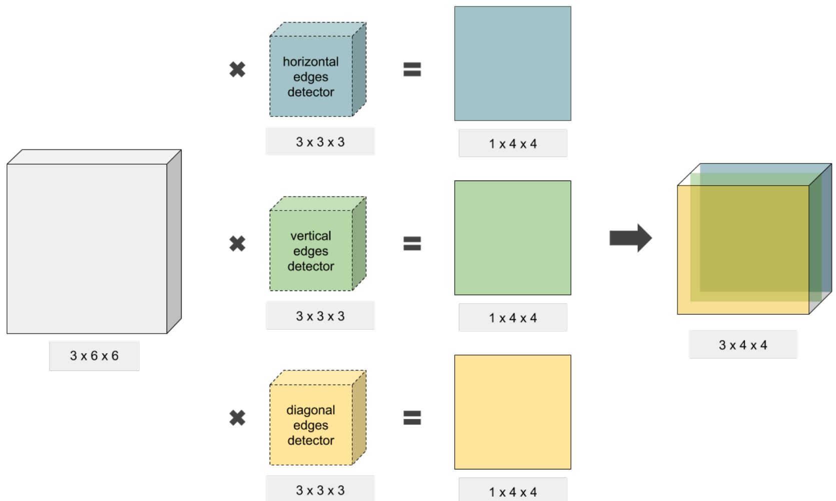


Figure 1: **Convolution.** The input image is convolved with 3 filters, each one detecting a different low-level feature. The final output is the output of each convolution stack in the channel dimension.

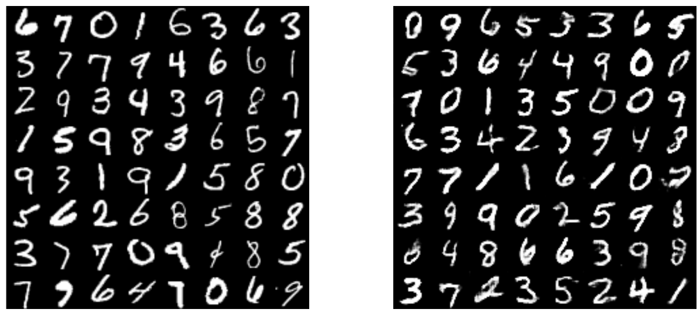
In order to generate new samples, two neural networks are trained to compete against each other. A common analogy can be found in coin forgery: while a fraud investigator is trying to detect real coins from fraudulent ones, the coin forger is trying to improve the falsification, so that the investigator is unable to tell the difference.

In computational world, the investigator is a neural network called *Discriminator* competing with the forger network called *Generator*. The discriminator is trained to classify samples from the original dataset as "real", and samples generated by the generator as "fake". The generator is trained to fool the discriminator, so that he cannot correctly classify the generated samples as fake.

Figure 2 shows example data points of the hand-written digits dataset MNIST [13], along with samples generated with a deep convolutional GAN [10] - the samples from the original and generated distributions are almost indistinguishable.

2.2.1 Architecture

Figure 3 describes a typical GAN architecture. The discriminator is a common Convolutional Neural Network [12], which takes an image as input and is trained to find filters that extract necessary information in order to classify the image.



(a) Original MNIST (b) Generated MNIST

Figure 2: **MNIST Example.** (a) Example of data points in the original MNIST dataset [13]. (b) Samples generated with a Deep Convolutional GAN [10].

The output is a 1-dimensional vector - or in case of GANs, a scalar value, classifying the image as real or fake.

The generator's architecture is similar, however it uses a transposed convolution, the so-called "deconvolution", to produce a 2 or 3-dimensional output from a 1-dimensional input vector. The generator input vector is sampled randomly.

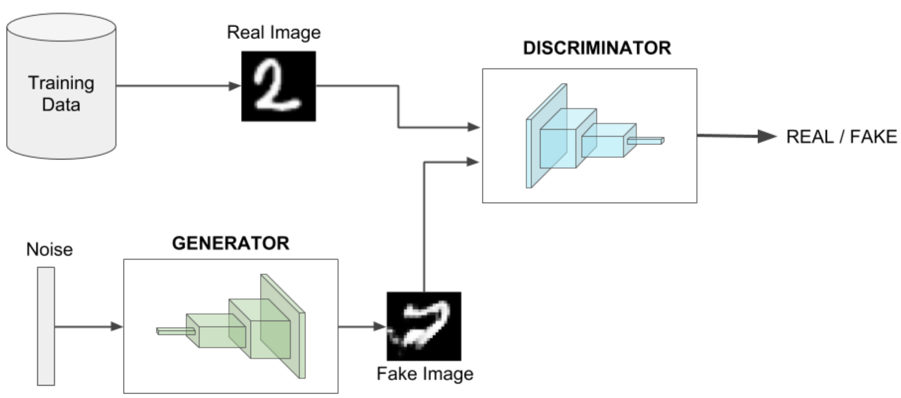


Figure 3: **GAN model.** The generator network produces fake images, which the discriminator network tries to distinguish from the real samples coming from the dataset.

The classification output of the discriminator is back-propagated through the discriminator, as well as the generator network, in order to adjust the networks' weights to improve the training objective.

2.2.2 Training Objective

Given input data with distribution $p(x)$, the generator G is a neural network, that maps random input noise z to the input space, as $G(z, \theta_g)$, learning the model distribution $\hat{p}(x)$. The discriminator D is a second neural network $D(x, \theta_d)$ with a single scalar output, classifying the input as real, sampled from $p(x)$, or as fake, sampled from the model distribution $\hat{p}(x)$.

The training objective of the discriminator is to maximize the probability of assigning the correct label, while the objective of the generator is to minimize this probability. The loss function of the GAN can be described as the minimax objective,

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log 1 - D(G(z))]. \quad (1)$$

In order to prevent vanish gradients, as a result of the discriminator saturating by confidently classifying the samples before the generator's update, [3] suggests G to be trained to maximize $\mathbb{E}_{z \sim p_z(z)}[\log D(G(z))]$ instead.

The parameters of the discriminator and generator, θ_d and θ_g, are updated using gradient descent algorithms, by back-propagating the gradients of the loss function in respect to each of the networks' parameters.

2.2.3 Training Difficulties

In theory, the minimax game described in equation 1 is played until generator has perfectly modeled the distribution $p(x)$, so that discriminator classifies the authenticity of the samples at random.

In reality, finding the right balance between the two networks can be difficult. If discriminator gets too good at determining which samples are fake, then generator has no chance to learn the distribution. On the other hand, if generator is updated too much, it can collapse too many values of z to the same value of x to have enough diversity to model the distribution.

Because a modification of the weights of the discriminator that decrease

2.2.4 Deep Convolutional GANs

Many GAN implementations base their network architecture on Deep Convolutional GANs, described by [18]. The authors introduced architectural guidelines

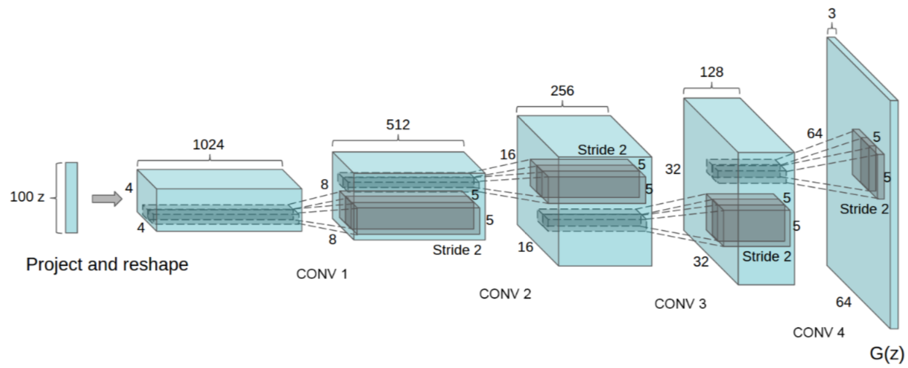


Figure 4: **DCGAN Generator.** An example architecture of a DCGAN generator $G(z)$. The random input noise z is mapped to a 64x64 image with 3 color channels via fractionally-strided convolution layers. Figure reprinted from [18].

for convolutional networks used in GANs, that stabilize the training and result in more realistic outputs.

After experimenting with CNN architectures commonly used in computer vision tasks, Radford et al. have found that this set of architecture modifications has a positive effect on the stability and convergence speed of the models:

- **All Convolutional Net**: A common practice in convolutional neural networks are *Pooling Layers*, which reduce the number of parameters by filtering the convolution outputs, taking maximum or average of a given region. However, the authors have found that using an All Convolutional Network [22], which replaces all pooling layers with a strided convolution, leads to more stable results.

- **Fully-Connected Layers**: The authors argue, that avoiding fully-connected layers deeper in the network stabilizes the network. Removing all fully-connected layers stabilizes the network, however it slows down the convergence. Therefore fully-connected layers are kept in the discriminator output layer and generator input layer.

- **Batch Normalization**: Batch Normalization [7] has been introduced in common image-classification to make models more robust to parameter initialization and speed up training. The DCGAN paper shows that including a batch-norm layer in all GAN layers (except discriminator output and generator input), helps generators start learning and produce diverse outputs.

- **Activation Functions**: As opposed to maxout activation suggested by the original GAN paper [3], GANs seem to benefit from *Leaky ReLU* activation function in discriminator, and *ReLU* activation function for generator, except last *tanh* layer.

2.2.5 Conditional GANs

In order to influence the output of a GAN, [15] introduced the Conditional Generative Adversarial Networks. By conditioning both the discriminator and the generator on additional information, the generator learns to output realistic samples given a condition.

Considering the previous example of MNIST hand-written digits synthesis, one can condition the networks using the digits as class labels, and control the network to generate an image of a specific digit. There have also been experiments with text-to-image translation [19], where the network is conditioned on a text description of an image, and various image-to-image translations such as [23], [23], [17].

The training objective of conditional GAN is following:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x|c)] + \mathbb{E}_{z \sim p_z(z)}[\log 1 - D(G(z|c))]. \quad (2)$$

The conditioning information c, such as a class label, text, or another image, is fed to both discriminator D and generator G as additional input layers.

3 Analysis

3.1 Data

As in most machine learning algorithms applied to visual tasks, the quantity and quality of training data is essential to produce high-quality results. It is also common, that data collection and data cleaning make up a significant part of a machine learning project.

In case of fashion images, there are several open-source datasets published online. However, after careful evaluation I have not found them sufficient for the purpose of this thesis. Therefore I have created an application to scrape online fashion stores and download images of fashion products and their description in a computer-readable format.

3.1.1 Requirements

In order to generate high-quality outputs using GANs, the collected dataset should fulfill the following requirements:

1. The fashion products should be photographed on a white background.
2. There should be a machine-readable description of each product, such as color, shape, category, etc.
3. The images should be in a sufficient resolution.
4. There should be a sufficient amount of images of various items.

I have defined these requirements on based on my previous experience with generative algorithms. It is generally easier for the algorithm to focus on the important attributes of the images, if there are no distractions in terms of different model poses, backgrounds, etc. And since the main point of the project is to modify different attributes of the products, the images need to be labeled.

3.1.2 Existing Fashion Datasets

I have evaluated several existing fashion datasets, however none of them satisfied the criteria for this thesis.

DeepFashion The Deep Fashion Set [14] includes more than 200.000 images of clothing images, labeled with 1000 attributes. However, this dataset does not fulfill the first requirement - its clothing items are photographed on people in various poses and backgrounds. This type of variation might be too complex for the algorithms and can lead to unsatisfactory results.

Fotolia Another available option, that has been provided by Prof. Dr. Barthel, is the Fotolia image dataset, which can be explored on the picsbuffet website [1]. To test the dataset, I have chosen all images with keywords "dress" and "isolated" and compared them to a template image, based on the distance between their 64-dimensional feature vectors calculated via Akiwi API [21]. Figure 5 shows the results of this search. Additionally to an undesired low resolution of the returned images - around 100 x 160 pixels - the results also have too much variety, such as different model poses, different zoom levels and backgrounds. The description of the images did not include many useful attributes, with most pictures being labeled with words such as: "beauty", "young", "person", and lacking information about colors, shapes and pattern.

(a) Template Image

(b) Fotolia Images

Figure 5: Fotolia images with keywords ”dress” and ”isolated” with closest feature vector distance from the template image.

3.1.3 Scraped Dataset

Based on the evaluation of existing fashion datasets, I have identified a need for creating a new dataset specifically for the requirements of the application. I evaluated several fashion e-shops, from which images of the products and their descriptions could be downloaded.

I chose 3 websites, based on the structure and amount of information they provide: zalando.de, aboutyou.de and fashionid.de. Each of them provides several thousand fashion products for women and men, with a photograph of the item on a white background and some basic description such as color, pattern, shape, length etc. For simplicity, I have limited the dataset to women's clothes, excluding products like accessories and shoes.

The *Fashion Scraper* is a python project, that scrapes images and data from all the mentioned websites and saves them locally for further processing, such as image processing, data cleaning, and description translating. This project was part of my Independent Coursework and the documentation of the project can be found in Attachments, or on github.com/sonynka/fashion_scraper.

The scraped dataset consists of 92.200 images of size 256x256 pixels in JPEG format. The images are split into folders by categories and the whole dataset is described in a CSV file, which includes the following information:

- **id**: ID of the product as defined by the website (SKU)
- **img_path**: relative local file path to the saved product image
- **img_url**: URL to the product image file hosted on the seller website
- **model_img_urls**: URLs to the images of the product worn by models
- **product_url**: if given, the URL to the product listing on the website

Figure 6: **Examples of images in the final dataset.** Each column contains images from different category as follows: blouses, dresses, jackets, knitwear, pants, skirts, tops.

- **brand**: product brand as displayed on the website
- **name**: product name as displayed on the website
- **attributes**: list of product attributes listed on the website in German, such as shape, length, material, size etc.

In addition to the website data, there are columns that contain processed and translated data. Below is the list of the column names and the values they can take.

- **category**: blouses, dresses, jackets, knitwear, pants, skirts, tops
- **color**: beige, black, blue, gray, green, pink, red, white, yellow
- **length**: 3-4, knee, long, normal, short or no value
- **sleeve_length**: half, long, short, sleeveless or no value
- **fit**: loose, normal, tight or no value
- **pattern**: floral, lace, polkadots, print, stripes, unicolors or no value
- **neckline**: back, deep, lines, round, v, wide or no value

There is also a CSV file that contains a list of image file paths and binary classification of the list of values mentioned in the above list.

A table with all attributes and num of images will be in attachments

Characteristics	pix2pix	CycleGAN	StarGAN	MUNIT	FaderNets
Supervised	Yes	No	No	No	No
Multi-Domain	No	No	Yes	No	No
Multi-Modal	No	No	No	Yes	No
Latent Space	No	No	No	Yes	Yes

Table 1: **Comparison of existing GAN models based on their characteristics.** Supervised training means that the dataset consists of image pairs, e.g: a dress and a person wearing the dress. Multi-Domain networks are able to train one model to change multiple attributes. Multi-Modal networks are able to generate more than one possible output. Networks with latent representations try to model the training data in a latent space, as opposed to pixel space.

3.2 Image-To-Image Translation

Image-To-Image translation using GANs has been widely researched in the past few years, with creative approaches and models published regularly. For the purpose of this thesis, I have reviewed and tested some of these projects, to evaluate what results can be achieved on the fashion dataset and which GAN model is best suited for what task..

Based on various characteristics, compared in Table 1, I have evaluated 5 networks: pix2pix [8], CycleGAN [24], StarGAN [2], MUNIT [6] and FaderNetworks [11]. I have compared the following attributes for each of the mentioned networks:

- Supervised/Unsupervised learning: Describes if the network needs a dataset consisting of paired images, such as the same skirt in a short and log version.
- Multi-Domain: Describes if the network is able to train only one model for different domain modifications, or if each domain requieres its own trained generator.
- Multi-Modal: Describes if the network is able to generate different versions from the same input.
- Latent representation: Describes if the network trains directly in pixel-space or uses latent representation of the data, usually content and style of the images.

3.2.1 Pix2Pix

The so-called Pi2Pix networks were first introduced by Isola et al. [8] as a framework for image-to-image translations. Some of the applications of Pix2Pix include mapping day photographs to night, sketches of shoes to realistic shoe images or colorizing black-and-white photos.

Pix2Pix uses the concept of conditional GANs [15] to influence the network's output by providing a condition c. In case of Pix2Pix, the condition is an image that is to be translated. To translate images from an input domain to a target domain, the model requires a dataset of paired images (x_{input}, x_{target}), such as a grayscale image and a corresponding color image.

As shown in Figure 7, the generator takes the input image x_{input} and tries to map it to the target domain $\hat{x}_{target}$, such as: $G : (x_{input}, z) \rightarrow \hat{x}_{target}$. The discriminator, also conditioned on the input image x_{input}, tries to predict real, if the target image comes from the data distribution and fake if it was generated. The discriminator is trying to minimize these predictions, while the generator tries to prevent correct classification.

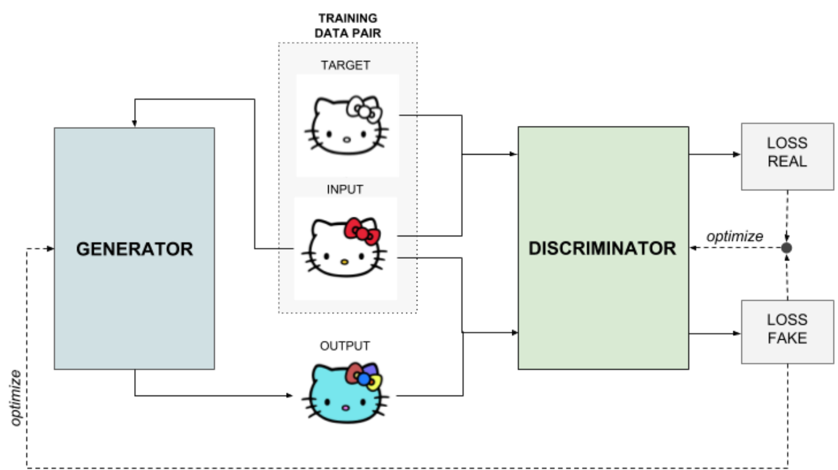


Figure 7: **Pix2Pix Training Process.** The dataset consists of paired input and target images. The generator takes the input image as condition to output the generated image. The discriminator tries to predict, given the input image, if the target image comes from the dataset or from the generator. This output is then used to further optimize both networks. Figure adapted from [5].

Training Objective The adversarial objective, which G is trained to minimize and D is trained to maximize, can be expressed as following:

$$\mathcal{L}_{adv}(D, G) = \mathbb{E}_{x_{in}, x_{trg}}[\log D(x_{in}, x_{trg})] + \mathbb{E}_{x_{in}, z}[\log 1 - D(x_{in}, G(x_{in}, z))] \quad (3)$$

Based on the results of previous conditional GAN approaches [17], Isola et. al [8] have also shown, that enforcing the generated output to be closer to the ground truth target by adding a second objective reduces artifacts in the results. They therefore suggest to use the L1 distance as reconstruction loss function for the generator to optimize.

$$\mathcal{L}_{rec}(G) = \mathbb{E}_{x_{in}, x_{trg}, z}[||x_{trg} - G(x_{in}, z)||_1] \quad (4)$$

The final objective of the generator is:

$$G^* = arg \min_G \max_D \mathcal{L}_{adv}(D, G) + \lambda \mathcal{L}_{rec}(G) \quad (5)$$

where λ controls the relative importance of the two loss functions.

Generator In image translation tasks, it is usual, and even desirable, that the input and target image domains share a common underlying structure. Modeling the generator with a simple encoder-decoder architecture, where all data must pass through a so-called "bottleneck" layer, can therefore result in an undesirable loss of the low-level details that the two domains share.

The authors therefore suggest to base the generator on a U-Net architecture. U-Net contains so-called skip connections, which pass the output directly from the encoder to the decoder, therefore skipping the bottleneck. The generator is therefore able to include low-level features from the input in the output.

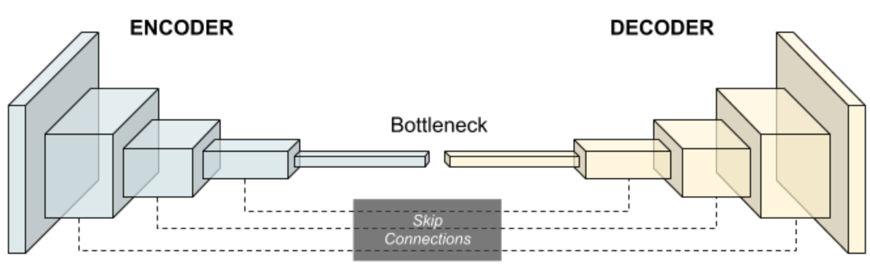


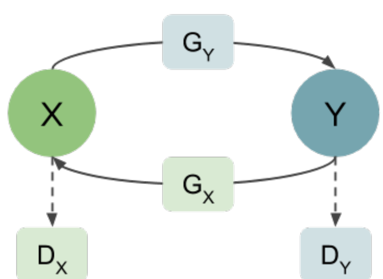
Figure 8: **U-Net Generator.** Skip connections are added between each encoder layer i and decoder layer $n - i$, where n is the total number of layers, so that information can bypass the bottleneck.

Discriminator The authors of Pix2Pix [8] argue, that while the use of L1 or L2 loss for image generation usually produces blurry outputs, they are sufficient to capture low-level frequencies, for example the colorfulness of the image. Therefore, when combining adversarial loss with an L1 loss, the discriminator only needs to focus on high-level frequencies.

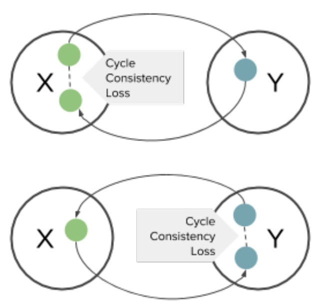
Pix2Pix introduces a *PatchGAN* architecture - a discriminator, which evaluates the high-frequency structure of the input image in patches. The PatchGAN discriminator can be described as a convolution running over the input image, classifying each patch as real or fake. The overall classification of an image is then calculated as the average of the decision for each patch. This allows the discriminator to scale efficiently to larger images and, as the authors have shown, forces sharp and colorful outputs.

3.2.2 CycleGAN

CycleGANs [24], unlike Pix2Pix, do not require a paired image set for the image domains to be translated. The unsupervised setting is preferred in most translation tasks, as obtaining image pairs of two domains can be difficult or even impossible - for example translating female faces to male, or Monet artworks to realistic photographs. The assumption hereby is that both image domains share certain underlying visual similarities.



(a) CycleGAN Model



(b) Cycle Consistency Loss

Figure 9: (a) The CycleGAN model consists of two mappings, G_X and G_Y and corresponding discriminators D_X and D_Y. (b) Cycle Consistency Loss measures the L1 distance between a real sample from one image domain and its encoded-decoded version. Figure adapted from [24].

The model consists of two generators, G_X and G_Y, which learn mapping from image domain X to image domain Y, $G_Y : X \rightarrow Y$, and vice versa, $G_X : Y \rightarrow X$. Each of the image domains has its own discriminator, D_X and D_Y, that check if the given image comes from the real distribution of the domain or is translated from the opposite domain.

The *Adversarial Loss* is therefore applied to both pairs of networks, (D_X, G_X) and (D_Y, G_Y).

$$\mathcal{L}_{adv}(D_X, G_X) = \mathbb{E}_x[\log D_X(x)] + \mathbb{E}_y[\log 1 - D_X(G_X(y))] \tag{6}$$

Additionally to the adversarial loss, CycleGAN also implements a so-called *Cycle Consistency Loss*, which enforces that an image encoded from one domain to another, can also be brought back to the original domain. The authors argue, that reducing the amount of possible mapping functions can help avoid the network to learn a mapping, which produces a random permutation of the target images for the same input images.

The *forward cycle consistency* defines that the an image x from domain X and its encoded-decoded version should be approximately the same: $G_X(G_Y(x)) \approx x$. The *backward cycle consistency* defines the same for an image from domain Y.

$$\mathcal{L}_{cyc}(G_X, G_Y) = \mathbb{E}_x[||G_X(G_Y(x)) - x||_1] + \mathbb{E}_y[||G_Y(G_X(y)) - y||_1] \tag{7}$$

The authors argue, that by enforcing this encoder-decoder logic, the generators learn not to contradict each other. The final objective is:

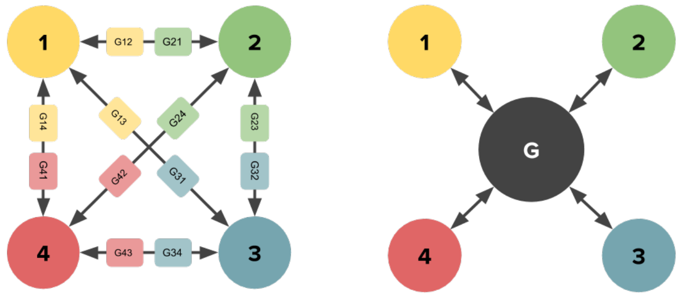
$$\begin{aligned} G_X^*, G_Y^* = arg \min_{G_X, G_Y} \max_{D_X, D_Y} \mathcal{L}_{adv}(D_X, G_Y) \; &+ \mathcal{L}_{adv}(D_Y, G_X) \\ &+ \lambda \mathcal{L}_{cyc}(G_X, G_Y) \end{aligned} \tag{8}$$

3.2.3 StarGAN

In case of Pix2Pix and CycleGAN, one of the disadvantages is the missing possibility multi-domain training. In other words, if there are more than 2 domains to translate between each other, one must train a new model for each domain pair. This can be time and computationally intensive.

StarGAN [2] is unique among the tested models, as it is able to train one single generator that maps input to multiple domains, as shown in Figure 10. Using the conditional GAN model, the generator is conditioned on a randomly chosen target domain each iteration, so that it can learn mapping for all given domains.

Given an image x and a target domain label c_{trg}, generator G tries to minimize the adversarial loss objective, to generate real-looking images conditioned



(a) Cross-Domain Models (b) StarGAN Model

Figure 10: **Comparison of cross-domain models and StarGAN model.** While cross-domain models need one generator per each domain pair, StarGAN only trains one generator for multiple domains. Figure adapted from [2].

on the target domain, while the discriminator classifying the source of the image, D_{src}, tries to maximize it.

$$\mathcal{L}_{adv} = \mathbb{E}_x[\log D_{src}(x)] + \mathbb{E}_{x,c_{trg}}[\log 1 - D_{src}(G(x, c_{trg}))] \tag{9}$$

Instead of feeding the target domain class c to the discriminator, as in common conditional networks, StarGAN uses an auxiliary classifier GAN, AC-GAN [16]. It forces the discriminator to output both the probability distribution over the sources of the input, and the probability of the target domain labels, $D : x \rightarrow D_{src}(x), D_{cls}(x)$. This improves the network's stability and performance, as it forces the discriminator to perform and additional task.

This modification to the discriminator introduces the *Domain Classification Loss* with two objectives, optimizing D and G respectively. Given training data with an image x and its original domain c_{in}, D learns to classify the domain label correctly by minimizing the following objective:

$$\mathcal{L}^r_{cls} = \mathbb{E}_{x,c_{in}}[-\log D_{cls}(c_{in}|x)]. \tag{10}$$

The generator objective is to generate images, that fool the auxiliary classifier and are classified as the target domain c_{trg}:

$$\mathcal{L}^f_{cls} = \mathbb{E}_{x,c_{trg}}[-\log D_{cls}(c_{trg}|G(x, c_{trg}))]. \tag{11}$$

StarGAN [2] also uses the *Cycle Consistency Loss* [24], in order to generate images that preserve the underlying content of the original image, and only change the domain-related attributes:

$$\mathcal{L}_{cyc} = \mathbb{E}_{x,c_{in},c_{trg}}[||x - G(G(x, c_{trg}), c_{in})||_1]. \tag{12}$$

The full training objective for D and G respectively is:

$$\mathcal{L}_D = -\mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}^r_{cls}, \tag{13}$$

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}^f_{cls} + \lambda_{cyc}\mathcal{L}_{cyc}, \tag{14}$$

where λ_{cls} and λ_{cyc} control the relative importance of the two loss functions against the adversarial loss.

3.2.4 MUNIT

All of the introduced models approach the image translation problem as one-to-one mapping. However, many of the image domain translation tasks are in fact multi-modal, meaning one single input can have multiple different outputs. The MUNIT network [6] introduces an unsupervised multi-modal method, which is able to capture the diversity of the output.

Instead of using translation in pixel space, the model tries to model the translation in a latent space. Based on the *partially shared latent space assumption*, a content latent code $c \in \mathcal{C}$ and a style latent code $s_i \in \mathcal{S}_i$ are separated, assuming that the two image domains share a common content space but each has an individual style space.

For each domain, there is an auto-encoder, which consists of a generator G and two encoders: content encoder E^c, and style encoder E^s. Given an image $x \in X$, it is encoded into a content and latent code, $E^c_x(x)$ and $E^s_x(x)$. To translate the image $x \in X$ to domain Y, its content code c_x is extracted using the content encoder E^c_x, and it is combined with a random style code s_y, $G_y(c_x, s_y)$.

Reconstruction Loss. The reconstruction loss is similar to the cycle consistency loss [24], enforcing the reconstruction between an image and its latent representation in both directions. $\mathcal{L}^x_{rec}$ encourages

$$\mathcal{L}^x_{rec} = \mathbb{E}_x[||x - G_x(E^c_x(x), E^s_x(x))||_1] \tag{15}$$

$$\mathcal{L}^{c_x}_{rec} = \mathbb{E}_{c_x,s_y}[||c_x - E^c_y(G_y(c_x, s_y))||_1] \tag{16}$$

$$\mathcal{L}^{s_y}_{rec} = \mathbb{E}_{c_x,s_y}[||s_y - E^s_y(G_y(c_x, s_y))||_1] \tag{17}$$

Adversarial Loss.

While the style code is assumed to have a global and simple effect and is therefore sufficiently represented by a low-dimensional vector, the content is assumed to be a high-dimensional vector describing the complex spatial structure of the data.

4 Other

4.1 Model Images

For some of the tasks of this project also images of people modeling the products were required. However, each product, has multiple images of a model wearing it, and for the purposes of this project, I had to use clustering methods to filter out only images where the models are facing the front and photographed full-bodied.

4.1.1 Clustering

The supervised algorithms require a paired image set - in the case of generating model image from product image, it was neccessary to obtain images showing a person modelling the product. This could be obtained from the scraped websites, however each product usually has multiple model images - unordered, usually front-facing, back-facing, detail and cut. I have applied the K-Means clustering algorithm in order to separate these types of images, and for each product, find the front-facing model image.

5 Products to Models

One of the tasks of the framework is to generate a realistic image of a model wearing a product, given an image of the product. The motivation is to expand the amount of similar images found - as some products might not have a photograph without a model available at all.

I acquired the dataset for this task by downloading all model images for each product, as listed in the scraped data file. However, each product usually has multiple images with variety of model poses and detail; usually unordered and with no pattern that would indicate which image displays what pose. Therefore I have applied the K-means clustering algorithm to try to cluster these image by the model pose and for each product, select the image where the model is photographed front-facing and full-bodied.

5.1 Clustering

References

[1] Picsbuffet. URL https://picsbuffet.com/fotolia/#0,147,1576.

[2] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. URL http://arxiv.org/abs/1711.09020.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[4] C. Hesse. Image-to-Image Demo - Affine Layer, . URL https://affinelayer.com/pixsrv/.

[5] C. Hesse. Image-to-Image Translation in Tensorflow - Affine Layer, . URL https://affinelayer.com/pix2pix/.

[6] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz. Multimodal Unsupervised Image-to-Image Translation. URL http://arxiv.org/abs/1804.04732.

[7] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. URL http://arxiv.org/abs/1502.03167.

[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. URL http://arxiv.org/abs/1611.07004.

[9] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. URL http://arxiv.org/abs/1710.10196.

[10] T. Kim. DCGAN-tensorflow: A tensorflow implementation of "Deep Convolutional Generative Adversarial Networks". URL https://github.com/carpedm20/DCGAN-tensorflow.

[11] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. DENOYER, and M. A. Ranzato. Fader Networks:Manipulating Images by Sliding Attributes. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5967–5976. Curran Associates, Inc. URL http://papers.nips.cc/paper/7178-fader-networksmanipulating-images-by-sliding-attributes.pdf.

[12] Y. LeCun and Y. Bengio. Convolutional Networks for Images, Speech, and Time-Series.

[13] Y. LeCun, C. Cortes, and C. Burges. MNIST handwritten digit database. URL http://yann.lecun.com/exdb/mnist/.

[14] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. URL http://mmlab.ie.cuhk.edu.hk/projects/DeepFashion/AttributePrediction.html.

[15] M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. URL http://arxiv.org/abs/1411.1784.

[16] A. Odena, C. Olah, and J. Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. URL http://arxiv.org/abs/1610.09585.

[17] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context Encoders: Feature Learning by Inpainting. URL http://arxiv.org/abs/1604.07379.

[18] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. URL http://arxiv.org/abs/1511.06434.

[19] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative Adversarial Text to Image Synthesis. URL http://arxiv.org/abs/1605.05396.

[20] rkjones4. GANGogh: Using GANs to create Art. URL https://github.com/rkjones4/GANGogh.

[21] P. D. K.-U. B. J. H. N. H. M. K. A. Sonnenberg. Akiwi - a keywording tool. URL http://www.akiwi.eu.

[22] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for Simplicity: The All Convolutional Net. URL http://arxiv.org/abs/1412.6806.

[23] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon. Pixel-Level Domain Transfer. URL http://arxiv.org/abs/1603.07442.

[24] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. URL http://arxiv.org/abs/1703.10593.