UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXIII

Guiding the matrix: conditional GANs and meta-learning for image synthesis

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Tutore:
Chiar.mo Prof. Andrea Prati

Dottorando: Tomaso Fontanini

Anni 2017/2020

to my parents, Stefania, Klathù and all the people in my life.

"Groovy."

Contents

List of Figures

List of Tables

Introduction

Long time ago
in a galaxy far far away...

Chapter 1

Context

Artificial intelligence, and Deep Learning in particular, is one of the hot topic in the computer vision scientific community for a few years. The reason is that Deep Learning techniques outperformed classical computer vision algorithms in many areas, such as image classification, detection and segmentation.

Nevertheless, the most impressive results of Deep Learning, that were almost unthinkable before, are to be found in the field of image synthesis and generation. Indeed, with the introduction of Generative Adversarial Network (GANs) [2], the generation of photorealistic images, such as high resolution human faces (Fig. 1.1), is a reality and distinguishing between real and generated images has become a very hard task for a human that often needs to rely on another algorithm to do that.

Even if GANs results are impressive, one of their main problem is that they are very difficult to control. In other words, guiding these models to produce a very specific output is not trivial. This is due to the fact that vanilla GANs use only random noise as input and no additional information is given to the network. For example, what if we want to colorize a grayscale picture of a cat and at the same time we want the cat to be red in the final result? Or what if we want to change a specific facial attribute in a face (*e.g.* adding mustache) without altering the identity of the person in the picture? This limitation can also lead to a poor application of generative model in real context where it is very important to have complete control over the output of

Figure 1.1: Human faces generated by NVIDIA StyleGAN [1].

an algorithm.

For this reason, conditioning GANs, in order to guide them, is a fundamental step to better understand the behaviour of Generative Adversarial Networks and to greatly improve their results and usability. More specifically, this means feeding additional information into the network in order to control the generated samples.

The concept of having additional information other than random noise is not new, but still is one of the main concern when a generative network is engineered. Tasks like style transfer, colorization or segmentation all need some conditioning. The most

common ways of conditioning a GANs include feeding a label [3, 9], an image [6, 7, 20, 8] or even text [21].

Chapter 2

Problems and solutions

The problems addressed in this thesis will cover a broad range of topics like shoe colorization, few shot colorization, multi-domain image-to-image translation, meta-learning, attention transfer and GANs compression. Each of them deals with conditioning an adversarial network in different ways. Indeed, conditioning in the various algorithm was the most important part in their definition. Also, new architectures and various improvement over existing ones will be introduced and described in detail.

The main contributions of this thesis are:

- a system able to automatically colorize shoe models starting from a blank or partially colored 3d model;
- a new architecture called MetalGAN that combines meta-learning and conditional GANs for few-shot image colorization;
- an extension of MetalGAN that tackles the problem of multi-domain image-to-image translation and outperforms existing methods;
- a system where attention maps can be used as knowledge to be transferred in a teacher-student paradigm for image-to-image translation improving the student performance.

2.1 Shoe model colorization using conditional GANs

Fashion industries have always relied on the work of talented designer in order to draw the next best selling shoe or cloth. Nevertheless, lots of the designer repetitive actions can be automated in order to make them freely create.

In particular, in this thesis we will discuss how to develop a system that is able, having at disposal a colored side of a shoe, to completely colorize the other side consistently. This will allow the designer to be much faster in their work without limiting their creativity, but rather enhancing it. The main challenge of this objective is that the colorization needs to be very sharp and precise, maintaining the color consistency of the shoes, *e.g.* the three stripes that are very common in the Adidas shoes are required to have the same color. A generative network conditioned on one side of a shoe will be developed in order to solve this problem, showing a real world application of GANs.

2.2 MetalGAN: A Cluster-Based Adaptive Training for Few-Shot Adversarial Colorization

In recent years, the majority of works on deep-learning-based image colorization have focused on how to make a good use of the enormous datasets currently available. But this quantity of data is not always given, expecially in real world application.

In this thesis, we will prove that is possible to train a colorization network even without a large quantity of data, still maintaining optimal results. The adopted approach is a mixed one, which uses an adversarial method for the actual colorization, and a meta-learning technique to enhance the generator model. This work will be validated over a small dataset of images, that is Mini-Imagenet [22], proving that meta-learning can be used succesfully in combination with conditional GANs.

2.3 MetalGAN: Multi-domain label-less image synthesis using cGANs and meta-learning

Image synthesis is currently one of the most addressed image processing topic in computer vision and deep learning fields of study. Researchers have tackled this problem focusing their efforts on its several challenging problems [9, 6, 7], *e.g.* image quality and size, domain and pose changing, architecture of the networks, and so on. Above all, producing images belonging to different domains by using a single architecture is a very relevant goal for image generation. In fact, a single multi-domain network would allow greater flexibility and robustness in the image synthesis task than other approaches. Existing single model architectures like [9] heavily relies on labels to guide the output of the translation, restraining the model on a fixed number of domains.

This thesis proposes a novel architecture and a training algorithm, which are able to produce multi-domain outputs using a single network. A small portion of a dataset is intentionally used, and there are no hard-coded labels (or classes). This is achieved by combining a conditional Generative Adversarial Network (cGAN) for image generation and a Meta-Learning algorithm for domain switch, and we called our approach MetalGAN. The approach has proved to be appropriate for solving the multi-domain problem and it is validated on facial attribute transfer, using CelebA and RAFD dataset.

2.4 Conveying knowledge in conditional GANs with attention distillation

Recently, visually explaining convolutional network, generating attention maps that highlight the most important region for the network objectives, got lots of attention. Several algorithms, like GradCAM [15], were developed and applied to classification network, but visually explaining generative network is still lacking.

In this thesis we will prove that is possible to generate sharp attention maps from a conditional generative adversarial network and also that it is possible to use them

to improve the performance of a smaller network sharing the same task. Following a teacher-student approach, we will transfer attention from the teacher generator to the student generator showing an increment in the quality of the samples generated by the student. We will also prove that is possible to transfer attention between a teacher and a student that share a similar but different task still improving the student result.

2.5 Structure of the thesis

This thesis is organized as follows:

- Part I will explore all the state-of-the-art related to the topics of the dissertation;
- Part II will present the research findings. Firstly, in Chapter 6 results on colorization of partially colored shoe images will be presented. Secondly, Chapter 7 and 8 will focus on the combination between conditional GANs and meta-learning. Finally, Chapter 9 will discuss attention maps transfer between conditional GANs in a teacher-student settings;
- Part III will present the conclusions and future works.

Part I

Background and related works

11

Did you ever hear
the tragedy
of Darth Plagueis The Wise?

Chapter 3

GANs and cGANs for image-to-image translation

In this chapter we will focus on the explanation of Generative Adversarial Networks (GANs) as well as on conditional Generative Adversarial Network (cGANs). Different architectures and approach for image-to-image translation will be presented as well.

3.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) can be considered a framework for estimating generative models via an adversarial process. They were first introduced by Ian J. Goodfellow in [2]. Up until that moment generative models have had much less of an impact than discriminative models, where an high-dimentional input is mapped to a class label. The reason was that generative models presented mainly two big limitations:

- the difficulty of approximating many intractable probabilistic computations;
- the difficulty of leveraging the benefits of piecewise linear units in the generative context

GANs were designed to overcome these two difficulties.

3.1.1 How do GANs work?

As well expressed in [23], GANs can be explained in terms of a game between two entities. The first one is called *generator* while the second one is called *discriminator*. The role of the generator is to create samples that are indistinguishable from distribution of the training data, while the role of the discriminator is to examine samples to determine whether they are real or fake. The discriminator is trained as a supervised discriminative model, classifying the input into two classes, that is real or fake. On the other side, the generator is trained with the objective of fooling the discriminator. The basic structure of a GAN applied on images can be seen in Fig. 3.1.

More specifically, in order to learns the generator's distribution p_g over data $\mathbf{x}$, a prior on input noise variables $p_z(z)$ is defined and a mapping to data space is represented as $G(z,\theta_g)$, where G is a multilayer perceptron with parameters θ_g. Then, a second multilayer perceptron that outputs a single scalar is defined as $D(\mathbf{c},\theta_d)$. $D(\mathbf{x})$ represents the probability that $\mathbf{x}$ belongs to the training data distribution and not from p_g. D is trained to maximize the probability of assigning the correct label to both training examples and samples from G. On the other side, G is trained to minimize $\log 1 - D(G(z))$. This means that D and G play a two-player minmax game with value function $V(G,D)$:

$$\underset{G}{min}\underset{D}{max}V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log 1 - D((G(z)))] \tag{3.1}$$

The main problem with equation 3.1 is that early in the training, when G is poor, D will reject samples with high confidence and the gradient of G will vanish. To solve this G can be trained to maximize $log(D(G(z)))$ instead of minimizing $\log 1 - D(G(z))$. Doing this, the objective function will provide a much stronger gradient early in the training.

Some generated images are showed in Fig. 3.2. One of the main feature that set apart GANs from other generative models is the fact that since the generator is

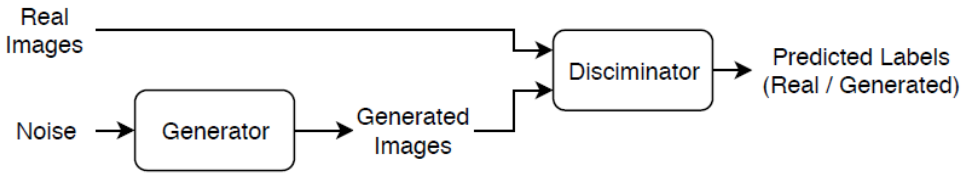


Figure 3.1: GAN framework: The discriminator learns to distinguish between real images and generated images while the generator tries to fool the discriminator.

not updated directly with data examples, but only with gradient flowing through the generator, it can more easily avoid to components of the input in its parameters. Also, adversarial networks produce results that are much sharper than methods based on Markov chains.

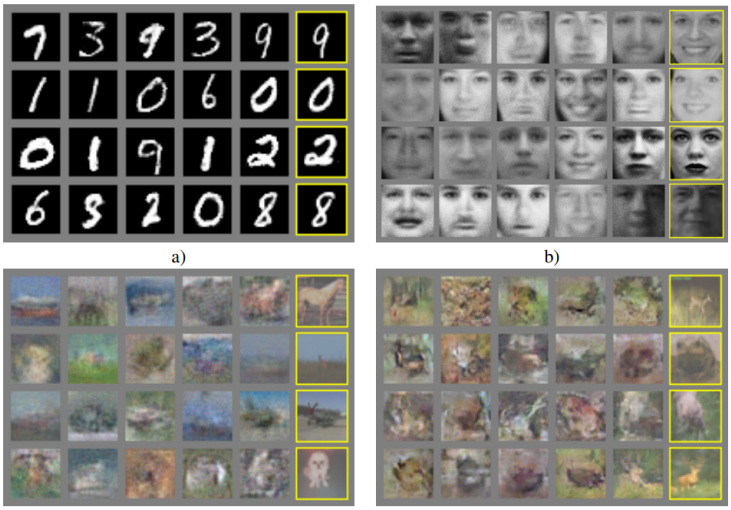


Figure 3.2: Visualization of samples from the model. The further right column represent the nearest training example of the neighboring samples [2].

3.2 Conditional GANs

In vanilla GANs there is no control on modes of the data being generated. On the other side, it is possible to condition the model with additional information in order to direct the data generation process. This kind of architecture is called conditional GAN (cGAN) [3].

In order to do so, the generator and the discriminator are conditioned with some extra information **y**, that is labels, images or any kind of auxiliary information. **y** is fed into both the generator and the discriminator. In the generator $p_{\mathbf{z}}(\mathbf{z})$ is combined with **y** in joint hidden representation while in the discriminator **x** and **y** are presented as inputs.

The objective function of a cGAN then becomes:

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log 1 - D((G(\mathbf{z}|\mathbf{y})))] \quad (3.2)$$

The structure of a standard conditional adversarial network is presented in Fig. 3.3.

Some results obtained with MNIST dataset [24] can be seen in Fig. 3.4.

3.3 Deep Convolutional GANs

A serious problem of GANs is that they are very unstable during training, which can make the generator produce nonsensical outputs. Deep Convolutional GANs (DCGAN) [4] were introduced to make the training of GANs stable in most settings. In addition to that, the authors also demonstrated that the trained discriminator can be used for image classification tasks, showing competitive performance with other unsupervised algorithms.

The main architectural changes to the GANs architecture presented in DCGAN are essentially three:

- All Convolutional Net, that is replacing deterministic spatial pooling functions

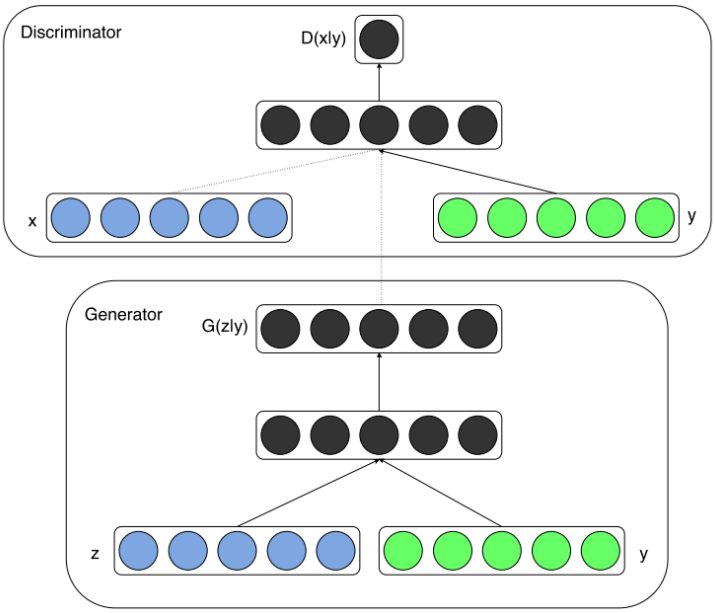


Figure 3.3: A conditional adversarial network [3].

whit strided convolutions;

- Eliminating fully connected layers on top of convolutional features;
- Use of Batch Normalization [25] which stabilizes learning by normalizing the input to each unit to have zero mean and unit variance.

Furthermore, ReLU activation [26] are used in the Generator with the exception of the last layer that uses a Tanh function, while LeakyReLU activations [27] are used in the discriminator for all layers.

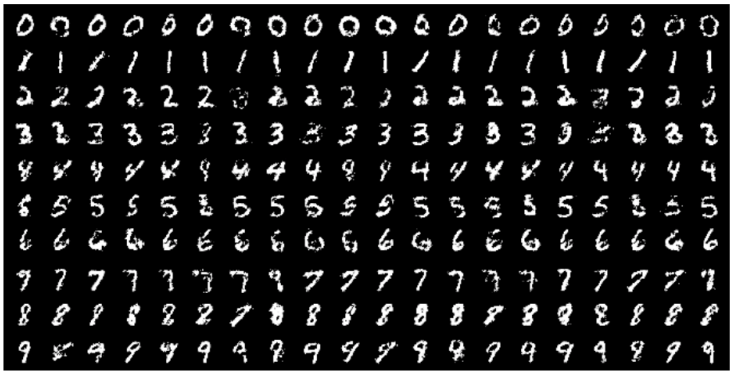


Figure 3.4: Generated cGAN images using MNIST dataset. Each rows is conditioned with a different label.

Figure 3.6: Some samples from a DCGAN.

The full architecture of a DCGAN generator is presented in Fig. 3.5, while some results on the LSUN bedrooms dataset [28] are shown in Fig. 3.6.

3.4 Least Squares Generative Adversarial Networks

The loss function of regular GANs could lead to the vanishing gradient problem during training. To overcome this issue, Least Squares Generative Adversarial Networks

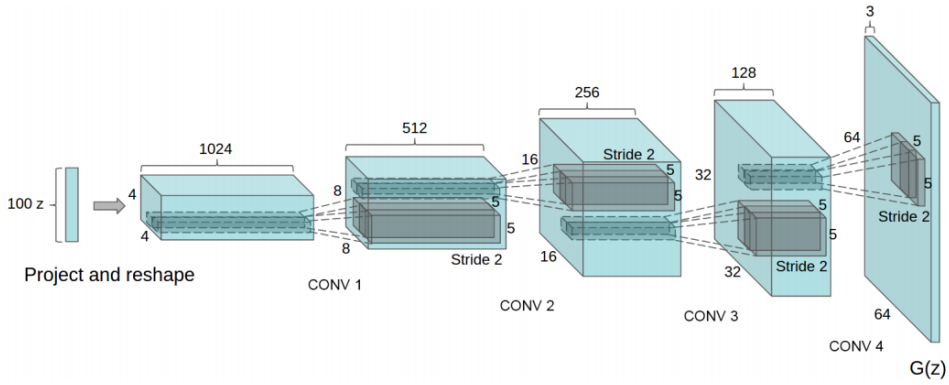


Figure 3.5: Generator of a DCGAN [4].

(LSGANs) [29] were introduced. LSGANs adopted the least square loss function for the discriminator.

Using the *a*-*b* coding scheme for the discriminator, where a and b are the labels for fake and real data, the objective function becomes:

$$\begin{aligned} \min_D V_{LSGAN}(D) &= \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[(D(\mathbf{x})-b)^2] + \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[(D(G(\mathbf{z}))-a)^2] \\ \min_G V_{LSGAN}(G) &= \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[(D(G(\mathbf{z}))-c)^2] \end{aligned} \tag{3.3}$$

where c denotes the value that G wants D to believe for fake data.

3.5 Wasserstein GANs

Wasserstein GANs (WGANs) [30] were introduced to further improve the stability of GANs training and to also reduce the mode dropping phenomenon. In addition, when implementing a WGAN it is not necessary to carefully balance the training of the generator and the discriminator nor to carefully design the network architecture.

WGANs make use of the Wassestein distance, that is:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|] \tag{3.4}$$

where $\prod(\mathbb{P}_r, \mathbb{P}_g)$ is the set of all joint distributions $\gamma(x,y)$ whose marginals are $\mathbb{P}_r$ and $\mathbb{P}_g$. In other words, $\gamma(x,y)$ can be seen as the mass needed to be moved from x to y in order to transform the distribution $\mathbb{P}_r$ into $\mathbb{P}_g$.

In practice, in WGANs the discriminator is replaced with a critic f that do not have an output sigmoid function. The cost function then becomes:

$$g_w \leftarrow \nabla_w \frac{1}{m} \sum_{i=1}^{m} [f(x^{(i)}) - f(G(z^{(i)}))] \tag{3.5}$$

for the critic and:

$$g_\theta \leftarrow \nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f(G(z^{(i)})) \tag{3.6}$$

for the generator.

For the previous equations to be true, f has to be a 1-Lipschitz function following the constraint:

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2| \tag{3.7}$$

In order to enforce the constraint, WGAN applies a clipping to restrict the maximum weight value in f:

$$\begin{aligned} w &\leftarrow w + \alpha \cdot RMSProp(w, g_w) \\ w &\leftarrow clip(w, -c, c) \end{aligned} \tag{3.8}$$

This only partially solve the problem, since, as stated by the authors of the paper, weight clipping is a clearly terrible way to enforce a Lipshitz constraint. If the clipping parameter is large, it will be very hard to train the critic optimally and if the clipping is small, this cal lead to vanishing gradients.

3.5.1 WGANs with gradient penalty

These problems were solved in [5] which introduced *gradient penalty* in the WGAN training to enforce the Lipshitz constraint.

If f^* if a differentiable 1-Lipshitz function, then it is true that f^* has gradient norm 1 almost everywhere under $\mathbb{P}_r$ and $\mathbb{P}_g$. The authors then proved the following proposition:
Let $\mathbb{P}_r$ and $\mathbb{P}_g$ be two distributions in $\mathcal{X}$, a compact metric space. Then, there is a 1-Lipshitz function f^* which is the optimal solution of $max_{\|f\|_L \leq 1} \mathbb{E}_{y\sim\mathbb{P}_r}[f(y)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)]$. Let π be the original coupling between $\mathbb{P}_r$and$\mathbb{P}_g$, defined a s the minimizer of: $W(\mathbb{P}_r,\mathbb{P}_g) = \inf_{\pi\in\Pi(\mathbb{P}_r,\mathbb{P}_g)}[\|x-y\|]$ where $\Pi(\mathbb{P}_r,\mathbb{P}_g)$ is the set of joint distributions $\pi(x,y)$ whose marginals are $\mathbb{P}_r$ and $\mathbb{P}_g$, respectively. Then, if f^* is differentiable, $\pi(x=y)=0$ and $x_t = tx+(1-t)x$ with $0\leq t\leq 1$, it holds the $\mathbb{P}_{(x,y)\sim\pi}[\nabla f^*(x_t) = \frac{y-x_t}{\|y-x_t\|}] = 1$.

In other words, points interpolated betweent real and fake data should have a gradient norm of 1 for f. For this reason, instead of weight clipping, WGANs with gradient penalty (WGAN-GP) penalizes the model if the gradient norm moves away from 1. The cost function then becomes:

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r}[D(x)]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|)^2]}_{\text{Gradient penalty}} \tag{3.9}$$

where $\hat{x}$ is sampled from $\tilde{x}$ and x following the equation $\hat{x} = t\tilde{x}+(1-t)x$ with $0\leq t\leq 1$ and λ is set to 10.

In Fig. 3.7 a comparison of different methods and architectures is presented. WGAN-GP is the only method that is able to successfully train on all the different scenarios.

Figure 3.7: A set of GAN architectures trained with different methods [5].

3.6 Image-to-Image Translation with Conditional Adversarial Networks

Conditional Adversarial network can be used also for solving very different image-to-image translation problems. In [6] the authors demonstrated that these networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping. For this reason, they can be applied on several different scenarios that previously would have required very different loss formulations as swown in Fig. 3.4.

In this case, a conditional GAN learns a mapping from observed image x and random noise vector z, to y, $G : \{x, z\} \rightarrow y$. The objective of the cGAN can be expressed

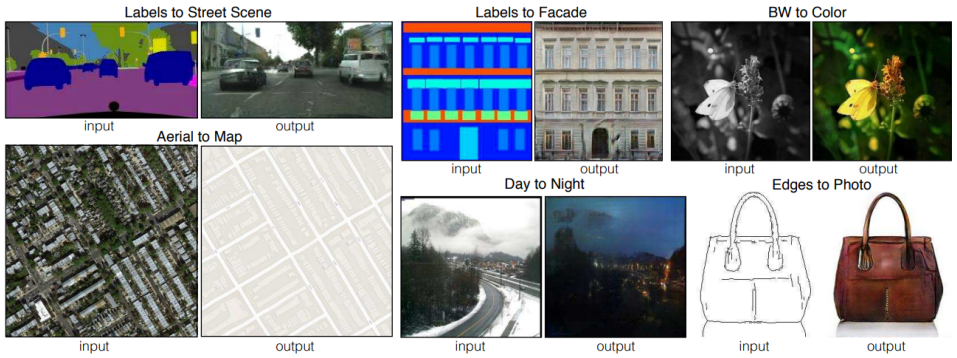


Figure 3.8: Results using conditional adversarial network to solve different image-to-image problems.

as:

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x,G(x,z)))] \tag{3.10}$$

An example of this training is presented in Fig. 3.9.

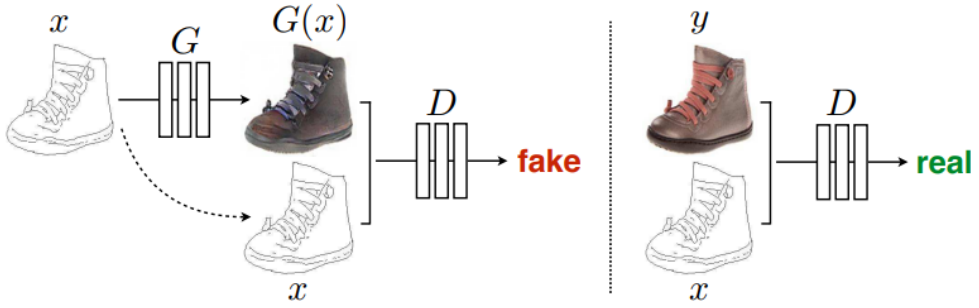


Figure 3.9: Training a cGAN on edges → photo. Unlike a standard GAN, both G and D observe the input edge map [6].

In order to further improve the results, the cGAN objective is mixed with a more traditional loss, such as L1 distance:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1] \tag{3.11}$$

The final objective then becomes:

$$G^* = \underset{G}{\operatorname{argmin}} \underset{D}{\operatorname{max}} \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G) \tag{3.12}$$

A comparison between the different losses can be seen in Fig. 3.10.

3.6.1 Network architecture

For many image translation problems, lots of low-level information in the input is shared with the output. For this reason, the generator implemented skip connection following the general shape of the U-Net [31].

Regarding the discriminator, since it is well know that L1 loss accurately capture the low frequencies, the authors decided to restricting it in order to model only the high-frequencies. The new discriminator was called *Patch*GAN and it only penalizes structure at the scale of patches. Therefore, the discriminator tries to classify if each $N \times N$ patch in an image is real or fake.

Figure 3.10: Comparison between the usage of different loss function [6].

3.7 Unpaired image-to-image translation using cycle-consistent adversarial networks

One of the major drawback in [6] was that, in order to properly work, the system needed a paired dataset of images, that is for each input image the desired output (ground truth) was present. This is unfeasible for several applications, since often a

ground truth is not present or is very difficult to obtain. In order to solve this limitation, Zhu *et al.* [7] proposed a new architecture called CycleGAN that is able to translate an image from a source domain X to a target domain Y in absence of a paired dataset. Since the mapping $G : X \rightarrow Y$ is highly under-constrained, it is coupled with an inverse mapping $F : Y \rightarrow X$ and a cycle consistency loss is introduced in order to push $F(G(X)) \approx X$.

The model contains two mapping function $G : X \rightarrow Y$ and $F : Y \rightarrow X$ and therefore two discriminators D_Y and D_X. D_Y helps G to produce images indistinguishable from domain Y and D_X do it the same for F and domain X. Adversarial training can learn mapping G and F, but with large enough capacity, a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an output distribution that matches the target distribution. For this reason, Cycle consistency was introduced, as showed in Fig. 3.11 to further reduce the space of possible mapping function. The idea is that, for each image x taken from domain X, the image translation cycle should bring x back to the original image, that is $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. This behavior is pushed by a *cycle consistency loss*:

$$\mathcal{L}_{cyc}(G,F) = \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|] + \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|] \tag{3.13}$$

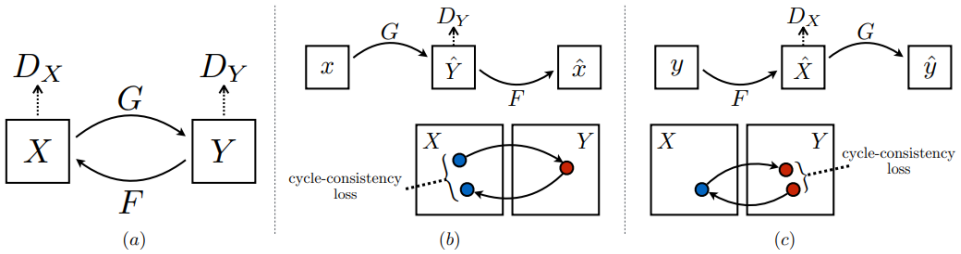


Figure 3.11: Network architecture of CycleGAN [7].

Some results of this approach are presented in Fig. 3.12.

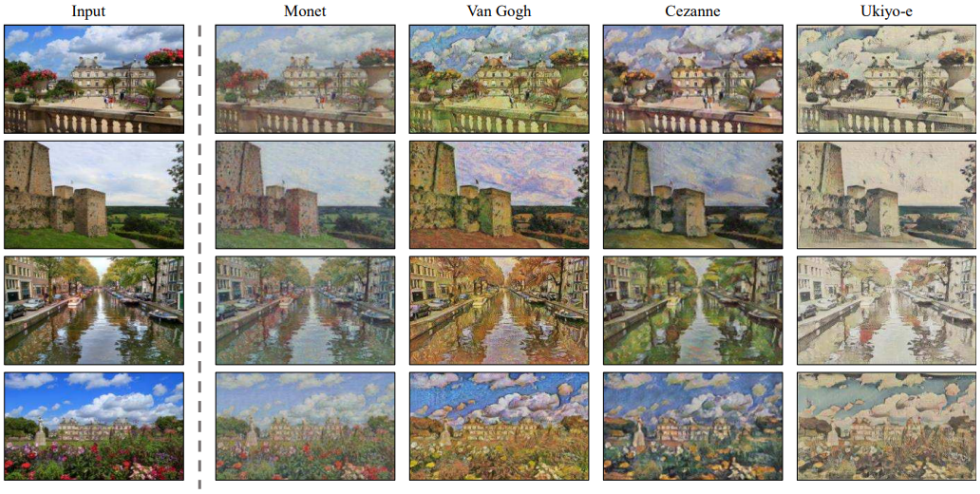


Figure 3.12: Some CycleGAN results. A picture is transferred into different artistic styles.

3.8 Scribbler: Controlling deep image synthesis with sketch and color

Image generation methods are able to produce outputs that are almost indistinguishable from real samples, but they remain very difficult to control the outcome of the generation. What if, for example we would like to generate a person wearing a blue dress and not a red one? Scribbler [20] was introduced to provide a sketch based image synthesis system that is controllable by the user through scribbles over the sketch that indicate the preferred color for the objects.

The network is designed as a feed-forward neural network that takes an image as input and generate a photo of the same resolution as output. In order to produce the expected results, Scribbler combines different loss functions. Firstly, a L2 loss between output and ground-truth, denoted as L_p. Secondly, a feature loss L_f defined as the L2 difference in feature space, with the feature extracted from a certain layer of a pre-trained neural network. Thirdly, an adversarial loss L_{adv} is added in order to encourage more variations and vividness in results. Finally, a total variation loss L_{tv}

in introduce to enhance smoothness in the output. The final objective then becomes:

$$L = w_p L_p + w_f L_f + w_{adv} Ladv + w_{tv} L_{tv} \tag{3.14}$$

To train the network to recognize scribbles at test time, synthetic strokes based on the color of the ground-truth images are generated and are then applied on the input sketches. Some Scribbler results generated following this method are presented in Fig. 3.13.

Figure 3.13: Guided sketch colorization results on held out test sketches for bedroom (left) and car (right).

3.9 Texturegan: Controlling deep image synthesis with texture patches

While previous image synthesis methods could be controlled by sketch and color strokes, TextureGAN [8] introduced texture control.

Fig. 3.14 and 3.15 present the TextureGAN pipeline that is composed by two different training phases. The objective of the first one is to pre-train the network to reproduce the exact texture of the ground-truth images, while the goal of the second one is to support a broader range of textures and to propagate unseen textures better by fine-tuning the network with a separate texture-only database. It is worth specifying that, TextureGAN uses **Lab** color space.

During the ground-truth pre-training the network uses a combination of different losses. In particular, the same feature loss L_f, adversarial loss L_{adv} and pixel loss L_p of [20] are employed as well as a style loss L_s to encourage the reproduction of texture

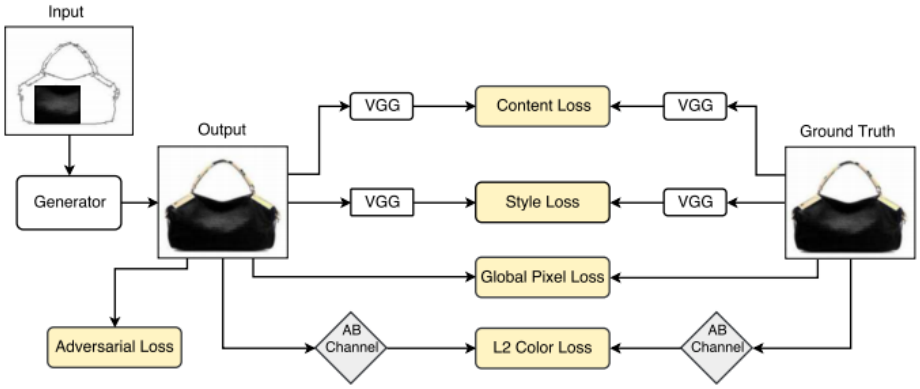


Figure 3.14: TextureGAN ground-truth pre-training [8].

details and a color loss L_c which is a L2 loss between the **ab** channels of the generated result and that of the ground-truth, since the previous losses are applyed only on the **L** channel. To be more specific, the style loss adopt the idea of matching the Gram matrices of the features extracted from certain layers of the pre-trained classification network (VGG-19). The Gram matrix $G^l_{ij} \in \mathcal{R}^{N_l \times N_l}$ is defined as $G^l_{ij} = \sum_k \mathcal{F}^l_{ik} \mathcal{F}^l_{jk}$.

On the other hand, during the external texture fine-tuning the network is fine-tuned to reproduce and propagate textures for which there is no ground-truth output. In this phase the network uses all the losses of the previous one except for the style loss L_s. Feature L_f and adversarial loss L_adv are kept unchanged, while pixel and color losses, $L^{'}_p$, $L^{'}_c$, are changed to compare the generated results with the entire input texture from which input texture patches are extracted. Furthermore, in order to encourage better propagation of texture, a local texture loss L_t is proposed. n patches are sampled from the generated result and the input texture I_t from a separate texture database. The local texture loss is composed by:

$$L_t = L_s + w_p L_p + w_{adv} L_{adv} \tag{3.15}$$

where L_{adv} is a local adversarial loss that decides whether a pair of texture patches have the same textures, while L_s and L_p are style loss and L2 loss on the cropped

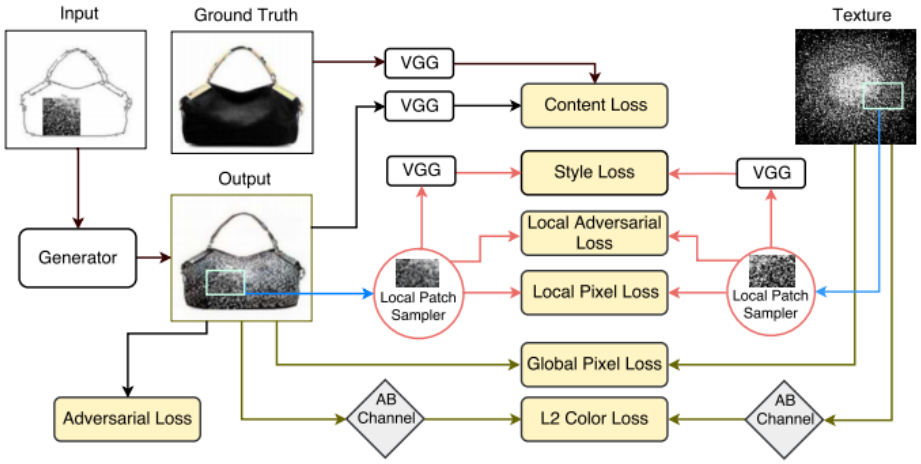


Figure 3.15: TextureGAN external texture fine-tuning [8].

patches.

Some results obtained with TextureGAN can be seen in Fig. 3.16.

3.10 StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation

Previous studies showed amazing results on image-to-image translation for two domains, but they have limited scalability and robustness in handling more than two domains. StarGAN [9] was designed to perform image-to-image translation for multiple domains using only a single model.

Fig. 3.17 shows an overview of the system, that is composed by two modules, a discriminator D and a generator G. D has the goal to distinguish between real and fake images and classify the real images to its corresponding domain. G generates a fake image taking both an image and the target domain as input with the objective of having the fake image classified as target domain by D. In addition to that, G also tries to reconstruct the original image from the fake one given the original domain

Figure 3.16: TextureGAN results on handbag sketches.

label.

The description of the model translates to a combination of different loss functions. First of all, an adversarial loss is used to make the generated images indistinguishable from real images:

$$\mathcal{L}_{adv} = \mathbb{E}_x[\log D_{src}(x)] + \mathbb{E}_{x,c}[\log(1 - D_{src}(G(x,c)))] \tag{3.16}$$

where G generates an image $G(x,c)$ conditioned on both the input image x and the target domain c, while D tries to distinguish between real and fake images. In particular $D_{src}(x)$ is the probasbility distribution over sources given by D.

Secondly, a domain classification loss is used to properly classify the output image y to the target domain c. This loss is decomposed into two terms: the first one is a domain classification loss of real images used to optimize D and the second one is a domain classification loss of fake images used to optimize G. Respectively:

$$\mathcal{L}^r_{cls} = \mathbb{E}_{(x,c')}[-\log D_{cls}(c'|x)] \tag{3.17}$$

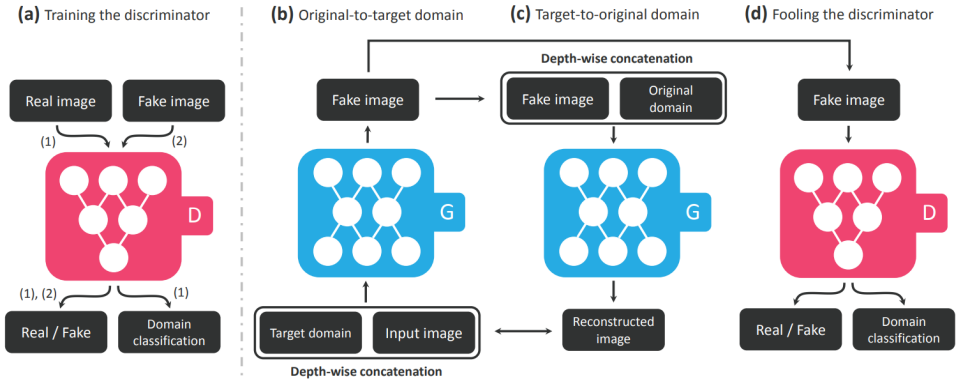


Figure 3.17: Overview of StarGAN [9].

and

$$\mathcal{L}_{cls}^{f} = \mathbb{E}_{(x,c)}[-\log D_{cls}(c|G(x,c))] \tag{3.18}$$

where c' is the original domain and $D_{cls}(c'|x)$ represents a probability distribution over domain labels computed by D.

Finally, a reconstruction loss is introduced in order to preserve the content the input images while changing only the domain-related part of the inputs. This loss is defined as:

$$\mathcal{L}_{rec} = \mathbb{E}_{(x,c',c)}[\| x - G(G(x,c),c') \|_1] \tag{3.19}$$

StarGAN was tested using CelebA dataset [32] and a comparison with previous methods can be seen in Fig. 3.18. It is worth noting that StarGAN is the only method between the ones in the figures that only use a single generator and discriminator still obtaining the best results.

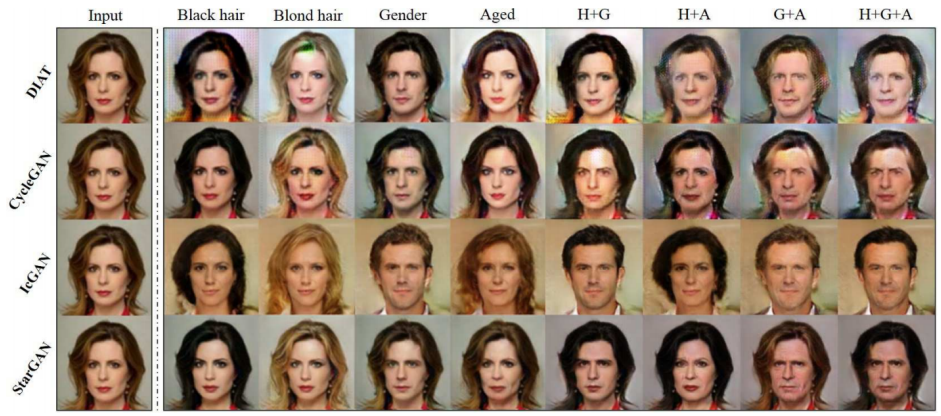
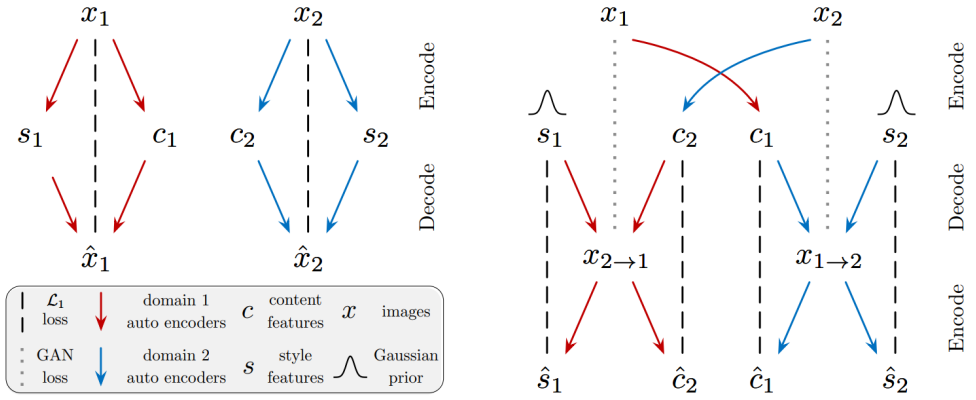


Figure 3.18: StarGAN results compared with previous methods..

3.11 Multimodal unsupervised image-to-image translation

Multimodal Unsupervised Image-to-image Translation (MUNIT)[10] proposed another framework to solve the problem of generating diverse outputs from a given source domain image by combining content and style codes of images. In order to translate an image to another domain, the content code of an image is combined with a random style code sampled from the style space of the target domain.

In Fig. 3.19 is presented an overview of the model and its learning process. The translation model is composed by an encoder E_i and a decoder G_i for each domain $\mathcal{X}_i (i = 1,2)$. The latent code of each auto-encoder is factorized into a content code c_i and a style code s_i, where $(c_i, s_i) = (E_i^c(x_i), E_i^s(x_i)) = E_i(x_i)$. The translation is performed by swapping encoder-decoder pairs. In other words, to translate an image $x_1 \in \mathcal{X}_1$ to $\mathcal{X}_2$, the content code $c_1 = E_1^c(x_1)$ is extracted a random style code s_2 from the prior distribution $q(s_2) \sim \mathcal{N}(0, \mathbf{I})$. Then G_2 is used to produce the final output image $x_{1\to 2} = G_2(c_1, s_2)$.



(a) Within-domain reconstruction (b) Cross-domain translation

Figure 3.19: Munit model [10].

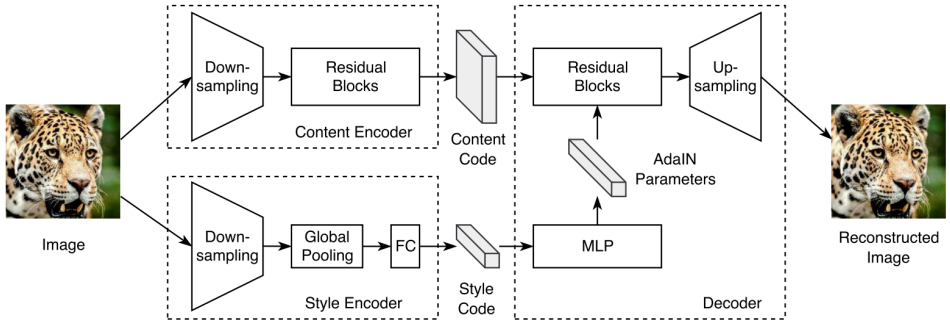


Figure 3.20: The auto-encoder architecture [10].

Fig. 3.20 shows the architecture of the auto-encoder. The content encoder is composed by several strided convolutional layers followed by residual blocks. On the other hand, the style encoder also contains several strided convolutional layer followed by a global average pooling layer and a fully connected layer. A MLP is used to produce a set of Adaptive Instance Normalization (AdaIN) [33] parameters from the style code.

The AdaIN is defined as:

$$AdaIN(z,\gamma,\beta) = \gamma\left(\frac{z-\mu(z)}{\sigma(z)}\right)+\beta \tag{3.20}$$

where z is the activation of the previous convolutional layer, μ and σ are channel-wise mean and standard deviation and γ and β are the parameters generated by the MLP. The main difference from previous approach is that the affine parameters are produced by a learned network.

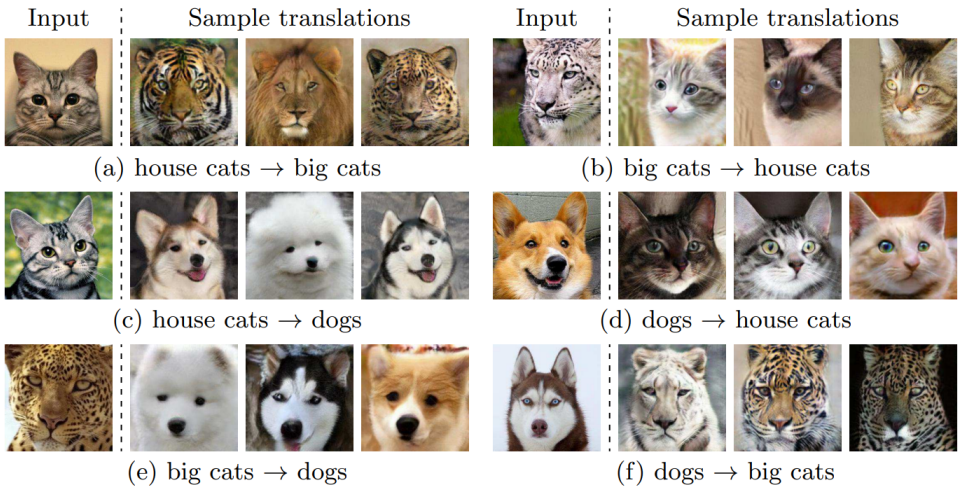


Figure 3.21: Example results of animal image translation.

In 3.21 different examples of animal image translation produced by MUNIT are presented.

3.12 Few-Shot Unsupervised Image-to-Image Translation

Unsupervised image-to-image translation methods usually require access to many images in both source and destination classes at training time. Few-Shot Unsupervised Image-to-Image Translation (FUNIT) [34] was proposed to address this problem and to produce a few shot, unsupervised image-to-image translation method that

can also work on unseen target classes.

In order to train FUNIT, images from a set of object classes (various animal species) are used. The source class images are used to train a multi-class unsupervised image-to-image translation similar to MUNIT[10]. During testing, few images from a novel class, called target class, are provided to the model that has to leverage the few target images to translate any source class image to analogous images of the target class.

The framework is composed by a conditional image generator G and a multi-task adversarial discriminator D. G takes as input a content image x belonging to class c_x and a set of K class images $\{y_1,\ldots,y_k\}$ belonging to class c_y and returns the output image $\bar{x}$ as:

$$\bar{x} = G(x,\{y_1,\ldots,y_k\}) \tag{3.21}$$

During training G learns to translate images between two randomly sampled source classes c_x and c_y with $c_x \neq c_y$, while during testing G maps any of the source classes to a unseen target class.

G consists of a content encoder E_x, a class encoder E_y and a decoder F_x. E_x maps the input content image x to a content latent space z_x, while E_y maps each of the individual class images $\{y_1,\ldots,y_k\}$to an vector and then computes its mean to obtain the class latent code z_y. Like MUNIT, F_x contains several AdaIN residual blocks. By feeding the class latent code to the decoder via the AdaIN layers, the class image controls the global look, while the content image determines the local structure.

On the other hand, the discriminator D is trained by solving multiple adversarial classification tasks simultaneously. Since there are $|\mathbb{S}|$ source classes, D produces $|\mathbb{S}|$ outputs.

The training is executing by solving a minimax optimization problem given by:

$$\min_{D}\max_{G} = \mathcal{L}_{GAN} + \lambda_R \mathcal{L}_R(G) + \lambda_F \mathcal{L}_F(G) \tag{3.22}$$

where $\mathcal{L}_{GAN}$ is the GAN loss, $\mathcal{L}_R$ is the content image reconstruction loss and $\mathcal{L}_F$ is the feature matching loss. Firstly, $\mathcal{L}_{GAN}$ is computed only using the corresponding

binary prediction score of the class. Secondly, $\mathcal{L}_R$ helps G learn a translation model. The idea is that if the same image is used for both the input content image and the input class image, the loss push G to generate an output identical to the input. Finally, $\mathcal{L}_{GAN}$ regularize the training.

Figure 3.22: Example results of Funit. y_1 and y_2 are the class images, x is the content image and $\bar{x}$ is the translation output

Fig. 3.22 presents some FUNIT results.

Chapter 4

Meta Learning

The goal of meta-learning is to train a model on a variety of learning tasks, such that it can solve new learning tasks using only a small number of training samples. There are many types of meta-learners. Some learn how to parameterize the optimizer of the network [35, 22], while others use a network as optimizer [36, 37, 38]. In this chapter will be presented some approaches to meta-learning based on hyper-parameterized gradient descent that are the most relevant to the topic of this thesis.

4.1 Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Model-Agnostic Meta-Learning (MAML) [11] was proposed to have a meta-learning algorithm that is general and model-agnostic. This means that it can be applied to any learning problem and model that is trained with a gradient descent procedure. The key idea of MAML is is to train the model's initial parameters such that the model has maximal performance on a new task after the parameters have been updated through one or more gradient steps computed with a small amount of data from that new task.

Given a model f, that maps observation $\mathbf{x}$ to outputs α a task is defined as $\mathcal{T} = \{\mathcal{L}(\mathbf{x}_1, \alpha_1, \ldots, \mathbf{x}_H, \alpha_H), q(\mathbf{x}_1), q(\mathbf{x}_{t+1}|\mathbf{x}_t, \alpha_t), H\}$, where $\mathcal{L}$ is a loss function, $q(\mathbf{x}_1)$ is a distribution over initial observations $q(\mathbf{x}_{t+1}|\mathbf{x}_t, \alpha_t)$ is a transition distribution and H

is an episode length.

In MAML the model is required to be able to adapt to a distribution over tasks $p(\mathcal{T})$. In the K-shot setting, the model is trained to learn a new task $\mathcal{T}_i$ drawn from $\mathcal{T}_i$ from only K samples drawn from q_i and feedback $\mathcal{L}_{\mathcal{T}_i}$ generated by $\mathcal{T}_i$. At the end of the meta-training, new tasks are sampled from $p(\mathcal{T})$ and meta-performance is measured by the model's performance after learning from K samples. Tasks used for meta-testing are held out during meta-training.

Formally, given a model represented by a parametrized function f_θ with parameters θ, when adapting to a new task $\mathcal{T}_i$, the model's parameters θ become $\theta_i^{'}$. $\theta_i^{'}$ is computed using one or more gradient descent updates on task $\mathcal{T}_i$:

$$\theta_i^{'} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta) \tag{4.1}$$

where α is the step size.

The model parameters are trained by optimizing for the performance of $f_{\theta_i^{'}}$ with respect of θ across task sampled from $p(\mathcal{T})$. Finally, the meta-optimization across tasks is performed via stochastic gradient descent (SGD), such that the model parameters θ are updated as follows:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i^{'}}) \tag{4.2}$$

where β is the meta step size.

The results of this approach is that the model's parameters become very sensitive to changes in the task, such that small changes in the parameters will produce large improvements on the loss function of any task drawn from $p(\mathcal{T})$, when altered in the direction of the gradient of that loss like is presented in Fig. 4.1.

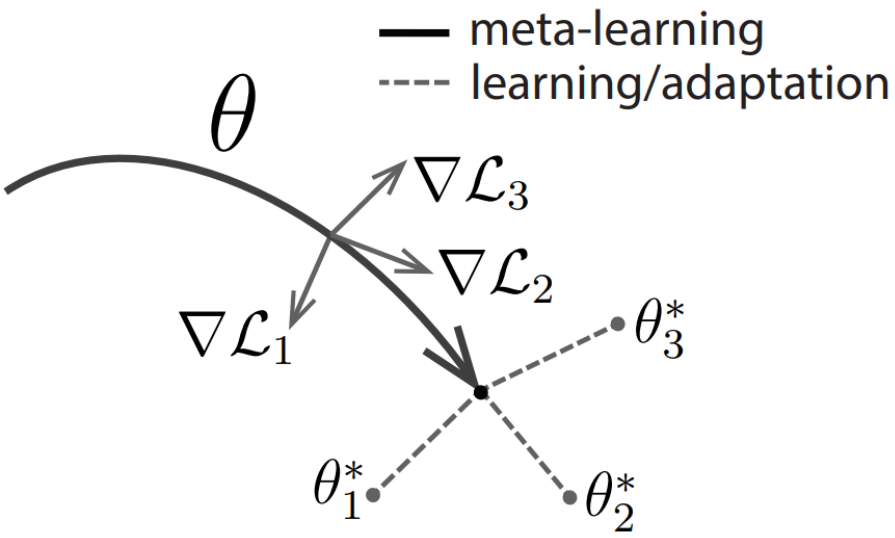


Figure 4.1: Diagram of MAML [11].

4.2 Reptile

Reptile [39], is a first-order meta-learning algorithm very similar to MAML as it learns an initialization for the parameters of a neural network model, such that when these parameters are optimized at test time, the model generalizes from a small number of examples from the test task.

The Reptile algorithm works as follow:

Algorithm 1 Reptile (serial version)

1: Initialize ϕ, the vector of initial parameters
2: **for** iteration = 1,2, ... **do**
3: Sample task τ, corresponding to loss L_τ of weight vectors $\tilde{\phi}$
4: Compute $\tilde{\phi} = U_\tau^k(\phi)$, denoting k steps of SGD or Adam
5: Update $\phi \leftarrow \phi + \varepsilon(\tilde{\phi} - \phi)$
6: **end for**

Reptile is remarkably similar to joint training on the expected loss $\mathbb{E}_\tau[L]_\tau$. Indeed, if

U is defined to be a single step of gradient descent, that is $k = 1$, the algorithm corresponds to stochastic gradient descent of the expected loss. On the other hand, if multiple gradient updates are performed in the partial minimization ($k > 1$), the expected update $\mathbb{E}_\tau[U_\tau^k(\phi)]$ does not correspond to taking a gradient step on the expected loss $\mathbb{E}_\tau[L]_\tau$. Instead, the update includes important terms coming from second-and-higher derivatives of L_τ. For this reason, Reptile converges to a solution that's very different from the minimizer of the expected loss $\mathbb{E}_\tau[L]_\tau$.

4.3 Metagan: An adversarial approach to few-shot learning

MetaGAN [40] is a general and flexible framework for few-shot learning. Given a decent K-shot N-way classifier, a conditional generative model is introduced with the objective to generate samples which are not distinguishable from true data sampled from a specific task. The dimension of the classifier output is increased from N to N + 1, to model the probability that input data is fake. The discriminator (classifier) and generator are trained in an adversarial setup.

The key idea behind MetaGAN is that imperfect generators in GAN models can provide fake data between the manifolds of different real data classes, thus providing additional training signals to the classifier as well as making the decision boundaries much sharper.

	5-way Acc	
Model	1-shot	5-shot
MAML(5 gradient steps)	48.70 ± 1.84	63.11 ± 0.92
MAML(1 gradient steps, first order)	48.07 ± 1.75	63.15 ± 0.91
MAML(1 gradient steps, first order)	43.64 ± 1.91	58.72 ± 1.20
METALGAN + MAML (1 step, first order)	46.13 ± 1.78	60.71 ± 0.89

Table 4.1: Few-shot classification results on Mini-Imagenet.

MetaGAN does not impose restrictions on the design of discriminator. For this reason, it can be adapted from almost any state-of-the-art few-shot learners, like MAML. Some results of this approach trained with Mini-Imagenet dataset (which

is a subset of Imagenet [41]) are presented in Table 4.1.

4.4 Figr: Few-shot image generation with reptile

FIGR [42] was the first attempt of combining meta-learning (reptile in particular) with image generation with the objective of producing very simple black and white images. In addition, the paper also introduces a novel dataset called FIGR-8 consisting in 1,548,944 black and white pictograms, ideograms, icons, emoticons, object or conception depictions categorized in 18,409 classes.

The few-shot image generation problem is then described. The main assumption is having the access to a set of tasks T containing multiple task τ where each of the individual task τ is an image generation problem consisting in one class of images X_τ and a loss L_τ. The objective is to find, using reptile, parameters ϕ that can quickly converge on a random task τ to minimize the associated loss L_τ, that is:

$$minimize_\phi \mathbb{E}_\tau[L_\tau(U_\tau^k(\phi))] \tag{4.3}$$

where $U_\tau^k(\phi)$ is the operator that updates ϕ k times using x_n, that is a total of n data points sampled from X_τ.

Some samples generated by FIGR are presented in Fig. 4.2.

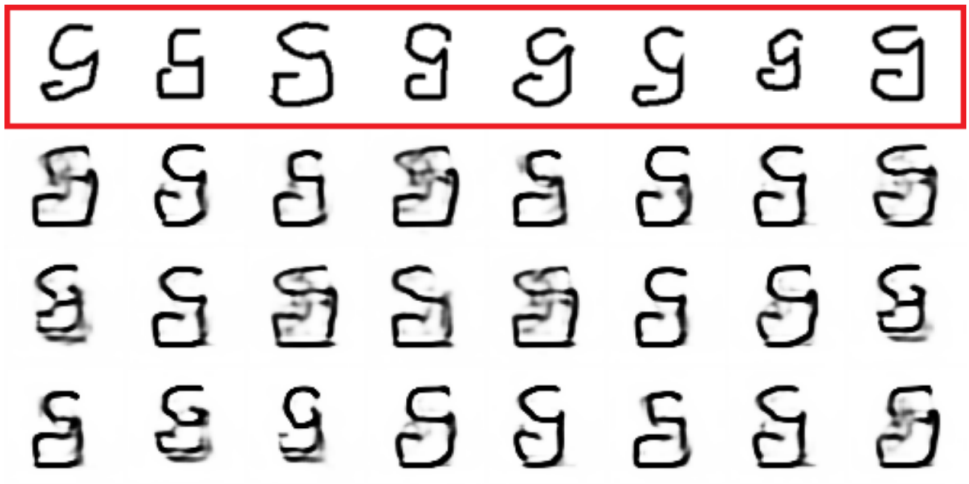


Figure 4.2: Sample generated by FIGR using Omniglot dataset [12]. The row highlighted in red represents the real samples, while the other ones are the generated images.

Chapter 5

Attention Learning

5.1 Visualizing and understanding convolutional networks

The work of Zeiler and Fergus [13] was the first one tackling the problem of understanding the behavior of Convolutional Networks in classification task. It introduced a novel visualization technique that gives insight into the function of intermediate feature layers and the operation of the classifier. The objective requires interpreting the feature activity in intermediate layers by mapping these activities back to the input pixel space.

This mapping is performed with a Deconvolutional Network, that is a convnet model that uses the same component like filtering, pooling, etc., but in reverse, so instead of mapping pixels to features does the opposite. To examine a convnet, a deconvnet is attached to each of its layers, providing a continuous path back to image pixels as illustrated in Fig. 5.1. The feature maps are feeded to the attached deconvnet layer and thn then unpooled, rectified and filtered to reconstruct the activity in the layer beneath.

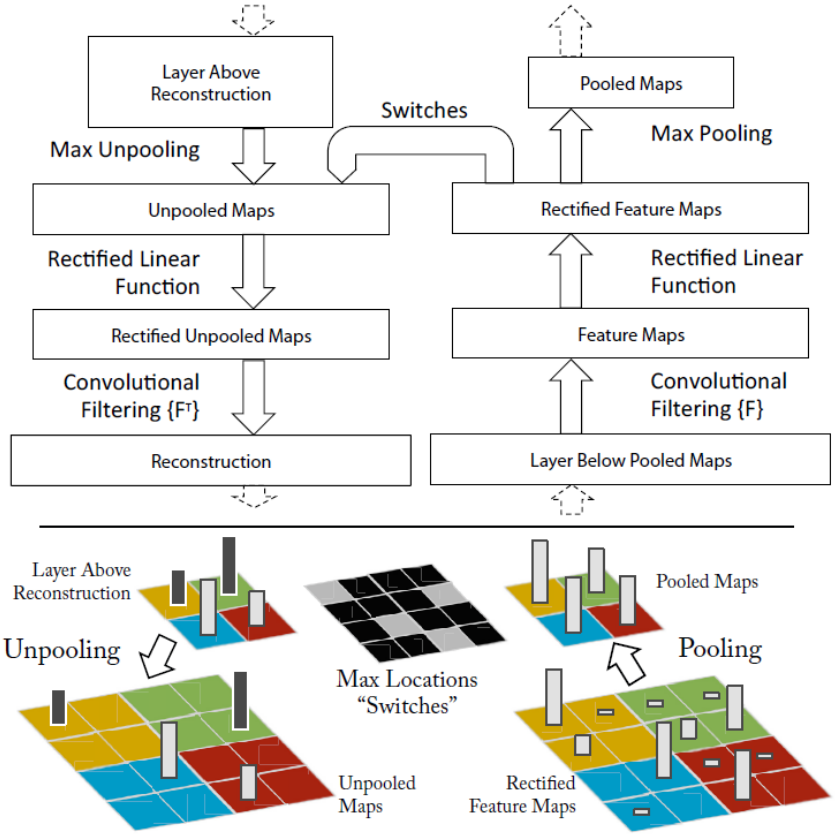


Figure 5.1: Top: A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath. Bottom: An illustration of the unpooling operation in the deconvnet, using switches which record the location of the local max in each pooling region (colored zones) during pooling in the convnet. The black/white bars are negative/positive activations within the feature map [13].

5.2 CAM: Learning Deep Features for Discriminative Localization

Global Average Pooling layers were introduced in [43] to act as a as a structural regularizer, preventing overfitting during training. Their use was extended in [14]

to generate *class activation maps* (CAM). A class activation map for a particular category indicates the discriminative image regions used by the CNN to identify that category.

The network designed to produce this results is a fairly standard classification network with the addition of a global average pooling layer just before the final output layer. The features obtained with global average pooling are then used for a fully-connected layer that produced the desired output. Given this simple connectivity structure, it is possible to identify the importance of the image regions by projecting back the weights of the output layer onto the convolutional feature maps. This technique is called class activation mapping.

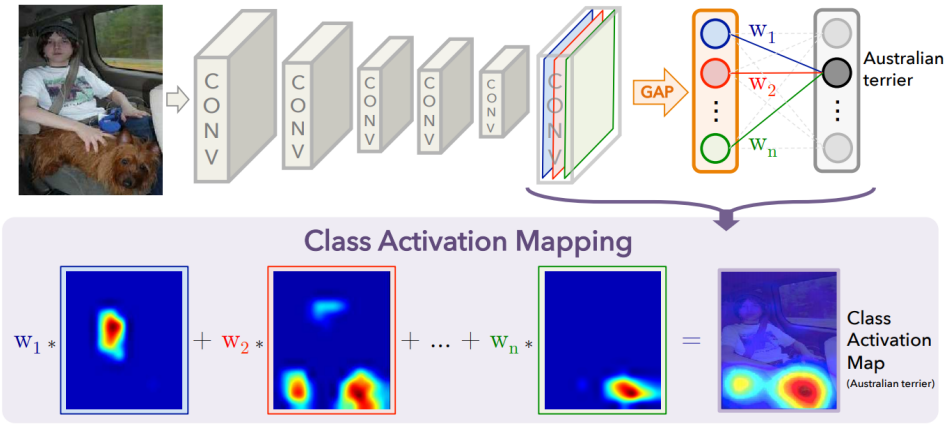


Figure 5.2: Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs) [14].

The whole process is depicted in Fig. 5.2. More specifically, for a given image, let $f_k(x,y)$ represent the activation of unit k in the last convolutional layer at spatial location (x,y). Then, for unit k, the results of performing global average pooling, F^k is $\sum_{x,y} f_k(x,y)$. If F^k is plugged into that class score, S_c, that is a softmax, the following is obtained:

$$S_c = \sum_{x,y} \sum_{k} w_k^c f_k(x,y) \tag{5.1}$$

where w_k^c is the weight corresponding to class c for unit k. M_c is defined as the class activation map for class c, where each spatial element is given by:

$$M_c = \sum_k w_k^c f_k(x,y) \tag{5.2}$$

5.3 Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

Gradient-weighted Class Activation Mapping (Grad-CAM) [15] was introduced as a generalization of CAM that is applicable to a wide variety of CNN model-families like CNNs with fully-connected layers, CNNs used for structured outputs, CNNs used in tasks with multi-modal inputs.

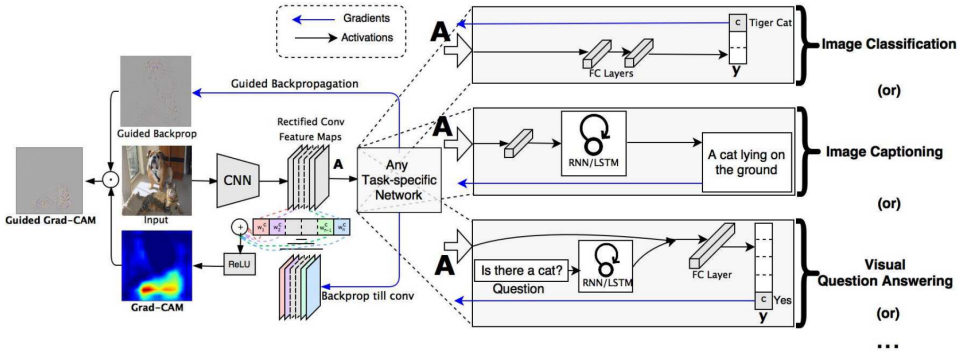


Figure 5.3: Grad-CAM overview: given an image and a class of interest as input, the image id propagated through the CNN part of the model and then through a task specific computation to obtain a raw score for the category. The gradients are set to zero for all classes except the desired class, which is set to 1. This signal is then back-propagated to the rectified convolutional feature maps of interest that is combined to compute the coarse Grad-CAM localization. As a last step, the heatmap is multiplied with guided backpropagation to get Guided Grad-CAM visualizations [15].

As shown in Fig. 5.3 Grad-CAM uses the gradient information flowing into the last convolutional layer of the CNN to understand the importance of each neuron for

a decision of interest. Firstly, the gradient of the score for class c, y^c is computed with respect to feature maps A^k of a convolutional layer, that is $\frac{\partial y^c}{\partial A^k}$. These gradients flowing back are globally average-pooled to obtain the neuron importance weights α_k^c:

$$w_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}} \tag{5.3}$$

The weight w_k^c captures the importance of feature map k for a target class c. Finally, a weighted combination of forward activation maps is performed, followed by a ReLU, obtaining:

$$L_{Grad-CAM}^c = ReLU \underbrace{\left(\sum_k w_k^c A^k\right)}_{\text{linear combination}} \tag{5.4}$$

5.4 Grad-CAM++: Generalized Gradient-based Visual Explanations for Deep Convolutional Networks

Grad-CAM++ [44] proposes a better visual explanations of CNN model predictions than Grad-CAM in terms of better localization of objects as well as explaining occurrences of multiple objects of a class in a single image.

In Grad-CAM++ the weights equation 5.3 is reformulated as:

$$w_k^c = \sum_i\sum_j \alpha_{ij}^{kc} ReLU\left(\frac{\partial Y^C}{\partial A_{ij}^k}\right) \tag{5.5}$$

where α_{ij}^{kc} are the gradient weights. This structure ensure that the weights w_k^c are a weighted average of the gradients as opposed to a global average.

A results comparison between Grad-CAM and Grad-CAM++ is presented in Fig. 5.4.

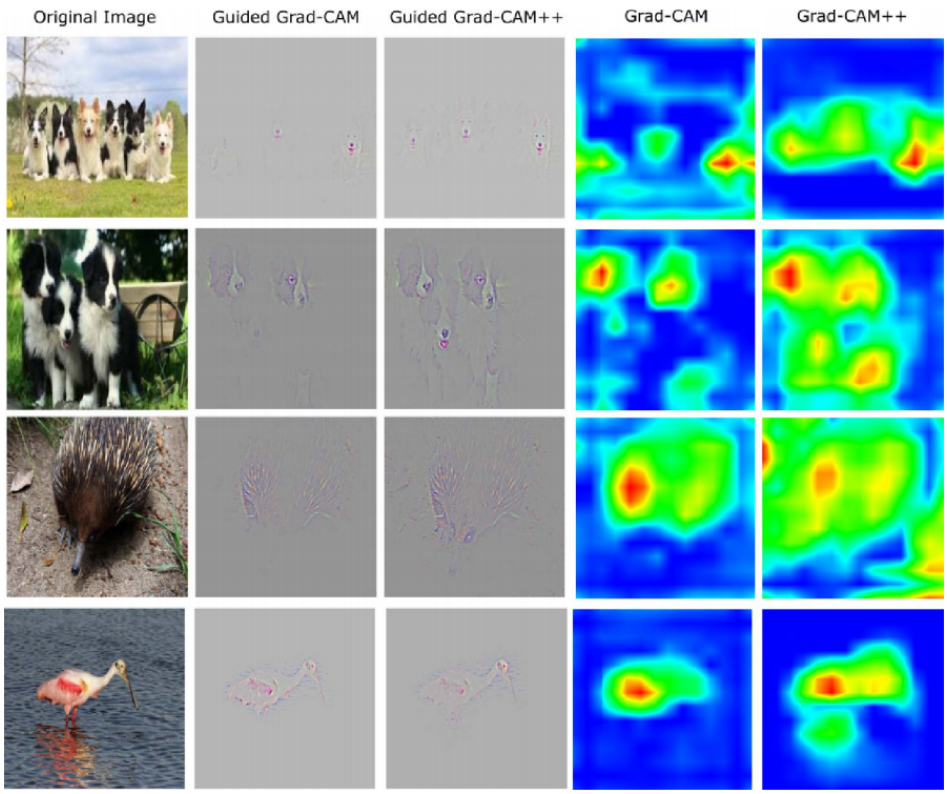


Figure 5.4: Attention maps generated with bog Grad-CAM and Grad-CAM++. Grad-CAM++ has much better results with multiple occurrences of the same class and localization capability of an object in an image.

5.5 Towards Visually Explaining Variational Autoencoders

Recent advances in convolutional neural network (CNN) model interpretability have led to impressive progress in visualizing and understanding model predictions. In particular, gradient-based visual attention methods have driven much recent effort in using visual attention maps as a means for visual explanations. A key problem, however, is these methods are designed for classification and categorization tasks, and their extension to explaining generative models, e.g., variational autoencoders (VAE) is not trivial. This limitation is addressed in [16] that presents a way of generating

visual attention from the latent space of VAE and proves that is possible to use this attention to localize anomalies in images.

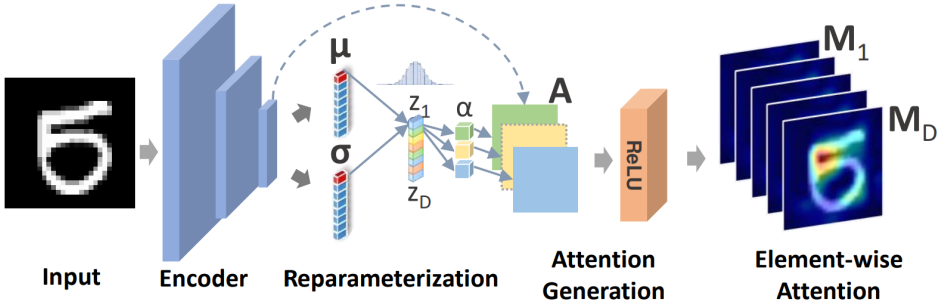


Figure 5.5: Element-wise attention generation with a VAE [16].

In Fig. 5.5 the method for generating attention from the VAE latent space is presented. In particular, given the posterior distribution $q(\mathbf{z}|\mathbf{x})$ inferred by the trained VAE for a data sample $\mathbf{x}$, the reparametrization trick is used to obtain a latent vector $\mathbf{z}$. For each element z_i, the gradients a re backpropagated to the last convolutional feature maps $\mathbf{A} \in \mathbb{R}^{n \times h \times w}$, giving the attention map $\mathbf{M}^i$ corresponding to z_i. Specifically, $\mathbf{M}^i$ is computed as the linear combination:

$$\mathbf{M}^i = ReLU\left(\sum_{k=1}^{n} \alpha_k \mathbf{A}_k\right) \tag{5.6}$$

where $\alpha_k = GAP\left(\frac{\partial z_i}{\partial \mathbf{A}_k}\right)$, with GAP being the global average pooling operation.

As previously mentioned, attention generation can be used to localize anomaly regions given a trained one-class VAE. The idea is to train the VAE on a specific class and then using a testing sample belonging to a different class to generate the attention map that will capture the anomalies between the two different classes.

Fig. 5.6 presents some results for the anomaly detection dataset MVTec Anomaly Detection (MVTec AD) [45].

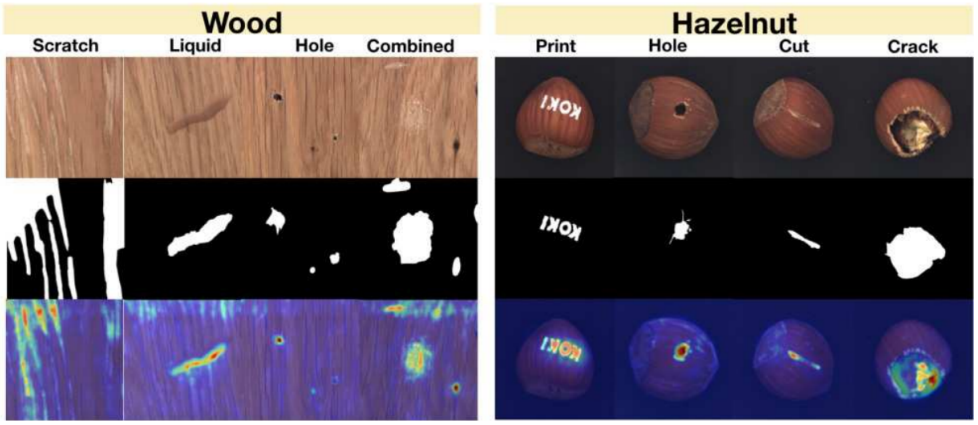


Figure 5.6: Qualitative results for anomaly detection from MVTec-AD.

5.6 Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer

The work of Zagoruyko and Komodakis [17] introduces the idea of using attention to significantly improve the performance of a student CNN network by forcing it to mimic the attention maps of a powerful teacher network.

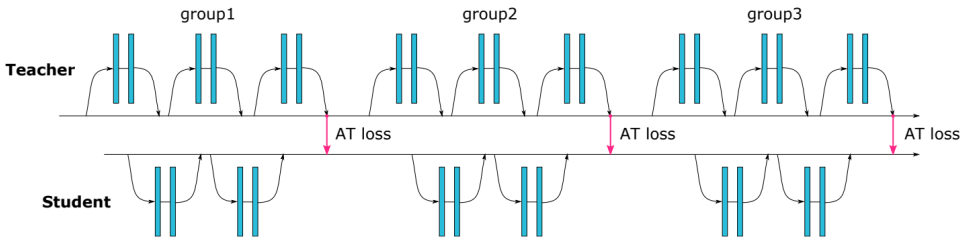


Figure 5.7: Schematics of teacher-student attention transfer [17].

The attention transfer is depicted in Fig. 5.7 and it is imposed by the following

loss:

$$\mathcal{L}_{AT} = \mathcal{L}(\mathbf{W}_S, x) + \frac{\beta}{2} \sum_{j \in \mathcal{I}} \| \frac{\mathcal{Q}_S^j}{\| \mathcal{Q}_S^j \|_2} - \frac{\mathcal{Q}_T^j}{\| \mathcal{Q}_T^j \|_2} \|_2 \tag{5.7}$$

where $\mathcal{Q}_S^j$ and $\mathcal{Q}_T^j$ are the j-th pair of student and teacher attention maps in vectorized form and $\mathcal{L}(\mathbf{W}_S, x)$ denotes a standard cross entropy loss.

5.7 Learning without memorizing

Incremental learning (IL) is an important task aimed at increasing the capability of a trained model, in terms of the number of classes recognizable by the model. The key problem in this task is the requirement of storing data (*e.g.* images) associated with existing classes, while teaching the classifier to learn new classes. To solve this problem, [18] proposes Learning without Memorizing (LwM) that uses an Attention Distillation Loss in order to penalize the changes in classifiers' attention maps as new classes are added.

The LwM approach is presented in Fig. 5.8. The method combines attention distillation loss ($\mathcal{L}_{AD}$) with distillation loss ($\mathcal{L}_D$) and classification loss ($\mathcal{L}_C$). The attention distillation loss is defined in a similar way as [17]:

$$\mathcal{L}_{AD} = \sum_{j=1}^{l} \| \frac{\mathcal{Q}_{S,j}^{I_n,b}}{\| \mathcal{Q}_{S,j}^{I_n,b} \|_2} - \frac{\mathcal{Q}_{T,j}^{I_n,b}}{\| \mathcal{Q}_{T,j}^{I_n,b} \|_2} \|_1 \tag{5.8}$$

where b is the top base class predicted by the student for an image I_n. Fig. 5.9 shows the effect of the application of $\mathcal{L}_{AD}$.

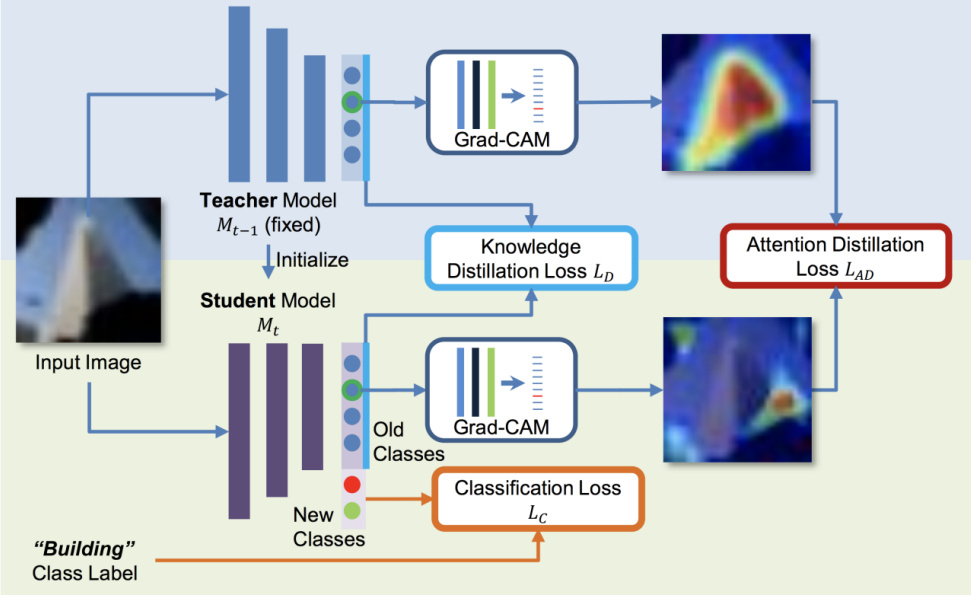


Figure 5.8: LwM architecture [18].

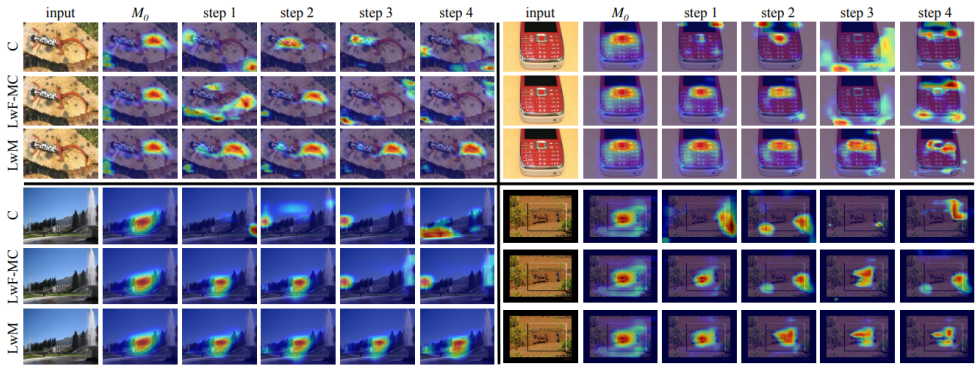


Figure 5.9: The example attention maps generated using attention distillation loss. The column M0 represents the corresponding base-class attention maps generated by the initial teacher model, and the columns step $1 \sim 4$ represent the corresponding base-class attention maps generated in four different incremental steps in temporal order. LwF-MC [19] acts as a baseline.

Part II

Research findings and applications

You have underestimated,
my power

Chapter 6

Shoe model colorization using conditional GANs

Remember:
your focus determines your reality.

6.1 Introduction

Since the first introduction of Generative Adversarial Networks (GANs) [2], one of the main goal that researcher had in mind was to use them in several artistic fields like drawing, painting, sketching and coloring. Nevertheless, using a neural network to completely substitute an artist remains a very challenging task. For this reason, a good compromise is to employs GANs to help creative minds to explore lots of different possibilities by suggesting them new ways of doing things.

This kind of use of GANs is also very suited to help companies product designer since their work involves lots of creativity, but also very repetitive action. Therefore, removing this repetition with the aid of deep learning can help them doing their work even better and faster. More in detail, if we consider a cloth designer, colorizing a sketch, suggesting new color combination or complete a partial colorization of a new cloth design are all tasks that can be performed by a generative network.

In this work, that was funded by Adidas™, the goal was to develop a system able to completely colorize a 2d projection of a partially colored 3d shoe model. The reason is that, after designing a new shoe model, the designer would colorize only one side of the 3d model leaving the other side blank and having a system that would automatically colorize this blank side would allow the work to be much quicker, leaving more time for creative tasks.

In order to solve this problem, a system composed by a custom conditional GAN was developed. This cGAN takes as input both the blank or partially colored side of a shoe model, conventionally the left side, together with the fully colored other side, conventionally the right side, and returns a fully colored left side in a single pass. Experiments were done also conditioning the cGAN in the residual space showing an improvement in the results.

The main contribution of this work are:

- developing a new cGAN architecture that is able to fully colorize shoe images using a colored side as hint to produce the correct colorization;
- an extensive ablation study using different conditioning and losses that shows their benefits and drawbacks;
- a validation of the system over a shoe dataset automatically augmented in order to contain a large amount of different models.

6.2 Related Work

Image colorization literature has mainly focused on coloring grayscale images [46], often guiding the colorization with fixed points of color [47, 20]. On the other side, in this work the objective is coloring a shoe side composed only by its edges using the informations present in the other colored side. This task fall in the image-to-image translation family of tasks where the main source of inspiration can be found in papers like [6, 7]. Since they do not use additional conditioning in order to produce the desired outputs, a different approach can be found in [8] where a texture patch

is used in order to apply the same texture over a blank image. Finally, FUNIT [34] used a target image to change the shape of a source image, but it does not focus on colorization and the output results does not maintain the same shape of the input which is a fundamental requirement of our system.

6.3 Creation of the shoe dataset

Figure 6.1: The 3D model of a shoe in the initial dataset.

The first step in the developing of the system was the dataset definition. The starting point was a set of 160 fully colored 3d shoe models as the one that can be seen in Fig. 6.1. Next, the 3d models were transformed into a set of cube maps containing the 4 2d projections, that are *left*, *right*, *top* and *bottom*. An example of a set of cube maps can be seen in Fig. 6.2.

The next step was to augment the dataset, that is producing a great number of shoes with a new colorization, since 160 images are to few to train a deep neural

Figure 6.2: The cube maps of the 3d shoe model in Fig. 6.1.

network properly. Nevertheless, augmenting the dataset needed to follow some rules: first of all, the color used to produce the new shoe models needs to be taken from a pool of colors that were already present in the initial dataset, secondly the new colorization must be coherent with the intrinsic rules of the shoes (for examples the three stripes that are usually found in Adidas shoes must have the same color) and finally the number of colors present in each shoe needs to be less than a maximum since there is no shoe containing a large amount of colors.

In order to follow all these rules, first of all a list of all colors contained in the initial shoe dataset was created and colors to be applyed on the new shoes were extracted from it. Then, a number was assigned to each part of the shoe: if two parts should have the same color they will share the same number. This allowed, during the data augmentation, to assign the same color to parts that had the same number assigned. Finally the number of maximum colors assigned to each shoe was fixed to 10. Some images produces by the data augmentation algorithm are presented in Fig. 6.3. With the data augmentation algorithm the number of images in the dataset went from 160 to 7000 and a full training of a generative network was possible.

Figure 6.3: Examples of some images produced by the data augmentation algorithm.

6.4 Network architecture

In order to colorize the shoe models in the best way possible, a custom conditional GAN was engineered. The network is composed by a generator and a discriminator following the standard architecture of a cGAN, with the generator taking as input both the blank left side shoe image x and the fully colored right side of the shoe x_s. In addition to that, the generator was also conditioned in the residual space to further push it to correctly colorize the input image. The discriminator takes as input both the output of the generator $\hat{x}$ and x_s in order to classify them as a real or fake tuple.

For the generator architecture, inspiration was taken from the UNet [31] using two downsampling and upsamplig layers with the addition of 4 residual block [48] to improve its performance. On the other side, the Discriminator architecture is the same as [6]. The network architecture can be seen in Fig. 6.4.

In order to fully train the network a set of losses was defined. First of all, an adversarial loss $\mathcal{L}_{adv}$ that consisted in a Wasserstein loss combined with gradient penalty as [5]. Then, a L1 loss $\mathcal{L}_{L1}$ between the output of the generator and the ground

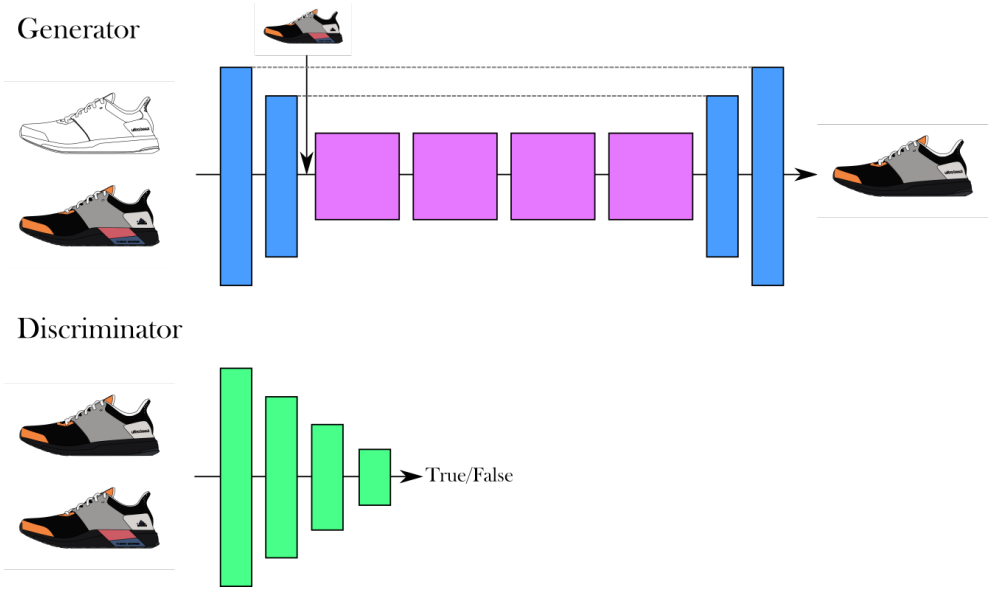


Figure 6.4: Network architecture for shoe colorization. Generator takes as input both the blank shoe image x and the colored other side x_s. In addition to that x_s is also feeded as input to the first residual block of the generator. The discriminator takes as input the tuple composed by the output of the generator $\hat{x}$ and the ground truth x_s.

truth defined as:

$$\mathcal{L}_{L1} = \| G(x, x_s) - \tilde{x} \|_1 \tag{6.1}$$

where $\tilde{x}$ represents the ground truth and $G(x, x_s)$ is the output of the generator G.

6.4.1 Cycle loss

After the definition of the initial network architecture, a new version was also tested introducing a cycle-consistency loss $\mathcal{L}_{cycle}$ to further improve the generated results. For this reason a new generator was employed which translates a colored shoe image to a white shoe image containing only edges. The cycle loss forces the image to be translate back to original input once the colorization happens and it is defined as:

$$\mathcal{L}_{cycle} = \| G_2(G_1(x, x_s)) - x \|_1 \tag{6.2}$$

where G_1 is the original generator network that colorizes the shoe images and G_2 in the new generator that goes back to original white input. This loss allows the generator to produce much sharper and precise results since the colorization and the reconstruction of the output sample needs to be very accurate in order to seamlessly go back to the original input.

6.4.2 Style loss

In order to impose a certain color style to the shoes, some experiments were made using a style loss $\mathcal{L}_{style}$. The style loss is defined as the euclidean distance between the Gram matrix of a feature extracted from a pre-trained network like VGG19 [49]. The Gram matrix results from multiplying a matrix with the transpose of itself:

$$G^l_{i,k} = \sum_k F^l_{i,k} F^l_{i,k} \tag{6.3}$$

Since each columns is multiplied with every row in the matrix, the Gram matrix contains non-localized information about the image, that is style.

The VGG19 layers from which features were extracted in order to calculate the style loss were *conv1_1*,*conv2_1*, *conv3_1*, *conv4_1* and *conv5_1*.

6.5 Experimental results

In this section the experimental results for the shoe colorization will be presented. The experiments were all executed using an Nvidia GTX 1080ti and the network was trained for 200 epochs using Adam [50] as optimizer. Different settings were explored, including the introduction of cycle loss and different combination of weights for the various losses. Finally, LAB color space was used in all of the experiments.

6.5.1 Ablation study

Fig. 6.5 shows a sample generated by a pass in the network without style loss or cycle loss. It can be noticed how the left and right side differs substantially from each other:

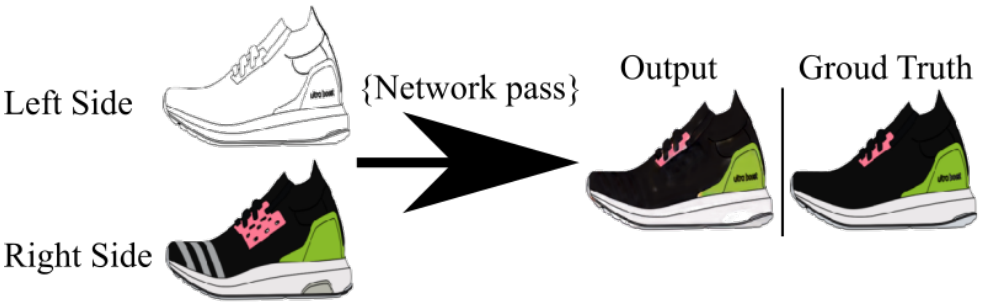


Figure 6.5: Example of the results of our architecture. The white left side is colored using the information contained in colored right side.

first of all, the pink patch over the shoe is much bigger in the right side comparing to the left side; secondly, in the right side there are three stripes that are not present in the left side; thirdly, the soil in the right side has a grey bump not present in the left side and finally, the green zone present some text in the left side but not in the right side.

This kind of differences between the two side of a shoe are very common and network needs to learn to ignore them in order to focus only on the colorization leaving the shape and features of the black side intact. In this particular case, the system does a very good job in colorizing the shoe without transferring the differences between the two sides in the final output.

Now different experimental setups and their results will be presented. In order to quantitatively validate them, FID score [51] was employed to measure how similar are the generated images and the real ones.

Fig. 6.6 illustrates a series of different results that will now be described. On the right of the ground truth the most basic configuration can be found, that is the network with only L1 loss and adversarial loss without any cycle loss or style loss and with no conditioning in the latent space. Even in this basic configuration the obtained results are visually very good. Next, the sample obtained with the style loss is presented. In this case, the style loss actually produces worse results since it encourage the output image to maintain features that are present in the right side. After this sample, the

Figure 6.6: Comparison between different configuration of the system. From left to right are shown: input blank left side, input colored right side, ground truth, output without using the cycle loss or the conditioning in the latent space, output using the style loss, output without the cycle loss but with the conditioning in the latent space, output using cycle loss and latent conditioning, output using cycle loss, latent conditioning and the tuple ($\hat{x}$ and x_s) as input to the conditioning.

output produced by introducing the conditioning in the latent space is presented. The colorization is very good, but, due to the additional conditioning it is more difficult for the network to remove the differences between the left and the right side. This problem is partially solved introducing the cycle loss in the next sample while it is almost completely removed in the last output. In this case the discriminator uses the tuple composed by the generator output and the right side as input. This means that a colored left shoe model will be considered real only if it would match with its other side, improving the overall result by a considerable amount.

After this qualitative evaluation a quantitative one based on FID score is presented in Fig. 6.7.

The quantitative evaluation confirms the qualitative one and further underline how in this case the style loss is not appropriate to produce a correct output that should have very sharp color and precise filling of the various sections of the shoe model.

6.6 Conclusion

In this work, the problem of fashion images colorization was tackled. In particular, the focus was on shoe colorization having at disposal a dataset composed by the

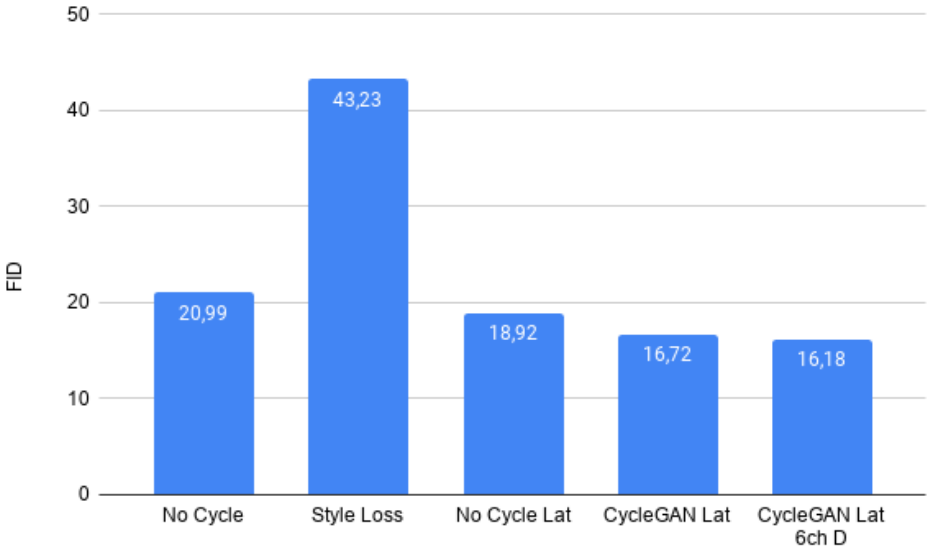


Figure 6.7: FID score for the various experiments presented in Fig. 6.6.

different side of the shoe models extracted from a 3D rendering. A system composed by a conditional GAN conditioned with a colored side of the shoe was developed and various loss were introduced in order to obtain a sharp and precise colorization.

Finally, results were presented showing both qualitative and quantitative results proving the effectiveness of the proposed algorithm.

Chapter 7

MetalGAN - Few shot image colorization

Blind we are, if the creation of this clone army
we could not see.

7.1 Introduction

The *automatic image colorization* task is an image processing problem that is fundamental and extensively studied in the field of computer vision. The task consists in creating an algorithm that takes as input a gray-scale image and outputs a colorized version of the same image. The challenging part is to colorize it in a plausible and well-looking way. Many systems were developed over the years, exploiting a wide variety of image processing techniques, but recently, the image colorization problem, as many other problems in computer vision, was approached with deep-learning methods. Colorization is a *generative* problem from a machine learning perspective. Generative techniques, such as *Generative Adversarial Networks* (*GANs*) [2], are then suitable to approach such a task. In particular, *conditional GANs* (*cGANs*) models seem especially appropriate to this purpose, since their structure allows the network

to learn a mapping from an image x and (only if needed) a random noise vector z to an output generated image y. On the contrary, standard GANs only learn the mapping from the noise z to y.

As many deep-learning techniques, the training of a GAN or a cGAN needs a large amount of images. Large datasets usually grant a great diversity among images, allowing the network to better generalize its results. Nevertheless, having a huge number of images is often not feasible in real-world applications, or simply it requires too much storage space for an average system, and high training computational times. Hence, porting the current deep-learning colorization technologies to a more accessible level and achieving a better understanding of the colorization training process are eased by using a smaller dataset.

For these reasons, one of the aims of this work is to achieve good performances in the colorization task using a little number of images compared to standard datasets. In *few-shot learning*, a branch of the deep-learning field, the goal is to learn from a small number of inputs, or from one single input in the ideal case (*one-shot learning*): the network is subject to a low quantity of examples, and it has to be capable to infer something when posed face-to-face to a new example. This problem underpins a high generalization capability of the network, which is a very difficult task and an open challenging problem in deep networks research.

Recently, some novel interesting ideas highlight a possible path to reach a better generalization ability of the network. These ideas are based on the concept of learning to learn, i.e., adding a meta-layer of learning information above the usual learning process of the network. The generalization is achieved by introducing the concept of *tasks distribution* instead of a single task, and the concept of *episodes* instead of instances. A tasks' distribution is the family of those different tasks on which the model has to be adapted to. Each task in the distribution has its own training and test sets, and its own loss function. A meta-training set is composed of training and test images samples, called episodes, belonging to different tasks. During training, these episodes are used to update the initial parameters (weights and bias) of the network, in the direction of the sampled task. Results of meta-learning methods investigated in literature are encouraging and obtain good performances on some few-shot datasets.

For this reason and since the goal of this work is to colorize images with a few number of examples, a meta-learning algorithm to tune the network parameters on many different tasks was employed. The chosen algorithm is Reptile [39], and it was combined with an adversarial colorization network composed by a Generator G and a Discriminator D. In other words, the proposed method approaches the colorization problem as a meta-learning learning one. Intuitively, Reptile works by randomly selecting tasks, then it trains a fast network on each task, and finally it updates the weights of a slow network.

In this proposal, tasks are defined as clusters of the initial dataset. In fact, a typical initial dataset is an unlabeled dataset that contains a wide variety of images, usually photographs. In this setting, for example, a task could be to color all seaside landscape, and another could be to color all cats photos. Those tasks refer to the same problem and use the same dataset, but they are very different at a practical level. A very large amount of images could overwhelm the problem, showing as much seasides and cats as the network needs in order to differentiate between them. The troubles start when only a small dataset is available. As a matter of fact, such a dataset could not have the suitable number of images for making the network learning how to perform both the two example colorization decently. The idea is to treat different classes of images as different tasks. For dividing tasks, features were extracted from the dataset using a standard approach—e.g., a Convolutional Neural Network (CNN)—and the images were clustered through K-means. Each cluster is thus considered as a single task. During training, Reptile tunes the network G on the specific task corresponding to an input query image and therefore it adapts the network to a specific colorization class.

The problems and main questions that emerge in approaching a few-shot colorization are various. First of all, how the clusterization should be made in order to generate a coherent and meaningful distribution of tasks? Does a task specialization really improve the colorization or the act of automatically coloring a photo is independent from the subject of the photo itself? Second, how the meta-learning algorithm should be combined with cGAN training, also to prevent overfitting the generator on few images? And last, since the purpose of the work is not to propose a solution to

the colorization problem in general, but to propose a method that substantially reduce the amount of images involved in training without—or with minor—losses in state-of-the-art results, how to evaluate the actual performance of the network compared to other approaches? In particular, what are the factors that should be taken in account to state an enhancement, not in the proper colorization, but in few-shot colorization? In the light of these considerations, the contributions of this work are summarized as follows:

- A new architecture that combines meta-learning techniques and cGAN called *MetalGAN* is proposed, specifying in detail how the generator and the discriminator parameters are updated;
- A clusterization and a novel algorithm are described and their ability to tackle image-to-image translation problems is highlighted;
- An empirical demonstration that a very good colorization can be achieved even with a small dataset at disposal during training is provided by showing visual results;
- A precise comparison between two modalities (i.e. our algorithm and only cGAN training) is performed at experimental time, using the same network model and hyper-parameters.

7.2 Related Work

Image retrieval. Since we need the clusterization to be as accurate as possible we reserved a particular attention to the recent image retrieval techniques that focus on obtaining optimal descriptors. Recently, deep learning allowed to greatly improve the feature extraction phase of image retrieval. Some of the most interesting papers on the subject are [52, 53, 54, 55, 56] and, in particular, MAC descriptors [57], that we ended up using.

Conditional GANs. When a GAN generator is not only conditioned with a random noise vector, but also with more complex information like text [21], labels [3], and

especially images, the model to use is a *conditional* GANs (*cGANs*). cGANs allow a better control over the output of the network and thus are very suitable in a lot of image generation tasks. In particular, cGANs conditioned on images were used both in a paired [6] and unpaired [7] way, to produce complex texture [8], to colorize sketches [20] or images [58] and more recently to produce outstanding image synthesis results [59, 60]. In this work, the output must be conditioned by the input gray-scale image, in order to train the network at only generating the colors of the image but not shapes, or the image itself.

Meta-learning. The most relevant meta-learning studies for this work are the Model-Agnostic Meta-Learning (MAML) [11] algorithm and Reptile [39] ones. In particular, we incorporate the Reptile algorithm inside the training phase, allowing the parameters of the generator to be updated in the same fashion as Reptile works. A similar work using MAML is MetaGAN [40], where a generator is used to enhance classification models in order to discriminate between real and fake data, providing generated samples for a task. The main purpose of MetaGAN is not to improve a generative network, but to perform a better few-shot classification, using generated images to sharpen the decision boundary of the problem. On the contrary, in our approach, the generator is fed with task-related images, and the meta-learner is used to enhance the generator itself, instead of a few-shot classifier. Both MAML and Reptile are based on hyper-parameterized gradient descent, and they learn how to initialize network parameters. Other types of meta-learners work differently. For example, there are many algorithms that learn how to parameterize the optimizer of the network [35, 22], or in other cases the optimizer itself is a network [36, 37, 38]. Moreover, one of the most general approach is to use a recurrent neural network trained on the episodes of a set of tasks [61, 62, 63, 64]. The most interesting result of these meta-learners is the achievement of high performance on small datasets [65, 66, 67], or datasets used for few-shot learning (e.g., Omniglot) [12].

Figure 7.1: Some of the results of the clusterization. It is evident how all the images have lots of features in common.

7.3 Algorithm

This section goes in detail within the algorithm we propose. Therefore, each subsection focuses on a different aspect of the method. Then, the complete architecture is explained.

7.3.1 Clusterization of the dataset

In order to exploit Reptile for image colorization we need to treat our image dataset as it would be composed by a series of separate tasks. For this reason we extract features from each image in the dataset using *activation_43* layer of Resnet50. Then, we calculate MAC descriptors by applying max pooling and L2 normalization on the features. Having these MAC descriptors set F, we first apply Principal Component Analysis (PCA) to reduce features dimension from 2048 to 512 and then apply K-means. K-means produces k clusters, and therefore it divides the dataset in k tasks.

Hence, we expect to find, in each of these clusters, images which are similar to each other, accordingly to their features. For example, a cluster could contain images with grass, another one images with pets and so on and so forth. A visual proof of this assumption is showed in Fig. 7.1.

7.3.2 cGAN

As generator architecture, we choose the U-net [31] which is one of the most common for this type of task and we built the discriminator following the classic DCGAN architecture [4], i.e., having each modules composed by Convolutions, Batch Normalization and ReLU layers. Lab is the color space used in this work, because is the one that best approximate human vision and therefore the generator takes as input a gray scale image x_i (the L channel) and outputs the ab channels. Then, we concatenate input and outputs and obtain the final results.

We use L1 loss to model the low-frequencies of our output images and adversarial loss to model the high-frequencies in a similar way of the pix2pix architecture proposed by Isola *et al.* [6].

Therefore, our objective function became:

$$\mathcal{L} = \mathbf{w_{adv}}\mathcal{L}_{\mathbf{adv}} + \mathbf{w_{L1}}\mathcal{L}_{\mathbf{L1}} \tag{7.1}$$

where $\mathbf{w_{adv}}$ and $\mathbf{w_{L1}}$ are weights assigned to the different losses, because we want L1 loss to be more effective than adversarial loss during training.

7.3.3 Meta-learning

As previously briefly mentioned, we approached the generator training with a Reptile meta-learner. This means that, once a task had been chosen, for a fixed number of meta-iterations, the task is sampled and the gradient of the generator loss function (7.1) is evaluated to perform a SGD step of optimization. Fixed the initial generator parameters as θ_G, the inner-loop training defines a sequence $\left(\tilde{\theta}_G^{(j)}\right)_{j=0}^{N_{\text{meta-iter}}}$, where $\tilde{\theta}_G^{(0)} = \theta_G$. Hence it updates the $\tilde{\theta}_G^{(j)}$ parameters in the direction of the task. Once the inner-loop is completed, the parameter are re-aligned with the Reptile rule:

$$\theta_G \leftarrow \theta_G + \lambda_{ML}\left(\tilde{\theta}_G^{(N_{\text{meta-iter}})} - \theta_G\right) \tag{7.2}$$

where λ_{ML} is the step size hyperparameter of Reptile.

7.3.4 Complete architecture of the system

The *MetalGAN* training process is detailed in Algorithm 3.

Algorithm 2 MetalGAN algorithm

for *epoch* **in** $0 \dots N_{\text{epochs}}$ **do**
  for q_i **in** Q **do**
    $K(q_i) \leftarrow$ retrieve_clusters(q_i)
    $\tau(q_i) \leftarrow$ get_task_from_cluster($K(q_i)$)
    for j **in** $0 \dots N_{\text{meta-iter}}$ **do**
      sample $\langle \text{input}, \text{target} \rangle$ from task $\tau(q_i)$
      $\varepsilon_{\text{GAN}} \leftarrow \nabla_{\theta_D} \mathcal{L}_{\mathbf{adv}}$(D(G(input)), label_real)
      $\varepsilon_{\text{L1}} \leftarrow \nabla_{\theta_G} \mathcal{L}_{\mathbf{L1}}$(D(G(input)), target)
      $\varepsilon_G \leftarrow \mathbf{w}_{\mathbf{adv}} \varepsilon_{\text{GAN}} + \mathbf{w}_{\mathbf{L1}} \varepsilon_{\text{L1}}$ ▷ calculates loss gradient
      $\tilde{\theta}_G^{(j)} \leftarrow \tilde{\theta}_G^{(j-1)} - \lambda_G \varepsilon_G$ ▷ updates inner-loop generator parameters
    end for
    $\theta_G \leftarrow \theta_G + \lambda_{ML} \left(\tilde{\theta}_G^{(N_{\text{meta-iter}})} - \theta_G \right)$ ▷ updates generator parameters
    for all $\langle \text{input}, \text{target} \rangle$ **in** $\tau(q_i)$ **do**
      $\varepsilon_{D_{\text{real}}} \leftarrow \nabla_{\theta_D} \mathcal{L}_{\mathbf{adv}}$(D(target), label_real)
      $\varepsilon_{D_{\text{fake}}} \leftarrow \nabla_{\theta_D} \mathcal{L}_{\mathbf{L1}}$(D(G(input)), label_fake)
      $\varepsilon_D \leftarrow \varepsilon_{D_{\text{real}}} + \varepsilon_{D_{\text{fake}}}$ ▷ calculates discriminator loss gradients
      $\theta_D \leftarrow \theta_D - \lambda_D \varepsilon_D$ ▷ update discriminator parameters
    end for
  end for
end for

The algorithm is parameterized by the number of epochs N_{epochs}, the number of meta-iterations $N_{\text{meta-iter}}$, the generator and discriminator learning rates λ_G and λ_D, the Reptile step size parameter λ_{ML}, and the loss weights $\mathbf{w}_{\mathbf{adv}}$ and $\mathbf{w}_{\mathbf{L1}}$. During training, we randomly select a query set $Q = \{q_0, \dots, q_z\}$. Each query q_i corresponds to a single cluster $K(q_i)$. It is worth noting that two queries could point to the same cluster. Having this set, we are able to pick z different images at each epoch by sampling the task $\tau(q_i)$ and to update the generator G as showed in Fig. 7.2.

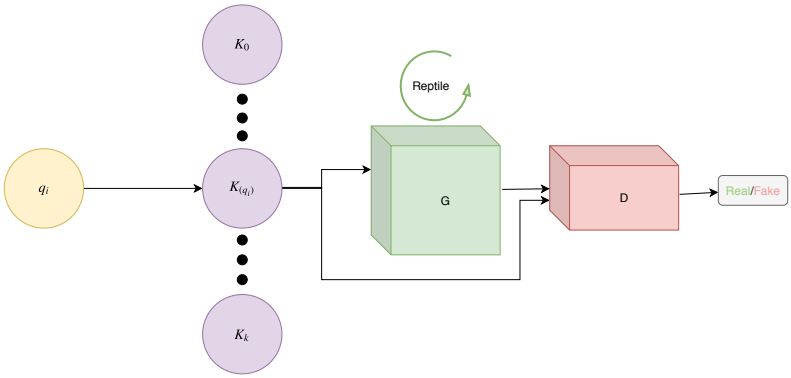


Figure 7.2: The MetalGAN architecture: the query q_i points to a cluster $K(q_i)$ that is used as a task to train the Generator G with Reptile.

The generator is updated by evaluating gradients of its loss functions (adversarial loss $\mathcal{L}_{\text{adv}}$ and L1 loss $\mathcal{L}_{\text{L1}}$), and by adding them to obtain the error ε_G. Then, the network parameters obtained in the inner-loop $\tilde{\theta}_G^{N_{\text{meta-iter}}}$ are used to update the outer-loop generator parameters θ_G. In the last step, all images of the task $\tau(q_i)$ are used to train the discriminator, calculating the gradients of the discriminator adversarial and L1 losses, and adding them to obtain the discriminator error ε_D. The discriminator parameters θ_D are updated consequently.

7.4 Experimental Results

For our experiments we choose a slightly modified version of Mini-Imagenet [22]. Since our goal is not classification, we create our training and test set using only images from the 64 classes contained in the training section of Mini-Imagenet. The total number of images in the dataset is 38392. We define two sets of experiments: the first one consists in training the cGAN without the use of Reptile and the second one introduces Reptile and the features clusterization. For both of them we set $\mathbf{w_{adv}} = 1$ and $\mathbf{w_{L1}} = 10^2$. Learning rates of both the generator and the discriminator were set to $\lambda_G = \lambda_D = 10^{-4}$. For K-means clusterization, the parameter k was set to 64 in order to have clusters as much disjoint as possible. For Reptile, we use 100 *meta-iter*, and

Figure 7.3: Results obtained using the cGAN only. Each group of three images is composed of the input of the network (gray scale image), the ground truth, and the output of the network.

a step size $\lambda_{ML} = 10^{-3}$. The 10% of the dataset images are used as query images. The number of epochs was set to 200. All tests have been executed on a GPU Nvidia 1080 Ti.

7.4.1 cGAN results

In Fig. 7.3 are reported some results produced after the training of the cGAN without the clusterization and without Reptile, i.e., with a standard adversarial algorithm. The training data at disposal are very scarce ($\sim$38k images compared to 1.3M of the whole Imagenet dataset) and, for this reason, the network is not able to produce compelling results. In particular, the network often fails to understand the difference between foreground and background objects and therefore it applies the colors without following edges and borders. In general, for the cGAN is very difficult to propagate the color correctly and is more common the tendency to apply uneven patches of color. Finally, due to the scarcity of data, the network cannot generalize in an acceptable way and hence the colors in the outputs are not sharp, but, on the contrary, the produced results are very blurry and often colors are applied almost randomly.

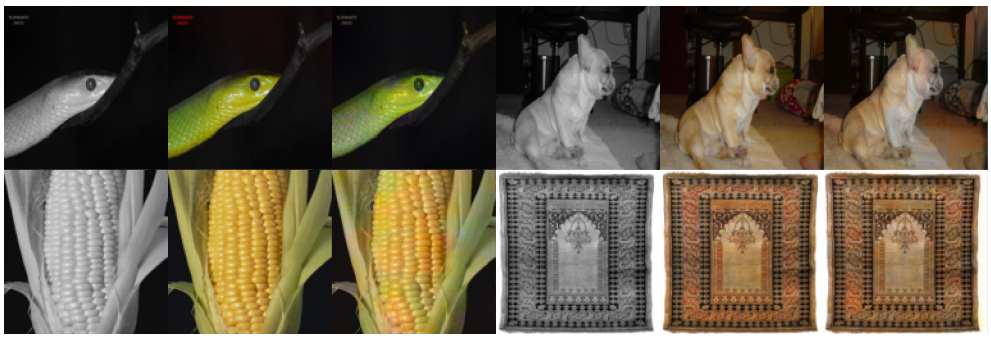


Figure 7.4: Results of MetalGAN. Each of three images consists of the gray scale input given to the network, the ground truth, and the output of the network. The four represented images belong to different clusters.

7.4.2 MetalGAN results

Results of MetalGAN are showed in Fig. 7.4. It is immediately evident how Reptile improves the results of the cGAN. In particular, colors are sharper and more bright. The reason is that Reptile tunes the generator on each cluster and therefore allows the network to focus more on the more predominant colors present in each task and, as a consequence, even with few examples the produced results are compelling and plausible. For example, in a task with lots of images containing grass or plants there will be an abundance of different shades of green and thus the network will learn very quickly to reproduce similar colors over the test set. On the contrary, an image that is very different from the majority of images in the rest of its task could be colorized poorly. This problem, however, is not very frequent since the difference has to be very large in order to produce nasty results.

Other examples can be found at `implab.ce.unipr.it/?page_id=1011`.

7.4.3 Quantitative evaluation

In order to evaluate the quality of the generated samples, we used the Inception Score [68], because it is a very good metric to simulate human judgment. We calculated the Inception Score of generated images using both cGAN and MetalGAN (see Ta-

Dataset	Mean	Std
cGAN	3.20	0.83
MetalGAN	9.16	1.12

Table 7.1: The Inception Scores are computed on generated images from the MiniImageNet dataset, mean and standard deviation are reported for both cGAN and MetalGAN results.

ble 7.1). The score also measures the diversity of the generated images, so a high score is better than a lower one. The MetalGAN approach significantly improves standard cGAN score.

In normal adversarial generative settings, having few images at disposal during training produces a complete failure in the colorization. In this work, we proposed a novel architecture which mix adversarial training with meta-learning techniques, called MetalGAN. As shown by experimental results, even with few images the network trained with MetalGAN was able to produce a well-looking colorization. The clusterization of the dataset and the use of clusters as tasks help at directing the colorization to the most probable suitable colors for the image, and meta-learning allows to train the network on few examples. As future developments, we plan to include the discriminator in the meta-learning training phase, and to test the method on other small datasets in order to prove the generalization capability of the proposed MetalGAN architecture.

This work led to the following publication: *Tomaso Fontanini, Eleonora Iotti, and Andrea Prati. "MetalGAN: A Cluster-Based Adaptive Training for Few-Shot Adversarial Colorization." International Conference on Image Analysis and Processing. Springer, Cham, 2019.*

Chapter 8

MetalGAN - Multi-domain label-less image-to-image translation

There's always
a bigger fish

8.1 Introduction

Image generation or synthesis consists in the act of producing a novel image—representing a subject of interest or whatever else—from an input that could be a random noise matrix, another (real) image, or a combination of these two possibilities, eventually put beside a label or a condition that somehow controls the output. The required output should belong to a specific domain, or it should have been obtained following a precise style. Conversely, in some cases the image domain or style could be not decided *a-priori*, and the developed system should perform multi-domain image generation.

Since the recent advances of deep learning techniques and architectures on image generation, the image synthesis task has become more and more accessible and

understood. Examples of such techniques are the use of *Generative Adversarial Networks* (*GANs*) (in particular *Deep Convolutional* GANs, called DCGANs) and *conditional GANs* (*cGANs*), which should have different architectures, such as *U-Nets* and *StyleGAN*, and different training methods, like *pix2pix*, *cycleGAN*, and so on. A high number of specific approaches were developed to face the aforementioned variety of image generation tasks, leading to a vast literature for each specific sub-problem. A brief outline of the major methodologies are reported in the next section.

This work focuses on the specific problem of image-to-image translation. Image-to-image translation is the act of transforming an arbitrary image in another, more useful, representation of the same data. Image colorization, semantic segmentation, style transfer are examples of image-to-image translations. In particular, this work approaches the task of transforming the input image over a range of so-called *domains*, i.e. recognizable sets of similar images which share common characteristics. One of the scope of this work is to develop a single architecture that is able to handle many different domains, i.e. a multi-domain image-to-image generator system. More specifically, in a system like that, the architecture is required to learn multiple mapping functions, that is one for each domain. In our case, the multiple domains are represented by different facial attributes like “blond hair” or“pale skin” and the mapping functions have the objective to apply these attributes on any face passing through the architecture. Moreover, the idea is to make the model capable to learn new, unseen, domains by using few images for each new domain, in order to gain a great flexibility and generalization capability of the proposed architecture and to tackle also those applications or domains where there is scarcity of available data. Hence, the topic covered in this work is *multi-domain image-to-image translation*, deeply entranced with the concept of domain adaptation.

One interesting take on this topic is that many of these image-to-image transformations are linked by a common way of working. For example, changing hair color, e.g. switching the domain to the new one of “people with blond hair”, needs to correctly segment hair in the same way of changing the domain in “people with black hair”. Similarly, changing a face into its older version and add glasses to a face both need to correctly locate the subject’s eyes. Yet, for long time, a single neural network

for each of these tasks had to be created, even if the tasks were quite similar. A solution to this kind of problem has been proposed with StarGAN [9], which had the intuition of bringing together multiple image-to-image transformations in the same network architecture.

Another observation is that most of the existing approaches to image-to-image translations perform a full training with input-output examples of images, to achieve high quality results. An evident drawback of this approach is that they need very large datasets to be trained. Dataset could be labeled or unlabeled, but usually the domain switch is controlled by a conditioning label, which indicates the target domain to transform the image to. Hence, image-to-image translation often requires a lot of labeled images, where the label denotes the domain(s). It is worth noting that an image could have more than one label: this is the reason for not addressing labels as classes.

Regarding the adaptation of the model to new domains with few images, a similar issue is the few-shot learning problem. Few-shot problems are often addressed by using *meta-learning* techniques, thanks to their ability to switch among a distribution of tasks during training. Training in a meta-learning settings means creating a learning system that includes another learning sub-system: the sub-system trains a model (such as a neural network) on a single task sampled from the distribution, and the meta-learning system trains the sub-system, thus adapting the model to all tasks. Meta-learning methods have proved to be successful in classification and regression scenarios, but there are still few papers [34, 40] in the field of image generation.

Linked to domain adaptation, another known problem of traditional training settings is that once a new set of tasks emerges, e.g. a new domain is added to the target (or desired) outputs, a full retraining of the whole system is needed. This happens even if the new task is similar to tasks that the network has already learned. The full re-training includes incorporating the new domain in the input examples and also it often needs architecture changes.

The main proposal of this work, and its principal contributions are:

- a system that consists in a single cGAN (i.e., two networks, a generator and a discriminator) performing image-to-image translation, trained on multiple

domains;

- both networks do not contain any reference to the label or domain of the input or output (*label-less*) therefore allowing a much more flexible architecture;
- the system is able to switch task with just few examples of a new, unseen domain, by means of a meta-learning training algorithm. This was impossible in previous architectures, representing a great limitation;
- the system uses knowledge accumulated at training time with well-known, largely represented classes, to easily learn new, unknown tasks in few iterations.

Taking into account all these contributions and the main proposed idea to fuse together meta-learning and GAN, we named our approach *MetalGAN*.

8.2 Related Work

Image-to-image translation. The main topic of this paper, i.e. image-to-image translation, has become a hot topic in machine learning researcher community after the introduction of encoder-decoder networks like U-Nets [31], *Fully Convolutional Neural networks* (*FCN*) [69] and conditional GANs [3]. The GAN approach, that can be considered a form of Artificial Curiosity as interestingly stated in [70], to image synthesis has proven an unprecedented quality of output results, reaching photorealism in many domains, such as face synthesis. While traditional GANs [2] generate images from noise, conditional GANs (cGANs) [3] in their many variations are able to generate images from labels or other input images, or both. To this extent, cGANs are often used to perform lots of different image-to-image translation tasks like producing sketch colorization and texture generation [20, 8], super-resolution of images [71] or to generate a photo-realistic image from a semantic label map [59, 60]. cGANs can be trained in both a paired [6, 72] or unpaired way [7, 73, 74].

In our approach, we use cGANs without a paired dataset with only input image but label-less, in order to maintain a great generalization capability of the generator

network. Moreover, we introduced skip connections in the generator network, as in U-Nets.

Multi-domain image-to-image translation. A common trait of most of the image-to-image methods is that they are only able to produce outputs belonging to a single domain or class. Regarding multi-domain facial attributes transfer, our main work of reference is StarGAN [9], though there exist other relevant works like [75, 76, 77]. StarGAN proposes an unified method for multi-domain image-to-image translation. It achieves great results in image synthesis taking strength from the multiple domain adaptations and it learns multiple domains at the same time using only one underlying representation. The main differences between StarGAN and the proposed method are: in our approach, networks do not use labels information (while StarGAN do); our training method relies on a small number of images per-iteration; and also a few-shot-like approach is employed when dealing with new domains during inference.

Few-shots learning. Few-shot problems are usually tackled with meta-learning techniques, since recent results show great performance of meta-learners on typical few-shot datasets and learning settings. There are many types of meta-learners. Some learn how to parameterize the optimizer of the network [35, 22], while others use a network as optimizer [36, 37, 38]. Furthermore, using a recurrent neural network trained on the episodes of a set of task is one of the most general approach [61, 62, 63, 64]. For our work, the most relevant meta-learners are the ones based on hyper-parameterized gradient descent such as Reptile [39] and MAML [11]. In fact, we use the Reptile algorithm applied to a generation problem, where Reptile tasks are identified with our domains. Reptile was already used in combination with GANs in [42] in order to generate very simple black and white images (such as MNIST digits) or in [40] that introduced an adversarial discriminator, conditioned on tasks.

Regarding few-shot image-to-image translation, a new method was recently introduced in [34], coupling an adversarial training scheme with a novel network design. Unlike our method, it does not use meta-learning and does not act as a proper domain

transfer algorithm, but rather as a style transfer one: for example, in the case of face image translation task, the translation output maintains the pose of the input content image, but the appearance is similar to the the faces of the target person.

8.3 Overview of the System

8.3.1 Idea and Notations

As briefly outlined in the introduction, there are some key points from which our work originates, namely, the need of a *few-shots* setting, the use of a *single* GAN architecture, the *absence of labels*, and the *multi-domain* adaptation. All these key points require a proper definition.

Starting from the most potentially ambiguous definition, we call a "domain" a set of images which share a well-defined common characteristic, clearly recognizable by using a single label or keyword: for example, "black-hair" in a dataset of faces denotes the domain of people with a black hair color. Given the example above, it is also clear that the type of dataset is also important: if the dataset contains both dogs and cats images, "black-hair" should have another meaning; if it contains only landscapes, "black-hair" should have no meaning at all. Moreover, domains are not mutually exclusive, rather they could intersect each other.

Closely related to the concept of domain, there is the concept of "label". Usually, when approaching multi-domains problems, labels are employed to identify which domains a certain image belongs to. This helps the networks in detecting a target domain and thus generating images belonging to such a domain. In our case, *label-less* means that the domain of the input and the target images have to be inferred from other information.

In a classic few-shots classification setting, from which we borrow the notations, there are n classes $\{c_1, c_2, \ldots, c_n\}$ and a certain number k of input examples per-class, e.g. $\{x_1, x_2, \ldots, x_{k_i}\}$ are the input of the i-th class c_i. During training only N classes per iteration are used over the total number of classes n, and for each of these N classes only K input examples over the total number of examples of a class are used, where $K << k$. Then, the trained N-classifier has to classify a new example

of a random class $\tilde{c}$. In our case, domains are treated as classes, and the generator-discriminator (from now on, called G and D) networks are trained on a single domain per meta-iteration ($N = 1$), in order to make G and D able to learn the domain they are working on, without labels. The number K of examples per domain varies according to the type of experiment performed (see Section 8.4), but we choose to perform an almost full training and a few-shot inference to allow G to learn adequately the reconstruction of images, and then to switch domain. The architecture of our multi-domain GAN is detailed in Section 8.3.2.

Finally, in the meta-learning nomenclature, we defined a *task* as a group of K images that belong to the same domain, used for the inner-iteration of the algorithm, explained in detail in Section 8.3.3.

Our approach uses a single GAN on different tasks. This forces the underlying weights structure of both G and D networks to learn a general yet effective representation for describing all tasks. G and D networks are conditioned with the use of a meta-learning algorithm, on each task/domain. Other approaches, like StarGAN, instead, needs target labels that condition the output for both G and D networks. In detail, the conditioning is implicitly provided by the task selection performed during meta-learning. For each meta-iteration, a single task is selected, and the network is trained on that single task for a number of internal iterations. In the next meta-iteration the training is performed on another, different but related task. With this training algorithm the network learns, meta-iteration after meta-iteration, a representation that is good (but not optimal) in performing all tasks, and just needs a little final push (few epochs of training) to be moved in the direction of the target task.

8.3.2 Architecture of the Network

One of the strengths of our proposal is that it completely removes the need of providing specific labels for the data, because the network does not use one-hot labels or similar. If data are already labeled, labels are only useful in the preprocessing phase for dividing into domains the dataset, since the main algorithm works task-by-task. It is worth noting that such domains could overlaps. On the other hand, unlabeled data has to be clustered into domains, a passage that can be completely automated (in

contrast with manual labeling), but using a clustering method, domains does not overlap. A clusterization followed by a meta-learning approach is shown in a preliminary work on colorization [78].

Our system is composed by a single cGAN. In particular, since the objective is to generate the face of a person with only a bunch of new attributes, without changing the peculiar traits of the person itself and without using labels, we conditioned the cGAN with the input face, in order to maintain the identity of the person, and, at the same time, changing the target attributes.

The generator network G is the same as the StarGAN one with the addition of skip-connections (inspired by the classic U-Net), but input labels are removed. The introduction of skip-connection in the generator architecture aims at enhancing the quality of the reconstruction; in other words, they are useful for keeping contents of the input image unchanged in the output image (for example, the face of a person remains the same, despite the changing of hair color).

On the other side, the D structure is the PatchGAN from pix2pix [6]. Since during each task the network tunes itself on a single domain, there is no need of a domain classification output for our discriminator. Instead, we choose to classify images both as real or fake and as belonging or not to the current domain. By doing so our discriminator has two outputs: $D_{adv}(x)$ and $D_{dom}(x)$, one for each probability distribution.

Finally, we define a set of losses in order to train our architecture.

Adversarial Loss For the discriminator, we use an adversarial loss to distinguish between real and generated images:

$$\mathcal{L}_{\text{adv}}(D,G) = \mathbb{E}_{y\sim p_\tau}\left[\log D_{adv}(y)\right] + \mathbb{E}_{x\sim p_{\text{data}}}\left[1-\log D_{adv}(G(x))\right], \tag{8.1}$$

where y is sampled over a distribution of current task images p_τ (real samples), and x over the distribution of the whole dataset p_{data} ($G(x)$ are the generated samples).

In particular, during each task, D tries to classify if an image (or a batch of images) y belongs or not to the current domain distribution τ (all images in the batch must belong to the domain). For example, if the current task is to produce people with blond hair, the discriminator has to determine if an image contains a person

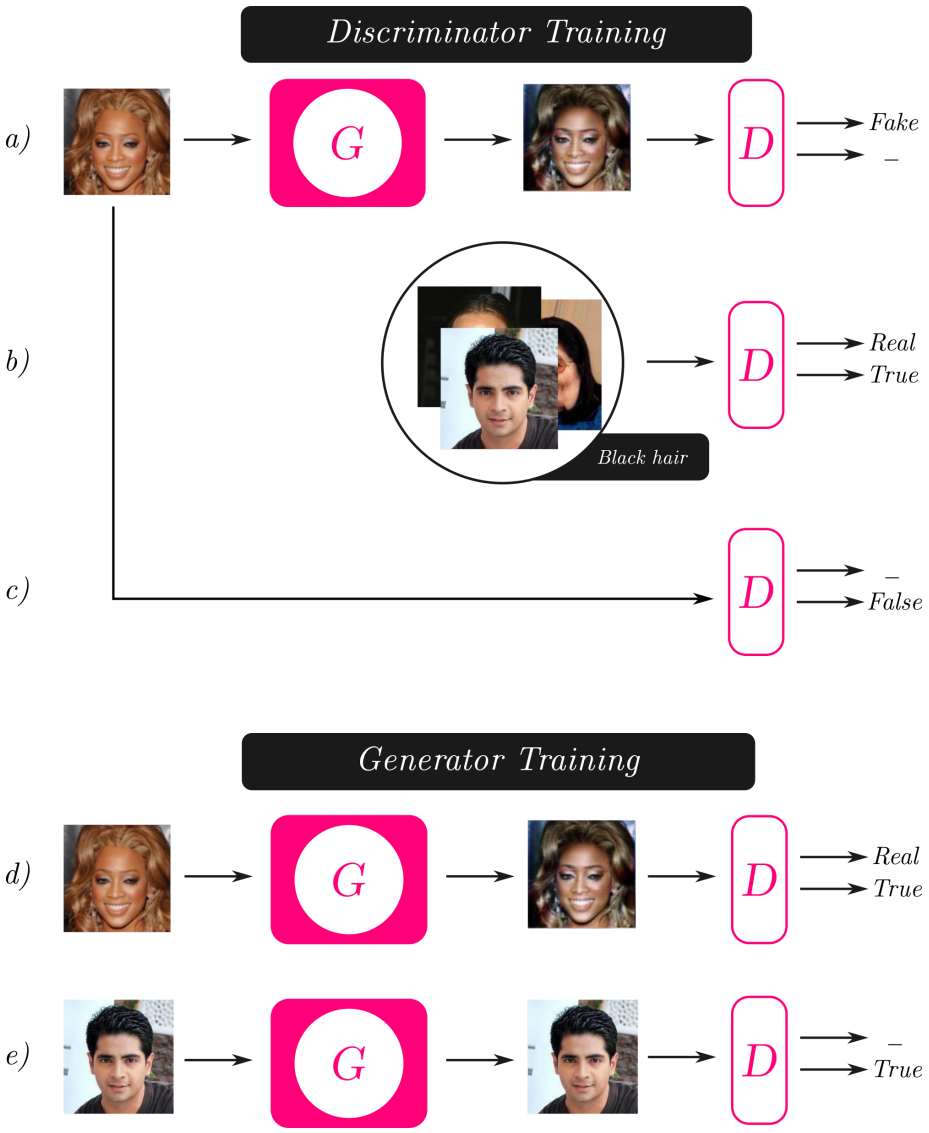


Figure 8.1: Complete network architecture. First, the discriminator is trained to distinguish between fake images (a) and real ones (b) and between images belonging to the current domain (b) and images that do not belong to it (c). Then, the generator is trained to fool the discriminator by labeling its outputs as real and as belonging to the current domain (d). Finally, the reconstruction step is executed and its results are labeled as part of the current domain (e).

with blond hair or not, while in classic adversarial settings it should simply decide if the image contains a face or not. The adversarial loss formula (8.1) does not reflect explicitly this aspect, since the only difference is the nature of the input y: in our work, y is not concatenated to any label.

Domain Loss After we select a new task, during the training of the discriminator, we want images sampled from the current task to be classified as such and, on the other side, images sampled from the whole dataset to be classified as not belonging to the current task.

$$\mathcal{L}'_{\text{dom}}(D) = 2 \cdot \mathbb{E}_{y \sim p_\tau} [\log D_{dom}(y)] + \mathbb{E}_{x \sim p_{\text{data}}} [1 - \log D_{dom}(x)], \tag{8.2}$$

where the multiplicative factor before $\mathbb{E}_{y \sim p_\tau} [\log D_{dom}(y)]$ is motivated by the fact that we need to take into account that an image x may also belong to the domain identified by the current task, since it is drawn from the whole data distribution. For this reason, the first part of the equation strongly reinforces the classification of examples of the target domain, while the second part weakly penalizes every domain (that is, also the target one).

Instead, during the generator training, the goal is that all the generated images, even the ones obtained from the reconstruction of the input, would be classified as belonging to the current task.

$$\mathcal{L}''_{\text{dom}}(D, G) = \mathbb{E}_{x \sim p_{\text{data}}} [\log D_{dom}(G(x))] + \mathbb{E}_{y \sim p_\tau} [\log D_{dom}(G(y))] \tag{8.3}$$

Adversarial and domain losses are visually described in Figure 8.1.

Reconstruction Loss This loss is crucial to guarantee that the generator maintains the content information of the source image. G has to be already tuned on the current domain/task in the meta-learning training. Since we completely removed the labels from our architecture and we tune the network on a new domain at each iteration, we cannot use a cycle consistency loss like in StarGAN, because G is able to produce images of only one target domain each task. The solution we choose to adopt is to

apply the reconstruction loss on the images y belonging to the target domain. The reason is that if an image already belongs to the target domain, it should be left unchanged by G. The equation for the reconstruction loss is as follows:

$$\mathcal{L}_{\text{rec}}(G) = \mathbb{E}_{y \sim p_\tau}[||G(y) - y||_1], \tag{8.4}$$

where $||\cdot||_1$ denotes the L_1 norm in the space of target images.

Feature Matching Loss In order to regularize the training, we also include a feature matching loss following the work of [59] and [34]. Feature Loss stabilizes the training since it is required to the Generator to produce natural statistic at multiple scale. We extract features from the discriminator layers located before the prediction layer. This feature extractor is called D_{feat}. The definition of the feature matching loss is as follows:

$$\mathcal{L}_{\text{feat}}(D_{\text{feat}}, G) = \mathbb{E}_{x,y \sim p_{\text{data}}, p_\tau}[||D_{\text{feat}}(G(x)) - D_{\text{feat}}(y)||_1]. \tag{8.5}$$

Full Objective Finally, our full objective becomes:

$$\mathcal{L}_D = \mathcal{L}_{\text{adv}} + \mathcal{L}'_{\text{dom}}, \tag{8.6}$$

$$\mathcal{L}_G = w_{\text{adv}}\mathcal{L}_{\text{adv}} + w_{\text{dom}}\mathcal{L}''_{\text{dom}} + w_{\text{rec}}\mathcal{L}_{\text{rec}} + w_{\text{feat}}\mathcal{L}_{\text{feat}}, \tag{8.7}$$

for the discriminator and generator, respectively. Furthermore, w_{adv}, w_{dom}, w_{rec}, w_{feat} are the weights assigned to the loss functions. The discriminator loss functions do not have weights assigned since adversarial and domain losses should contribute equally to discriminator training to obtain balanced results. Weights choices for the generator loss are more properly discussed in Section 8.4.

8.3.3 Algorithm

Our approach relies on a meta-learning algorithm based on Reptile [39] and adapted to the image generation problem.

The problem setting is as follows. A large dataset of images, called $\mathcal{D}$, is used to extract random input images. Let τ_j be a single task, where j ranges over the number of chosen training domains, here called N_τ. Each task dataset consists of a restriction of $\mathcal{D}$ on the images of a single domain, called $\mathcal{D}|_{\tau_j}$. Hyper-parameters of the algorithm are the inner learning rates of G and D networks, respectively λ_G and λ_D; the loss weights w_{adv}, w_{dom}, w_{rec}, and w_{feat}; two thresholds t and T for keeping discriminator accuracy into a certain range (in order to neither over- nor under-train D); and a learning rate for the outer networks, i.e. a meta-learning rate λ_{ML}. Parameters of the networks, i.e. network weights and biases, are indicated as θ_G and θ_D for G and D, respectively. The algorithm is divided into two phases, as illustrated in Figure 8.2. The first one is the training phase, and the second one is the inference phase. In the training phase, the *G-D* network is trained repeatedly on a single task, randomly extracted at each epoch from the set of available tasks. During inference phase, instead, a new task τ_I is used for a last-time few-shot training to adapt the network to the new domain. A detailed explanation of training phase is given in the next paragraph and in Figure 8.3, and it is followed by another paragraph devoted to the description of the inference phase.

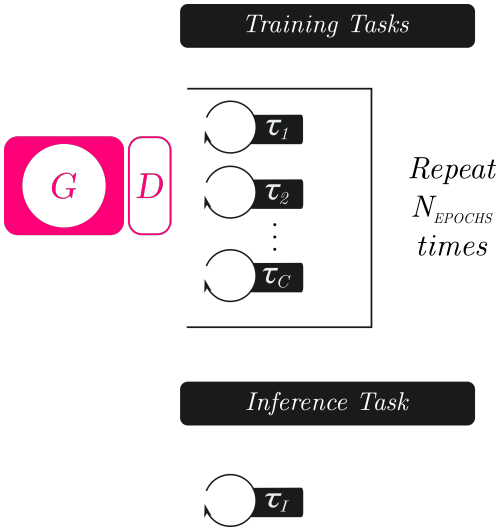


Figure 8.2: A full overview of the system: during the training phase, the network is trained for N epochs on a set of tasks, and then, during the inference phase, a new unknown task, i.e. not present in the training phase, is selected and added to the network.

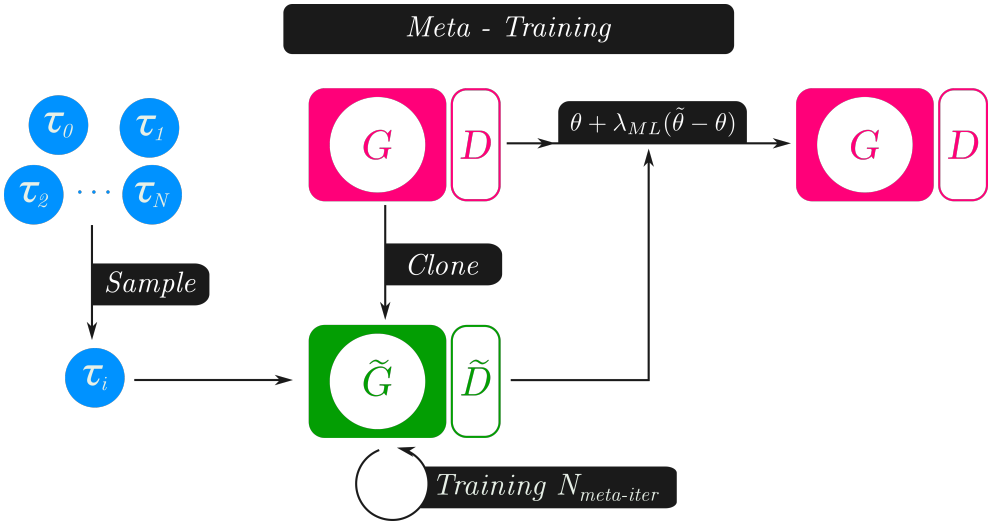


Figure 8.3: Scheme of a single training epoch. A task τ_i is sampled and $N_{\text{meta\_iter}}$ inner iterations are performed on cloned networks. Then θ and $\widetilde{\theta}$ are used to update the networks parameters using the Reptile equation.

Algorithm 3 MetalGAN algorithm

Require: N_{epochs} : number of epochs
Require: N_{τ} : number of selected domains
Require: λ_{ML} : meta-learning rate
1: load entire dataset : $\mathcal{D}$
2: load datasets restricted to each single task $\tau_j : \mathcal{D}|_{\tau_j}$ for $j \in \{0, \ldots, N_{\tau}\}$
3: **for** $epoch \in \{0, \ldots, N_{\text{epochs}}\}$ **do**
4: extract randomly τ_j
5: clone D into $\widetilde{D}$ of parameters $\theta_{\widetilde{D}}$
6: clone G into $\widetilde{G}$ of parameters $\theta_{\widetilde{G}}$
7: **Inner training loop** on τ_j
8: $\theta_G \leftarrow \theta_G + \lambda_{\text{ML}} \left(\theta_{\widetilde{G}} - \theta_G\right)$ ▷ updates generator parameters
9: $\theta_D \leftarrow \theta_D + \lambda_{\text{ML}} \left(\theta_{\widetilde{D}} - \theta_D\right)$ ▷ updates discriminator parameters
10: **end for**

Training The algorithm for training consists of an outer and an inner loop, similar to Reptile. The outer loop is responsible of training the actual *G*-*D* networks, updating their parameters, epoch-by-epoch as a traditional learning algorithm. At each epoch, a task τ is randomly sampled from a distribution of tasks. It is recalled that, in our case, a task is a domain and the associated dataset is the few-shot subset of the set of images in the domain. Then, G and D networks are cloned into $\widetilde{G}$ and $\widetilde{D}$ networks of parameters $\theta_{\widetilde{G}}$ and $\theta_{\widetilde{D}}$, respectively.

The cloned networks are trained in the inner training loop, where the traditional DCGAN training is performed by using task images, here indicated as y. This is needed in order to learn the current domain. Also a small portion of generic images from $\mathcal{D}$ is used, to teach the generator to perform domain switch. These images are called x. It is required to the generator to learn the transformation from the random image x of the dataset into an output image "similar" to y, i.e., the generated image should belong to the extracted task.

Finally, the obtained parameters are used to update G and D weights, with the Reptile rule (a sort of SGD step where the gradient is approximated by the difference between inner and outer weights). This baseline is illustrated in Figure 8.3.

In detail, the outer training loop is shown in Algorithm 3. Lines 8–9 of Algo-

rithm 3 are responsible of the parameter adaptation for networks G and D and such an operation is performed layer by layer.

Algorithm 4 Inner training loop

Require: τ : extracted task in the outer loop
Require: $N_{\text{meta\_iter}}$: number of inner epochs
Require: λ_D, λ_G : learning rates of D and G
Require: $w_{\text{adv}}, w_{\text{dom}}, w_{\text{rec}}, w_{\text{feat}}$: adversarial, domain, reconstruction, and feature weights
Require: t, T : minimum and maximum thresholds for discriminator accuracy

for $i \in \{0, \dots, N_{\text{meta\_iter}}\}$ **do**
  sample y from $\mathcal{D}|_\tau$
  sample x from $\mathcal{D}$
  ▷ **Discriminator training:**
  $\varepsilon_D \leftarrow \nabla_{\theta_{\widetilde{D}}} \mathcal{L}_{\text{adv}}(\widetilde{D}, \widetilde{G})$ ▷ x is considered fake, y real
  $\varepsilon_{\text{dom}} \leftarrow \nabla_{\theta_{\widetilde{D}}} \mathcal{L}'_{\text{dom}}(\widetilde{D})$ ▷ x is considered false, y true
  calculate accuracy $a_{\widetilde{D}}$ of discriminator $\widetilde{D}$
  if $a_{\widetilde{D}} < T$ **then**
    $\theta_{\widetilde{D}} \leftarrow \theta_{\widetilde{D}} - \lambda_D(\varepsilon_D + \varepsilon_{\text{dom}})$
  end if
  if $a_{\widetilde{D}} > t$ **or** $i = 0$ **then**
    ▷ **Generator training:**
    $\varepsilon_G \leftarrow \nabla_{\theta_{\widetilde{G}}} \mathbb{E}_{\widetilde{G}(x) \sim p_\tau}[\log \widetilde{D}(\widetilde{G}(x))]$ ▷ $\widetilde{G}(x)$ is considered real
    $\varepsilon_{\text{task\_rec}} \leftarrow \nabla_{\theta_{\widetilde{G}}} \mathcal{L}_{\text{rec}}(\widetilde{G})$ ▷ the reconstruction is made with y
    $\varepsilon_{\text{dom}} \leftarrow \nabla_{\theta_{\widetilde{G}}} \mathcal{L}''_{\text{dom}}(\widetilde{D}, \widetilde{G})$ ▷ both y and $\widetilde{G}(x)$ are considered true
    $\varepsilon_{\text{feat}} \leftarrow \nabla_{\theta_{\widetilde{G}}} \mathcal{L}_{\text{feat}}(\widetilde{D}_{\text{feat}}, \widetilde{G})$
    $\theta_{\widetilde{G}} \leftarrow \theta_{\widetilde{G}} - \lambda_G(w_{\text{adv}}\varepsilon_G + w_{\text{rec}}\varepsilon_{\text{rec}} + w_{\text{dom}}\varepsilon_{\text{dom}} + w_{\text{feat}}\varepsilon_{\text{feat}})$
  end if
end for

The inner training loop is illustrated in Algorithm 4. It is nothing more than a classic DCGAN training, but performed on the cloned networks. For each iteration, a small part of two datasets, that is the task dataset and the full training dataset, is used. The whole $\mathcal{D}$ is sampled randomly only for $N_{\text{meta\_iter}}$ iterations, using only few batches of images. The chosen task dataset $\mathcal{D}|_\tau$ is used for extracting domain specific

images. The first part is the discriminator training. Domain loss and adversarial loss are computed as in Section 8.3.2, and $\widetilde{D}$ parameters are updated if the accuracy of the discriminator is under a certain threshold T. On the contrary, the second part, that is the generator training, is executed only if the accuracy of the discriminator is above a certain threshold t. During this step, adversarial, task reconstruction, domain, and feature losses are all employed to update $\widetilde{G}$ parameters.

Inference The inference part is also a crucial one. In our work, we experiment the use of few images for adapting the trained model to new, unseen, domains, directly during the inference phase. The idea is to feed the trained *G-D* networks with images from new domains, moving the obtained parameters θ_G and θ_D in a new optimal direction to include the new tasks. A sort of fine-tuning is performed, by showing to the model few images from a new domain, and then few images from another new domain, and so on.

The settings of the inference algorithm are the following. A set of unseen tasks (or domains) $\mathcal{T}$ is adopted. A test dataset $\mathcal{D}^{(\text{test})}$ containing all domains, where $\mathcal{D}^{(\text{test})} \cap \mathcal{D} = \emptyset$, is used. As for the training dataset, for each new domain to infer $\tau \in \mathcal{T}$, the adequate restriction of dataset is used, $\mathcal{D}|_\tau \subset \mathcal{D}$, in order to avoid overlaps between test and training datasets.

Algorithm 5 shows the inference method. It is divided in two main parts: a few-shot fine-tuning, and a test phase where unseen images are transformed into target domain images. In the first part, new domains are used to learn a new parameter adaptation, using few images per class. Moreover, the meta-learning rate for inference is greater than the one used in training, in order to ensure a faster adaptation. The second part is used only for generating the results, and it resembles a more classic inference. The inner training loop of the meta-learning algorithm is used, but obtained parameters are not updated for the next domain.

An explanation of inference algorithm is visually provided in Figure 8.4 where two exemplar cases are shown: the top row of the figure (Figure 8.4(a) and 8.4(b)) shows the case of the seen domain ‘Black Hair’, whereas the bottom row (Figure 8.4(c) and 8.4(d)) shows the case of unseen domain ‘Heavy Makeup’. In the image,

Algorithm 5 Inference

Require: λ_{ML} : meta-learning rate for inference
Require: $\mathcal{T}$: set of **new** tasks/domains
Require: $N_{\text{inf\_epochs}}, N_{\text{inf\_train}}, N_{\text{inf\_test}}$: number of inference iterations
load few-shot test dataset: $\mathcal{D}^{(\text{test})}$
load few-shot restricted test dataset on each $\tau \in \mathcal{T} : \mathcal{D}|_{\tau}$
▷ **Fine-tuning on unseen domains:**
for $epoch \in \{0, \dots, N_{\text{inf\_epochs}}\}$ **do**
    for $\tau \in \mathcal{T}$ **do**
        clone *G-D* networks
        Inner training loop on τ, for $N_{\text{inf\_train}}$ iterations
        update θ_G and θ_D with Reptile rule
    end for
end for
▷ **Inference on unseen domains:**
for $\tau \in \mathcal{T}$ **do**
    clone *G-D* networks
    Inner training loop on τ, for $N_{\text{inf\_test}}$ iterations
    for $x \in \mathcal{D}^{(\text{test})}$ **do**
        generate output image $\widetilde{G}(x)$
    end for
                ▷ *G-D* parameters are not updated anymore
end for

the model, called M, consists of the *G-D* networks trained as in Algorithm 3 on five different example domains. In Figure 8.4(a), a naïve illustration of the domain space is provided. Given the dots as domains, and the violet dots as the seen domains, the model M has learned, during training, a sub-optimal representation for the seen domains. Intuitively, such a representation permits the model to easily move towards the optimal one in few steps of the inner training loop. In Figure 8.4(b), an example inference on 'Black Hair' domain is shown. Since 'Black Hair' belongs to the trained domains set, the model is cloned and with some steps of inner training loop, the cloned model learns an optimal representation for the domain. Finally, parameters of the cloned model are not saved nor used for updating the main model, so the situation returns to the one depicted in Figure 8.4(a), ready to make another inference. When a domain, or a set of domains, are completely new to the generator and discrimina-

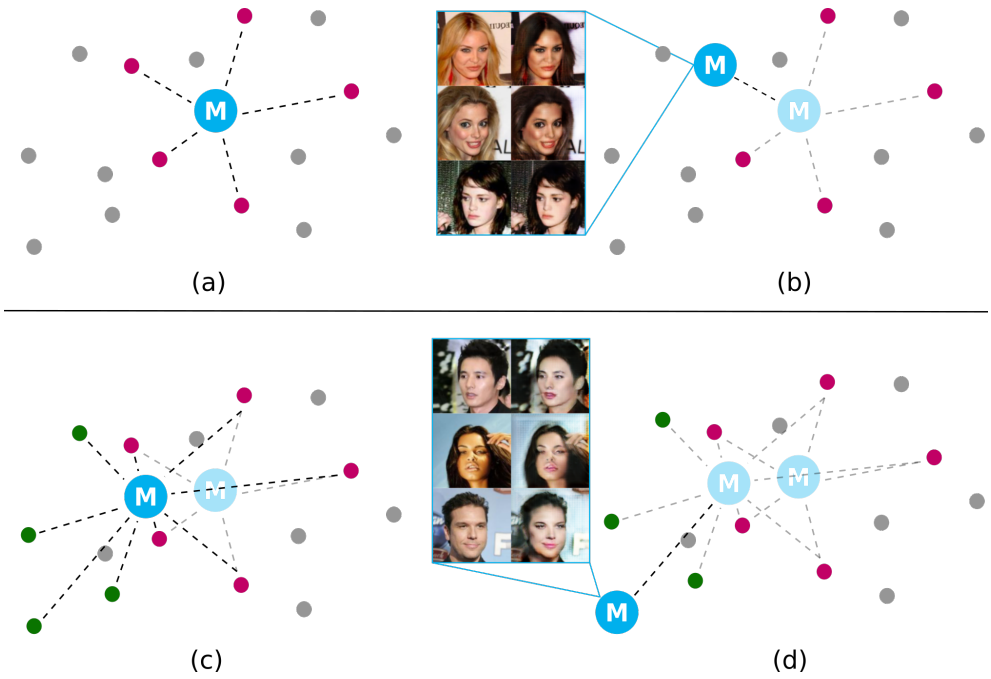


Figure 8.4: Inference algorithm in the case of one of the training domains, 'Black Hair' (b) and in the case of an unseen domain, 'Heavy Makeup' (d). The violet dots represent the domains used during training while the green dots are the unseen domains. The model *M* can move towards a known model, i.e. from state (a) to state (b); the other option is fine-tuning towards a set of unseen domains (c), then converge to a specific one (d).

tor, a preliminary fine-tuning is needed. In Figure 8.4(c), green dots represents these unseen domains. The inference pre-training moves the (already trained) model in a sub-optimal position for both the trained domains and the new domains. In Figure 8.4(d), the inference on the updated model is shown. As Figure 8.4(d) shows, the fine-tuning for the unseen domains moves the model M in a new "position" in the space, closer to the unseen domains. In this way, during the inference (Figure 8.4(d)) the 'Heavy Makeup' domain is better learnt and more correct images are generated, even if the domain has not been seen during the training.

Please note that the figure is completely exemplifying, since it depicts domains

in random positions, and does not take in account the intersection between them, nor their 'real' positioning in an actual domain space (which is unknown). The idea of the model moving towards a sub-optimal yet effective representation during training epochs—minimizing the expected distance from all tasks—, and an informal proof of the idea using Euclidean distances in the manifold of optimal solutions of a task, is provided in Reptile paper [39].

8.4 Experimental Results

This section presents visual and quantitative results of performed experiments of MetalGAN, compared with StarGAN ones. All our experiments were conducted using the CelebA dataset [32] which is a large-scale face attributes dataset with more than 200k celebrity images, each with 40 attribute annotations. We decided to test our algorithm on this dataset for three main reasons: first of all, since it contains images of faces with all kind of attributes, it is suitable for multi-domain image-to-image task; secondly, it was used by StarGAN so it allows a clear comparison between the results of the two different algorithms; and finally, even though our approach is completely label-less, it is very easy to automatically divide a-priori the dataset in its different domains.

Experiments are divided into two categories: test results on seen domains (i.e., tasks the *G-D* networks were trained on), and results on unseen domains. In case of experiments on StarGAN, since their algorithm requires labels, we trained their network on some domains with a few number of images (1000), and we call these "unseen" domains. This workaround permits us to compare StarGAN with our inference on unseen domains. It is worth to note that this approach is unfair for us, since StarGAN is fully trained for each of these "unseen" domains, while we only perform a small inference step. This is due to the fact that we can choose to add new domains to our network at every time, while StarGAN needs to define all the domains at the training stage.

Table 8.1: Hyper-parameters of MetalGAN training phase.

N_{epochs}	100000
λ_{ML}	0.01
λ_G, λ_D (Adam)	0.0001
$N_{\text{meta\_iter}}$	20
batch size	16
w_{adv}	1
w_{dom}	1
w_{rec}	10
w_{feat}	1

8.4.1 Results on Trained Domains

We trained *G-D* networks model on 5 domains, namely 'Eyeglasses', 'Male', 'Blond Hair', 'Black Hair', and 'Pale Skin' for $N_{epochs} = 100000$ using the MetalGAN algorithm, and on the same domains for 200000 epochs using StarGAN.

Table 8.1 presents the main settings for our experiments. We set the Reptile learning rate λ_{ML} to 0.01 and optimized the generator and discriminator networks using Adam with a learning rate equals to 0.0001. Furthermore, we set the number of meta-iterations N_{meta_iter} during training equals to 20 since we empirically found that this value represents the best trade-off between speed and accuracy of the algorithm. For coherence with StarGAN, batch size is set to 16 during training. Weights for MetalGAN objective during training are left to 1 except for reconstruction weight, that is set to 10, in order to obtain an accurate reconstruction of the image and gain more quality in results.

Figure 8.5 shows some visual results on a batch of eight input images. Figure 8.5(a) contains the outputs of MetalGAN algorithm, while Figure 8.5(b) shows the StarGAN outputs. In addition, a greater number of results on some of the training classes are shown in Figure 8.6 and 8.7. Figure 8.6 shows generated images on 'Eyeglasses' domain, where input images are put side-by-side to MetalGAN outputs and StarGAN outputs. In the same fashion, results on 'Black Hair' domain are reported

(a) MetalGAN outputs. (b) StarGAN outputs.

Figure 8.5: Results on training classes. In the first column, the input images. From second to fifth column there are the outputs of the model moved towards the respective domain, in case of MetalGAN, or labeled with the indication of the domain, in case of StarGAN. The last column of MetalGAN results is the output of the model without moving it from the sub-optimum.

in Figure 8.7. We decided to choose these two domains since they are very different in terms of features and since our method performs very well on 'Eyeglasses' and, on the contrary, it is not so good on 'Black Hair'. It is also worth noting that MetalGAN on 'Eyeglasses' produces a great variability of examples, compared to StarGAN, generating both simple glasses and sunglasses.

However, the image generation should be considered successful and visually close to StarGAN one. As a matter of fact, we can see how our label-less approach produces results that are visually very similar to the ones produced by StarGAN. In particular, our algorithm is able to understand the different target domains just by seeing few examples of them each epoch, and can correctly produce these domains

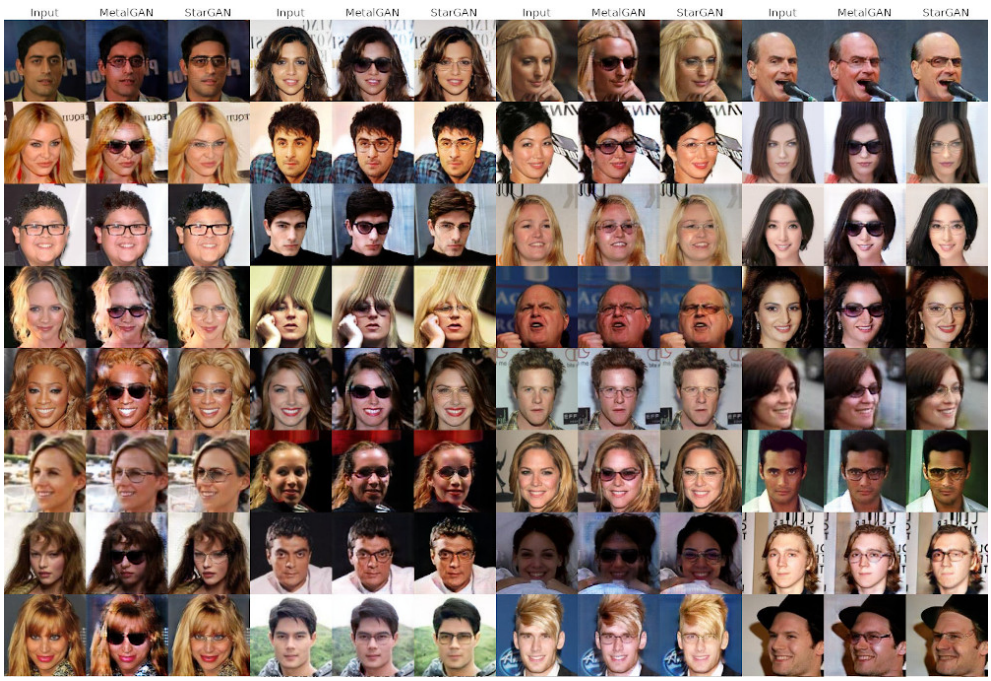


Figure 8.6: Results on training domain Eyglasses. The image triplets are composed by input image, MetalGAN output and StarGAN output.

from the input images even without labels or supervision.

In addition, we performed quantitative analysis of the produced results. As far as we know, no pure theoretical framework is available for a precise quantification of our model contributions and advantages, in order to compare it to others, but there exists some relevant metrics that are suitable for a numerical placement of our proposal. Metrics considered in this work are FID (Frechet Inception Distance) [51] and PRD (Precision and Recall for Distributions) [79], described below. We use FID to calculate the distribution matching between the original CelebA images for each training domains and our results, and we compare our score with the one obtained on StarGAN images. This comparison is presented in Figure 8.8 with lower values indicating the better scores. Our method performs slightly better than StarGAN for 'Eyeglasses', 'Male' and 'Blond Hair' domains and slightly worse than StarGAN for

Figure 8.7: Results on training domain Black Hair. The image triplets are composed by input image, MetalGAN output and StarGAN output.

'Black Hair' and 'Pale Skin', confirming the visual evaluation of the images.

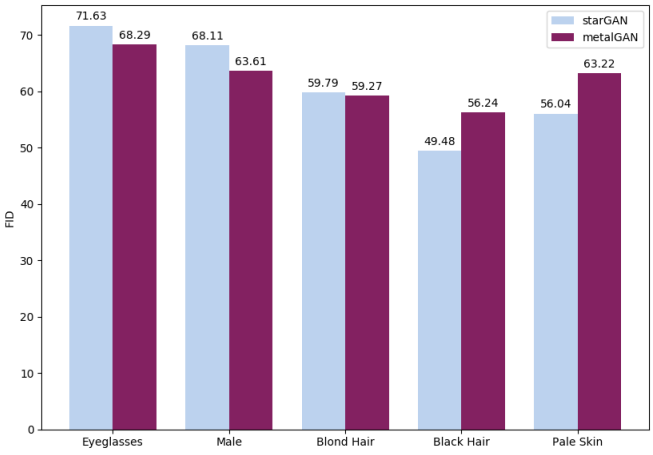
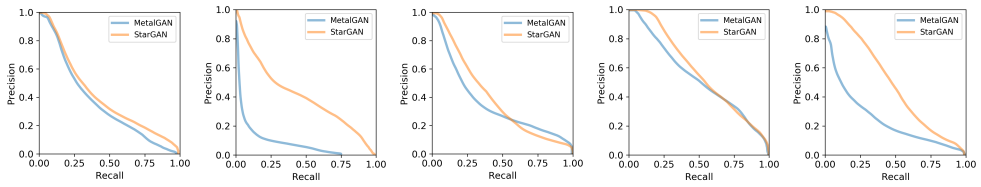


Figure 8.8: FID score on training domains (the lower the better).

Another quantitative analysis is based on PRD for both StarGAN and MetalGAN methods, using classes of images of CelebA as target datasets. Precision is a measure of raw quality of generated images, and does not take in account the internal variability of the distribution, while recall measures how well the generated images resembles the "class distribution" of the target dataset. We choose to measure a single domain at once. Results are shown in five different graphics, in Figure 8.9. As shown,



(a) Eyeglasses (b) Male (c) Blond Hair (d) Black Hair (e) Pale Skin

Figure 8.9: PRD on each training domains.

MetalGAN PRD on 'Eyeglasses', 'Blond Hair', and 'Black Hair' are very similar to each other and close to StarGAN results. The main difference between StarGAN and

MetalGAN in case of hair domains is that StarGAN is usually more precise (it produces images with a better quality w.r.t. the target distribution), but it has a lower recall, meaning that the distribution of StarGAN generated images is less varied than MetalGAN one. Regarding 'Male' and 'Pale Skin', the precision of MetalGAN suffers from the fact that such domains require a significant change in all the faces in the input image, highlighting a weakness in MetalGAN global reconstruction. On the other hand, the domain change is successful, as confirmed for 'Male' FID score. In

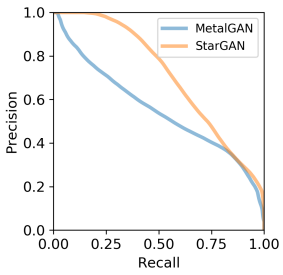


Figure 8.10: Global PRD on training domains.

Figure 8.10, global PRD, computed on all training domains at once, resembles the previous consideration, showing a worse precision of MetalGAN generated distribution, but a similar high recall for StarGAN and MetalGAN distributions.

It is worth emphasizing once more that MetalGAN achieves these results without labels, showing in any case comparable quantitative results and often better qualitative results.

8.4.2 Results on Unseen Domains

During the inference step, we modify the hyper-parameters of MetalGAN as shown in Table 8.2. In particular, in order to allow the network to quickly adapt to the new domains, we increment the λ_{ML} to 0.1 and we set w_{adv} and w_{dom} to 100. Furthermore, since the network already learned to reconstruct the content of the input images we lower w_{rec} to 1.

Finally, we tested the MetalGAN trained model on 6 unseen domains, namely

Table 8.2: Hyper-parameters of MetalGAN inference phase.

N_{epochs}	10
λ_{ML}	0.1
λ_G, λ_D (Adam)	0.0001
$N_{\text{inf\_train}}$ (train)	20
$N_{\text{inf\_test}}$ (test)	100
batch size	16
w_{adv}	100
w_{dom}	100
w_{rec}	1
w_{feat}	1

'Big Lips', 'Bushy Eyebrows', 'Heavy Makeup', 'Smiling', 'Gray Hair', and 'Mustache' using the MetalGAN inference. MetalGAN, trained on the 5 seen domains of Section 8.4.1, performs 10 further outer iterations (on each new domain), each of them consisting of 20 inner iteration, where 320 task images are seen for the first time. In this way, a fine-tuned model is obtained, as in Figure 8.4(c). Then, images are generated by specializing the fine-tuned model on the chosen domain, as in Figure 8.4(d). Such a specialization is done performing 100 inner iterations per domain.

On the other side, we trained 6 *new* StarGAN models with the same domains used during training, plus one unseen domain for each model, i.e. we obtained a StarGAN model specialized also in 'Big Lips', another StarGAN model specialized also on 'Bushy Eyebrows', and so on. This is necessary, since StarGAN uses image labels, so adding a new domain is possible only by retraining the model. All six new StarGAN models were fully trained for $N_{\text{epochs}} = 200000$. For StarGAN, "unseen" means that only 1000 input images are selected for that domain, as already described in the beginning of Section 8.4.

Visual qualitative results for unseen domains for both MetalGAN and StarGAN are presented in Figure 8.11, 8.12 and 8.13. In particular, for MetalGAN, the results produced without performing the fine-tuning iterations are also shown. Our algorithm

Figure 8.11: Results on unseen domains Big Lips and Bushy Eyebrows. The image triplets are composed by input image, MetalGAN output without fine-tuning, MetalGAN output with fine-tuning and StarGAN output.

is able to produce compelling images even in this case and further improves the visual appearance of the images after the fine-tuning step. In addition, MetalGAN applyes the unseen domains to the input images in a more soft and natural way than StarGAN. This is particularly evident in 'Big Lips' and 'Smiling', where StarGAN produced results that could be described as "creepy". A further consideration is the fact that sometimes, during the unseen domain transfer, MetalGAN tends to apply unwanted features to the images. For example in 'Bushy Eyebrows' the network often changes the hair color to black or applyes mustaches. This is due to the fact that people with bushy eyebrows generally have darker hair and facial hair. The same reasoning can be applyed to 'Gray Hair', where the network tends to produce older people, because people with gray hair are usually old. The reason for this behavior is that, because of the lack of labels, the network has to infer which is the domain to be transferred without any help. This is also a big advantage, because produces a much greater flexibility to the network and allows to add new domains to the network very easily.

We calculated both FID and PRD also for inference domains, as in the previous section. In Figure 8.14, FID scores are reported. As for training, FID scores depend heavily on the selected domain, but in general, StarGAN and MetalGAN scores are close to each other. In particular, MetalGAN performs better on 'Big Lips', 'Smiling', and 'Bushy Eyebrows', confirming visual evaluation of results.

In Figure 8.15, PRD graphs for each unseen domain are reported. All results show how both StarGAN and MetalGAN decrease their precision in this phase, as reasonable. As we can see in Figure 8.11, 8.12, and 8.13, the overall quality of the reconstruction is slightly worse than the one of trained domains. However, despite the unfair comparison, PRD for MetalGAN and StarGAN are pretty similar. Looking at the global PRD, calculated on all six unseen domains at once (Figure 8.16), MetalGAN shows better performances especially on distribution recall.

Figure 8.12: Results on unseen domains Heavy Makeup and Smiling. The image triplets are composed by input image, MetalGAN output without fine-tuning, metalGAN output with fine-tuning and starGAN output.

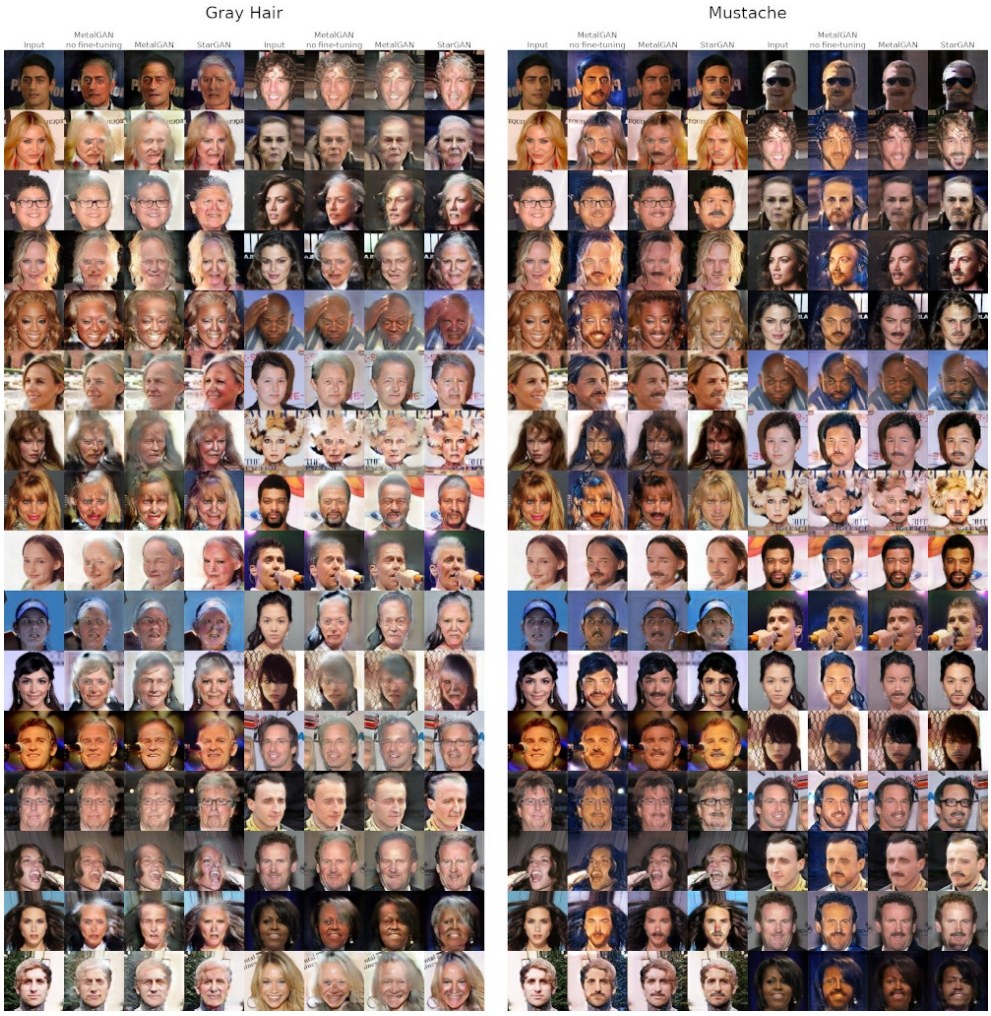


Figure 8.13: Results on unseen domains Gray Hair and Mustache. The image triplets are composed by input image, MetalGAN output without fine-tuning, metalGAN output with fine-tuning and starGAN output.

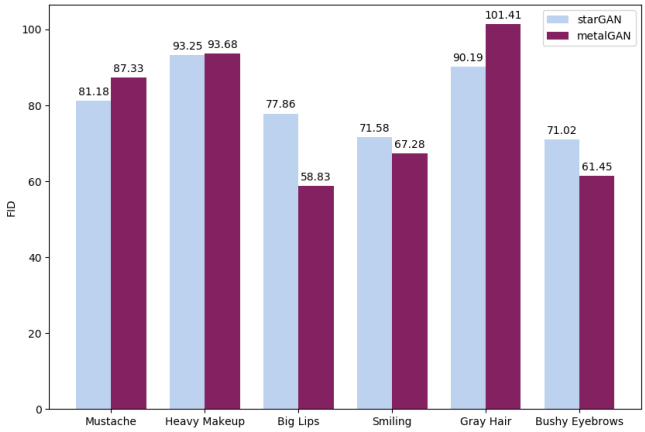
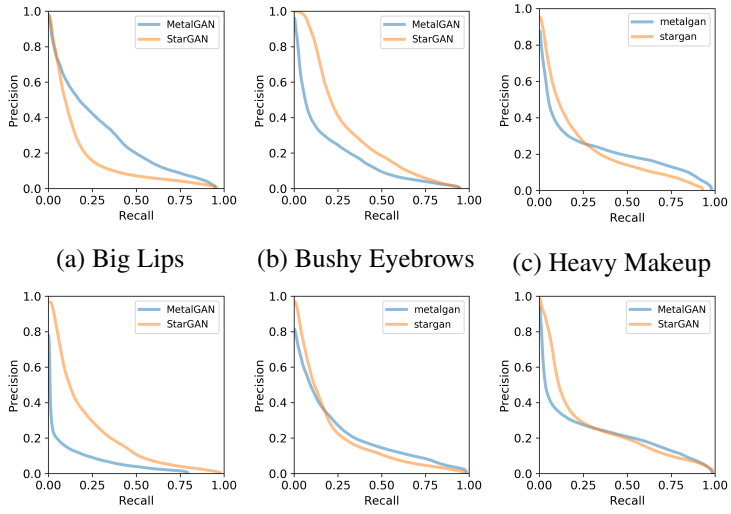


Figure 8.14: FID score on inference domains (the lower the better).

8.4.3 Additional Experiments

Results on Radboud Faces Database

In order to further prove the effectiveness of our method, we also trained the *G-D* network on another multi-domain dataset, with MetalGAN algorithm. Such a dataset is Radboud Faces Database (RAFD) [80]. RAFD is a set of pictures of 67 models displaying 8 emotional expressions. We trained the model for 20k iterations with MetalGAN on 5 different emotions (*disgusted*, *fearful*, *happy*, *sad* and *surprised*), maintaining the same configuration used with the CelebA dataset. A batch of visual results is shown in Figure 8.17 (a), where only the trained domains are tested. Then, additional unseen domains were added to further test our method on this new dataset, as for CelebA. Such new domains are *angry*, *contemptuous* and *neutral*. Inference configurations are the same of Table 8.2. In Figure 8.17 (b), the visual results of MetalGAN inference on RAFD are reported. Results are comparable to trained ones, even if they were obtained by few iterations on the trained model, and with few input images. The naïve reason could be that changing the facial expression involves few attributes of the image, thus switching from the input facial expression domain to an unseen one shares a lot of knowledge with the switching between the input and



(a) Big Lips (b) Bushy Eyebrows (c) Heavy Makeup

(d) Smiling (e) Gray Hair (f) Mustache

Figure 8.15: PRD graphs on inference domains.

the trained domains. In other words, the main task is the same: changing the facial expression, and little differences between domains are handled easily by the inference steps.

In addition to the qualitative comparison, we also trained a classification network in order to obtain a quantitative evaluation of our method. We choose ResNet-18 as classification network (following the StarGAN paper) and we produced classification results on the different emotions both for our architecture as well as for StarGAN trained on the same domains. Results can be seen in Table 8.3. Following the considerations that were made for the CelebA results, our results for the RAFD dataset are in line with the StarGAN ones, but without the use of label or supervision. The only exception is the *sad* domain where our network tends to only change the mouth leaving the rest of the face almost unchanged. Therefore, if the input image has another emotion strongly characterized by the eyes or by the eyebrows (such as *surprised*), such features are not changed during the domain switch leading to misclassification.

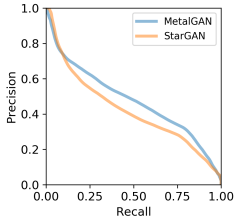


Figure 8.16: Global PRD on inference domains.

Table 8.3: Classification results on RAFD dataset.

	disgusted	fearful	happy	sad	surprised
StarGAN	98.6%	97.2%	98.6%	97.7%	97.1%
MetalGAN (ours)	98.4%	93.1%	97.3%	69.7%	95.2%

Contribution of Loss Functions in the Experiments

The effectiveness of each loss function of the architecture (see Section 8.3.2) is discussed below. The basic GAN structure, as theoretically proposed in [2], relies on the Adversarial Loss function (being a minimax two-players game between G and D networks). Hence, every GAN model takes advantage of such a loss function. We did not choose to empirically test its effectiveness since the only way is to get rid of the loss and see what changes happen to the results. But performing an experiment without the Adversarial Loss function is unfeasible for GAN architectures, unless the goal is to change the fundamentals of GAN, which is clearly out of the scope of this work.

The necessity of the Domain Loss in the MetalGAN algorithm is not self-evident: for this reason, we performed an experiment with the same configurations of MetalGAN standard training (see Table 8.1), but nullifying the domain weight, i.e. $w_{\text{dom}} = 0$. It is worth noting that this setting still relies on meta-learning, that is the major boost for domain adaptation without labels. Nevertheless, visual results on CelebA, for MetalGAN without Domain Loss, show that the domain change loses quality, as it should be seen in Figure 8.18.

The Reconstruction Loss is used to keep unchanged the facial feature of the sub-

ject of the input image. Figure 8.5 last column of MetalGAN results shows, in fact, the only contribution of Reconstruction Loss, before moving the model to the optimum position for a certain domain (inference). Reconstruction Loss also makes sure that input images that already belong to the chosen output domain are left completely unchanged by our architecture. Performed experiments show this behavior, e.g. in Figure 8.5 fifth row, second column: the input subject wear eyeglasses, so there is no need to change its domain, and MetalGAN simply transfer the eyeglasses of the input image into the output one, without changing facial attributes.

Finally, the Feature Loss is a base building block of cGAN for regularization. As detailed in [59], and briefly remarked in Section 8.3.2, Feature Loss stabilizes the training and it is a common state-of-the-art method for cGAN.

8.5 Conclusions

We proposed a new architecture for multi-domain label-less image-to-image translation. Our system has many features that distinguish it from the state-of-the-art.

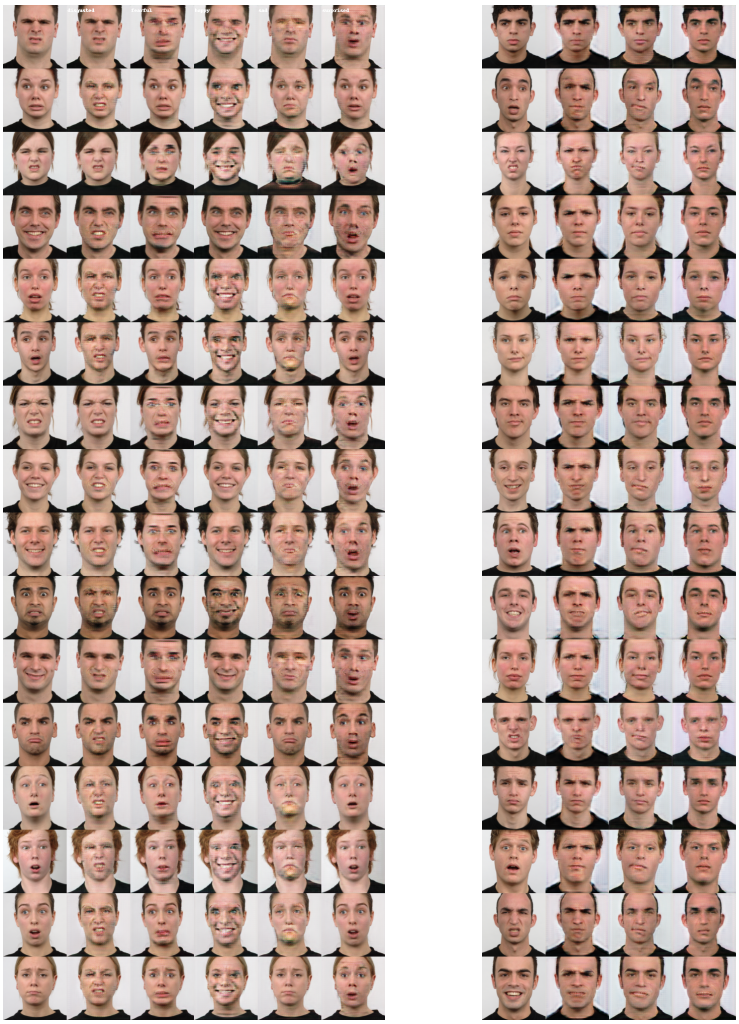
First of all, instead of relying on labels for switching the domains, like other state-of-the-art architectures, we had chosen to use meta-learning and in particular Reptile. Furthermore, getting rid of labels allowed the architecture to be more flexible, since there is no need of providing hard-coded vectors of labels. It is possible to arbitrarily change the number of domains, and to add a new one during inference. Such an approach was completely unfeasible in previous algorithms like StarGAN, that needs hard-coded labels at training time, and this was a very serious limitation. Finally, beside the lack of labels, an immediate advantage of the meta-learning approach is that such a method has been used for few-shot learning. Not only, as highlighted above, a new, unseen, task can be added, but in order to do so, just few examples are needed, and neither tedious and long-lasting annotations of labels, nor a full retraining of the model are required.

We proved the effectiveness of our approach with face attributes transfer using the CelebA dataset, and we evaluated it using both FID and PRD quantitative metrics. Moreover, we performed additional experiments on RAFD dataset, and tested our

approach by nullifying the contribution of Domain Loss, showing its necessity.

Regarding future work, our first objective would be to explore more deeply the possibilities and limitations of meta-learning in order to further improve our algorithm and to prove its effectiveness on others tasks like image generation and semantic segmentation.

This work led to the following publication: *Tomaso Fontanini, Eleonora Iotti, Luca Donati, Andrea Prati. "MetalGAN: Multi-domain label-less image synthesis using cGANs and meta-learning." Neural Networks (2020).*



(a) Training domains. (b) Unseen domains.

Figure 8.17: MetalGAN results on RAFD on trained and unseen domains. Columns in first image represent disgusted, fearful, happy, sad, and surprised domains. Columns in second image represent angry, contemptuous, and neutral domains.

Figure 8.18: Results on CelebA without the contribution of Domain Loss.

Chapter 9

Conveying knowledge in conditional GANs with attention distillation

Always two there are,
no more no less.
A master and an apprentice.

9.1 Introduction

Recently, understanding and explaining neural networks has seen great progress and has led to many applications in real-world tasks. Inspired by Zeiler and Fergus [13], much effort has been put in visualizing and understanding feature activations in convolutional neural networks (CNNs) by generating attention maps that highlight regions which are considered to be important for the network goals. In other words, given a trained CNN model, attention maps indicate where an object is localized in the image, *e.g.*, a plane in the top area of the image. This kind of visualization using attention maps can help explaining why this image is classified to the *plane* class. Following this line of work, Wenqian *et al.* [16] proposed a technique to visually explain

Variational Autoencoders (VAEs), using gradient-based attention maps, showing the capability of attention maps in understanding generative models by means of visual explanations.

Based on those advances in model explainability, efforts were made in exploring how to utilize valuable information contained in attention maps in order to improve the performance of CNNs. [81] used generated attention maps as visual guidance for training CNNs and observed advances in model generalizability in traditional image classification and segmentation tasks. Zagoruyko *et al.* [17] built a system to transfer attentions from a teacher network to a smaller student network and observed improvements of the student network in terms of image classification performance. Dhar *et al.* [18] proposed an approach with attention distillation loss for incremental learning. Nevertheless, transferring knowledge in Generative Adversarial Networks (GANs) by using attention maps as visual explanations is an area that has not been explored yet.

With the introduction of GANs and its many variants, image generation has seen a gigantic leap forward, especially in terms of photorealism. Models that are able to produce highly detailed images like [82] are more and more common and their outputs are almost indistinguishable from real images. Still, often it is necessary to condition the generative model in order to have control over its output. This is the case of image-to-image translation where the objective is to have a more useful representation of the input data that can be later used for several applications. Examples of this approach include style transfer, semantic or instance segmentation and sketch colorization.

In this work we argue that visually explaining image-to-image conditional adversarial network is a fundamental step in order to improve them. Following the work of [15], we are able to generate attention maps in the generator of a multi-domain image-to-image conditional GAN with only a small change in the model. Multi-domain image-to-image translation task requires the system to translate a single input image into multiple domains (*e.g.* facial attributes) using only a single generator and discriminator. The produced attention maps highlight the section in the features where the network is focusing for a certain domain and also allow to identify which lay-

ers in the generator are more devoted to the domain translation task as presented in Fig. 9.1. Having this information, it is possible to use the attention maps to convey knowledge to another network that shares the same image-to-image translation task following a teacher-student paradigm and to improve its generated samples. To this end, we present a learning objective that uses attention distillation loss and show how this loss term can be integrated with conditional GANs flexibly.

The most interesting fact about this knowledge transfer is that the teacher network is not required to have better results than the student network, but simply to produce a better visual explanation over the translation task, in order to guide the training of the student network.

In addition to that, another possible application is to use what we call “pseudo"-attention maps, that are attention maps generated from a set of domains and used as supervisions for a different set of domains in order to help the generator to produce better samples belonging to this new set of domains.

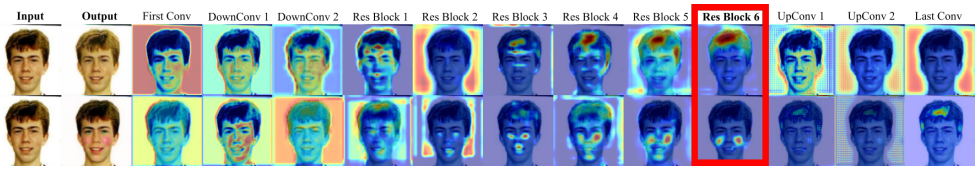


Figure 9.1: Visual explanations over the conditional GAN in the image-to-image task using attention maps. The most refined attention can be seen in the last residual block (highlighted in red). In the attention map, the more red the region is, the more important it is to the network. In the first row the task is *blonde hair* while in the second row is *rosy cheeks*.

The main contributions of this work are the following:

- extending the concept of generating visual explanations to image-to-image translation conditional GANs, proving that with this kind of additional information it is possible to visually explain their behaviour;
- a system consisting in a teacher and a student network, where the teacher uses only the attention generated in its latent space to improve the performance of

the student. The system forces the student model to 'look' at the images in the same way as the teacher model, directing the student training to be as close as possible to the one of the teacher.

- a system that derives and utilizes "pseudo"-attention maps generated from a set of domains from the teacher model as supervisions to help the student model to learn a set of different domains.

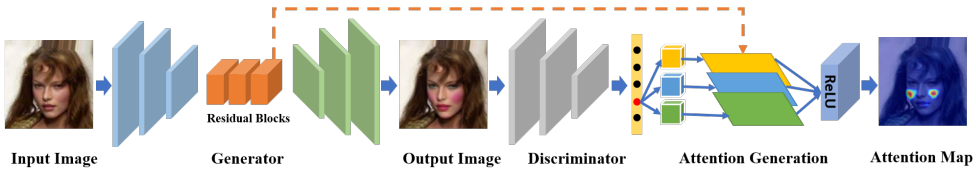


Figure 9.2: Attention generation with image-to-image conditional GAN.

9.2 Related Work

Conditional GANs for image-to-image translation. Generative Adversarial Networks (GANs) [2] in their many variations represent the state-of-the-art for photo-realistic image synthesis today. In particular, when a much finer control over the output is required, conditional GANs (cGANs) [3] allow the generation of images from text [21, 83], class labels [84], sketches or textures [20, 8]. Furthermore, while initially a paired dataset was required [85], CycleGAN [7] proved that a conditional GAN can be successfully trained in an unpaired way. Another relevant feature that most cGANs lack is the ability of producing images belonging to different classes or domains using a single architecture. Some models [10, 34] achieve that by using adaptive instance normalization layers [33] combined with a class-specific encoder and a content-specific encoder. On the other hand, StarGAN [9] and its variants [86, 87] take as input both an image and the target domain label learning to flexibly execute the translation using only one underlying representation. Finally, cGANs can also be combined with meta-learning for greater flexibility and robustness [88].

CNNs visual attention explanation. Deep Convolutional Networks have achieved astounding results in most computer vision tasks, but they are still mainly used as black boxes. For this reason, explaining their behaviour by visualizing 'where they look' when making a decision has attracted lots of interest in the last years. Particularly, after the initial work of [89] and [13], Zhou *et al.* [14] provided a method for generating *class activation mappings* (CAM) by using the global average pooling. The main drawback of this solution is that it is applicable only to a specific set of CNNs, that is the ones having the features maps directly preceding softmax layers.

For this reason, Grad-CAM [15] and its variant Grad-CAM++ [44] were proposed. They both use gradient of the score for a class c with respect to a specific feature maps F_k to obtain the class-discriminative localization map and, thus, can be considered gradient-based. Moreover, they are applicable to a very wide range of architectures without requiring any structural change in the network, unlike response-based approaches [90, 91, 14] that instead use additional trainable units. Recently, the concept of visual attention was also extended to GANs [92] and VAE [16].

Knowledge distillation in neural networks. Knowledge distillation involves transferring knowledge from a more complex network (teacher) to a simpler and lighter network (student) sharing the same task [93]. The goal is to have the student to reach almost the same results as the teacher. Many techniques have been developed in this area [94, 95, 96], with [17] being the first to use attention transfer to improve the performance of a student classification network. Previous methods were almost entirely applied over recongnition or classification models, while [97] introduced a method working on unconditional GANs. Recently, Li *et al.* [98] proposed a compression algorithm for cGANs using feature distillation and neural architecture search.

9.3 Approach

9.3.1 Generate Explanations for Image-to-Image Translation

Starting from the fundamental framework of Grad-CAM [15], we can obtain an attention map corresponding to the input sample from our image-to-image conditional GAN model given an input image x and a set of target domains $\boldsymbol{c}$. For each class c

from the ground-truth labels of target domains, we computed the gradients of score y^c corresponding to the class c. We backpropagated the gradients directly from the classification output of the discriminator to the convolutional layers of the generator with feature maps $\mathbf{F} \in \mathbb{R}^{n \times h \times w}$, obtaining attention maps $\mathbf{A}^c$ corresponding to y^c. Specifically, $\mathbf{A}^c$ is calculated as:

$$\mathbf{A}^c = ReLU\left(\sum_{k=1}^{n} \alpha_k^c \mathbf{F}_k\right) \tag{9.1}$$

where the scalar $\alpha_k^c = \text{GAP}\left(\frac{\partial y^c}{\partial \mathbf{F}_k}\right)$ and $\mathbf{F}_k$ is the k^{th} feature channel ($k = 1, \ldots, n$) of the feature maps $\mathbf{F}$, with $\frac{\partial y^c}{\partial \mathbf{F}^k}$ representing the gradient of the score y^c with respect to the feature maps $\mathbf{F}^k$. The global average pooling (GAP) operation is used to obtain scalar α_k^c as:

$$\alpha_k^c = \frac{1}{S} \sum_{m=1}^{h} \sum_{n=1}^{w} \left(\frac{\partial y^c}{\partial F_k^{mn}}\right) \tag{9.2}$$

where $S = h \times w$ and F_k^{mn} is the pixel value at location (m,n) of the $h \times w$ matrix $\mathbf{F}_k$. The above pipeline of generating attention maps is illustrated in Fig. 9.2. Note that we take the conditional GAN in the image-to-image translation task as the study case in our work, but this pipeline to generate attention maps can be applied in a wider variety of GAN networks, which consist of a generator-discriminator structure with classification stream integrated in the discriminator. Some examples of attention maps generated are shown in Fig. 9.1. We observe that those facial attributes used as target domains which are expected to be applied over input images are highlighted as regions in attention maps. For example, *cheekbones* of the face are highlighted as "red" in the attention map (**Res Block 6**) in Fig. 9.1, which means that the network is focusing on those regions so that the facial attribute *Rosy Cheeks* would be applied to obtain the output images (**Output**).

9.3.2 Network Architecture

Our system is composed by a teacher and a student network. Both of them are a multi-domain image-to-image conditional GAN, meaning that they are composed by

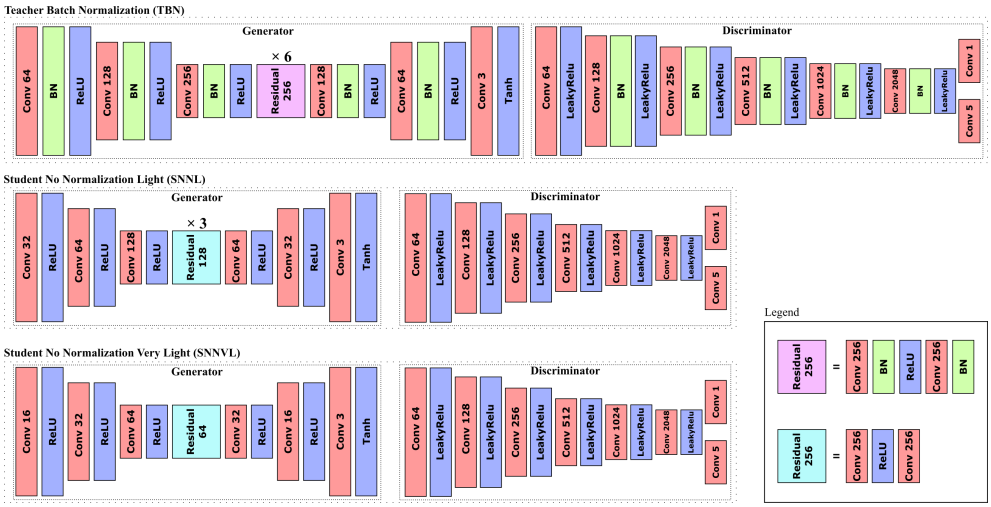


Figure 9.3: Summary of the different architectures used by the system. Firstly, the Teacher Batch Normalization (TBN) represents the full model with batch normalization layers, 64 feature maps in the first layer of the generator and 6 residual blocks. Secondly, the Student No Normalization Light (SNNL) is a smaller version of TBN without normalization layer and only 32 features maps in the first layer of the generator and 3 residual blocks. Finally, the Student No Normalization Very Light (SNNVL) is an even smaller version of SNNL with just 16 feature maps in the first layer and one residual block.

a generator and a discriminator conditioned by the label of the class that we want to transfer over the input image. The generator takes as input an image and the target class and returns the translated image, while the discriminator takes the translated image as input and returns an adversarial output and a domain classification output. For the network architectures, we draw inspiration from StarGAN [9] making only some minor changes.

Starting from the teacher architecture, we removed Instance Normalization (IN) layers from the generator, since, when calculating attention, we noticed droplet-like artifact in the images, as already stated in [82], and we replaced them with batch normalization (BN) layers. The reason for this choice is that, even if batch normal-

ization is not the best for image-to-image translation task, it allowed us to get the best attention visualization, which is consistent with classification results of the discriminator, that is the most important factor for our system. Following this intuition, we also added BN layers to the teacher discriminator. The difference between IN and BN layers in the network when generating attention can be clearly seen in Fig. 9.4, while the teacher architecture is presented in depth in Fig. 9.3 under the Teacher Batch Normalization (TBN) name.

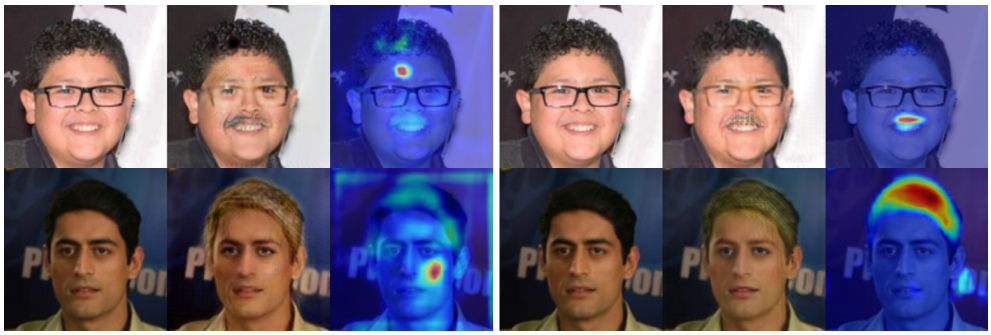


Figure 9.4: Comparison between attention obtained from a network with IN layers (on the left) as opposed to a network with BN layers (on the right). Images are disposed in a set of three: the first one is the input image, the second one is the translation output and the third one is the attention map. First row translation task is applying mustache, while the second row translation task is coloring the hair blonde.

Regarding the student architecture, in this case we choose to not have any normalization layer both in generator and discriminator as this setting slightly improves the generated images [99]. We introduced two different student models reducing feature maps dimensions and residual block number since doing so has a very heavy effect on the complexity of the network: the first student network has only 32 feature maps in the first generator layer and 3 residual block, while the second one has 16 feature maps in the first generator layer and just 1 residual block. The discriminator is the same for all two different student architectures. In Fig. 9.3 they are called, respectively, Student No Normalization Light (SNNL) and Student No Normalization Very Light (SNNVL). We defined SNNL and SNNVL because we want to reach

competitive results with these light networks to show the effectiveness of our system by using attentions as knowledge. The number of parameters for both teacher and student generators are presented in Table 9.1.

Model	Number of parameters
TBN	8.4 M
SNNL	1.2 M
SNNVL	0.16 M

Table 9.1: Number of parameters of different generators.

Finally, in order to train the full system, we defined a set of losses. We use adversarial $\mathcal{L}_{adv}$, reconstruction $\mathcal{L}_{rec}$ and domain classification loss $\mathcal{L}_{cls}$ as [9] for pre-training the teacher. On the other hand, we use the same losses to train the student with the addition of an attention distillation loss $\mathcal{L}_{att}$ for transferring the visual explanation from teacher to student.

The attention distillation loss is defined as:

$$\mathcal{L}_{att} = \mathbb{E}_{x,c}\left[\| \mathbf{A}_T^c(x) - \mathbf{A}_S^c(x) \|\right] \tag{9.3}$$

where $\mathbf{A}_T^c(x)$ is the attention map calculated over the convolutional layer of the teacher generator for an input image x and a target domain class c, while $\mathbf{A}_S^c(x)$ is the same attention but obtained from the convolutional layer of the student generator. The purpose of this loss is to push the student network to learn to generate the same attention as the teacher network in order to focus on the same areas of the input images when doing the translation and therefore producing better results.

Finally, the full student training objective is:

$$\mathcal{L}_D^{stud} = \mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}_{cls} \tag{9.4}$$

$$\mathcal{L}_G^{stud} = \mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{rec}\mathcal{L}_{rec} + \lambda_{att}\mathcal{L}_{att} \tag{9.5}$$

for the discriminator and the generator, respectively. We use $\lambda_{cls} = 1$, $\lambda_{rec} = 10$ and $\lambda_{att} = 10$ in our experiments.

9.3.3 Attention Knowledge Transfer

In this section, we will present our approach for attention distillation between the teacher and the student network. First of all, we produced visual explanations over the generator of the teacher network using the method introduced in Sec. 9.3.1 and the generated attention maps can be seen in Fig. 9.1. Analyzing these attention maps, it is clear how the better visual explanation for the image-to-image translation task can be found in the last residual block, while no useful information can be extracted from the downsampling and upsampling layers. This is a crucial observation for the development of our system, since it implies that we should direct the attention distillation loss over the maps extracted from the last convolutional layer of the last residual block, as opposed to extracting them from each layer of the teacher generator, reducing the training complexity by a considerable amount.

Training with Attention Knowledge Transfer

In order to train our student with attention loss, we first fully train the teacher network TBN. After that, we can use the pre-trained teacher discriminator model to get attention from the student and the teacher generators.

We will now present the training of the full teacher-student system. Both the generator and the discriminator of the student network are trained from scratch.

During each student training step, we trained the discriminator to distinguish between real and fake samples and to classify correctly the images into the multiple domains. On the other hand, the generator is trained to fool the discriminator by producing better samples belonging to the current target domain. The reconstruction loss serves the purpose of maintaining the content of the input image even after the translation, *i.e.* when applying mustache to a person we do not want to change its identity.

Finally, we applied the attention distillation loss to push the results from the stu-

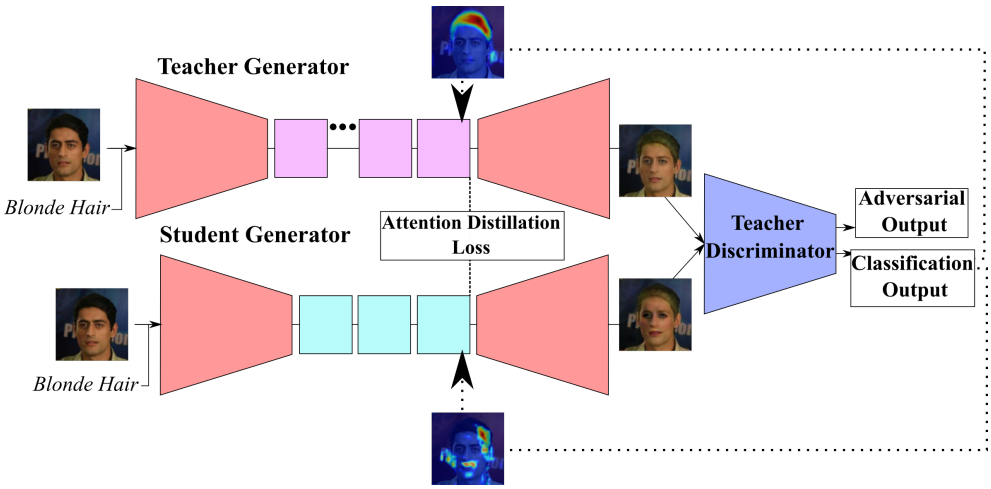


Figure 9.5: Attention distillation loss: the attention maps for the teacher and the student are calculated from the same input image and target class using the classification output of the teacher discriminator.

dent network to be as close as possible to the ones from the teacher network. More in detail, as illustrated in Fig. 9.5, given an input image x and a target domain c, we calculated the attention for both the teacher and the student. Regarding the teacher, the attention map $\mathbf{A}_T^c$ is calculated backpropagating the class score y^c from the teacher discriminator to the last convolution of the last residual block of the teacher generator. Moving to the student attention $\mathbf{A}_S^c$, we still use the teacher discriminator to backpropagate to class score to the last convolution of the last residual block of the student discriminator. The reason for not using the student discriminator to get attention is due to the fact that, especially at the beginning of the training, the student discriminator will very often miss-classify the domains of the images, and the quality of the attention strongly relies on a good classification in order to provide meaningful visual explanation.

9.3.4 Pseudo-Attentions Knowledge Transfer

In this section, we will discuss how to transfer knowledge from a teacher trained on a set of domains to a student that employs a new set of different domains. Our intuition is that when the input images are translated to a different domain, the modified areas in the samples might be shared through different domains. For example, in our task, where domains refer to facial attributes, like hair color, in order to translate an input image to the output images with attributes *black hair*, *blonde hair* or *brown hair*, the network is supposed to mainly modify the color of the region corresponding to the attribute *"hair"* in the input image. Given these observations, in this section, we start from defining the "pseudo"-attention maps and present how to derive attention distillation loss using the "pseudo"-attention in our image-to-image translation task.

Training to Transfer Knowledge with Pseudo-Attention

As introduced in Sec. 9.3.2, we employed a teacher-student paradigm in order to transfer knowledge using attention maps on a multi-domain image-to-image translation task. Firstly, given a set of target domains $\boldsymbol{c}$ for translating, we selected a number of domains $\boldsymbol{c}^{pse}$ where images of the domain c^{pse}, indicated by a certain attribute, would share common regions with the images of the domain c, indicated by another certain attribute. Next, we fully trained the teacher network (TBN), translating input images to output images belonging to the domains $\boldsymbol{c}^{pse}$, in order to calculate attention maps from it. Finally, we started training our teacher-student system, where the student network is trained to translate input images to output images of domains $\boldsymbol{c}$. While training the student network, we generated attention maps and calculated the attention distillation loss to transfer knowledge from the teacher network to the student network. More in detail, given an input image x and a target domain c, we calculated the attention maps using the corresponding domain c^{pse} for both the teacher and the student networks. For the teacher network, the class score $y^{c^{pse}}$ is backpropagated from the discriminator to the last convolution of the residual block of the generator, obtaining the attention $\mathbf{A}_T^{c^{pse}}$. Regarding the student network, attention $\mathbf{A}_S^{c^{pse}}$, is obtained by backpropagating the same class score $y^{c^{pse}}$ to the last convolution of the

residual block of the generator using the same method. Because the student network is trained to translate input images to a set of target domains $\boldsymbol{c}$, which are different domains from $\boldsymbol{c}^{pse}$, we call the attentions obtained in this way “pseudo"-attentions. Thus, the distillation loss with “pseudo"-attentions is calculated as:

$$\mathcal{L}_{att}^{pse} = \mathbb{E}_{x,c}[\| \mathbf{A}_T^{c^{pse}}(x) - \mathbf{A}_S^{c^{pse}}(x) \|] \tag{9.6}$$

Our intuition with the “pseudo"-attention distillation loss is to train the student network having the teacher network trained with a different set of domains, which however share some common regions in the images with the set of domains in the student network. In other words, one pre-trained teacher network can be used for different student networks targeting different domains by transferring knowledge with the “pseudo"-attentions that we defined.

The training objective of discriminator is the same as Equation 4 and the objective of generator in student network is:

$$\mathcal{L}_G^{stud} = \mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{rec}\mathcal{L}_{rec} + \lambda_{att}\mathcal{L}_{att}^{pse} \tag{9.7}$$

9.4 Experiments

In this section, experimental results will be presented. In particular, we will show two different experiment settings: in the first one, teacher and student networks were trained using the same set of domains and the attention was used to make the student mimic the teacher behaviour, while, in the second one the teacher were trained on a set of domains and the student on a set of different domains and the “pseudo"-attention was used as guidance to help the student to learn the new set of domains. For each experiment we will show qualitative and quantitative evaluation using FID score [51] to measure the overall visual quality of the samples and classification accuracy to determine to quality of the image translation.

All experiments were performed using the CelebA dataset [32], which is a large-scale face attributes dataset with more than 200k images and 40 attributes. Facial attributes are also very suited to show the benefits of our system, since, in order to

correctly transfer them, the network needs to learn their location in the human face and visual attention can catch this kind of information.

9.4.1 Attention-based Knowledge Transfer

In this set of experiments the objective is to use attention to transfer knowledge between a teacher network and a smaller student network like SNNL or SNNVL from Fig. 9.3. Firstly, we used TBN and SNNL architectures as our teacher and student, respectively. We trained the models to translate the input into 5 different facial attributes that are *goatee*, *rosy cheeks*, *eyeglasses*, *mustache* and *blond hair* and we transferred attention from the teacher generator last residual block to the student generator last residual block.

Fig. 9.6 shows how the attention (and therefore the translated outputs) changes after applying the attention loss. Each row presents three images on the left and three images on the right. In those sets of three images the first one is the input, the second one is the translated output and the third one is the attention map corresponding to the current domain. On the left are the results without attention loss, while on the right the attention loss was applied during the training. We can clearly see that, after the training with the attention loss, the attention maps in the student network are less noisy and they segment and identify the target domains in the image almost perfectly. In addition to that, we can also observe some qualitative changes in the output images. In particular, *blonde hair*, *rosy checks* and *mustache* domains are applied much more strongly to the images and also some undesired changes that can happen during the translation of certain domains are less frequent. Expanding on this last observation, when translating some facial attributes related to one particular gender, *i.e. goatee*, it can happen that that gender is also translated to the output image. For example, in the second row of Fig. 9.6 *goatee* is applied to a woman and we can observe how without attention loss her appearance resembles the one of a man much more than when attention loss is applied.

In order to further prove the effectiveness of our method we also performed a quantitative evaluation of these results. First of all, we fine-tuned a Resnet-50 classifier pre-trained on the ImageNet [41] on the 5 target domains used in our experiments

and we then ran it over the generated samples from our system getting a classification accuracy. Then, we calculated the FID score [51] (the lower the better) for the same samples. The results of these quantitative evaluation steps can be found in Table 9.2, respectively.

	Blonde Hair ↑	Mustache ↑	Goatee ↑	Eyeglasses ↑	Rosy Cheeks ↑	Mean ↑	FID ↓
Stargan [9] NN	82.32%	22.55%	36.67%	66.07%	43.79%	50.88%	-
TBN	63.05%	15.04%	29.53%	44.09%	33.04%	36.05%	14.97
SNNL w/o att loss	75.46%	24.71%	**51.38%**	56.48%	30.59%	47.72%	20.65
SNNL with att loss	**77.81%**	**24.87%**	48.46%	**62.65%**	**35.40%**	**49.83%**	**20.03**
SNNVL w/o att loss	**58.13%**	31.74%	64.17%	**45.46%**	10.23%	41.94%	**42.42**
SNNVL with att loss	54.91%	**45.93%**	**66.98%**	37.39%	**14.24%**	**43.89%**	46.12

Table 9.2: Classification results and FID scores over 5 different facial attributes. First row is the StarGAN model without normalization layers (acting as upper bound). The bold results represent the best results between the two sets of students SNNL and SNNVL trained without and with attention loss, respectively.

Looking at the classification results, the student model trained with attention distillation loss outperforms the student trained from scratch proving the effectiveness of our approach. Transferring the attention without any other additional information like features or classification score has showed to be enough to improve the performance of a weaker network, demonstrating that visual explanation serves more purpose than just visualization and can be used during training with success.

Regarding the FID score, SNNL trained with attention distillation loss maintains the same visual quality of SNNL trained without attention distillation loss but with a much better ability in translating the domains.

After this main experiment, we tried a more extreme test using the SNNVL network from Fig. 9.3 as student model. Quantitative results from this test are also shown in Table 9.2.

Even if this was a very difficult setting, looking at the classification results, we still managed to improve the overall results of the network, but, as the FID score indicates, with such a small network the overall visual quality of the images suffers from the constraints imposed by the attention distillation loss that, in this case, should probably be combined with others knowledge distillation methods.

9.4.2 Pseudo-Attention-Based Knowledge Transfer

In this experiment, the objective is to use “pseudo"-attentions defined in Sec. 9.3.4 to transfer knowledge between a teacher network targeting on a set of domains and a smaller student network targeting on a different set of domains. We used TBN and SNNL architectures from Fig. 9.3 also in this experiment. Firstly, we trained one teacher model (TBN^{pse}) to translate input images to 4 facial attributes (indicated as $\boldsymbol{c}^{pse}$) that are *black hair*, *blonde hair*, *wearing hat* and *wavy hair*. Next, we trained the student network (SNNL) to translate input images to 4 different facial attributes (indicated as $\boldsymbol{c}$) that are *brown hair*, *gray hair*, *bald* and *straight hair*) with the attention distillation loss to transfer knowledge between the teacher network and the student network by calculating our defined “pseudo"-attentions using the method introduced in Sec. 9.3.1.

We firstly show attentions obtained from the teacher model (TBN^{pse}) in Fig. 9.7. On the left are attention maps generated from the trained teacher network when translating input images to a set of domains $\boldsymbol{c}^{pse}$ that specifically are *wearing hat* and *black hair*. On the right are attention maps generated from another trained teacher network when translating input images to another set of domains $\boldsymbol{c}$ that specifically are *bald* and *brown hair*. From each row, we can see that attention maps of each pair of domains (*black hair* vs. *brown hair* and *wearing hat* vs. *bald*) share some common regions in the input images and this provides a strong evidence that it is possible to transfer knowledge by defining and using “pseudo"-attentions from a teacher network to a student network learning a new set of domains.

Next, we show how the attentions (and the translated outputs) change after applying the attention distillation loss with “pseudo"-attentions in Fig. 9.8. Each rows presents three images on the left and three images on the right. On the left are the results from the student network trained without attention distillation loss, while on the right the attention distillation loss with “pseudo"-attentions was applied during the training. We can see that, after the training with the attention distillation loss, the attention maps in the student network can indicate important regions corresponding to target domains (facial attributes here) in the images. Specifically, in the first row, the highlighted region in the attention map when the attention distillation loss is ap-

plied (the last image) covers more area related to the attribute *hair* than the attention map obtained without using attention distillation loss (the third image). Moreover, the generated image shown in the second last image is better than the generated image in the second image. In the second row, after training with attention distillation loss using “pseudo"-attentions, attentions indicating the attribute *hair* areas are starting to appear, which is shown in the last image, compared with the third image where it is difficult to find meaningful attentions.

To prove the effectiveness of our method further, following the quantitative evaluation experiment in Sec. 9.4.1, we also fine-tuned a classifier with a ImageNet [41] pre-trained Resnet-50 on the 4 target domains $\boldsymbol{c}$ used in our experiments. From results in Table 9.3, the student network trained with attention distillation loss using “pseudo"-attentions outperforms the network trained from the scratch, which proves the effectiveness of our approach.

	Brown Hair ↑	Gray Hair ↑	Bald ↑	Straight Hair ↑	Mean ↑	FID ↓
TBN	72.61%	14.46%	5.40%	68.60%	40.27%	29.012
SNNL w/o att loss	79.16%	32.46%	8.03%	56.48%	44.03%	27.026
SNNL with att loss	**82.77%**	**32.66%**	**12.43%**	**86.13%**	**53.5%**	**24.726**

Table 9.3: Classification results and FID scores over 4 different facial attributes. First row is the teacher network, second row is the light student network trained from scratch and finally, the last row is the light student network without normalization trained with the attention distillation loss using “pseudo"-attentions.

9.5 Summary and Future Work

In this work, we discussed the generation of visual explanation for conditional generative adversarial network and show how they can be used during training to improve the generated results. In particular, we firstly developed a system where a teacher transfer its attention to a smaller student network using an attention distillation loss in order to make the student produce better samples. We tested this approach on multi-domain image-to-image translation with facial attributes and proved that the attention transfer can help guiding the student network during training. Finally, we

tried a setup in which the teacher and the student are trained on different, but similar in shape, domains and used “pseudo"-attention obtained from the teacher to improve the training of the student.

Future work for this project mainly include improving the attention generation (since its generation is not consistent in quality for all the domains) and testing our system on a broader range of architectures.

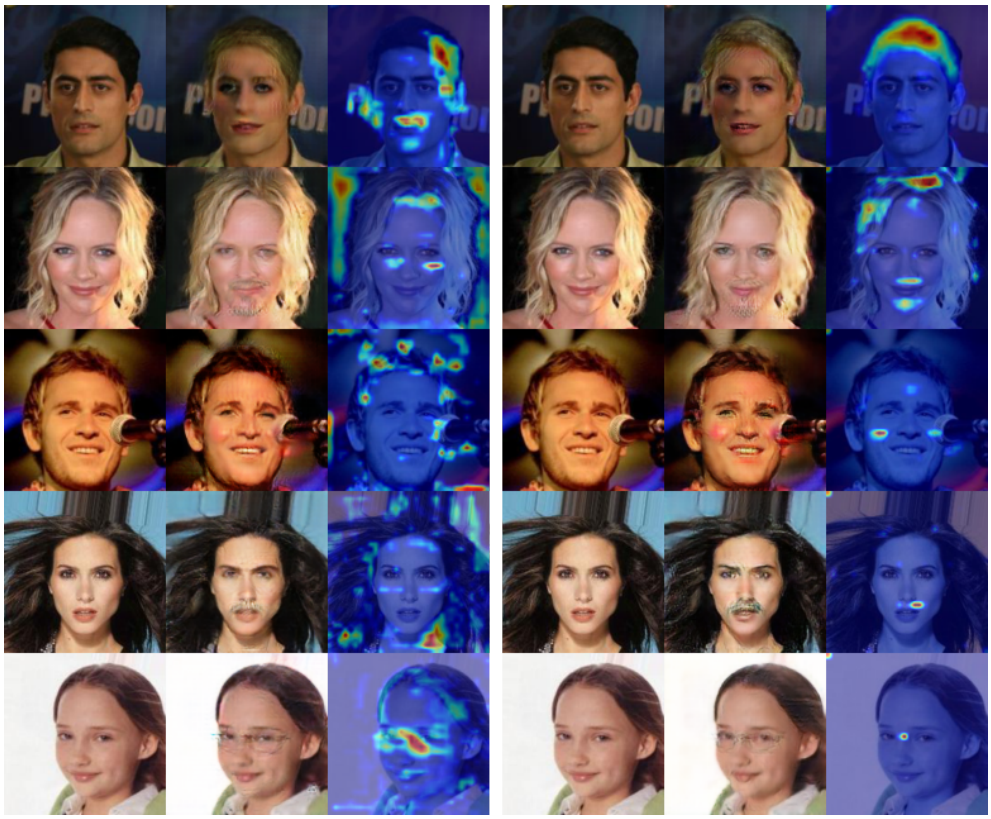


Figure 9.6: Results without attention loss (left) and with attention loss (right) in SNNL. Translation tasks starting from the first row are: *blonde hair*, *goatee*, *rosy cheeks*, *mustache* and *eyeglasses*.

Figure 9.7: “Pseudo"-Attention maps generated from teacher networks targeting on two sets of domains. From left to right, attributes in first row are: *wearing hat*, *bald*; Attributes in second row are: *black hair*, *brown hair*

Figure 9.8: Results without attention distillation loss (left) and with attention distillation loss using “pseudo"-attentions (right) in SNNL. Domains to translate to: starting from the first row are *brown hair*, *gray hair*, *bald* and *straight hair*.

Part III

Conclusions

Never tell me
the odds.

This thesis focused on improving conditional GANs in order to solve several challenges ranging from coloring fashion image, to combining cGANs with meta-learning to build a more flexible system, to extracting attention from cGANs and use it as knowledge in a teacher-student paradigm.

In chapter 6 a system designed to automatically colorize shoe models was presented. The system consisted of a conditional GAN conditioned with a fully colored right side of a shoe and was trained to generate a colored version of the left side. The objective was to ease the work of the designers that would not need to manually colorize the full shoe model but only one of its side. This means that they could focus more on the creative side of their work and less on the repetitive actions. The model was able to produce a very sharp and consistent colorization, learning also to consider the fundamental relations between the colors inside a shoe model like, for example, the three stripes often found in Adidas shoes that were always automatically colored with the same tone. A complete ablation study with different architectures and losses was explored in order to design the system and both a quantitative and qualitative evaluation was performed to validate the obtained results.

In chapter 7 a conditional GAN was combined with the meta-learning algorithm Reptile in order to learn to colorize gray-scale images with a small quantity of data at disposal. In other words, the generative adversarial network was used for the actual colorization, while the meta-learning technique was employed to enhance the generator model. To ensure a task-oriented division, that is common in the approaches based on meta-learning, a clusterization of the initial dataset was performed based on the feature extracted from a pre-trained classification network. The system proved to be able to produce a good colorization even with a very small dataset used for training, outperforming a standard cGAN colorization network.

In chapter 8 the concept of combining meta-learning and conditional GANs was refined in order to enhance the performance of a multi-domain image-to-image translation network. A novel architecture was proposed that is able to translate an image into multiple domains, in this case facial attributes, using a single architecture. In

addition to that, no hard-coded labels are used during training allowing for a greater flexibility. This was possible thanks to the combination of an image-to-image translation cGAN and the Reptile meta-learning algorithm. Each iteration the network was cloned and trained on a single random domain. After that, the parameters of the cloned network were merged with the ones of the initial network following the Reptile equations. The system reached state-of-the-art results on multi-domain image-to-image translation and in addition it was possible to add new unseen domains after the training with a brief fine-tuning step requiring a very small quantity of images.

Finally, in chapter 9 the main objective was to provide visual explanation for condition GANs and to use the generated attention maps as knowledge to be distilled in a teacher-student paradigm for image-to-image translation tasks for improving the results of the student architecture. Furthermore, “pseudo"-attention were designed in order to be employed during training when teacher and student were trained on different domains that still share some similarities, like hair colors for example. A full quantitative and qualitative evaluation was performed over facial attributes transfer proving that the visual attention can be effectively used during training.

To conclude, in this thesis we explored several different ways of conditioning generative adversarial networks and showed in particular that meta-learning is a powerful tool to be used in this context. Indeed, meta-learning can remove the need for supervision during training and allows to add new unseen domains during inference. In addition to that, with meta-learning it is also possible to train the network using a very small dataset making it suitable for few-shot learning. The main drawback of this approach is that meta-learning add an inner cycle of meta-iteration making the training of the network much slower than a standard training. Nevertheless, a good tradeoff between speed and results can be achieved by reducing the number of meta-iteration and using a small batch size. Another limitation of meta-learning consists in the need of an inference step to produce output samples. This further slow down its application.

Chapter 10

Future works

Regarding shoe colorization, future works includes mainly two aspects. First of all, a step further in the conditioning would be to be able to use a different view of the shoe, like the top view, to guide the colorization. This would imply that the network needs to learn to translate the perspective in order to generate a side view from a top view. Secondly, a great improvement in the system would be to generate a fully colored side of a shoe model using a fully colored side of a different shoe model. In other words, a system able to translate a color style from different models maintaining in the output sample the shape of the input shoe and the color correlations between the different parts of the target shoe.

Furthermore, regarding future works about the combination of meta-learning and conditional GANs, the first thing to explore would be improving the speed of the overall system. Indeed, meta-learning tends to slow down the training of the cGAN by a considerable amount and reducing the training time would be a great improvement. In addition to that, exploring novel meta-learning algorithms and changing the way meta-learning is applied to the network, for example selecting only a few layers of the network to be included in the meta-training step, could push the results of the system even further.

Finally, for visual explanation and attention generation in cGANs, next steps will consists into improving the visual attention generation and testing the system on a

broader range of architectures. Regarding the first consideration, the visual explanation quality is not consistent in all of the tested domains and therefore needs to be improved in order to have better generated samples. Furthermore, testing the attention transfer approach on different architectures would prove even more its effectiveness and robustness.

Conditioning generative adversarial network was proved to be a fundamental part of their training and allowed to have control over their output and to produce better results. Nevertheless, exploring new ways of doing that will be the main objective of future works. In particular, a novel way of achieving control over the generated samples is focusing over the latent space of the model. Imposing rules and regularization in the inner representation of the network can indeed allow to move the generated output in the desired direction and greatly boost the capability to learn from few sample, pushing the few-shot approach even more. In addition to that, meta-learning can help to model the latent space by dividing the learning problem in a set of tasks and focusing on learning the optimal latent geometry for each of them individually. In addition to that, pushing to a truly few-shot approach to image generation will be a very interesting area of future research since existing approach are more about few-shot style transfer or domain adaptation.

Publications

Journal Papers:

- Tomaso Fontanini, Eleonora Iotti, Luca Donati and Andrea Prati. “MetalGAN: Multi-domain label-less image synthesis using cGANs and meta-learning” Neural Networks, 2020

Book Chapters:

- Federico Magliani, Tomaso Fontanini, and Andrea Prati. "Landmark Recognition: From Small-Scale to Large-Scale Retrieval." Recent Advances in Computer Vision. Springer, Cham, 2019. 237-259.

Conference Papers:

- Federico Magliani, Tomaso Fontanini, and Andrea Prati. "Efficient Nearest Neighbors Search for Large-Scale Landmark Recognition." International Symposium on Visual Computing. Springer, Cham, 2018.
- Federico Magliani, Tomaso Fontanini, and Andrea Prati. "A Dense-Depth Representation for VLAD Descriptors in Content-Based Image Retrieval." International Symposium on Visual Computing. Springer, Cham, 2018.
- Tomaso Fontanini, Eleonora Iotti and Andrea Prati. "MetalGAN: a Cluster-based Adaptive Training for Few-Shot Adversarial Colorization" International Conference on Image Analysis and Processing, 2019.

- Tomaso Fontanini, Runze Li, Luca Donati, Andrea Prati and Bir Bhanu "Conveying knowledge in conditional GANs with attention distillation" CVPR (under review).

Workshop Papers:

- Giulio Angiani, Laura Ferrari, Tomaso Fontanini, Paolo Fornacciari, Eleonora Iotti, Federico Magliani, Stefano Manicardi "A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter." KDWeb. 2016.

Bibliography

[1] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[4] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[5] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[7] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[8] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2018.

[9] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018.

[10] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[12] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[13] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[14] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[15] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[16] Wenqian Liu, Runze Li, Meng Zheng, Srikrishna Karanam, Ziyan Wu, Bir Bhanu, Richard J Radke, and Octavia Camps. Towards visually explaining variational autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8642–8651, 2020.

[17] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[18] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5138–5146, 2019.

[19] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[20] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2017.

[21] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[22] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[23] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[26] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[27] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[28] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[29] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[30] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[32] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[33] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[34] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10551–10560, 2019.

[35] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

[36] Ke Li and Jitendra Malik. Learning to optimize neural nets. *arXiv preprint arXiv:1703.00441*, 2017.

[37] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989, 2016.

[38] Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. *arXiv preprint arXiv:1703.04813*, 2017.

[39] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[40] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems*, pages 2365–2374, 2018.

[41] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[42] Louis Clouâtre and Marc Demers. Figr: Few-shot image generation with reptile. *arXiv preprint arXiv:1901.02199*, 2019.

[43] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[44] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847. IEEE, 2018.

[45] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9592–9600, 2019.

[46] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[47] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *arXiv preprint arXiv:1705.02999*, 2017.

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[50] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[51] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[52] Ali S Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. Visual instance retrieval with deep convolutional networks. *ITE Transactions on Media Technology and Applications*, 4(3):251–258, 2016.

[53] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision*, pages 392–407. Springer, 2014.

[54] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *European conference on computer vision*, pages 584–599. Springer, 2014.

[55] Joe Yue-Hei Ng, Fan Yang, and Larry S Davis. Exploiting local features from deep networks for image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 53–61, 2015.

[56] Konda Reddy Mopuri and R Venkatesh Babu. Object level deep feature pooling for compact image representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 62–70, 2015.

[57] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. *arXiv preprint arXiv:1511.05879*, 2015.

[58] Yun Cao, Zhiming Zhou, Weinan Zhang, and Yong Yu. Unsupervised diverse colorization via generative adversarial networks. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 151–166. Springer, 2017.

[59] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[60] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[61] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[62] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[63] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[64] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[65] Danilo Jimenez Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*, 2016.

[66] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

[67] Sasi Kiran Yelamarthi, Shiva Krishna Reddy, Ashish Mishra, and Anurag Mittal. A zero-shot framework for sketch based image retrieval. In *European conference on computer vision*, pages 316–333. Springer, 2018.

[68] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

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

[70] Jürgen Schmidhuber. Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). *Neural Networks*, 2020.

[71] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[72] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in neural information processing systems*, pages 465–476, 2017.

[73] Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. *arXiv preprint arXiv:1802.10151*, 2018.

[74] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.

[75] Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen. Attgan: Facial attribute editing by only changing what you want. *IEEE Transactions on Image Processing*, 28(11):5464–5478, 2019.

[76] Taihong Xiao, Jiapeng Hong, and Jinwen Ma. Dna-gan: Learning disentangled representations from multi-attribute images. *arXiv preprint arXiv:1711.05415*, 2017.

[77] Taeksoo Kim, Byoungjip Kim, Moonsu Cha, and Jiwon Kim. Unsupervised visual attribute transfer with reconfigurable generative adversarial networks. *arXiv preprint arXiv:1707.09798*, 2017.

[78] Tomaso Fontanini, Eleonora Iotti, and Andrea Prati. Metalgan: A cluster-based adaptive training for few-shot adversarial colorization. In *International Conference on Image Analysis and Processing*, pages 280–291. Springer, 2019.

[79] Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems*, pages 5228–5237, 2018.

[80] Oliver Langner, Ron Dotsch, Gijsbert Bijlstra, Daniel HJ Wigboldus, Skyler T Hawk, and AD Van Knippenberg. Presentation and validation of the radboud faces database. *Cognition and emotion*, 24(8):1377–1388, 2010.

[81] Kunpeng Li, Ziyan Wu, Kuan-Chuan Peng, Jan Ernst, and Yun Fu. Guided attention inference network. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[82] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[83] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1947–1962, 2018.

[84] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[85] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708, 2017.

[86] Md Mahfuzur Rahman Siddiquee, Zongwei Zhou, Nima Tajbakhsh, Ruibin Feng, Michael B Gotway, Yoshua Bengio, and Jianming Liang. Learning fixed points in generative adversarial networks: From image-to-image translation to disease detection and localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 191–200, 2019.

[87] Po-Wei Wu, Yu-Jing Lin, Che-Han Chang, Edward Y Chang, and Shih-Wei Liao. Relgan: Multi-domain image-to-image translation via relative attributes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5914–5922, 2019.

[88] Tomaso Fontanini, Eleonora Iotti, Luca Donati, and Andrea Prati. Metalgan: Multi-domain label-less image synthesis using cgans and meta-learning. *Neural Networks*, 131:185–200, 2020.

[89] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[90] Hiroshi Fukui, Tsubasa Hirakawa, Takayoshi Yamashita, and Hironobu Fujiyoshi. Attention branch network: Learning of attention mechanism for visual explanation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10705–10714, 2019.

[91] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

[92] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[93] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[94] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Advances in neural information processing systems*, pages 742–751, 2017.

[95] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019.

[96] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[97] Angeline Aguinaldo, Ping-Yeh Chiang, Alex Gain, Ameya Patil, Kolten Pearson, and Soheil Feizi. Compressing gans using knowledge distillation. *arXiv preprint arXiv:1902.00159*, 2019.

[98] Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5284–5294, 2020.

[99] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems*, pages 3927–3936, 2019.

Acknowledgments

Grazie ad Andrea Prati per la guida ed il supporto durante questi tre anni.

Grazie a Jochen Suessmuth per lo stimolo continuo.

Grazie ai miei colleghi Luca, Leonardo, Eleonora, Federico ed Akbar.

Grazie a tutti i tesisti e a tutti i Professori del Dipartimento di Ingeneria e Architettura.