Image inpainting using generative adversarial networks (GANs)

A Thesis

submitted to the designated
by the General Assembly
of the Department of Computer Science and Engineering
Examination Committee

by

Eleftherios Panagiotis Kostakis

in partial fulfillment of the requirements for the degree of

COMPUTER SCIENCE AND ENGINEERING

University of Ioannina

October 2021

Examining Committee:

- **Christophoros Nikou**, Professor, Department of Computer Science and Engineering, University of Ioannina (Supervisor)
- **Lisimachos P. Kondi**, Professor, Department of Computer Science and Engineering, University of Ioannina
- **Marina Plissiti**, Member of the Laboratory Teaching Staff, Department of Computer Science and Engineering, University of Ioannina

Acknowledgements

I am profoundly grateful to professor Michalis Vrigkas for his guidance and the constructive discussions we had. I also want to thank the University of Ioannina for providing me with the necessary equipment to conduct the experiments.

Table of Contents

Abstract

Image inpainting refers to the art of repairing a damaged image. As a consequence of advancements in computation power, image inpainting task became widely deployed in the field of computer vision, coming up with a variety of deep-learning-based implementations. Generative adversarial networks (GANs) are one of the biggest innovations of the last decade in deep generative modeling, because of their ability to generate extremely plausible content and therefore had quite an influence in image inpainting domain. In order to further improve inpainting results, attention mechanisms are introduced. Attention mechanisms are an essential part of many generative model structures and just like human visual attention system works, attention is integrated in order to enhance specific regions in the image. During the training process of a GAN, the features of the training images are compressed in the latent space of the GAN. In order to reconstruct a given image, its latent representation has to be inferred from the latent space and this method is known as GAN inversion. However, there is a knowledge gap concerning the inversion process of a GAN. To bridge this gap, Gu $et\ al.$ [8] introduced $mGANprior$, an optimization-based inversion method. In this thesis, the aforementioned method is extensively analyzed and modified, incorporating an attention mechanism in an attempt to achieve better inpainting results. In the experiments, $mGANprior$ and the modified model are compared, showing that the results between them are almost identical.

Περιληψη

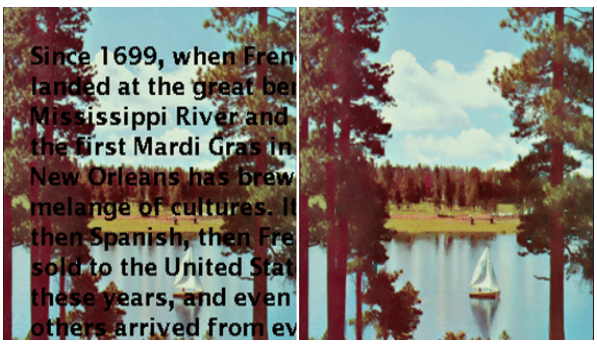
Η εκτίμηση άγνωστου περιεχομένου της εικόνας αναφέρεται στην τέχνη της επιδιόρθωσης μιας κατεστραμμένης εικόνας. Ως συνέπεια των εξελίξεων στην υπολογιστική ισχύ, η εργασία για την εκτίμηση άγνωστου περιεχομένου εικόνας αναπτύχθηκε ευρέως στον τομέα της υπολογιστικής όρασης, καταλήγοντας σε μια ποικιλία εφαρμογών βασισμένων στη βαθιά μάθηση. Τα παραγωγικά αντιπαλικά δίκτυα (ΠΑΔ) είναι μία από τις μεγαλύτερες καινοτομίες της τελευταίας δεκαετίας στη βαθιά δημιουργική μοντελοποίηση, λόγω της ικανότητάς τους να δημιουργούν εξαιρετικά αληθοφανές περιεχόμενο και ως εκ τούτου είχαν μεγάλη επιρροή στον τομέα της εκτίμησης άγνωστου περιεχομένου της εικόνας. Προκειμένου να βελτιωθούν περαιτέρω τα αποτελέσματα της εκτίμησης άγνωστου περιεχομένου, εισάγονται μηχανισμοί προσοχής. Οι μηχανισμοί προσοχής αποτελούν ουσιαστικό μέρος πολλών παραγωγικών μοντέλων και όπως ακριβώς λειτουργεί το σύστημα οπτικής προσοχής του ανθρώπου, η προσοχή ενσωματώνεται προκειμένου να ενισχυθούν συγκεκριμένες περιοχές της εικόνας. Κατά τη διάρκεια της διαδικασίας κατάρτισης ενός ΠΑΔ, τα χαρακτηριστικά των εκπαιδευόμενων εικόνων συμπιέζονται στο κρυμμένο χώρο του ΠΑΔ. Για να αναδημιουργηθεί μια δοθείσα εικόνα, η κρυμμένη αναπαράστασή της πρέπει να συναχθεί από τον κρυμμενό χώρο και αυτή η μέθοδος είναι γνωστή ως αντιστροφή του ΠΑΔ. Ωστόσο, υπάρχει ένα χάσμα γνώσης σχετικά με τη διαδικασία αντιστροφής ενός ΠΑΔ. Για να γεφυρωθεί αυτό το χάσμα, ο Gu κ.ά. [8] εισήγαγε το mGANprior, μια μέθοδο αναστροφής που βασίζεται στη βελτιστοποίηση. Σε αυτή τη διατριβή, η προαναφερθείσα μέθοδος αναλύεται και τροποποιείται εκτενώς, ενσωματώνοντας έναν μηχανισμό προσοχής σε μια προσπάθεια να επιτευχθούν καλύτερα αποτελέσματα της εκτίμησης άγνωστου περιοχομένου της εικόνας. Στα πειράματα, το mGANprior και το τροποποιημένο μοντέλο συγκρίνονται, δείχνοντας ότι τα αποτελέσματα μεταξύ τους είναι σχεδόν πανομοιότυπα.

Chapter 1

Introduction

(a) Image restoration.

(b) Object removal.



(c) Text removal.

(d) Face inpainting.

1.1 Motivation and thesis goal

Inpainting, also called *retouching*, was used first by artists for painted public images restoration. The task of image restoration was used in damaged artworks to maintain their perception quality. Later, the inpainting tasks passed naturally on film and photography, while the purposes were the same. Nowadays, image inpainting is mainly done by computers with great success. The superiority of digital inpainting includes demanding inpainting tasks that are almost impossible to be implemented by an artist, such as object and text removal. Moreover, digital image inpainting succeeds to a larger extent in both contextual and textural inpainting at a much better time. Hence, image inpainting can be described as the process of filling missing pixels in an image in order to follow its original context with high level of photo-realism. One of the most promising and impressive innovations of the last decade are, without doubt, the generative adversarial networks [1] and a great amount of research is done on image inpainting with GANs. GANs have already become widely known for their outstanding results in generating data and more specifically images, videos and speech voices among others. Their capability of producing high-quality images make GANs applicable to many image processing tasks, such as face editing [2], super-resolution [3] and image inpainting [4]. The objective of GANs is to learn how to generate photo-realistic images from a latent distribution and the state-of-the-art GANs, such as PGGAN [5] and StyleGAN [6] are capable of synthesizing a wide variety of high-quality images with great success.

In order to reconstruct and modify real images for the purposes of various image processing tasks, it is indispensable to better understand and interpret the generation process of GAN models. These models encapsulate rich semantics in a low-dimensional space that is called latent space and it can be explored with different techniques forming various latent spaces, such as the $\mathcal{Z}$ space or $\mathcal{W}$ space. Manipulating the latent space $\mathcal{Z}$ of a GAN model should result in different outcomes in image space $\mathcal{X}$, although the fact that a GAN model cannot output a latent code $z \in Z$ from a given image is a significant limitation. To mitigate this limitation, many attempts have been made to explore and interpret the $\mathcal{Z}$ space through an extra encoder model [7] or more successfully and cost-efficient through optimization of the latent code z [8, 9, 10]. Despite the efforts that have been made, inverting the gener-

ator of GAN models remain a subject to be explored further. An attempt with very impressive results came from Gu *et al.* [8], who introduced multi-code GAN prior ($mGANprior$) model that demonstrated the necessity of using multiple latent codes in order to faithfully reconstruct a given image and also make use of the interpretative codes in terms of semantics.

Just like human vision focuses on particular areas of an image and it is easily to predict certain regions, e.g., when upper part of a dog is attended, four legs are expected to be captured in the lower part of the image. Similarly, attention mechanisms in fields of computer vision indicating or predicting the important regions in order to improve the perception of a given image. Attention mechanisms in computer vision was first introduced by Larochelle *et al.* [11], although it became widely popular after Mnih et al. [12] introduced a recurrent neural network model that outperformed the state-of-the-art in image classification tasks at the time. Bahdanau *et al.* [13] introduced a novel attention mechanism in neural machine translation, that made a tremendous impact and later was integrated in various research papers in computer vision. Attention module was initially integrated in GAN architecture from Zhang *et al.* [14], introducing SAGAN, which integrated an attention mechanism in the generator and the discriminator of the model and later attention mechanism used also in BigGAN [15], which is used for high fidelity natural image synthesis.

As mentioned before, Gu *et al.* [8] proposed the increase of the number of latent codes to improve the inversion quality, as with a single latent code it is not possible to reconstruct any high resolution image at a satisfying level of photo-realism and therefore no image processing task can be applied. The $mGANprior$ model is chosen to get modified, because of partially overcoming the big issue that GAN models present: taking a given image as the input to infer its latent code. This is accomplished by composing the feature maps of multiple latent codes at a particular layer of a pre-trained GAN model in order to reconstruct the input image. Furthermore, $adaptive\ channel\ importance$ is introduced, which works as an attentive mechanism. Hence, the fact that $mGANprior$ can achieve great results in various image processing tasks, such as semantic manipulation and image inpainting, gives the ability to enable pre-trained GAN models as prior to the aforementioned tasks without retraining.

1.2 Thesis contributions

In this thesis, the reverse of generation process has been considered by inverting a given image from the image space back to the latent space [16] for image inpainting task, using the general concept of $GAN\ inversion$. More specifically, modification of inversion process of the generator has been made, integrating an extra attention layer in a pre-trained GAN model. Both $convolutional\ block\ attention\ module\ (CBAM)$ [17] and $self-attention$ [18] with dot-product and scaled dot-product scoring functions are tested in order to improve inverted image quality during the optimization process of the multiple latent codes. Moreover, the modified inversion of the generator has been tested in various datasets and multiple hyper-parameters modifications have been used to get the optimal inversion quality. In contrast to authors of $mGANprior$, results were better with different hyper-parameters settings, e.g., composing layer or number of latent codes, than the proposed (default) options.

Chapter 2

Related work

Among the most important capabilities of deep convolutional neural network is capturing and using a great amount of low-level image statistics as prior to any learning. Generative models that contain such statistical information, are capable of coping with standard generator model inversion problems, such as super-resolution and image inpainting. As already mentioned in Ch. 1 $mGANprior$ introduced by Gu $et\ al.$ [8] is a model that can be used as prior to various image processing tasks. Despite the fact that the research field of using models as prior to image processing tasks is in a preliminary stage, there are several interesting works in the literature, that make use of pre-trained networks or use techniques without any beforehand training process. More precisely in this research field, Ulyanov $et\ al.$ [19] introduced $deep\ image\ prior$ and demonstrated that there are image priors which are not trained on large-scale dataset of example images, and still are able to bring in state-of-the-art results in image processing by capturing image statistics from the structure of the generative network. For this task, the only things needed for the reconstruction are the degraded input image and the structure of the deep convolutional network. In other words, in order to search for the answer in the image space, $deep\ image\ prior$ search for it in the space of neural network's parameters. Yet another remarkable work was done by Upchurch $et\ al.$ [20], who introduced $deep-feature\ interpolation$ (DFI), a new technique at the time, used for semantic image transformation on high-resolution images. More concretely, DFI relies only on simple linear interpolation of deep convolutional features from pre-trained convolutional networks. DFI achieves great results, similar

to the state-of-the-art, in tasks of semantic manipulation (mentioned as image transformation), like aging or adding facial hair. However, the importance of the particular research paper stands in the simplicity and the flexibility of the proposed method, as it does not need any specialized network architecture or any pre-trained deep convolutional network.

Attention-based methods for image inpainting is an even more recent research field. Attention mechanism in combination with a GAN-based or CNN-based architectures provides additional guidance in order to capture semantics in the image. To this end, Yu $et\ al$ [4] introduced generative image inpainting with contextual attention, which uses the WGAN-GP loss [21]. In an attempt to improve missing parts of structures and textures in images, this method make use of surrounding image features to make better predictions to help the inpainting process. It is divided into two stages, where the first stage consists of a dilated convolutional network that is trained with reconstruction loss to make an initial prediction about the missing regions in the image, which is producing a blurry outcome. The second stage which produces the final outcome, consists of a refinement network, in which the contextual attention is integrated. Contextual attention implementation is based on the features of known patches, which are used as convolutional filters to process the generated patches. Another significant attention-based work comes from Liu [22], who proposed $coherent\ semantic\ attention$ (CSA) layer. U-Net architecture is being modified, inserting the CSA layer in the encoder of the refinement part, which constructs the correlation between the deep features of hole regions. An even more recent work with remarkably results comes again from Liu $et\ al$ [23] and is related to facial inpainting. An attention-based multi-level generative network is being used to maintain soft correlation with the neighbouring regions. Uddin $et\ al$ [24] proposed a coarse-to-fine architecture using both local and global attention for free-form image inpainting. The combination of global contextual attention and local similarity information is used along with a mask feature pruning mechanism, resulting in a great improvement in terms of structural consistency and texture clarity. The coarse network is an encoder-decoder network, which calculates the global dependencies among the features and contains the mask pruning layer, in which the less important features are getting pruned. In the refinement network, pruned feature patches are used for the calculation of the local similarities. The results are quite encouraging, however this method is limited to free-form masks.

Chapter 3

Image inpainting

(a) Image inpainting applied on a scratched image.

(b) Image inpainting for a large missing region.

In the real world, it is usual that images suffer from noise. It can be a scratch on the lens or just a kid painted them or even it is a whole object that has to be removed. Image inpainting algorithm can be applied for image restoration purposes, such as a scratch or text removal [25] or photo-editing tasks like removal of an object [26, 27]. To restore the damaged image by filling the missing pixels has been a research field for many years and there are many inpainting methods in the literature divided into sequential-based, CNN-based, and GAN-based approaches.

3.1 Sequential-based approaches

Classic image inpainting methods [28] and sequential-based conventional approaches are either patch-based or diffusion-based methods [29, 30], which in order to fill in the missing pixels they use variational algorithms or searching for similar patches in the image. More common are the patch-based methods and there are many implementations. Ružić *et al.* [31] proposed a Markov random field based method, while Kawai *et al.* [32] showed how to remove real objects from video images and fill in the missing regions with plausible background textures in real-time.

The aforementioned techniques are tested in a big recent survey [33] and brings satisfactory results. Although these methods are doing well with textural synthesis, they cannot deliver when the missing region is too large or dealing with non-repetitive texture, because these methods cannot create new content for challenging inpainting tasks, but they search for a similar patch or content in image to fill in the hole. Thus, it is impossible to capture high-level semantics like face features or an object and are limited to work well only with low-level features of an image. Moreover, searching for a compatible patch among thousands of images in a dataset can be extremely time-consuming.

3.2 CNN-based approaches

Computer vision algorithms are used almost in every smart device and technology machine. All these algorithms have the same architecture, a convolutional neural network (CNN), with different adjustments. That was a breakthrough in computer vision tasks, like semantic image segmentation, object detection and also had a big impact on image inpainting achieving great results. CNNs are a feed-forward network and consist of building blocks, such as convolution, pooling and fully-connected layers in order to extract the deep features of the input image. Image filters (sets of weights) are being applied to an input image (represented as a matrix) and create a stack of feature maps, that indicates the depth of the network, each one containing different deep features (like textures or edges) of the input image. Like conventional neural networks, CNNs also use the technique of backward propagation in order to update the error in aspect to all the weights (and biases) and stochastic gradient descent to minimize the cost.

That being said, convolutional neural networks can overcome the problem that sequential based methods appear. A CNN is trained in very large datasets and gains the ability to capture image semantics and global structure of the input data and even more to recreate a large missing region. There are already many implementations using CNNs in computer vision and more specifically for the task of image inpainting. Shift-Net [34] modified U-Net [35] architecture, adding a shift-connection layer to it, achieving sharp structures and fine-detailed textures. One of the most common implementations for image inpainting using CNNs are the encoder-decoder network structure. Pathak *et al.* [36] introduced an encoder-decoder network called context encoder, that is capable of capturing the global structure and the semantics of visual structures. Another implementation using encoder-decoder architecture is made by Zhu *et al.* [37] for inpainting forensics. CNN-based approaches have expanded to almost all image inpainting tasks, such as removal of an object and inpaint the region from a video using VORNet [38] approach and text removal from images introduced by Nakamura *et al.* [39].

3.3 GAN-based approaches

GANs architecture is widely deployed in computer vision applications and without a doubt had an impact on image inpainting. However, comparing to CNN-based methods, GAN-based approaches are relatively a few. More concretely, Chen *et al.* [40] introduced *progressive inpainting*, a GAN-based method for semantic image inpainting. In this approach the generator of GAN is used to inpaint the missing regions, while it uses back-propagation in order to find the appropriate input image distribution. The progressive strategy in the method is similar to progressive GAN training strategy, as it is a pyramid-like strategy, starting from a low-resolution image to a higher one. Another significant proposal is introduced by Shin *et al.* [41], is a network called *PEPSI++*. It is a modified coarse-to-fine network with a contextual attention module, that consists of a single shared encoding network and parallel decoding networks. Its primary contribution is the reduced training time along with great inpainting results.

Wang *et al.* [42] proposed a new inpainting algorithm based on *context encoders* [36], which is a multi-scale adversarial network that consists of a global and a local discriminator network. The local discriminator is a four-layered convolutional network, that takes as input a locally generated image and the output is a real number from 0 to 1, which is used to discriminate the local consistency. On the other hand, global discriminator is also a convolutional network with five layers and similar to local discriminator outputs a real number from 0 to 1 and produces the final output image with replacing the missing region with the generated image. Vitoria *et al.* [43] proposed a semantic image inpainting method, that modifies WGAN [44] introducing a new implementation of both the generator and the discriminator. Along with the WGAN modification, a new optimization loss is introduced, that makes use of the essential semantic information of the degraded image and the contextual information derived from the gradients and image values. Another approach found in the literature comes from Dong *et al.* [45], in which a deep convolutional GAN (DCGAN) is used for the restoration of the missing regions. Han *et al.* [46] introduced FiNet, shorted for fashion inpainting networks, whereas Liu *et al.* [47] proposed an inpainting method, in which the neighborhood loss function and gradient loss are used in repair process.

Chapter 4

Generative adversarial networks (GANs)

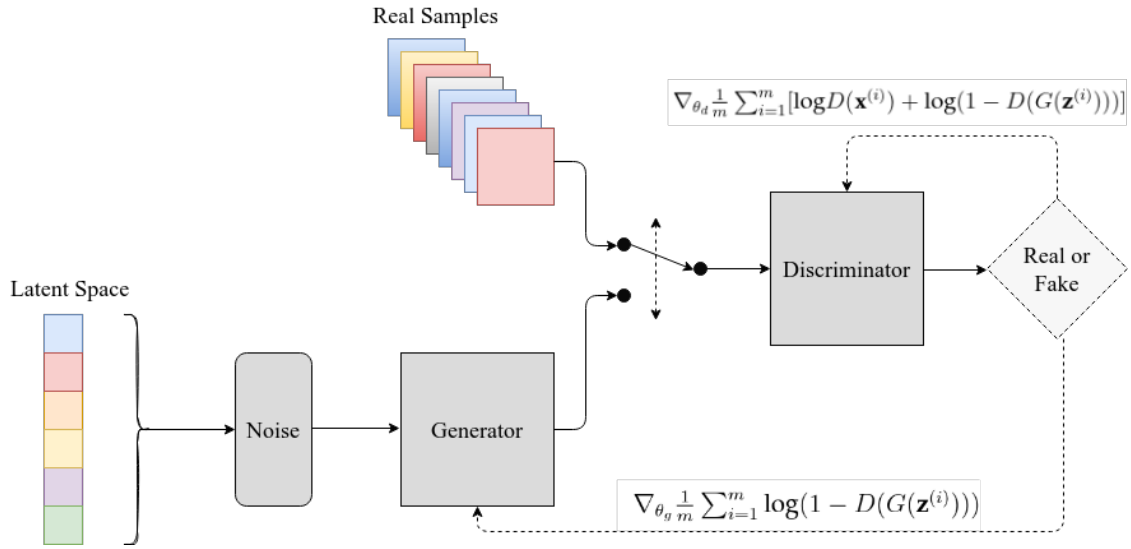


Figure 4.1: General concept of a GAN

Generative adversarial networks [1] are a class of semi-controlled learning and unsupervised learning algorithms used to produce images and videos, generate sym-

bols, creating audio signals and speeches and so on. In this approach, two neural networks compete with each other in a game (zero-sum game in game theory). Generative adversarial networks main ability lies in generating new content from a given distribution. The two main entities are the generator (G) and the discriminator (D), which compete with each other until the generator learns the ability to generate real-looking images in order to fool the discriminator that the input image is coming from the real distribution instead of the fake one. On the other hand, the discriminator's role is to distinguish the real data from the generated. The convergence is succeeded when the generator is able to fool the discriminator around half of the time and the discriminator obviously not to get fooled the other half of time. During the process of GANs training, the main goal is to reach convergence. That means forcing the generated distribution to get as close as possible to the true distribution. Generator and discriminator are both neural networks and they can be trained jointly, with the objective function (4.1) represented as a minimax function, where the generator and the discriminator tries to minimize and maximize it respectively.

$$\min_{\theta_g} \max_{\theta_d} [\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))] \tag{4.1}$$

Over the years, there have been plenty of architectures modifying an existing GAN, such as WGAN [44] and WGAN-GP [21] or coming up with new ideas[5, 6, 48, 49] with varying loss functions and implementations. That is happening because of different tasks they are called to accomplish, not only in computer vision, but also in audio [50] and text [51] as well. The most popular GAN architectures are the deep convolutional GAN (DCGAN) [49] and the Wasserstein GAN (WGAN) [44] known for improving the stability of the training process in GANs. Another popular extension to the GAN is CycleGAN [48] and it is used to learn unsupervised image-to-image translation between images of two different domains. Also, there are GANs for generating images of very high-resolution, such as the StyleGAN [6] and progressive growing GAN (PGGAN) [5].

As already mentioned the GAN training process consists of two competing neural networks (generator and discriminator), which are optimizing the minimax GAN loss (4.1) in parallel. That means, the discriminator trains for one or more epochs

and similar to it, the generator also trains for one or more epochs, as this process is being repeated until the convergence. Generator has to be kept constant during the discriminator training phase, in order to find flaws in the generator network. At first, the generator samples from a distribution and adds noise to input samples, resulting in completely random outputs. To prevent learning an untrained generator's flaws, the technique of alternating training is being used, where generator and discriminator are trained by turns. When convergence is reached, the generator should fool the discriminator around half of the times and exactly half of the times when it is trained perfectly. However, in practice, it is almost impossible to have perfect training. Hence, the main goal of the generator stands in producing samples that follow the real distribution and for the discriminator is to make equal predictions.

Algorithm 4.1 Minibatch stochastic gradient descent training of generative adversarial networks as it was firstly introduced in [1].

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)},...,\mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)},...,\mathbf{x}^{(m)}\}$ from data generation distribution $P_{data}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)})))].$$

end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)},...,\mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(\mathbf{z}^{(i)}))).$$

end for

Nevertheless, GAN training is a difficult process and many issues might arise. In GAN, it is a usual phenomenon to never achieve the desired convergence, where the model parameters destabilize and never converge. Also, the problem of diminished gradient is a possibility, when the discriminator is too good and the generator's gradient vanishes. As well, the generator might not produce a variety of noise samples and stick to a single mode that leads the network to mode collapse. In general, it is quite a demanding task to reach convergence in a minimax game, where there is the possibility one player outsmarts the other continuously.

4.1 GAN inversion

The task of GAN inversion refers to inverting the generation process of a fixed GAN model. In order to make GANs work better for real-world applications, the ability to invert a well-trained GAN model is very important. Varying image processing tasks have been used through GAN inversion, such as real image editing [52, 53, 54], image inpainting [8] and image super-resolution [8, 55]. GAN inversion is a process that aims to find the most accurate latent code, exploring the latent space, in order to recover the input image. Exploring and manipulating the latent space of GANs [56, 57, 58] offers the ability to edit images and a better understanding of deep generative models.

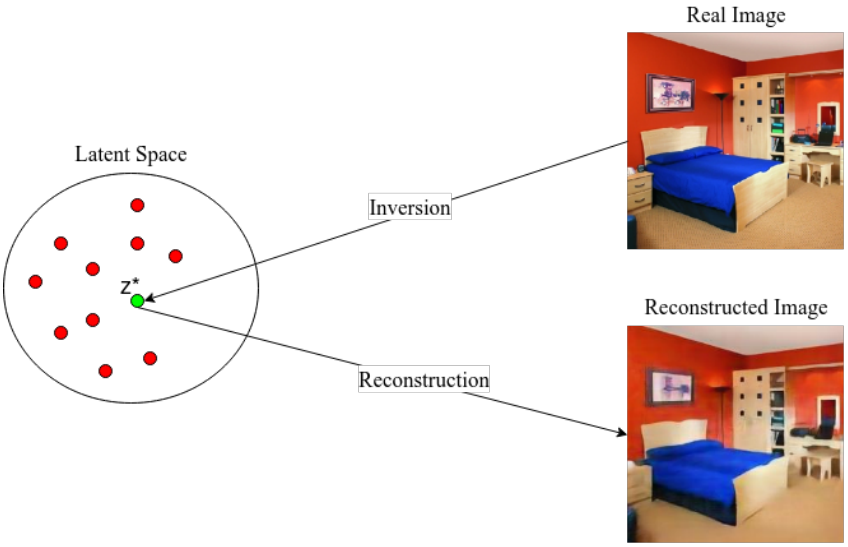


Figure 4.2: GAN Inversion

As shown in Figure 4.2, given an image to a well-trained generator G, inversion is the process of mapping a real image x back to a latent code in the latent space of GAN. The obtained latent code z^* is used to reconstruct the initial image by $x^* = G(z^*)$. Exploration and interpretation of the latent space of a GAN is needed to achieve various image editing tasks. Specifically, drifting the latent code z^* to different directions e.g. n_1 and n_2, then $z^* + n_1$ and $z^* + n_2$ corresponds to various attributes, like the shape of wardrobe and bed in 4.2 or a face expression in a face image. Recent GAN inversion implementations, like $mGANprior$ [8] achieves great results in both photo-realism and inversion accuracy, while preserving the ability of feature manipulation.

The process of GAN inversion can be formulated as:

$$z^* = \arg\min_z \mathcal{L}(G(z), x) \tag{4.2}$$

where x is the real image, z is the latent representation, $G(z)$ is the generative model and $\mathcal{L}(\cdot, \cdot)$ denotes a distance metric, like l_1, l_2 or perceptual metrics. The difficulty of this optimization problem lies in the non-convexity of $G(z)$ making it unable to reconstruct an image from a single latent code, this is the reason that $mGANprior$ [8] makes a proposal of optimizing multiple latent codes to reconstruct an image.

According to the recent survey in GAN inversion [16], there are three main GAN inversion techniques, which include optimization-based [8, 10, 59, 9], learning-based [52, 60, 61, 62] and hybrid GAN inversion [7]. Optimization-based GAN inversion methods have given the best results so far, while learning-based lag behind. Hybrid GAN inversion is a combination of optimization-based and learning-based approaches. In the case of optimization-based methods, the goal is to optimize the latent vector, while in the case of learning-based methods an encoder is trained to map an image back to the latent space.

Latent space refers to the hidden space of a generative model, which can be understood as a space of compressed data. The method of GAN inversion embeds a chosen latent space with images' most important features (such as edges or angles) in order to distinguish different objects. Similar features in latent space are located close to each other, which means sampling and optimizing different latent codes that are located close to each other in the latent space, gives visually similar images in image space. For example, even if people might have completely different writing styles, in a well-trained GAN, the same handwritten digits (or alphabet letters) will be bundled up closer in the latent space, while the other digits will be farther apart. Latent space representation helps to further analyze the data making them more simple whilst maintaining only the important features. The most simple and common space a GAN can choose for its latent representations, is the $\mathcal{Z}$ space. In the latent $\mathcal{Z}$ space, all the embedded latent representations ($z \in \mathcal{Z}$) are vectors containing random values usually initialized from a normal distribution. In the most common GAN architecture, known as DCGAN [49], $\mathcal{Z}$ space is a 100-dimensional uniform distribution.

4.2 Progressive growing of GANs

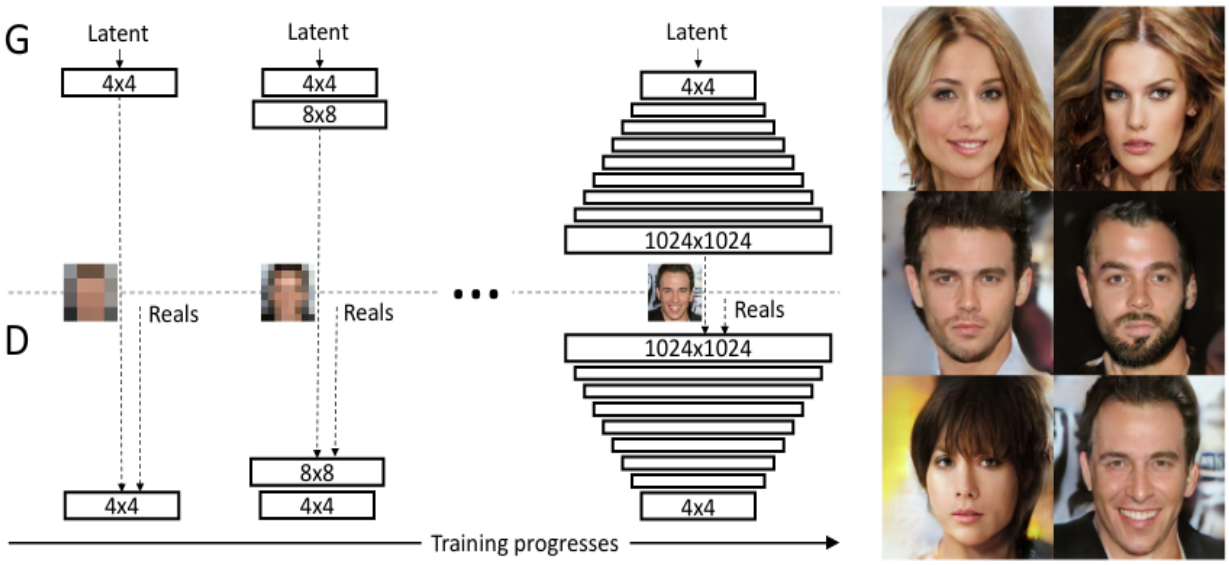


Figure 4.3: Progressive growing of GANs as proposed in [5].

Progressive growing of GANs introduced by Karras *et al.* [5]. At the time, GAN architectures were limited in generating low-resolution images. PGGAN biggest impact was the ability to produce extremely realistic images at a very high resolution, such as 1024×1024 resolution. With the purpose of doing this, a new training methodology approach was introduced, in which both the generator and the discriminator are growing progressively, starting from a low spatial resolution of 4×4, the resolution increases exponentially in powers of 2 as it is visualized in Fig. 4.3. While adding new higher-resolution layers to generator and discriminator, the previous layers also can be trained throughout the training process, which is significantly decreasing the training time. Along with the new GAN architecture, the training process became remarkably more stable, while variation also increased in order to avoid mode collapse. As in the previous GAN architectures, also in PGGAN both the generator and discriminator models are consisting of convolutional blocks of predefined number of filters with the size of 3×3 with LeakyReLU (with slope 0.2) as the activation function. For the purpose of resampling the training images, nearest neighbor and average pooling are used for upsampling and downsampling respectively. As the loss function of the model, WGAN-GP [21] loss function is used. The latent vector is fed in the first layer of the generator and it is a 512-dimensional vector of Gaussian distribution.

4.3 Multi-code GAN prior

Gu *et al.* [8] introduced multi-code GAN prior (mGANprior) and it belongs to optimization-based GAN inversion methods. It outperformed many existing GAN inversion methods, more specifically it was tested against: the optimization of a single latent code [63], learning an encoder [53] and using the encoder as initialization for optimization [64]. The challenge stands in utilizing N latent codes $\{z_n\}_{n=1}^{N}$, each of which is responsible for the reconstruction of a particular region in an image. The main difficulty of the proposed model lies in the integration process of the latent codes, provided that the image space $\mathcal{X}$ is a non-linear space and therefore there is a high chance that linear combinations will produce inconsequential images.

In order to overcome the issue of latent codes integration, the generator of a pre-trained GAN model is inverted from the image space to some intermediate feature space, instead of the more puzzling latent space. Hence, as shown in Fig. 4.4 the generation process is achieved by dividing the generator G into two sub-networks, the first one is responsible for generating the intermediate spatial features $\mathbf{F}_n^{(\ell)} = G_1^{(\ell)}(\mathbf{z}_n)$, while the second sub-network takes as input the composed spatial features of each latent code to produce the inverted output image. Concerning the distinguish of the important channels in the feature map $\mathbf{F}_n^{(\ell)}$, *adaptive channel importance* is introduced. More concretely, $\boldsymbol{\alpha}_n \in \Re^C$ is a C-dimensional vector, which initial values are $1/N$ and C represents the number of channels in the selected composition layer of $G(\cdot)$. Consequently, the inverted image proposed in [8] is formulated as:

$$\mathbf{x}^{inv} = G_2^{(\ell)}(\sum_{n=1}^{N} \mathbf{F}_n^{(\ell)} \odot \boldsymbol{\alpha}_n), \tag{4.3}$$

where $\odot$ denotes the channel-wise multiplication as

$$\{\mathbf{F}_n^{(\ell)} \odot \boldsymbol{\alpha}_n\}_{i,j,c} = \{\mathbf{F}_n^{(\ell)}\}_{i,j,c} \times \{\boldsymbol{\alpha}_n\}_c, \tag{4.4}$$

with i, j, c indicating the spatial location and c is the channel index.

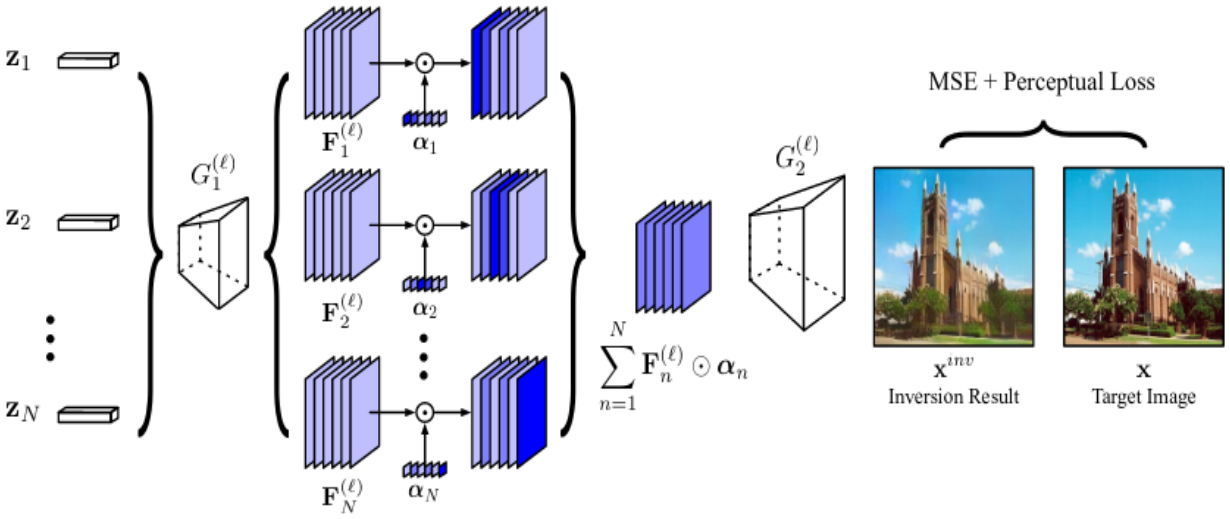


Figure 4.4: GAN inversion process as proposed in mGANprior [8].

In the sequel, the total parameters to optimize are each set of latent code $\{\mathbf{z}_n^*\}_{n=1}^N$ and importance vector $\{\boldsymbol{\alpha}_n^*\}_{n=1}^N$, an overall of $2N$ sets of parameters. Hence, the objective function in Eq. (4.2) is reformulated as:

$$\{\mathbf{z}_n^*\}_{n=1}^N, \{\boldsymbol{\alpha}_n^*\}_{n=1}^N = \underset{\{\mathbf{z}_n\}_{n=1}^N, \{\boldsymbol{\alpha}_n\}_{n=1}^N}{\arg\min} \mathcal{L}(\mathbf{x}^{inv}, \mathbf{x}). \tag{4.5}$$

The authors of *mGANprior* use VGG-16 model [65] as the backbone network and the output of layer conv_43 is used. The objective function $\mathcal{L}(\cdot, \cdot)$ is a combination of per-pixel loss function, like mean squared error (MSE), and perceptual loss function (VGG Loss) in order to make use of both low-level and high-level information respectively.

Therefore, the final form of the objective function is formulated as:

$$\mathcal{L}(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2 + \|\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)\|_1, \tag{4.6}$$

where $\phi(\cdot)$ denotes the perceptual feature extractor using the l_1 distance between the perceptual features extracted from the two images and the gradient descent algorithm for the optimization of the aforementioned sets of parameters.

After every iteration of inversion process (4.3), the reconstructed result is applied

as multi-code GAN prior to image inpainting task. The image inpainting task presuppose a known binary image, known as mask ($\mathbf{m}$), which is the pretreatment step where the known pixels are distinguished from the missing region. The mask can have different forms, such as text, scratches or whichever geometric shape. Thus, applying a mask to an intact image I_{ori} helps to create the data needed in order to reconstruct it with $mGANprior$. The reconstructed image x^{inv} is being post-processed in every iteration until the x^{inv} is close to a masked I_{ori}. The pre-trained GAN is responsible for filling in the missing pixels, since it possesses the ability to approximate the missing semantic features.

Hence, extending Eq. (4.6) the post-processing function as described in [8] can be formulated as:

$$\mathcal{L}_{inp} = \mathcal{L}(\mathrm{x}^{inv} \circ \mathbf{m}, I_{ori} \circ \mathbf{m}), \tag{4.7}$$

where $\circ$ denotes the element-wise product.

Chapter 5

Attention mechanisms in image inpainting

Attention technique is used for processing long-range dependencies in images, in contrast to convolution blocks which are processing a local neighborhood. Integrating an attention mechanism into a deep network allows it to learn the important information instead of focusing on non-useful features. Bahdanau *et al.* [13] first proposed the attention mechanism also known as additive attention in the field of natural machine translation followed by Luong *et al.* [66] introducing multiplicative attention, overcoming the problem of a sequence represented as a fixed-length vector in order to face long sequences. Thus, using an attention mechanism to a model gives the ability to remember the useful context of a longer sequence. In computer vision instead of word sequences, different types of data are being used, like images. Attention in an image used to focus on important features (like a nose or ears if it is a human), while the background is considered unimportant information. Xu *et al.* [67] showed the difference between attention mechanisms and divided them into two categories, $soft\ attention$, and $hard\ attention$.

$Soft\ attention$ techniques are the most common architectures and have been used

in many fields of computer vision, such as image segmentation, classification, object detection, etc. Implementations of $soft\ attention$ are spatial attention, channel attention, self-attention and mixed attention (where different types of attention are used, such as convolutional block attention module). $Hard\ attention$ and $soft\ attention$ mechanisms are different in the way they learn the alignment weights and look for the dependencies. In soft attention, the weights are updated by comparing particular regions to all other patches in the source image. These methods have achieved state-of-the-art results and the only important disadvantage is the cost, especially when the input images are large, where there are a high number of patches in the image. On the other hand, $hard\ attention$ mechanisms are significantly limited compared to $soft - attention$ mechanisms. One recent hard attention approach is $attention\ agent$ [68], which divides the input image into several blocks and uses a modified self-attention architecture, where only the relevant blocks are taken into account while the remaining blocks are ignored. Obviously, hard attention approaches are cost-efficient, come at the price of a non-differentiable and much more complex model.

The most common attention mechanism is the $self - attention$ technique introduced by Vaswani [18], proposing the $transformer$ and it was a breakthrough in the field of Natural Language Processing encountering long-range dependencies with excellence. $Self - attention$, also known as $intra - attention$, because of its ability to calculate the dependencies within the input elements, instead of general attention that is using input and output elements. $Self-attention$ became the most popular attention mechanism in the technology of GANs after Zhang *et al*. [14] introduced the self attention generative adversarial networks (SAGAN). There are already many attention-based implementations in computer vision [14, 4, 69, 70] covering various image processing tasks.

The main idea of spatial and channel-wise attentions in image captioning came from Chen *et al*. [71], demonstrating the usefulness of multi-layered attention (spatial and channel-wise attention) in a CNN. Convolutional block attention module (CBAM) [17] (introduced in 2018) concept is based on [71], is a lightweight and general module that can be incorporated in any CNN implementation. CBAM became widely known not only for its flexibility and low-cost operations, but mainly for its great results in

various fields of computer vision, like object detection and image classification. CBAM belongs to the mixed attention category as mentioned before, combining both spatial and channel attention, which indicates 'what' and 'where' to look for. Although CBAM has been published recently, similar to the aforementioned attention mechanisms, there is a decent amount of research in the literature. More specifically, Hu *et al.* [72] have proposed a model that makes use of channel attention with the difference that channel-wise attention is computed with global average-pooled features instead of average and max-pooled features used in CBAM. Furthermore, no spatial attention is used. More close to CBAM, Park *et al.* [73] exploits both channel and spatial-wise attention, placing the module at every bottleneck of the network instead of every convolutional block in CBAM.

5.1 Self-attention module

The mechanism of $self - attention$ consists of $queries$, $keys$ and $values$, which are represented as vectors. The challenge stands in mapping a $query$ against a set of $keys$ (of the same dimensionality as $query$), computing their dot products (multiplicative attention), followed by a softmax function to obtain the weights on the values. As previously mentioned, the $self - attention$ module makes use of input elements in order to produce their attention, therefore the input data, which are the feature maps in computer vision, are divided in order to shape the vectors of $queries$, $keys$ and $values$. These vectors are continuously updated during the training or optimization process.

For every $query$ vector that represents the deep features of an image, there are the key vectors which represent all the other data- except the query. The calculation of the score is taken by multiplying the $query$ vector with all key vectors, in order to get the dot-product of the matrices. The next step is to normalize the scores using the softmax activation function for the particular $query$ vector and then the normalized scores are multiplied by the $values$ vector as shown in the figure below 5.1.

There have been significant modifications concerning the calculation of alignment score functions the attention module uses. Luong *et al.* [66] used dot-product score

function to calculate the attention map and also introduced general attention, where among dot-products there is a trainable weight matrix in the attention layer. Graves *et al*. [74] content-based attention, which calculates the corresponding attention map using the cosine similarity instead of dot-product. In the most attention architectures, dot-product alignment score function and $scaled\ dot-product$ have prevailed. $Scaled\ dot-product$ attention introduced by Vaswani *et al*. [18] and a slightly modified scoring function is used. Specifically, it is almost the same as $dot-product\ attention$, except for a scaling factor, that is the square root of dimension of the keys vector as shown in 5.1.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{5.1}$$

The scaling factor $\sqrt{d_k}$ is suitable at times when the dimension of keys vector is large. That leads to large dot-products and regions with very small gradients for softmax function causing ineffective training. Thus, scaling the dot-products makes the training process more effective and more computationally efficient than the other scoring functions, given the fact that the dimension of keys and queries are vectors of the same lengths.

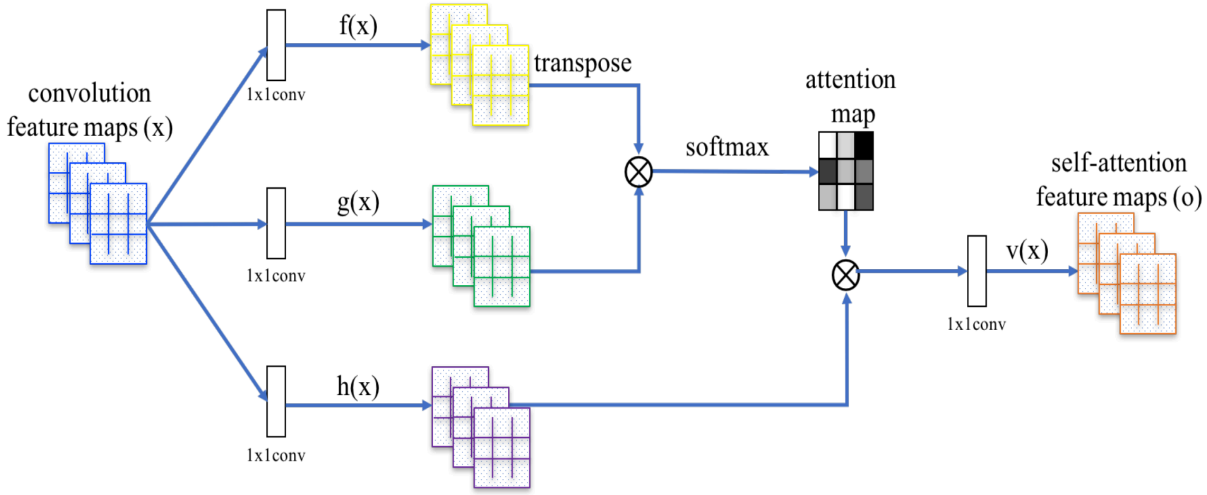


Figure 5.1: Self-attention module as proposed in SAGAN [14], $f(x)$, $g(x)$ and $h(x)$ represent the key, $query$ and $value$ vectors respectively. The $\otimes$ denotes matrix multiplication. The softmax operation is performed on each row.

5.2 Convolutional block attention module (CBAM)

CBAM consists of two sequential sub-modules, *channel attention module* (CAM) and *spatial attention module* (SAM). These two modules are applied at every convolutional block in the network in order to adaptively refine the intermediate feature map and according to the authors [17] the optimal order is to apply CAM followed by SAM.

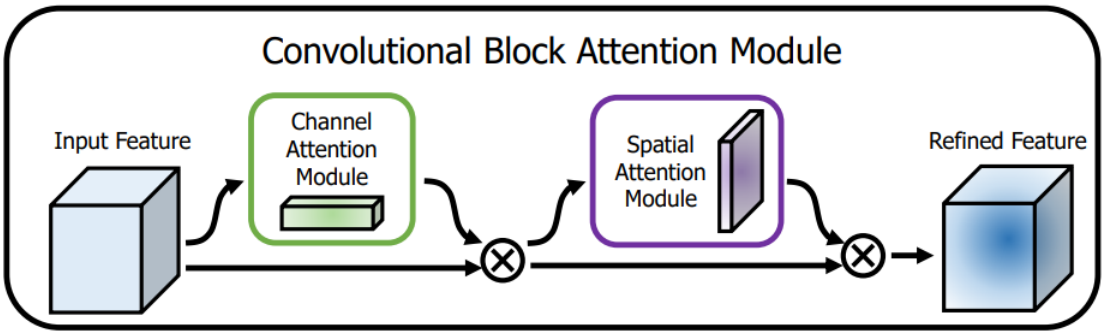


Figure 5.2: Convolutional block attention module (CBAM) as introduced in [17].

As demonstrated in Fig. 5.2 the input feature is a feature map $\mathbf{F} \in \Re^{C \times H \times W}$ from which *channel attention module* generates a 1D channel attention map $\mathbf{M_c} \in \Re^{C \times 1 \times 1}$, while *spatial attention module* produces a 2D spatial attention map $\mathbf{M_s} \in \Re^{1 \times H \times W}$. In [17] that process is summarized as:

$$\begin{aligned} \mathbf{F}' &= \mathbf{M_c}(\mathbf{F}) \otimes \mathbf{F}, \\ \mathbf{F}'' &= \mathbf{M_s}(\mathbf{F}') \otimes \mathbf{F}', \end{aligned} \tag{5.2}$$

where $\otimes$ denotes element-wise multiplication. During multiplication, the attention values are broadcasted (copied) accordingly: channel attention values are broadcasted along the spatial dimension, and vice versa. $\mathbf{F}''$ is the final refined output.

5.2.1 Spatial attention module

Spatial attention looks for the 'interesting' regions in the image, like the objects instead of useless background regions. *Spatial attention* mechanism in order to focus on 'where' are the semantic regions in the image, attempts to generate an attention

mask on a given feature map, enhancing particular regions (features) with the intention of emphasizing on desirable objects or regions.

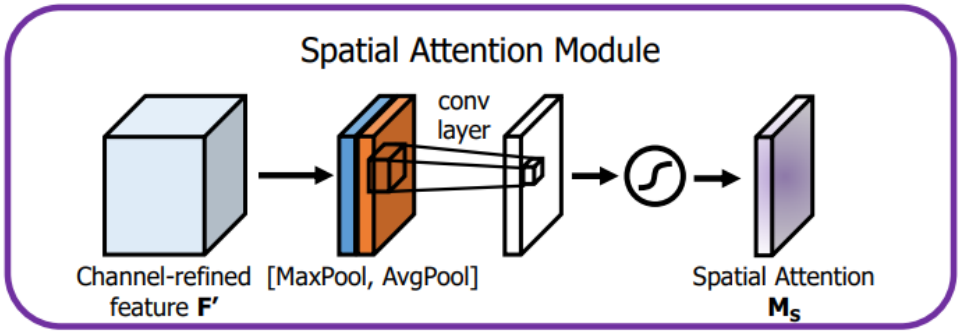


Figure 5.3: Spatial attention module (SAM) as introduced in [17].

As reflected in Fig. 5.3, *spatial attention module* (SAM) consists of a three-stage sequential procedure. Initially, the *channel pool* section takes part, in which from an input feature map $\mathbf{F} \in \Re^{C \times H \times W}$ two feature maps are generated, $\mathbf{F}^{\mathbf{s}}_{\mathbf{avg}} \in \Re^{1 \times H \times W}$ from average-pooling operation and $\mathbf{F}^{\mathbf{s}}_{\mathbf{max}} \in \Re^{1 \times H \times W}$ from max-pooling operation. Next part is a simple convolutional block with a large 7×7 kernel size in order to increase the receptive field, that takes as input the concatenated feature maps from *channel pool* part. The output of the convolution layer is normalized and scaled using batch normalization as the default option, although the ReLU activation function can be used too. In the last part, the output of the previous convolution block is passed to a sigmoid activation layer, mapping all the values in the range of 0 to 1.

The final output is a 2D spatial attention map $\mathbf{M_s} \in \Re^{1 \times H \times W}$ and according to [17] is formulated as:

$$\begin{aligned} \mathbf{M_s}(\mathbf{F}) &= \sigma(f^{7\times7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])) \\ &= \sigma(f^{7\times7}([\mathbf{F}^{\mathbf{s}}_{\mathbf{avg}}; \mathbf{F}^{\mathbf{s}}_{\mathbf{max}}])), \end{aligned} \tag{5.3}$$

where σ denotes the sigmoid function and $f^{7\times7}$ represents a convolution operation with the filter size of 7×7.

5.2.2 Channel attention module

In contrast to *spatial attention*, *channel attention* generates a channel attention map based on 'what' is useful among multiple channels, highlighting meaningful features.

As mentioned before, the *channel attention module* in CBAM is very similar to the 'squeeze-excite' (SE) layer proposed by Hu *et al.* [72].

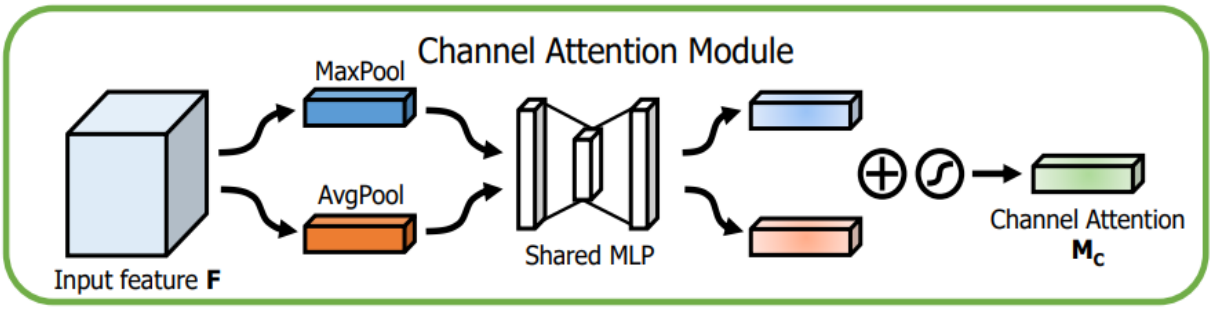


Figure 5.4: Channel attention module (CAM) as introduced in [17].

As illustrated in Fig. 5.4, the first step in *channel attention* computation is decomposing the input feature $\mathbf{F} \in \Re^{C \times H \times W}$ into two spatial context subsequent vectors, $\mathbf{F^c_{avg}} \in \Re^{C \times 1 \times 1}$ stands for *global average pooling* as used also in [72] and the difference is made from $\mathbf{F^c_{max}} \in \Re^{C \times 1 \times 1}$ which stands for *global max pooling*. According to Woo *et al.* [17] bringing in *max pooling* operation improves the network, as opposed to single *average pooling* operation. Each pooling operation has its own merits: *average pooling* is used for spatial information aggregation resulting in a smoothing output, while *max pooling* is capable of producing sharper contextual information, such as the edges of an object. Subsequently, the two vectors produced from *max pooling* and *average pooling* operations are the inputs in the shared multi-layer perceptron (MLP) network, which is applied to each vector, consists of one hidden layer and its number of neurons is compromised of a reduction ratio variable r that is set to 16 in the official implementation of CBAM. The activation function of MLP is ReLU as implemented by the authors of CBAM and the shared MLP denotes that both the input vectors update the same weights. The two output vectors of the shared MLP as it is also shown in Fig. 5.4 are being summed up element-wise and finally are forwarded to a sigmoid activation function for the purpose of computing the final channel attention map $\mathbf{M_c} \in \Re^{C \times 1 \times 1}$ that contains the channel weights (scores). The complete channel attention formula as introduced in [17]:

$$\begin{aligned} \mathbf{M_c}(\mathbf{F}) &= \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F}))) \\ &= \sigma(\mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{avg}})) + \mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{max}}))), \end{aligned} \tag{5.4}$$

where σ denotes the sigmoid function and the weights $\mathbf{W_0} \in \Re^{C/r \times C}$, $\mathbf{W_1} \in \Re^{C \times C/r}$.

5.3 Proposed method

For the purpose of improving the output image quality of $mGANprior$ model, an extra attention mechanism is added to an intermediate layer of the pre-trained GAN model.

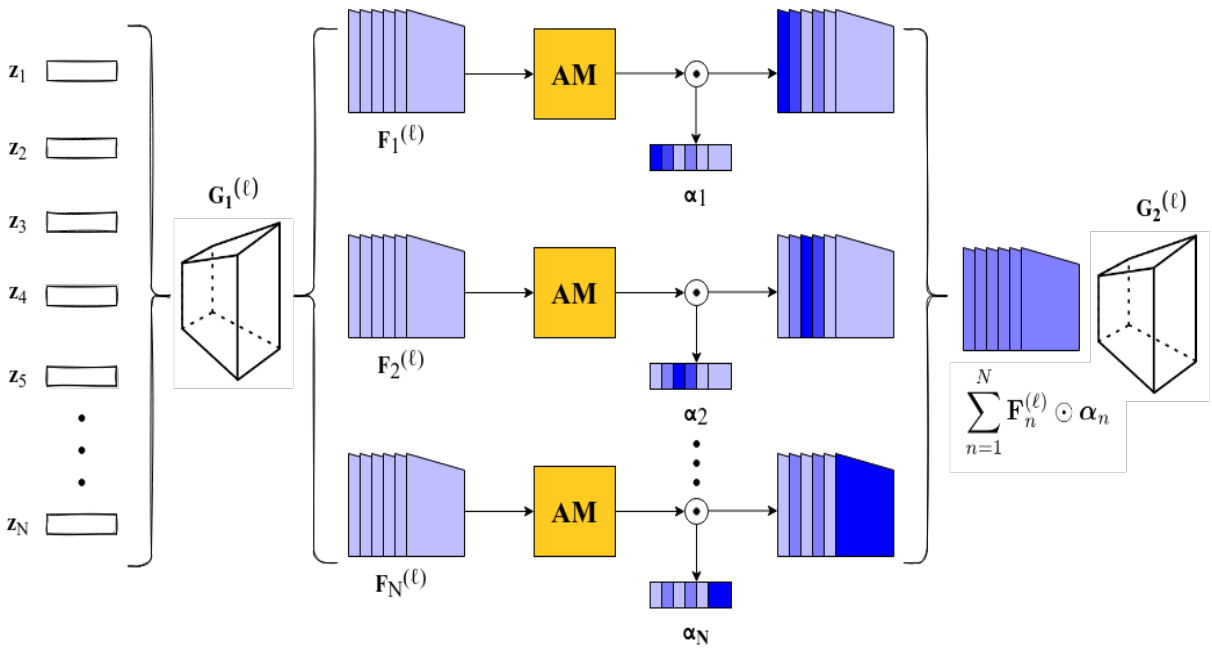


Figure 5.5: GAN inversion process with the extra attention module layer.

More concretely, different attention mechanisms are applied to original $mGANprior$, while the optimization objective (4.7) remains the same. Attention mechanisms are able to enhance feature details in some intermediate feature space in every iteration of the inversion process. The aforementioned attention mechanisms, self-attention and convolutional block attention mechanism (CBAM) as extensively described in Sec. 5.1, 5.2, are able to process the intermediate feature space, both channel and spatial-wise. Channel-wise attention before the feature composition layer has the ability to distinguish the more important channels of a high-dimension layer in order to further refine them, while spatial-wise attention focuses on the important regions within a feature map. This modification aims to capture the long-range dependencies at the same time with the inpainting process, with the intention of helping the pre-trained GAN to extract the missing features and fill in the appropriate missing pixels.

Chapter 6

Experiments

For the evaluation of the proposed multi-code GAN prior with attention layer 5.3 against the original $mGANprior$, extensive experiments conducted on state-of-the-art progressive growing GAN (PGGAN) [5]. LSUN [75] datasets are selected, because they are commonly used in the GAN inversion methods. Also, PGGAN training is a challenging and long standing task, which prerequisites high-tech hardware and there is always the risk of modal collapse or non-convergence as described in Ch. 4. Therefore, for reliability and time-saving reasons, the weights of the pre-trained PGGAN models are obtained from the official repository[1] for each dataset. More accurately, PGGAN models trained on three common datasets:

1. LSUN-bedroom, containing over 3 million images of 256×256 resolution.
2. LSUN-dog, containing over 5 million images of 256×256 resolution.
3. LSUN-cat, containing over 1.5 million images of 256×256 resolution.

For quantitative evaluation of the inversion results for the image inpainting task, two different methods were used. Peak signal-to-noise ratio (PSNR) measures the similarity between the original input and the reconstruction result from pixel level. PSNR is an objective method suitable for measuring image quality of reconstructed images.

[1] https://drive.google.com/drive/folders/15hvzxt_XxuokSmj0uO4xxMTMWVc0cIMU

It computes the mean-square error (MSE) in its core and measures the intensity of noise in decibel units (dB). Given the original image and the inversion result x^{inv}, higher PSNR values signify better reconstruction results as well as smaller MSE values. PSNR is based on MSE, which is poorly correlated with human perception of the visual system. To alleviate this limitation, the second method used for the inversion evaluation is structural similarity index measure (SSIM) [76]. SSIM is a perceptual metric that quantifies the image quality degradation caused by the inversion process. Similar to PSNR, SSIM is a full reference metric that requires the original image and the processed image. The comparison between the two images is performed on the basis of three components, these are luminance, contrast and structure. Similar to PSNR the higher values of SSIM indicate a better similarity between the original image and the processed image.

Examining image inpainting task with three pre-trained PGGAN models as prior, a total of 300 unique real images were inverted, that is 100 images for each PGGAN model. For each dataset, three different masks were used, which consisted of missing blocks of pixels in random regions in the input image. More precisely, the three masks that are applied, conceal 12%, 25% and 40% of pixels in the input image.

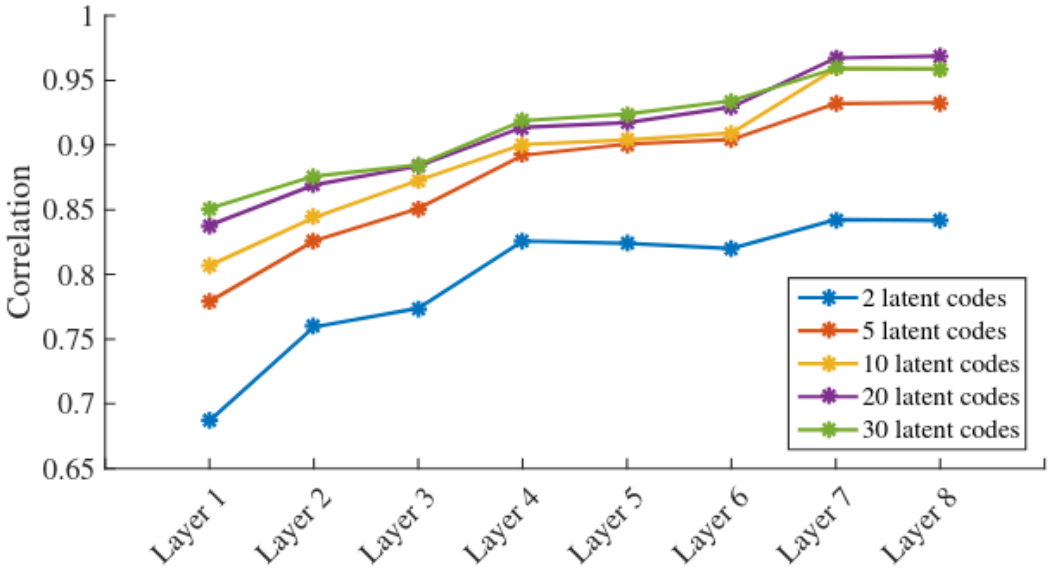


Figure 6.1: Effects on inversion performance by the number of latent codes used and the feature composition position as proposed in [8].

Based on the authors of $mGANprior$ [8] the optimal number of latent codes are 20, since there is no significant improvement observed by increasing the number of latent codes, this also can be noticed in Fig. 6.1 with tested results.

Figure 6.2: Qualitative comparison on LSUN-bedroom dataset. It is obvious, that inversion using 20 latent codes preserves more details. For inversion it is used the $mGANprior$ 4.3 model with different number of latent codes, Adam [77] optimization method with 3000 iterations and Gaussian initialization for importance and latent codes vectors. Feature composition layer is set to 8 and randomly cropped 12% of pixels.

Hence, using 20 latent codes not only results in a better inversion quality, but also is more cost-efficient, since the total number of parameters to optimize is $N \times (C + C)$, where N is the number of latent codes and there are two C, each of which represents the number of channels for the latent codes vector and the importance vector. Different feature composition layers have different number of channels and that is entirely

dependent on the generative model.

Apart from the number of latent codes, yet another crucial point for inversion quality is the selection of feature composition layer. The authors of $mGANprior$ [8] suggest that a higher composition layer could lead to a better reconstruction, however the lower layers of PGGAN contain high-level semantics and are suitable for enabling the re-usability of the semantic knowledge learned by PGGAN. Bau $et\ al.$ [78] compared different layers of a progressive GAN, showing that semantic objects appearing after the layer 4, while later layers are dominated by low-level edges and colors. In a more recent research [79], it is extensively examined how different layers correspond to variation factors from different abstraction levels. More concretely, in Fig. 6.3 it is demonstrated that the layers of the generator in a GAN are specialized to compose semantics in a hierarchical manner: the bottom layers determine the layout, the lower layers and upper layers control category-level and attribute-level variations respectively, while color scheme is mostly rendered at the top.

	Bottom	Lower	Upper	Top
Layout	95%	5%	0%	0%
Objects	10%	90%	0%	0%
Attributes	0%	5%	85%	5%
Color Scheme	0%	0%	25%	75%

Figure 6.3: How different layers affect different types of semantics, based on a user study in [79].

Nevertheless, due to the fact that the generation process in a GAN remains largely unresolved and that the previous research [79] is mainly based on scene images, it is urgent to conduct qualitative experiments as well, concerning the selection of feature composition layers. In the PGGAN model that is trained to produce 256×256 images, there are 15 convolutional layers. The authors of $mGANprior$ [8] suggest layer 4 as the optimal layer to perform feature composition, due to high-level semantics captured in lower layers.

For a **qualitative** evaluation visual comparisons are provided in Fig. 6.4, 6.5, 6.6 for LSUN-church_outdoor, LSUN-dog and LSUN-bedroom respectively. All the results are coming from the trained PGGAN models without using any post-processing. For inversion it is used the $mGANprior$ 4.3 model with 20 latent codes, Adam [77] optimization method with 3000 iterations and Gaussian initialization for importance and latent codes vectors. Also, 12% of pixels are randomly cropped.

Figure 6.4: Qualitative comparison on LSUN-bedroom dataset.

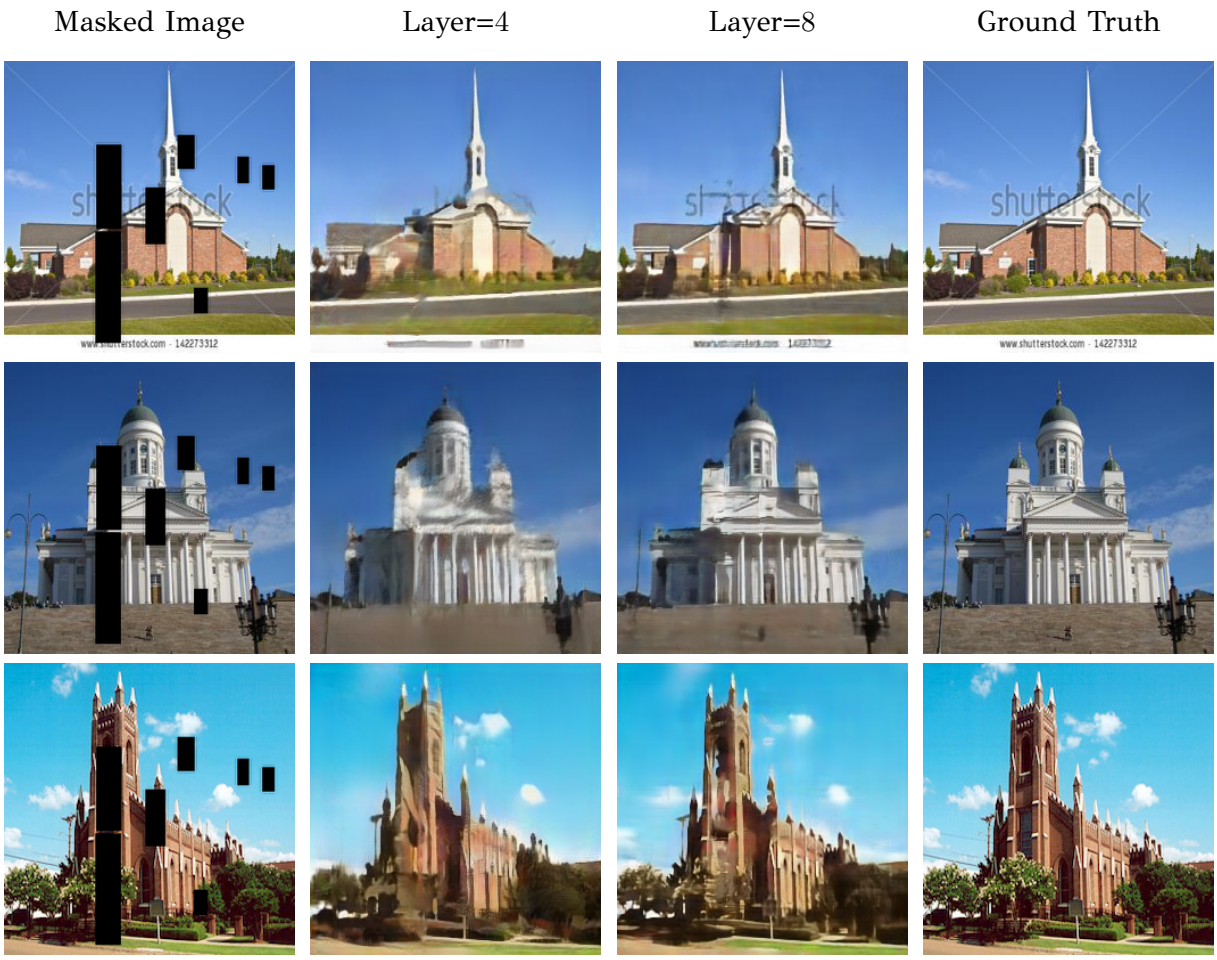


Figure 6.5: Qualitative comparison on LSUN-church_outdoor dataset.

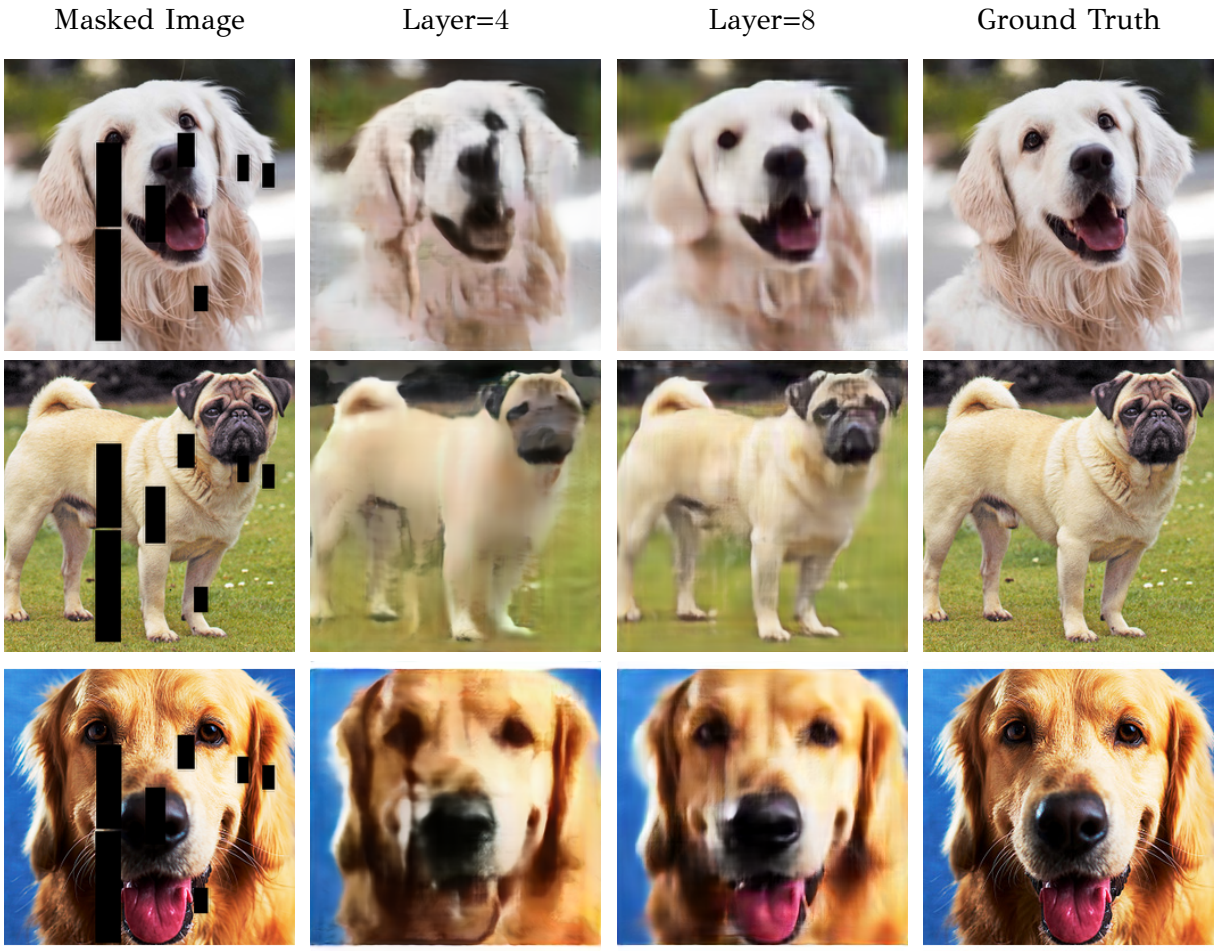


Figure 6.6: Qualitative comparison on LSUN-dog dataset.

In all datasets, selecting layer 8 as the feature composition layer, performing much better in terms of texture and structure. It is deliberate, that a relatively small mask has been chosen for the inversion process, in order to observe which is the optimal layer to perform the composition, independently of the image inpainting task. It is evident even with visual inception, that composing at layer 8 results in a better quality output image. Apart from color that cannot be captured at a lower layer, the structures are much more distinct and details like angles are more observable. However, feature composition at layer 8 is more costly since the resolution of generated feature maps is 32×32, while at layer 4 the resolution of the corresponding feature maps is 8×8 channels.

Besides the above, composition layer 8 is much preferable also in terms of attention mechanisms 5. This is because, attention mechanisms tend to deliver better results when applied to layers with larger feature maps. This can be understood, as the larger the feature maps are the better capture of long-range dependencies can be achieved. Also, attention mechanisms implemented on higher dimension space reduce similarity between the produced attention maps.

For **quantitative** evaluation, PSNR and SSIM metrics are introduced, which are widely deployed in the field of image inpainting as already mentioned. Three datasets are chosen as before, LSUN-church_outdoor, LSUN-dog and LSUN-bedroom. For each dataset 100 real images are inverted. The different methods that are tested, are:

1. original 4.3 $mGANprior$.
2. 5.3 with $Self - Attention$ mechanism (dot-product scoring function).
3. 5.3 with $Self - Attention$ mechanism (scaled dot-product scoring function).
4. 5.3 with $Convolutional\ Block\ Attention\ Module$.

All the results are coming from the trained PGGAN models without using any post-processing. The above models are used with 20 latent codes, composition layer 8, Adam [77] optimization method with 3000 iterations and Gaussian initialization for importance and latent codes vectors.

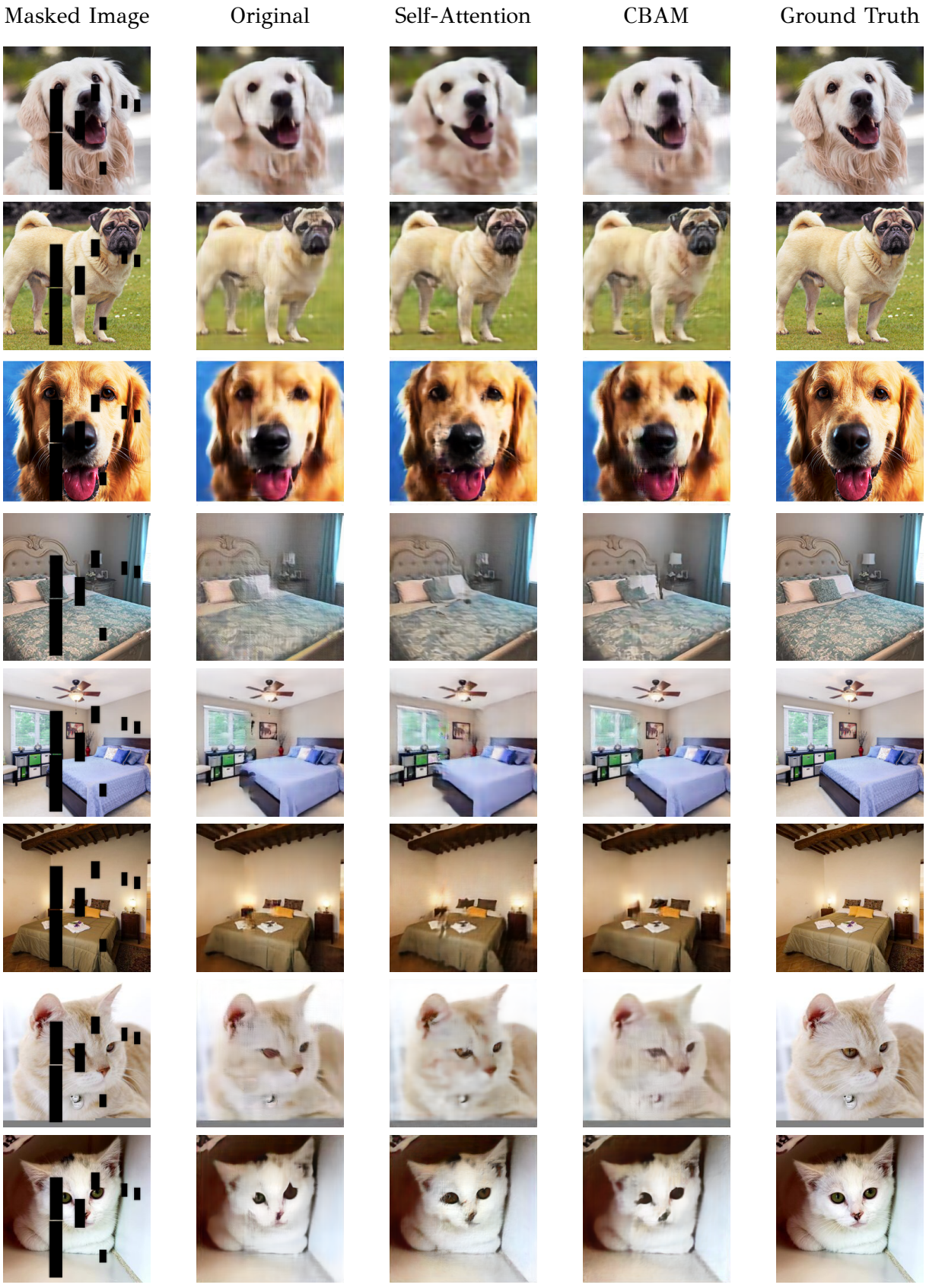


Figure 6.7: Quantitative comparison among the three datasets. Randomly cropping 12% pixels.

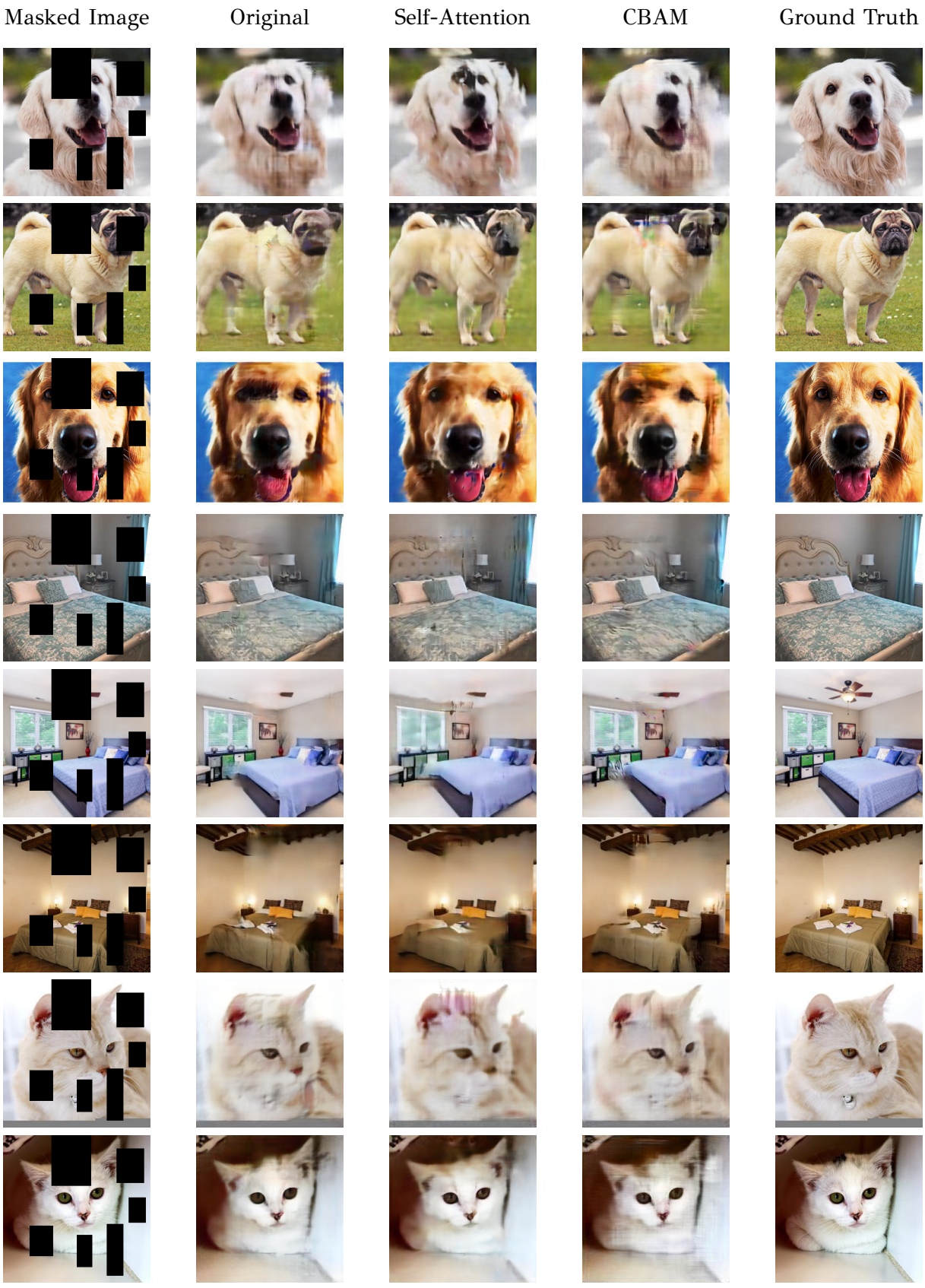


Figure 6.8: Quantitative comparison among the three datasets. Randomly cropping 25% pixels.

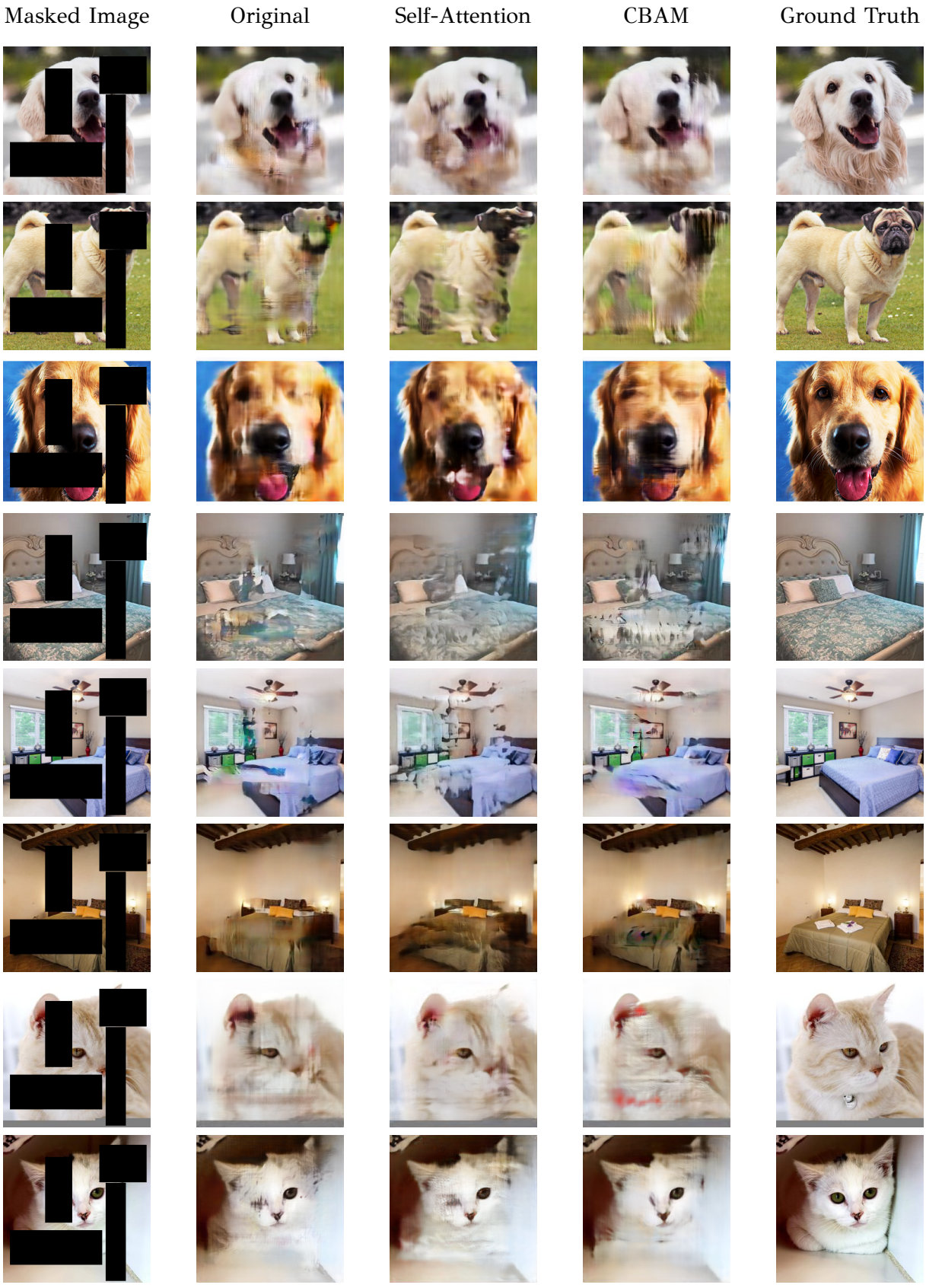


Figure 6.9: Quantitative comparison among the three datasets. Randomly cropping 40% pixels.

Tab. 6.1, 6.2, 6.3 and 6.4 show the quantitative comparisons using different inpainting methods.

Table 6.1: Randomly cropping 12% pixels as shown in Fig. 6.7. ↑ means higher score is better.

	LSUN-bedroom		LSUN-dog		LSUN-cat	
Method	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
original 4.3	**25.54**	**0.83**	**25.02**	**0.75**	**25.43**	**0.75**
Self-Attention	25.22	0.80	24.91	0.74	25.35	0.75
CBAM	25.46	0.82	24.97	0.74	25.22	0.74

Table 6.2: Randomly cropping 25% pixels as shown in Fig. 6.8. ↑ means higher score is better.

	LSUN-bedroom		LSUN-dog		LSUN-cat	
Method	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
original 4.3	**22.08**	**0.77**	**21.40**	**0.69**	22.44	**0.70**
Self-Attention	21.76	0.73	21.33	**0.69**	**22.49**	**0.70**
CBAM	21.93	0.76	21.34	**0.69**	22.37	0.69

Table 6.3: Randomly cropping 40% pixels as shown in Fig. 6.9. ↑ means higher score is better.

	LSUN-bedroom		LSUN-dog		LSUN-cat	
Method	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
original 4.3	**18.16**	**0.68**	18.70	**0.62**	19.63	**0.63**
Self-Attention	18.06	0.63	**18.75**	0.61	**19.68**	**0.63**
CBAM	17.84	0.66	18.73	**0.62**	19.49	**0.63**

Table 6.4: Scaled-dot product scoring function tested on LSUN-bedroom. ↑ means higher score is better.

LSUN-bedroom	Mask 1		Mask 2		Mask 3	
Self-Attention	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Dot-product	**25.22**	**0.80**	**21.76**	**0.73**	**18.06**	**0.63**
Scaled dot-product	24.16	0.75	18.53	0.64	16.23	0.60

Quantitative results show that all modified $mGANprior$ models are close to each other, although the original $mGANprior$ 4.3 is slightly more efficient in most cases. As it is shown in Tab 6.1, the inversion process is slightly ahead in every dataset. With only 12% of the total pixels missing, the modified inversion methods seem unable to improve the original implementation, however SSIM values are almost the same. Cropping the one-fourth of thhe pixels in the images, original version of $mGANprior$ is still ahead except from LSUN-cat dataset, where the $self-attention$ modified implementation is slightly ahead, while SSIM values are almost equal, as the previous mask. When almost half of the image is cropped (40% of pixels), the results are mixed. More precisely, $self-attention$ is ahead in LSUN-dog and LSUN-cat datasets by the smallest of margins, while in LSUN-bedroom $self-attention$ lags behind. $CBAM$ modification even if it is never ahead in terms of PSNR values, it is very close to the other two methods in SSIM values. Finally, integrating $self-attention$ with scaled dot-product scoring function is way behind compared to all other methods in terms of both PSNR and SSIM values. In overall, the $self-attention$ and $CBAM$ mechanisms seem to not greatly affect the outcome, in a positive or negative way.

All experiments for the original $mGANprior$ and all of its modifications were conducted in a single NVIDIA Titan XP GPU with TensorFlow-GPU 1.14, with CUDA 9.2 version, cuDNN 7.6.5 and Torch 1.8. In total, there were over one thousand inversions made, that lasted around 55 days.

Chapter 7

Conclusion

In this thesis, the inversion process of a pre-trained GAN was examined and further modified. At first, image inpainting domain was looked into in detail, analyzing different existing approaches. In the sequel, a closer look at the upcoming technology of generative adversarial networks was provided and more specifically the progressive growing GAN (PGGAN) extension. It is clarified, how the generation process can be inversed, exploring and optimizing the GAN's latent space, in order to use the model as a prior to image inpainting task. For the purpose of improving the existing model, different attention mechanism methods were applied. More concretely, both $self-attention$ and $convolutional\ block\ attention$ modules were tested, which have brought in results similar to the original implementation. The attempt of improving the existing method stood in modifying the existing model, incorporating an attention layer into it. Experiments were conducted on three different datasets, and two different evaluation metrics were used. It is demonstrated, that the applied attention modules are not giving better results all the time. This proves, that the main factor of the inversed image quality depends mainly on the ability of the trained GAN to extract the semantics of the missing regions in the image. A second factor stands in the optimization process, in which randomness plays a significant role. In the future, when the concept of how GANs synthesize images will be more understandable, the inpainting results through the inversion process will be inevitable much better.

Bibliography

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[2] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato, "Fader networks: Manipulating images by sliding attributes," *arXiv preprint arXiv:1706.00409*, 2017.

[3] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. 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*, 2017, pp. 4681–4690.

[4] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5505–5514.

[5] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[6] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[7] J. Zhu, Y. Shen, D. Zhao, and B. Zhou, "In-domain gan inversion for real image editing," in *European conference on computer vision*. Springer, 2020, pp. 592–608.

[8] J. Gu, Y. Shen, and B. Zhou, "Image processing using multi-code gan prior," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3012–3021.

[9] R. Abdal, Y. Qin, and P. Wonka, "Image2stylegan++: How to edit the embedded images?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8296–8305.

[10] Z. C. Lipton and S. Tripathi, "Precise recovery of latent vectors from generative adversarial networks," *arXiv preprint arXiv:1702.04782*, 2017.

[11] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order boltzmann machine," *Advances in neural information processing systems*, vol. 23, pp. 1243–1251, 2010.

[12] V. Mnih, N. Heess, A. Graves *et al.*, "Recurrent models of visual attention," in *Advances in neural information processing systems*, 2014, pp. 2204–2212.

[13] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[14] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.

[15] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[16] W. Xia, Y. Zhang, Y. Yang, J.-H. Xue, B. Zhou, and M.-H. Yang, "Gan inversion: A survey," *arXiv preprint arXiv:2101.05278*, 2021.

[17] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[19] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9446–9454.

[20] P. Upchurch, J. Gardner, G. Pleiss, R. Pless, N. Snavely, K. Bala, and K. Weinberger, "Deep feature interpolation for image content changes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7064–7073.

[21] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[22] H. Liu, B. Jiang, Y. Xiao, and C. Yang, "Coherent semantic attention for image inpainting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4170–4179.

[23] J. Liu and C. Jung, "Facial image inpainting using attention-based multi-level generative network," *Neurocomputing*, vol. 437, pp. 95–106, 2021.

[24] S. Uddin and Y. J. Jung, "Global and local attention-based free-form image inpainting," *Sensors*, vol. 20, no. 11, p. 3204, 2020.

[25] M. Richard and M. Y.-S. Chang, "Fast digital image inpainting," in *Appeared in the Proceedings of the International Conference on Visualization, Imaging and Image Processing (VIIP 2001), Marbella, Spain*, 2001, pp. 106–107.

[26] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on image processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[27] J. Wang, K. Lu, D. Pan, N. He, and B.-k. Bao, "Robust object removal with an exemplar-based image inpainting approach," *Neurocomputing*, vol. 123, pp. 150–155, 2014.

[28] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 417–424.

[29] H. Li, W. Luo, and J. Huang, "Localization of diffusion-based inpainting in digital images," *IEEE transactions on information forensics and security*, vol. 12, no. 12, pp. 3050–3064, 2017.

[30] K. Li, Y. Wei, Z. Yang, and W. Wei, "Image inpainting algorithm based on tv model and evolutionary algorithm," *Soft Computing*, vol. 20, no. 3, pp. 885–893, 2016.

[31] T. Ružić, A. Pižurica, and W. Philips, "Markov random field based image inpainting with context-aware label selection," in *2012 19th IEEE International Conference on Image Processing*, 2012, pp. 1733–1736.

[32] N. Kawai, T. Sato, and N. Yokoya, "Diminished reality based on image inpainting considering background geometry," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 3, pp. 1236–1247, 2015.

[33] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, and Y. Akbari, "Image inpainting: A review," *Neural Processing Letters*, vol. 51, no. 2, pp. 2007–2028, 2020.

[34] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan, "Shift-net: Image inpainting via deep feature rearrangement," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 1–17.

[35] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[36] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[37] X. Zhu, Y. Qian, X. Zhao, B. Sun, and Y. Sun, "A deep learning approach to patch-based image inpainting forensics," *Signal Processing: Image Communication*, vol. 67, pp. 90–99, 2018.

[38] Y.-L. Chang, Z. Yu Liu, and W. Hsu, "Vornet: Spatio-temporally consistent video inpainting for object removal," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[39] T. Nakamura, A. Zhu, K. Yanai, and S. Uchida, "Scene text eraser," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 832–837.

[40] Y. Chen and H. Hu, "An improved method for semantic image inpainting with gans: progressive inpainting," *Neural Processing Letters*, vol. 49, no. 3, pp. 1355–1367, 2019.

[41] Y.-G. Shin, M.-C. Sagong, Y.-J. Yeo, S.-W. Kim, and S.-J. Ko, "Pepsi++: Fast and lightweight network for image inpainting," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 252–265, 2020.

[42] H. Wang, L. Jiao, H. Wu, and R. Bie, "New inpainting algorithm based on simplified context encoders and multi-scale adversarial network," *Procedia computer science*, vol. 147, pp. 254–263, 2019.

[43] P. Vitoria, J. Sintes, and C. Ballester, "Semantic image inpainting through improved wasserstein generative adversarial networks," *arXiv preprint arXiv:1812.01071*, 2018.

[44] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 214–223. [Online]. Available: http://proceedings.mlr.press/v70/arjovsky17a.html

[45] J. Dong, R. Yin, X. Sun, Q. Li, Y. Yang, and X. Qin, "Inpainting of remote sensing sst images with deep convolutional generative adversarial network," *IEEE geoscience and remote sensing letters*, vol. 16, no. 2, pp. 173–177, 2018.

[46] X. Han, Z. Wu, W. Huang, M. R. Scott, and L. S. Davis, "Compatible and diverse fashion image inpainting," *arXiv preprint arXiv:1902.01096*, 2019.

[47] H. Liu, G. Lu, X. Bi, J. Yan, and W. Wang, "Image inpainting based on generative adversarial networks," in *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE, 2018, pp. 373–378.

[48] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[49] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[50] J. Nistal, C. Aouameur, S. Lattner, and G. Richard, "Vqcpc-gan: Variable-length adversarial audio synthesis using vector-quantized contrastive predictive coding," *arXiv preprint arXiv:2105.01531*, 2021.

[51] Y. Zhang, Z. Gan, and L. Carin, "Generating text via adversarial training," in *NIPS workshop on Adversarial Training*, vol. 21. academia. edu, 2016, pp. 21–32.

[52] G. Perarnau, J. Van De Weijer, B. Raducanu, and J. M. Álvarez, "Invertible conditional gans for image editing," *arXiv preprint arXiv:1611.06355*, 2016.

[53] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *European conference on computer vision*. Springer, 2016, pp. 597–613.

[54] D. Bau, H. Strobelt, W. Peebles, J. Wulff, B. Zhou, J.-Y. Zhu, and A. Torralba, "Semantic photo manipulation with a generative image prior," *arXiv preprint arXiv:2005.07727*, 2020.

[55] K. C. Chan, X. Wang, X. Xu, J. Gu, and C. C. Loy, "Glean: Generative latent bank for large-factor image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 245–14 254.

[56] Y. Shen, J. Gu, X. Tang, and B. Zhou, "Interpreting the latent space of gans for semantic face editing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[57] H. Yang, L. Chai, Q. Wen, S. Zhao, Z. Sun, and S. He, "Discovering interpretable latent space directions of gans beyond binary attributes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 12 177–12 185.

[58] Z. Li, R. Tao, H. Niu, M. Yue, and B. Li, "Interpreting the latent space of gans via correlation analysis for controllable concept manipulation," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 1942–1948.

[59] A. Creswell and A. A. Bharath, "Inverting the generator of a generative adversarial network," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 7, pp. 1967–1974, 2018.

[60] S. Pidhorskyi, D. A. Adjeroh, and G. Doretto, "Adversarial latent autoencoders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 104–14 113.

[61] S. Guan, Y. Tai, B. Ni, F. Zhu, F. Huang, and X. Yang, "Collaborative learning for faster stylegan embedding," *arXiv preprint arXiv:2007.01758*, 2020.

[62] Y. Nitzan, A. Bermano, Y. Li, and D. Cohen-Or, "Disentangling in latent space by harnessing a pretrained generator," *arXiv preprint arXiv:2005.07728*, vol. 2, no. 3, 2020.

[63] F. Ma, U. Ayaz, and S. Karaman, "Invertibility of convolutional generative networks from partial measurements," 2019.

[64] D. Bau, J.-Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, and A. Torralba, "Seeing what a gan cannot generate," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4502–4511.

[65] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[66] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[67] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*. PMLR, 2015, pp. 2048–2057.

[68] Y. Tang, D. Nguyen, and D. Ha, "Neuroevolution of self-interpretable agents," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 414–424.

[69] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," *arXiv preprint arXiv:1906.05909*, 2019.

[70] A. Pandey and D. Wang, "Dense cnn with self-attention for time-domain speech enhancement," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1270–1279, 2021.

[71] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua, "Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5659–5667.

[72] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[73] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "Bam: Bottleneck attention module," *arXiv preprint arXiv:1807.06514*, 2018.

[74] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[75] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[76] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[77] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[78] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, "Gan dissection: Visualizing and understanding generative adversarial networks," *arXiv preprint arXiv:1811.10597*, 2018.

[79] C. Yang, Y. Shen, and B. Zhou, "Semantic hierarchy emerges in deep generative representations for scene synthesis," *International Journal of Computer Vision*, vol. 129, no. 5, pp. 1451–1466, 2021.