ATML Report

Giorgia Adorni
giorgia.adorni@usi.ch

Felix Boelter
felix.boelter@usi.ch

Stefano Carlo Lambertenghi
stefano.carlo.lambertenghi@usi.ch

Abstract

The field of image origination through generative modeling is abundantly discussed at the moment, as it can be used for an extremely varied range of applications such as up-scaling already existing images, creating non-existing objects such as interior design scenes, products or even human faces but also to achieve transfer-learning processes. In this context, GANs (Generative Adversarial Networks) are a class of widely studied machine learning frameworks first appeared in the paper "*Generative adversarial Nets*" by Goodfellow et al. [1] that achieve the aforementioned goal. In our work we will reproduce and evaluate a novel variation of the original GAN network, the GANformer, proposed in "*Generative adversarial Transformers*" by Hudson and Zitnick [2].

1 Introduction

This objective of this project is to investigate the reliability and reproducibility of a paper accepted for publication in a top machine learning conference. The models have been implemented using code and information provided by the authors. Actually we are not participating formally to the *Fall Edition of the ML Reproducibility Challenge 2021*.

With this work, we are going to verify the empirical results and claims of the paper "*Generative adversarial transformers*" by Hudson and Zitnick [2], by reproducing three of the computational experiments performed by the authors: (1) the **StyleGAN2** by Karras et al. [3, 4], a GAN network which uses one global latent style vector to modulate the features of each layer, hence to control the style of all image features globally, (2) the **GANformer** with **Simplex Attention** by Hudson and Zitnick [2], a generalisation of the StyleGAN design with k latent vectors that cooperate through attention, allowing for a spatially finer control over the generation process since multiple style vectors impact different regions in the image concurrently, in particular permitting communication in one direction, in the generative context – from the latents to the image features, and (3) the **GANformer** with **Duplex Attention** by Hudson and Zitnick [2], which is based on the same principles as the previous but propagating information both from global latents to local image features, enabling both top-down and bottom-up reasoning to occur simultaneously.

The first model is used as baseline, while the remaining are the architectures introduced by the authors. Thy consider the GANformer has "a novel and efficient type of transformer" which demonstrates its strength and robustness over a range of task of visual generative modelling — simulated multi-object environments (real-world indoor and out-door scenes) — achieving state-of-the-art results in terms of both image quality and diversity, while benefiting from fast learning and better data-efficiency.

2 Related works/Background

2.1 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) [1], are deep-learning-based generative models which learn to determine whether a sample is from the model or the data distribution.

The classic architecture, shown in Figure 1, is composed by two main neural networks: the *generator* $G(z)$, which take as input a sample from the latent space z, or a vector drawn randomly from a Gaussian distribution, and use it to generate new plausible examples in the problem domain — images in our case — , and the *discriminator* $D(x)$, which takes an example from the domain as input, and predicts a binary class label which classify the examples as real (coming from the training dataset) or fake (generated by G).

The two models are trained together: the discriminator D estimates the probability that sample x is generated by G or is a real sample, and aims at maximising the probability of assigning the correct label to both real and fake samples,
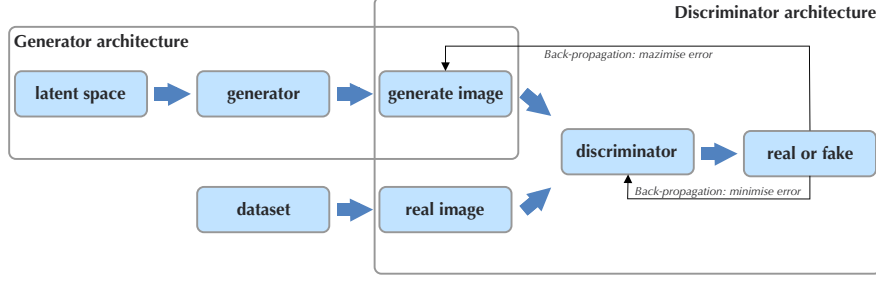
Figure 1: **GAN architecture.** Visualisation of the two components of a GAN: generator and discriminator.

while the generator G is trained to maximise the probability of the discriminator D making a mistake, so it aims to minimise $\log(1 - D(G(z))))$.

Combining the two objectives for $G(z)$ and $D(x)$ we get the *GAN min-max game* with the value function $V(G, D)$:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p*(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))))] \tag{1}$$

GANs typically work with images, as in our case, for this reason, both the generator and discriminator models use Convolutional Neural Networks (CNNs).

2.2 StyleGAN2

StyleGAN are a re-design of the GANs generator architecture, which aim is to control the image synthesis process [4]. The architecture is shown in Figure 2.

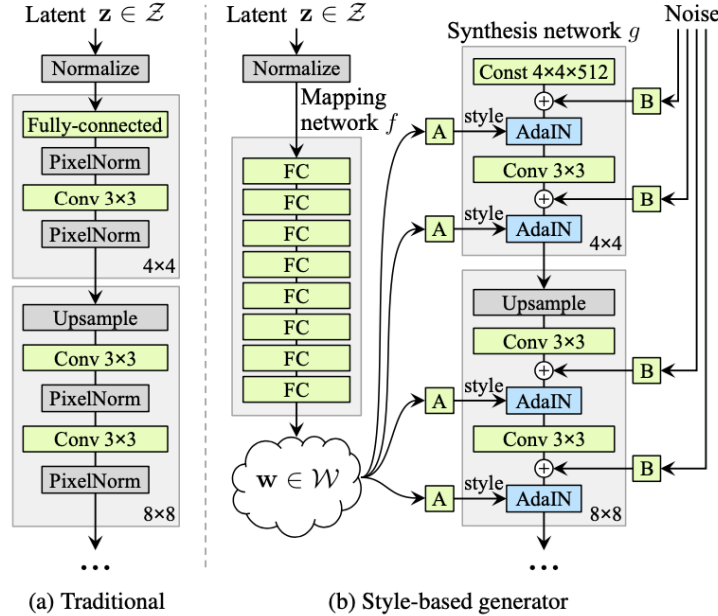(a) Traditional

(b) Style-based generator

Figure 2: **StyleGAN architecture.** Visualisation of the two components of a GAN: generator and discriminator.

The StyleGAN approach departs from the design of a common generator network, consisting in a multi-layer CNN. A traditional GAN generator feeds the latent code z though the input layer only, to start the up-sampling process. Karras et al. [4] introduce a feed-forward mapping network $f : Z \to W$ which processes the latent code z in the input latent space Z, and output an intermediate latent vector $w \in W$. w, in turn, interacts directly with each convolution through the synthesis network g and, in particular, with the *Adaptive Instance Normalisation (AdaIN)* [5] aligns the mean and variance of the content features with those of the style features, meaning that ia able to globally controls these parameters and so the strength of image features at different scales.

The *AdaIN* operation is defined as:

$$\text{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y), \tag{2}$$

where x is the content input, y a style input, and the channel-wise mean and variance of x are aligned to match those of y.

Moreover, by introducing explicit Gaussian noise inputs after each convolution, Karras et al. [4] provide the generator with a direct means to generate stochastic detail: an automatic and unsupervised separation of high-level attributes (e.g., pose, identity) from stochastic variation (e.g., freckles, hair) is obtained in the generated images, and intuitive scale-specific mixing and interpolation operations are enabled.

StyleGAN2 [3] are a revisiting of the architecture of the StyleGAN synthesis network. Figure 3 shows the various changes made to the original architecture up to the final network.
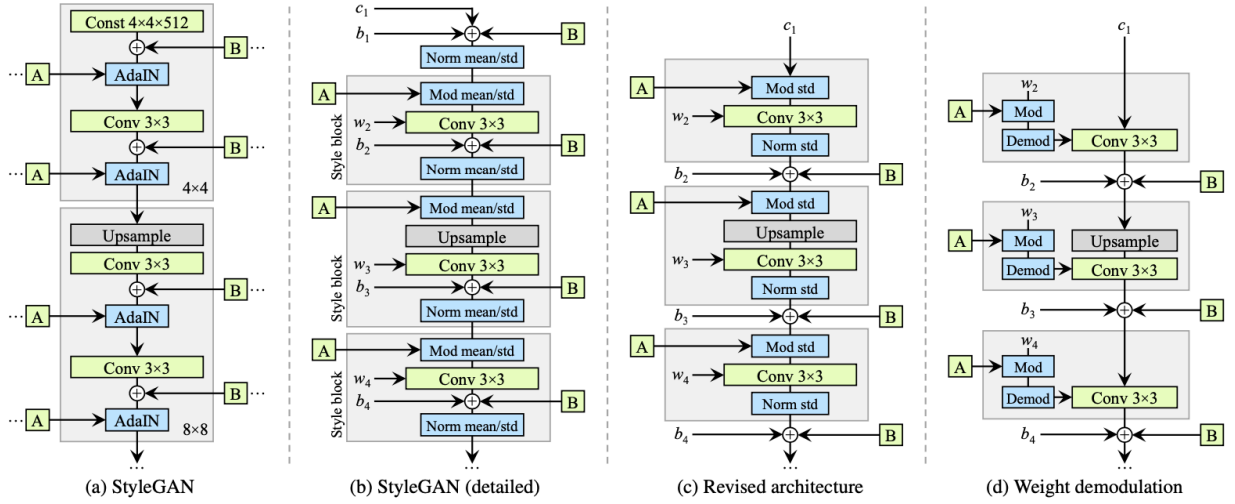

Figure 3: **StyleGAN2 architecture.** Visualisation of the two components of a GAN: generator and discriminator.

In particular, some redundant operations have been removed from the original StyleGAN architecture. Bias and noise are operations which might have conflicting interests, and their application within the style block caused their relative impact to be inversely proportional to the current style's magnitudes. For this reason, they have been moved outside the style block and separated, allowing to obtain more predictable results. Furthermore, after this change, the mean is no longer necessary, but it is sufficient for the normalisation and modulation to operate on the standard deviation alone. Moreover, the application of bias, noise, and normalization to the constant input has been safely removed.

When the style block consists in a modulation, a convolution, and a normalisation operation, it is possible to restructure the AdaIN has a demodulation operation, which is applied to the weights associated with each convolution layer. AdaIN uses different scale and shift parameters to align different areas of w — the activations of intermediate activations — with different regions of the feature map (either within each feature map or via grouping features channel-wise by spatial location), while weight demodulation takes the scale and shift parameters out of a sequential computation path, instead baking scaling into the parameters of convolutional layers.

The StyleGAN2 architecture makes it possible to control the image synthesis via scale-specific modifications to the styles. In particular, this approach attains layer-wise decomposition of visual properties, allowing StyleGAN to control global aspects of the picture such as pose, lighting conditions or colour schemes, in a coherent manner over the entire image.

But while StyleGAN successfully disentangles global properties, it is more limited in its ability to perform spatial decomposition, as it provides no direct means to control the style of a localized regions within the generated image.

3

2.3 Transformers

Transformers are deep learning models based on an *attention mechanism* designed to handle sequential input data and evaluate the relationship between each input-output item [6]. This models, unlike Recurrent Neural Networks (RNNs), avoid using convolutions or aligned sequence, and do not necessarily require ordered input data to be processed.

The architecture is composed by an encoder-decoder structure where, the *encoder* maps an input sequence of symbol representations $(x_1, \ldots, x_n)$ to a sequence of continuous representations $z = (z_1, \ldots, z_n)$, while the *decoder*, given z, generates an output sequence $(y_1, \ldots, y_m)$ of symbols one element at a time. Each transformer module (encoder-decoder) is connected to each other via feed-forward layers.

An *attention function* [6], can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

The input consists of queries and keys of dimension d_k, and values of dimension d_v. The dot products of the query with all keys is computed, each divided by $\sqrt{d_k}$ and then a softmax function is applied to obtain the weights on the values. In practice, the attention function is computed on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

Instead of performing a single attention function, transformers use multiple self-attentions, called *multi-head attention*, allowing the model to jointly attend to information from different representation subspaces at different positions, learning attention relationship independently [6]. Vaswani et al. [6] used multiple multi-head attentions stacked on the top of each other, in this way, the parallel attention layers enables to pass multiple input sequences simultaneously instead of one at a time, allowing for more parallelisation and therefore reducing training times if you have access to sufficient computational resources.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1 \ldots, \text{head}_h)W^O, \tag{4}$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$, and the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$.

The transformer uses multi-head attention in three different ways:

1. In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence.

2. The encoder contains self-attention layers, in which all of the keys, values and queries come from the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.

3. Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections.

3 Methodology

3.1 Generative Adversarial Transformers

The Generative Adversarial Transformers (GANsformer), as presented in [2], are models which combine GANs and the transformers to generate better and more realistic examples.

GANs, and in particular the StyleGAN2 model [3], presented in Section 2.2 and shown in Figure 3, is used as a starting point for the GANformer design for the properties it owns as CNN: they are powerful generators for the overall style of the image, since by nature they merge the local information of the pixels together with the general information regarding the image. However, they are less powerful regarding small details of localised regions within the generate image itself, since they miss out on the long range interaction of the faraway pixel.

Accordingly, GANsformer take advantage of the transformers attention mechanism to make the StyleGAN2 architecture even more powerful: the integration of attention in the architecture allows the network to draw global dependencies

between input and output, and understand the context of the image thanks to the transformer's strength for long-range interactions. Thus, rather than focusing on using global information and controlling all features globally, transformer use attention to propagate information from the local pixels to the global high-level representation and vice versa.

The *bipartite transformer* structure computes *soft attention*, iteratively aggregating and disseminating information between the generated image features and a compact set of *latent variables* enable bidirectional interaction between these dual representations.

The *transformer network* corresponds to the *multi-layer bidirectional transformer encoder* (BERT) described by Devlin et al. [7], which interleaves *multi-head self-attention* and *feed-forward layers*. Each pair of self-attention and feed-forward layers is intended as a *transformer layer*, hence, a transformer is a stack of several such layers.

The *self-attention layer* considers all pairwise relations among the input elements, so to update each single element by attending to all the others. The *bipartite transformer* generalises this formulation, featuring instead a bipartite graph between two groups of variables — in the GAN case, latents and image features. There are two attention operations that could be computed over the bipartite graph, depending on the direction in which information propagates, (1) *simplex attention* permits communication either in one way only, in the generative context, from the latents to the image features, and (2) *duplex attention* which enables it both top-down and bottom-up ways.

Attention can be used in several ways:

1. Simplex Attention: when attention is applied in one direction only from the latents to the image features (top-down).
2. Duplex Attention: when attention is applied in the two directions: latents to image features (top-down) and then image features back to latents (bottom-up), so that each representation informs the other iteratively.
3. Self Attention between latents: can also be used so to each direct interactions between the latents.
4. Self Attention between image features (SAGAN model): prior approaches used attention directly between the image features, but this method does not scale well due to the quadratic number of features which becomes very high for high-resolutions.

In particular, the attention layers are added in between the convolutional layers of both the generator and discriminator.

The discriminator model performs multiple layers of convolution downsampling on the image, reducing the representation's resolution gradually until making final prediction. Optionally, attention can be incorporated into the discriminator as well where it has multiple k aggregator variables, that use attention to adaptively collect information from the image while being processed. We observe small improvements in model performance when attention is used in the discriminator, although note that most of the gain in using attention based on our observations arises from the generator.

3.1.1 Simplex attention

As already mentioned, simplex attention distributes information in a single direction over the bipartite transformer graph.

Formally, let $X^{n \times d}$ denote an input set of n vectors of dimension d — where, for the image case, $n = W \times H$ — and $Y^{m \times d}$ denote a set of m aggregator variables — the latents, in the generative case. Specifically, the attention is computed over the derived bipartite graph between these two groups of elements, as in Equation (3), moreover:

$$a(X, Y) = \text{Attention}(q(X), k(Y), v(Y)), \tag{5}$$

where $q(\cdot), k(\cdot), v(\cdot)$ are functions that respectively map elements into queries, keys, and values, all maintaining dimensionality d. The mappings are provided with positional encodings to reflect the distinct position of each element (e.g. in the image). This bipartite attention is a generalisation of self-attention, where $Y = X$.

Standard transformers implement an additive update rule of the form:

$$u^a(X, Y) = \text{LayerNorm}(X + a(X, Y)), \tag{6}$$

however, [2] used the retrieved information to control both the scale as well as the bias of the elements in X, in line with the practice promoted by the StyleGAN model [4]:

$$u^s(X, Y) = \gamma(a(X, Y)) \odot \omega(X) + \beta(a(X, Y)), \tag{7}$$

where $\gamma(\cdot), \beta(\cdot)$ are mappings that compute multiplicative and additive styles (gain and bias), maintaining dimensionality d, and $\omega(X) = X - \mu(X)$ normalises each element with $\sigma(X)$ respect to the other features. By normalizing X (image features), and then letting Y (latents) control the statistical tendencies of X, the information propagation from Y to X is enabled, allowing the latents to control the visual generation of spatial attended regions within the image, so as to guide the synthesis of objects or entities. The multiplicative integration permits significant gains in the model performance.

3.1.2 Duplex attention

Duplex attention can be explained by taking into account the variables Y to set their own key-value structure: $Y = (K^{n \times d}, V^{n \times d})$, where the values store the content of the Y variables, as before (e.g. the randomly sampled latent vectors in the case of GANs) while the keys track the centroids K of the attention-based assignments between Y and X, which can be computed as $K = a(Y, X)$ — namely, the weighted averages of the X elements using the bipartite attention distribution derived through comparing it to Y. Consequently, the new update rule is defined as follows:

$$u^d(X, Y) = \gamma(A(X, K, V)) \odot \omega(X) + \beta(A(X, K, V)), \tag{8}$$

where, two attention operations are compound on top of each other: first compute the *soft attention* assignments between X and Y, by $K = a(Y, X)$, and then refine the assignments by considering their centroids, by $A(X, K, V)$. This is analogous to the *k-means algorithm* and works more effectively than the simpler update u^a defined above in Equation (7).

Finally, to support bidirectional interaction between X and Y (the image and the latents), two reciprocal simplex attentions are chain from X to Y and from Y to X, obtaining the duplex attention, which alternates computing $Y := u^a(Y, X)$ and $X := u^d(X, Y)$, such that each representation is refined in light of its interaction with the other, integrating together bottom-up and top-down interactions.

3.1.3 Vision-specific adaptations

Some adaptations, that will be later described in Section 4.2, have been applied to the structure of the GANformer in order to foster an interesting communication flow. Rather than densely modelling interactions among all the pairs of pixels in the images, instead it supports *adaptive long-range interaction* between far away pixels in a moderated manner, passing through a compact and global latent bottleneck that selectively gathers information from the entire input and distributes it back to the relevant regions.

3.1.4 The Generator and Discriminator Networks

The generator likewise is composed of two parts, a mapping network and a synthesis network. The mapping network of a GANformer is the same as that of the styleGAN2. In the synthesis network, while in the StyleGAN2, a single global w vector controls all the features equally, the GANformer uses attention so that the k latent components specialise to control different regions in the image to create it cooperatively, and therefore perform better especially in generating images depicting multi-object scenes, allowing also for a flexible and dynamic style modulation at the region level.

The bipartite transformer offers a solution to the StyleGAN limitation in its ability to perform spatial decomposition which leaded to the impossibility of controlling the style of a localised regions within the generated image.

Instead of controlling the style of all features globally, the new attention layer is used to perform *adaptive region-wise modulation*. The latent vector z is split into k components, $z = [z_1, \ldots, z_k]$ and, as in StyleGAN [4], pass each of them through a shared mapping network, obtaining a corresponding set of intermediate latent variables $Y = [y_1, ..., y_k]$.

During synthesis, after each CNN layer in the generator, the feature map X and latents Y play the roles of the two element groups, mediating their interaction through our new attention layer (either simplex or duplex). This setting thus allows for a flexible and dynamic style modulation at the region level.

Since soft attention tends to group elements based on their proximity and content similarity, the transformer architecture naturally fits into the generative task and proves useful in the visual domain, allowing the model to exercise finer control in modulating local semantic regions. This capability turns to be especially useful in modelling highly-structured scenes.

For the discriminator, attention is applied after every convolution using trained embeddings to initialise the aggregator variables Y, which may intuitively represent background knowledge the model learns about the task. At the last layer are concatenated these variables Y to the final feature map X to make a prediction about the identity of the image source. This structure empowers the discriminator with the capacity to likewise model long-range dependencies, which can aid it in its assessment of the image fidelity, allowing to acquire a more holistic understanding of the visual modality.

4 Implementation

The code has been implemented using code and information provided by the authors.

4.1 Datasets

The original paper [2] explored the GANformer model on four datasets for images and scenes: CLEVR [8], LSUN-Bedrooms [9], Cityscapes [10] and FFHQ [4].

Initially, we tried to use the Cityscapes dataset, since it is the smaller among the four: it contains 24998 images with 256x256 resolution. However, the time to complete the training was to high on this dataset (TODO: how much?), even if using computational resources like GPU.

For this reason, we switch to another dataset, the Google Cartoon Set [11][1], containing 10k 2D cartoon avatar images with 64x64 resolution, composed of 16 components that vary in 10 artwork attributes, 4 colour attributes, and 4 proportion attributes (see in Table 1).

Table 1: Attributes of the Cartoon Set `https://google.github.io/cartoonset/download`.

		# Variants	Description
Artwork	chin_length	3	Length of chin (below mouth region)
	eye_angle	3	Tilt of the eye inwards or outwards
	eye_lashes	2	Whether or not eyelashes are visible
	eye_lid	2	Appearance of the eyelids
	eyebrow_shape	14	Shape of eyebrows
	eyebrow_weight	2	Line weight of eyebrows
	face_shape	7	Overall shape of the face
	facial_hair	15	Type of facial hair (type 14 is no hair)
	glasses	12	Type of glasses (type 11 is no glasses)
	hair	111	Type of head hair
Colors	eye_color	5	Color of the eye irises
	face_color	11	Color of the face skin
	glasses_color	7	Color of the glasses, if present
	hair_color	10	Color of the hair, facial hair, and eyebrows
Proportions	eye_eyebrow_distance	3	Distance between the eye and eyebrows
	eye_slant	3	Similar to eye_angle, but rotates the eye and does not change artwork
	eyebrow_thickness	4	Vertical scaling of the eyebrows
	eyebrow_width	3	Horizontal scaling of the eyebrows

4.2 Hyper-parameters

Note that in the code provided by the author [2], the hyper-parameters are not the same mentioned in the article.

A *kernel size* of $k = 3$ is used after each application of the attention, together with a *Leaky ReLU non-linearity* after each convolution and then upsample or downsample the features X, as part of the generator or discriminator respectively, as in e.g. StyleGAN2 [3]. To account for the features location within the image, we use a sinusoidal positional encoding along the horizontal and vertical dimensions for the visual features X, and trained positional embeddings for the set of latent variables Y. Overall, the bipartite transformer is thus composed of a stack that alternates attention (simplex or duplex), convolution, and upsampling layers, starting from a 4×4 grid up to the desirable resolution.

Both the simplex and the duplex attention operations enjoy a bilinear efficiency of $\mathcal{O}(mn)$ thanks to the network's bipartite structure that considers all pairs of corresponding elements from X and Y. Since, as we see below, we maintain Y to be of a fairly small size, choosing m in the range of 8–32, this compares favourably to the prohibitive $\mathcal{O}(n^2)$ complexity of self-attention, which impedes its applicability to high-resolution images.

As to the loss function, optimization and training configurations, we adopt the settings and techniques used in Style-GAN2 [3], including in particular style mixing, stochastic variation, exponential moving average for weights, and a non-saturating logistic loss with a lazy R1 regularization.

4.3 Experimental setup

The source code of our work is available at the following GitHub repository: `https://github.com/GiorgiaAuroraAdorni/gansformer-reproducibility-challenge`.

---

[1] `https://google.github.io/cartoonset`

The approaches proposed in both the original paper codebase by Karras et al. [3] and by Hudson and Zitnick [2] have been implemented in Python using TensorFlow [12], so, according to that, we used the same setup. We created a Jupyter Notebook which runs all the experiments in Google Colaboratory, which allows us to write and execute Python in the browser.

All the models have been trained on a Tesla P100-PCIE-16GB (GPU) provided by Google Colab Pro.

4.4   Computational requirements

In the original paper [2], they evaluate all models under comparable conditions of training scheme, model size, and optimization details, implementing all the models within the codebase introduced by the Style-GAN authors [3]. All models have been trained with images of 256×256 resolution and for the same number of training steps, roughly spanning a week on 2 NVIDIA V100 GPUs per model (or equivalently 3-4 days using 4 GPUs).

Considered that we had available just one GPU and not enough time to reproduce this settings, we decided to resize the images from 256×256 to 64x64 resolution.

For the GANformer, we select k–the number of latent variables, from the range of 8–32.

All models have been trained for the same number of steps, which ranges between 5k to 15k kimg training samples.

The paper present results after training 100k, 200k, 500k, 1m, 2m, 5m and 10m samples

For the StyleGAN2 model we present results after training 100k (2h30m), 200k (6h), 500k, obtaining good results. Note that the original StyleGAN2 model has been trained by its authors [3] for up to 70k kimg samples, which is expected to take over 90 GPU-days for a single model.

For the GANformer, the authors [3] show impressive results, especially when using duplex attention: the model manages to learn a lot faster than competing approaches, generating astonishing images early in the training. This model is expected to take 4 GPU-days.

However, we are not able to replicate this achievements, first because this model learns significantly slower than the StyleGAN2, which is instead able to produce high-quality images in approximately x-times less training steps than the GANformer.

(in the paper they reach better results with the GANformer with 3-times less training steps than the StyleGAN2, but they don't specify the time required for this training steps...)

5   Results

Sample images from different points in training are based on the same sampled latent vectors, thereby showing how the image evolves during the training.

For CLEVR and Cityscapes, we present results after training to generate 100k, 200k, 500k, 1m, and 2m samples. For the Bedroom case, we present results after 500k, 1m, 2m, 5m and 10m generated samples during training

These results show how the GANformer, especially when using duplex attention, manages to learn a lot faster than competing approaches, generating impressive images early in the training

The Frechet Inception Distance (FID) is one of the most popular metrics for evaluating GANs, which provides stable and reliable indications of *image fidelity* and *diversity*. It is a measure of similarity between curves that takes into account the location and ordering of the points along the curves. For this specific application, FID is used to measures the feature distance between the real and the generated images, but it can be used for measuring the distance between two distributions as well.

Visualisation of the models' comparison throughout the training.
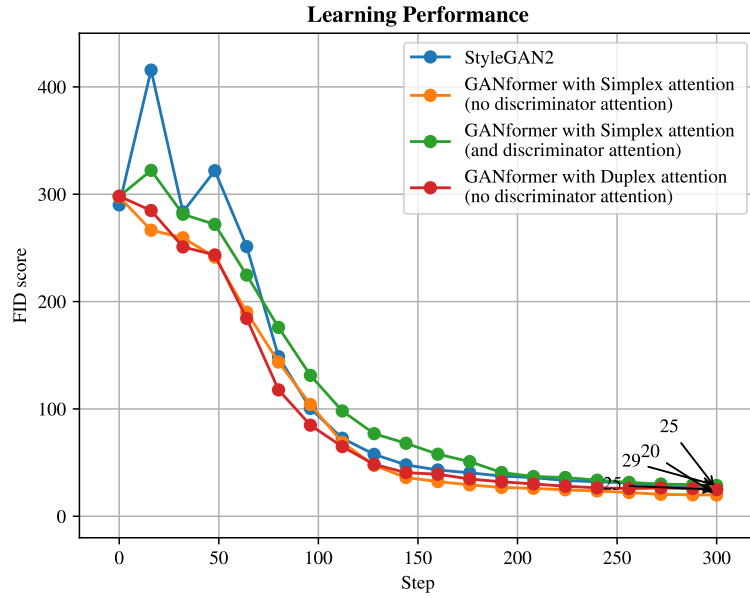
Comparison of results

Figure 4: **Comparison between the StyleGAN2 and the GANformer models.** We evaluate the models according to FID score along 300k image samples. The score is computed every sixteen checkpoints.
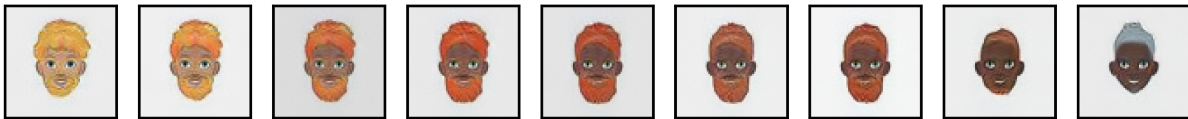

(a) StyleGAN2.

(b) GANformer model with Simplex attention and discriminator without attention.

Figure 5: **Visualisation of 9 images generated with the various models**.

(a) StyleGAN2.

(b) GANformer model with Simplex attention and discriminator without attention.

Figure 6: **Simple z interpolation using of the various models**.

Table 2: Original paper's reported results (mean of results over the 4 Datasets used by the authors)

Model	FID	IS	Precision	Recall
StyleGAN2	11.2925	2.7425	52.015	23.9775
GANformer, Simplex attention	10.2925	2.82	56.7625	18.205
GANformer, Duplex attention	7.215	2.7825	55.4475	33.9375

Table 3: Our results (One dataset)

Model	FID	IS	Precision	Recall
StyleGAN2	24.7722	2.5001	todo	todo
GANformer, Simplex attention	28.1098	2.5806	0.0015	0.0076
GANformer, Duplex attention	todo	todo	todo	todo

6 Discussion and conclusion

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680. MIT Press, 2014. URL http://arxiv.org/abs/1406.2661.

[2] Drew A. Hudson and C. Lawrence Zitnick. Generative Adversarial Transformers. 2021. URL http://arxiv.org/abs/2103.01209.

[3] 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. URL http://arxiv.org/abs/1912.04958.

[4] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. URL http://arxiv.org/abs/1812.04948.

[5] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. URL http://arxiv.org/abs/1706.03762.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. URL http://arxiv.org/abs/1810.04805.

[8] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017. URL http://arxiv.org/abs/1612.06890.

[9] 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. 2015. URL http://arxiv.org/abs/1506.03365.

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. URL http://arxiv.org/abs/1604.01685v2.

[11] Cole Forrester, Mosseri Inbar, Krishnan Dilip, Sarna Aaron, Maschinot Aaron, Freeman Bill, and Fuman Shiraz. Cartoon set. https://google.github.io/cartoonset/.

[12] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.