A CONDITIONAL PYTORCH STYLEGAN2

Aurélien COLIN[1,2], *Ronan FABLET*[1], *Pierre TANDEO*[1], *Nicolas LONGEPE*[2], *Samir SAOUDI* [1]

1 - IMT Atlantique, Brest, France
2 - Collecte Localisation Satellites (CLS), Plouzané, France

ABSTRACT

We release a PyTorch implementation of the second version of the StyleGan2 architecture. Besides a conditional component to the latent vector, we enable the possibility to create pictures from a categorization dataset, by choosing the class of the generated sample.

The implementation is published on GitHub : [TODO: insert url]

Index Terms— GAN, Conditional Generation

1. INTRODUCTION

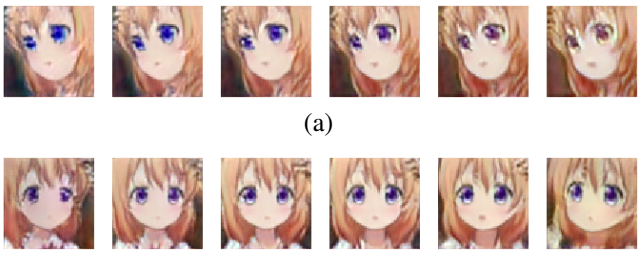
The recent years have seen an increase in the number of image generation algorithms using deep learning architectures. The two most popular architectures are the Generative Adversarial Networks (GAN) introduced in [1] and the variational auto-encoders [2]. Both rely on the representation of a picture in a latent space and the possibility to randomly sample latent vectors in this latent space. These models then transform any latent vector into an image. Among the variety of GAN architectures, StyleGAN2 [3] is one of the most promising.

Within StyleGAN2 framework, one cannot directly control the generated image.One could use the acceptance-rejection method [4], that is to say asking to generate pictures until we found one with some expected characteristics. This approach implies an increase in the computation time which could be detrimental for the generation of rare features.

Some works have been focused on the modification of a picture based on their latent encoding. Two primary ways have been explored: an *a posteriori* edition of the latent vector [5][6] and the *a priori* integration of semantic information during the training [7][8]. The first approach does not add semantic information to the input of the GAN but needs the latent vectors. Inverting a GAN to retrieve the input given a picture is difficult. However, several methods have been built to access those latent vectors. A secondary model is trained to inverse the generator of the GAN [9] or the GAN is integrated into a autoencoder framework [10]. With these methods, the latent vector can be computed by running the invertor model on any pictures. Another popular method is to optimize a random latent vector in order to progressively decrease a reconstruction loss according to some input picture [11]. Being an iterative process, the inference is slower. However, it does not need to train a network or modify the architecture of the generator.

Once latent vectors have been obtained, they are altered to provide some preferential features. This can be done by interpolating [12] or mixing [13] the latent vectors of two pictures. However, this methodology does not allow the edition of a single picture as we need, at least, two latent vectors.

Achieving this kind of edition is equivalent to find point in the latent space where most of the features have not changed except for a few of them. Figure 1 show how it is possible to change the face direction or the eye color while maintaining the same kind of picture overall.



(a)

(a)

Fig. 1. Movement on one dimension of the latent space to modify the eye color (a) and the face direction (b).

Such path finding has been notably studied in [14] and [15].

Alternatively, it is possible to let a secondary network learn to ”rig” a latent vector to match some characteristics as it has been shown in [16].

The second approach is to constrain the latent space during the training of the generator such that it obtains properties known beforehand.

Conditional GANs [17] belong to this category as they divide the latent space into several subspaces according to a condition enforced in the input. The generator is trained in such a way that all the pictures of a subspace associated with a class share the same features.

From the introduction of StyleGAN [13], few variations have been implemented. LoGAN [18] was, to the best of our

knowledge, the first to integrate the conditional framework with the StyleGAN architecture. Recently, a few modifications of StyleGAN have been made [3], trying to improve the performance of this kind of architecture. In particular, it solve the "droplet" issue, namely the appearance of blob-shaped artefacts caused by the normalisation component of the AdaIN operation.

In this short paper, an implementation of the StyleGan2 architecture using the cGAN framework is presented.

2. NOTATIONS

As explained in the introduction, the goal of the GAN is to translate a latent vector $\tilde{x}$ into a picture y. The model performing this mapping is called the generator $\mathcal{G}$. From the domain $\mathbb{R}^d$, with d the dimension of the latent space, it returns an element of $[0,1]^{r_1 r_2 c}$, where r_1 and r_2 define the resolution and c the number of canals.

However, the latent space containing $\tilde{x}$ is not directly accessible: elements cannot be extracted from it directly. Thus, a second function, the mapper $\mathcal{M}$ is needed, taking a vector x of independent and identically distributed random variables generated from the standard distribution. With $\mathcal{M}$ the mapper, the latent vector is obtained by $\tilde{x} = \mathcal{M}(x)$. Usually, the dimension of x is set to be the same as the one of $\tilde{x}$ [13]. In other words, $\mathcal{M}$ is a non-linear application of $\mathbb{R}^d$ into $\mathbb{R}^d$.

Denoting $\mathcal{G}$ the generator network, the picture y is computed by the following equation:

$$y = \mathcal{G}(\tilde{x}) = \mathcal{G}(\mathcal{M}(x)) \tag{1}$$

With the GAN framework, an adversary is needed to estimate if a picture is generated by $\mathcal{G}$ or comes from the dataset. This is the work of the discriminator Δ. Δ is trained so that its output is null (and resp. one) if the picture is fake (resp. real).

To introduce that conditioning, the class information is encoded. The one-hot encoded vector l is used and defined with a class index c by the following formula.

$$l_i(c) = \begin{cases} 0 & if \quad i \neq c \\ 1 & if \quad i = c \end{cases} \tag{2}$$

3. STYLEGAN ARCHITECTURE WITH A CONDITIONAL COMPONENT

To extend the architecture of StyleGAN2 into the category-driven cGAN framework, an indicator of the class is included in both the discriminator and the generator [17]. The modification of the discriminator Δ is the most straightforward.

3.1. Building a New Discriminator

As the discriminator of a unconditionnal GAN has only to determine if an input is synthetic or genuine, it has only one output. For the conditional version of the generator, the last layer is modified so that it outputs one value per class. Thus, the validity of conditional discriminator $\Delta_{\mathcal{C}}$ is defined for the image y and the class c by:

$$\Delta_{\mathcal{C}}(y, c) = \sum \Delta(y) \cdot l(c) \tag{3}$$

Since l is a one-hot encoded vector, it is equivalent to take the c^{th} value of Δ. In other words, $\Delta_{\mathcal{C}}$ computes one discrimination value for each class and uses only the appropriate one (even if the picture is correct for another class).

Another configuration is to use the equation 4 in place of equation 2. Using this formulation, the generator is able to decrease the loss when the generation is accurate for a different label.

$$\tilde{l}_i(c) = \frac{l_i(c) + \epsilon}{1 + n * \epsilon} \tag{4}$$

3.2. Two Solutions for the Generator Block

For the generator block, we may integrate the category-driven conditioning as input to operators $\mathcal{M}$ or $\mathcal{G}$. Choosing between these two options is mainly dataset-dependent.

The first option is to concatenate the class information to the latent vector. Using a n-dimensional representation (one-hot????), the images of the same class thus share the n last elements of their latent vector. It implies that the latent manifold is constituted of disjoint sub-manifold. It is suitable for most categorization datasets since the labels are often exclusive. Such structure is depicted in the figure 2.b.

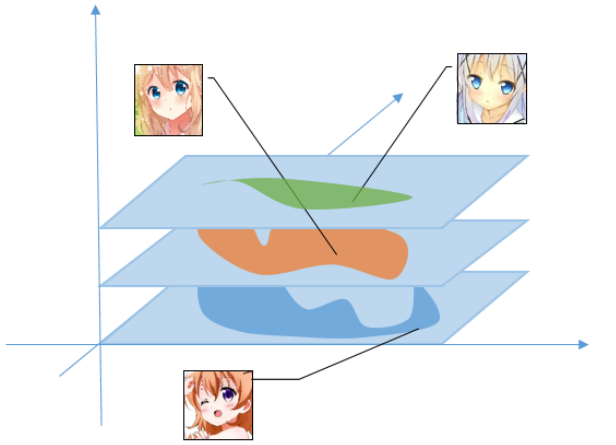
However, some datasets does not have this property. In the figure 2.a, the classes "7" and "1" are considered overlapping since some images could be rightfully attributed to any of these two classes. One could want to avoid the sub-manifolds being disjoint. The drawback is that the classes could be encoded with multiple elements of the latent vector.

3.3. Implementation issues

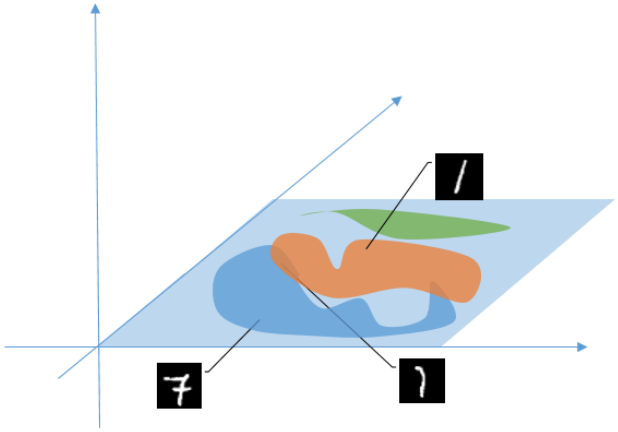
4. RESULTS

We report in this section some experiments with the proposed styleGan2 framework. To assert the quality of the generation, three goals are used:

- The generated images are convincing. This means that a human user will have difficulties to identify which samples are fake from a pool of both generated and real pictures.
- The output samples keep some diversity. If the generator is only able to produce almost identical images, its use is reduced.



(a)



(b)

Fig. 2. Illustration of the latent manifold if we consider the classes to be strictly disjoint (a) or to overlap (b).

- The generated samples are correctly associated with its label.

Considered dataset

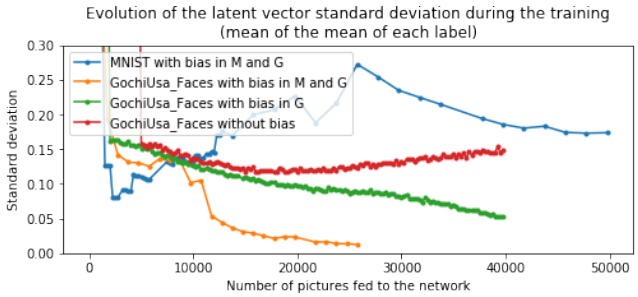
We consider two datasets. This first one, MNIST [19] contains 28x28x1 numerisation of the ten digits hand written. It is a key dataset for computer vision as it contains information mainly in shapes as the pixel take either 0 or 1 as value. Also, it contains 60k images for training. As cStyleGAN2 can only use power of two as the width or height of the picture, we are resizing all the samples to have a size of 32x32x1.

The second dataset is the recent GochiUsa_Faces [20]. It contains, in the train set, around 43k pictures from nine classes. As the resolution ranges from 26x26x3 to 987x987, only samples of more than 64x64 were kept. Thus, a dataset of 39572 pictures was created. Bicubic interpolation was used to reach a constant size of 128x128 pixels. In contrast with MNIST, the discrimination of the GochiUsa_Faces' classes rely primarly on the colorspace, as the shapes of the faces (with the exception of the bangs) are often similar.

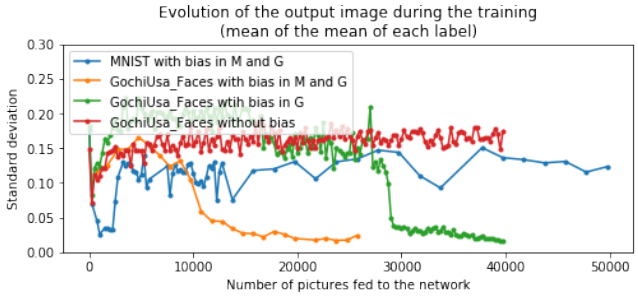
Diversity of the generation

An indicator of a good training is the diversity in the outputs. During our experiments, the model was collapsing in some configurations. The mode collapse refers to an issue occurring when the model output is constant with any inputs [21]. To illustrate this issue, the standard deviations of both the output of $\mathcal{M}$ and $\mathcal{G}$ are computed. Mode collapse implies a convergence to 0 of the standard deviation of y, the output image. However, if the mapper is subject to mode collapse, the standard deviation of the latent vector also converges to 0.

In the image space, pictures with identical features but shifted with small translations have a high standard deviation. Thus, the only value of the standard deviation giving information on the similarity (or dissimilarity) of a set of pictures is 0, as it means that they are identical pixel-wise. In other words, having a non-null standard deviation does not imply that the diversity of the images is high.



(a)



(b)

Fig. 3. Standard deviation of the latent vector, i.e. M's ouput, (a) and the output image, i.e. G's output, (b) through the training with either the MNIST or the GochiUsa_Faces dataset. In this later case, different configuration of bias (bias in M and G, bias in G and bias neither in M nor G) are considered.

Figure 3 illustrates that the model is not affected by mode collapse, when working with the MNIST dataset [19] even

if the model uses biases in $\mathcal{M}$ and $\mathcal{G}$. However, with the GochiUsa_Faces [20], using bias either in the mapper or the generator induces mode collapse at some point. Indeed, the mode collapse is caused by the magnitude difference between the biases of the dense layers and the weights of their inputs. Since the input of the mapper is generated with a normal distribution, it has a negligible probability to have high values. It means that if the biases are too high and the other weights too low, the output is almost equal to the biases no matter the input. This way, the networks can be led to generate almost constant outputs (the model suffer of mode collapse). Forcing the bias to zero prevents this behaviour.

Correctness of the class-condition

Another point is that the GAN correctly generates samples for each class. This can be checked by running a classifier on the output of $\mathcal{G}$. The figure 4 illustrates that the conditioning appears very quickly on datasets with colour-based information (GochiUsa_Faces) but also on shape-based datasets (MNIST). For each dataset, three models were trained without bias to decrease the risk of a statistical hazard.

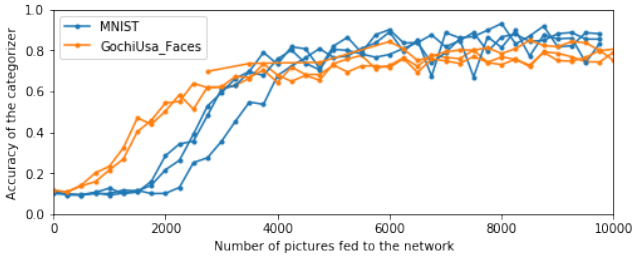
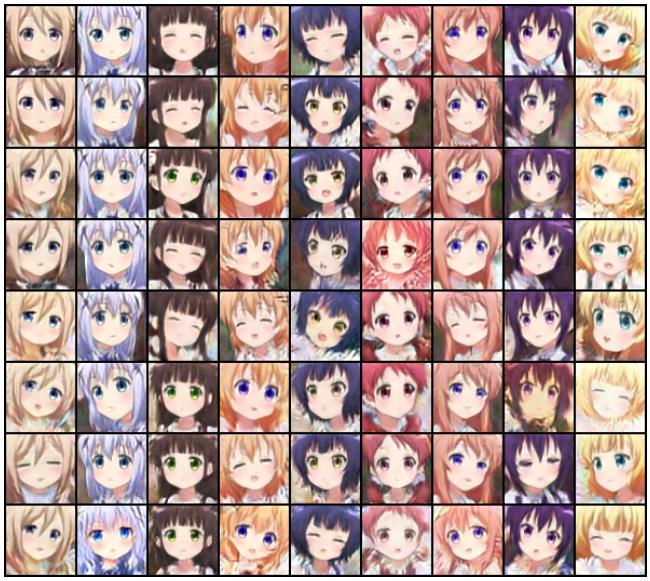


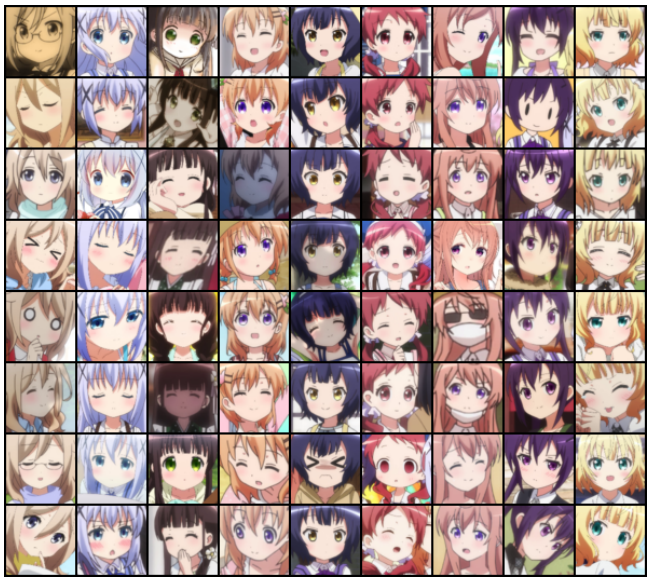
Fig. 4. Performance of a classifier on the generated samples during the training. The model is either trained on MNIST or GochiUsa_Faces

Visual inspection

Finally, the figure 5 depicts some examples for both the groundtruth GochiUsa_Faces dataset and the generation by the cGAN. The portraits depicted in the generation are not as diverse as the picture of the dataset. However, the proposed architecture is able to produce different positions and facial expressions.



(a)



(b)

Fig. 5. Samples generated by the cStyleGAN (a) and the groundtruth dataset (b)

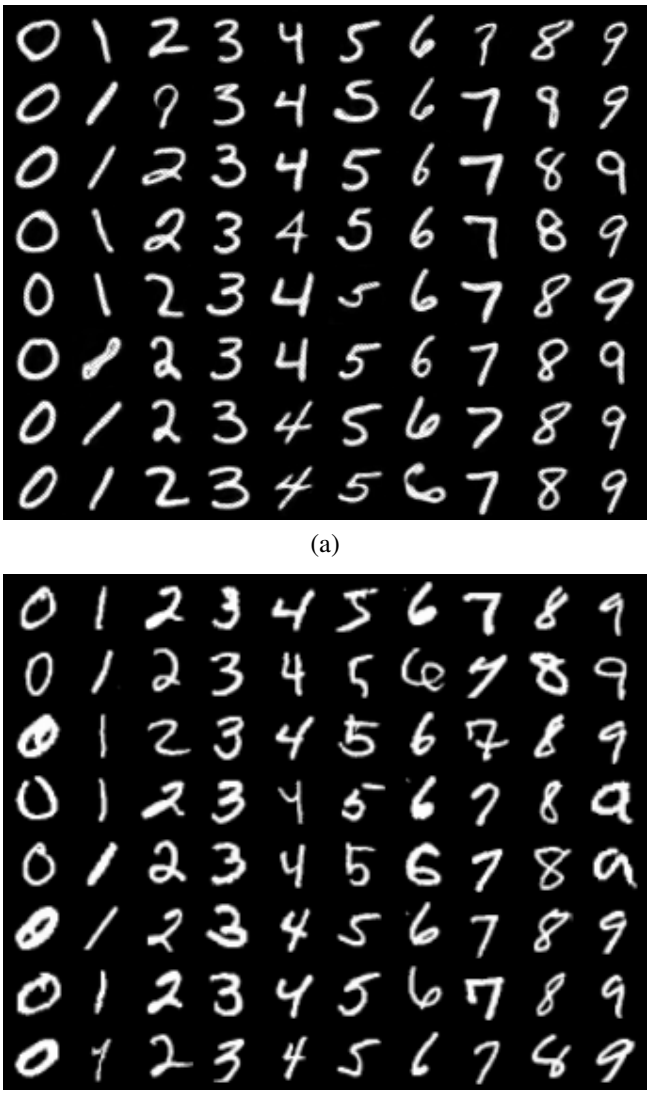
5. CONCLUSION

The preprint is the accompanying document of the implementation of the StyleGAN2 architecture published online at: xxxx. Besides a pytorch implementation, we extend the original StyleGAN2 architecture to a category-driven conditioning of the image generation process. This conditioning addresses both xxx and xxx.

Through experiments on a xxx dataset, we have illustrated

We publish a conditional implementation of the StyleGAN2 architecture. Modifications are applied both on the generator and the discriminator of the GAN to include a class conditioning.

The implementation enables to either integrate the conditioning to the mapper or the generator input, to generate respectively overlapping or disjoint classes.



(a)

(b)

Fig. 6. Samples generated by the cStyleGAN (a) and the groundtruth dataset (b) on the MNIST dataset

6. REFERENCES

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial networks," 2014.

[2] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," 2013.

[3] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila, "Analyzing and improving the image quality of stylegan," 2019.

[4] Ryan Turner, Jane Hung, Eric Frank, Yunus Saatci, and Jason Yosinski, "Metropolis-hastings generative adversarial networks," 2018.

[5] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou, "Interpreting the latent space of gans for semantic face editing," *CoRR*, vol. abs/1907.10786, 2019.

[6] Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen, "Attgan: Facial attribute editing by only changing what you want," 2017.

[7] Shuchang Zhou, Taihong Xiao, Yi Yang, Dieqiao Feng, Qinyao He, and Weiran He, "Genegan: Learning object transfiguration and attribute subspace from unpaired data," *CoRR*, vol. abs/1705.04932, 2017.

[8] Wei Shen and Rujie Liu, "Learning residual images for face attribute manipulation," *CoRR*, vol. abs/1612.05363, 2016.

[9] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville, "Adversarially learned inference," 2016.

[10] Stanislav Pidhorskyi, Donald Adjeroh, and Gianfranco Doretto, "Adversarial latent autoencoders," 2020.

[11] Antonia Creswell and Anil A. Bharath, "Inverting the generator of A generative adversarial network (II)," *CoRR*, vol. abs/1802.05701, 2018.

[12] Andrew Brock, Jeff Donahue, and Karen Simonyan, "Large scale GAN training for high fidelity natural image synthesis," *CoRR*, vol. abs/1809.11096, 2018.

[13] Tero Karras, Samuli Laine, and Timo Aila, "A style-based generator architecture for generative adversarial networks," *CoRR*, vol. abs/1812.04948, 2018.

[14] Paul Upchurch, Jacob R. Gardner, Kavita Bala, Robert Pless, Noah Snavely, and Kilian Q. Weinberger, "Deep feature interpolation for image content changes," *CoRR*, vol. abs/1611.05507, 2016.

[15] Ali Jahanian, Lucy Chai, and Phillip Isola, "On the "steerability" of generative adversarial networks," *CoRR*, vol. abs/1907.07171, 2019.

[16] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zöllhofer, and Christian Theobalt, "Stylerig: Rigging stylegan for 3d control over portrait images, cvpr 2020," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, june 2020.

[17] Mehdi Mirza and Simon Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[18] Cedric Oeldorf and Gerasimos Spanakis, "Loganv2: Conditional style-based logo generation with generative adversarial networks," *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, Dec 2019.

[19] Yann LeCun and Corinna Cortes, "MNIST handwritten digit database," 2010.

[20] "Gochiusa_faces, a dataset of faces from the gochiusa anime," 2020.

[21] Ian J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, 2017.