A CONDITIONAL PYTORCH STYLEGAN2

Aurélien COLIN

Institut Mines Telecom Atlantique (IMT-A), Plouzané, France
Collecte Localisation Satellites (CLS), Plouzané, France

ABSTRACT

We release a PyTorch implementation of the second version of the StyleGan2 architecture. With the addition of 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. Nowadays, the two main architectures are the Generative Adversarial Networks (GAN) introduced by Ian Goodfellow 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 pick elements, the latent vector, in this latent space. Thus, the model translates the latent vector into an image.

As the number of different GAN is increasing very fast, the state of the art is highly unstable. However, at the time of this paper redaction, StyleGAN2 [3] gives one the best, if not the best, results.

However, as we randomly pick the latent vector, we don't have any control on the generated picture. One could use the acceptance-rejection method, meaning asking to generate pictures until we found one with some selected characteristics. Such approach will imply an increase of the computation time. In the case of the generation of rare features, this increase could be quite dramatic.

The conditional generation [4] can solve this issue by forcing the latent space to behave in such a way that we could pick, randomly, a latent vector in a subspace of the latent space for which we know all the pictures have the feature we want to generate.

From the introduction of StyleGAN [5], a few variations have been implemented. LoGAN [6] was, at the best of our knowledge, the first to integrate the conditional framework with the StyleGAN architecture. In this short paper, we present an implementation of the StyleGan2 architecture using the cGAN framework.

2. NOTATIONS

As we have explained in the introduction, the goal of the GAN is to translate a latent vector, we denote it by $\tilde{x}$ into a picture y. The model doing this translation is called the generator $\mathcal{G}$.

However, the latent space containing $\tilde{x}$ is not directly accessible : we can't pick elements from it directly. Thus, we need a second function, the mapper $\mathcal{M}$ which takes a random vector $x \sim \mathcal{N}(\overrightarrow{0}, I_d)$, where d is the dimension of the random space, and returns $\tilde{x} = \mathcal{M}(x)$.

Thus, to sample a picture from a random vector, we have to apply the formula 1.

$$x \sim \mathcal{N}(\overrightarrow{0}, I_d) \quad ; \quad y = \mathcal{G}(\mathcal{M}(x)) \tag{1}$$

As we are working with the GAN framework, we need an adversary to estimate if a picture was generated by $\mathcal{G}$ or is genuine. This is the work of the discriminator Δ. Δ is trained so that its output is null if the picture was fake and one if it was real.

To introduce that conditioning, we need an encoding of the class information. We use the one-hot encoded vector l, defined with a class index c by the formula 2.

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

3. THE STYLEGAN ARCHITECTURE WITH A CONDITIONAL COMPONENT.

To add the condition to the GAN, we have to include the indication of the class in both the discriminator and the generator [4]. The modification of Δ is the most straightforward.

Building a New Discriminator

From the original discriminator Δ, we modify its last layer so that it has one output per class (up from only one output). Thus, we build the conditional discriminator $\Delta_{\mathcal{C}}$ as defined by the formula 3.

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

Since l is a one-hot encoded vector, it is equivalent to take the c^{th} value of Δ. In other words, $\Delta_{\mathcal{C}}$ will compute one discrimination value for each class and use only the appropriate one. It implies that the generation of a sample incorrectly associated with its label will be penalized the same way that would any picture categorized as fake.

Another configuration is to use the equation 4 in place of equation 2. With this formula, the generator will be able to decrease the loss when the generation was accurate for a different label.

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

Two Solutions for the Generator Block

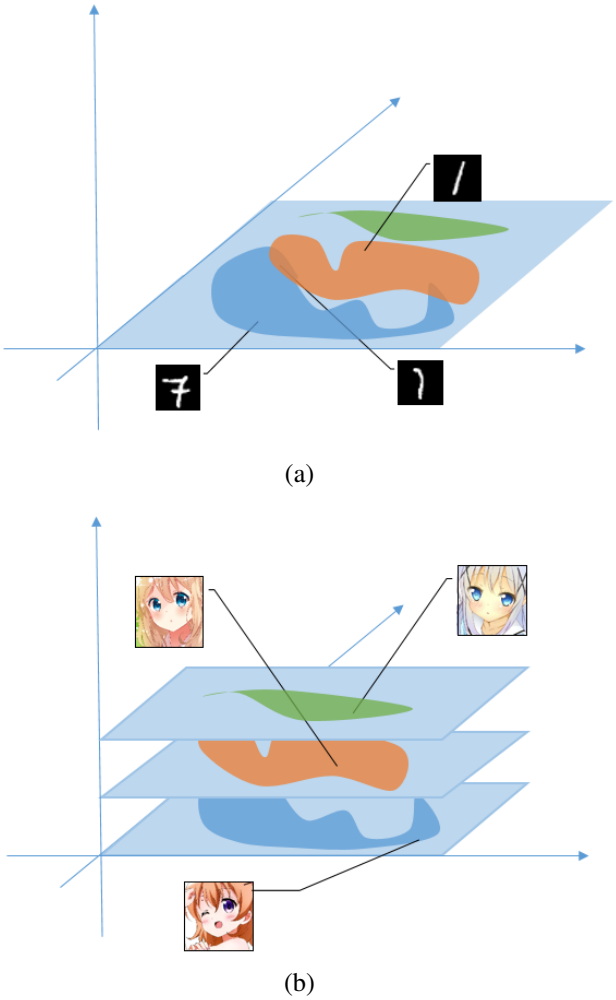
For the generator block, we have two options: we can integrate the condition to the input of $\mathcal{M}$ or on those of $\mathcal{G}$. The two options are possible as it depends mainly on the kind of dataset that we use.

$$x \sim \mathcal{N}(0,1) \quad ; \quad y = \begin{cases} \mathcal{G}(\mathcal{M}(x)^\frown l(c)) & [I] \\ \\ \mathcal{G}(\mathcal{M}(x^\frown l(c))) & [II] \end{cases} \tag{5}$$

With equation 4.I, the pictures of the same class will share the n last elements of their latent vector. Thus, the latent manifold will be 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 1.b.

However, some dataset does not have this property. We can cite the TenGeoP-SARwv [7] where some pictures could describe two different phenomena. Thus, one could want to avoid the sub-manifolds being disjoint (figure 1.a).

In our implementation, we leave this choice to the user.



(a)

(b)

Fig. 1. Illustration of the latent manifold if we consider the classes to overlap (a) or to be strictly disjoint (b).

4. RESULTS

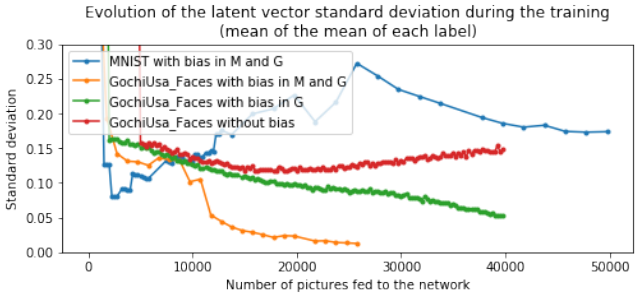
It is difficult to evaluate the quality of a GAN. However, some metrics can be used, in addition with a visual inspection, to assert that images are correctly generated.

An indicator of a good training is the diversity in the outputs. During our experiments, we saw that, in some configuration, the model was collapsing. The mode collapse [8] refers to an issue occurring when the model output is constant with any inputs. To illustrate this issue, we compute the standard deviation of both the output of $\mathcal{M}$ and $\mathcal{G}$. Mode collapse implies a convergence to 0 of the standard deviation of y, the output picture. However, if the mapper is subject to mode collapse, the standard deviation of the latent vector will also converge to 0.

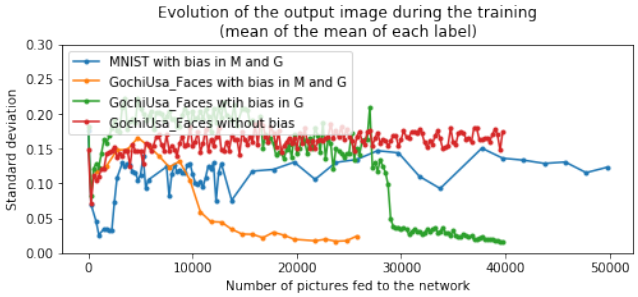
Since small translations of the output image can cause high variations of the standard deviation, this only indicator is the convergence to 0, as it illustrates that the same vector is output each time. It is not possible to draw conclusions about any other values, especially for the output of $\mathcal{G}$.

The figure 2 illustrates that the model isn't affected by mode collapse when working with the dataset MNIST [9] even if the model use biases in $\mathcal{M}$ and $\mathcal{G}$. However, with the GochiUsa_Faces [**?**], using bias either in the mapper or the generator induce 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. If the biases are too high and the other weights too low, the output will be almost equal to the biases and constant with the input. Thus, forcing the bias to zero prevent the mode collapse.

Another point is that the GAN is correctly generating samples for each class. This can be checked by running a classifier on the output of $\mathcal{G}$. The figure 3 illustrates that the conditioning appear very quickly on datasets with colour-



(a)



(b)

Fig. 2. Standard deviation of the latent vector (a) and the output image (b) through the training.

based information (GochiUsa_Faces) but also on shape-based datasets (MNIST). For each of these two datasets, three models were trained without bias to decrease the risk of a statistical hazard.

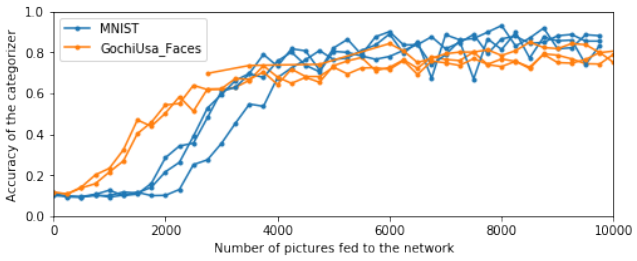


Fig. 3. Performance of a classifier on the generated samples during the training

Finally, the figure 4 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, we are able to get a lot of different position and facial expressions. The model was trained for twelve hours on the GPU provided by Google Collab.

5. CONCLUSION

We publish a conditional implementation of the StyleGAN2 architecture. Modifications are applied both on the generator and the discriminator of the GAN, however, a place for implementation choices is left for implementation choice as the characteristics of the dataset could lead to a preference as to whether implement the conditioning in the mapper input or the output.

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

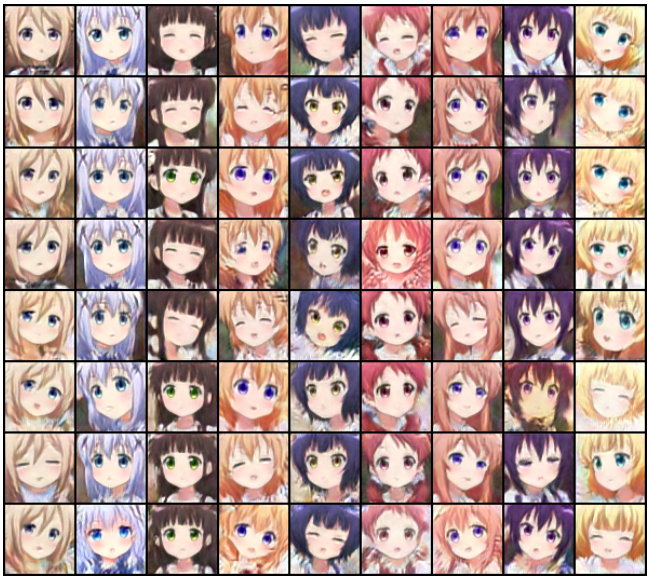
[5] Tero Karras, Samuli Laine, and Timo Aila, "A style-based generator architecture for generative adversarial networks," *CoRR*, vol. abs/1812.04948, 2018.

[6] 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.

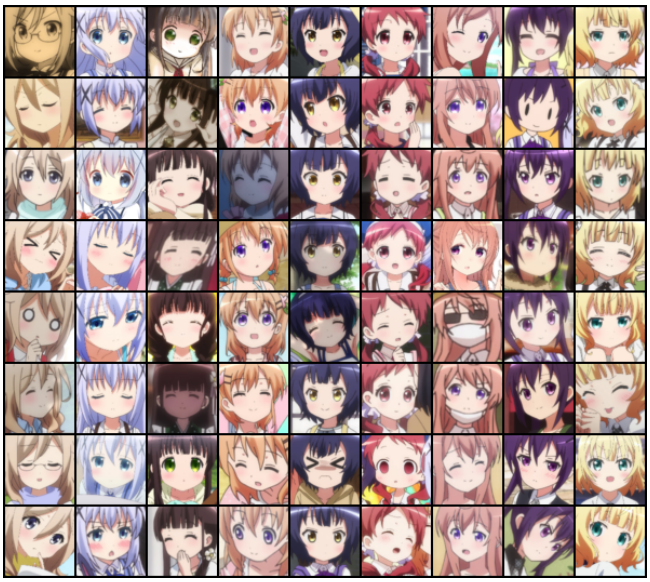
[7] Chen Wang, Alexis Mouche, Pierre Tandeo, Justin E. Stopa, Nicolas Longépé, Guillaume Erhard, Ralph C. Foster, Douglas Vandemark, and Bertrand Chapron, "A labelled ocean sar imagery dataset of ten geophysical phenomena from sentinel-1 wave mode," *Geoscience Data Journal*, vol. 6, no. 2, pp. 105–115, 2019.

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

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



(a)



(b)

Fig. 4. Samples generated by the cStyleGAN (a) and the groundtruth dataset (b)