cDCGAN: conditional Deep Convolutional Generative Adversarial Networks

Edoardo De Matteis
La Sapienza University of Rome
dematteis.1746561@studenti.uniroma1.it

Abstract

GANs have been described as one of the most interesting idea in the last years of machine learning, and have been integrated with CNNs to have strong priors on images. Conditional GANs allow us to condition generation through supplementary input, in this project. I combine both these models in a conditional deep convolutional GAN, code is available at https://github.com/edodema/cDCGAN.

1. Introduction

Since they were first introduced in 2014 GANs have changed the game [1], to the point that Yann LeCun considers them the most interesting idea in the last 10 years of machine learning [4]. Their ability to generate realistic images has amazed both specialists and the general publi, e.g. the website *This Person Does Not Exist* ()https://thispersondoesnotexist.com/) based on *StyleGAN2* [8]. StyleGAN2 has been recently overcome by *StyleGAN3* [2], both were developed by *NVIDIA* with the latter being trained for a complexive time of 96 years[1]. That casts a light upon one big issue: GANs need are very hard to train and rely a lot on GPUs.

2. Related works

VAE We call autoencoder (AE) any model that outputs a reconstruction of the input, they are tipically defined by an encoder-decoder architecture, we call the bottleneck *latent representation*. The problem with AEs is that they can map similar inputs to arbitarily distant regions of the latent space, but we want to learn a distribution. A *variational autoencoder* (VAE) [3] constructs the parameters of a probability distribution on the latent space, with the distribution being fixed a priori (often a gaussian) and training data being seen as a sampling of the learned distribution.

GAN The idea behind *generative adversarial network* [1] is based on VAEs: we can split the architecture and call the encoder **discriminator** and the decoder **generator**. Yet they differ from a VAE: the decoder's output is a probability of its input being real, way different from generator's input i.e. a sample from a probability distribution (often Gaussian). Discriminator and generator play a zero-sum game where the generator aims to fool the discriminator creating fake data, while the discriminator want to catch the generator.

DCGAN GANs were introduced based on *multi-layer perceptrons* only, it came natural to extend them with CNNs to exploit priors on images. In [6] are laid down some guidelines for a more stable convolutional GAN:

- Replace deterministic spatial pooling with strided convolutions, allowing the network to learn its own downsampling or upsampling functions.
- Remove fully connected layers on top of CNNs (e.g. classification heads) in favor of deeper architectures, preferring one-dimensional convolutions and vector transformations.
- Employ batch normalization in both the discriminator and the generator, being careful that too many batch norms can make the latter unstable.
- Use ReLU activations in the generator for all layers except for the output, in which they use a $\tanh$ activation. In the discriminator LeakyReLU is preferred.

cGAN Until down GANs generate data with no concept at all of classes. In [5] is shown a model that receives in input some conditioning along with data, in this specific case this conditioning is a one hot vector representing some class but can potentially be anything.

2.1. Task and goals

In this work I aim to combine a DCGAN based structure with some conditioning, thus creating a cDCGAN. Class conditional synthesis can significantly improve the quality of generated samples [7], and such model could be used

[1] It was trained on multiple GPUs at a time.

for specialized data enrichment or could have applications in entertainment (e.g. *FaceApp*). Its development could also lead to advancements in other branches of machine learning e.g. injecting natural language knowledge as conditional data.

3. Methodology

3.1. Dataset and preprocessing

The model has learned on three datasets: *MNIST*, *FashionMNIST* and *CIFAR10*, data are z-score normalized with specific mean and standard deviation previously computed on each dataset. The reason behind that is that is easier to learn on data with 0 mean and 1 standard deviation. Images are reshaped to be 64×64 by default, this way we can test the same model on each dataset and while in general we can learn more on high resolution data it could also be that upsampling procedures add noise.

Conditioning is embedded as a one-hot vector, manipulated such that it could be added as a 64×64 matrix to the image through concatenation as an additional channel. Noise samples have a size of 64 as well.

3.2. Model

Discriminator The discriminator is based on a basic block composed of a convolution, a batch normalization and a LeakyReLU activation function whose negative slope equals to 0.2. The discriminator model is defined as:

1. Concatenation of input image and its conditioning on the channels dimension.
2. Convolution with LeakyReLU activation.
3. Basic block define above, repeated three times.
4. Convolution + sigmoid function.

The model is shown in figure 1, implementation details can be seen on the official repo.

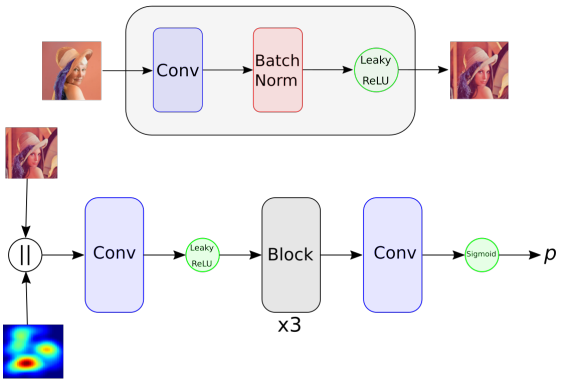

Figure 1. Discriminator scheme.

Generator Generator's basic block is similiar to discriminator's one, yet instead of a convolution we employ transpose convolution to upsample, and we use ReLU as activation function. The model is in a way simpler than the previous one:

1. A sampled noise matrix is concatenated to its conditioning.
2. We repeat generator's basic block 4 times.
3. The model ends with a transpose convolution and a $\tanh$ activation function.

The model is shown in figure 2, code is available on the code linked in the abstract.

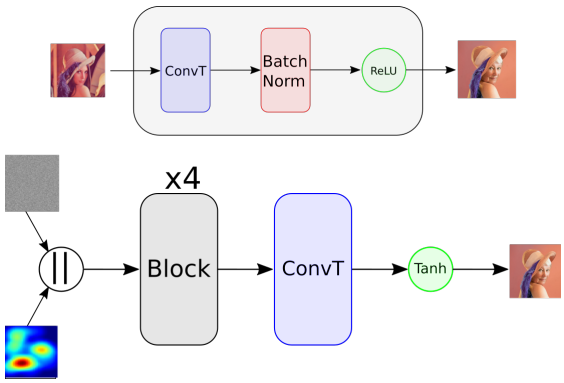

Figure 2. Generator scheme.

3.3. Loss

In their seminal paper [1], GAN's loss is defined as a minmax loss (1), where p_d and p_z represent the distribution over real and fake data respectively. This is quite incovenient and in the same paper it is approximated to a easier loss.

$$\min_{\gamma} \max_{\delta} \mathbb{E}_{p_d} \log D(x) + \mathbb{E}_{p_z} \log(1 - D(G(z))) \quad (1)$$

Another approach consists in using a classical binary cross entropy loss, one each for discriminator and generator. Discriminator's ground truth is a $\vec{0}$ for fake data and a $\vec{1}$ vector for real data, the generator's has a $\vec{1}$ too. The generator would maximize on fake data i.e. $\vec{0}$ yet it is easier to minimize thus we flip zeros with ones.

4. Experiments and results

For the MNIST and FashionMNIST datasets the network has been trained for 5 and 10 epochs respectively with a learning rate equal to 0.0002 and results are overall good. The same cannot be said for CIFAR10, trained for 25 epochs with 0.0002 and 0.00002 learning rates for

discriminator and generator respectively, while we can see some familiar and sensible shapes images are not realistic at all. This makes sense since CIFAR10 scenes are way more complicated than MNIST and FashionMNIST, also having 3 channels instead of 1 could make things harder for the model.

Weighting Weighting the conditioning can give us interesting results, on the MNIST and FashionMNIST it enhances contrast and since images are black and white it means clearer outputs.

Multiclass conditioning Conditioning does not necessarily have to be a one hot vector, if more indices are equal to 1 the generator will morph images from more classes. We can assing values different from one, even decimal, depending on what we want to create and with which strength.

Figure 3. MNIST generated images.

Figure 4. FashionMNIST generated images.

Figure 5. CIFAR10 generated images.

5. Conclusions

I developed a conditional GAN that exploits convolutions, considering the relatively short training time it is quite good on simple datasets. If the datasets gets a little more complicated results do not stand out, a plausible reason is that training GANs, other than being notoriously hard, requires large computations.

Future works For this project I only had the free GPUs offered by *Google Colab*, it would be interesting to see how the model does when trained properly. Another interesting approach could be to plug in classification pretrained models as a feature extraction backbone and fine tune them.

References

[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*, 27, 2014.

[2] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten,

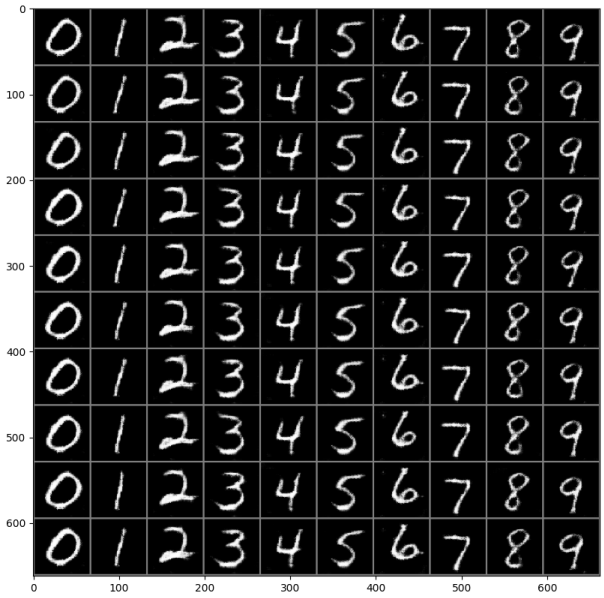

Figure 6. MNIST predictions with weighted conditioning vector.

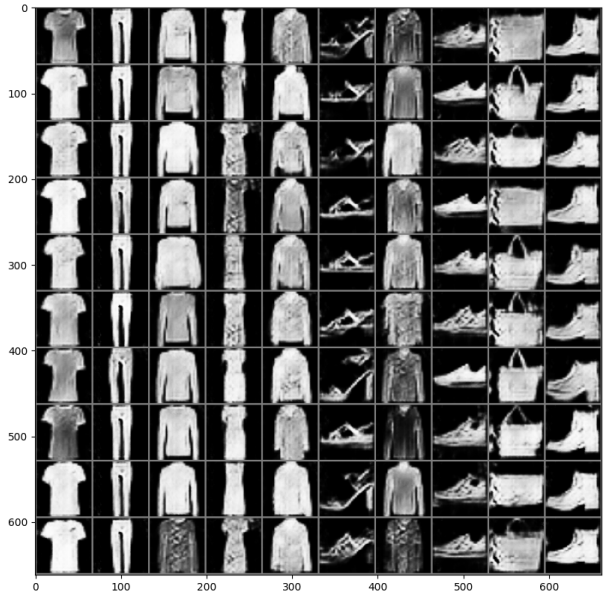

Figure 7. FashionMNIST predictions with weighted conditioning vector.

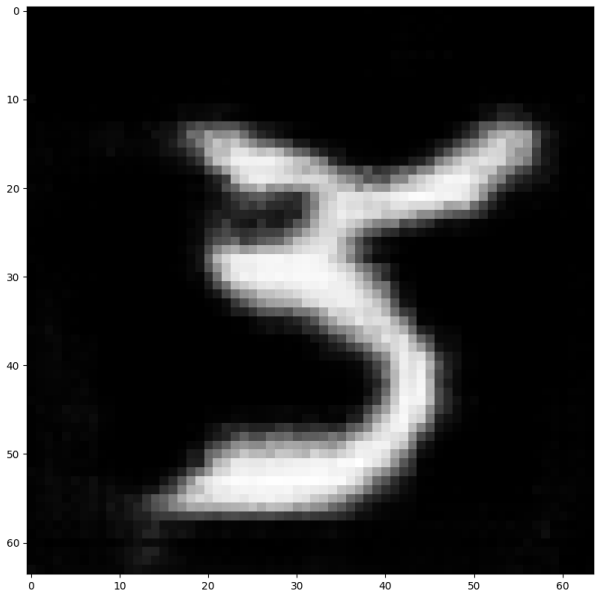

Figure 8. MNIST prediction of both 3 and 5.

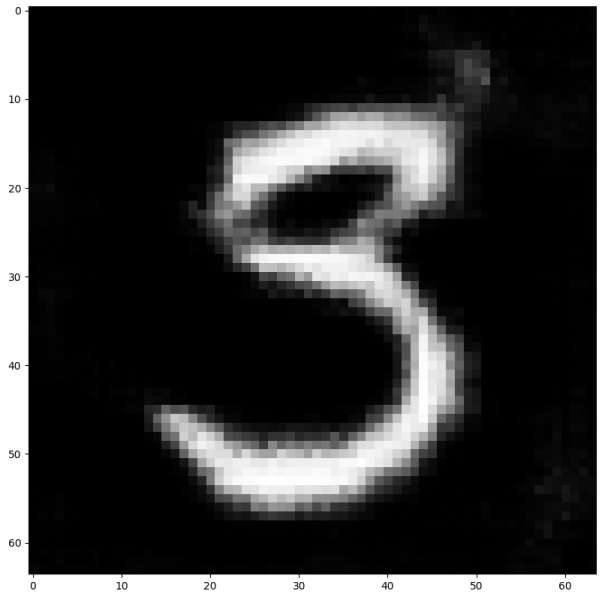

Figure 9. MNIST prediction of both 3 and 5, with 3 being weighted more.

J. Lehtinen, and T. Aila. Alias-free generative adversarial networks. *CoRR*, abs/2106.12423, 2021.

[3] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] Y. LeCun. Ri seminar: Yann lecun : The next frontier in ai: Unsupervised learning, 2016.

[5] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

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

[7] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016.

[8] Y. Viazovetskyi, V. Ivashkin, and E. Kashin. Stylegan2 distillation for feed-forward image manipulation. In *European Conference on Computer Vision*, pages 170–186. Springer, 2020.

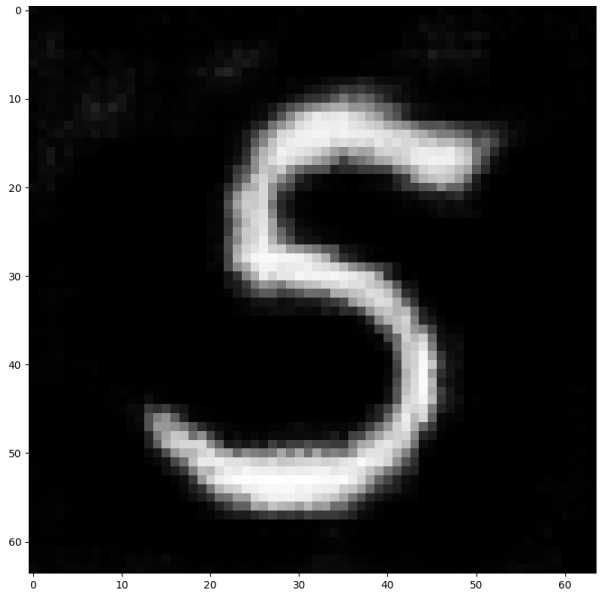

Figure 10. MNIST prediction of both 3 and 5, with 5 being weighted more.

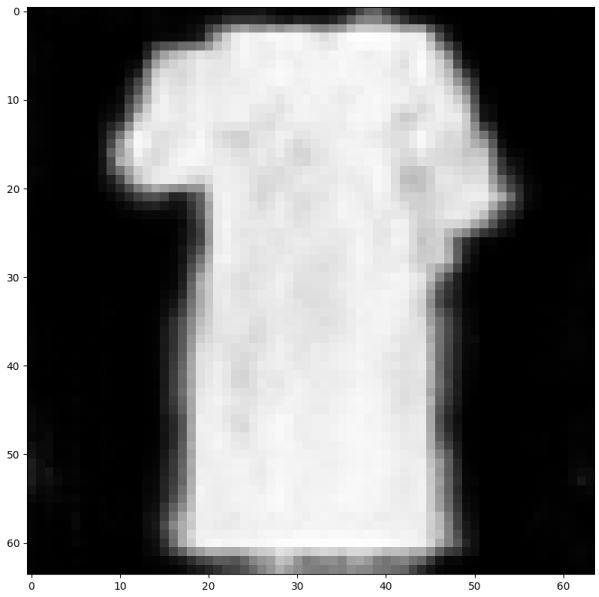

Figure 11. FashionMNIST prediction of a shirt.

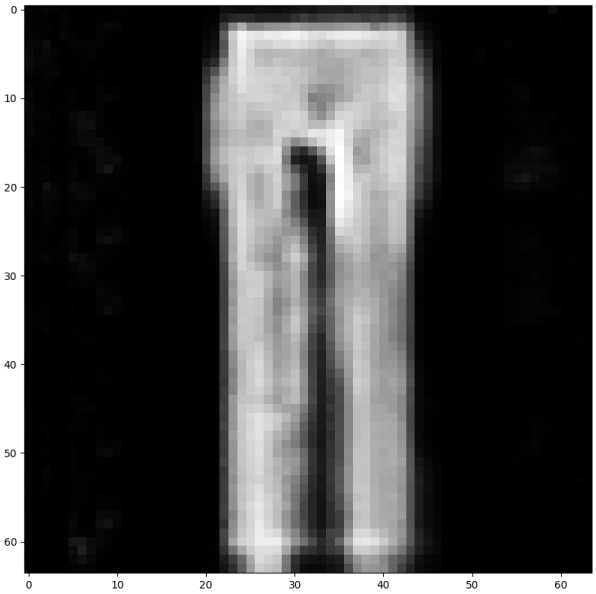

Figure 12. FashionMNIST prediction of a pair of pants.

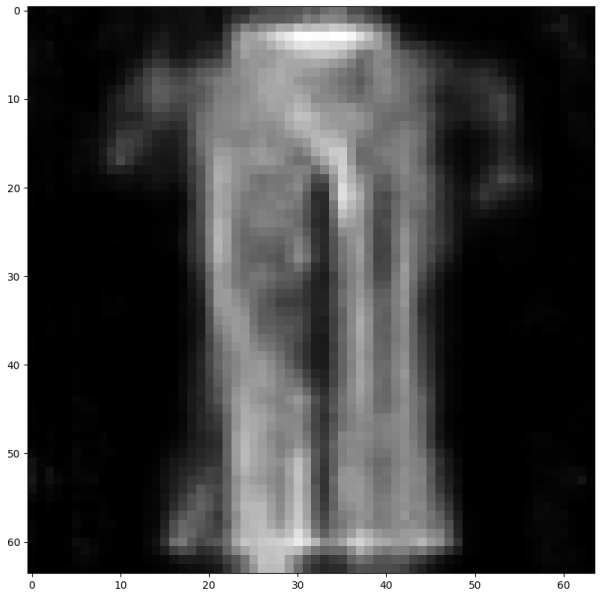

Figure 13. FashionMNIST prediction of both shirt and pants, in this example pants are weighted 0.7 otherwise they tend to hog the scene.