Deep Convolutional Generative Adversarial Networks

Théo Viel

15 April 2020

Summary

Summary

GAN Principle

- Generator $\mathcal{G}$: Generates images of good quality starting from a noise vector z
- Discriminator $\mathcal{D}$: Distinguishes between real and fake images

The GAN objective

$$\min_{G} \max_{D} \quad \mathbb{E}_{x \sim p_{\text{data}}}[\log \mathcal{D}(x)] \; + \; \mathbb{E}_{z \sim p_z}[\log (1 - \mathcal{D}(\mathcal{G}(z)))]$$

DCGAN Ideas

- Replace fully connected layers : Convolutions of kernel size 1
- Replace pooling layers : Strided convolutions
- Upsampling : Transposed convolutions
- Batch Normalization
- Rectified Linear Units activations

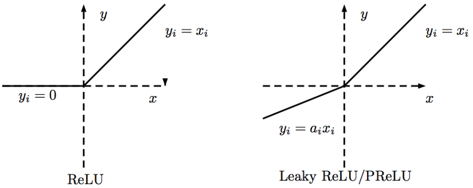


Figure – ReLU [4] and LeakyReLU [2] activations.

Data

- 20,580 dog images from ImageNet [1]
- 64×64 image size
- High variety, much harder than generating faces

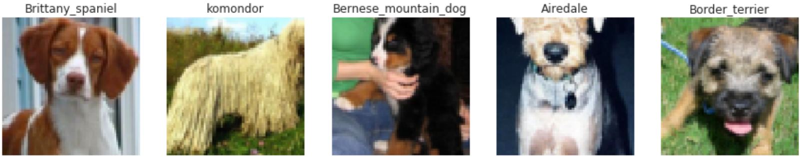


Figure – Pre-processed training data samples.

Fréchet Inception Distance (FID)

Fréchet Distance

For two multivariate Gaussians $X_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$, $X_2 \sim \mathcal{N}(\mu_2, \Sigma_2)$:

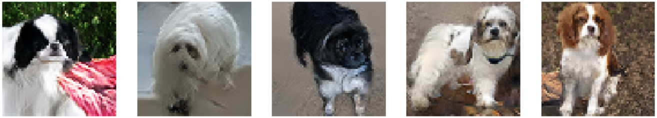
$$d^2(X_1, X_2) = \|\mu_1 - \mu_2\|^2 + \mathrm{Tr}(\Sigma_1 + \Sigma_2 - 2\sqrt{\Sigma_1 \Sigma_2}).$$

- Metric to evaluate GANs
- Use an hidden layer of the Inception [7] network :
  - X_1 = activations on the real data
  - X_2 = activations on the fake data
- Slow to compute and not a perfect indicator of performance

FID Visualisations

(a) DCGAN, FID = 56

(b) DCGAN, FID = 79



(c) BigGAN, FID = 13

Figure – Example of generated dogs for models with different FIDs

Summary

DCGAN Discriminator

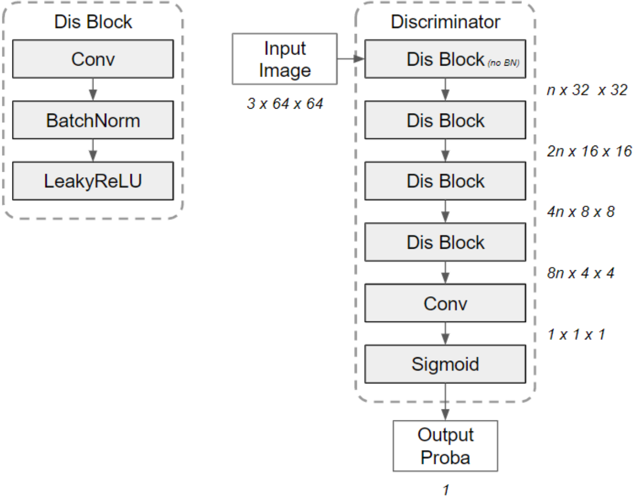


Figure – Architecture of the DCGAN discriminator.

DCGAN Generator

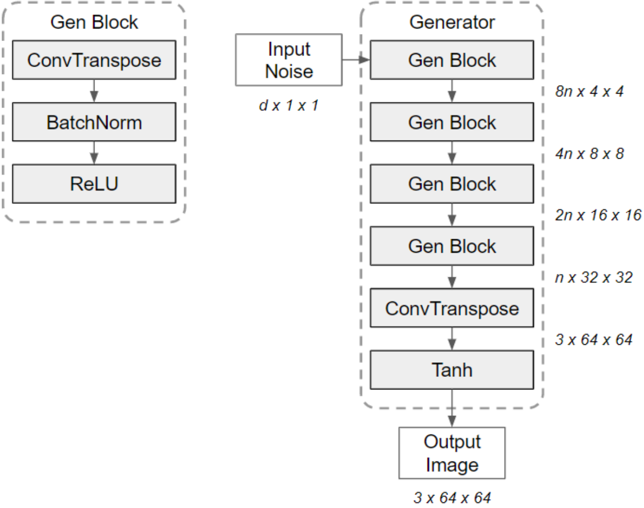


Figure – Architecture of the DCGAN generator.

Algorithm 1 DCGAN training

for $\mathcal{I}_{\text{real}}$ in batch(data) **do**
  # Update discriminator
  $p_{\text{real}} = \mathcal{D}(\mathcal{I}_{\text{real}})$
  $\text{Loss}_{\text{real}} = \mathrm{BCE}(p_{\text{real}}, 1)$
  sample $z \sim \mathcal{N}(0, 1)$
  $\mathcal{I}_{\text{fake}} = \mathcal{G}(z)$
  $p_{\text{fake}} = \mathcal{D}(\mathcal{I}_{\text{fake}})$
  $\text{Loss}_{\text{fake}} = \mathrm{BCE}(p_{\text{fake}}, 0)$
  Back-propagate $\text{Loss}_{\text{fake}} + \text{Loss}_{\text{real}}$

  # Update generator
  sample $z \sim \mathcal{N}(0, 1)$
  $\mathcal{I}_{\text{fake}} = \mathcal{G}(z)$
  $p_{\text{fake}} = \mathcal{D}(\mathcal{I}_{\text{fake}})$
  $\text{Loss} = \mathrm{BCE}(p_{\text{fake}}, 1)$
  Back-propagate Loss
end for=0

First Experiments

Table – Influence of the learning rate and batch size.

Batch size	Discriminator lr	Generator lr	**FID**
128	2×10^{-4}	2×10^{-4}	67.9
64	1×10^{-3}	1×10^{-3}	60.2
64	5×10^{-4}	5×10^{-4}	66
64	5×10^{-4}	1×10^{-3}	**56.5**

Table – Influence of the latent dimension d and model size n

d	n	epoch length	**FID**
100	128	44s	**56.5**
256	128	44s	62.2
100	64	11s	78

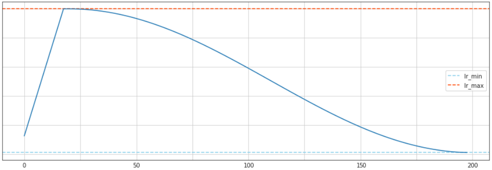
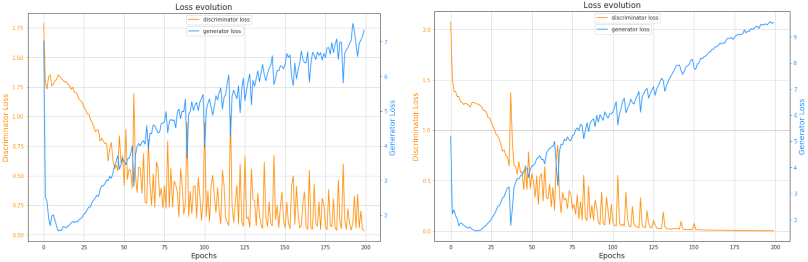


Figure – Cosine learning rate scheduling.

Table – Adding a learning rate scheduler

Discriminator lr	Generator lr	FID
$5 \times 10^{-4} \rightarrow 1 \times 10^{-5}$	$1 \times 10^{-3} \rightarrow 1 \times 10^{-5}$	58
$1 \times 10^{-3} \rightarrow 1 \times 10^{-5}$	$2 \times 10^{-3} \rightarrow 1 \times 10^{-5}$	**55.5**
$1 \times 10^{-3} \rightarrow 1 \times 10^{-4}$	$2 \times 10^{-3} \rightarrow 2 \times 10^{-4}$	57.8
$2 \times 10^{-3} \rightarrow 2 \times 10^{-5}$	$4 \times 10^{-3} \rightarrow 4 \times 10^{-5}$	58.7



(a) Without Scheduling (b) With Scheduling

Figure – Influence of adding a cosine learning rate scheduling on the generator loss (blue) and discriminator loss (orange).

Summary

Feature Matching Idea

- One of the ideas of [6]
- New objective for the generator
- Generator trained to generate images that has features matching those of the real data
- Use the discriminator to extract features, noted $f_{\mathcal{D}}$

Feature Matching objective

$$\min_{G} \quad \|\mathbb{E}_{z\sim p_z}[f_{\mathcal{D}}(\mathcal{G}(z))] - \mathbb{E}_{x\sim p_{\text{data}}}[f_{\mathcal{D}}(x)]\|_2^2$$

Feature Matching Algorithm

Algorithm 2 DCGAN training with Feature Matching

for $\mathcal{I}_{\text{real}}$ in batch(data) **do**
  # Update discriminator
  [...] (Same as algo. 1)

  # Update generator
  sample $z \sim \mathcal{N}(0, 1)$
  $\mathcal{I}_{\text{fake}} = \mathcal{G}(z)$
  $x_{\text{fake}} = f_{\mathcal{D}}(\mathcal{I}_{\text{fake}})$
  $x_{\text{real}} = f_{\mathcal{D}}(\mathcal{I}_{\text{real}})$
  Average x_{fake} and x_{real} over the batch axis
  Loss $= \mathrm{MSE}(\bar{x}_{\text{fake}}, \bar{x}_{\text{real}})$
  Back-propagate Loss
end for=0

DCGAN Discriminator with Feature Matching

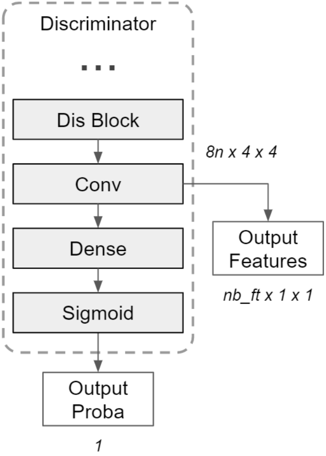


Figure – Adding feature matching to the DCGAN discriminator.

DCGAN with Feature Matching Results

Table – Influence of adding feature matching to our DCGAN.

Feature Matching	**FID**
Without	**55.5**
With	55.9

Summary

Conditional GAN Idea

- Introduced in [3]
- Use class knowledge in the GAN training

Conditional GAN objective

$$\min_G \max_D \quad \mathbb{E}_{x \sim p_{\text{data}}}[\log \mathcal{D}(x|y)] \; + \; \mathbb{E}_{z \sim p_z}[\log (1 - \mathcal{D}(\mathcal{G}(z|y)))]$$

Conditional GAN Algorithm

Algorithm 3 Conditional GAN training

for $\mathcal{I}_{\text{real}}, y_{\text{real}}$ in batch(data) **do**

Update discriminator

$p_{\text{real}} = \mathcal{D}(\mathcal{I}_{\text{real}}, y_{\text{real}})$

$\text{Loss}_{\text{real}} = \mathrm{BCE}(p_{\text{real}},\ 1)$

sample $z \sim \mathcal{N}(0, 1)$

sample $y_{\text{fake}} \sim \mathcal{U}([\![0, c]\!])$

$\mathcal{I}_{\text{fake}} = \mathcal{G}(z, y_{\text{fake}})$

$p_{\text{fake}} = \mathcal{D}(\mathcal{I}_{\text{fake}}, y_{\text{fake}})$

$\text{Loss}_{\text{fake}} = \mathrm{BCE}(p_{\text{fake}}, 0)$

Back-propagate $\text{Loss}_{\text{fake}} + \text{Loss}_{\text{real}}$

Update generator

sample $z \sim \mathcal{N}(0, 1)$

sample $y_{\text{fake}} \sim \mathcal{U}([\![0, c]\!])$

$\mathcal{I}_{\text{fake}} = \mathcal{G}(z, y_{\text{fake}})$

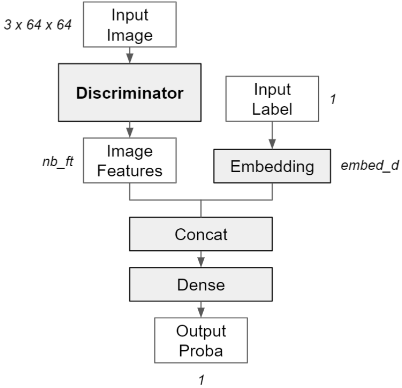
$p_{\text{fake}} = \mathcal{D}(\mathcal{I}_{\text{fake}}, y_{\text{fake}})$

$\text{Loss} = \mathrm{BCE}(p_{\text{fake}}, 1)$

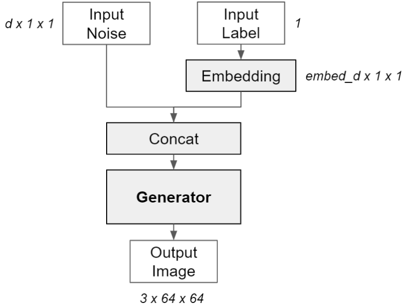
Back-propagate Loss

end for=0

Conditional DCGAN Implementation



(a) Conditional Discriminator



(b) Conditional Generator

Figure – Adding conditioning to the DCGAN architecture.

Projection Discriminator

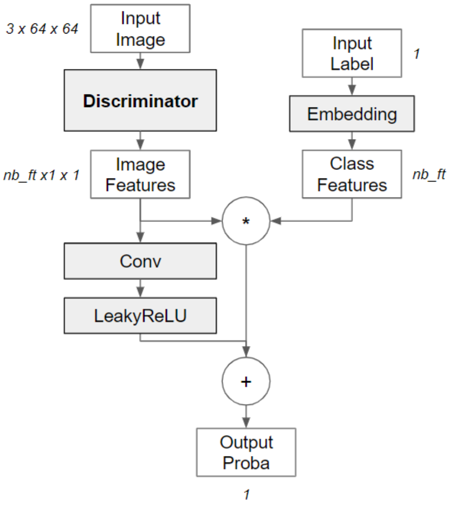


Figure – Architecture of the projection discriminator.

Conditional DCGAN Results

Table – Influence of adding conditioning to our DCGAN.

Conditioning	*nb_ft*	**FID**
Without		**55.5**
With	8	60.3
Projected	1024	62.8

Summary

Auxiliary Classifier GAN Idea

- Introduced in [5]
- Same generator as for the Conditional GAN
- Use the discriminator to predict the class label, noted $\mathcal{D}_c$
- Combination of the two following objectives :

Auxiliary Classifier GAN objectives

$$\min_G \max_D \quad \mathbb{E}_{x\sim p_{\text{data}}}[\log \mathcal{D}(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - \mathcal{D}(\mathcal{G}(z|y)))]$$

$$\max_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log \mathcal{D}_c(x) = y] + \mathbb{E}_{z\sim p_z}[\log \mathcal{D}_c(\mathcal{G}(z|y)) = y]$$

Algorithm 4 Auxiliary Classifier DCGAN training

for $\mathcal{I}_{\text{real}}, y_{\text{real}}$ in batch(data) **do**
  # Update discriminator
  $p_{\text{real}}, p^c_{\text{real}} = \mathcal{D}(\mathcal{I}_{\text{real}})$
  $\text{Loss}_{\text{real}} = \mathrm{BCE}(p_{\text{real}}, 1) + \mathrm{CE}(p^c_{\text{real}}, y_{\text{real}})$
  sample $z \sim \mathcal{N}(0, 1)$
  sample $y_{\text{fake}} \sim \mathcal{U}([\![0, c]\!])$
  $\mathcal{I}_{\text{fake}} = \mathcal{G}(z, y_{\text{fake}})$
  $p_{\text{fake}}, p^c_{\text{fake}} = \mathcal{D}(\mathcal{I}_{\text{fake}})$
  $\text{Loss}_{\text{fake}} = \mathrm{BCE}(p_{\text{fake}}, 0) + \mathrm{CE}(p^c_{\text{fake}}, y_{\text{fake}})$
  Back-propagate $\text{Loss}_{\text{fake}} + \text{Loss}_{\text{real}}$

  # Update generator
  sample $z \sim \mathcal{N}(0, 1)$
  sample $y_{\text{fake}} \sim \mathcal{U}([\![0, c]\!])$
  $\mathcal{I}_{\text{fake}} = \mathcal{G}(z, y_{\text{fake}})$
  $p_{\text{fake}}, p^c_{\text{fake}} = \mathcal{D}(\mathcal{I}_{\text{fake}}$
  $\text{Loss} = \mathrm{BCE}(p_{\text{fake}}, 1) + \mathrm{CE}(p^c_{\text{fake}}, y_{\text{fake}})$
  Back-propagate Loss
end for=0

DCGAN Discriminator with an Auxiliary Classifier

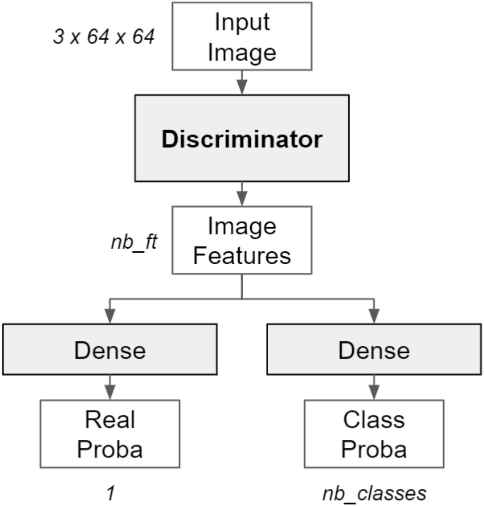


Figure – Adding an auxiliary classifier to the DCGAN discriminator.

Auxiliary Classifier DCGAN Results

Table – Influence of adding an auxiliary classifier to our DCGAN.

Auxiliary classifier	**FID**
Without	**55.5**
With	57.9

Summary

Training Evolution

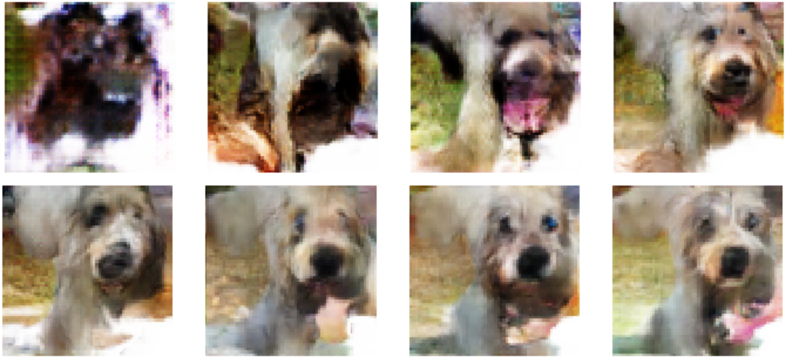


Figure – Evolution of the output for a fixed noise during the training.

Best Generated Dogs

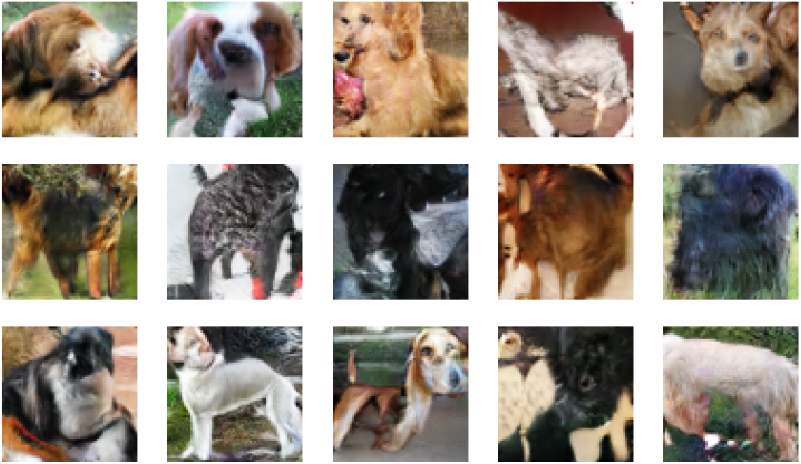


Figure – Generated dogs for the best model, achieving an FID of 55.5.

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei.
ImageNet : A Large-Scale Hierarchical Image Database.
In *CVPR09*, 2009.

[2] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng.
Rectifier nonlinearities improve neural network acoustic models.
In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[3] Mehdi Mirza and Simon Osindero.
Conditional generative adversarial nets.
CoRR, abs/1411.1784, 2014.

[4] Vinod Nair and Geoffrey E. Hinton.
Rectified linear units improve restricted boltzmann machines.
In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress.

[5] Augustus Odena, Christopher Olah, and Jonathon Shlens.

Conditional image synthesis with auxiliary classifier GANs.

In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2642–2651, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[6] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen.

Improved techniques for training gans.

In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich.

Going deeper with convolutions.

CoRR, abs/1409.4842, 2014.