Deep Learning in Data Science

Harald Stiff   Erik von Keyserlingk   Jacob Stuart

Image Generation Using Deep Convolutional Generative Adversarial Networks

Outline

Outline

Problem Formulation

- Generate images resembles the training set
- Generate images that are diverse and covers the features of the whole training set
- Investigate the mapping from latent space to generated image space

Main inspiration

Our work and models are based on (Radford et al., 2015)
Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks

- Simple guidlines for good results
- Stability during training
- Accepted as state of art for GAN architecture designs

Tools and resources

Tools used:

- Python3
- Tensorflow
- Keras
- Numpy
- Google computation was used for running the code

Outline

Gans

Composed of two models:

$$\underbrace{G(\boldsymbol{z}, \Theta_g)}_{\text{Generator}} \qquad \& \qquad \underbrace{D(\boldsymbol{x}, \Theta_d)}_{\text{Discriminator}}$$

where

$$\underbrace{\Theta_g = \{\theta_g^i\}_{i=1}^n}_{\text{Generator parameters}} \qquad \& \qquad \underbrace{\Theta_d = \{\theta_d^j\}_{j=1}^m}_{\text{Discriminator parameters}}$$

Gans

Composed of two models:

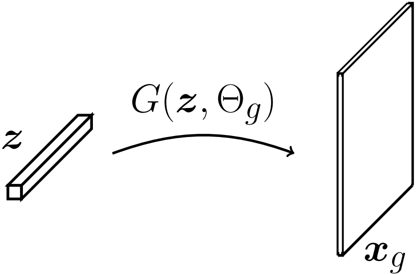
$$\underbrace{G(\boldsymbol{z}, \Theta_g)}_{\text{Generator}} \qquad \& \qquad \underbrace{D(\boldsymbol{x}, \Theta_d)}_{\text{Discriminator}}$$

where

$$\underbrace{\Theta_g = \{\theta_g^i\}_{i=1}^n}_{\text{Generator parameters}} \qquad \& \qquad \underbrace{\Theta_d = \{\theta_d^j\}_{j=1}^m}_{\text{Discriminator parameters}}$$

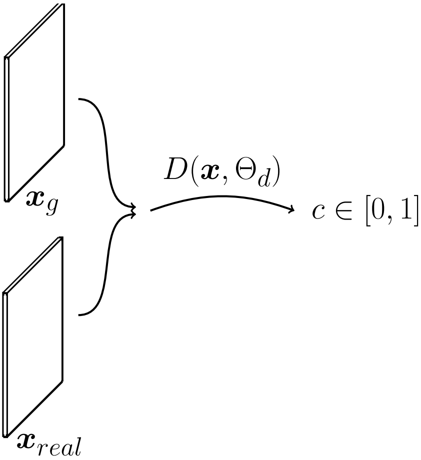
Gans

Generator: Maps an input vector $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$ to a tensor $\boldsymbol{x}_g$.



Gans

Discriminator: Maps tensors $\boldsymbol{x}_{real} \sim p_{data}(\boldsymbol{x})$ and $\boldsymbol{x}_g$ to a scalar score c.



Discriminator update:

$$\underbrace{\mathcal{B}_d}_{\textbf{Batch}} = \left\{ (\boldsymbol{x}^i_{real}, y = 1), (\boldsymbol{x}^i_g, y = 0) \right\}_{i=1}^{N/2}$$

$$\underbrace{\mathcal{L}(\mathcal{B}_d, \Theta_d)}_{\textbf{Loss}} = -\frac{1}{|\mathcal{B}_d|} \sum_{(\boldsymbol{x},y)\in\mathcal{B}_d} y \log(D(\boldsymbol{x})) + (1-y)\log(1 - D(\boldsymbol{x}))$$

$$\underbrace{\frac{\partial \mathcal{L}(\mathcal{B}_d, \Theta_d)}{\partial \theta}}_{\textbf{Gradients}}, \quad \forall \theta \in \Theta_d$$

Discriminator update:

$$\underbrace{\mathcal{B}_d}_{\textbf{Batch}} = \left\{ (\boldsymbol{x}^i_{real}, y = 1), (\boldsymbol{x}^i_g, y = 0) \right\}_{i=1}^{N/2}$$

$$\underbrace{\mathcal{L}(\mathcal{B}_d, \Theta_d)}_{\textbf{Loss}} = -\frac{1}{|\mathcal{B}_d|} \sum_{(\boldsymbol{x}, y) \in \mathcal{B}_d} y \log(D(\boldsymbol{x})) + (1 - y) \log(1 - D(\boldsymbol{x}))$$

$$\underbrace{\frac{\partial \mathcal{L}(\mathcal{B}_d, \Theta_d)}{\partial \theta}}_{\textbf{Gradients}}, \quad \forall \theta \in \Theta_d$$

Discriminator update:

$$\underbrace{\mathcal{B}_d}_{\textbf{Batch}} = \Big\{ (\boldsymbol{x}^i_{real}, y = 1), (\boldsymbol{x}^i_g, y = 0) \Big\}_{i=1}^{N/2}$$

$$\underbrace{\mathcal{L}(\mathcal{B}_d, \Theta_d)}_{\textbf{Loss}} = -\frac{1}{|\mathcal{B}_d|} \sum_{(\boldsymbol{x},y)\in\mathcal{B}_d} y \log(D(\boldsymbol{x})) + (1-y)\log(1-D(\boldsymbol{x}))$$

$$\underbrace{\frac{\partial \mathcal{L}(\mathcal{B}_d, \Theta_d)}{\partial \theta}}_{\textbf{Gradients}}, \qquad \forall \theta \in \Theta_d$$

Generator update:

$$\underbrace{\mathcal{B}_g}_{\textbf{Batch}} = \left\{ (\boldsymbol{z}_i, y = 1) \right\}_{i=1}^{N}$$

$$\underbrace{\mathcal{L}(\mathcal{B}_g, \Theta_d, \Theta_g)}_{\textbf{Loss}} = -\frac{1}{|\mathcal{B}_g|} \sum_{(\boldsymbol{z}, y) \in \mathcal{B}_g} y \log D(G(\boldsymbol{z}))$$

$$\underbrace{\frac{\partial \mathcal{L}(\mathcal{B}_g, \Theta_d, \Theta_g)}{\partial \theta}}_{\textbf{Gradients}}, \quad \forall \theta \in \Theta_g$$

Generator update:

$$\underbrace{\mathcal{B}_g}_{\textbf{Batch}} = \left\{ (\boldsymbol{z}_i, y = 1) \right\}_{i=1}^{N}$$

$$\underbrace{\mathcal{L}(\mathcal{B}_g, \Theta_d, \Theta_g)}_{\textbf{Loss}} = -\frac{1}{|\mathcal{B}_g|} \sum_{(\boldsymbol{z}, y) \in \mathcal{B}_g} y \log D(G(\boldsymbol{z}))$$

$$\underbrace{\frac{\partial \mathcal{L}(\mathcal{B}_g, \Theta_d, \Theta_g)}{\partial \theta}}_{\textbf{Gradients}}, \quad \forall \theta \in \Theta_g$$

Gans

Generator update:

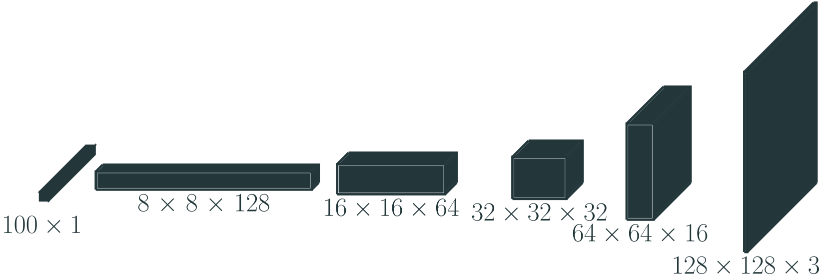
$$\underbrace{\mathcal{B}_g}_{\textbf{Batch}} = \left\{ (\boldsymbol{z}_i, y = 1) \right\}_{i=1}^{N}$$

$$\underbrace{\mathcal{L}(\mathcal{B}_g, \Theta_d, \Theta_g)}_{\textbf{Loss}} = -\frac{1}{|\mathcal{B}_g|} \sum_{(\boldsymbol{z},y)\in\mathcal{B}_g} y \log D(G(\boldsymbol{z}))$$

$$\underbrace{\frac{\partial \mathcal{L}(\mathcal{B}_g, \Theta_d, \Theta_g)}{\partial \theta}}_{\textbf{Gradients}}, \quad \forall \theta \in \Theta_g$$

Outline

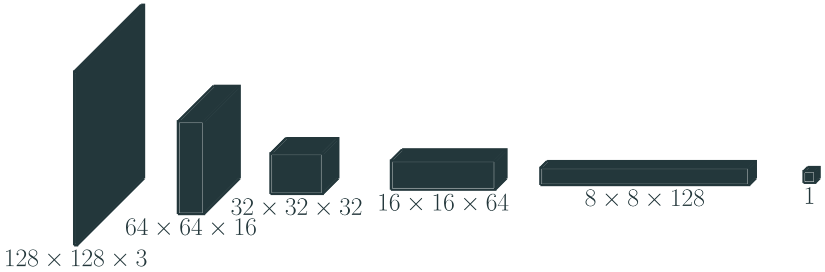
Generator



Generator

Generator architecture:

- Input: Noise vector 100×1,uniformly distributed white noise
- Deconvolutional layers
- Kernel size 5×5
- Fully connected first layer
- Activation function: LekyReLU
- Activation function last layer: tanh


$128 \times 128 \times 3$
$64 \times 64 \times 16$
$32 \times 32 \times 32$
$16 \times 16 \times 64$
$8 \times 8 \times 128$
1

Discriminator

Discriminator architecture:

- Input: an image
- Convolutional layers
- Kernal size 5×5
- Activation function: LekyReLU
- Fully connected last layer
- Activation function last layer: Sigmoid

Outline


A FEW EPOCHS LATER

Results (MNIST)

Figure 2: MNIST dataset

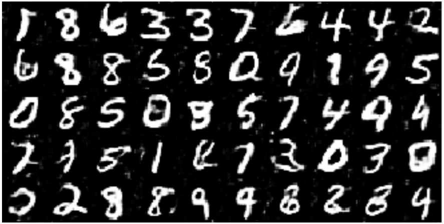
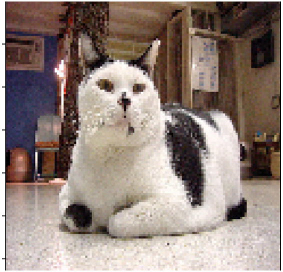


Figure 3: Generated numbers

Results (CIFAR 10) 100 Epochs

Results (Flowers) 100 Epochs

Cats 100 Epochs



Results (Cats)

Figure 4: Generated images after 55 epochs of training.

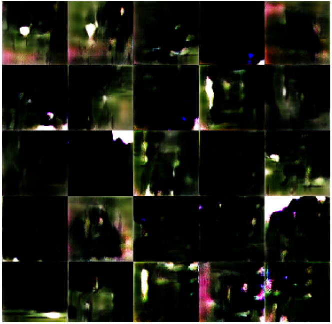


Figure 5: Generated images after 60 epochs of training.

Latent space exploration (MNIST)

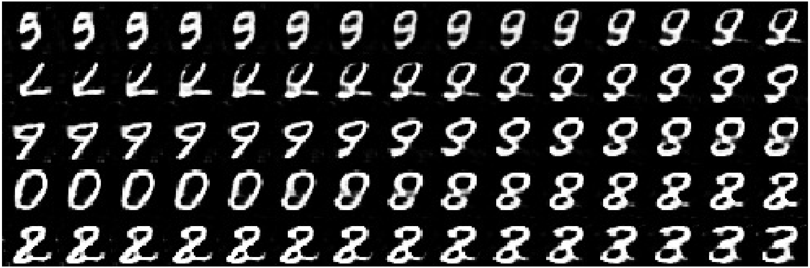


Figure 6: Latent space exploration for MNIST

Latent space exploration (Flowers & CIFAR 10)

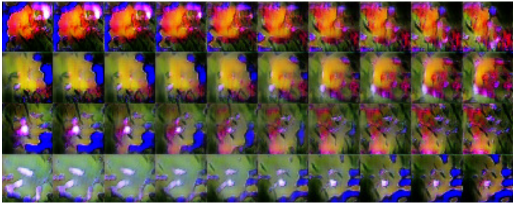


Figure 7: Latent space exploration for flowers

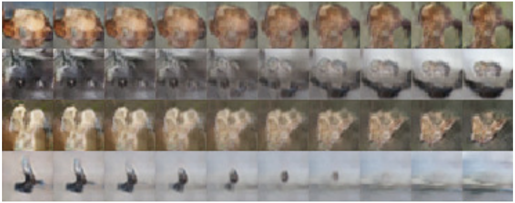


Figure 8: Latent space exploration for CIFAR10

Outline

Model collapse

When one of the networks outperforms the other

Our case the discriminator outperformed

Generator found a local minimum

Figure 9: Generated images after 55 epochs of training.

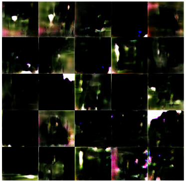


Figure 10: Generated images after 60 epochs of training.

The background in cat images are diverse and does not have anything with the cats attribute to do.

Figure 11: Cat

Figure 12: Cat

Conclusion

DCGAN works well with MNIST (handwriten numbers)

The network works okay with images of flowers

Hard for the network to generate images with diverse backgrounds, such as cats

Thank you for your attention!

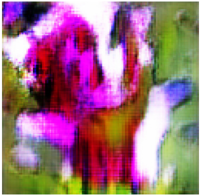


Figure 13: Summer flower