Deep Learning Project: Synthesis

Generating images by training Generative Adversarial Networks (GANs)

Johannes Loevenich

January 29, 2019

University of Bonn

Table of contents

Motivation

GENERATIVE ADVERSARIAL NETWORKS

Motivation

- Understanding Generative Adversarial models especially DCGAN's.

Motivation

- Understanding Generative Adversarial models especially DCGAN's.
- Can we measure the quality of the **discriminator** by removing the real/fake classifier and feeding the convolutional features into a new classifier? This would show that the model learned general, useful features.

Motivation

- Understanding Generative Adversarial models especially DCGAN's.
- Can we measure the quality of the **discriminator** by removing the real/fake classifier and feeding the convolutional features into a new classifier? This would show that the model learned general, useful features.
- Is it possible to show that there are dedicated parts of the **generator** that control properties of its output? In other words, do we reach some vector arithmetics on the input vector noise.

Training experiences

Motivation

- Training a DCGAN is some kind of an art.

Motivation

- Training a DCGAN is some kind of an art.
- While training the DCGAN on IMAGENET-1K the **discriminator** always got too strong.

Motivation

- Training a DCGAN is some kind of an art.
- While training the DCGAN on IMAGENET-1K the **discriminator** always got too strong.
- We had to interrupt training periodically and then just trained the **generator** for a few epochs.

Motivation

- Training a DCGAN is some kind of an art.
- While training the DCGAN on IMAGENET-1K the **discriminator** always got too strong.
- We had to interrupt training periodically and then just trained the **generator** for a few epochs.
- With our GPU time it was not possible to reach nearly a good classification accuracy as they did in the papers.

Motivation

- Training a DCGAN is some kind of an art.
- While training the DCGAN on IMAGENET-1K the **discriminator** always got too strong.
- We had to interrupt training periodically and then just trained the **generator** for a few epochs.
- With our GPU time it was not possible to reach nearly a good classification accuracy as they did in the papers.
- Training on celebA worked well and we got some nice results for the vector arithmetics.

Theory

What is a GAN?

- GANs are a framework for teaching a DL model to capture the training datas distribution so we can generate new data from that same distribution.

What is a GAN?

- GANs are a framework for teaching a DL model to capture the training datas distribution so we can generate new data from that same distribution.
- They are made of two distinct models, a **generator** and a **discriminator**.

What is a GAN?

- GANs are a framework for teaching a DL model to capture the training datas distribution so we can generate new data from that same distribution.
- They are made of two distinct models, a **generator** and a **discriminator**.
- The job of the **generator** is to spawn fake images that look like the training images.

GENERATIVE ADVERSARIAL NETWORKS

What is a GAN?

- GANs are a framework for teaching a DL model to capture the training datas distribution so we can generate new data from that same distribution.
- They are made of two distinct models, a **generator** and a **discriminator**.
- The job of the **generator** is to spawn fake images that look like the training images.
- The job of the **discriminator** is to look at an image and output whether or not it is a real training image or a fake image from the generator.

Discriminator Notation

- Let x be data representing an image. $D(x)$ is the discriminator network which outputs the (scalar) probability that x came from training data rather than the generator.

Discriminator Notation

- Let x be data representing an image. $D(x)$ is the discriminator network which outputs the (scalar) probability that x came from training data rather than the generator.
- Here, since we are dealing with images the input to $D(x)$ is an image of HWC size $3x64x64$.

Discriminator Notation

- Let x be data representing an image. $D(x)$ is the discriminator network which outputs the (scalar) probability that x came from training data rather than the generator.
- Here, since we are dealing with images the input to $D(x)$ is an image of HWC size 3x64x64.
- Intuitively, $D(x)$ should be HIGH when x comes from training data and LOW when x comes from the generator.

Generator Notation

- For the generators notation, let z be a latent space vector sampled from a standard normal distribution.

Generator Notation

- For the generators notation, let z be a latent space vector sampled from a standard normal distribution.
- $G(z)$ represents the generator function which maps the latent vector z to data-space.

Generator Notation

- For the generators notation, let z be a latent space vector sampled from a standard normal distribution.
- $G(z)$ represents the generator function which maps the latent vector z to data-space.
- The goal of G is to estimate the distribution that the training data comes from p_{data} so it can generate fake samples from that estimated distribution p_g.

GENERATIVE ADVERSARIAL NETWORKS

Generator Notation

- For the generators notation, let z be a latent space vector sampled from a standard normal distribution.
- $G(z)$ represents the generator function which maps the latent vector z to data-space.
- The goal of G is to estimate the distribution that the training data comes from p_{data} so it can generate fake samples from that estimated distribution p_g.
- So, $D(G(z))$ is the probability (scalar) that the output of the generator G is a real image.

Training by playing a MinMax game

- As described in Goodfellows paper, D and G play a **MinMax game** in which D tries to maximize the probability it correctly classifies reals and fakes ($\log D(x)$), and G tries to minimize the probability that D will predict its outputs are fake ($\log(1 - D(G(x)))$).

Training by playing a MinMax game

- As described in Goodfellows paper, D and G play a **MinMax game** in which D tries to maximize the probability it correctly classifies reals and fakes ($\log D(x)$), and G tries to minimize the probability that D will predict its outputs are fake ($\log(1 - D(G(x)))$).
- The GAN **loss function** is:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 D(G(x)))]$$

Training by playing a MinMax game

- As described in Goodfellows paper, D and G play a **MinMax game** in which D tries to maximize the probability it correctly classifies reals and fakes ($\log D(x)$), and G tries to minimize the probability that D will predict its outputs are fake ($\log(1 - D(G(x)))$).
- The GAN **loss function** is:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 D(G(x)))]$$

- In theory, the solution to this **MinMax** game is where $p_g = p_{data}$ and the discriminator guesses randomly if the inputs are real or fake.

What is a DCGAN?

- A **DCGAN** is a direct extension of the **GAN** described above, except that it explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively.

What is a DCGAN?

- A **DCGAN** is a direct extension of the **GAN** described above, except that it explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively.
- The **discriminator** is made up of strided convolution layers, batch norm layers, and LeakyReLU activations.

What is a DCGAN?

- A **DCGAN** is a direct extension of the **GAN** described above, except that it explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively.
- The **discriminator** is made up of strided convolution layers, batch norm layers, and LeakyReLU activations.
- The input is a $3x64x64$ input image and the output is a scalar probability that the input is from the real data distribution.

What is a DCGAN?

- A **DCGAN** is a direct extension of the **GAN** described above, except that it explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively.
- The **discriminator** is made up of strided convolution layers, batch norm layers, and LeakyReLU activations.
- The input is a $3x64x64$ input image and the output is a scalar probability that the input is from the real data distribution.
- The **generator** is comprised of convolutional-transpose layers, batch norm layers, and ReLU activations.

What is a DCGAN?

- A **DCGAN** is a direct extension of the **GAN** described above, except that it explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively.
- The **discriminator** is made up of strided convolution layers, batch norm layers, and LeakyReLU activations.
- The input is a $3x64x64$ input image and the output is a scalar probability that the input is from the real data distribution.
- The **generator** is comprised of convolutional-transpose layers, batch norm layers, and ReLU activations.
- The input is a latent vector, z, that is drawn from a standard normal distribution and the output is a $3x64x64$ RGB image.

STRUCTURE OF THE DCGAN NETWORK

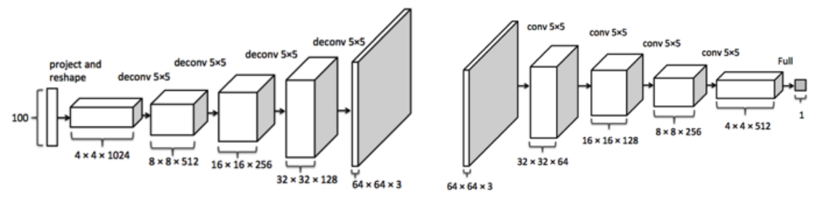


Figure 1: DCGAN network

Experiment 1: Classification of Food-101

CLASSIFICATION ON FOOD-101

The Data:

- IMAGENET-1K: pictures of 1000 different objects
- food-101: 101.000 images in 101 different food categories

The Experiment:

- Train a DCGAN on the IMAGENET-1K dataset. Use the **discriminator** as feature extractor for food-101.
- Use the output of all the convolutional layers of the **discriminator** as feature extractor and train a linear SVM on food-101.

RESULTS CLASSIFICATION ON FOOD-101

Overall accuracy

- We tested the **discriminator** as feature extractor on the whole food-101 dataset (101.000 pictures in 101 categories) and on 10 random categories.

RESULTS CLASSIFICATION ON FOOD-101

Overall accuracy

- We tested the **discriminator** as feature extractor on the whole food-101 dataset (101.000 pictures in 101 categories) and on 10 random categories.
- To measure our classification results we used the models overall **accurracy** and **precision** and **recall** on each of datasets classes.

RESULTS CLASSIFICATION ON FOOD-101

Overall accuracy

- We tested the **discriminator** as feature extractor on the whole food-101 dataset (101.000 pictures in 101 categories) and on 10 random categories.
- To measure our classification results we used the models overall **accurracy** and **precision** and **recall** on each of datasets classes.
- While **recall** expresses the ability to find all relevant instances in a dataset, **precision** expresses the proportion of the data points our model says was relevant actually were relevant.

$$recall = \frac{tp}{tp + fn}$$

$$precision = \frac{tp}{tp + fp}$$

RESULTS CLASSIFICATION ON FOOD-101

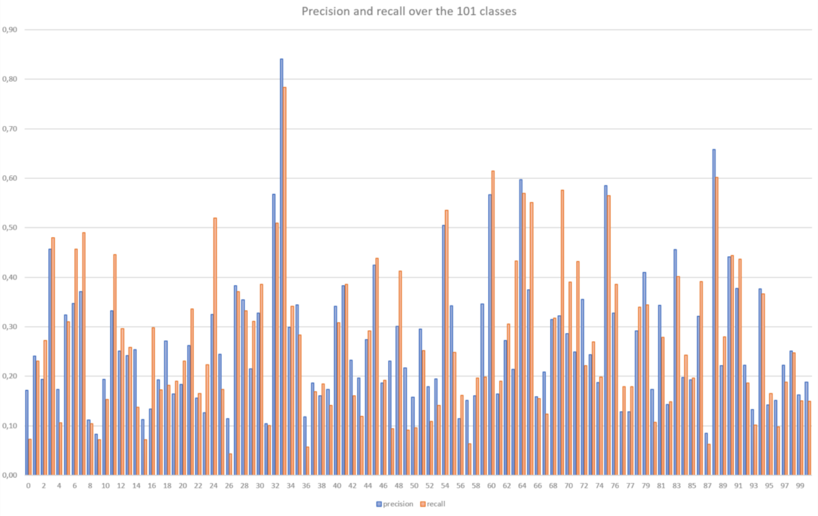


Figure 2: precision vs. recall over all 101 food categories

RESULTS CLASSIFICATION ON FOOD-101

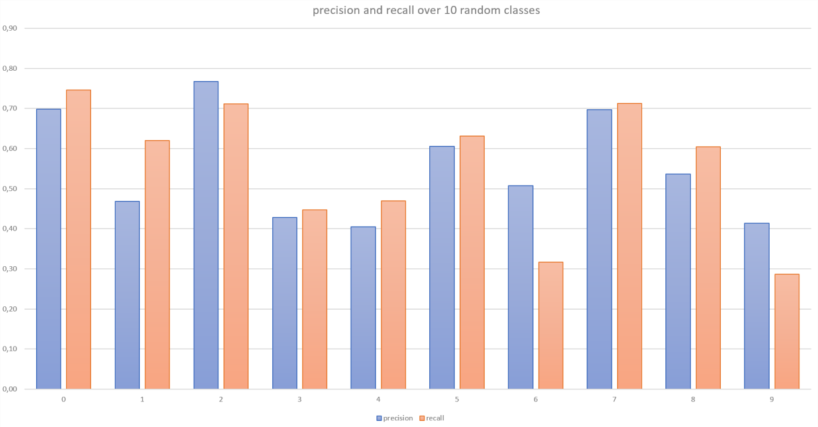


Figure 3: precision vs. recall over 10 random food categories

Experiment 2: LSUN conference-room

LSUN CONFERENCE-ROOM

The Data:

- LSUN conference room: 229.069 images of conference-rooms

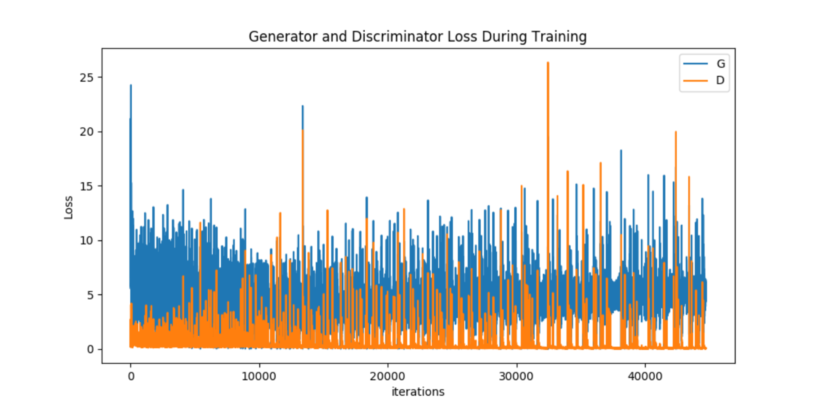
The Experiment:

- Train a **Generator** on the LSUN conference-room dataset.

LSUN CONFERENCE-ROOM

Generator and Discriminator losses

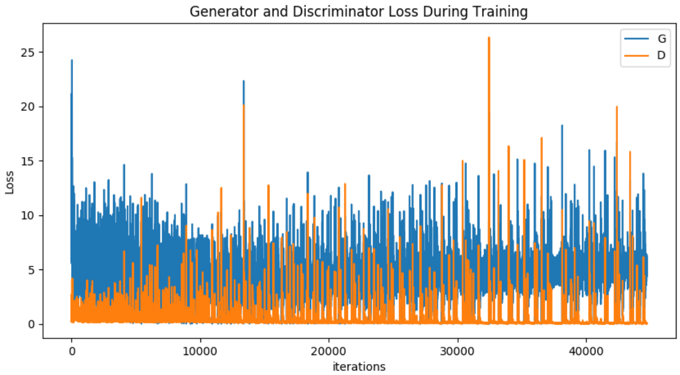
- $Loss_D$: **discriminator loss** calculated as the sum of losses for the all real and all fake batches $\log(D(x)) + \log(D(G(z)))$.



LSUN CONFERENCE-ROOM

Generator and Discriminator losses

- $Loss_D$: **discriminator loss** calculated as the sum of losses for the all real and all fake batches $\log(D(x)) + \log(D(G(z)))$.
- $Loss_G$: **generator loss** calculated as $\log(D(G(z)))$



LSUN CONFERENCE-ROOM REAL VS. FAKE

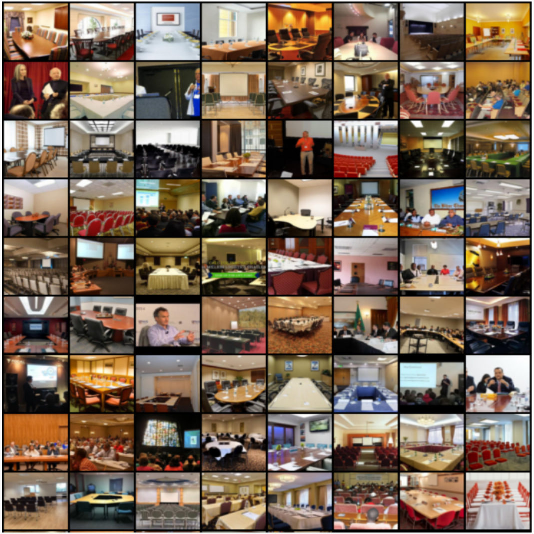


Figure 5: Real samples from the LSUN conference-room dataset

LSUN CONFERENCE-ROOM REAL VS. FAKE

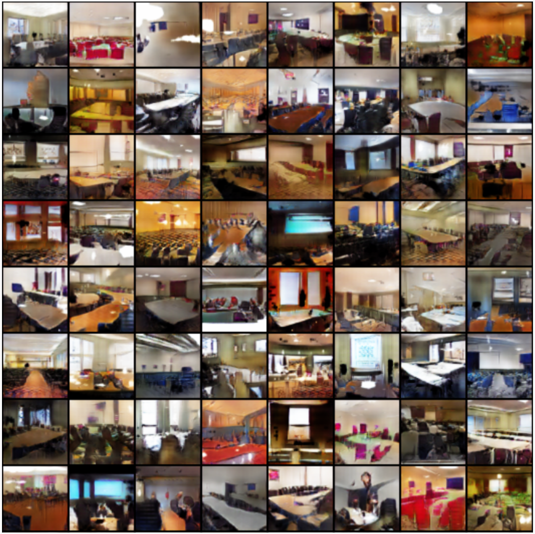


Figure 6: Fake samples after 25 epochs

Experiment 3: Vector arithmetics for celebA

VECTOR ARITHMETICS FOR CELEBA

The Data:

- celebA faces dataset: 10.177 number of identities , 202.599 number of face images

The Experiment:

- Train a **DCGAN** on the celebA faces dataset. Take two sets of 3 normal distributed sample vectors A and B. Calculate both subsets means ($mean(A)$ and $mean(B)$). Then return the **generator's** output of $mean(a) + mean(B)$.

RESULTS VECTOR ARITHMETICS FOR CELEBA

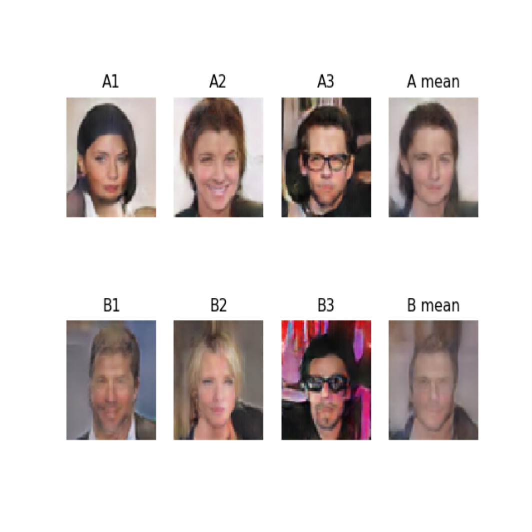


Figure 7: Means

RESULTS VECTOR ARITHMETICS FOR CELEBA

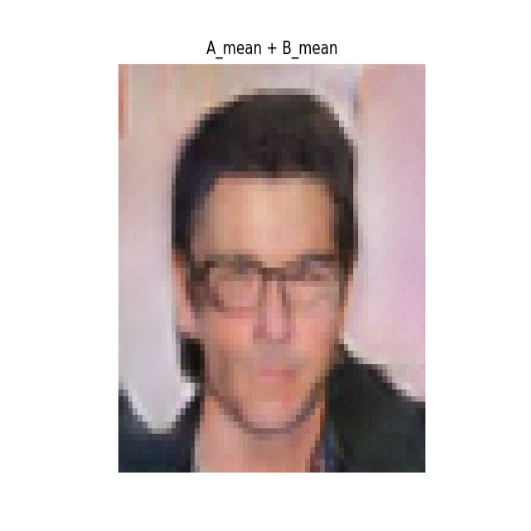


Figure 8: Sum of the means

Questions?