Képgenerálás Generatív Versengő Hálózatokkal

Generating images with Generative Adversarial Networks

Bordák Máté, Pelle Mátyás, Németh Gábor

BUDAPEST, 2019

Abstract

Képek generálása az egyik újszerű alkalmazása a neurális paradigmának. A gépi tanulás ezen része az ún. felügyelet nélküli tanuláshoz tartozik, ez az az eset, ahol bemeneteinkhez nem tartoznak elvárt kimenetek, ezek alapján kell valamilyen eloszlást megtanulnunk vagy klaszterekbe csoportosítani a tanítómintáinkat. Jelen esetben a tanítóminták halmazát a Google Open Image Dataset-ből szereztük, jelen esetben kutyákat generáltunk. Természetesen ennek valós alkalmazásai is lehetnek, például emberek „öregítése", vagy különböző filterek létrehozása, esetleg a jövőben lehetséges lehet akár a ruhapróbálás otthonról. A képgenerálást generatív versengő hálózatokkal valósítottuk meg, ebben az esetben egy két szereplős játék folyik le egy generátorhálózat és egy diszkriminátorhálózat között. A generátor megpróbál olyan képeket generálni, amelyek átverik a diszkriminátor hálót, a diszkriminátor pedig megpróbálja megkülönböztetni a valódi és a generált mintákat egymástól, így a diszkriminátoron keresztül tanul a generátor, ami problémát jelenthet[7]. A dolgozatunkban kitérünk a hálózat tanítása során felmerülő esetleges problémákra, illetve azok megoldásaira.

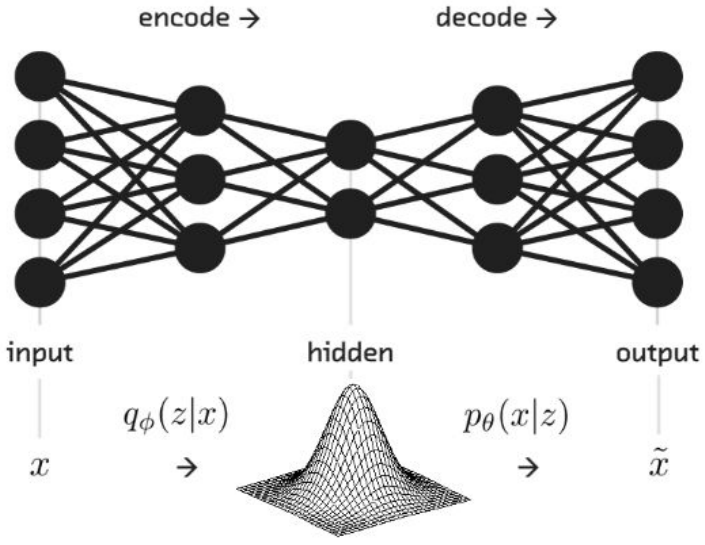
Generating images is one of the newest applications of the neural paradigm. This part of machine learning is called unsupervised learning, which means that there are no expected values linked to the inputs, only the inputs itself. This way we can learn the distribution function or cluster the data. In this case we obtained our data from the Google Open Image Dataset, with the aim of generating dogs. Of course, there are several actual applications of generating images, for example „aging" the person on a picture or creating different filters and perhaps in the future trying on clothes from your apartment might be an opportunity. We solved the problem with Generative Adversarial Networks (GAN). A GAN is basically a two-player game. The generator network tries to generate pictures that deceive the discriminator network, meanwhile the discriminator network tries to guess whether the picture came from the generator or from the real world. This way the generator network is trained through the discriminator network, which might cause some trouble[7]. In our study, we tried to cover the problems of training a GAN, and suggest solutions to the problems.

Introduction

Neural networks and deep learning(DL) is one of the most researched areas of artificial intelligence(AI). This has come so far that if we see an article in the news which mentions AI, we are 99% accurate if we say that the device uses some sort of deep learning algorithms. One of the most shocking interactions with AI today might be when we look at a picture and we wouldn't even think that it was made by a machine, but it turns out to be made by one. This is where unsupervised learning comes to the table. The job of the network is to learn some of the features of the target we want to create pictures of. For example a human has two eyes, one nose and two ears (well, most of the cases), so it has to be represented somehow by the network. Different networks use different methods to represent a higher level knowledge of the object.

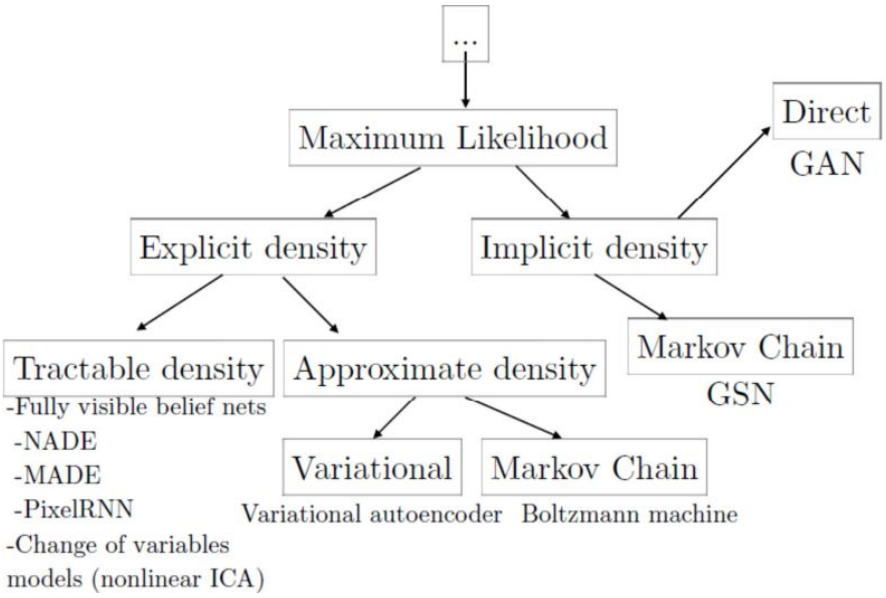
Related Work

Since neural networks and computer vision is one of the most researched areas, there are several solutions to the problem of image generation. One of the first approaches are Autoencoders[1]. The points of the encoder part is to transfer the input pictures to a latent dimension, that represent a higher level knowledge, this could interpreted as compressing the data. After the encoding, the decoder part of the network decodes encoded input vector. This way if we generate latent vectors with unimodal density functions and we feed it into the decoder network, the output is a desired picture.



1 Visualization of an Autoencoder network[5]

Altough there are several ways of learning density functions - shown in picture 2 -, we decided to use GANs.[7] We created both of the networks in keras, with dense layers and then tried to optimize the hyperparameters so that we get perfectly cute, yet not real dogs.



2 Methods used to generate pictures[4]

Our Work:

Our work had one purpose and one purpose only, which is to generate perfectly real looking dog pictures using GAN.

Preparing the data:

We gathered our data from the Google Open Image Dataset V5. This dataset contains 6 million images of a variety of different objects. Since this dataset is too large for us to download it all, and we only need images of dogs, had to filter it first, then download only what we needed. At first we were using all of the pictures from this dataset that has a dog on them, but our results were not satisfying. Later we realised that this might be because on many of the pictures we were using, the dogs were in the background, barely visible, or in some cases there was no dog at all. We had no other option, we had to filter the images manually. We did not have weeks to go through the entire dataset, so we filtered about a 1000 pictures. To increase the size of out training set, we used some basic data augmentation techniques too[9].

Training and optimizing:

In the beginning, the training did not start off well, but it would have been a miracle if it did. In the first iterations, the loss of the generator network was way bigger than the loss of the discriminator network. We quickly realised, that our networks were too shallow. Then we started experimenting with deeper networks, which clearly had a good effect on the performance.We discovered, that if the discriminator was too deep, we had a problem with vanishing gradients[10]. In our initial model, we used Leaky ReLU as activation functions in both the discriminator, and the generator networks. Then we experimented with tha normal ReLU in the generator network, and it seemed to perform better. In the earlier versions our networks had no convolutional layers, but that did not work out because it had lost its sense of consecutive pixels. Earlier we thought that the discriminator does not need many layers, but we realised that it does. After this, the generator quickly overlearned, so we had to use play with learning rates, which resulted in better learning. As the weight initialization, we used the Glorot (also known as Xavier) [11] technique. We tried many different alpha learning rates, different amounts of layers, batch sizes, strides, kernel sizes. Bigger kernel sizes resulted in better performance.

Evaluation, testing:

Our results were clearly not as good as we expected, but just for the sake of it, we did a formal test:

We asked some of our friends to tell if a picture was from the real world or is it generated by the network. We showed them pictures from the training data and from our generated dataset. Out of 3 participants, the 3 pictures below got the following realness percentages:

The real pictures:

3 real, 0 fake

2 real, 1 fake

3 real, 0 fake

And the generated pictures:

0 real, 3 fake

0 real, 3 fake

0 real, 3 fake

So our generated images did not manage to fool anybody. We may get better results next time.

Here is a couple of selected images, these are some of the best results we have gotten during the project:

Future plans, conclusion

Spectral normalization[8] could be tried in the future to stabilize the training of the discriminator. A persisting challenge in the training of GANs is the performance control of the discriminator. In high dimensional spaces, the density ratio estimation by the discriminator is often inaccurate and unstable during the training, and generator networks fail to learn the structure of the target distribution.

We attempted to make our GAN network as good as we could, but getting to the point where it has the performance we need is not easy. GAN networks have a lot of parameters. We tried a variety different network structures, without and with convolutional layers, with smaller kernel sizes and with bigger ones, and a range of different number of neurons and number of layers. We found that the GAN networks performed better with less layers, and with bigger kernel sizes.

Despite all our efforts, our final results are not the best, at the beginning of the project we were hoping for something better. We think the most important thing is that we learned a lot along the way. The experience we gained from this project will help us a lot in our future work.

References

[1] http://proceedings.mlr.press/v27/baldi12a/baldi12a.pdf

[2] https://arxiv.org/pdf/1511.06434.pdf

[3] https://arxiv.org/pdf/1411.1784.pdf

[4] http://www.iangoodfellow.com/slides/2016-12-9-gans.pdf

[5]https://blog.fastforwardlabs.com/2016/08/22/under-the-hood-of-the-variational-autoencoder-in.html

[6] https://lanpartis.github.io/deep%20learning/2018/03/12/tricks-of-gans.html

[7] https://arxiv.org/pdf/1406.2661.pdf

[8] https://arxiv.org/pdf/1802.05957.pdf

[9] https://arxiv.org/pdf/1711.04340.pdf

[10] https://arxiv.org/pdf/1804.00140.pdf

[11] http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf