Project Report

Artificial Intelligence and Applied Methods

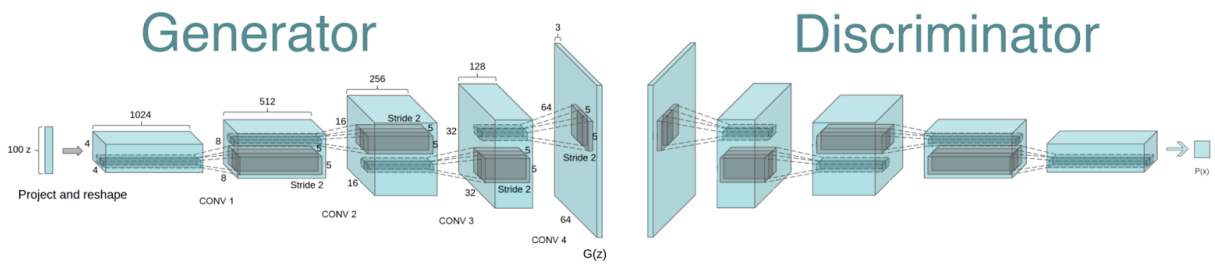
Nikhil Nadig
Hanstavägen 49, 1302, Kista 16453
nadig@kth.se
+46 793164360

Óttar Guðmundsson
Trubadurvågen 1 171 69 Solna Stockholm
ottarg@kth.se
+354 6933881

Image from epoch 800

Initial Study

Generative adversarial networks (GANs) are a class of artificial intelligence algorithms used in unsupervised machine learning, implemented by a system of two neural networks contesting with each other in a zero-sum game framework. GANs were introduced by Ian Goodfellow in 2014.
This technique can generate photographs that look at least superficially authentic to human observers, having many realistic characteristics (though in tests people can tell real from generated in many cases).
In GANs, one network generates results and the other network evaluates them. The generative network learns to map from a latent space to a particular data distribution of interest, while the discriminative network discriminates between instances from the true data distribution and candidates produced by the generator. The generative network's training objective is to increase the error rate of the discriminative network. In other words, the objective of the generative network is to "fool" the discriminator network by producing novel synthesized instances that appear to have come from the true data distribution.



Design

In neural networks, the internal implementation of the networks are not clear. This makes it difficult to define a proper system design for the solution. There is no user interaction needed at any point in the running of the solution. The closest design the project is that of a pipe-and-filter. The album covers were the input to the system and the network generated images were the output.
There is file called "references.txt" which contains a list of all the resources that were used for the implementation of the solution.

Implementation

To create the network, the idea was to use a neural network as a base. After reading a lot about deep learning, computer vision and tutorials on GAN's we figured out that Tensorflow was the perfect library for the project implementation. Other libraries like SciKit-learn could have been used, but it wasn't as advanced as the google library for image and pattern recognition. The IBM SPSS was also not a good solution to our implementation since it clearly isn't designed around tasks such as image generation. Tensorflow was not included in the course so a little research was needed before using it.
To run it at first was easier said than done. To install Tensorflow, Anaconda was needed to be installed it since a lot of package dependencies needed to be managed in a proper way. Finally a few python libraries were needed such as numpy and scikit-learn.

After implementing a solution that didn't crash after a few runs, a more professional environment was needed to do the calculations in proper time. For that we used Floydhub which was also not covered in the course, but using it is easy if their documentation is followed thoroughly.

The whole code is documented and it can be read under *modded_GAN.py*. It's split into four different parts. The first one is the data gathering and structuring. After that, a generator and discriminator objects are created and finally the training cycle itself.

Testing

For this project, the testing was pretty limited. Neural networks aren't countable for a guaranteed solutions since we don't know what is happening inside of them. Thus, user tests weren't implemented. We did however check if the network was doing something after a few training rounds by seeing if it loaded the images to data tensors, if the loop ran successfully through and if it generated and saved the image properly. Later on we also added the checkpoint loading so we could load the network and how it was trained at different epochs instead of running through the whole session all over again.
The hardest part about the testing was to simply know what was going on. Tensorflow is quite different to other languages based on the input in functions, variables in scopes and thinking in dimensions and matrixes instead of object and variables. A lot of tutorials and code examples were needed to stitch together a working solution .

Delivery

Unzip the file and open the FinalProject folder. There, a file is included called "howToRun.txt" where all of the information needed on how to run the code and generate something on a local computer. If running it on another virtual machine like Floydhub is needed, simply load the whole FinalProject folder to the machine and call the python command. The images created and checkpoints will be saved under C://output/our.

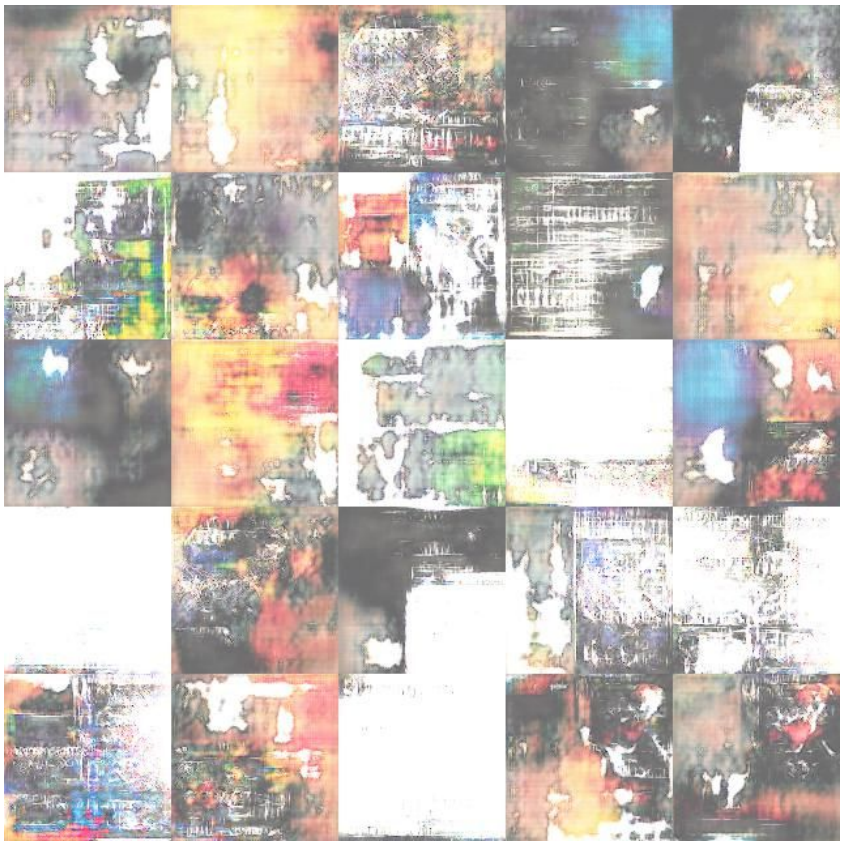


Image created from checkpoint 2000