GAN for Houses

Generating Houses With A Generative Adversarial Network

Mohamed Asni
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
aasni089@uottawa.ca

Alexandre Boyer
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
aboye018@uottawa.ca

Abstract— **This paper describes our adventure in learning deep machine learning concepts as well as how we applied said concepts to our project. This research and project experience was inspired by Ian Goodfellow's paper about generative adversarial networks (GANs) published in 2014. Our objective was to develop a GAN capable of generating houses with plausible features. With the help of tools such as Tensorflow and Keras we managed to successfully develop such a GAN. We started by creating a GAN that could generate MNIST numbers as a learning step. Once we successfully completed that step, we then implemented our GAN that is capable of generating houses with realistic features. Our model was not as accurate as we hoped since time was a scarce resource all throughout the process. A larger dataset, better network architecture, and correctly normalized data are major steps that could be undertook in order to improve our model.**

Keywords— GAN; neural network; generator; machine learning; University of Ottawa;

I. Introduction

This document serves as a final report for our Computer Architecture 3 project. Our supervisor and mentor, Daniel Shapiro, assigned this project to us as well as directed us towards resources that we used in order to accumulate the necessary knowledge to successfully complete our project. Generative Adversarial Networks, or GAN for short, are a class of unsupervised machine learning algorithms. The idea of such a neural network was introduced by Ian Goodfellow in his 2014 paper. GANs are comprised of two neural networks that are contesting with each other in what can be described as a "zero-sum game". The process is undertook where one model, the generator, attempts to generate realistic data in order to fool the other model, the discriminator. At the same time the discriminator attempts to predict whether or not the input data it has been given is real or fake. Where the fake data is generated by the generator neural network and the real data is directly provided from a predefined dataset. In short the generator network is tasked to generate realistic data, while the discriminator network attempts to tell whether the provided data is real or fake. After each iteration, the discriminator network provides feedback to the generator about how to make the generated data look more realistic. [3]

At the start, the generator creates data that are completely fake and can be considered as pure noise. However, as the training process progresses, and as the generator receives feedback from the discriminator, it progressively creates more and more realistic data. The training process is complete once the discriminator can no longer tell the difference between the generated data and the real data. Ian Goodfellow claims that how the human brain works, with the right hemisphere competing against the left hemisphere, is what inspired him to come up with the generative adversarial network model. In fact Ian states in an AI podcast — "If you take a deep neural network and you teach it to read, to actually look at photos and recognize letters that it can see in the photo, it can do that about as well as a human being can. But the process of learning to do it doesn't look anything like the process that a human follows to learn to read." Here, Goodfellow is basically stating that although neural network models may behave as humans do, they still do not learn the same way humans do. [2] The following describes the process we undertook to develop our own generative adversarial network.

II. Preparing the data

A. Cleaning the data

To ensure the data is useful and not redundant, certain procedures were undertook in order to clean the data. The first step was removing certain attributes that were deemed unecessary for the task at hand. Next all the qualitative data was converted into number representations using a key-value pair algorithm. For instance, if an attribute had three possible choices, this attribute would be represented by a number from 0 to 2.

B. Normalizing the data

A technique used in machine learning to increase the performance of models is to normalize the input data. This normalization restricts values from being larger than 1 and smaller than -1. For example, the attribute representing the year of construction of the house clearly needed to be normalized. We then used this formula on each attribute for the entirety of the dataset. Let $\bar{v}$ be the normalized attribute of the i th house, $f(i)$ the non normalized attribute of the i th house and N the total number of houses. Hence, we can represent our normalization step with the following formula.

$$\bar{v} = \frac{f(i)}{\sum_{j=0}^{N-1} f(i)^2}$$

III. Developing the GAN

The GAN was developed following a certain methodology [1]. The GAN that Faizan Shaikh had described in his work generated and classified MNIST photos. His model had to be adapted and transformed into a model that would be able to generate and classify our own data. The model, like all traditional GANs is comprised of a Discriminator and a Generator.

C. The Discriminator

The discriminator's task is to predict whether or not a given input is real or fake. The network is comprised of 12 layers. The first layer, our input layer, specifies the shape of the input tensor. The second layer, our batch normalization layer, normalizes the activation functions of the previous layer to keep the mean of the previous layer's activation values near 0 and its standard deviation near 1. We noticed that when we apply this layer as our second layer, it seems to stabilize the loss of the discriminator and the generator throughout training. If we were to remove that layer, the model seemed to oscillate between a loss value of 5 and 20. As previously stated, time was a scarce resource in this project, as such we did not have enough time to further investigate why such an effect takes place with a batch normalization layer as the second layer. The next layers consist of 2 Dense layers of 50 and 70 neurons, respectively. Next, a dropout layer with a probability of 5% was added. The reason for adding a dropout was to reduce chances of overfitting. We then added 5 Dense layers of 100, 70, 40, 25, and 20 neurons, respectively. This increased the likelihood of the discriminator learning to successfully classify the training data. Finally, the last layer is a dense output layer with a single neuron. All the dense layers use Relu as their activation functions except for the very last layer. A sigmoid activation function was used for the output layer in order to give a binary representation of the output probability. Each dense layer also used kernel regularizers of $L1L2(1e-5,\ 1e-5)$. Kernel regularization applies a penalty to changing weights considered by the loss function. This is another tool that we plan to further investigate as the lack of time did not allow us to fully understand the scope of its effects. We still added it because it seemed to reduce the loss of the model.

D. The Generator

The generator's architecture is not so different from that of the discriminator. The generator neural network consists of 9 layers. The first one being the input layer which defines the shape of the input tensors. The second layer is a dense layer made up of 20 neurons. The third layer, is a batch normalization layer that normalizes the activation as was done with the discriminator. Following the third layer are 5 dense layers made up of 25, 60, 80, 60, and 40 neurons, respectively. The number of neurons were increased in the hidden layers in order to increase the likelihood of the generator's ability to model more complex behavior. The final layer, the reshape layer, ensures the data is in proper format. This will allow the generated output to be used as input for the discriminator. Once more, all the the dense layers used Relu as the activation function. As had been done with the discriminator, for the purpose of better results, we added kernel regularizers.

IV. Training the model

Certain steps were undertook to efficiently and successfully train the model. GANs are notoriously difficult to train as many pitfalls exist during the training process. Certain "good practice" steps were implemented in order to increase training effectiveness. Such steps include batch normalization, dropout, and pretraining the discriminator. [4]

E. Pretraining the discriminator

One of the most effective ways to make sure that a GAN successfully trains, is by pretraining the discriminator for n epochs. [4] This technique was mentioned in many sources during our research phase. Essentially, the idea is to train the discriminator on real data first as to allow it to be able to tell the difference between real and fake data. [1]

F. Training both neural networks

The training step of both networks was done in a sequential manner. The reason we chose to do it as such, is because, based on research, it seemed to better fit the scope of this project while other techniques did not or were beyond the scope of this project. Yet another part of our process that needs further investigation if time permitted. As we trained the discriminator, the generator was put into a frozen state. Meaning no backpropagation and only forward pass. On the next batch, the opposite was performed. Meaning as the generator was being trained the discriminator was put into a frozen state. [1] We trained the GAN over 1000 epochs and noticed that the model seemed to stabilize at just over 100 epochs. We therefore decided to stop the training at 100 epochs.

G. Results

Once the training of our model had completed, sample results were generated. These results are described here bellow.

TABLE I. Sample generator output

Attributes	Generated house data sample
Lot area (ft^2)	3.36092529e+03
Overall quality (from 0 to 10)	2.25536890e+03
Overall condition (from 0 to 10)	3.37884883e+04
Year built (yr)	2.00879180e+04
Total basement surface (ft^2)	-1.33716125e+02
1st floor surface (ft^2)	2.17533057e+03
2nd floor surface (ft^2)	7.86969678e+03

Table 1 gives a sample of the 40 attributes generated by the generator post training. It is painfully obvious that something went wrong in the process, we will discuss possible reasons for this inaccurate result shortly. First of all, we can see in the first row that a lot area of 3360 ft^2 is rather small and the year built of the house is in the 20 thousands. It also seems that the other big results have negatively impacted the overall quality and condition attribute, given the 2000 and 30 000 that should have been from 0 to 10.

V. Possible mistakes

During the learning process of the model, certain grave mistakes were committed. These mistakes were the root cause of the inaccurate output shown in Table 1. The following discusses some of the mistakes.

H. Normalization Mistake

We firmly believe that the biggest mistake that was made was that the data was not properly normalized. The technique we used seems to have introduced a certain magnitude error in our data. For example, for the exterior material, brick could have been encoded by 1 and wood could have been encoded with 2. It would be foolish to say that wood is twice brick, but 2 is twice 1. It is therefore fair to assume that the GAN might have took for granted this kind of relationship which would result in inaccurate results. It later came to our attention that one hot encoding is a much more efficient way to encode qualitative data.

I. Neural Network Architecture

The approach used to build the architecture of the GAN was naive. We had no prior knowledge of neural network architecture before we attempted this project. This proved challenging because in order to build a proper architecture to represent a proper model, one either needs to have a good idea of what the model should resemble in terms of complexity or enough experience with neural networks to have a general idea of where to start in terms of designing a proper architecture. Nevertheless, this project was a great medium through which we immersed ourselves into the magnificent and complex world of deep machine learning.

J. Training dataset

Another mistake we think caused problems with our results is that the dataset that was used was rather small. With a dataset of a mere 1500 samples, we believe that given a large dataset, the model would have been able to learn better and provide more accuracy in terms of outputs. A larger dataset accompanied with adjustments to the mistakes described previously, would have allowed for better and more accurate results.

Acknowledgment *(Heading 5)*

We would like to thank Daniel Shapiro for proposing this extremely fun project to us and providing guidance to us towards resources that made it possible to accomplish some form of a final product. We would also like to thank Dr. Miodrag Bolic for allowing us to do this project as part of our Computer Architecture 3 course. Finally we would also like to thank Phil Williams for his invaluable help throughout this project as a consultant.

References

[1] F. Shaikh, "Introductory guide to Generative Adversarial Networks (GANs)", Analytics Vidhya, 2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/06/introductory-generative-adversarial-networks-gans/. [Accessed: 12- Dec- 2017].

[2] A. Toh, "An Argument in a Bar Led to the Generative Adversarial Networks", The Official NVIDIA Blog, 2017. [Online]. Available: https://blogs.nvidia.com/blog/2017/06/08/ai-podcast-an-argument-in-a-bar-led-to-the-generative-adversarial-networks-revolutionizing-deep-learning/. [Accessed: 12- Dec- 2017].

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, Generative Adversarial Nets. Montreal, 2017.

[4] S. Chintala, "soumith/ganhacks", GitHub, 2017. [Online]. Available: https://github.com/soumith/ganhacks. [Accessed: 12- Dec- 2017].