

Report on GANs and Learning from MNIST Dataset

Name | Course Title | Date

# Introduction

## Generative adversarial networks, or GANs, are a strong type of neural network used for deep learning and unsupervised learning where unlabeled data is fed to the model.

GANs, particularly their application to the MNIST dataset. The MNIST dataset is a vast collection of 28x28 grayscale images which are of handwritten digits from 0 to 9. It serves as an excellent standard for testing the capabilities of GANs in image generation tasks. This report aims to look into the architecture, training process, and potential applications of GANs using the MNIST dataset.

A screenshot of a computer

Description generated with very high confidence

Figure Generative Adversarial Nets. Credits ~Ian J. Goodfellow

Training Process of GANs

Because a GAN contains two separately trained networks, its training algorithm must address two complications:

* GANs must juggle two different kinds of training (generator and discriminator).
* GAN convergence is hard to identify.

We keep the generator constant during the discriminator training phase. As discriminator training tries to figure out how to distinguish real data from fake, it has to learn how to recognize the generator's flaws. That's a different problem for a thoroughly trained generator than it is for an untrained generator that produces random output.

Similarly, we keep the discriminator constant during the generator training phase. Otherwise the generator would be trying to hit a moving target and might never converge.

It's this back and forth that allows GANs to tackle otherwise intractable generative problems. We get a toehold in the difficult generative problem by starting with a much simpler classification problem. Conversely, if you can't train a classifier to tell the difference between real and generated data even for the initial random generator output, you can't get the GAN training started.

The generator tries to produce images so realistic that the discriminator can't tell them apart from actual pictures. This tag-team helps us make computer-generated images that look just like the real deal. They're versatile and can be used in various fields like making lifelike photos, writing in a human-like manner, and even helping in technical tasks.

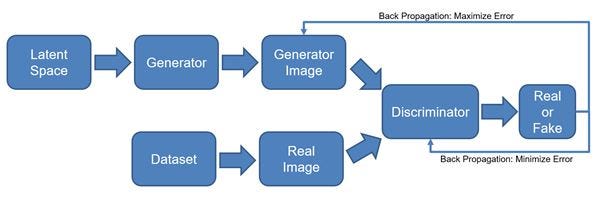


Figure GAN Block Diagram

OUR EXPERIMENT

In our experiment, we carefully designed three distinct models, each with its unique characteristics and configurations. These models enable us to compare and determine the most effective approach for our task.

Model 1, our initial focus, implements LeakyReLU as the activation function for both the Generator and the Discriminator. The Generator employs 4 Dense layers with 256, 512, and 1024 neurons respectively. This model proved to be effective, especially after training the 'Generator\_model.h5' for 30,000 iterations, resulting in the generation of high-quality images.

For Model 2, we drew inspiration from TensorFlow's practices to gain an external perspective on generative AI. This allowed us to explore alternative techniques and performance standards.

Model 3 represents an extension of Model 1, where we fine-tuned the hyperparameters to a more intricate level. We introduced the Exponential Linear Unit (ELU) activation function, a higher learning rate, and more complex Dense layers. This added complexity aims to further enhance the model's capabilities and image generation quality.

Our comprehensive experiment considers the strengths and weaknesses of each model, providing valuable insights into the most effective strategies for generative AI.

The results obtained from our model utilizing the LeakyReLU activation function were quite impressive. After training the model for 30,000 iterations, we observed significant improvements in image generation. The generated images exhibited a remarkable level of detail and realism, closely resembling the handwritten digits in the MNIST dataset. This demonstrated the effectiveness of the LeakyReLU activation function and the chosen architecture in producing high-quality images.

Detailed Review on the MODEL 1 (Training the Model using LeakyReLU)

The First model which we have trained uses the LeakyReLU activation function which is (rectified linear unit) which we used to overcome the "dying ReLU" problem, where neurons can become inactive during training. The primary advantage of the Leaky ReLU activation function is that it allows a small gradient for negative values, preventing neurons from becoming completely inactive during training. This can lead to faster convergence and more robust training of deep neural networks. We have constructed sequential model which contains both the generator and the discriminator. The discriminator model consists of **three dense layers** with varying numbers of neurons (256, 512, 1024) and **LeakyReLU** activation functions. The generator model also consists of **three dense layers** with varying numbers of neurons (256, 512, 1024) and **LeakyReLU** activation functions.

**epochs** - The number of training epochs. This parameter has been trained over 30000 times.

**Batch size** - The size of the mini-batch used for training. We have considered a batch size of 32 for the model 1.

**noise shape** - The shape of the noise input to the generator (1D array of size 100)

**Binary Cross-Entropy** (BCE) loss is used for the discriminator. This is a common choice for GANs.

**ADAM** optimizer has been used to maintain the stability of the model.

We train the model over the MNIST dataset and observe the following changes in the loss between the generator and the discriminator, accuracy and the quality of the generated images by the model. The LeakyReLU has been quite a successful model and it is widely used due to it’s reliable generations of images, sharp details and a steady learning curve.

The Generator employs Dense layers with 256, 512, and 1024 neurons. This model proved to be effective, especially after training the 'Generator\_model.h5' for 30,000 iterations, resulting in the generation of high-quality images

The generated image by the model is displayed below at epoch level 30000.

A number on a black background

Description generated with high confidence

Figure LeakyReLU model image at 30000 epoch

The loss plot between the discriminator and the generator are shown below

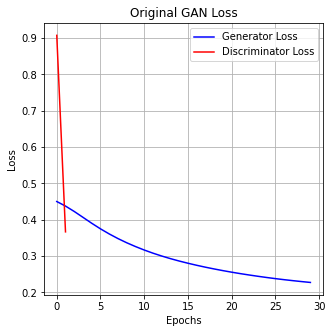


Figure Generator vs Discriminator loss

Detailed Review on the MODEL 2 (Training the Model using Tensorflow 2 with LeakyReLU)

The model is inspired from Tensorflow who are credited to the original work, which enhances the power of leakyrelu and Provide an even better output which when trained properly surpasses the model 1 which uses vanilla LeakyReLU. The Model gives a potentially good result and an inspiring outcome which promotes the sharpness and clarity of the images at a lower level training compared to the model 1 which when trained for almost 100k iterations gives a promising result.

The generator is responsible for creating fake images from random noise.

It starts with a dense layer with 7x7x256 (12544) neurons, using a LeakyReLU activation and batch normalization. This layer generates a feature map. The feature map is reshaped to (7, 7, 256) to resemble an image-like structure. The generator then applies three transposed convolutional layers.

Each layer up samples the feature map. The first two layers use 128 and 64 filters, respectively, with LeakyReLU activation and batch normalization.

The final transposed convolutional layer outputs a 28x28x1 image with a tanh activation function, ensuring the pixel values are within the range [-1, 1].

**epochs** - The number of training epochs. This parameter has been trained over 500 times.

**Batch size** - The size of the mini-batch used for training. We have considered a batch size of 16 for the model 2.

**noise shape** - The shape of the noise input to the generator (1D array of size 100)

**Binary Cross-Entropy** (BCE) loss in the model.

**tanh** activation function has been used in this model.

We define the generator model using the Sequential API provided by Keras. The generator starts with a fully connected layer that takes a random noise vector of dimension latent\_dim as input. The LeakyReLU activation is applied.

A graph of blue and red lines

Description generated with high confidence

Figure Tensorflow Gan Model

The result at only 5 epochs gives a very worthy result which is human readable.

Detailed Review on the MODEL 3 (Training the Model using ELU)

ELU, or Exponential Linear Unit is another activation function which is used to train the Neural Networks. The ELU activation function is a very well known activation function which is known for its ability to Avoid Dead Neurons and having smoothness which can make it easier for gradient-based optimization algorithms to converge whilst also being able to produce negative values which can make it easier for gradient-based optimization algorithms to converge. The other changes in this particular model includes the increased complexity in between the corresponding dense layers which makes for a better result. We have also used Exponential decay in order to gradually reduce the learning rate during training by applying an exponential decay function. We have seen promising results in this model as well when we compare the level of iterations trained.

The activation function used is ELU (Exponential Linear Unit) with an alpha parameter of 1.0. ELU is an alternative to LeakyReLU that can help with training stability.

Batch normalization is applied after the ELU layer with a momentum of 0.9. Adjusting the momentum can impact training stability and convergence.

**Dense Layer 1:**

Neurons: 512

Activation: ELU (Exponential Linear Unit)

Batch Normalization 1:

Batch normalization layer follows the first ELU activation.

**Dense Layer 2:**

Neurons: 1024

Activation: ELU(Exponential Linear Unit)

**epochs** - The number of training epochs. This parameter has been trained over 12000 times.

**Batch size** - The size of the mini-batch used for training. We have considered a batch size of 32 for the model 3.

**noise shape** - The shape of the noise input to the generator (1D array of size 100)

**Binary Cross-Entropy** (BCE) loss in the model.

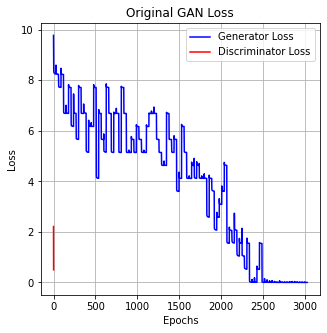
**tanh** activation function has been used in this model

A group of white squares with numbers

Description generated with high confidence

Figure ELU model at 12000 epochs

The loss of the generator and the discriminator for the ELU model are plotted down below which gives for diverse results:



Results

* The GAN model using the LeakyReLU activation function showed impressive results.
* After 30,000 training iterations, we observed significant improvements in image generation.
* The generated images displayed a remarkable level of detail and realism, closely resembling the handwritten digits in the MNIST dataset.
* This demonstrated the effectiveness of the LeakyReLU activation function and the selected architecture in producing high-quality images.
* The GAN model using the ELU activation function showed exceptionally impressive performance.
* It utilized a more complex architecture and hyperparameter tuning, resulting in images with higher detail, sharpness, and realism.
* The higher learning rate and complex dense layers, along with ELU activation, enabled the model to capture intricate dataset variations.
* The generated images closely resembled human-written digits, highlighting the versatility and potential of GANs with different activation functions and hyperparameter tuning.
* This GAN model with ELU activation demonstrated remarkable capabilities in generating high-quality images.

Conclusion

In conclusion, our experiment with Generative Adversarial Networks (GANs) using different activation functions and architectural variations has yielded promising results. We explored the effectiveness of LeakyReLU and ELU activation functions in generating high-quality images with a focus on the MNIST dataset.

The results from the LeakyReLU-based model showed impressive performance, with detailed and realistic images generated after 30,000 training iterations. This underscores the capacity of GANs to produce quality images with the right architecture and activation functions.

The ELU activation function took our experiment to the next level. By adopting a more complex architecture and fine-tuning hyperparameters, we achieved even higher levels of image detail, sharpness, and realism. The generated images closely resembled human-written digits, demonstrating the versatility and potential of GANs in different scenarios.

In essence, our experiment underscores the critical role of activation functions and architectural design in GANs. It highlights that GANs can be harnessed to generate high-quality images in various applications, provided the right combination of architectural elements and activation functions is used. Further research and experimentation can continue to unlock the full potential of GANs for image generation and other domains.