

Figure Conditional-GAN model

Report on Conditional GANs and Learning from MNIST Dataset

Name | Course Title | Date

# Introduction

Conditional Generative Adversarial Networks often abbreviated as cGANs are an extension or an advanced variation of the traditional GAN model which consisted of a generator and a discriminator. In the Conditional GANs, we provide the generator and discriminator extra information, such as class labels, attributes to give a more accurate result in the tasks related to image-to-image translation, style transfer, super-resolution, and data augmentation etc.

The MNIST dataset as we already know 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.

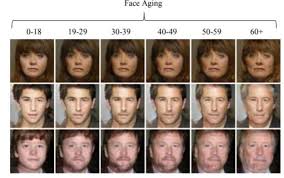


Figure Conditional-GAN used to predict facial aging. Credits ~AI-Learner

Training Process of Conditional GANs

The training process of Conditional GANs shares some similarities with traditional GANs but involves an additional level of complexity due to the conditional nature of the generator and discriminator. GANs must juggle two different kinds of training (generator and discriminator).

CGANs undergo the simultaneous training of both the generator and the discriminator of the model but with the added constraint of conditioning which means an additional input to handle and process information. The discriminator learns to classify data as real or fake while considering the conditioning information. The discriminator basically acts as a binary classifier which classifies the images into real or fake in their nature.

We keep the discriminator same during the generator training phase. Or else the generator would be trying to hit a target which is moving and might never converge.

The generator and discriminator of the CGAN networks are initially provided with random weights and the conditional information which probably is the class labels, attributes, or any other relevant information to prepare for training. The generator takes random noise (latent vectors) and the desired conditions as inputs.

During the discriminator training phase, the generator's parameters are kept constant. The discriminator is presented with pairs of real data (e.g., real images) and their associated conditions, along with generated data and corresponding conditions.

Convergence in CGANs is said to occur when the generated data satisfies the conditioning criteria and looks very real to the extent that the discriminator cannot decide it from real data. The discriminator's accuracy in classifying real and generated data becomes hazy.



Figure MNIST Output CGAN

OUR EXPERIMENT

The experiment performed today has a goal to create and evaluate the CGAN model onto the MNSIT data to generate images of all the digits and to plot the losses of the generator and the discriminator consisted of various steps including

**Dataset Selection**

The MNIST dataset has been used for this experiment, which contains the collection of hand-written digits. This dataset is often for machine learning and image processing tasks which are used very widely

**Model Architecture**

The experiment has involved the implementation of a Conditional GAN. The conditional-GAN model consists of a generator and a discriminator just like the traditional GAN model with an additional input noise.

**Conditional Generation**

Conditional-GAN Models provides additional input to both the generator and discriminator. This information is typically in the form of class labels or any other relevant attributes.

**Training the Generator:**

During the training of the Conditional-GAN, the generator learns to map the combination of the latent vector and class label to realistic images of the specified digit. The generator's loss function encourages it to produce images that not only look real but also correspond to the provided class labels.

**Loss and Optimization:**

The model considers the use of binary cross-entropy losses for training the generator and the discriminator. The Adam optimizer was used to update model parameters as an additional optimization technique.

**Model Evaluation:**

The experiment evaluated the performance of the trained cGAN by generating images of MNIST digits and assessing their quality and conditional accuracy.

Detailed Review on the MODEL1

**Network Architectures**

**For the discriminator**

The discriminator uses a convolutional neural network (CNN) based architecture with multiple convolutional layers which are as per our wish.

**LeakyReLU** activation functions are used in the hidden layers.

The last layer of the discriminator uses a **sigmoid** activation function to predict the output between zero and one.

**For the generator:**

The generator also uses a CNN architecture with **Conv2DTranspose** layers for image generation to generate the fake images.

**LeakyReLU** activation functions are used in the hidden layers.

The final output layer uses a **tanh** activation function.

The **Conditional-GAN** model combines the generator and discriminator and uses **binary cross-entropy loss**.

**Data Preprocessing**

The MNIST dataset is loaded and preprocessed. Images are normalized to the range [-1, 1] and reshaped as needed.

Training period of the model lasted for over 20 epochs ranging for about 2 hours.

**Special Techniques used for better output**

The discriminator and generator are trained alternatively in a GAN training loop.

Discriminator and generator models are updated separately. The discriminator is trained on real and fake samples (half batch each).

The generator is trained to generate samples that aim to trick the discriminator into labeling them as real.

Binary cross-entropy loss is used for both the discriminator and the Conditional-GAN.

The Adam optimizer is used for both the discriminator and the Conditional-GAN.

The generator model is saved after training into the json and h5 format to access the model with ease.

The loss graph is plotted to showcase the discriminator and the generator losses. Plotting the losses in a Conditional Generative Adversarial Network (Conditional-GAN) is a crucial step in monitoring the training progress and assessing the model's performance. By visualizing the generator and discriminator losses over training epochs.

A screenshot of a graph

Description generated with high confidence

Figure Loss graph of model 1



Figure Image generated for the model 1

Detailed Review on the MODEL2

the model 2 uses Increased filters along with sigmoid activation function by Merge image gen and label input to give more fine output.

**Network Architectures**

**For the discriminator**

The discriminator uses a convolutional neural network (CNN) based architecture with multiple convolutional layers.

RMSprop is used as the optimizer in the model.

**LeakyReLU** alpha changed to 0.3 in the hidden layers.

The last layer of the discriminator uses a **sigmoid** activation

**For the generator:**

The generator also uses a CNN architecture with **Conv2DTranspose** layers for image generation to generate the fake images.

**LeakyReLU** activation functions are used in the hidden layers.

The final output layer uses a **tanh** activation function.

The **Conditional-GAN** model combines the generator and discriminator and uses **hinge**

A screenshot of a graph

Description generated with high confidence

Figure loss graph of model 2



Figure sample generation of digit 9

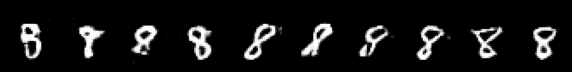


Figure sample generation of model 2

Detailed Review on the MODEL3

the model 3 uses exponential decay along with increased filters which allow for high level fine-tuning of the model inhand.

**Network Architectures**

**For the discriminator**

The discriminator uses a convolutional neural network (CNN) based architecture with multiple convolutional layers.

**LeakyReLU** alpha changed to 0.3 in the hidden layers.

The last layer of the discriminator uses a **sigmoid** activation

**For the generator:**

The generator also uses a CNN architecture with **Conv2DTranspose** layers for image generation to generate the fake images.

**Exponential decay** is used in the hidden layers.

The final output layer uses a **sigmoid** activation function.

The **Conditional-GAN** model combines the generator and discriminator and uses **mae**

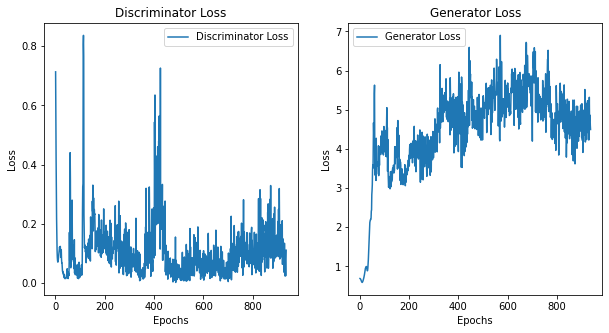
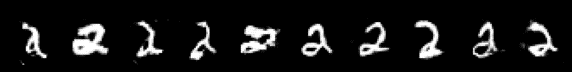
****

Figure loss graph for model 3





Results

* The discriminator model, tailored for conditional GAN (cGAN), effectively differentiates between real and generated images.
* Its architecture comprises convolutional layers, LeakyReLU activation, and fully connected layers.
* the model 3 uses exponential decay along with increased filters which allow for high level fine-tuning of the model in-hand.
* the model 2 uses Increased filters along with sigmoid activation function by Merge image gen and label input to give more fine output.
* Model 1 used The Conditional-GAN model combines the generator and discriminator and uses binary cross-entropy loss. And trained for the longest time gave the best results.

Conclusion

The experiment performed today has a goal to create and evaluate the CGAN model onto the MNSIT data to generate images of all the digits and to plot the losses of the generator and the discriminator .

In conclusion, the experiment involving a Conditional Generative Adversarial Network (Conditional-GAN) applied to the MNIST dataset has demonstrated the potential and hybrid model can showcase a lot of strength with very less epochs or training time. The model which we have seen showcased possibility to predict the mnist dataset into great detail.

In our experiment, about Conditional Generative Adversarial Networks (cGANs) applied to the MNIST dataset. The aim was to harness the potential of this advanced model to generate images that closely resemble handwritten digits, such as those found in the MNIST dataset. What stood out in our findings was the cGAN's remarkable ability to produce high-quality images even with a relatively short training period.