HAND WRITTEN DIGIT DETECTION USING GAN
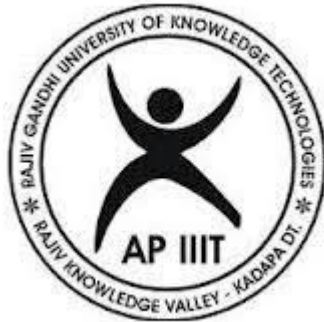
By
D.Vardhan   R170060
P.Anitha   R171019
G.Naveena   R170030

Under the Guidance of

Mr.P.Harinadha

Department of Computer Science and Engineering

Rajiv Gandhi University of Knowledge Technologies (RGUKT),
R.K.Valley , Kadapa , Andhra pradesh.

as a part of
Partial fulfillment of the degree of Bachelor of
Technology in Computer Science and Engineering

Date : 16-09-2022

<u>**CERTIFICATE OF PROJECT COMPLETION**</u>

       This is to certify that the report entitled **"Hand written Digit Detection using GAN"** submitted by **D.Vardhan** bearing ID.No. **R170060**, **P.Anitha** bearing ID.No. **R171019** and **G. Naveena** bearing ID.No. **R170030** in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out by them under my supervision and guidence.

       The report has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Project Guide,   **Head of the Department ,**

P.Harinadha,   P.Harinadha,

Assistant Professor,   Assistant Professor,

Dept of CSE,   Dept of CSE,

RGUKT RK.Valley,   RGUKT RK.Valley,

Declaration

We, hereby declare that this report entitled **"Hand Written Digit Detection Using GAN"** submitted by us under the guidance and supervision of **Mr.P.Harinadha** is a bonafied work. We also declare that it has not been submitted previously in part or in full to this university or other university or institution for the award of any degree or diploma.

We will be solenly responsible if any kind of plagiarism is found.

Date:16-09-2022   R170060,D.Vardhan
Place:RK Valley   R171019,P.Anitha
                                                   R170030,G.Naveena

Acknowledgement

We would like to express our sincere gratitude to **P.Harinadha**, Our project guide for valuable suggestions and keen interest throughout the period of project work.

We are grateful to **P.Harinadha, HOD OF CSE** for providing excellent computing facilities and a congenial atmosphere for progressing with our project.

At the outset, I would like to thank **Rajiv Gandhi University of Knowledge Technologies,RK Valley** for providing all the necessary resources for the successful completion of our project work.

We express our thanks to all those who contributed for the successfull completion of our project work.

With gratitude,

D.Vardhan   R170060
P.Anitha   R171019
G.Naveena   R170030

ABSTRACT

This is the study of Detection of Hand-written Digits(DHD) using Generative Adverserial Network(GAN).To achieve this study,initially we learned about Convolution Neural Networks (CNN) which includes 4 layers namely Convolution layer,ReLU,Pooling and Fully Connected. Later we studied regarding Generative Adverserial Network(GAN),where we majorly concentrated on Generator,Discriminator and its implementation. Also we briefly explored about MNIST Dataset. We implemented detection of hand written digit with the help of MNIST dataset and the model ran for 33 epochs.

Table of Contents

INTRODUCTION

1.1 Introduction to project

The aim of a Handwritten Digit Detection system is to convert handwritten digits into machine readable formats.This is the study of Detection of Hand-written Digits(DHD) using Generative Adverserial Network(GAN).To achieve this study,initially we learned about Convolution Neural Networks (CNN) which includes 4 layers namely Convolution layer,ReLU,Pooling and Fully Connected. Later we studied regarding Generative Adverserial Network(GAN),where we majorly concentrated on Generator,Discriminator and its implementation. Also we briefly explored about MNIST Dataset.

1.2 Importance of the Project

The main objective of this work is to ensure effective and reliable approaches for Detection of handwritten digits and make banking operations, detection of vehicle number, banks for reading cheques, post offices for arranging letter, and many other tasks easier and error free.

LITERATURE SURVEY

2.1 About CNN Architecture

CONVOLUTION : It is a mathematical combination of dataset andfeature matirx to produce a
feature map.

NEURAL NETWORK : It is an interconnection of nodes, where input is provided through input
layer and passes to hidden layers and develops output based on the
related functions.

WHAT IS CNN??

A Convolution Neural Network(CNN) is a type of Neural Network Used in Image recognition and
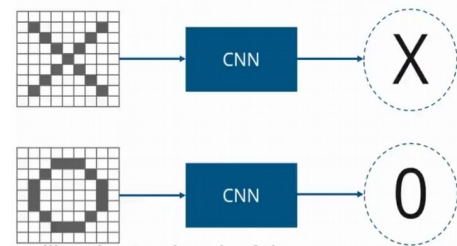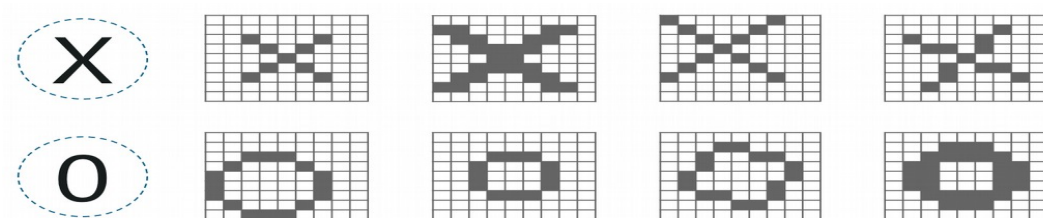Processing that is specifically designed to process pixel data.
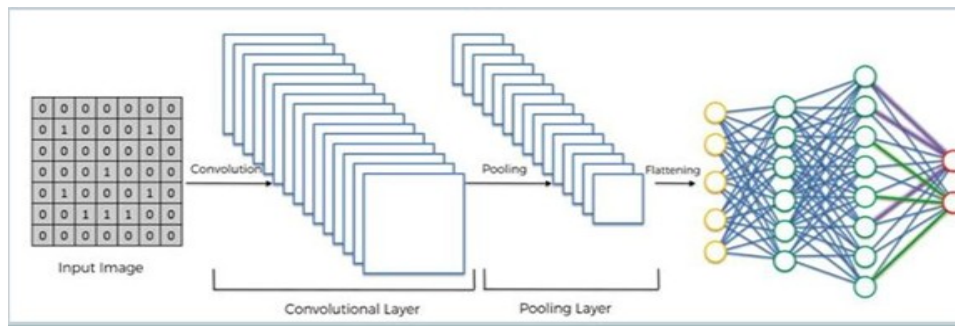
CNN LAYERS

Covolution Neural Networks have following layers:
    1.Convolution
    2.ReLU Layer
    3.Pooling
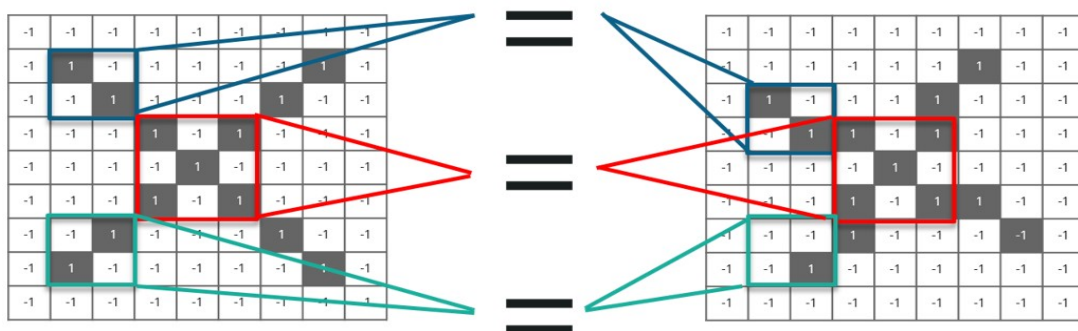   4.Fully Connected Layer

Trickier Case

The above images are deformed images of the original images,even we need to classify them with respective original image.

HOW CNN WORKS??
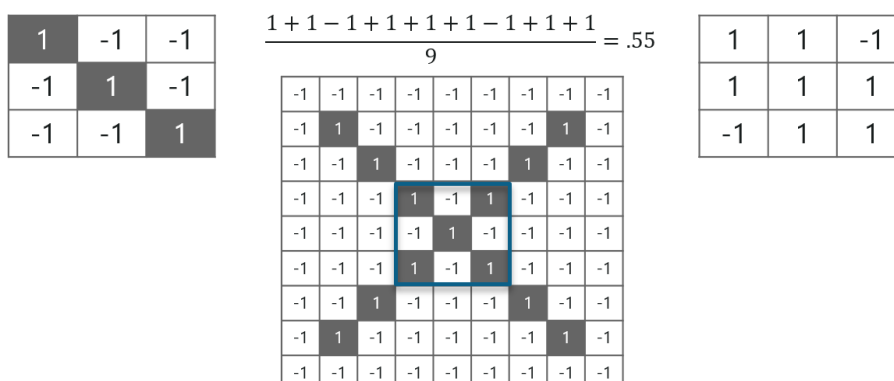
CNN compares the images piece by piece.The pieces that it looks for are called features. By finding rough feature matches, in roughly the same position in two image.
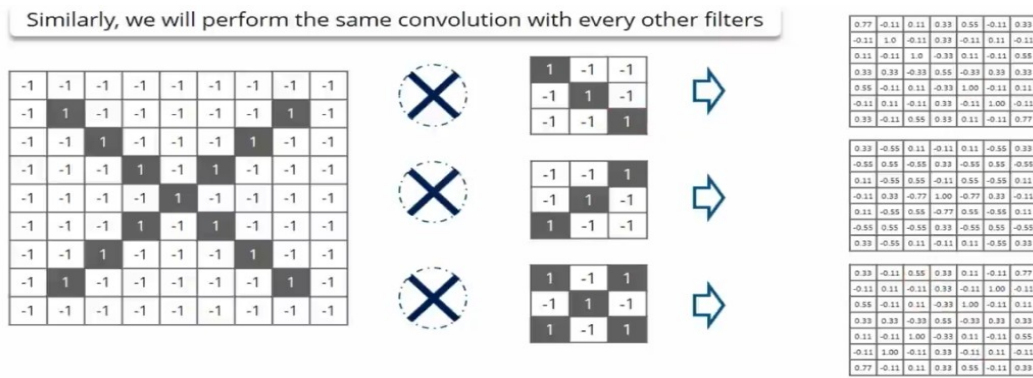

CONVOLUTIONLAYER

We execute a convolution by sliding the filter over the input. At every location, a matrix multiplication is performed and sums the result onto the feature map.
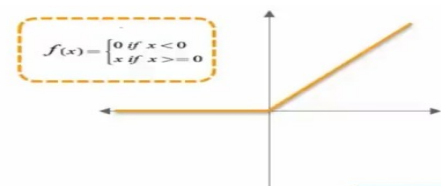
$$\frac{1+1-1+1+1+1-1+1+1}{9} = .55$$

OUTPUT OF CONVOLUTION LAYER:-

Similarly, we will perform the same convolution with every other filters

ReLU Layer

ReLU – **Rectified Linear Unit(ReLU).** In this Layer we remove every negative values from the filtered images and replaces it with Zero's.

x	f(x)=x	F(x)
-3	f(-3) = 0	0
-5	f(-5) = 0	0
3	f(3) = 3	3
5	f(5) = 5	5

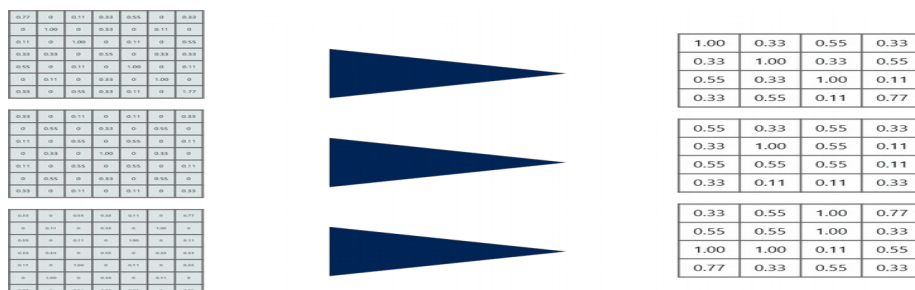$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x >= 0 \end{cases}$$

POOLING LAYER

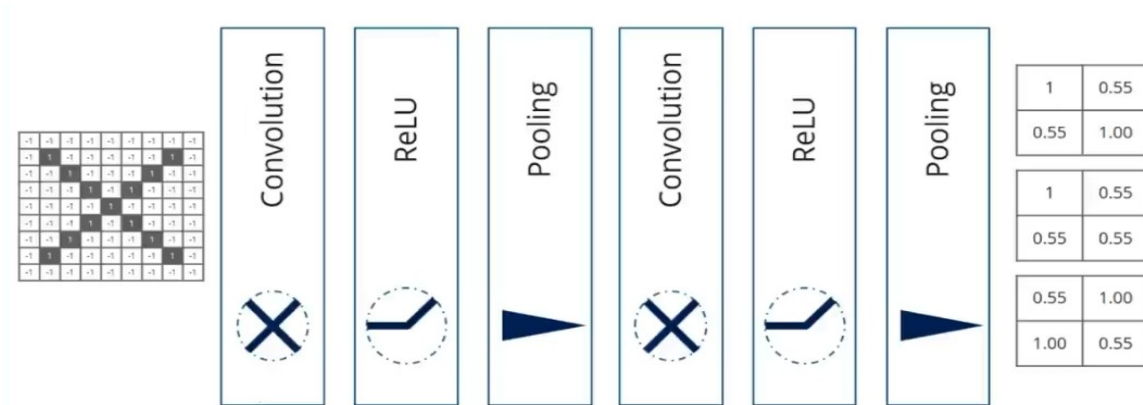In this layer we shrink the image stack into a smaller size using following steps:

1.Pick a window size(usually 2 or 3).

2.Walk your window across your filtered images.

3.From each window, take the maximum value.

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	1.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

OutPut After Passing Through Pooling Layer

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

| 0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

Stacking up the Layers

Fully Connected Layer

This is the final layer where the actual the actual classification happens. Here we take our filter and shrinked images and put them into a single list.

Output

When we feed in, 'X' and 'O'.Then there will be some element in the vector that will be high. Consider the image below, as u can see for 'X' there are different elements that are high and similarly, for 'O' we have different elements that are high.

Input Image   Vector for 'X'

Input Image   Vector for 'O'

11

.91   .51
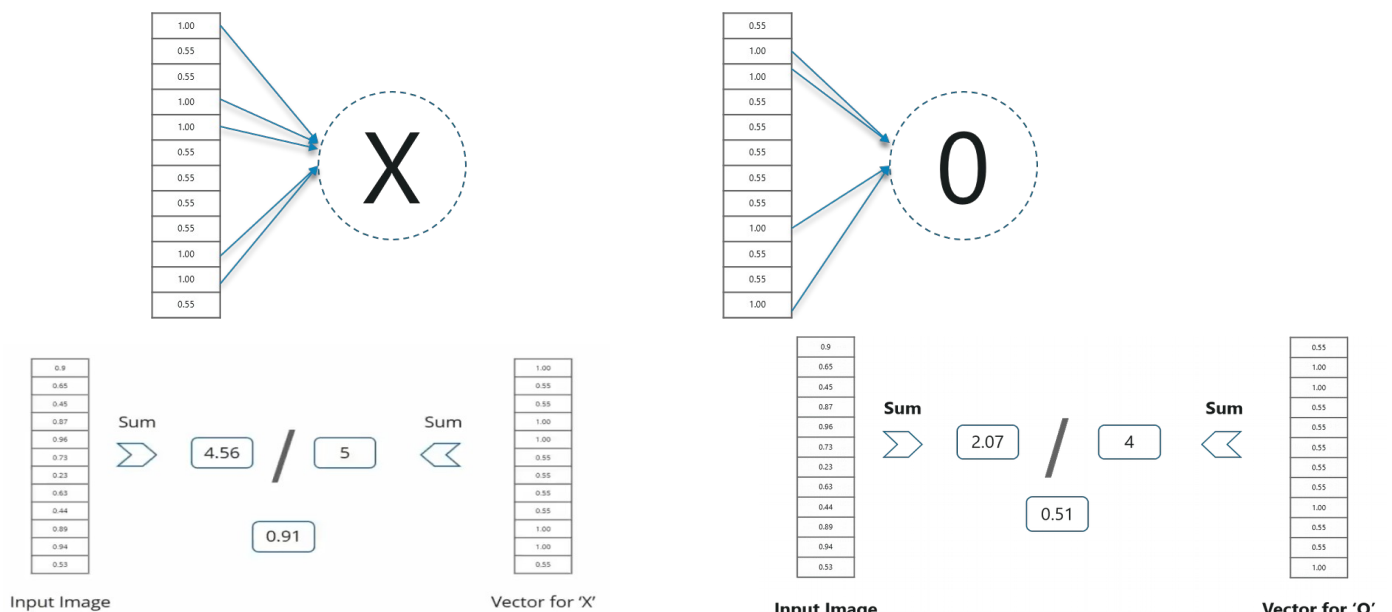
A generative adversarial network (GAN) is a machine learning (ML) model in which two neural networks(Generator and Discriminator) compete with each other to become more accurate in their predictions.

Generator generates the fake sample.

Discriminator classifies between the real sample in available dataset and fake sample generated by generator,where discriminator acts as a classifier.

Generator usually generates more realistic images,therefore to increase the classification error of discriminator.

Discriminator works in the manner to minimise the classification error.

GAN MODEL
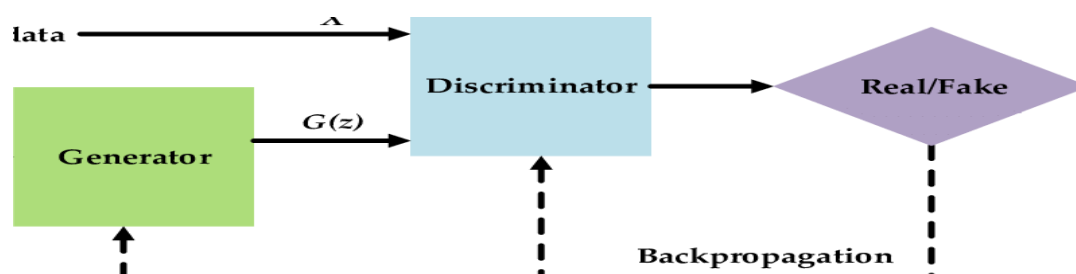
Classification Error

● Total Loss = Real Sample Loss + Fake Sample Loss
● Real Sample Loss = $E(\log(D(x)))$
● Fake Sample Loss = $E(\log(1-D(x)))$
● Total Loss = $E(\log(D(x))) + E(\log(1-D(x)))$

3.PROPOSED MODEL

3.1.NOTE ON MNIST DATASET

The MNIST database (Modified National Institute of Standards and Technology dataset) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. The MNIST database contains 60,000 training images and 10,000 testing images and it contains of handwritten digits.

Input to the Generator
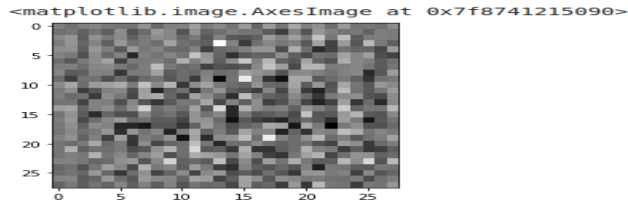
3.2.ABOUT GENERATOR MODEL

It takes random noise as an input to the model generates a low level resolution of the size 7*7*256,and we do upscaling to match the size of the image in the training dataset. We will create a neural network by using keras.sequential(). And the image available in the MNIST Dataset is of the size 28*28*1 and in order to match this size the random noise will be performed upsampling for three times(7*7*128,14*14*64,28*28*1) with the help of conv2DTranspose function available in keras.

3.3.About Discriminator model

Fake sample generated by generator and real data sample in training dataset will be given input to the discriminator model to detect the hand written dataset. We will create a neural network by using keras.sequential().We do downsampling by conv2D() function where 64 filters will be used of the size (5,5) each and downsampling is performed using strides of size (2,2), and it will be repeated for one more time and finally we flatten the data into single node and it is helpful to classify between fake and real sample using sigmoid function.

4.RESULT

Images Generated at 32 epochs
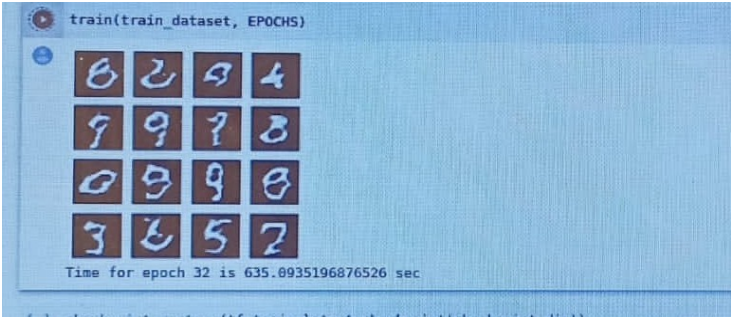
Images generated at epoch-33

5.IMPLEMENTATION OF MODEL

```python
import tensorflow as tf
from tensorflow.keras.layers import (
                                      Dense,
                                      BatchNormalization,
                                      LeakyReLU,
                                      Reshape,
                                      Conv2DTranspose,
                                      Conv2D,
                                      Dropout,
                                      Flatten)
import matplotlib.pyplot as plt
```

```python
(train_images,train_labels),(_, _) = tf.keras.datasets.mnist.load_data()

train_images = train_images.reshape(train_images.shape[0],28,28,1).astype('float32')
train_images = (train_images-127.5)/127.5

BUFFER_SIZE  = 60000
BATCH_SIZE = 256

train_dataset = tf.data.Dataset.from_tensor_slices(train_images).shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
```

```python
def generator_model():
    model = tf.keras.Sequential()
    model.add(Dense(7*7*256, use_bias=False, input_shape=(100,)))
    model.add(BatchNormalization())
    model.add(LeakyReLU())

    model.add(Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256) # Note: None is the batch size

    model.add(Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)
    model.add(BatchNormalization())
    model.add(LeakyReLU())

    model.add(Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same', use_bias=False))
    assert model.output_shape == (None, 14, 14, 64)
    model.add(BatchNormalization())
    model.add(LeakyReLU())

    model.add(Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same', use_bias=False, activation='tanh'))
    assert model.output_shape == (None, 28, 28, 1)

    print(model.summary())

    return model
def discriminator_model():
    model = tf.keras.Sequential()

    model.add(Conv2D(64, (5, 5), strides=(2, 2), padding='same', input_shape=[28, 28, 1]))
    model.add(LeakyReLU())
    model.add(Dropout(0.3))

    model.add(Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
    model.add(LeakyReLU())
    model.add(Dropout(0.3))

    model.add(Flatten())
    model.add(Dense(1))

    print(model.summary())

    return model
```

```python
discriminator = discriminator_model()
```

```python
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)

def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.ones_like(real_output), real_output)
    fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
    total_loss = real_loss + fake_loss
    return total_loss

def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)

#to update the weights during backpropagation
generator_optimizer = tf.keras.optimizers.Adam(1e-4)
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)
```

```python
def generate_and_save_images(model, epoch, test_input):
  # Notice `training` is set to False.
  # This is so all layers run in inference mode (batchnorm).
  # 1 - Generate images
  predictions = model(test_input, training=False)
  # 2 - Plot the generated images
  fig = plt.figure(figsize=(4,4))
  for i in range(predictions.shape[0]):
      plt.subplot(4, 4, i+1)
      plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='gray')
      plt.axis('off')
  # 3 - Save the generated images
  plt.savefig('image_at_epoch_{:04d}.png'.format(epoch))
  plt.show()
```

```python
import time
from IPython import display # A command shell for interactive computing in Python.

def train(dataset, epochs):
  # A. For each epoch, do the following:
  for epoch in range(epochs):
    start = time.time()
    # 1 - For each batch of the epoch,
    for image_batch in dataset:
      # 1.a - run the custom "train_step" function
      # we just declared above
      train_step(image_batch)

    # 2 - Produce images for the GIF as we go
    display.clear_output(wait=True)
    generate_and_save_images(generator,
                             epoch + 1,
                             seed)

    # 3 - Save the model every 5 epochs as
    # a checkpoint, which we will use later
    if (epoch + 1) % 5 == 0:
      checkpoint.save(file_prefix = checkpoint_prefix)

    # 4 - Print out the completed epoch no. and the time spent
    print ('Time for epoch {} is {} sec'.format(epoch + 1, time.time()-start))

  # B. Generate a final image after the training is completed
  display.clear_output(wait=True)
  generate_and_save_images(generator,
                           epochs,
                           seed)
```

6. Conclusion

In this study we presented Detection of Hand Written Digits using GAN with help of MNIST dataset.In order to understand the implementation of GAN we studied the basics required like Convolution neural Network CNN, Backpropagation and Activation functions like sigmoid function to the classify the image that is real or fake.We observed how Backpropagation is performed with the help of Adam optimiser,upsampling and downsampling is performed in Generator and Discriminator respectively.

7.Future Scope

We ran the model initially for 32 epochs and later for 33 epochs and further we need to run the model for 60 epochs to get more clarity of an image. If in case still the clarity of images is not satisfied then we need to run the model for more than 60 epochs.

8.References

- Nptel – Prof.Prabir Kumar Biswas(IIT KGP)
- Youtube
- Edureka channel
- Google
- Geeks for Geeks