In [1]:

```
"""
Fake Image Generation using GAN
Author: Amruth Karun M V
Date: 13-Nov-2021
"""

import imageio
import glob
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
from IPython import display
import warnings
warnings.filterwarnings('ignore')

BATCH_SIZE = 32
NUM_FEATURES = 100


def load_data():
    """
    Loads MNIST digits dataset and plots
    sample images
    Arguments: None
    Returns: MNIST dataset
    """
    (x_train, _), (x_test, _) = tf.keras.datasets.mnist.load_data()
    x_train = x_train.astype(np.float32) / 255.0
```

```
    x_test = x_test.astype(np.float32) / 255.0
    plt.figure(figsize =(10, 10))
    for i in range(25):
        plt.subplot(5, 5, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.imshow(x_train[i], cmap = plt.cm.binary)
    plt.show()
    return x_train, x_test


def create_batch(x_train):
    """
    Creates a batch dataset from the train dataset
    Arguments:
        x_train   -- train dataset
    Returns: Batch dataset
    """
    dataset = tf.data.Dataset.from_tensor_slices(x_train).shuffle(1000)
    dataset = dataset.batch(BATCH_SIZE, drop_remainder = True).prefetch(1)
    return dataset


def load_gan_model():
    """
    Loads the GAN architecture with
    generator and discriminator layers
    Arguments: None
    Returns: GAN model
    """
```

```
generator = keras.models.Sequential([
    keras.layers.Dense(7 * 7 * 128, input_shape =(NUM_FEATURES,)),
    keras.layers.Reshape([7, 7, 128]),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2DTranspose(64, (5, 5), (2, 2), padding ="same", activation ="relu"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2DTranspose(1, (5, 5), (2, 2), padding ="same", activation ="tanh")
])
generator.summary()

discriminator = keras.models.Sequential([
    keras.layers.Conv2D(64, (5, 5), (2, 2), padding ="same", input_shape =[28, 28, 1]),
    keras.layers.LeakyReLU(0.2),
    keras.layers.Dropout(0.3),
    keras.layers.Conv2D(128, (5, 5), (2, 2), padding ="same"),
    keras.layers.LeakyReLU(0.2),
    keras.layers.Dropout(0.3),
    keras.layers.Flatten(),
    keras.layers.Dense(1, activation ='sigmoid')
])
discriminator.summary()
discriminator.compile(loss ="binary_crossentropy", optimizer ="adam")
# make discriminator no-trainable as of now
discriminator.trainable = False
# Combine both generator and discriminator
gan = keras.models.Sequential([generator, discriminator])
# compile generator using binary cross entropy loss and adam optimizer
gan.compile(loss ="binary_crossentropy", optimizer ="adam")
return gan
```

```
def train_dcgan(gan, dataset, num_features, epochs = 5):
    """
    Trains the Deep Convolutional Generative
    Adverarial Network (DCGAN)
    Arguments:
        gan          -- GAN network
        dataset      -- train dataset
        num_features -- No. of input features
        epochs       -- No. of training iterations
    Returns: Generated images for each epoch
    """
    generator, discriminator = gan.layers
    for epoch in tqdm(range(epochs)):
        print("\nEpoch {}/{}".format(epoch + 1, epochs))
        for X_batch in dataset:
            noise = tf.random.normal(shape =[BATCH_SIZE, num_features])
            generated_images = generator(noise)
            X_fake_and_real = tf.concat([generated_images, X_batch], axis = 0)
            y1 = tf.constant([[0.]] * BATCH_SIZE + [[1.]] * BATCH_SIZE)
            discriminator.trainable = True
            discriminator.train_on_batch(X_fake_and_real, y1)
            noise = tf.random.normal(shape =[BATCH_SIZE, num_features])
            y2 = tf.constant([[1.]] * BATCH_SIZE)
            discriminator.trainable = False
            gan.train_on_batch(noise, y2)

        # generate images for the GIF as we go
        seed = tf.random.normal(shape =[BATCH_SIZE, 100])
        generate_and_save_images(generator, epoch + 1, seed)
```

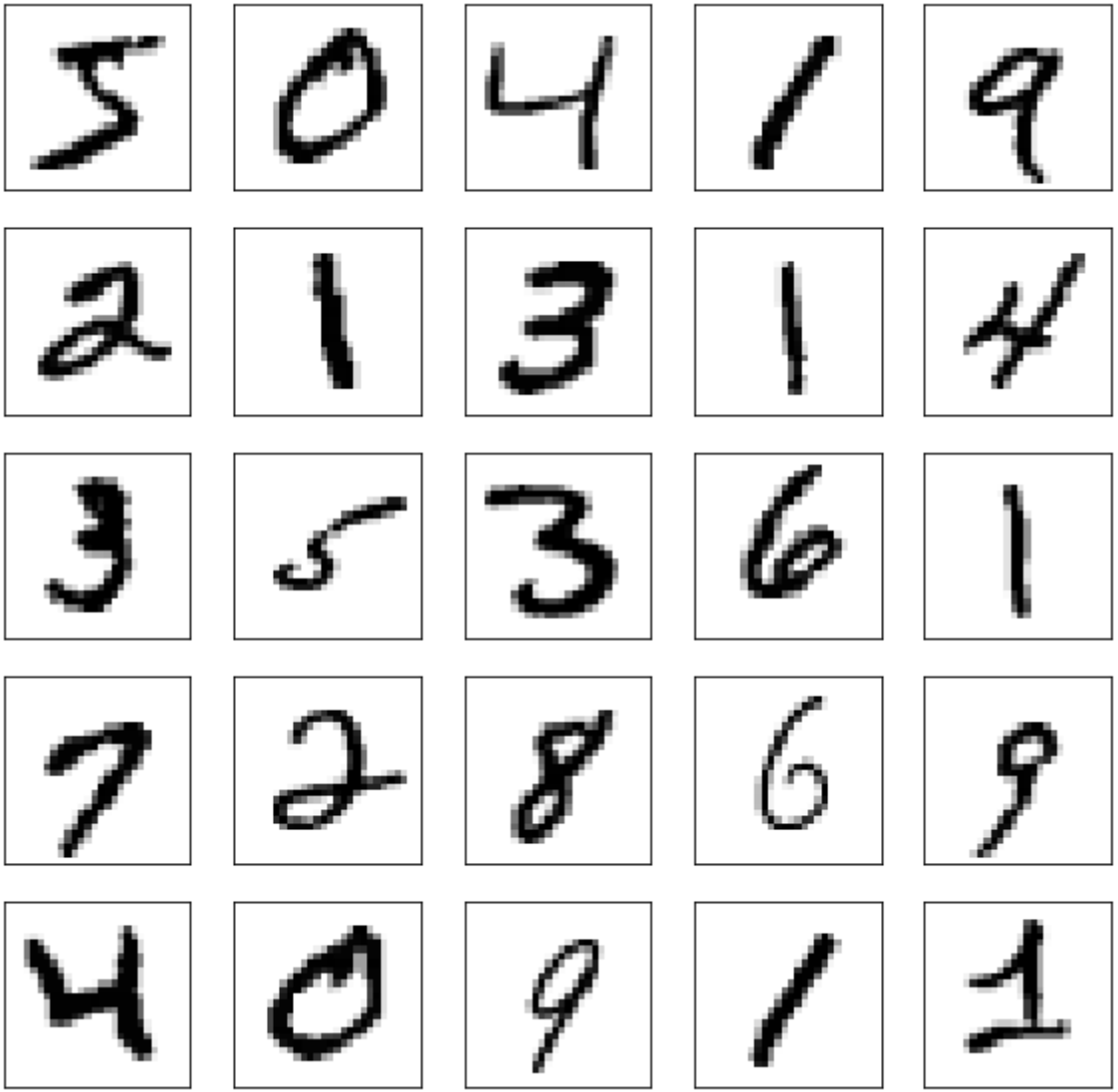
```
def generate_and_save_images(model, epoch, test_input):
    """
    Generate digit images using network predictions
    and plot the results for each epoch
    Arguments:
        model      -- generator model
        epoch      -- input epoch
        test_input -- random test input
    Returns: Plots generated images
    """
    predictions = model(test_input, training = False)
    fig = plt.figure(figsize =(10, 10))
    for i in range(25):
        plt.subplot(5, 5, i + 1)
        plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap ='binary')
        plt.axis('off')
    plt.savefig('image_epoch_{:04d}.png'.format(epoch))
```

In [2]:

```
x_train, x_test = load_data()
x_train_dcgan = x_train.reshape(-1, 28, 28, 1) * 2. - 1.
dataset = create_batch(x_train_dcgan)
gan = load_gan_model()
train_dcgan(gan, dataset, NUM_FEATURES, epochs = 50)


anim_file = 'dcgan_results.gif'
with imageio.get_writer(anim_file, mode ='I') as writer:
    filenames = glob.glob('image*.png')
    filenames = sorted(filenames)
    last = -1
    for i, filename in enumerate(filenames):
        frame = 2*(i)
        if round(frame) > round(last):
            last = frame
        else:
            continue
        image = imageio.imread(filename)
        writer.append_data(image)
display.Image(filename = anim_file)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 6272)              633472
_________________________________________________________________
reshape (Reshape)            (None, 7, 7, 128)         0
_________________________________________________________________
batch_normalization (BatchNo (None, 7, 7, 128)         512
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 14, 14, 64)        204864
_________________________________________________________________
batch_normalization_1 (Batch (None, 14, 14, 64)        256
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 28, 28, 1)         1601
=================================================================
Total params: 840,705
Trainable params: 840,321
Non-trainable params: 384
_________________________________________________________________
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 14, 14, 64)        1664
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 14, 14, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 64)        0
_________________________________________________________________
```

```
conv2d_1 (Conv2D)            (None, 7, 7, 128)         204928
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 7, 7, 128)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 6273
=================================================================
Total params: 212,865
Trainable params: 212,865
Non-trainable params: 0
_________________________________________________________________


  0%|          | 0/50 [00:00<?, ?it/s]

Epoch 1/50
```