Assignment 3

Specifications: Google cloud, MATLAB, GPU Tesla K80

Question 1: Use GAN network to generate an image

A GAN consists of two networks that train together:

1. **Generator** — Given a vector of random values (latent inputs) as input, this network generates data with the same structure as the training data.
2. **Discriminator** — Given batches of data containing observations from both the training data, and generated data from the generator, this network attempts to classify the observations as "real" or "generated".

I have used Cifar100's one folder with 'apple' images as a dataset. It was having overall 500 images. So, firstly I did the data augmentation process on the dataset by rescaling and flipping the images and resized these images to 64x64x3. Then, passed it through generator whose configuration can be seen in figure 1.

```
>> GAN
Load the cifar100 'apple' folder images to train in GAN network.
Augment the data images by resizing, flipping images
Configuration of Generator to generate images from random values
  15x1 Layer array with layers:

     1   'input'             Image Input              1x1x50 images
     2   'transposedConv1'   Transposed Convolution   800 4x4 transposed convolutions with stride [1  1] and cropping [0  0  0  0]
     3   'batchNorm1'        Batch Normalization      Batch normalization
     4   'relu1'             ReLU                     ReLU
     5   'transposedConv2'   Transposed Convolution   400 4x4 transposed convolutions with stride [2  2] and cropping [1  1  1  1]
     6   'batchNorm2'        Batch Normalization      Batch normalization
     7   'relu2'             ReLU                     ReLU
     8   'transposedConv3'   Transposed Convolution   200 4x4 transposed convolutions with stride [2  2] and cropping [1  1  1  1]
     9   'batchNorm3'        Batch Normalization      Batch normalization
    10   'relu3'             ReLU                     ReLU
    11   'transposedConv4'   Transposed Convolution   100 4x4 transposed convolutions with stride [2  2] and cropping [1  1  1  1]
    12   'batchNorm4'        Batch Normalization      Batch normalization
    13   'relu4'             ReLU                     ReLU
    14   'transposedConv5'   Transposed Convolution   3 4x4 transposed convolutions with stride [2  2] and cropping [1  1  1  1]
    15   'tanhLayer'         Tanh                     Hyperbolic tangent
```

Figure 1. Generator network

So, I forwarded the random valued input for the generator with the size as 1x1x50. I used 5 transposed convolutional layers with 4x4 sized filter followed with batch normalization and ReLu layer. Then we designed a discriminator network which distinguished between real and generated images which is shown in figure 2. It consists of 5 convolutional layers, followed with batch normalization and leaky ReLu layers.

```
Configuration of Discriminator that classifies real and generated images
  13x1 Layer array with layers:

     1   'input'        Image Input           100x100x3 images
     2   'conv1'        Convolution           100 4x4 convolutions with stride [2  2] and padding [1  1  1  1]
     3   'leakyrelu1'   Leaky ReLU            Leaky ReLU with scale 0.3
     4   'conv2'        Convolution           200 4x4 convolutions with stride [2  2] and padding [1  1  1  1]
     5   'batchnorm1'   Batch Normalization   Batch normalization
     6   'leakyrelu2'   Leaky ReLU            Leaky ReLU with scale 0.3
     7   'conv3'        Convolution           400 4x4 convolutions with stride [2  2] and padding [1  1  1  1]
     8   'batchnorm2'   Batch Normalization   Batch normalization
     9   'leakyrelu3'   Leaky ReLU            Leaky ReLU with scale 0.3
    10   'conv4'        Convolution           800 4x4 convolutions with stride [2  2] and padding [1  1  1  1]
    11   'batchnorm3'   Batch Normalization   Batch normalization
    12   'leakyrelu4'   Leaky ReLU            Leaky ReLU with scale 0.3
    13   'conv5'        Convolution           1 4x4 convolutions with stride [1  1] and padding [0  0  0  0]
Train images through GAN network
Generate new images
```

Figure 2. Discriminator network.

So, the training parameters for these two networks has been set as below:

Number of epochs: 1000
Mini-batch size: 200
Learning rate: 0.0002
Gradient decay factor: 0.45
Execution Environment: GPU

After designing the networks, we have to train the GAN For each epoch, shuffle the datastore and loop over mini-batches of data. For each mini-batch, first rescaled the images in the range [-1 1], converted the data to dlarray objects with underlying type single and specify the dimension labels 'SSCB' (spatial, spatial, channel, batch) and generated a dlarray object containing an array of random values for the generator network. Later, For GPU training, converted the data to gpuArray objects. After this conversion, Evaluate the model gradients using dlfeval and the modelGradients function. Update the network parameters using the adamupdate() function. Plot the scores of the two networks. After every validationFrequency iterations, a batch of generated images will be displayed for a fixed held-out generator input. This program took around 4hours to run.

So, the newly generated images can be seen in below figure 3.

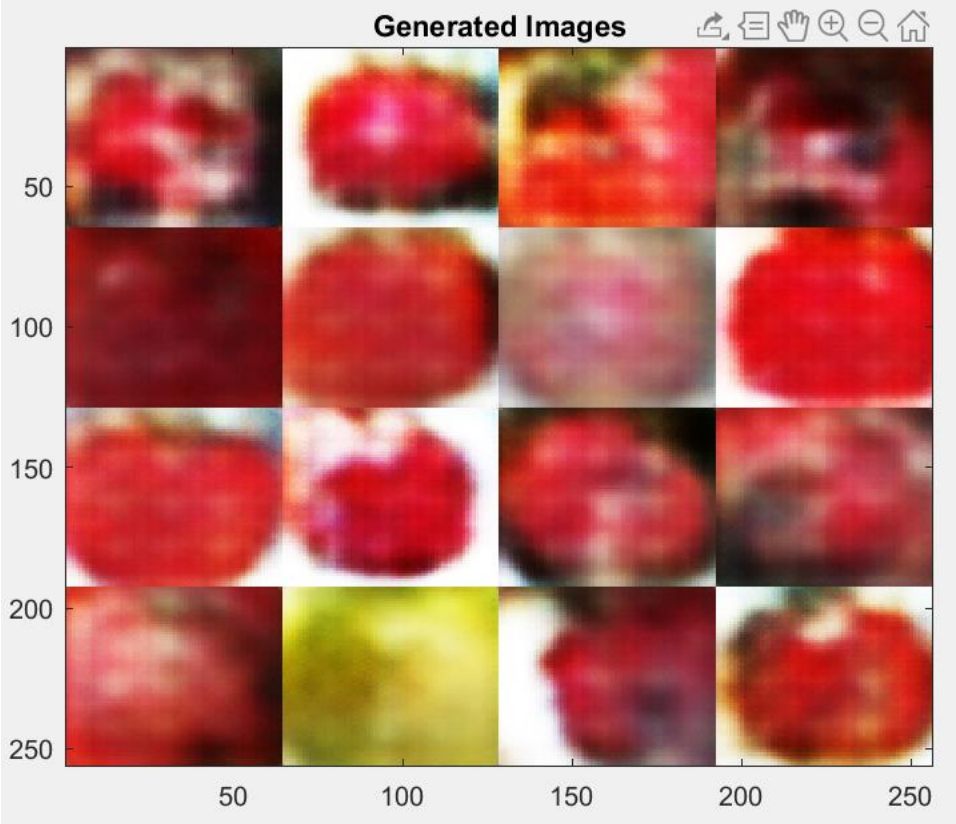

Figure 3. Generated images through GAN network.

Question 3: LSTM for time series problem recognition

To forecast the values of future time steps of a sequence, we can train a sequence-to-sequence regression LSTM network, where the responses are the training sequences with values shifted by one-time step. That is, at each time step of the input sequence, the LSTM network learns to predict the value of the next time step. To forecast the values of multiple time steps in the future, use the predictAndUpdateState() function to predict time steps one at a time and update the network state at each prediction.

I have used Covid-19 dataset and have considered death rates as a potential label for prediction in future. Firstly, I converted the dates into numerical format. And drew a graph of time vs death cases in hourly basis. It can be seen in figure 4. Then I manipulated the duplicated data and merged the same date data into one row likewise over whole dataset. Then normalized the data because the death cases attribute was having random numbers. I applied a network of 3 LSTM with 500,250,150 hidden neurons respectively and 3 fully connected layers with 100, 36 hidden neurons and 1 in the last layer as output to give it as an input to regression layer. This layer structure can be seen in figure 5.

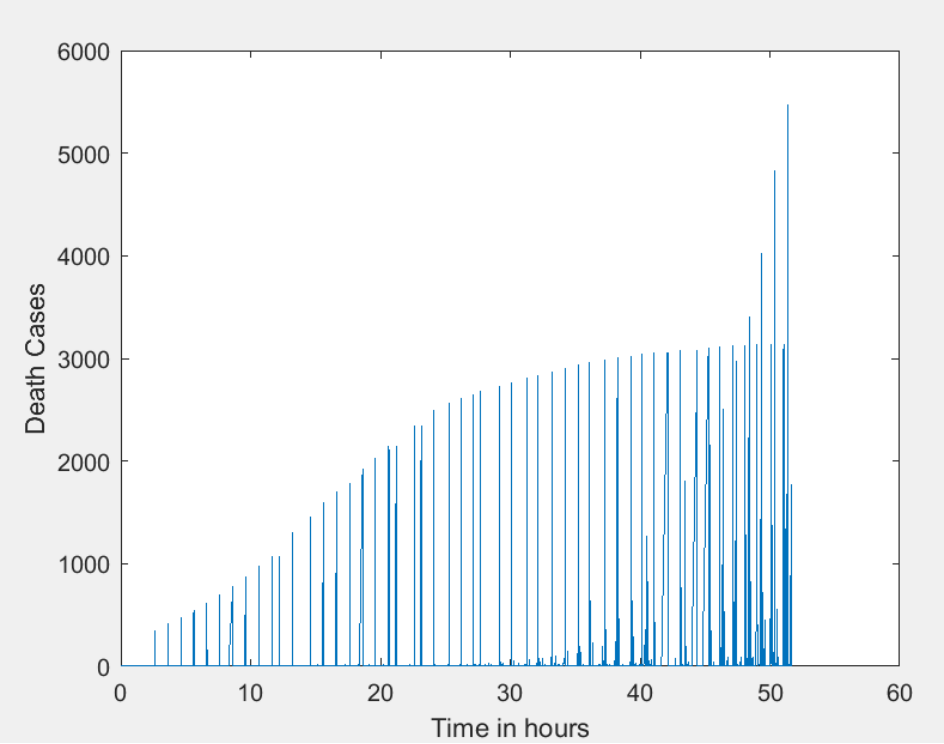

Figure 4: Time vs death cases

```
8x1 Layer array with layers:

    1   ''   Sequence Input      Sequence input with 1 dimensions
    2   ''   LSTM                LSTM with 500 hidden units
    3   ''   LSTM                LSTM with 250 hidden units
    4   ''   LSTM                LSTM with 150 hidden units
    5   ''   Fully Connected     100 fully connected layer
    6   ''   Fully Connected     36 fully connected layer
    7   ''   Fully Connected     1 fully connected layer
    8   ''   Regression Output   mean-squared-error
```

Figure 5. Time series prediction model summary

We trained this network with below training parameters:

Number of epochs: 300
Learning rate: 0.001
LearningRateDropPeriod: 50
LearningRateDropFactor: 0.5

This gave us a training progress plot in figure 6.

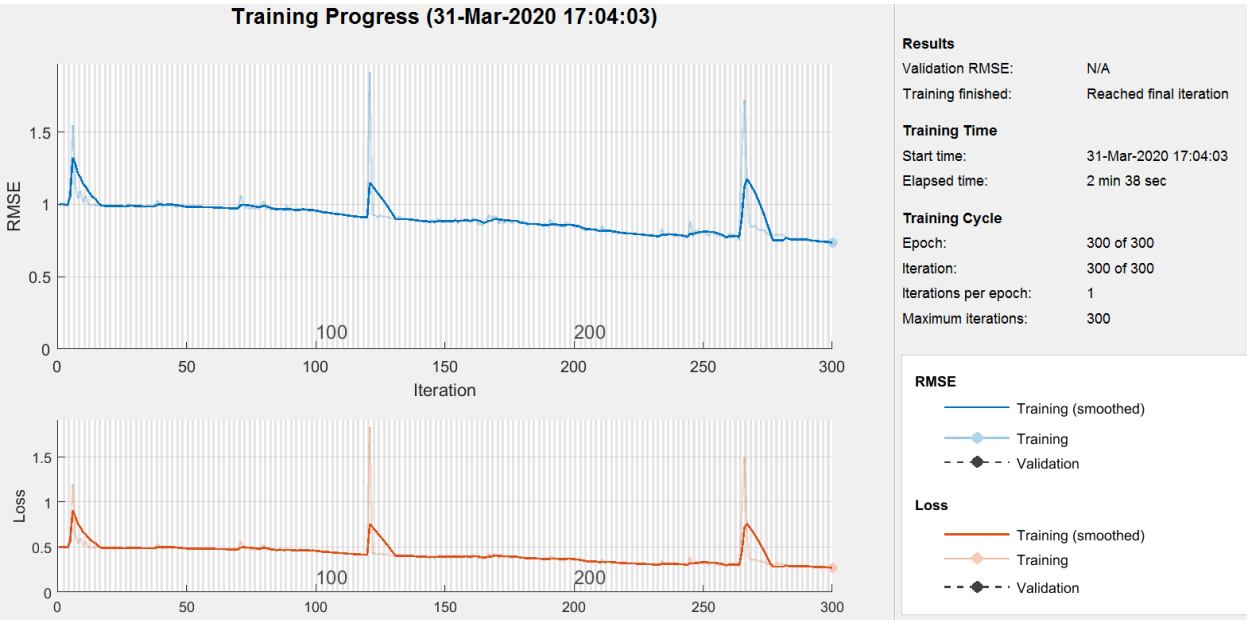

Figure 6: Training a LSTM network

Then to forecast the values of multiple time steps in the future, we used predictAndUpdateState function, and which gave us an output something like in figure 7.

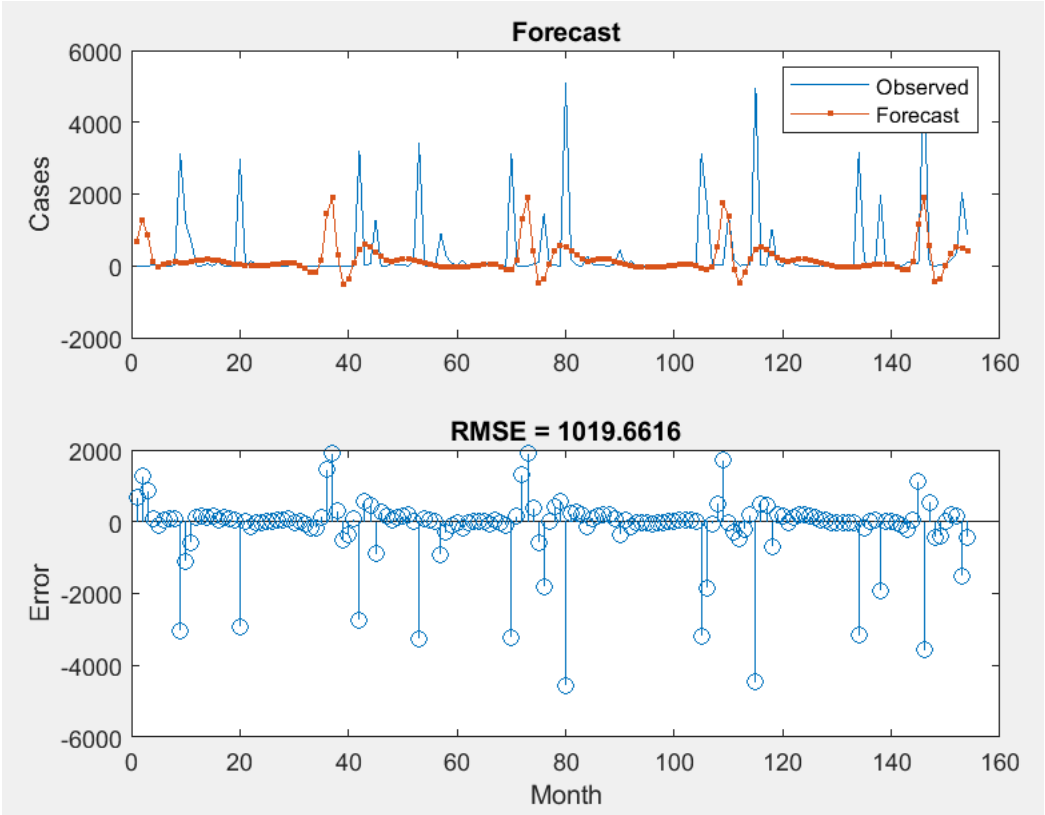

Figure 7. Forecasted time steps in future according to observed data.

So, to compare the predicted time steps with observed values you can look at the figure 8.

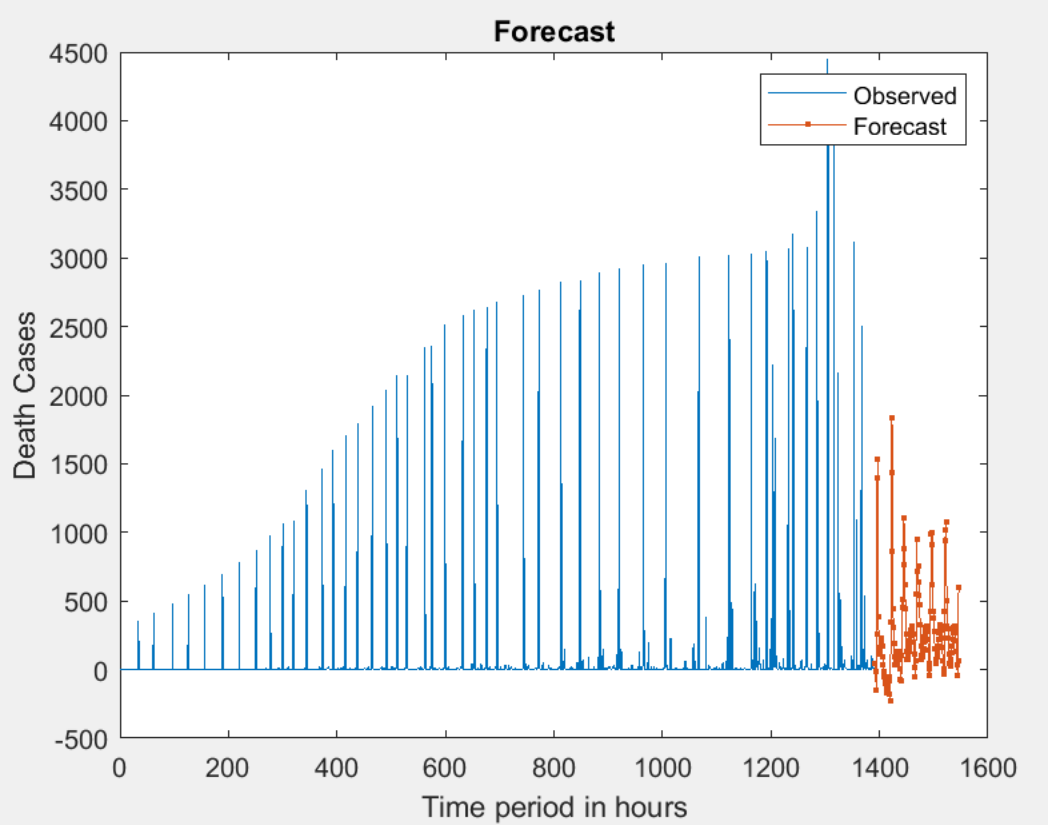

Figure 8. Comparison of predicted and observed time steps for death cases

This network gave us around 1019.6616 RMSE which can be improved by modifying the trainable parameters or by modifying the network that we have. This program took around 5minutes to run.

Step to execute the programs:

1. You can directly run these two '1095709_GAN.m', '1095709_LSTM.m' programs, by using the respective image folders.