Final Report
CMPSC 448-001
Achyut Dudhat

Task, Dataset and preprocessing:

Task: To create two types of Deep Learning algorithms, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), to address tasks of computer vision (i.e., Image analysis) and capture information based on previous inputs in the sequence respectively.

Dataset: The dataset chosen for creating the Deep Learning Algorithm is MNIST (Modified National Institute of Standards and Technology) data. This is a dataset consisting of digits from 0 to 9 in the various handwritings in the form of images. Originally, the dataset was as a 128x128 binary (black and white) images but was processed/normalised to fit into a 28x28 pixel grayscale image.

Preprocessing: Using the X_Train, Y_Train, X_Test, and Y_Test variables the dataset was loaded from the keras library of tensorflow. Following that, in order to check the size of train and test data, len(variable_name) was implemented to check the appropriate size of data assigned to each variable. In X_Train and Y_Train, we have 60,000 trained digits images for both and similarly, we have 10,000 images for X_Test and Y_Test which is a decent dataset size.

After assigning and knowing the variables with their size, matplotlib library was employed to check the contents for X variables (i.e, images) and Y variables (i.e., labels). The images need to have normalised pixel values between 0 and 1 by dividing the X_Train and X_Test by 255.

Since, the Convolutional Neural Network (CNN) is predominantly used for image processing (i.e., it can use two dimensional images), the dataset is great, but, the Recurrent Neural Network (RNN) is predominantly used for sequential data. However, the Recurrent Neural Network (RNN) can still be used for images by flattening the image into a one dimensional sequence. For example, 4x4 image can be flattened to a sequence of length 16.

Similarly, the training dataset was flattened for RNN because we want to convert the 28x28 image into a single dimensional array that will have 784 elements using pandas' function called 'reshape'.

Implementation of RNN and CNN

Initially, Sequential from keras library is utilised to create an instance of a sequential neural network model

CNN: The goal is to keep reducing the inputs into smaller inputs by process of convolution and pooling when appropriate and repeat the process twice and then pass it to the dense layer with appropriate activation. Initially, a two dimensional convolutional layer will be added to the neural network model for image processing tasks with thirty two 3x3 filters. The rectified linear unit (ReLu) activation function is used to bring in non-linearity to the model of shape 28x28 with 1 colour. Then a two dimensional max-pooling layer is added to the model in order to reduce the size of computation with a 2x2 pooling window. This process of convoluting with sixty four 3x3 filters with ReLu activation is done and a two dimensional max pooling is done again. Now, a flatten layer is employed which is used for taking the outputs from convolving and pooling layers and converting a one dimensional layer to a fully connected layer in neural networks. Finally, a dense layer is used where there are 10 neurons in the network with a softmax activation which normalises or converts the output scores into probabilities.

RNN: The goal is to model previous sequential inputs from data with appropriate activation. Initially, an input layer is added to the neural network model from keras library with 28x28 pixels as it's a one dimensional model taking sequences. Following the input, a Simple Recurrent Neural Network (Simple RNN) is added to the model with 164 number of neurons in the layer and rectified linear unit (ReLu) activation to introduce the non-linearity to the neural network model. Now, a flatten layer is employed which is used for taking the outputs from the SimpleRNN to fully connected layers in neural networks. Finally, a dense layer is used where there are 10 neurons in the network with a softmax activation which normalises or converts the output scores into probabilities.

Training Details of Two Deep Learning systems:

In order to successfully train the CNN and RNN models, we have successfully compiled them. For compilation of these two models, 'compile' was used to put together the model after implementation of CNN and RNN with various parameters. Adam's optimizer (stochastic gradient descent) is, a learning rate optimization technique in deep learning which normalises the parameters by changing the weights of a neural network during training, used along with loss function of SparseCategoricalCrossentropy (setting from_logits to True) which is used for classification purpose since, the data consists of various handwritten images from 0 to 9. With this cross entropy loss function, the labels are expected to be integers. Following this, model.fit will be employed with a trained dataset consisting of X_Train and Y_Train to test on a cross validation set consisting of X_Test and Y_Test. Both the model will be trained for 5 iterations implying it will go through the training dataset 5 times in order to minimise the loss function. Ultimately, ‘evaluate’ will be utilised for evaluation of the model with the ‘verbose’ parameter as 2 for displaying both the loss as well as the accuracy in the evaluation.

Results and Observation

Results: For CNN, the test accuracy is ~98.6% at 5 iterations. While for RNN, the test accuracy comes at ~95% at 5 iterations.

Observations: One interesting factor to note is that since the dataset is about image processing where CNN algorithm is predominantly used, CNN has better accuracy. As RNN yields better results for Sequential or Text data it allows CNN to perform better. The other interesting observation would be that after training a model for more than threshold times (i.e., iterations), the accuracy of the testing data does not change significantly.

Conclusions***:*** After certain iterations, the accuracy does not change significantly implying a reasonable number of iterations has to be used. The reason being that if the iterations are more and the result is similar, then more computation and time is being used for the same accuracy which is not fruitful.

Challenges and Obstacles/Problems with Solutions :

1) Using a reasonable number of iterations for the evaluation because if iterations are excessive, then more computation power and time is being used for the same result. Hence, running the algorithm for certain iterations and then changing the number of iterations to see from where the results are stagnant.

2) For CNN, choosing how many times to Convol and Maxpool is crucial in order to ensure that inputs are not being altered to an extent that it might not be accurate to train and then test.

3) For RNN, figuring out when to use GRU, SimpleRNN and LSTM as they are dependent on the flexibility of the type of dataset being used along with appropriate number of neurons passed as parameters.

4) Using an appropriate library is a must as there are many libraries but choosing a library based on the dataset is a crucial lesson. Initially, different functionality of various libraries were used but at the end the most fruitful result was provided by tensorflow along with matplotlib and numpy.

References:

1) Canvas Slides and Class Lectures, CMPSC 448, Section 001: Mach Learning and Ai (22381--UP---P-CMPSC---448-------001-): Pages (instructure.com)

2) Tensorflow Documentation, https://www.tensorflow.org/guide/keras