Lab Assignment 1: CNNs and RNNs

Due Date: Sunday September 29, 2024

Purpose:

The purpose of this Lab assignment is to:

1. To get hands-on experience of applying Deep Neural Networks, namely CNNs and RNNs to solve a classification problem

General Instructions:

Be sure to read the following general instructions carefully:

1. This assignment must be completed individually by all students.
2. Only provide the requested screenshots and make sure to have a complete screenshot, partial screenshots will not earn any marks.
3. You will have to provide a **demonstration video for your solution** and upload the video together with the solution on **eCenntenial** through the assignment link. See the **video recording instructions** at the end of this document.
4. In your 5-minute demonstration video you should explain your solution clearly, going over the main code blocks and the purpose of each module/class/method also demoing the execution of exercises #1. YouTube links and links to google drive or any other media are not acceptable, the actual recording must be submitted.
5. Any submission without an accompanying video will lost 70% of the grade.
6. In your analysis report make sure you provide an introduction and clearly state the facts and findings. Any submission missing Analysis report will lost 70%.

Submission:

There are three elements to be submitted for this assignment in one zipped folder (All subject to grading as per rubric for this assignment):

1. For each exercise that require code, please create a project folder and include all project python scripts/modules and screenshot of output, as needed. Name all python scripts your firstname_linear.py. Name the folder "Exercise#X_firstname", where X is the exercise number and firstname is your first name. (In total 1 folders for this assignment).
2. For all questions that require written or graphic response create one "Word document" and indicate the exercise number and then state your response. Name the document "Written_response_firstname", where firstname is your firstname. (In total on word or pdf document).

3. All submissions need to be accompanied with a recorded demonstration video not to exceed 5 minutes in length, focus on showing the key code functionalities and run the code.

Create on zipped folder containing all of the above, name it lab1assignment_firstname where firstname is your firstname.

Assignment – exercises:

1. **Exercise #1:** Fashion MNIST and CNN/RNN (100 marks)

   Requirements:

   a. Get the data:
      1. Import and load the 'fashion_mnist' dataset from TensorFlow. Using 2 dictionaries with keys *'images'* and *'labels'*, store the fashion_mnist datasets into *train_firstname* and *test_firstname*, where firstname is your firstname. *train_firstname* will contain the images and labels of the training data from 'fashion_mnist' and *test_firstname* will contain the images and labels of the testing data from 'fasion_mnists'. For more info checkout: https://keras.io/api/datasets/fashion_mnist/#load_data-function

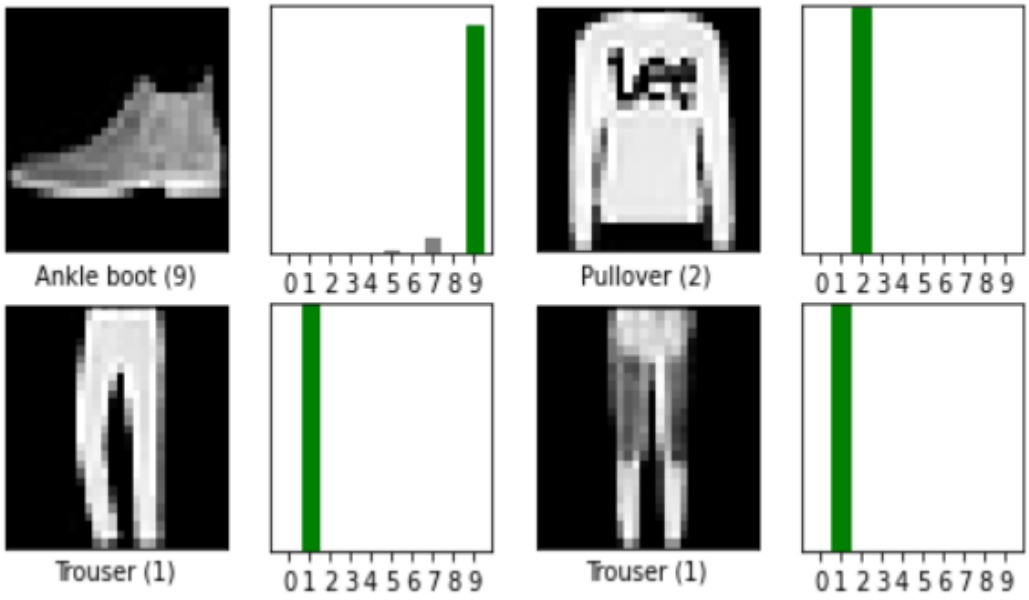
   b. Initial Exploration
      1. Display (print) the size of the training and testing dataset
      2. Display (print) the image resolution (dimension) of the input images.
      3. Display (print) the largest pixel value in the dataset using numpy.amax(). For more info checkout: https://numpy.org/doc/stable/reference/generated/numpy.amax.html

   c. Data Pre-preprocessing
      1. Normalize the pixel values in the dataset to a range between 0-1 using the info identified in Step b. Store result back into *train_firstname['images']* and *test_firstname['images']*
      2. Using tenflow's build in method to_cateogircal() to one-hot encode the labels. Store results back into *train_firstname['labels']* and *test_firstname['labels']*. For more info checkout: https://www.tensorflow.org/api_docs/python/tf/keras/utils/to_categorical
      3. Display (print) the shape of the *train_firstname['labels']* and *test_firstname['labels'].* Take note of the number of possible labels in the dataset

d. Visualization
    1. Create a function that displays (plots) an image with its true label using matplotlib. Remove xticks and yticks when plotting the image.
    2. Using the function created in Step d.1, plot the first 12 data samples in the training dataset using a figure size of 8x8 and a subplot dimension of 4x3
e. Training Data Preparation
    1. Using Sklearn's train_test_split() method split the training dataset in 80% training and 20% validation. Set the random seed to be the last two digits of your student ID number. Store the training data in a dataframe named: *x_train_firstname* for the feature (predictors) and the training labels *y_train_firstname*. Store the validation data as follows: *x_val_firstname* and *y_val_firstname*
f. Build, Train, and Validate CNN Model
    1. Use TensorFlow's Sequential() to build a CNN mode (name the model cnn_model_firstname) with the following architecture:
        i. Input = Set using info identified in Step b.
        ii. 1st Layer = Convolution with 32 filter kernels with window size 3x3 and a 'relu' activation function
        iii. 2nd Layer = Max Pooling with window size 2x2
        iv. 3rd Layer = Convolution with 32 filter kernels with window size 3x3 and a 'relu' activation function
        v. 4th Layer = Max Pooling with window size 2x2
        vi. 5th Layer = Full connected layer with 100 neurons (Note: Input to fully connected layer should be flatten first)
        vii. Output = Set output size using info identified in Step c.3 and a softmax activation function
    2. Compile the model with 'adam' optimizer, 'cateogrical_crossentropy' loss function, 'accuracy' metric
    3. Display (print) a summary of the model using summary(). Draw a diagram illustrating the structure of the neural network model, making note of the size of each layer (# of neurons) and number of weights in each layer.
    4. Using TensorFlow's fit() to train and validate the cnn model with 8 epochs and batch size of 256. Store training/validation results in *cnn_history_firstname.*
g. Test and analyze the model
    1. Display (plot) the Training Vs Validation Accuracy of the CNN Model as a line graph using matplotlib. Provide proper axis labels, title and a legend. Use different line color's for training and validation accuracy. Compare and analyze the training and validation accuracy in your report.

2. Evaluate the cnn model with the test dataset using Tensorflow's evaluate() and display (Print) the test accuracy. Compare and discuss the test accuracy to the validation accuracy in your report
3. Create predictions on the test dataset using TensorFlow's predict(). Name in the predictions *cnn_predictions_firstname*.
4. Create a function that plots the probability distribution of the predictions as a histogram using matplotlib. The function takes in the true label of the image and an array with the probability distribution. Probability of true labels are colored in green and predicted labels are colored in blue. Calling the function should produce a plot similar to below:

5. Using the created function in Step d.1 and g.4. display (plot) the first 4 images from the test dataset starting from the last 2 digits of your student number (i.e. if last 2 digits is 23, then display images 24-27) with their prediction probability distribution. For example:



6. Analyze and discuss the prediction probability distribution in your report

7. Display (plot) the confusion matrix of the test prediction using matplotlib, seaborn, and sklearn's confusion matrix. For more info checkout the following: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html and https://seaborn.pydata.org/generated/seaborn.heatmap.html

8. Analyze and discuss the confusion matrix in your report

h. Build,Train,Validate,Test and Analyze RNN Model
    1. Repeat Steps f and g for an RNN model with the following architecture
        i. Input = Set using info identified in Step b (Note: you can consider image height as the timestep in the RNN).
        ii. 1st Layer = LSTM with hidden state size 128 units
        iii. Output = Set output size using info identified in Step c.3 and a softmax activation function

------ End of Exercises ------

Rubric

Evaluation criteria	**Not acceptable**	**Below Average**	**Average**	**Competent**	**Excellent**
	0% - 24%	**25%-49%**	**50-69%**	**70%-83%**	**84%-100%**
Functionality	Missing all functionalities required	Some requirements are implemented.	Majority of requirements are implemented but some are malfunctioning.	Majority of requirements implemented.	All requirements are implemented Correctly.
Classes	Classes have been created incorrectly or completely missing.	Classes have been defined but have errors. Instances are incorrectly used.	Classes have been defined correctly but instances are used incorrectly or not created at all.	Classes have been defined correctly but some instances are used incorrectly.	Classes are correctly defined and makes use of its own functions which are called somewhere else in the code. Instances have been created and used correctly.
Documentation	No comments explaining code changes.	Minor comments are implemented.	Some code changes are correctly commented.	Majority of code changes are correctly commented.	All code changes are correctly commented.
Design	No adherence to object design principles.	Minor adherence to object design principles.	Some object oriented and modulus design principles are adhered to.	Majority of Object oriented and modulus design principles are adhered to.	Object oriented and modulus design principles are adhered to.
Testing & Evaluation	No evidence of testing and evaluation of the requirements.	Minor evaluation and testing efforts.	Some of the requirements have been tested & evaluated.	Majority of requirements are tested & evaluated.	Realistic evaluation and testing, comparing the solution to the requirements.

Demonstration Video	Very weak no mention of the code changes. Execution of code not demonstrated.	Some parts of the code changes presented. Execution of code partially demonstrated.	All code changes presented but without explanation why. Code demonstrated.	All code changes presented with explanation, exceeding time limit. Code demonstrated.	A comprehensive view of all code changes presented with explanation, within time limit. Code demonstrated.

Demonstration Video Recording

Please record a short video (max 4-5 minutes) to explain/demonstrate your assignment solution. You may use the Windows 10 Game bar to do the recording:

1. Press the Windows key + G at the same time to open the Game Bar dialog.

2. Check the "Yes, this is a game" checkbox to load the Game Bar.

3. Click on the Start Recording button (or Win + Alt + R) to begin capturing the video.

4. Stop the recording by clicking on the red recording bar that will be on the top right of the program window.

(If it disappears on you, press Win + G again to bring the Game Bar back.)

You'll find your recorded video (MP4 file), under the Videos folder in a subfolder called Captures.

Submit the video together with your solution and written response.