Lab Assignment 2: Autoencoders and Transfer Learning

Due Date: Sunday February 20, 2022

Purpose:

The purpose of this Lab assignment is to:

1. To get hands-on experience of applying Deep Neural Networks, namely Autoencoders and Transfer Learning

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 lose 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 lose 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_lab2.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 lab2assignment_firstname where firstname is your firstname.

Assignment – exercises:

1. **Exercise #1:** Autoencoders and Transfer Learning (100 marks)

Requirements:

a. Get the data:
    1. Import and load the 'fashion_mnist' dataset from TensorFlow. Using 2 dictionaries store the fashion_mnist datasets into *unsupervised_firstname* and *supervised_firstname*, where firstname is your firstname. The first 60,000 data samples will be stored in *unsupervised_firstname* directory with one key '*images*', which will contain the images for unsupervised learning. The next 10,000 data samples will be stored in *supervised_firstname* directory with keys '*images*' and '*labels*', which will contain the images and labels for supervised learning
    For more info checkout:
    https://keras.io/api/datasets/fashion_mnist/#load_data-function
b. Data Pre-preprocessing
    1. Normalize the pixal values in the dataset to a range between 0-1. Store result back into *unsupervised_firstname['images']* and *supervised_firstname['images']*
    2. Using tenflow's build in method to_cateogircal() to one-hot encode the labels. Store results back into *supervised_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 *unsupervised_firstname['images'], supervised_firstname['images']* and *supervised_firstname['labels'].*
c. Data Preparation (Training, Validation, Testing)

1. Using Sklearn's train_test_split() method split the unsupervised dataset into training (57,000 samples) and validation (3,000 samples). <u>Set the random seed to be the last two digits of your student ID number.</u> Store the training and validation data in a dataframe named: *unsupervised_train_firstname* and *unsupervised_val_firstname* for the feature (predictors) of the training and validation data respectively.
2. Using Sklearn's train_test_split() method randomly discard 7,000 samples from the supervised dataset. <u>Set the random seed to be the last two digits of your student ID number.</u>
3. Using Sklearn's train_test_split() method split the remaining supervised dataset (3,000 samples) into training (1800), validation(600) and testing(600). <u>Set the random seed to be the last two digits of your student ID number.</u> Store the datasets in a dataframe named: *x_train_firstname, x_val_firstname, and x_test_firstname* for the feature (predictors) and the training labels *y_train_firstname, y_val_firstname, and y_test_firstname*.
4. Display (print) the shape of the *unsupervised_train_firstname, unsupervised_val_firstname, x_train_firstname, x_val_firstname, x_test_firstname, y_train_firstname, y_val_firstname, and y_test_firstname.*

d. Build, Train, and Validate a baseline CNN Model

1. Use TensorFlow's Sequential() to build a CNN mode (name the model cnn_v1_model_firstname) with the following architecture:
   i. Input = Set based on image size of the fashion MNIST dataset.
   ii. 1st Layer = Convolution with 16 filter kernels with window size 3x3, a 'relu' activation function, 'same' padding, and a stride of 2.
   iii. 3rd Layer = Convolution with 8 filter kernels with window size 3x3, , a 'relu' activation function, 'same' padding, and a stride of 2.
   iv. 4th Layer = Full connected layer with 100 neurons (Note: Input to fully connected layer should be flatten first)
   v. Output = Set output size using info identified in Step b.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() and the training/validation supervised dataset to train and validate the cnn model with 10 epochs and batch size of 256. Store training/validation results in *cnn_v1_history_firstname.*

e. Test and analyze the baseline model

1. Display (plot) the Training Vs Validation Accuracy of the baseline 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. 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
5. Analyze and discuss the confusion matrix in your report

f. Add random noise to unsupervised dataset
    1. Using tf.random.normal(), add random noise to the training and validation unsupervised dataset with a noise factor of 0.2. Set the random seed to be the last two digits of your student ID number. Store results into *x_train_noisy_firstname* and *x_val_noisy_firstname*. For more info, reference: https://www.tensorflow.org/api_docs/python/tf/random/normal
    2. Using tf.clip_by_value(), clip the values of the noisy dataset to a range between 0 and 1. Store results back into *x_train_noisy_firstname* and *x_val_noisy_firstname*. For more info, reference: https://www.tensorflow.org/api_docs/python/tf/clip_by_value
    3. Display (plot) the first 10 images from the *x_val_noisy_firstname* using matplotlib. Remove xticks and yticks when plotting the image.

g. Build and pretrain Autoencoder
    1. Use TensorFlow's Model() [For more info, reference: https://www.tensorflow.org/api_docs/python/tf/keras/Model] to build an autoencoder mode (name the autoencoder_firstname) with the following architecture:
        i. Input = Set based on image size of the fashion MNIST dataset. Store layer as *inputs_firstname.*
        ii. Encoder Section (Store layers as *e_firstname*)
            1. Convolution with 16 filter kernels with window size 3x3, a 'relu' activation function, 'same' padding, and a stride of 2.
            2. Convolution with 8 filter kernels with window size 3x3, a 'relu' activation function, 'same' padding, and a stride of 2.

        iii. Decoder Section (Store layers as *d_firstname*)
            1. Transposed Convolution with 8 filter kernels with window size 3x3, a 'relu' activation function, 'same' padding, and a stride of 2. For more info reference: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2DTranspose
            2. Transposed Convolution with 16 filter kernels with window size 3x3, a 'relu' activation function, 'same' padding, and a stride of 2.
            3. Convolution with 1 filter kernels with window size 3x3, a 'sigmoid' activation function, and 'same' padding.

2. Compile the model with 'adam' optimizer, and 'mean squared error' loss function.
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() and the training/validation unsupervised dataset to train and validate the cnn model with 10 epochs, batch size of 256 and shuffle set to True. Use the noisy images from step f as input and original images from step c.1 as output.
5. Create predictions on the unsupervised_val_firstname dataset using TensorFlow's predict(). Name in the predictions *autoencoder_predictions_firstname*.
6. Display (plot) the first 10 predicted images from step 5 the using matplotlib. Remove xticks and yticks when plotting the image. Note: You can use Numpy's mean() to remove the remove the 3rd axis to properly plot the predicted images. For more info reference: https://numpy.org/doc/stable/reference/generated/numpy.mean.html

h. Build and perform transfer learning on a CNN with the Autoencoder
    1. Use TensorFlow's Model() [For more info, reference: https://www.tensorflow.org/api_docs/python/tf/keras/Model] to build a cnn mode (name the cnn_v2_firstname) with the following architecture:
        i. Input = Transferred from Autoencoder. See step g.1.i
        ii. 1st layer = Transferred from encoder section of Autoencoder (step g.1.ii
        iii. 2nd layer = Full connected layer with 100 neurons (Note: Input to fully connected layer should be flatten first)
        iv. Output = Set output size using info identified in Step b.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() and the training/validation supervised dataset to train and validate the cnn model with 10 epochs and batch size of 256. Store training/validation results in *cnn_v2_history_firstname.*

i. Test and analyze the pretrained CNN model
    1. Display (plot) the Training Vs Validation Accuracy of the pretrained 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. 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
    5. Analyze and discuss the confusion matrix in your report

j. Compare the performance of the baseline CNN model to the pretrained model in your report
    1. Display (plot) the Validation Accuracy of the Baseline vs the Pretrained model as a line graph using matplotlib. Provide proper axis labels, title and a legend. Use different line color's for the baseline and pretrained accuracy. Compare and analyze the validation accuracy in your report.
    2. Compare and analyze the test accuracy in your report.

------ 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.