ASSIGNMENT #3 – NEURAL NETWORKS

Context

This assignment is an opportunity to demonstrate your knowledge and practice solving problems about convolutional and recurrent neural networks. This assignment contains an implementation component (i.e. you will be asked to write code to solve one or more problems) and an experiment component (i.e. you will be asked to experiment with your code and report results).

Logistics

Assignment due date: March 13, 2023

Assignments are to be submitted electronically through Brightspace. It is your responsibility to ensure that your assignment is submitted properly and that all the files for the assignment are included. Copying of assignments is NOT allowed. High-level discussion of assignment work with others is acceptable, but each individual or small group is expected to do the work themselves.

Programming language: Python 3

For all parts of the implementation, you may use the Python Standard Library (https://docs.python.org/3/library/) and the following packages (and any packages they depend on): Keras, NumPy, Pandas, scikit-learn, SciPy, TensorFlow. Unless explicitly indicated below or explicitly approved by the instructor, you may not use any additional packages.

You must implement your code yourself; do not copy-and-paste code from other sources. Please ensure your implementation follows the specifications provided; it may be tested on several different test cases for correctness. Please make sure your code is readable; it may also be manually assessed for correctness. You do not need to prove correctness of your implementation.

You must submit your implementation as a single file named “assignment3.py” with classes functions as described below (you may have other variables/functions/classes in your file). Attached is skeleton code indicating the format your implementation should take.

You must submit your report on experimental results as a single file named “assignment3.pdf”.

Implementation Component

For the implementation, we will use the Hill Valley dataset. The task for this dataset is to predict, given a sequence of real scalar values, whether the sequence contains a hill (a "bump" in the terrain) or a valley (a "dip" in the terrain).

This dataset is publicly available, and it comes from the UCI machine learning repository (where a full description of the dataset can be found): https://archive.ics.uci.edu/ml/datasets/Hill-Valley.

For this task, we will use two files in the dataset: Hill_Valley_with_noise_Training.data and Hill_Valley_with_noise_Testing.data. The former contains a set of instances to be used for training and validation. The latter contains a set of instances to be used for testing your model. Note that these .data files are effectively .csv files with a different extension.

Each row in the file represents one sequence. The first 100 columns represent the values of that sequence. The last column represents the label associated with the sequence (0 = valley, 1 = hill).

Question 1 [10 marks]

Create a Keras data generator class for this dataset. The data generator should be usable for training or testing a Keras model.

The data generator class must be called "HillValleyDataGenerator" and it should have three member functions: "__init__", "__len__", "__get_item__" (you may also use other helper functions).

The function "__init__" is a constructor that should take two input arguments (in addition to "self"). The first input argument is the full path to a CSV file containing data. The second input argument is the batch size.

The function "__len__" should take zero input arguments (in addition to "self"). It should return the total number of batches in one epoch.

The function "__get_item__" should take one input argument (in addition to "self"). The first input argument is a batch index. The function should return two values: (1) a batch of data and (2) the labels associated with the batch.

Question 2 [10 marks]

Write a function that creates, trains, and evaluates a convolutional neural network to predict whether a sequence has a valley or a hill. The network should perform one-dimensional convolution over the sequence. You may consider adding fully connected layers at the end of your model for prediction of the binary label.

The function must be called "hill_valley_cnn_model".

The function should take one input argument: (1) the full file path to the "Hill_Valley_with_noise_Training.data" dataset.

The function should return three values: (1) a Keras model object that is trained to predict whether a sequence has a valley or a hill, (2) performance of the model on the training set, and (3) the performance of the model on the validation set.

(Hint: use the data generator you wrote)

Question 3 [10 marks]

Write a function that creates, trains, and evaluates a recurrent neural network to predict whether a sequence has a valley or a hill. The network should use a many-to-one design pattern with recurrence over the sequence. You may consider adding fully connected layers at the end of your model for prediction the binary label.

The function must be called “hill_valley_rnn_model”.

The function should take one input argument: (1) the full file path to the “Hill_Valley_with_noise_Training.data” dataset.

The function should return two values: (1) a Keras model object that is trained to predict whether a sequence has a valley or a hill, (2) performance of the model on the training set, and (3) the performance of the model on the validation set.

(Hint: use the data generator you wrote)

Experiment Component

Question 4 [10 marks]

Conduct a series of experiments on the convolutional and recurrent neural networks you implemented for predicting whether a sequence has valleys or hills. For each of these experiments, plot the performance on the training set and validation set as a function of the parameters.

a. Experiment on the size of the kernels in your convolutional neural network.
b. Experiment on the size of the hidden states in your recurrent neural network.
c. Experiment on the number of epochs of training for both the convolutional and recurrent neural networks.

Report the parameters and performance of the convolutional and recurrent neural networks with the best performance in the above models.

Using the best performing convolutional and recurrent neural networks found, compute performance of these models on the held-out test set (i.e. Hill_Valley_with_noise_Testing.data).

(Hint: use the data generator you wrote)