ASSIGNMENT #4 – NEURAL NETWORKS

Context

This assignment is an opportunity to demonstrate your knowledge and practice solving problems about autoencoders and practical strategies for building 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 27, 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 “assignment4.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 “assignment4.pdf”.

Implementation Component

For the implementation, we will use the Sign Language MNIST dataset. This dataset consists of static images of hand gestures for 24 classes of letters in American signal language (J and Z are excluded as they require motion).

This dataset is publicly available from Kaggle. You may download it and find more details on the dataset here: https://www.kaggle.com/datasets/datamunge/sign-language-mnist.

For this task, we will use two files: sign_mnist_train.csv and sign_mnist_test.csv. The first file contains a set of instances to be used for training. The second file contains a set of instances to be used for testing your model.

Each row in the file represents one 28x28 greyscale image. The initial column represents the label associated with the images. The remaining 784 columns represent the pixel intensities of the image. For our purposes, we will ignore the label associated with each file.

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 should perform appropriate pre-processing on the images (e.g. scaling, normalization, etc.).

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

The function "__init__" is a constructor that should take four 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 a list of rows of the CSV file to use images from (for example, if this argument is [1, 3, 5], your data generator should only use the images in rows 1, 3, 5 of your CSV file). The third 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 "__getitem__" 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 same batch of data.

Question 2 [10 marks]

Write a function that creates, trains, and evaluates a convolutional neural network autoencoder to encoder an image of an American sign language gesture. The encoder component of the network should perform two-dimensional convolution over the input. The decoder component of the network should perform two-dimensional transposed convolution over the input.

The function must be called "sign_language_autoencoder".

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

The function should return three values: (1) a Keras model object that is an autoencoder for this dataset, (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; randomly split the dataset into a training set and a validation set)

Experiment Component

Question 3 [10 marks]

Conduct a series of experiments on the autoencoder you implemented for this. 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 encoding.
b. Experiment on the number of convolutional layers in the encoder and decoder.
c. Experiment on the number of epochs of training.

Report the parameters and performance of the model with the best performance.

Using the best performing model you found, compute performance (i.e. reconstruction loss) on the held-out test set (i.e. sign_mnist_test.csv).

(Hint: use the data generator you wrote)

Question 4 [10 marks]

Conduct another series of experiments on the autoencoder you implemented for this. 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 use of pre-processing within the data generator.
b. Experiment on the use of batch normalization following each layer.
c. Experiment on the use of L1 and L2 regularization of the parameters.
d. Experiment on the use of dropout.
e. Experiment on the value of the learning rate.

Report the parameters and performance of the model with the best performance.

Using the best performing model you found, compute performance (i.e. reconstruction loss) on the held-out test set (i.e. sign_mnist_test.csv).

(Hint: use the data generator you wrote)