COMP5313 Artificial Intelligence
Department of Computer Science
Lakehead University

Assignment 1: Multi-Class Connectionist AI

Instructions to run the code:

The two python files:

• SiddhiJariwala_RL.py

• SiddhiJariwala_RL.ipynb

➢ These two files can be run on Python IDE PyCharm platform or jupyter notebook.

The used dataset contains 28K medium quality animal images belonging to 10 categories: dog, cat, horse, spider, butterfly, chicken, sheep, cow, squirrel, elephant.

Initially, I have imported all the required libraries such as:

Tensorflow, Numpy, Cv2, Pandas, Seaborn, Sklearn, matplotlib.

Flow of the program:

- After loading the necessary libraries and tools, data preparation is carried out by retrieving the photos from the dataset's 10 folders and resizing them. Following that, I constructed testing and training sets and shuffled them. Data normalization is done to remove duplicate data and preserve data quality because the data may include it. Additionally, it was attempted to identify the class that had the greatest number of cases, which turned out to be dog and spider, ultimately leading to the model being biased. Therefore, a random selection approach has been developed in order to prevent it.

- A random selection for the class labels of the test data (test_X_set). The ‘total_sum’ is the sum of the elements in the list ‘total_classes’. A for-loop is used to compute the random class label for each sample in the test data using the np.random.choice function with the class probabilities computed as the relative frequencies of each class in ‘total_classes’. The resulting list of randomly selected class labels is stored in ‘random_choice’.

The first 100 elements of ‘random_choice’ is printed for inspection. The performance of this random classifier is then evaluated by computing the classification report using the ‘classification_report’ function from scikit-learn library with the actual class labels ‘Y_test’ and the randomly selected class labels ‘random_choice.’ The target names for the classes in the report are provided by ‘all_classes’.

- The code defines two functions, one for training a model (Training_Function) and another for testing the model's performance (Testing). In the Training_Function, the model is trained using the fit method, with the training data (train_X_set), training labels (Y_train), number of epochs (15), batch size (1000), validation split (0.1), and verbose (1). The training history is saved as a dataframe and displayed.

  In the Testing function, the model's performance is evaluated on the test data (test_X_set) using the predict method. The predictions are then transformed into class labels using np.argmax and the classification report is printed using the classification report function from scikit-learn library. The target names for the classes in the report are provided by ‘all_classes’.

- Using the Keras API, the code constructs a sequential model. The model has many Dense layers and Convolutional Neural Network (CNN) layers. The first layer, a Conv2D layer with 32 filters, a kernel size of 3, and an activation function of relu, specifies the input shape of the model. Multiple MaxPooling2D layers in the model down sample the input data's spatial dimensions. The Flatten layer is used to flatten the data into a single dimension, which is required as input to the fully connected layers (Dense layers). A Dense layer with the activation function SoftMax and 10 units, which equals the number of classes in the dataset, makes up the final layer. The Adam optimizer and sparse categorical cross entropy loss function are used to build the model. After compilation, the model's summary is shown.

The model is then trained and tested. The output of the classification report of the testing phase contains the precision, recall, and f1-score for each of the classes (dog, horse, elephant, butterfly, chicken, cat, cow, sheep, squirrel, and spider). It also shows the overall accuracy, macro average, and weighted average for all the classes.