Dobre Bogdan-Mihai
Group 512, NLP

Deep Learning Project Documentation

For this task I achieved the best results by using 2 models: CNN and a pretrained VGG on ImageNet (the VGG16 model). Below I will provide the details about the implementation of the 2 models. The accuracy for the 2 models were as follows: 0.63 for the CNN model and 0.43 for the VGG model.

Each model has 3 functionalities: training on the entire training data and predicting labels for the test data, doing a 5 fold cross validation using the training data and using a grid search to find an optimal learning rate. I created 5 subdirectories, one for each class, and added the corresponding images to their respective subdirectories. I did this to easily read them using the keras function image_dataset_from_directory.

The CNN model has 9 repetitions of the following layers group: Conv2D, Relu activation, and Batch Normalisation. After the 3rd, 6th and 9th groups there is both a MaxPooling layer and a Dropout layer. At the end there is a Dense layer with 512 nodes, another dropout layer, a Dense layer with 5 nodes (the number of classes) and an activation layer with the Softmax layer. The first 3 Conv2D layers have 62 filters while the following 6 have 128 filters. I used the image_dataset_from_directory function from Keras to load the training data in a Keras dataset. I trained the model on this dataset using the adam optimizer, the sparse categorical cross entropy loss and accuracy as a metric. Then I use it to predict the labels of the test data and write the results to the output.csv file.

The 3 functions for this model are described below:

1) The function used to train on the training data and predict labels for the test data is called train_and_predict. It trains the model on the training data, reads the test data using csv and keras and writes the id and labels to an output file called "output.csv" using csv. There are 25 epochs used for training for this function.
2) The function used to do a 5 fold cross validation is called n_fold_cross_validation. In order to achieve the 5 runs necessary I used the KFold class from sklearn. The model is trained at each step on the current training data and evaluated on each run against the current validation data. The accuracies are collected in an array and written to an output file called "results.csv" using csv. The accuracy was around 0.62 for each run. There are 25 epochs used for training for this function.
3) The function used to do a grid search for an optimal learning rate is called grid_search_learning_rate. Due to how much time training takes, for this function the model is trained on just 7 epochs instead of 25 for this function in order to find some results in an acceptable time. The function iterates through some possible learning rates, uses KFold to split the training data between training and validation data, trains and evaluates the model and outputs the performance for each learning rate. The best learning rate from the ones tested seems to be 0.001. A table detailing the results can be found at the end.

The VGG model uses the VGG16 model imported from tensorflow.keras.applications.vgg16. It uses the expected input shape of the images (128, 55, 3), the imagenet weights and it doesn't include the top layer. A new model is created from it by adding the following layers for its output: a Resciling layer, a Flatten layer, a Dense layer with 1024 nodes, a Dropout layer with 0.5 rate and a Dense layer with 5 nodes (the number of classes). I used the

image_dataset_from_directory function from Keras to load the training data in a Keras dataset. I train the model on this dataset using the adam optimizer, the sparse categorical cross entropy loss and accuracy as a metric. Then I use it to predict the labels of the test data and write the results to the output.csv file.

The 3 functions for this model are described below:

1) The function used to train on the training data and predict labels for the test data is called train_and_predict. It trains the model on the training data, reads the test data using csv and keras and writes the id and labels to an output file called "output.csv" using csv. There are 25 epochs used for training for this function.
2) The function used to do a 5 fold cross validation is called n_fold_cross_validation. In order to achieve the 5 runs necessary I used the KFold class from sklearn. The model is trained at each step on the current training data and evaluated on each run against the current validation data. The accuracies are collected in an array and written to an output file called "results.csv" using csv. The accuracy was around 0.62 for each run. There are 25 epochs used for training for this function.
3) The function used to do a grid search for an optimal learning rate is called grid_search_learning_rate. Due to how much time training takes, for this function the model is trained on just 7 epochs instead of 25 for this function in order to find some results in an acceptable time. The function iterates through some possible learning rates, uses KFold to split the training data between training and validation data, trains and evaluates the model and outputs the performance for each learning rate. The best learning rate from the ones tested seems to be 0.001. A table detailing the results can be found at the end.

Both models have a function called get_model that has an optional parameter for the learning rate. In each model, the specific get_function method is called by each of the other 3 functions when needed. Both models also have a generic batch size of 32, except for when the value of 15500 is used to get the whole data in a single batch in order to easily convert the data to numpy. This happens in the n_fold_cross_validation and the grid_search_learning_rate for both models.

Regarding tables and figures, for the 5 mean cross validation for both models the results were almost equal, as stated above, so a table for that would have been redundant. Below there is a table detailing the performance of each model for some learning rates. It is important to mention that these results are for 7 epochs instead of 25, as a grid search for that many epochs proved unfeasible. Another aspect that is important to mention is that due to how much time training takes for each model, grid searching more than one parameter at a time also proved unfeasible.

Accuracy for different combinations of learning rate and model:

	Model	VGG	CNN
Learning rate			
0.001		0.43	0.58
0.01		0.42	0.55
0.1		0.2	0.22