COMP- 8220 Machine Learning Section B

Suhash Reddy Immareddy

45693242

Question 11:

45693242 => A = 2, B = 4, C = 2

Therefore A = 1 conv layer of kernel size 3, a max pool layer with pool size 2, followed by a Dense layer.

B = ELU

C = Include a batch normalization after other appropriate Hidden layers.

For importing the data, we have already given the code so not including here, Input shape is 28 * 28 since Given is MNIST data

i) Defining the model: Since Filter size is not mentioned I used 32 for the 1st conv layer

```
Model = keras.models.Sequential([
Keras.layers.Conv2d(32, 3, activation = "elu", input_shape = [28, 28]), # Input Layer
Keras.layers.BatchNormalization(), # Normalization layer
Keras.layers.MaxPooling2D(2), # Max Pooling layer
Keras.layers.Conv2d(32, 3, activation = "elu"), # 1st Hidden Layer
Keras.layers.BatchNormalization(), # Normalization layer
Keras.layers.MaxPooling2D(2), # Max Pooling layer
Keras.layers.Flatten(), # Flattening
Keras.layers.Dense(250, activation = "elu"), # Fully connected layer with 250 neurons
Keras .layers.Dense(10, activation = "softmax") # Output Layer
])
```

ii) Compiling the Model: Choosing Adam Optimizer and accuracy metrics.

```
opt = keras.optimizers.Adam
Model.compile(loss= "sparse_categorical_crossentropy", optimizer=opt, metrics = ['accurracy'])
```

iii) Training the model: Using 300 as Batch Size, 20 Epochs for training the model

```
History = Model.fit(X_train, y_train, batch_size=300, epochs=20, Validation_data = (X_valid, y_valid))
```

Question 12:

45693242 => A = 2, B = 4, C = 2

Therefore A = 1 conv layer of kernel size 3, a max pool layer with pool size 2, followed by a Dense layer.

B = ELU

C = Include a batch normalization after other appropriate Hidden layers.

For importing the data, we have already given the code so not including here,

i) Defining the model: Since Filter size is not mentioned I used 32 for the 1st conv layer

Model = Sequential()

Embedding_layer = Embedding(encoder.vocab_size, 100, weights = [], input_length = [] , trainable = False),

Model.add(Embedding_Layer)

Model.add(Conv1D(32, 3, activation = "elu"), **# Hidden Layer**

Model.add(BatchNormalization()), **# Normalization layer**

Model.add(MaxPooling1D(2)), **# Max Pooling layer**

Model.add(GlobalMaxPooling1D())**# Global Max Pooling layer**

Model.add(Dense(250, activation = "elu")), **# Fully connected layer**

Model.add(Dense(1, activation = "sigmoid")) **# Output Layer**

ii) Compiling the Model: Choosing Adam Optimizer and accuracy metrics.

opt = keras.optimizers.Adam
Model.compile(loss= "binary_crossentropy", optimizer=opt, metrics = ['accurracy'])

iii) Training the model: 20 Epochs for training the model

History = Model.fit(train_batches, epochs=20, Validation_data = test_batches, verbose = 1)

Sources:

https://www.tensorflow.org/tutorials/keras/text_classification_with_hub

https://stackabuse.com/python-for-nlp-movie-sentiment-analysis-using-deep-learning-in-keras/

https://realpython.com/python-keras-text-classification/

Question 13:

Here we need to identify the type of the tile by just taking the roof photo, which can be done using Transfer learning.

Selecting a Suitable Existing Architecture: Here while selecting the type of transfer learning model we need to check whether a model is trained on the similar datasets or not. If trained on similar data, then we can make use of the information from the lower layers and use them to classify our output.

For example in this case we can use existing model trained on the Roof Data (which is similar to our case) which is a CNN trained on RCB Data (or) we can use Resnet50v2 as it is trained on the ImageNet which contains 1000 classes which involves roof, house, tile images so, we can use that particular features for our task.

How much of chosen architecture is used: Assuming that I selected Resnet50v2, At first we can use al the existing layers from the Resnet50v2 Except the output layer (not needed since our task is different which is classification of tiles). Example 1, 2, …50 (50 is output) layers we can use till layer 49 as all these layers will contribute to some information when using in our case.

How to Build Transfer learning with new layers: we can build our own layers after the transfer learning models. For example, if we choose to use first 49 input layers from resnet50v2 we can later add as many dense layers (fully connected layers) and then a output layer with 57 as no of output classes and a SoftMax activation.

How the new model is Trained: Firstly we can Try Freezing all the layers that we selected to use by using "traininable.layers = "False" such that the weights will be fixed. Then we can train the layers that we added to the transfer learning if the validation accuracy is increasing and loss is decreasing (or) if not than we can make some layers in the transfer learning as trainable.

Secondly, we can try Freezing the layers till layer 30 since we can use the information form the lower layers, and train the layers after 30 till 49 as trainable along with our newly added layers, then we can inspect the model performance on validation datasets. Still it did not improve the model we continue training some more extra layers and inspect the result.

Question 14:

Here we need to identify the type of the mode for a new social media account.

Selecting a Suitable Existing Architecture: Here while selecting the type of transfer learning model we need to check whether a particular pre trained embedding would suite with our data or not. If trained on similar data, then we can make use of the information from their pre trained embeddings.

For example, we can use Transfer learning by making use of NNLM-128 and USE-512 sentence Embeddings.

How much of chosen architecture is used: These Pre trained embeddings can be used in two ways. One, where we freeze the embeddings and the second where we retrain the Embeddings with some fine Tuning.

How to Build Transfer learning with Pretrained Embeddings: we can build our own layers after the transfer learning models. For example, if we choose to use NNLM-128 embeddings we can later add as many dense layers (fully connected layers), Convolution layers (CONV1D) and then an output layer with 4 as no of output classes and a SoftMax activation.

How the new model is Trained: Firstly, we can Try using the pre trained embeddings by freezing them and the we can add our own layers and train the whole model. If the validation accuracy is increasing and loss is decreasing (or) if not than we can look at the following case.

Secondly, we can try training the pre trained embeddings with unfreezing them such that we now can fine tune them with our own layers (CONV1D, Dense, Output). Then we can see some improvement upon comparing to the above one.

Source: https://towardsdatascience.com/deep-transfer-learning-for-natural-language-processing-text-classification-with-universal-1a2c69e5baa9