

 Return to "Artificial Intelligence Nanodegree and Specializations" in the classroom

DISCUSS ON STUDENT HUB

DNN Speech Recognizer

REVIEW
CODE REVIEW 5
HISTORY

Meets Specifications

Congratulations for passing this project! Good luck with future submissions:)

To really build a good speech recognition model using DNN's, I would urge you to see the following links:

https://research.google.com/pubs/archive/38131.pdf https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/CNN_ASLPTrans2-14.pdf

STEP 2: Model 0: RNN

The submission trained the model for at least 20 epochs, and none of the loss values in model_0.pickle are undefined. The trained weights for the model specified in simple_rnn_model are stored in model_0.h5.

This simple model doesn't fit the data very well and thus the loss is very high.

STEP 2: Model 1: RNN + TimeDistributed Dense

The submission includes a sample_models.py file with a completed rnn_model module containing the

correct architecture.

Adding batch normalization and a time distributed layer improves the loss by a lot! Try different units here, SimpleRNN, LSTM, and GRU to see how their performance differs.

The submission trained the model for at least 20 epochs, and none of the loss values in model_1.pickle are undefined. The trained weights for the model specified in rnn_model are stored in model_1.h5.

STEP 2: Model 2: CNN + RNN + TimeDistributed Dense

The submission includes a sample_models.py file with a completed cnn_rnn_model module containing the correct architecture.

These models are very powerful but have a tendency to severely overfit the data. You can add dropout to combat this.

The submission trained the model for at least 20 epochs, and none of the loss values in model_2.pickle are undefined. The trained weights for the model specified in cnn_rnn_model are stored in model_2.h5.

STEP 2: Model 3: Deeper RNN + TimeDistributed Dense

The submission includes a sample_models.py file with a completed deep_rnn_model module containing the correct architecture.

Adding additional layers allows your network to capture more complex sequence representations, but also makes it more prone to over tting. You can add dropout to combat this.

The submission trained the model for at least 20 epochs, and none of the loss values in model_3.pickle are undefined. The trained weights for the model specified in deep_rnn_model are stored in model_3.h5.

STEP 2: Model 4: Bidirectional RNN + TimeDistributed Dense

The submission includes a sample_models.py file with a completed bidirectional_rnn_model module

containing the correct architecture.

These models tend to converge quickly. They take advantage of future information through the forward and backward processing of data.

The submission trained the model for at least 20 epochs, and none of the loss values in model_4.pickle are undefined. The trained weights for the model specified in bidirectional_rnn_model are stored in model_4.h5.

STEP 2: Compare the Models

The submission includes a detailed analysis of why different models might perform better than others.

This is a pretty good analysis of each individual model and explains why different models perform better than others.

Additionally, you can also compare these in a table by looking at things such as overfitting, the number of parameters that need to be tuned, and the training time for each individual model.

For Speech Recognition models, Batch Normalization and the TimeDistributedDense layer are very useful in general.

STEP 2: Final Model

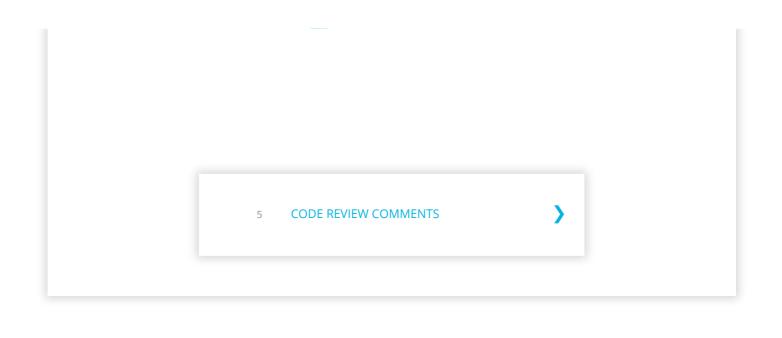
The submission trained the model for at least 20 epochs, and none of the loss values in model_end.pickle are undefined. The trained weights for the model specified in final_model are stored in model_end.h5.

You can use the stopping callback function in keras where you don't have to manually tune the number of epochs:). You can also consider adding more dropout and recurrent dropouts.

The submission includes a sample_models.py file with a completed final_model module containing a final architecture that is not identical to any of the previous architectures.

The submission includes a detailed description of how the final model architecture was designed.

Nice job. Your reasoning is sound here. Did the model perform as well as you thought it would? Why or why not?



RETURN TO PATH

Rate this review