Assignment 5. COVID-19 detection from coughs

Jordi Puig Rabat & Marc Vernet Sancho

June 12th, 2021

The VGGish pre-trained model is a Speech Embedder, more precisely, it maps audio sequences to a 128 dimensional embedding space. This model may work very well for speech recognition but (as seen in the baseline) performed poorly with the given task of COVID-19 detection from coughs. We have centered our efforts in improving the performance of this model in this particular task, validating the proposed ideas and comparing the results obtained.

1 Fine tuning VGGish

The first approach was to classify the 128 dimensional vector outputted by the pre-trained model with a single hidden layer perceptron. As shown in the baseline Notebook, the AUC achieved was around 0.66; we took the model and applied different strategies to fine-tune it to perform better at the given task obtaining results above 0.75 AUC. First let's remember the VGGish architecture. This model (the trainable module) is composed of two main parts: the Feature Extractor (FE) –which is a CNN– and the Embedder –which is a fully connected MLP. The output of the FE has shape (512, 6, 4) which is then flattened to a 12288 dimensional vector, fed to the Embedder and the output of the Embedder is a 128 dimensional vector.

The central idea that we will present is to fine-tune the parameters of the Embedder. We took two main lines: fine-tune the module starting from the pre-trained parameters and training all the parameters from scratch. In the latter, we tested different strategies of parameter initialization and different hyperparameters for the MLP.

1.1 Initialization strategies

As stated, there were two different strategies to initialize the weights of the Embedder, from scratch and from pre-trained. When initializing from scratch we worked with the PyTorch default initialization and the Xavier Uniform Initialization [1]. With all the strategies, we needed a last layer to classify the output that must be initialized from scratch.

1.2 Tests with different architectures

When we tested the scratch initialization, we considered the idea of varying the number of hidden units per layer; thus, we tested different hyperparameters relating the size of each hidden layer. Although changes in the architecture were present, we didn't change the number of hidden layers of the model which is always 3 plus the output layer.

Moreover, dropout was added between the fully connected layers, this dropout was between 0.5 and 0.75. In the case of the scratch initialization, it was easy to add this dropout as we defined a new model; but in the case of the pre-trained initialization, these layers had to be added in the middle of the model.

We tested a bunch of different parameters as the nature of the task let us make a lot of fast tests. We tested 81 variations for the scratch initialization (as it let us make more changes) and 5 variations for the models initialized from pre-trained.

Aside from the tests with fully connected MLP, we tested CNN architectures in the firsts layers of the embedding module. To this end we needed to reshape and transpose the output. We took the first dimension as the sequence length (512) and the other two dimensions were joined to create the feature space dimension with size 24. We tried to use two and three CNN layers with Batch Normalization and Max Pooling and two hidden fully connected layers. The results were improved with respect the baseline but didn't outperform the other approaches.

In the following sections we will present only the architectures that gave us best validation results.

2 Validation Method

The problem proposed came with a training set and a validation set, so the first way to validate the models –which was given by the professor– was computing the AUC in the validation set. Later, as we wanted to obtain more reliable metrics, we implemented a Cross Validation routine that used all the available data to train and validate. More precisely, we used 10-fold Cross Validation. We considered this was a better option than doing the validation only with the training data because as more data is used, more reliable the results are. The training routines were the same as the given ones (with early stopping) and the mean along the folds was taken. The results obtained with this validation method are presented in the following section.

3 Results

In the following table we can see the AUC scores for a variety of models and initialization methods. We considered the validation score obtained with 10-fold Cross Validation and the public score from the Kaggle competition. For each architecture and initialization we show the model with best CV score and it s respective Public score. As discussed above, different hyperparameters were tested for each combination. The hyperparameters of each architecture can be found in the notebook attached to this document. We see that the best CV score corresponds to the Fully Connected model with Xavier Uniform initialization wich had three hidden layers of size 4096, 1024 and 32 respectively and was trained with a dropout factor of 0.75.

Architecture	Initialization	Validation Score	Public Score
CNN	Xavier uniform	0.74805	0.71467
CNN	Default	0.75184	0.69615
FC	Pre-trained	0.75632	0.67678
FC	Default	0.76400	0.71004
FC	Xavier uniform	0.77049	0.73251

Table 1: The best results for each pair of architecture and initialization strategy.

3.1 Submitting predictions to competition

After validating some models with all the available data and cross validation, we encountered a non trivial question: how can we submit the best model to obtain results close to the CV score. The first approach was to train a model 50 epochs with all the data available but this strategy didn't gave us good results in public score. The second strategy was to replicate the idea from the baseline notebook: choose the weights obtained in the epoch with best validation score; maintaining the original train-validate split. This strategy gave us better results although the public score varied, normally down but in some cases up. We also tried a last strategy that involved cross validation in the training and the selection of the trained model; we trained the model 10 times (each time starting from initialization), one for each fold of the data, and selected the weight configuration that gave us best validation results. For each fold we saved the state that obtained higher score and then from the ten best scores we selected again the best. Note that this strategy doesn't guarantee the best results in the test set, but it seemed a possible option to consider. The results in the test set didn't improve, this didn't surprise us because, as said, there is no guarantee that the model with best score in a particular fold generalizes better than other models.

We used the second approach (the one proposed originally in the notebooks) to use for final score of the competition.

4 Conclusion

The results obtained with all the tests sowed that re-training the embedder module from the VGGish model improved performance in this task. More precisely, the best results in cross validation showed us that using the same architecture as the original embedder (fully connected) with few changes and initializing the weights with Xavier Uniform [1] achieved the best results. It is not easy to interpret the meaning and the reason of this when it comes to the embedding space. It is obvious that the original embeddings didn't carry much information related to respiration and coughing, and that is why the fine-tuned model performed a lot better. But there is an interpretation question with

the last hidden layer of all our models. We consider it can't be interpreted as an embedding layer because it captures features related to COVID-19 and not respiration and coughing in a more general sense. Thus, when we come from the weights of the pre-trained embedder it is hard to find new values for the weights in order to classify. That could be a reason why the pre-trained initialization didn't achieve the best results.

References

[1] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html.