Austin Kothig & Lukas Grasse
001182645 & 001172543
CPSC 4310

Final Report

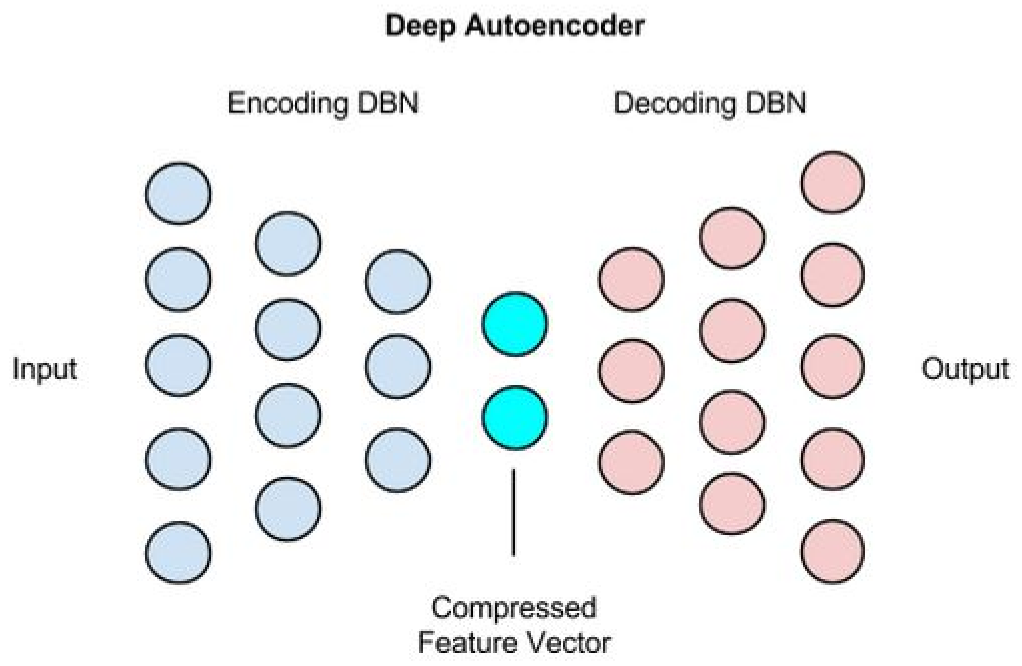
1st December 2017

Summary of the Component Being Implemented

The goal of our project has been to classify recordings of acoustic scenes into categories. The dataset that was used was from challenge 1 of the DCASE 2017 (Detection and Classification of Acoustic Scenes and Events)[2] conference. This dataset contains 10 second audio recordings of 15 different scenes.

The first experiment we ran used high-level embedding features generated from a pre-trained Convolutional Neural Network based on the VGG architecture[3]. These embeddings were represented as numpy arrays, that were fed into a network that runs batch normalization on the embeddings and passes them to a bidirectional layer of long short term memory nodes. These are then passed to a densely connected layer of size 1024, and then lastly connected to our output layer. This approach of building on top of a pre-trained network scales quite well on extremely large data sets.

The second experiment we ran was to train a similar network using embeddings generated from a wavenet autoencoder. An autoencoder is a network that is trained to produce its input as output, with the constraint that the network shrinks to a small feature space before producing the output. E.g.



https://deeplearning4j.org/deepautoencoder

In this case we generated the embeddings represented by the turquoise nodes using the pre-trained autoencoder, and then trained our network on top of these embeddings. The autoencoder was originally trained on musical instruments, so it was interesting to see how accurately features derived from different datasets could be applied to scene classification.

Implementation Details

VGG Embedding Classifier

The cnn_embedding_classifier folder contains the code for our VGG Embedding Classifier experiment. The generate_embeddings.py script was adapted from the VGGish scripts and is used to generate the sparse embeddings that our neural networks are trained on. The train_classifier.py, train_simple_classifier.py, and train_lstm_classifier.py scripts were used to train machine learning models using keras. The best performing model was the train_lstm_classifier.py script, which used a layer of bidirectional lstm nodes. This approach of building on top of a pre-trained network performs quite well on extremely large data sets. The results of this model are shown in the report.

Wavenet Embedding Classifier

The wavenet_classifier folder contains train_lstm_classifier.py, the script we used to train models on embeddings generated from the NSynth Wavenet Autoencoder. The autoencoder is described in the paper "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders". The nsynth package can be installed using pip, and the command to generate embeddings from audio is similar to:

'''nsynth_save_embeddings --checkpoint_path=//wavenet-ckpt/model.ckpt-200000 --source_path=/ --save_path=/ --batch_size=4'''

Once the embeddings are generated, the train_lstm_classifier.py script can be used.

CNN Spectrogram Classifier

The cnn_classifier folder contains the scripts used to train a model directly on the audio spectrograms. The implementation is based on the paper "Convolutional Neural Networks with Binaural Representations and Background Subtraction for Acoustic Scene Classification". Although we did not evaluate this model in the report, we have included the source code as a reference for a standard CNN model for this task.

Evaluation Strategy

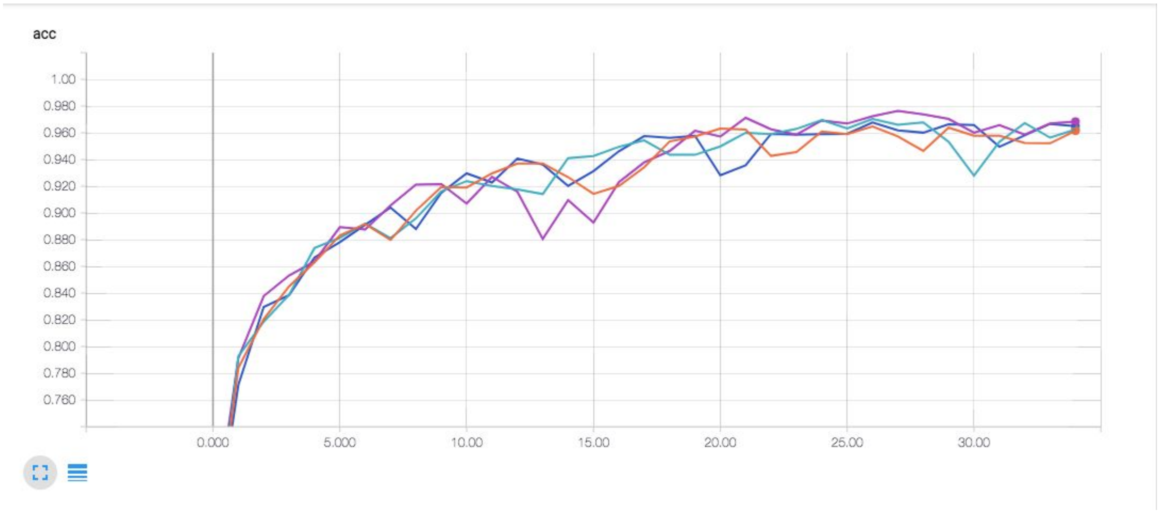
For evaluation we used the provided training/testing data from DCASE. The audio samples came in four separate folds, each with their own training and testing data with 15 classes. For evaluation we ran the first two networks against each of the four folds using the testing data for evaluation.

Results of Evaluation

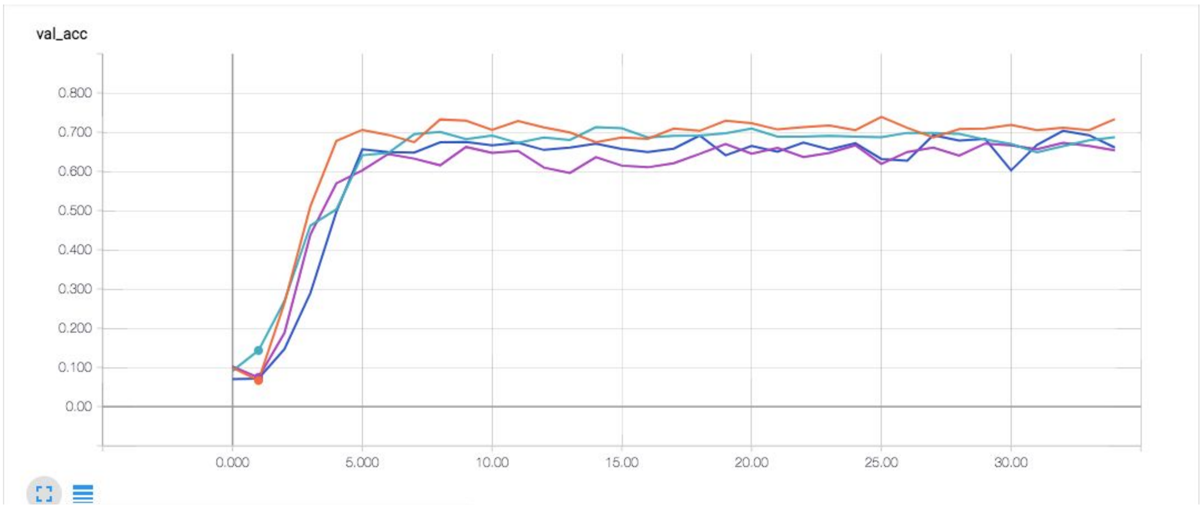
VGG Embedding Classifier

Here are the results of the model trained on top of the embeddings from the VGG pretrained model.

Training Accuracy



Validation Accuracy



The validation accuracy for each fold is:

fold 1: 0.7333

fold 2: 0.7127

fold 3: 0.6726

fold 4: 0.7034

This means the cross-validation accuracy is (0.7333+0.7127+0.6726+0.7034)/4 = 0.7055

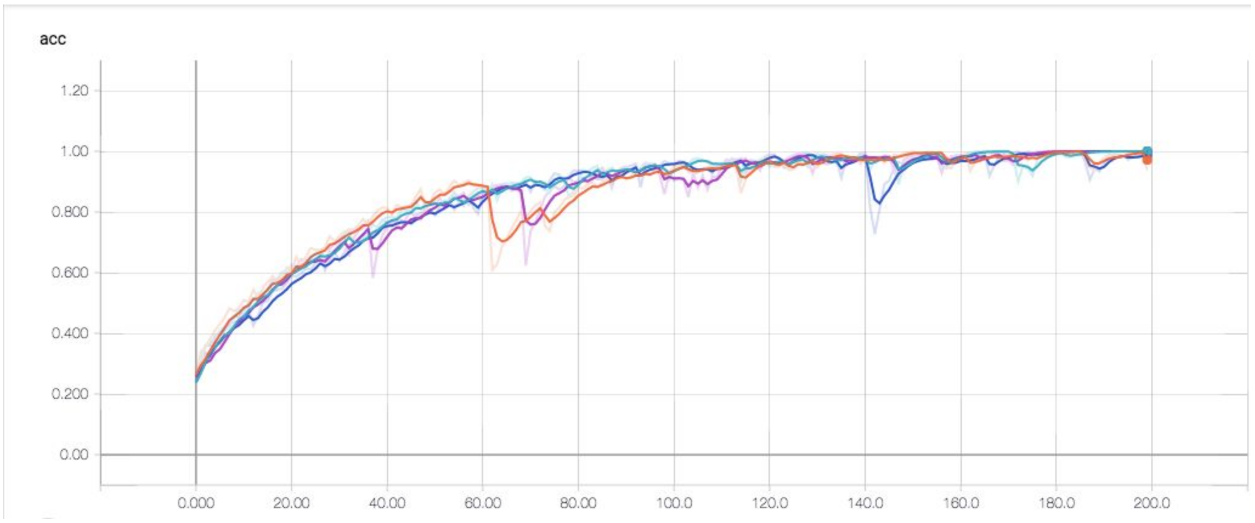
= 70.55%

This is close to the baseline system, and performed fairly well considering the vgg embeddings were trained on a different dataset.

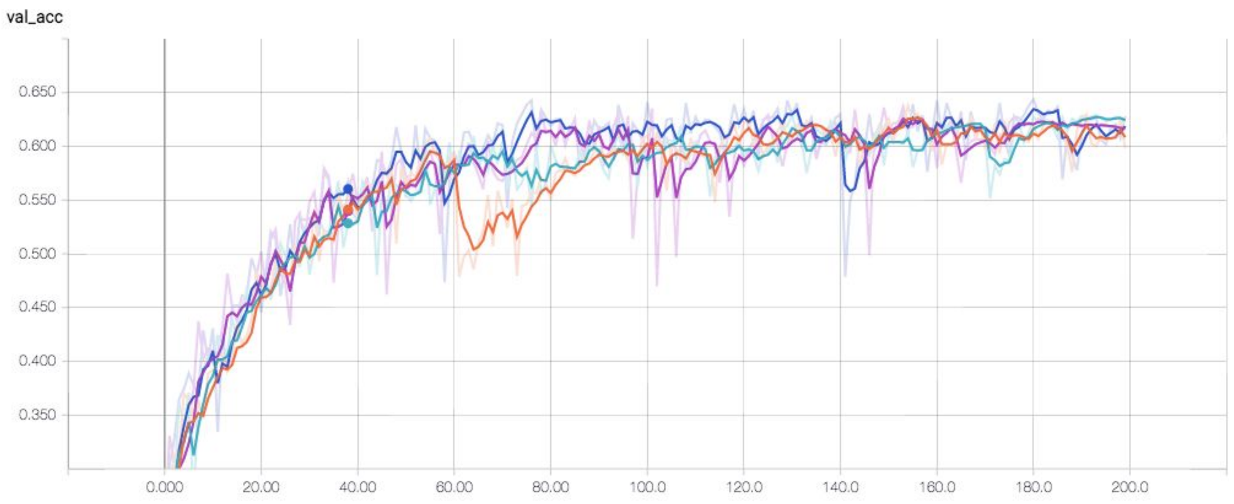
Wavenet Embedding Classifier

Here are the results of the model trained on top of the embeddings from the pretrained Wavenet Autoencoder.

Training Accuracy



Validation Accuracy



fold 1: 0.6376

fold 2: 0.6292

fold 3: 0.6368

fold 4: 0.6393

This means the cross-validation accuracy is (0.6376+0.6292+0.6368+0.6393)/4 = 0.0.6357

= 63.57%

The wavenet embedding features performed considerably worse than the VGGish embeddings. The fact that the Wavenet was trained on musical instruments, whereas the VGG network was trained on youtube videos might provide an explanation for the worse performance. This is because Youtube videos are probably more similar to acoustic scenes than musical instruments.

Conclusions

The conclusion we came to was that although there is a degree of performance lost when using transfer learning from a pretrained model, it is extremely fast to train a new classifier using this method, and allows easier scaling to larger data sets. Although the average classification of our networks came just shy of meeting the baseline of 74.8%, we still produce strong accuracies with the VGG network after only 35 episodes of training.

Sources

http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-acoustic-scene-classification

[1] Gemmeke, Jort F., et al. "Audio Set: An ontology and human-labeled dataset for audio events."

***IEEE ICASSP*. 2017.**

[2] Mesaros, Annamaria, et al. "DCASE 2017 challenge setup: Tasks, datasets and baseline system." DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events. 2017.

[3] Hershey, Shawn, et al. "CNN architectures for large-scale audio classification." *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017.