Jaden He

11-775 HW 1

Feb 14th, 2022

Pipeline

1. Extract the audios part in .wav and .mp3 formats from the dataset using ffmpeg.
2. Get the Mel-Frequency Cepstral Coefficients (MFCCs) features from the .wav audios using opensmile 3.0.
3. Select 40% of the MFCCs using select_frames.py, generating selected.mfcc.csv, for k-means clustering.
4. Set k = 25, 50, 200 respectively to train train_kmeans.py in order to get feature codebook from MFCCs features. I get kmeans.25.model, kmeans.50.model, kmeans.200.model in the models folder.
5. Extract bag-of-feature representations for each audio by get_bof.py. I store them in bof_25, bof_50, and bof_200 folders respectively.
6. Get the SoundNet features using extract_feats.py and sound8.npy from the .mp3 audios. Then extract the y_scns features using get_avg.pool.py.
7. Get the PANNs features using panns.py from the .mp3 audios.
8. Get the PaSST scene embeddings features using past.py from the .wav audios.
9. Train the classification models on SVM with RBF kernel and Sigmoid kernel, with inputs bof_25/, bof_50/, and bof_200/. Trained SVM with RBF kernel with inputs soundnet/avg_pooling/y_scns/, panns/, and passt/scene/.

10. Train the classification models on MLP with different hidden_layer_sizes (from 128 to 4096), with inputs bof_25/, bof_50/, bof_200/, soundnet/avg_pooling/y_scns/, panns/, and passt/scene/.
11. Evaluate the performance of the models by 5-fold cross validation, taking 80% of the set to train and 20% to validate, which is done 5 times with different folds and calculate average training accuracy, average validation accuracy, and best validation accuracy in the end. Top-1 accuracy metric was used since it is used on the Kaggle competition.

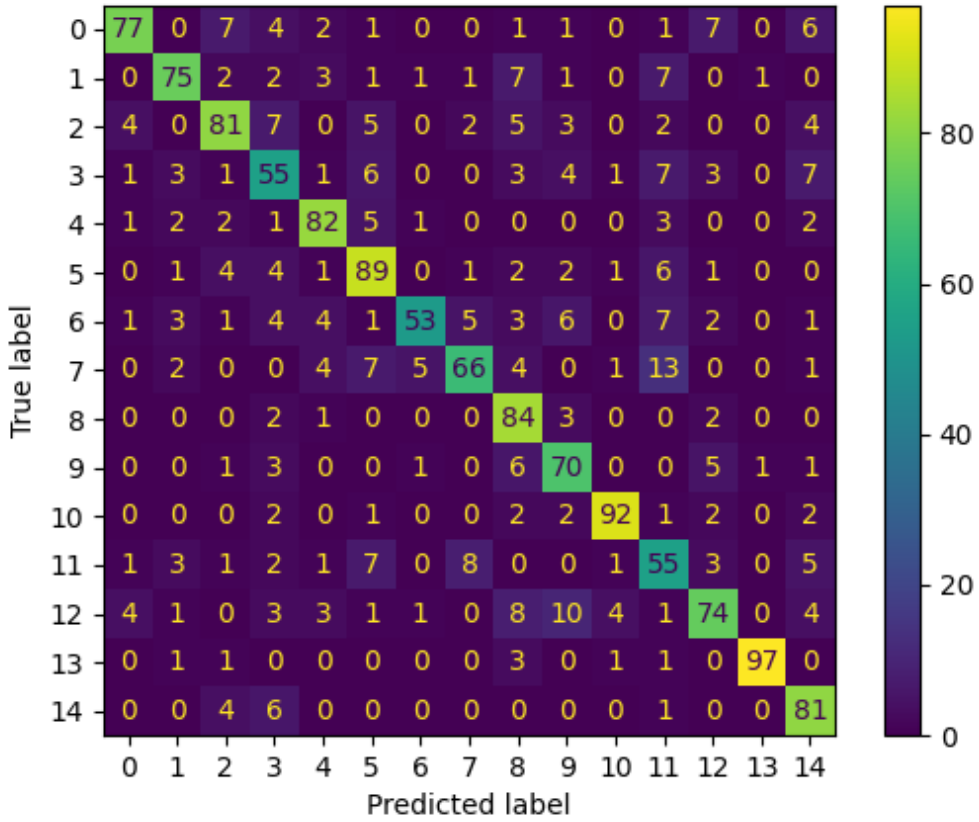
Table of results

Model	Input	Average Training Accuracy	Average Validation Accuracy	Best Validation Accuracy	Test Accuracy
SVM-RBF kernel	MFCC-25	0.33767	0.29195	0.31041	
SVM-RBF kernel	MFCC-50	0.39601	0.32292	0.34423	
SVM-RBF kernel	MFCC-200	0.50541	0.35750	0.37717	
SVM-Sigmoid kernel	MFCC-25	0.19750	0.19984	0.21696	
SVM-Sigmoid kernel	MFCC-50	0.23154	0.22774	0.24299	
SVM-Sigmoid kernel	MFCC-200	0.32386	0.29929	0.31375	
SVM-RBF kernel	SoundNet-y_scns-1024	0.11268	0.09817	0.10733	
SVM-RBF kernel	PANNs-2048	0.78601	0.71455	0.72849	0.70535
SVM-RBF kernel	PaSST-scene-1295	0.76267	0.74083	0.76451	
MLP-hidden_layer_sizes=(512,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	MFCC-25	0.63476	0.31237	0.31976	

MLP-hidden_layer_sizes=(512,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	MFCC-50	0.88813	0.31958	0.33445	
MLP-hidden_layer_sizes=(512,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	MFCC-200	0.98308	0.34548	0.36115	
MLP-hidden_layer_sizes=(128,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	MFCC-200	0.97377	0.33040	0.34446	
MLP-hidden_layer_sizes=(256,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	MFCC-200	0.98218	0.33440	0.35981	0.37500
MLP-hidden_layer_sizes=(1024,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	MFCC-200	0.98268	0.35603	0.36983	
MLP-hidden_layer_sizes=(128,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	SoundNet-y_scns-1024	0.06703	0.06509	0.06800	
MLP-hidden_layer_sizes=(1024,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	SoundNet-y_scns-1024	0.18468	0.17938	0.41067	0.46428
MLP-hidden_layer_sizes=(2048,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	SoundNet-y_scns-1024	0.21562	0.20740	0.43400	0.06250
MLP-hidden_layer_sizes=(4096,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	SoundNet-y_scns-1024	0.15309	0.14580	0.40360	0.13839
MLP-hidden_layer_sizes=(100,50,25,50,100), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	SoundNet-y_scns-1024	0.21019	0.20220	0.41667	
MLP-hidden_layer_sizes=(128,256,512,1024,2048), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	SoundNet-y_scns-1024	0.15627	0.14471	0.36333	
MLP-hidden_layer_sizes=(512,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	PANNs-2048	0.98319	0.69615	0.71733	
MLP-hidden_layer_sizes=(1024,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	PANNs-2048	0.97986	0.69774	0.72000	0.72767

MLP-hidden_layer_sizes=(1024,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	PaSST-scene-1295	0.95531	0.73470	0.75050	
MLP-hidden_layer_sizes=(2048,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	PaSST-scene-1295	0.92860	0.74963	0.77133	
MLP-hidden_layer_sizes=(4096,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000	PaSST-scene-1295	0.94878	0.75043	0.76267	0.80357

Confusion matrix and best model



The model with highest average validation top-1 accuracy is the MLP with hidden layer size of 4096, ReLU activation, Adam solver, alpha of 1e-3, and max_iter of 2000, trained from PaSST-scene embeddings features. The best top-1 validation accuracy on it is 0.76267. However, the best top-1 validation accuracy on all my models actually is the MLP with hidden layer size of 2048, other parameters are same. Since I have limited times to test on the official testset, I choose the hidden layer size of 4096 to submit, and get test accuracy of 0.80357. To reproduce the results, train MLP using *python3.9 train_mlp.py passt/scene 1295 labels/train_val.csv models/passt-hidden_layer=4096.mlp.model*, and test it by *python3.9 test_mlp.py models/passt-hidden_layer=4096.mlp.model passt/scene 1295 labels/test_for_students.csv results/passt-hidden_layer=4096.mlp.csv*. The process should take about 40-50 minutes.

There are some misclassified places in the confusion matrix. The most prominent one is the model predicting 7 (mowing_lawn) as 11 (shoveling_snow). It actually makes sense since both behaviors are using tools to remove things from the ground and the sounds they make do have similarities. To my surprise, the model did pretty well on classifying the three instrument activities: 8 (playing_drums), 9 (playing_guitar), 10 (playing_piano). However, when the true label is 12 (singing), sometimes the model predicts them as 8 (playing_drums) and 9 (playing_guitar). It might be because the person in the video singing with instrumentals or singing while playing an instrument. Drum and guitar are pretty common in any kind of music scene. Another important finding is, the model performed very well at percussive events. For example, in 4 (hitting_baseball), 8 (playing_drums), 13 (tapping_pen), and 14 (tickling), they have little misclassified cases.

Insights and Error analysis

During audio extraction, I got 8246 .mp3 and .wav files from 8249 videos. Three of them was ignored (LTQ5ODI3NjU5MTQ3OTQ4NTAwOQ==, NTkxNzA4MjE4OTM1ODg4NTYxOA==, LTgxOTM5Mzg2MTMwNzM4NjQzNzg=). The reason was there are three videos without sounds. During MFCC extraction, I got 8240 .csv. Besides the three videos without sounds, six more videos are lost (LTE0MDE2NzE2NTE5Nzc5NTIxMjY=, LTE0MDI0OTk2MzE2ODk4MDcxMDg=, LTE0MDUwNzEwMzYxMjUzNDU0NDk=, LTE0MDY5MDc3NjkyNTg3MjkzNzg=, LTE0MDc3MzUzOTE3NTQyMDYzOTA=, LTE0MTE3Njg0MzI4MzEzNTY1ODU=). It is mentioned in the writeout that this situation is normal. I looked into the six videos, and find they are normal. The problem might cause by OpenSMILE.

For the SVM classifier, I tried RBF and Sigmoid kernels, and observed that RBF kernel works better on our task. The number of K-Means clusters improves the performance of all the models. For MLP training, the default maximum number of iterations is 200, and it causes my model did not converge when reached the maximum iteration. So I changed max_iter to 2000 and it worked well.

MFCC bag of features representations is relatively worse than other neuron features. The models with highest best validation accuracy trained by MFCC is SVM-RBF kernel by MFCC-200, which is 0.37717. Since I did not have enough Kaggle submission attempts, the best test accuracy I got was 0.37500 achieved by MLP (hidden_layer_sizes=(256,), activation=”relu”, solver=”adam”, alpha=1e-3, max_iter=2000) with MFCC-200.

SoundNet features are supposed to outperform MFCC much, but during my experiment, their accuracies are unstable. It has higher best validation accuracy than MFCC, but average training accuracy and average validation accuracy are lower. Average training accuracy and average validation accuracy are close, while best validation accuracy is significantly higher. This is a failure part in my experiment, I did not try to adjust the number of layers of SoundNet. I should try to match the number of layers of the SoundNet with the parameters of the MLP so that they achieve good results.

Generally, increase the size of hidden layer improve the performance, but when I input SoundNet features, it gets unstable. When I try to test it on the official test set, the result of hidden layer sizes of 2048 and 4096 was bad, much lower than the result of hidden layer sizes of 1024. The best model with SoundNet features is MLP (hidden_layer_sizes=(1024,), activation="relu", solver="adam", alpha=1e-3, max_iter=2000), which got 0.41067 best validation accuracy and 0.46428 test accuracy.

The neuron features extracted using PANNs and PaSST are pretrained models trained with large datasets, and their performance does outperform MFCC and SoundNet. They have longer training times and larger feature dimensions. The dimension of PANNs is 2048 and the dimension of PaSST is 1295. Although the dimension of PaSST scene feature is smaller than that of PANNs, its accuracy is higher under the same conditions.

Running time

1. Extracting .wav and .mp3 audios from videos takes about half-hour respectively.
2. Extracting 40% of MFCCs takes about one and a half hours.
3. Extracting BoW features takes roughly 10 minutes for k=25, 15 minutes for k=50, and one hours for k=200.
4. Extracting SoundNet features takes about one and a half hours.
5. Extracting PANNs and PaSST scene features take two hours respectively.
6. Training the SVM models with MFCC features usually take less than one minute, and testing takes about one second. Training the SVM RBF model with SoundNet features takes 338 seconds, and testing takes 5 seconds. Training the SVM RBF model with PANNs features takes 313 seconds, and testing takes 6 seconds. Training the SVM RBF model with PaSST scene features takes 235 seconds, and testing takes 4 seconds.
7. Traning the MLP models with MFCC features and SoundNet features both usually take about 10 minutes, and testing takes about two seconds. Some cases with large BoW MFCC or large hidden layer size could make the time much longer. For example, MLP with bof_200 and hidden layer=1024 takes 1477 seconds to train; MLP with SoundNet feature and hidden layer=4096 takes 1389 seconds to train. Training the MLP model with PANNs features takes 15-30 minutes depend on the hidden layer size, and testing takes about 3 seconds. Training the MLP model with PaSST scene features takes 10-50 minutes depend on the hidden layer size, and testing takes about 2 seconds.

AWS credits

I performed all computations on my laptop, so I have all my $50 AWS credits on my account.