Using Machine Learning to Classify Emotions from Voice Waves

Alexander Adams, Calvin Capulong, Samuel Cuervo, and Shida Yang

Abstract - Machine Learning utilizes methods that are capable of classifying a variety of different test subjects. For purposes of this assignment, we were given data samples of voices speaking two predetermined phrases while using various emotions. Data was split into training and test sets. The training set allowed us to build a model that could predict the labels for each wave in the test set. The most accurate method we found in order to correctly classify each emotion was a combination of a convolutional neural network and k-nearest neighbors in an ensemble majority voting learning model. A few distinct parameters were tested in the k-nearest neighbors algorithm and convolution neural network to improve accuracy. The final experimentation resulted in our model reaching an 88.75% accuracy score when labeling the test data.

I. Introduction

Through a combination of a convolutional neural network and k-nearest neighbors in an ensemble majority voting learning model, our model can classify between the emotions of (1) neutral, (2) calm, (3) happy, (4) sad, (5) angry, (6) fearful, (7) disgust, and (8) surprise. Each ensemble model trained using a pre-provided data set with different readings and different emotions. The data set was pre-processed using MFCC to extract a feature set to train our models. Using MFCC for pre-processing the data, the complexity of training all the different models for ensemble learning was reduced significantly. For the k-nearest neighbor model, SFTF was introduced in conjunction with MFCC for a simple Fourier transform feature extraction. SFTF and MFCC were concatenated together for the k-nearest neighbor training data. Once all models had been trained, the top three predictions from all the CNN models were compared with the k-NN prediction to see if any of the best predictions agree with k-NN.

II. Implementation

MFCC Preprocessing - MFCC, or Mel-frequency cepstral coefficients, are the central coefficients that we used as features. MFCC works by windowing the input signal, applying Mel filterbanks in the frequency domain after a Fourier transform (FFT) on each window, and extracting features from a discrete cosine transformation (DCT) on the filterbank outputs [1]. The MFCC pre-processing technique is very popular for speech recognition as the mel domain and cepstral coefficients attempt to replicate how our ears listen to the sound.

k-Nearest Neighbors (k-NN) - k-Nearest Neighbors or k-NN is one of the vital models for the ensemble decision. k-NN uses the training data set as anchor points and then finds the distance between the anchor and test points to determine what class that test point falls under. The decision is based on a majority of neighbors (a parameter set by the user to determine how many neighbors to compare to), but each point can be given a weight so that outlier points and points close to boundaries can be properly classified. Our k-NN model was verified using k-fold cross validation which found three neighbors, manhattan, and distance weights parameters to have the highest accuracy. Manhattan finds the distances using the formula sum(|x-y|). The three neighbors parameter checked the three closest points to the test point and determined the classification based on weighted distance values from the test point. We provided the k-NN model with our chosen parameters and the training data(run through MFCC and SFTF pre-processing) with its associated labels. Each test was then compared to the training data using the manhattan formula described above. The predictions from k-NN were used later for the overall ensemble learning model.

Convolutional Neural Network (CNN) - The input to our CNN model was the MFCC of the audio signals. Our CNN architecture has two major parts: the convolution layers and the dense layers. The convolution layers use convolution to extract features from the MFCC. We used four convolution layers with different filter sizes in order to capture differently sized features from the MFCC. The features were flattened before passing into the three dense layers to make final classifications. The dropout layers were used throughout the architecture to reduce overfitting. For convolution layers, the dropout rates were 0.3 because we wanted to keep most information that might be useful later on. The dropout rates for dense layers were 0.5 to reduce overfitting. We also used batch normalization layers in between convolution layers and dense layers to prevent vanishing gradient.

Ensemble Learning - We trained 20 learners to perform ensemble learning. Each learner was trained using the same CNN architecture but a randomly sampled subset of the training data. When making the

final decision, we summed the softmax output of each learner and chose the class with the largest confidence value. Since each learner had a different randomly sampled training set, it was unlikely for them to agree on the same output due to overfitting, so the combined decision was likely the correct output.

Combining k-NN and Ensemble Learners - When examining the combined softmax output of our ensemble model, sometimes two or three classes had similar confidence values, and that's usually when misclassifications occurred. In our final test algorithm, when there was a tie, we chose one of the top three ensemble learning outputs on which our k-NN model agreed.

III. Experiments

k-Nearest Neighbors - Finding the best parameters for k-NN starts with extracting the best features from the audio signal. A few features were considered including MFCC with 13 coefficients, STFT, Mel spectrogram, chromagram, zero-crossing rate, root mean square energy, and spectral roll-off. Due to their ability to quantitatively describe various aspects of an audio signal, these are often the most important features to extract for a classification model [1].

Furthermore, since the k-NN algorithm struggles with classifying three-dimensional data, and these features are often extracted in the third or higher dimensions, some form of dimensional reduction was required. We opted to simply find the mean along one axis by creating a 1xM vector for each data point based on the length of each feature. All features were then combined into a single matrix size N x $M_1 + \ldots + M_i$ where N is the number of data points, and M is the length of each feature.

After fitting a model with various combinations of features, we found that MFCC and STFT performed the best. One reason is that these features contain the most information about the whole audio. Certain features such as ZCR and RMSE only contain a single value, and their inclusion creates confusion within the model. Another reason is that some of these features are redundant with many being covered by MFCC.

With the best features extracted, we then needed to tune the hyperparameters of our k-NN model. This was performed through a grid search cross-validation by comparing metrics of euclidean, manhattan, and chebyshev, weights of uniform and distance, and nearest neighbor values between 3 and 30. The cross validation found that the model worked best with three neighbors, manhattan metric, and distance-based weights.

CNN Number of Epochs - When determining the number of epochs we needed, we trained the model with a large number of epochs and checked the training loss and validation loss plot as shown below. The testing model has the following parameter settings: learning rate = 0.0005, batch size = 64. Initially, we expected the validation loss function to decrease and increase again after a certain number of epochs due to the model overfitting the training data. However, our validation loss did not go back up, so we chose a number of epochs with which the validation loss stabilized at a low value.

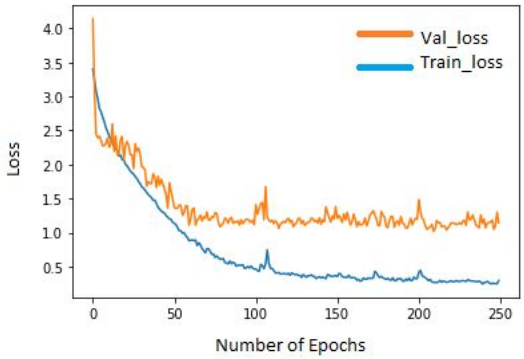

Fig. 1. Validation loss and training loss plots used to find the required number of epochs

CNN Learning Rate - The learning rate determines if a model converges to an optimal solution and how fast it converges to that solution, so we needed to find a proper learning rate to make our model work well. We tested different learning rates as shown in the plot below. The testing model had a batch size of 64. When the learning rate was too low, the loss function had a flat slope and converged very slowly, which means the training took an extended amount of time. When the learning rate was too large, the loss function became unstable and was unable to converge to an optimal solution. Therefore, our final choice for the learning curve was 0.0005, which offered a balance between stability and convergence speed.

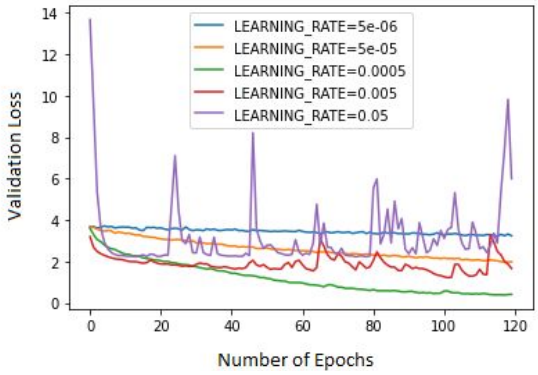

Fig. 2. Plot of validation loss with different learning rates

CNN Batch Size - Choosing a proper batch size will make the model converge fast, smoothly, and to a favorable solution. We tested different batch sizes as shown in the plot below. The testing model had a learning rate of 0.0005. When the batch size was small, the model converged quickly, but to a less optimal solution and had an unstable loss function. When the batch size was large, the model converged slower to a better solution with a stable loss function. To pick a balance between the convergence speed and the final solution quality, we used a batch size of 64 in our final design.

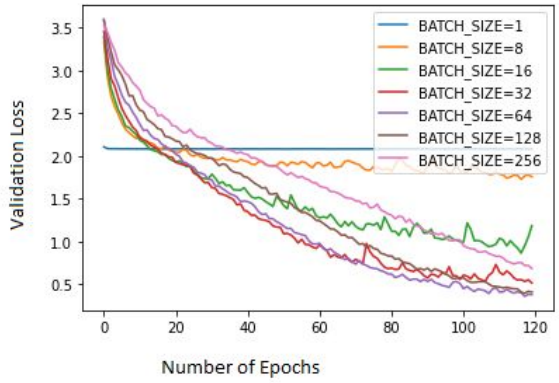

Fig. 3. Plot of validation loss with different batch sizes

CNN Data Augmentation - Data augmentation helps reduce overfitting by adding more data to the training set. The plot below shows the validation loss when performing data augmentation using different levels of noise. Without data augmentation, the loss function showed high variance and bias. When the noise was low, we essentially just added more similar data points to the training set, so the loss function had a low bias and variance. As the noise increased, the loss function still had low variance as it had a lot more data points compared to the original set. However, its bias grew higher because the training set became dissimilar to the validation set. Therefore, in our final model, we used noise equal to 0, which just added a copy of our training data.

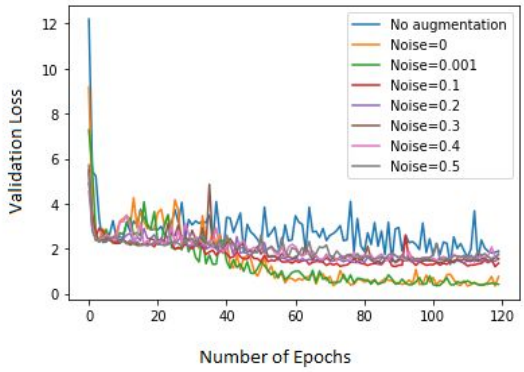

Fig. 4. Plot of validation loss with data augmentation using different amount of noise

Number of Ensemble Learners - In the graph below, the horizontal axis is the number of models and the vertical axis is the test accuracy. As we added more models, the validation accuracy increased and became more stable as shown by the blue line even when some of the models we are adding are performing poorly. We decided to use 20 learners in our final design since the test accuracy stabilizes at around 82% after 20 learners, and adding more just slowed down our model without adding much benefit.

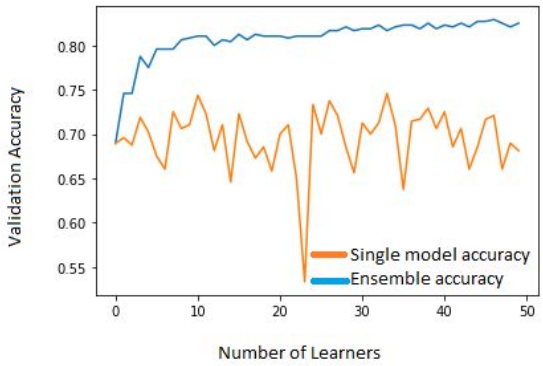

Fig. 5. Plot of ensemble learning model accuracy as learners are added

IV. Conclusions

Our model was able to accurately label test data at a rate of 88.75%. Based on the confusion matrix, most misclassifications came from class 1 and class 2, which was not surprising since calm and neutral emotions sound similar. For other more distinct emotion classes, there weren't many misclassifications. Therefore, we are very confident that our fusion of the k-nearest neighbors algorithm and an ensemble-learning convolution neural network can predict vocal emotions in other test sets.

```
[54,  9,  1,  1,  0,  0,  0,  0]
[13, 46,  0,  0,  0,  0,  1,  0]
[ 0,  0, 46,  0,  1,  3,  0,  0]
[ 0,  0,  0, 49,  0,  1,  0,  0]
[ 0,  0,  0,  0, 54,  2,  1,  1]
[ 0,  0,  0,  3,  0, 64,  1,  0]
[ 0,  1,  2,  0,  0,  3, 59,  4]
[ 0,  0,  2,  0,  1,  2,  1, 54]
```

Fig. 6. Final model confusion matrix

References

[1] Sirko Molau, Michael Pitz, Ralf Schluter and Hermann Ney. (2001) “Computing Mel frequency Cepstral Coefficients on the power spectrum.”IEEE Transactions on Audio, Speech and Language Processing

[2] J. Jogy, “How I Understood: What features to consider while training audio files?,” *Towards Data Science*, 06-Sep-2019. [Online]. Available: https://towardsdatascience.com/how-i-understood-what-features-to-consider-while-training-audio-files-eedfb6e9002b. [Accessed: 11-Dec-2020].