# Human Emotion Detector Final Update

Tofu Family

# Outline

Introduction

Dataset

Feature Extraction

Baseline Model

More Feature Extraction

NN Model and Metrics

Conclusion

# Our Family

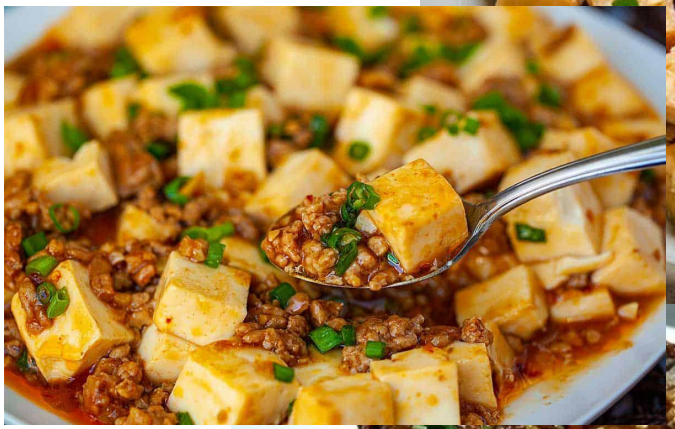
Tiffany Tse

Shulin Li

Jake Zhong

# Tofu Family

All of our team members are tofu lovers 😋



# Project Description

When we were brainstorming project ideas, some questions came to our mind:

- If we were business owners, how could we know about our customer experience without interrupting their shopping?
- If we were teachers, how could we know if one of our students was in depression and help him/her out from this situation?

With these questions in mind, we decided to create a speech  classifier that can detect human emotion by their voice.

## Additional use cases:

- Detect emotion in automated calls to speed-up customer service..
- Detect emotion in WhatsApp chatbots voice notes.
- Improves tasks in smart speakers (alexa, google assistant, etc.)

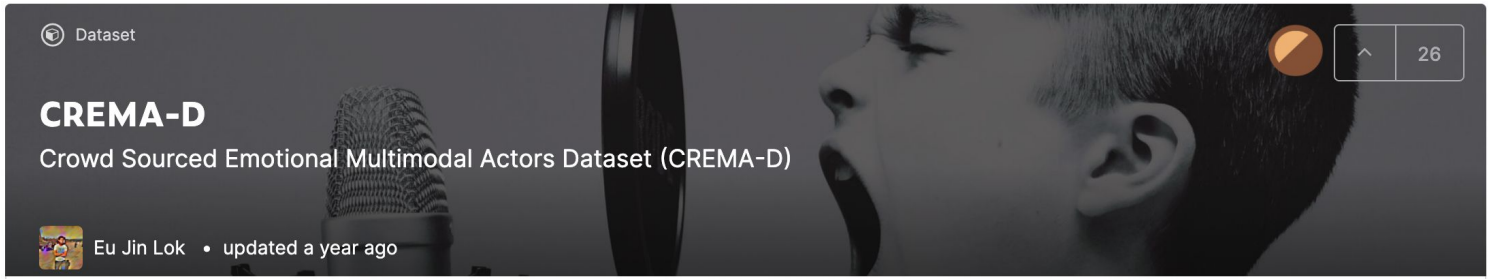
# Prediction (Goal)

We aim to discern six different emotions from ~7K+ audio speeches.

# Dataset

## Crowd Sourced Emotional Multimodal Actors Dataset (CREMA-D)

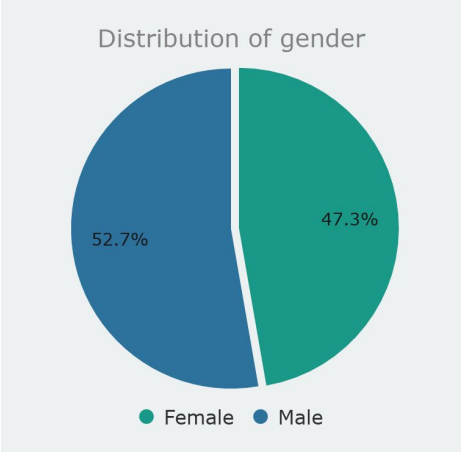
CREMA-D contains **7442** audio clips from **91** actors, including **48** male and **43** female actors,  between the ages of **20** and **74** coming from a variety of races and ethnicities.
Actors spoke from a selection of **12** sentences. The sentences were presented using one of **6** different emotions (Anger, Disgust, Fear, Happy, Neutral and Sad).
original paper: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4313618/
github: https://github.com/CheyneyComputerScience/CREMA-D

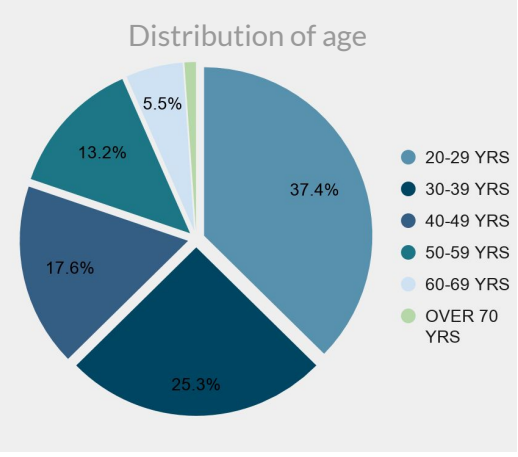


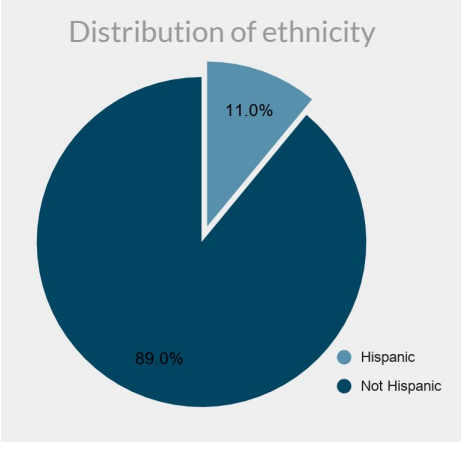
# Distribution of Actors

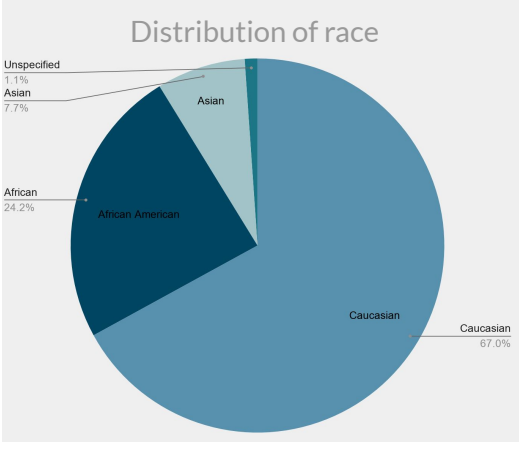
| | ActorID | Age | Sex | Race | Ethnicity |
|---|---|---|---|---|---|
| **0** | 1001 | 51 | Male | Caucasian | Not Hispanic |
| **1** | 1002 | 21 | Female | Caucasian | Not Hispanic |
| **2** | 1003 | 21 | Female | Caucasian | Not Hispanic |
| **3** | 1004 | 42 | Female | Caucasian | Not Hispanic |
| **4** | 1005 | 29 | Male | African American | Not Hispanic |
| **...** | ... | ... | ... | ... | ... |
| **86** | 1087 | 62 | Male | Caucasian | Not Hispanic |
| **87** | 1088 | 23 | Male | African American | Not Hispanic |
| **88** | 1089 | 24 | Female | Caucasian | Not Hispanic |
| **89** | 1090 | 50 | Male | Asian | Not Hispanic |
| **90** | 1091 | 29 | Female | Asian | Not Hispanic |

91 rows × 5 columns



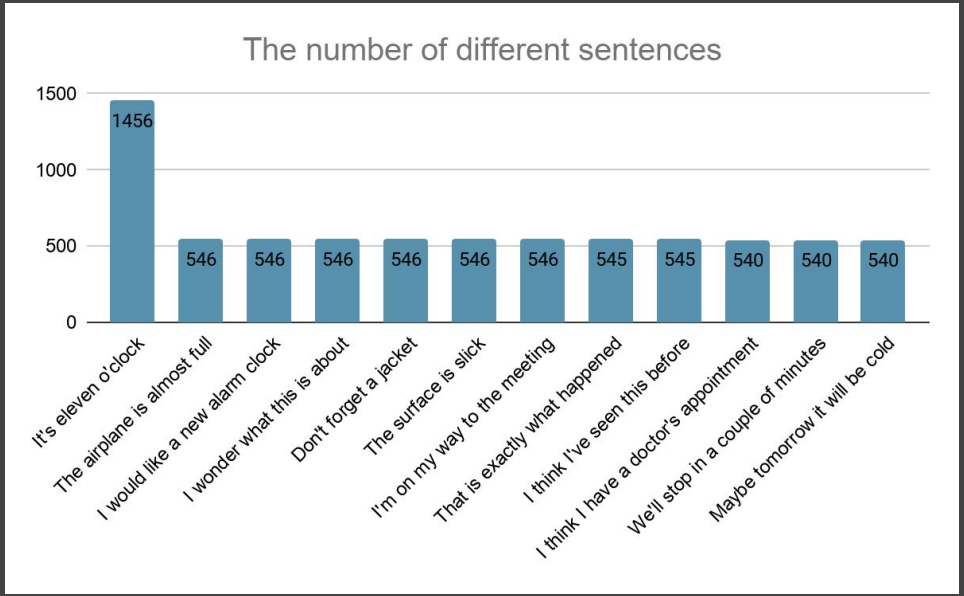



# Sentences in audio

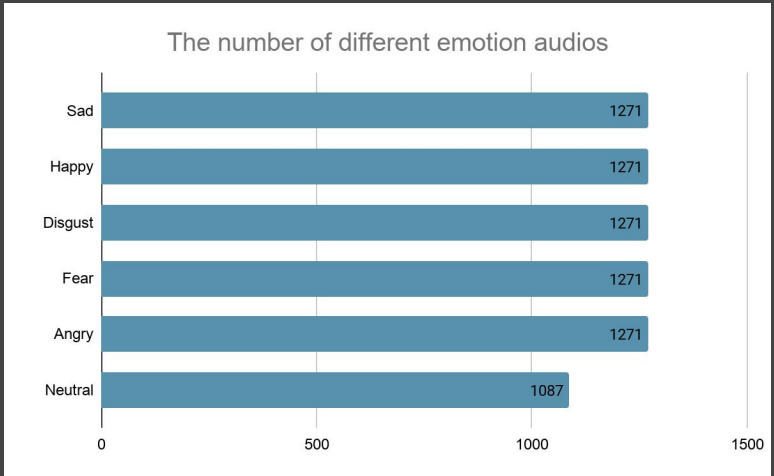
- It's eleven o'clock.
- That is exactly what happened.
- I'm on my way to the meeting.
- I wonder what this is about.
- The airplane is almost full.
- Maybe tomorrow it will be cold.
- I would like a new alarm clock
- I think I have a doctor's appointment.
- Don't forget a jacket.
- I think I've seen this before.
- The surface is slick.
- We'll stop in a couple of minutes.

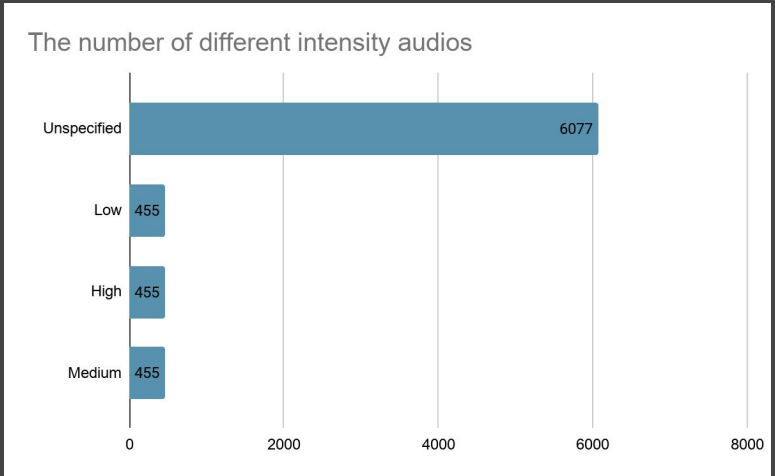


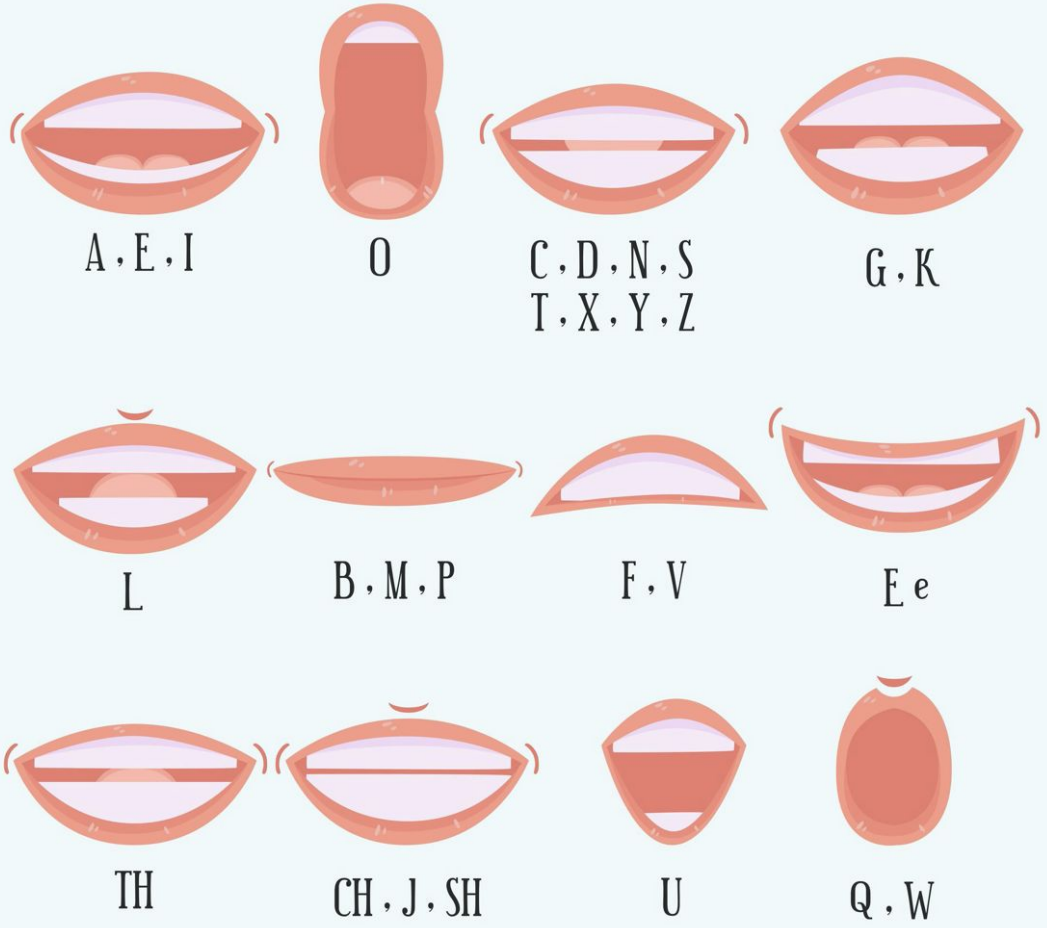
# Emotions in audio

**Emotion**
Anger
Disgust
Fear
Happy
Neutral
Sad

**Intensity**
Low
Medium
High
Unspecified






# Feature Extraction

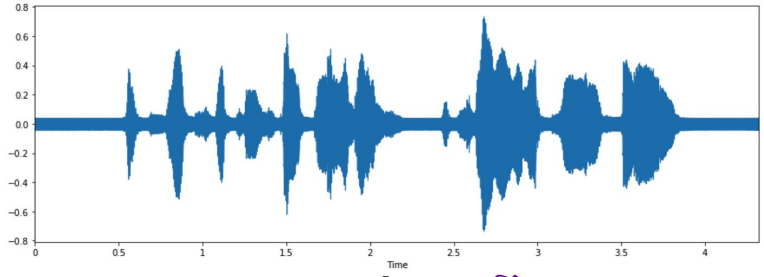
# MFCC

The MFCC feature extraction technique basically includes:

1) Windowing the signal
2) Applying the DFT
3) Taking the log of the magnitude
4) Warping the frequencies on a Mel scale
5) Applying the inverse DCT

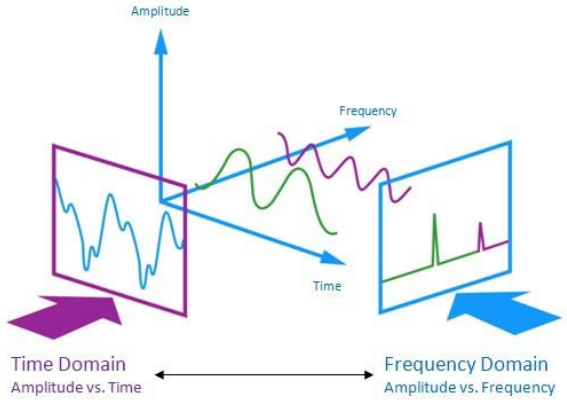
# MFCC Extraction

XXX.wav → librosa → Data, Sampling rate → librosa



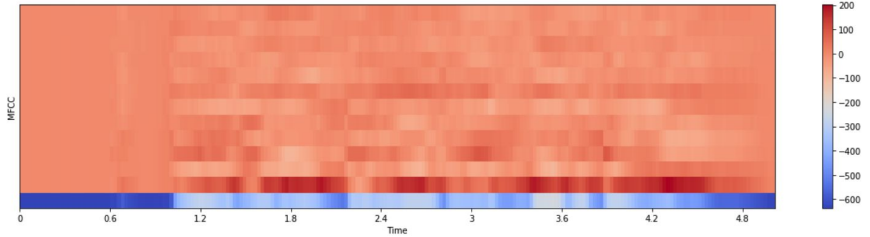

$$n = \frac{sr + (\frac{sr}{2})}{2} = \frac{3 * sr}{4}$$

Source by:
https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1685&context=etd_projects

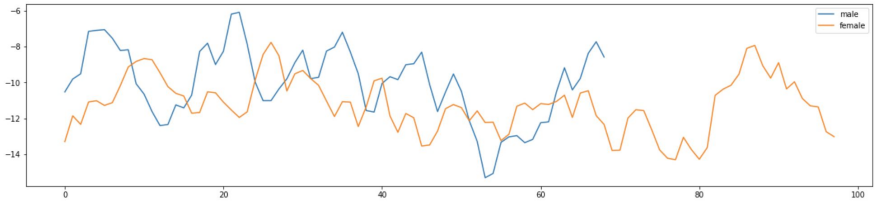



Source by:
https://www.kaggle.com/shivamburnwal/speech-emotion-recognition

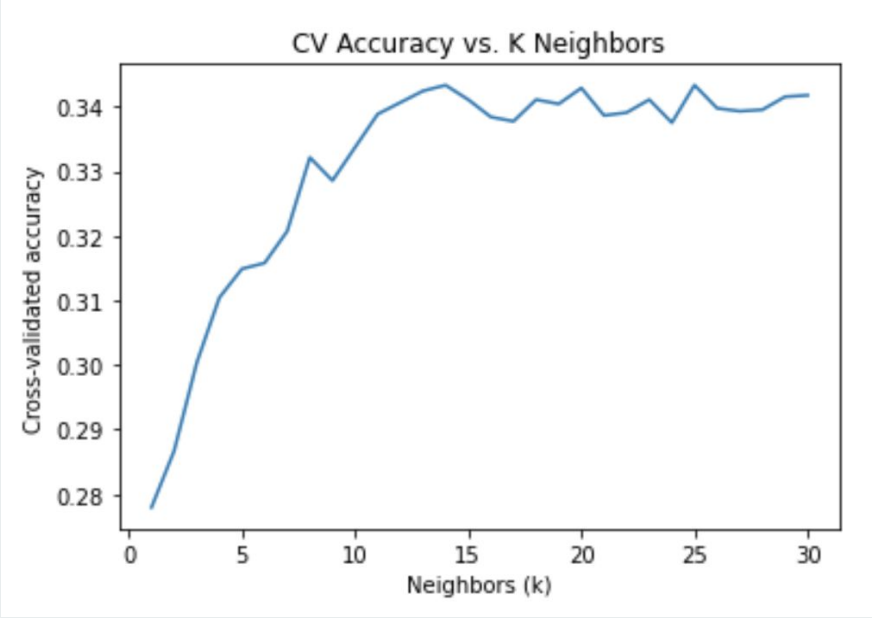
librosa




NumPy



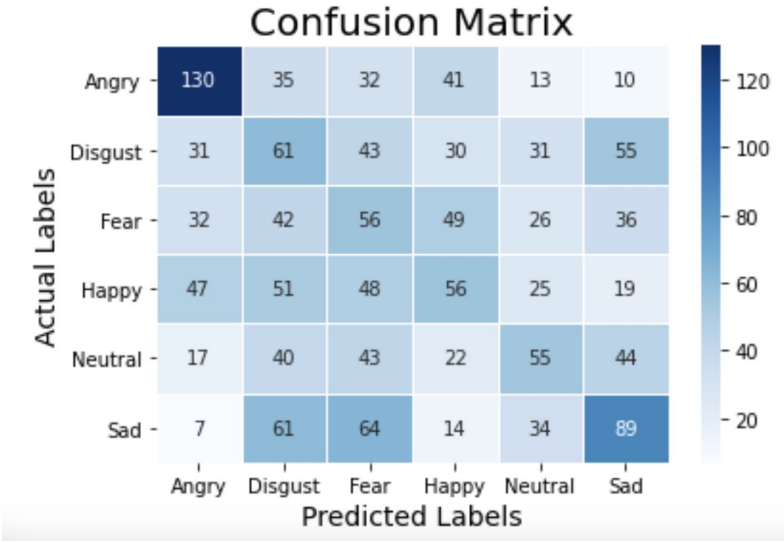
# Model 1: KNN




## Why KNN?

Simple classification algorithm with few hyperparameters to tune, but fast run-time at the cost of some accuracy. It computes distance with each vector in the dataset and a majority voting is applied to k vectors with the smallest distances.

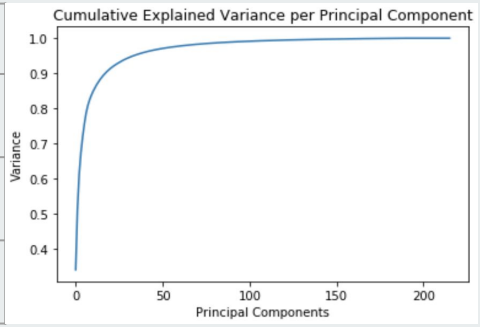
Accuracy (K= 5): 0.300
Accuracy (K = 20): 0.343



# Model 2: SVC

Dimensionality reduction (PCA) was performed and tested on 3 levels of explained variance.

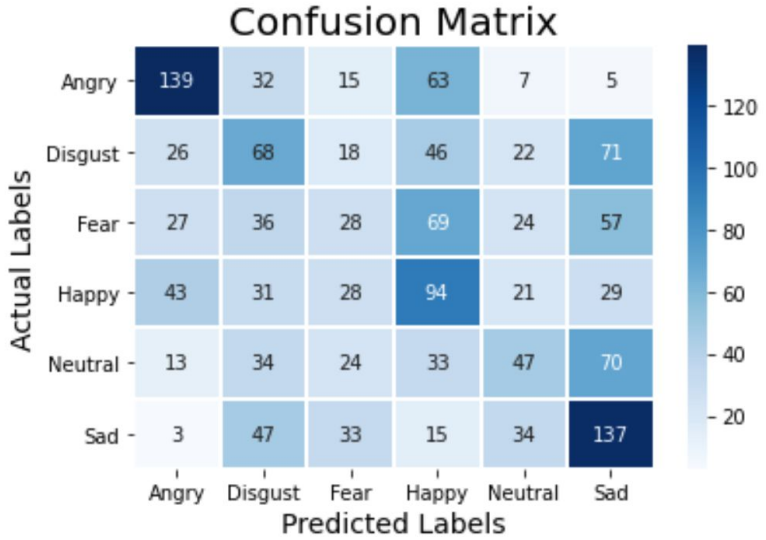
| PCA Dimension | RBF Accuracy |
|---|---|
| 17 | 0.379 |
| 34 | 0.361 |
| 65 | 0.367 |




## Why SVC?

Unstructured data like audio may be comprehended better using a SVM kernel trick. Classifiers that are non-linear can become linear in the feature space by selecting the appropriate non-linear kernel function.

Accuracy (RBF): 0.345
Accuracy (poly): 0.287



# Model 3: DNN

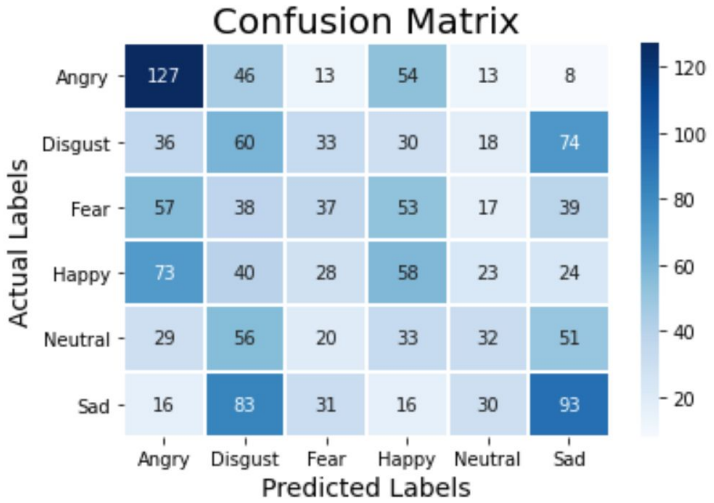
```
def __init__(self, n_inputs, n_outputs, hidden_size):
    super(FeedForward, self).__init__()
    self.inputs = n_inputs
    self.outputs = n_outputs
    self.hidden = hidden_size

    self.fc1 = torch.nn.Linear(self.inputs, self.hidden)
    self.fc2 = torch.nn.Linear(self.hidden, self.hidden)
    self.fc3 = torch.nn.Linear(self.hidden, self.hidden)
    self.fc4 = torch.nn.Linear(self.hidden, self.hidden)
    self.fc5 = torch.nn.Linear(self.hidden, self.outputs)

    self.relu = torch.nn.ReLU()
    self.init_weights()
```

## Why DNN?

DNN(or MLP) is a class of feedforward artificial neural network which is easy to train. -> overfitting

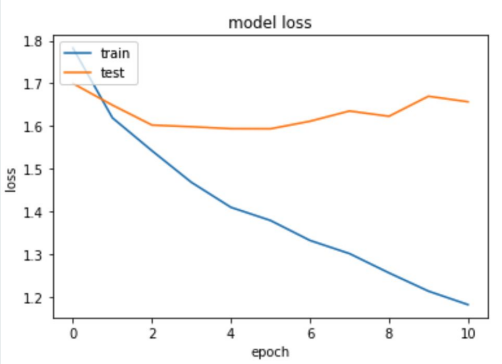
Loss: 2.71

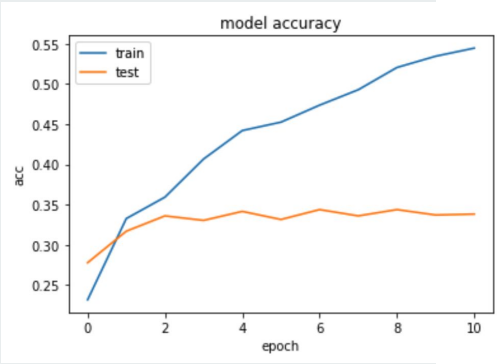
Accuracy: 0.273



# Model 4: LSTM

```
def create_model():
    model = Sequential()
    model.add(InputLayer(input_shape=(1, 216)))
    model.add(SpatialDropout1D(0.1))
    model.add(LSTM(256, dropout=0.1, recurrent_dropout=0.2))
    model.add(Dense(6, activation='softmax'))
    opt = Adam(learning_rate=0.0001)
    model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=opt)
    return model
```

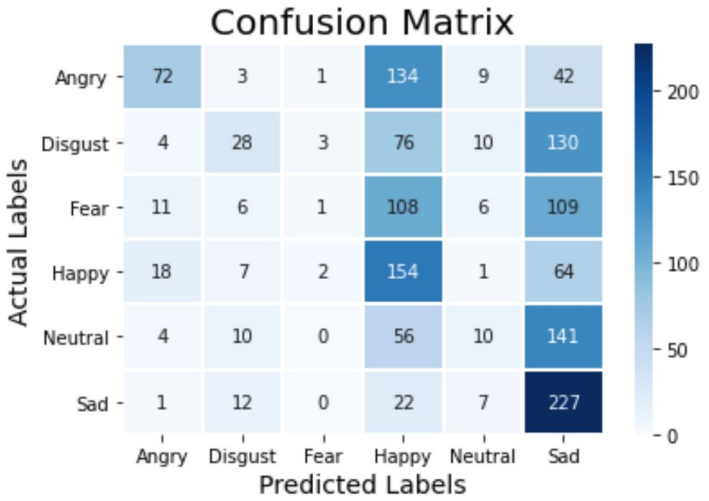



## Why LSTM?

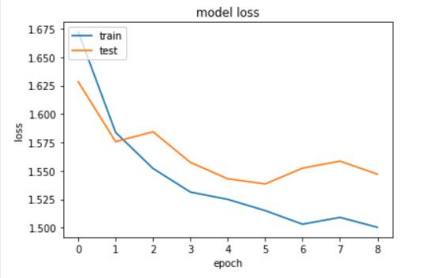
LSTM(Long short-term memory) are well-suited to classifying, processing and making predictions based on time series data.

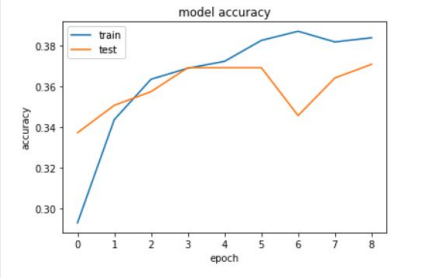
Batch_size: 16, Learning Rate: 0.0001
Loss: 1.594
Accuracy: 0.330



# Model 5: CNN

```
# New model
lr=0.0001
model = Sequential()
model.add(Conv1D(256, 8, padding='same',input_shape=(X_train.shape[1],1)))
model.add(Activation('relu'))
model.add(Conv1D(256, 8, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(MaxPooling1D(pool_size=(8)))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(MaxPooling1D(pool_size=(8)))
model.add(Conv1D(64, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(64, 8, padding='same'))
model.add(Activation('relu'))
model.add(Flatten())
model.add(Dense(classes))
model.add(Activation('softmax'))
opt = keras.optimizers.RMSprop(lr=lr, decay=1e-6)
model.summary()
```



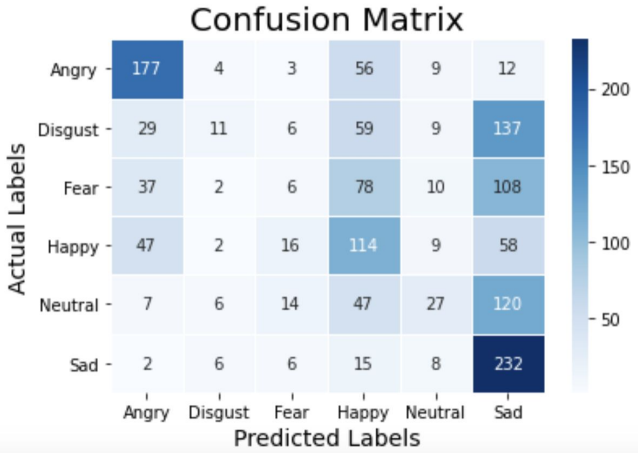

## Why CNN?

CNN's typically make good classifiers and perform particular well with image classification tasks, which will be very effective at finding patterns within the MFCCs.

Batch_size: 16, Learning Rate: 0.0001

Loss: 1.503

Accuracy: 0.385



# Baseline Performance Comparison

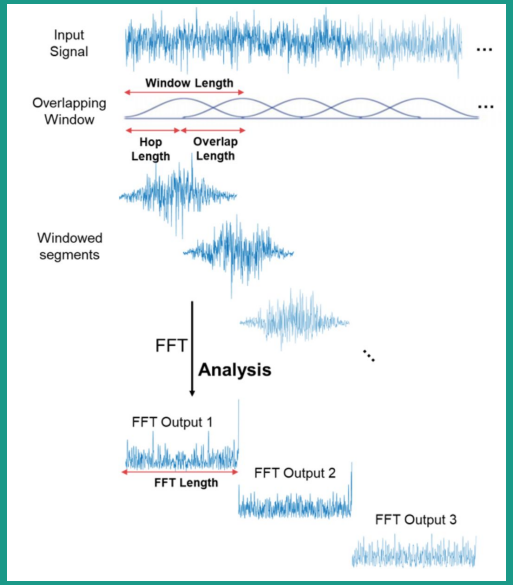
| Model | KNN | SVC | DNN | LSTM | CNN |
|---|---|---|---|---|---|
| Accuracy | 0.343 | 0.345 | 0.273 | 0.330 | 0.385 |
| Potential | Low | Low | Medium | High | High |

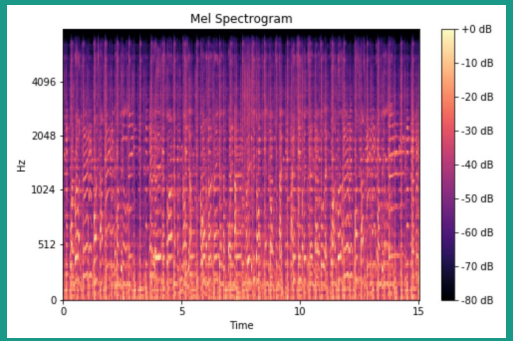
# More Feature Extraction

- **Complete MFCC**
  The MFCC feature extraction technique basically includes windowing the signal, applying the DFT, taking the log of the magnitude, and then warping the frequencies on a Mel scale, followed by applying the inverse DCT.
- **Mel-Spectrogram**
  Mel spectrogram is a spectrogram where the frequencies are converted to the mel scale.
  Mel scale is a unit of pitch such that equal distances in pitch sounded equally distant to the listener.
- **Chroma**
  Chroma features represent for music audio in which the entire spectrum is projected onto 12 bins representing the 12 distinct semitones (or chroma) of the musical octave.
- **Zero Crossing Rate**
  The rate of sign-changes of the signal during the duration of a particular frame.

# More Feature Extraction

| | |
|---|---|
| Mean MFCC | [0.320] |
| Complete MFCC | [0.335] |
| MFCC.T + Mel-Spectrogram.T | [0.419] |
| MFCC.T + Chroma.T | [0.343] |
| MFCC.T + Zero Crossing Rate.T | [0.335] |
| MFCC.T + Mel-Spectrogram.T + Chroma.T + ZCR.T | [0.421] |

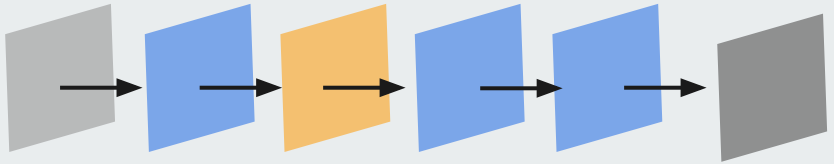



https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53

# Model 4: LSTM

```
def create_model():
    model = Sequential()
    model.add(InputLayer(input_shape=(1, 145)))
    model.add(Bidirectional(LSTM(256, dropout=0.2, recurrent_dropout=0.2, return_sequences = True)))
    model.add(SpatialDropout1D(0.1))
    model.add(Bidirectional(LSTM(256, dropout=0.2, recurrent_dropout=0.2, return_sequences = True)))
    model.add(Bidirectional(LSTM(256, dropout=0.2, recurrent_dropout=0.1)))
    model.add(Dense(6, activation='softmax'))
    opt = Adam(learning_rate=0.0001)
    model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=opt)
    return model
```

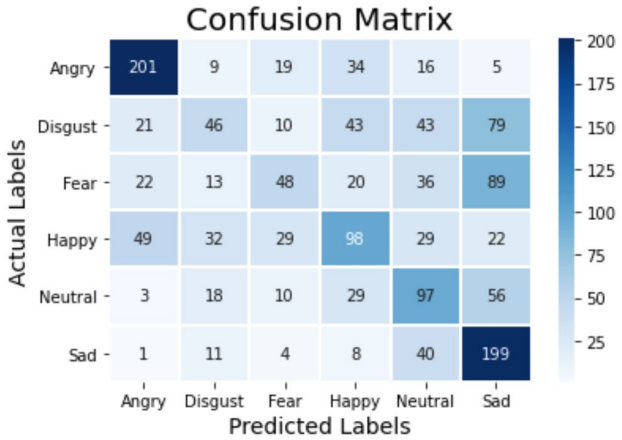


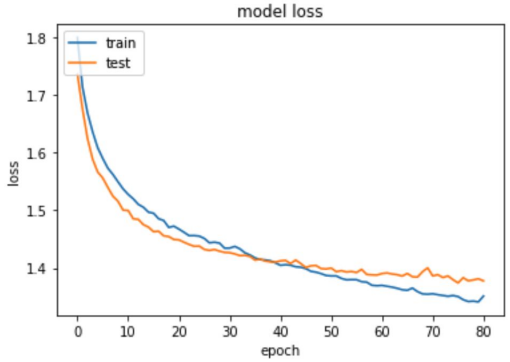

## Improvements

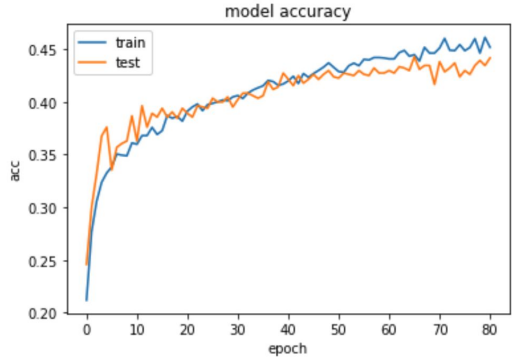
3 More Layers, Batch_size: 32, Learning Rate: 0.0001

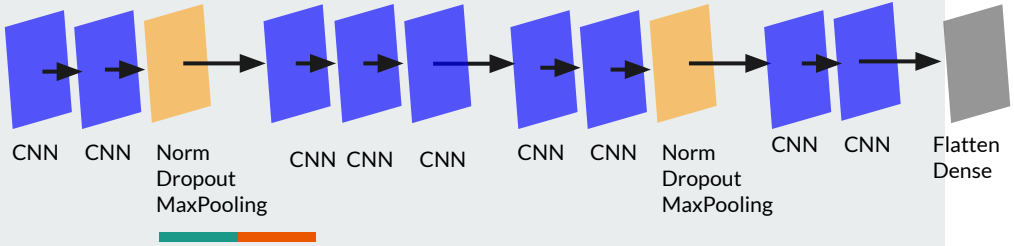
Loss: 1.594 -> 1.371

Accuracy: 0.330 -> 0.463





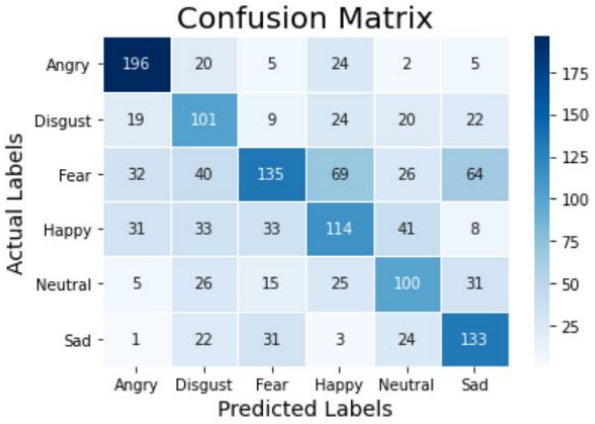

# Model 5: CNN

## Improvements

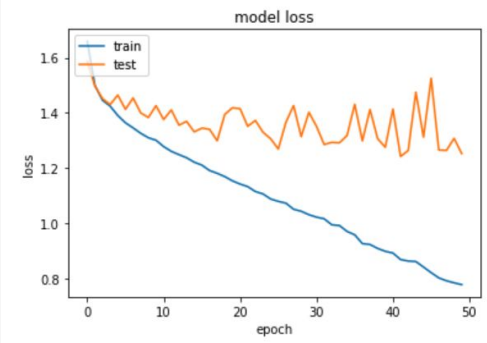
Batch_size: 8, Learning_rate: 0.00001

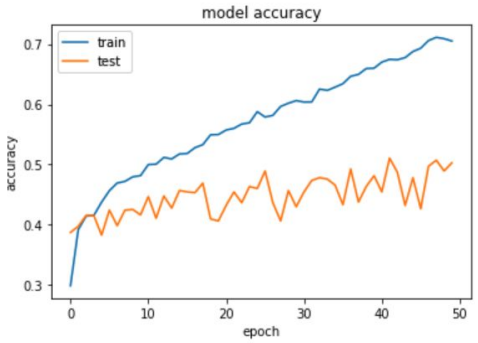
Loss: 1.503 -> 1.194

Accuracy: 0.385 -> 0.523



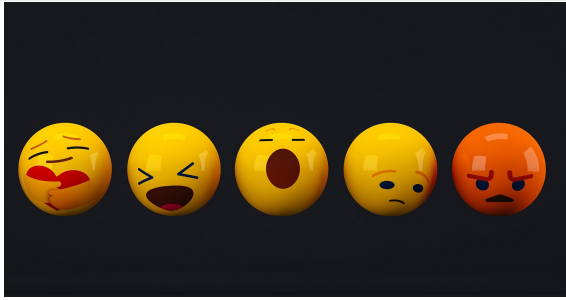

```
# New model
lr = 0.00001
model = Sequential()
model.add(Conv1D(256, 8, padding='same',input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Activation('relu'))
model.add(Conv1D(256, 8, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(MaxPooling1D(pool_size=(8)))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(MaxPooling1D(pool_size=(8)))
model.add(Conv1D(64, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(64, 8, padding='same'))
model.add(Activation('relu'))
model.add(Flatten())
model.add(Dense(6))
model.add(Activation('softmax'))

opt = keras.optimizers.RMSprop(lr=lr, decay=1e-6)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
model.summary()
```



# Improved Performance Comparison

| Model | **LSTM** | **CNN** |
| --- | --- | --- |
| **Loss** | 1.371 | 1.194 |
| **Accuracy** | 0.463 | 0.523 |



# Demo

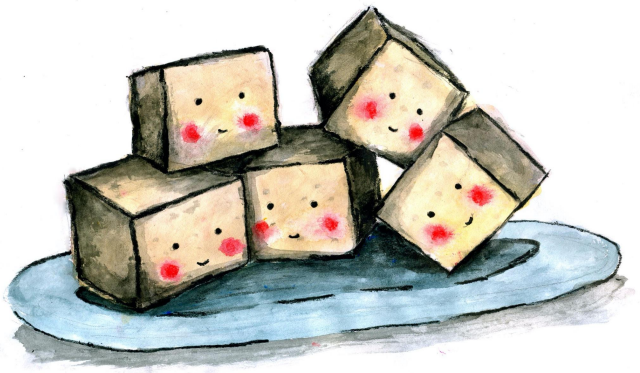
Happy, Sad, Anger, Neutral, Disgust, Fear

| Statement | Audio |
|---|---|
| It’s eleven o’clock. | |
| | |
| | |
| | |
| | |
| | |

| Answers | Predictions |
|---|---|
| Anger | Anger |
| Disgust | Disgust |
| Fear | Fear |
| Happy | Happy |
| Neutral | Happy |
| Sad | Sad |

# Concluding Remarks

- ➔ Re-classify the emotions into positive, neutral and negative.
- ➔ Separate gender observations
- ➔ Combine other open-source datasets (SAVEE, RAVDESS, TESS)
- ➔ Get more training data with data augmentation
- ➔ Continue feature exploration (tempo, MFCC differentials, etc.
- ➔ Test on new data



# Thank you!

# Questions?