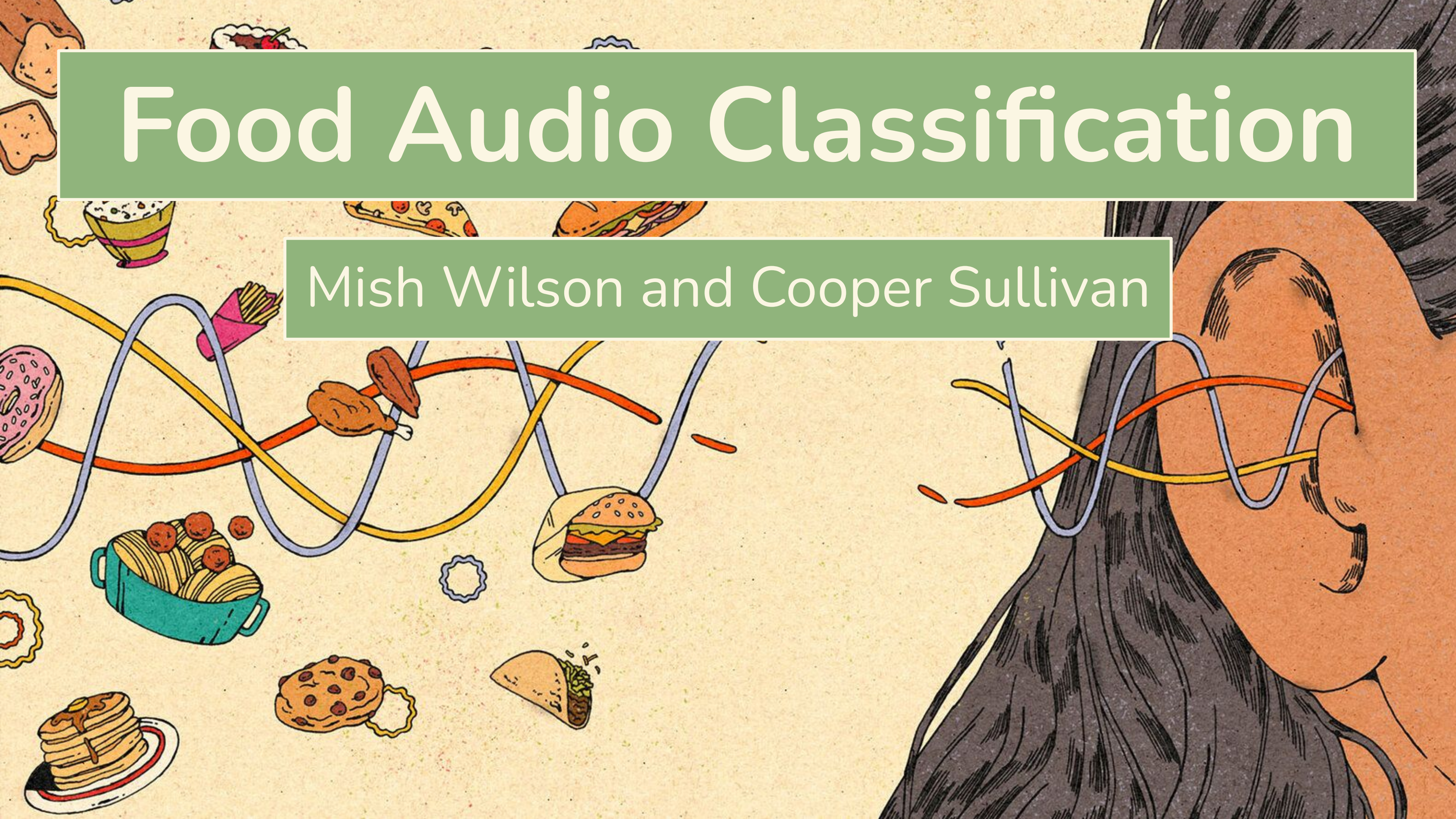
Food Audio Classification

Mish Wilson and Cooper Sullivan



Problem Description

Data Description

ML Approaches

Next Steps

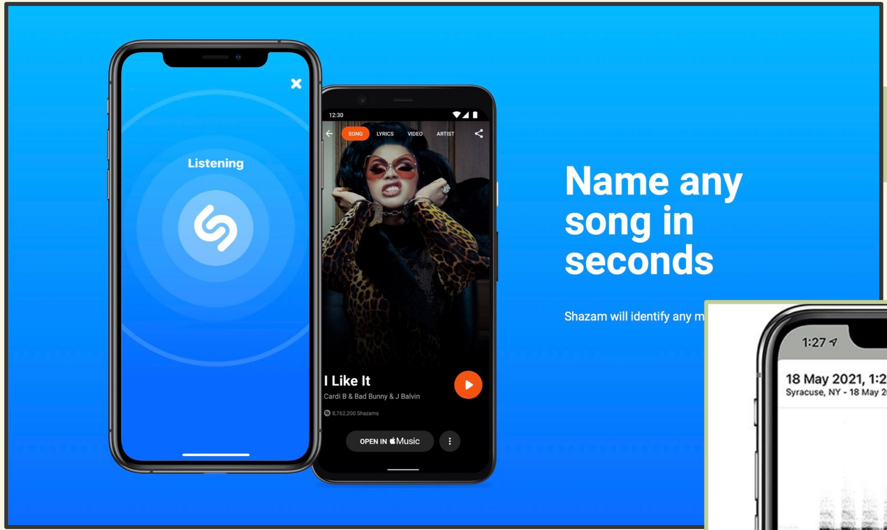
Problem Description

Data Description

ML Approaches

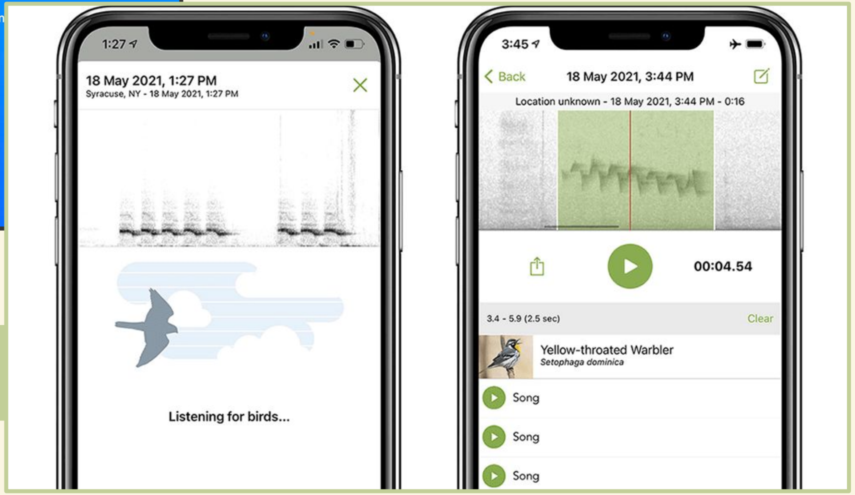
Next Steps

The problem: Develop a ML Model that can identify the type of food a person is eating based on sound.



<- Shazam

Merlin ->



Outside of novelty, we believe food audio classification could be an excellent supplement to other systems.

- Has potential to be useful for different purposes.
- Additional inputs into traditional vision based classification systems
- Assist people with low eyesight and dull taste buds
- Automatic Dietary tracking

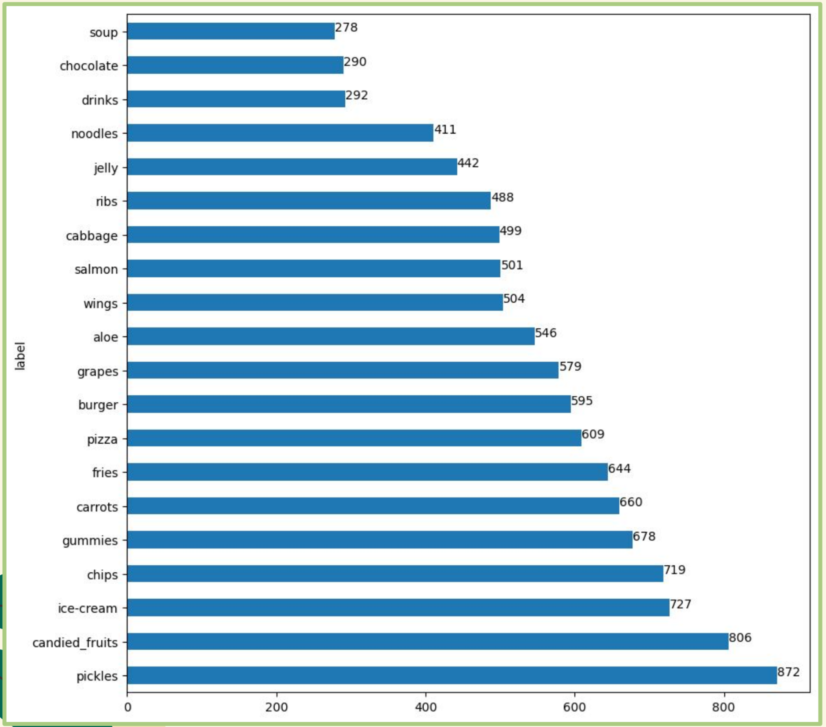
Problem Description

Data Description

ML Approaches

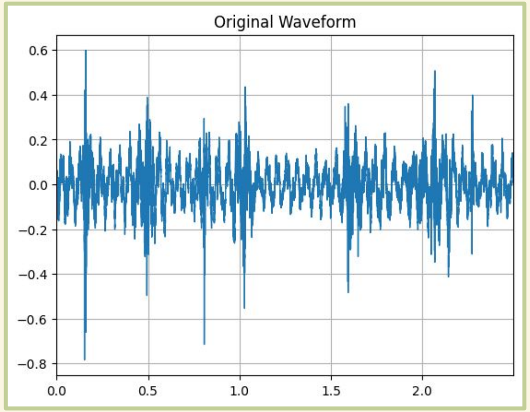
Questions?

The data for our model is comprised of audio clips from food ASMR Youtube videos.



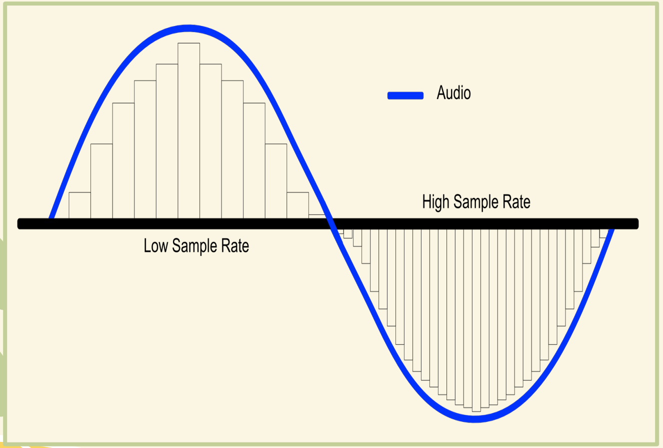
Summary Facts:

- 20 Food Categories (labels)
- 11.1k .wav clips
- ~12 videos per category
- 2 - 6 seconds in length

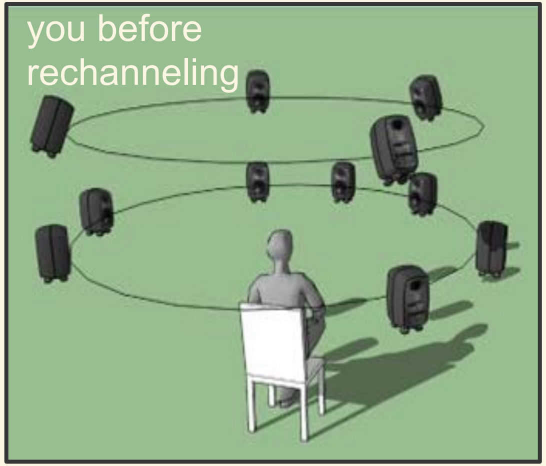


We apply standardized transformations to each clip to ensure uniformity and compatibility with the training model.

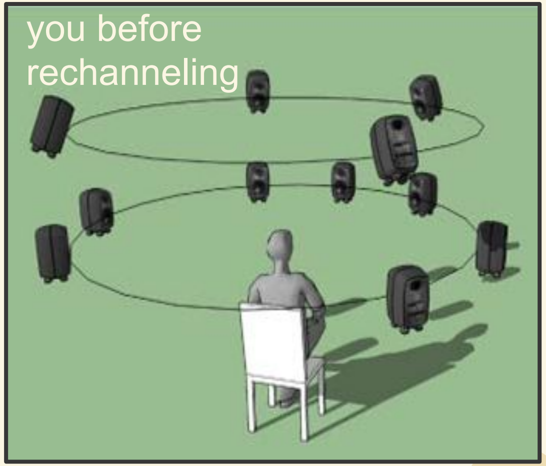
Resample: 44100 Hz



Resize: 3 Seconds



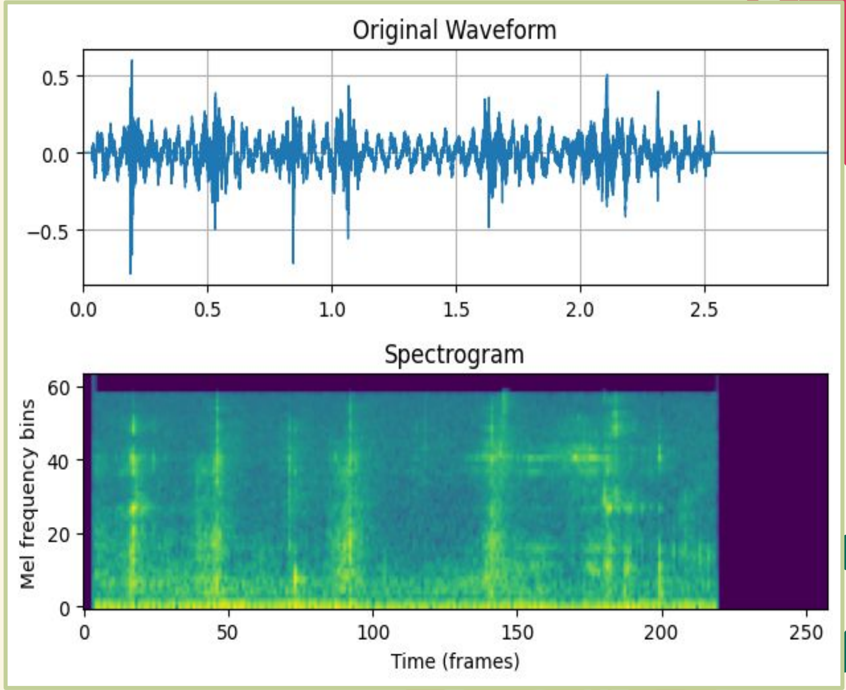
Rechannel: 1 Channel



Our audio data is then converted into a spectrogram, a visual representation of the frequencies in the audio signal

Acts as way to extract and emphasize features from the raw waveform

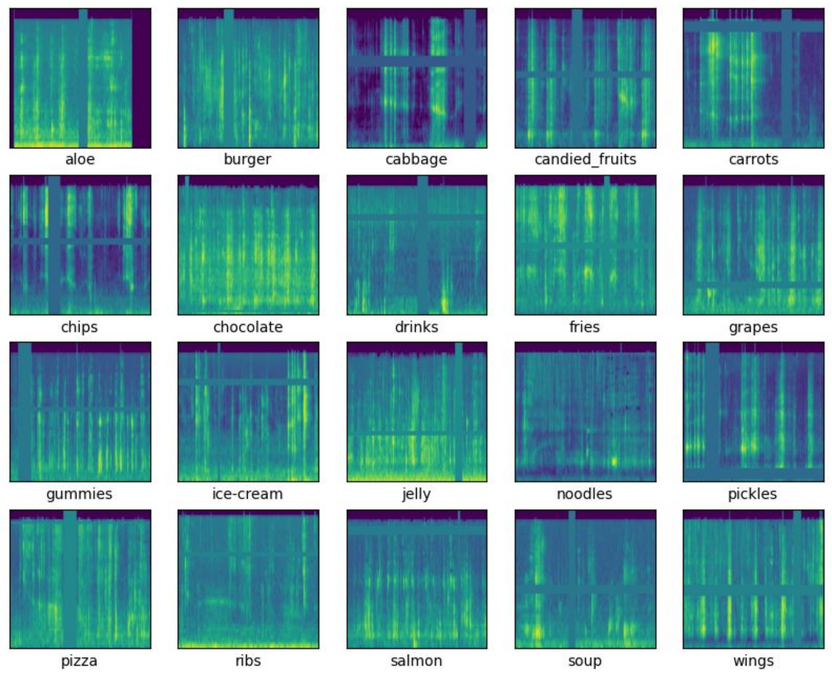
Useful for how we will later perform classification since spectrograms are visually distinct.



Frequency and time masks are add to the spectrogram to improve the models ability to generalize and its robustness.

Frequency mask: randomly block out a range of consecutive frequencies.

Time mask: randomly block out ranges of time from the spectrogram by using vertical bars.



Problem Description

Data Description

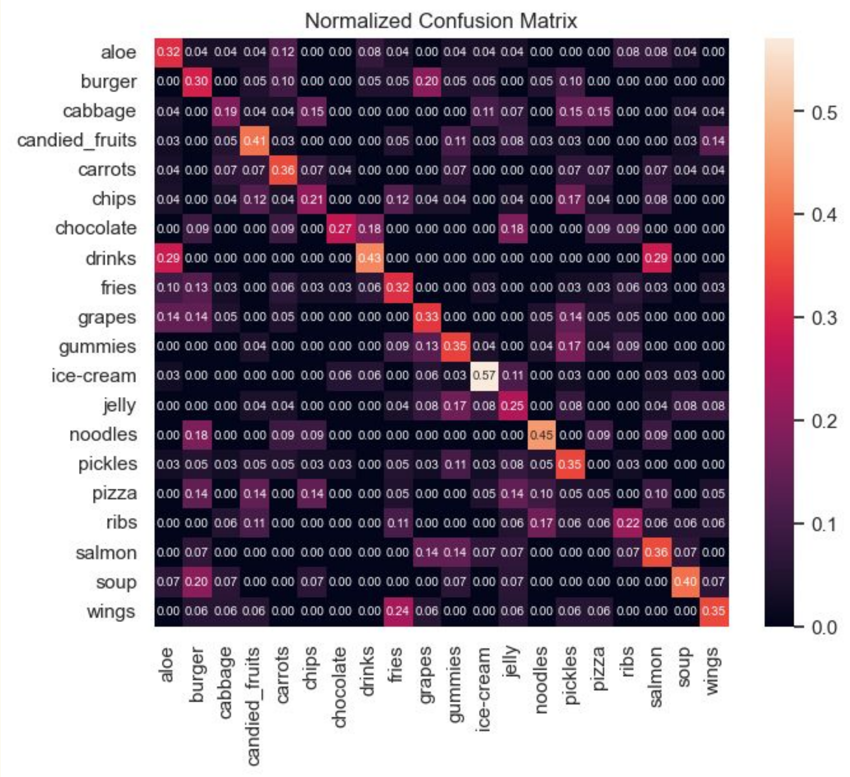
ML Approaches

Questions?

Our initial approach involved running our data on built-in scikit-learn models.

	Classifier	Accuracy Score
0	SVC	66.14%
1	DecisionTreeClassifier	35.87%
2	RandomForestClassifier	56.95%

Bad Results!!



A Convolutional Neural Network (CNN) model was constructed, as it's more suited to handle and extract features from visual data.

Key Features:

- Initial Learning Rate = 0.01
- AdamOptimizer
  - 1 Cycle Policy
- Dropout of 0.5 every other layer.
- ReLu activation function
- Data split of [70, 20, 10] throughout our tests

```
Model summary :
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 64, 258]             320
              ReLU-2          [-1, 32, 64, 258]               0
       BatchNorm2d-3          [-1, 32, 64, 258]              64
         MaxPool2d-4          [-1, 32, 32, 129]               0
            Conv2d-5          [-1, 64, 32, 129]          18,496
              ReLU-6          [-1, 64, 32, 129]               0
       BatchNorm2d-7          [-1, 64, 32, 129]             128
         MaxPool2d-8           [-1, 64, 16, 64]               0
           Dropout-9           [-1, 64, 16, 64]               0
           Conv2d-10          [-1, 128, 16, 64]          73,856
             ReLU-11          [-1, 128, 16, 64]               0
      BatchNorm2d-12          [-1, 128, 16, 64]             256
        MaxPool2d-13           [-1, 128, 8, 32]               0
           Conv2d-14           [-1, 256, 8, 32]         295,168
             ReLU-15           [-1, 256, 8, 32]               0
      BatchNorm2d-16           [-1, 256, 8, 32]             512
AdaptiveAvgPool2d-17            [-1, 256, 1, 1]               0
          Dropout-18            [-1, 256, 1, 1]               0
          Flatten-19                  [-1, 256]               0
           Linear-20                  [-1, 128]          32,896
             ReLU-21                  [-1, 128]               0
...
Params size (MB): 1.62
Estimated Total Size (MB): 26.59
```

For our model we wanted to experiment with different training tasks to see how it would affect its accuracy.

Task 1: 20-way Classification
Comparing 20 classes at a time to determine which class a given sample belongs to.

- **Uniform Holdout:** Clips from different videos are split uniformly across the test, train and validation data

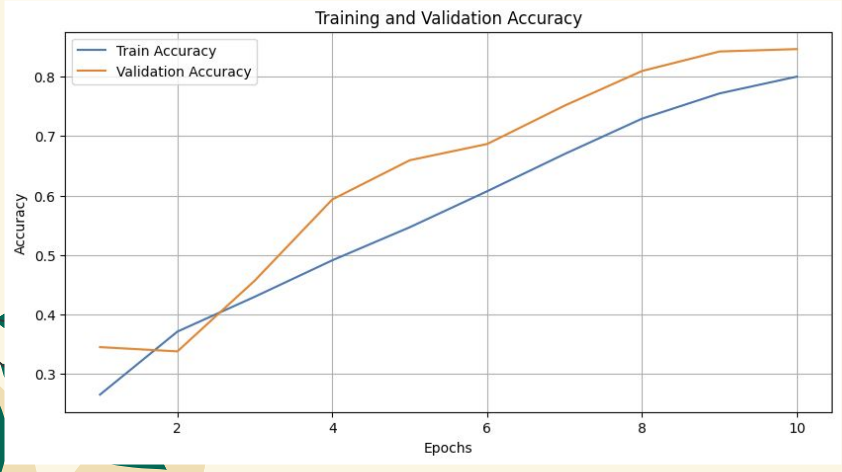
- **Groups Holdout:** Data was split by video group, avoiding clips from the same video being shared between test, train and validation data.
  - To see how the model would classify to foreign data we did groups holdout evaluation
  -

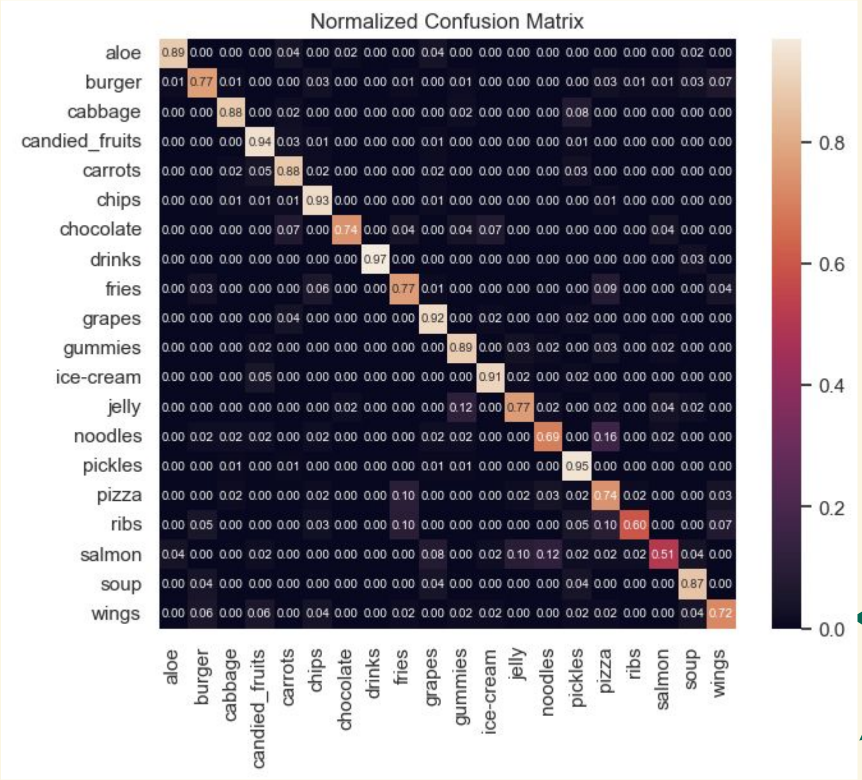
Task 2: Pairwise Classification
Comparing two classes at a time to determine which class a given sample belongs to

Uniform Holdout: Since the data in split uniformly, the model may have picked up on patterns/similarities from clips that share videos.

Test Accuracy: 0.8285

Final Results:
Train Accuracy: 0.8002
Val Accuracy: 0.8465

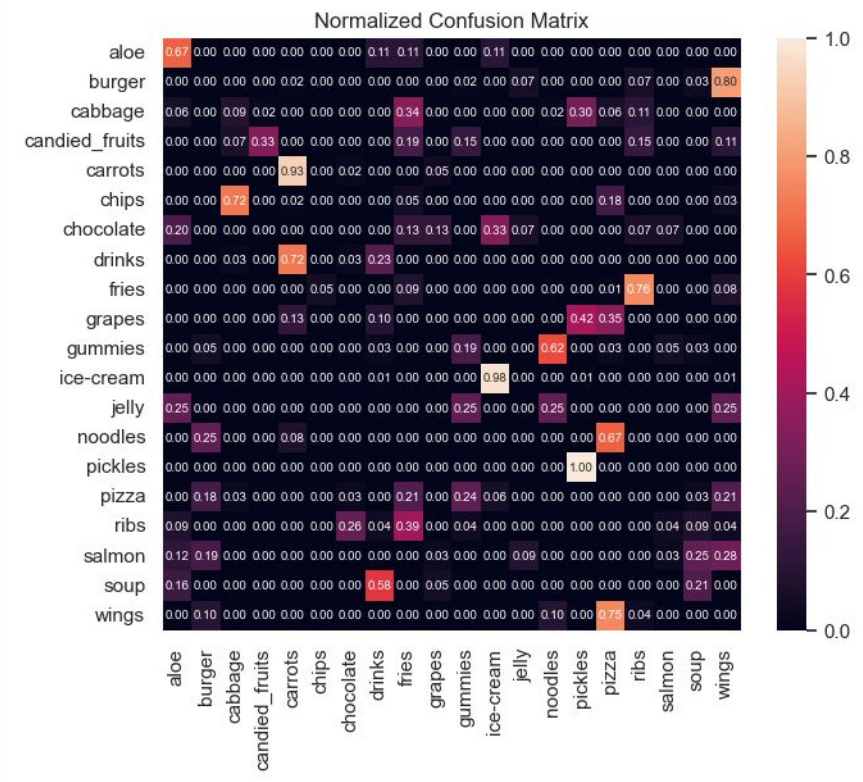




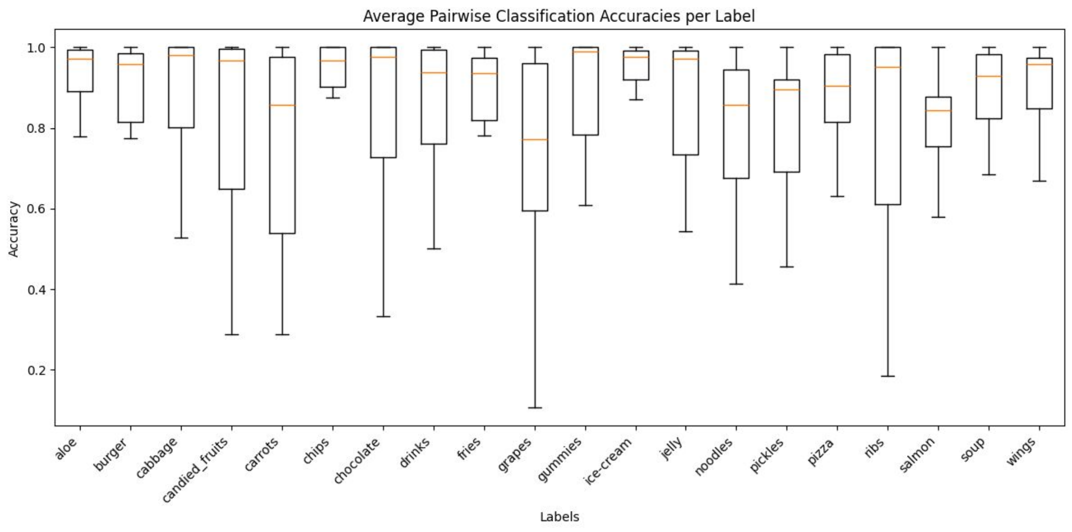
Grouped Holdout: By testing the model on a set of clips it hasn't seen before it gives us an idea of how it would run on real world input.

Test Accuracy: 0.2871

Final Results:
Train Accuracy: 0.8260
Val Accuracy: 0.2759



Pairwise Classification: Boxplot shows the average pairwise classification accuracies for the different food labels



Key Insights:

- Distinct: Aloe, Burger, ice-cream
- Confused: Grapes, Ribs
- 50-50: Candied fruit, carrots

Cool Insights:

- Drinks vs Soup = 21%
- Ribs vs [Pizza, Wings, Burger] = ~10%

Problem Description

Data Description

ML Approaches

Next Steps

Next Steps:

- Expand number of categories to include broader range of food.
- Look into food attributes that may classify broader groups (Crunchy, Chewy, Snappy, etc).
- Refine CNN Model architecture

Questions???