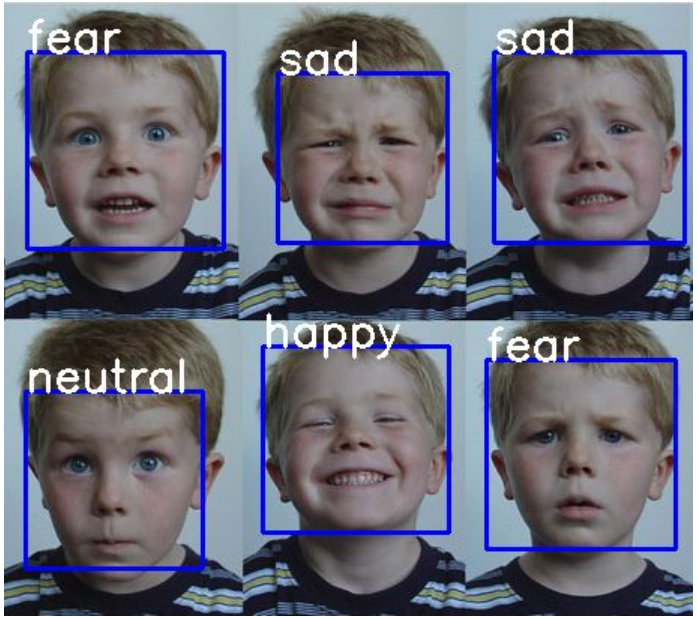
Facial Expression Recognition

Henry Kam, Thomas Kong

Facial Expression Recognition

- Faces are valuable for interpreting emotion
- Allows machines to better understand:
  - Social Cues
  - Human Emotions
  - Human Intentions
- Applied in many fields



Our Dataset

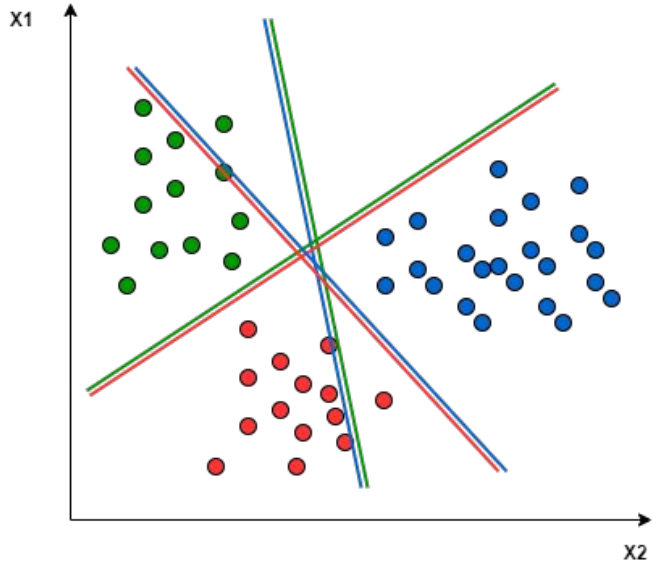
- Kaggle: FER 2013
- 48x48 Grayscale Images
- Seven categories
- Training Set: 28,000 examples
- Test Set: 3,500

Objective

- Multiclass classification
- Correctly identify emotion given facial image

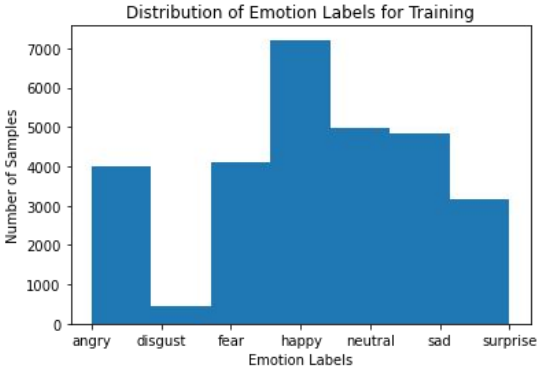
Objective

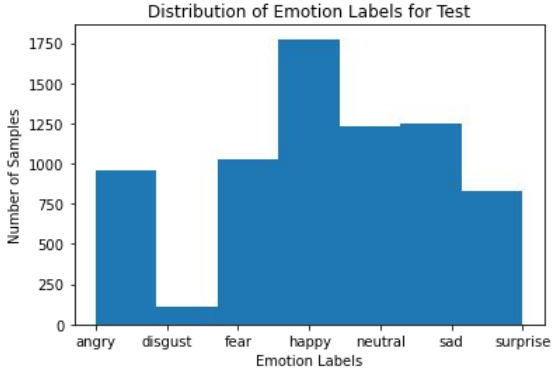
- Multiclass classification
- Given facial image -> Correctly identify emotion



Data Preparation

- Images already preprocessed
- Dummy Variable Encoding
- One vs Rest classification
- Distribution of Training Set ≈ Distribution of Test Set





Models

- Logistic Regression
- Support Vector Machines (SVM)
- Neural Networks

Logistic Regression

- Why use Logistic Regression?
  - Popular algorithm in machine learning for classification problems
  - Commonly used for binary classification, can used for multi-classification with “one vs rest” approach
    - Train multiple binary logistic regression models (each model trained to distinguish one class from rest of the other classes)
    - Each trained models predict probability and highest probability will be final prediction
    - For this project: 7 different classes for 7 different emotions.
  - Logistic regression serves as the base model for our project.
  - Available in sklearn.linear_model library

Logistic Regression

- Feature Transformations:
  - None
  - Squared
  - Cubed
- Regularization:
  - L1 (Lasso)
  - L2 (Ridge)
- Hyperparameter C-value:
  - [.1. 1, 10]

Cost	Regularization	Train Accuracy	Test Accuracy
.1	L1	0.3972	0.3799
1	L1	0.4736	0.3711
10	L1	0.5008	0.3451
.1	L2	0.4537	**0.3834**
1	L2	0.4892	0.3571
10	L2	0.5014	0.3422

No Feature Transformation Dataset

Logistic Regression

- Feature Transformations:
  - None
  - Squared
  - Cubed
- Regularization:
  - L1 (Lasso)
  - L2 (Ridge)
- Hyperparameter C-value:
  - [.1. 1, 10]

Cost	Regularization	Train Accuracy	Test Accuracy
.1	L1	0.3940	0.3651
1	L1	0.4688	0.3564
10	L1	0.4999	0.3390
.1	L2	0.4511	**0.3685**
1	L2	0.4858	0.3539
10	L2	0.4985	0.3394

Squared Feature Transformation Dataset

Logistic Regression

- Feature Transformations:
  - None
  - Squared
  - Cubed
- Regularization:
  - L1 (Lasso)
  - L2 (Ridge)
- Hyperparameter C-value:
  - [.1. 1, 10]

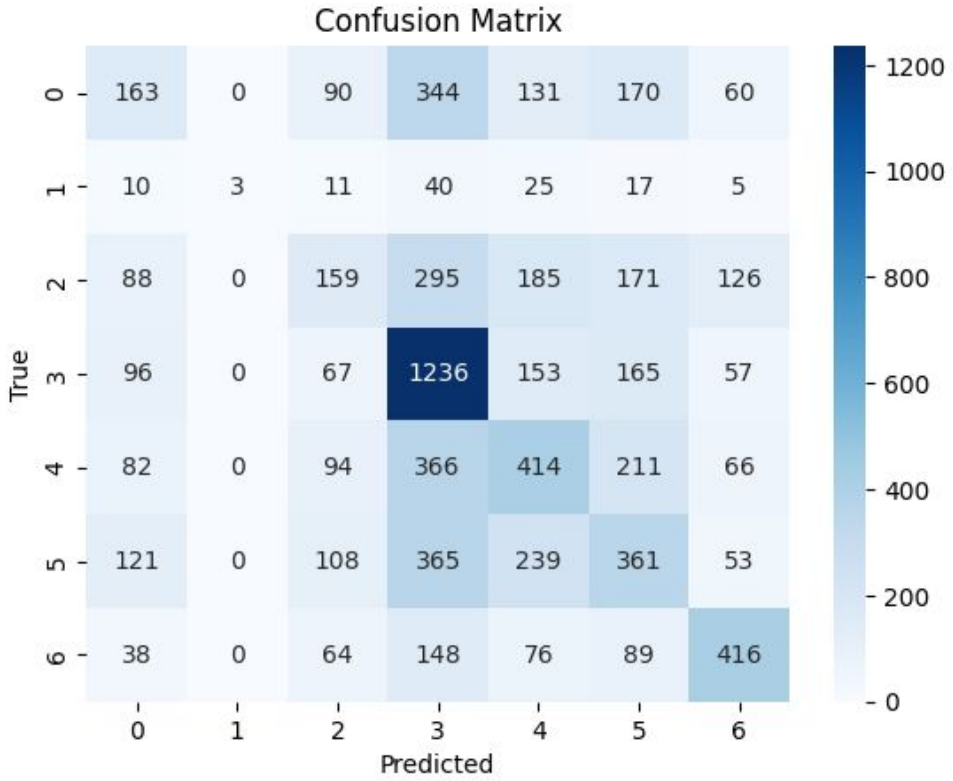
Cost	Regularization	Train Accuracy	Test Accuracy
.1	L1	0.3872	0.3502
1	L1	0.4665	0.3454
10	L1	0.4952	0.3302
.1	L2	0.4478	**0.3544**
1	L2	0.4832	0.3415
10	L2	0.4952	0.3293

Cubed Feature Transformation Dataset

Logistic Regression

- Best Feature Transformations

  Accuracies:
  - None = 38.34%
  - Squared = 36.85%
  - Cubed = 35.44%
- Best Regularization:
  - L2 (Ridge)
- Hyperparameter C-value:
  - C = .1



Confusion Matrix for No Feature Transformation

(L2 Regularization & C = .1)

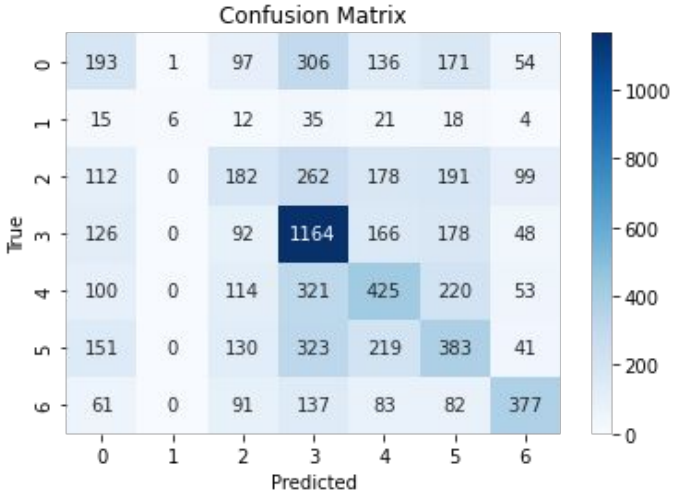
SVM

- Different Kernel functions to experiment with
- Images have high possibility of noise
- High dimensional feature spaces
- Efficient feature selection

SVM

- Kernel Function:
  - Linear
  - Polynomial
  - RBF
  - Sigmoid
- Cost Parameters:
  - 0.1
  - 1.0
  - 10.0

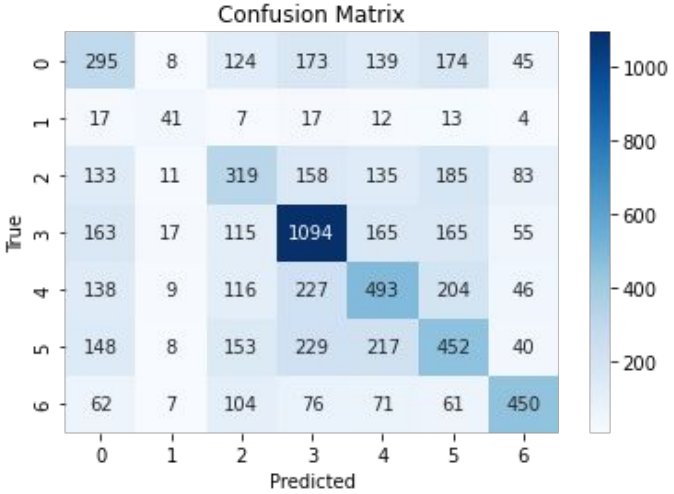
Cost	Training Accuracy	Test Accuracy
0.1	0.4886	0.3803
1.0	0.5582	0.3526
10.0	0.6106	0.3320



SVM

- Kernel Function:
  - Linear
  - Polynomial
  - RBF
  - Sigmoid
- Cost Parameters:
  - 0.1
  - 1.0
  - 10.0

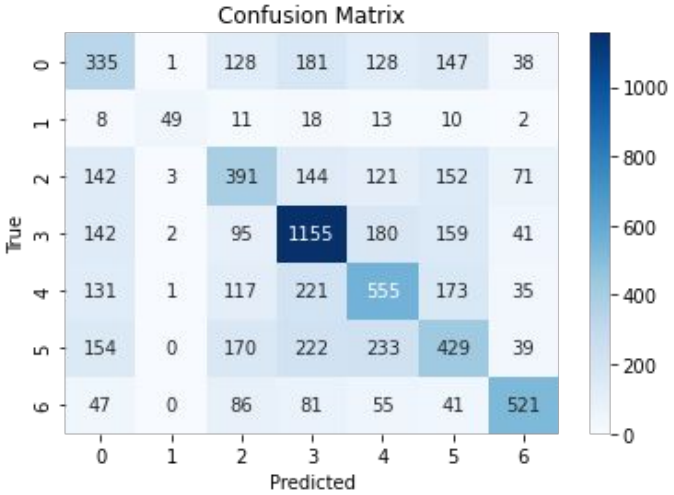
Cost	Training Accuracy	Test Accuracy
0.1	0.6382	0.4263
1.0	0.9235	0.4380
10.0	0.9929	0.4221



SVM

- Kernel Function:
  - Linear
  - Polynomial
  - RBF
  - Sigmoid
- Cost Parameters:
  - 0.1
  - 1.0
  - 10.0

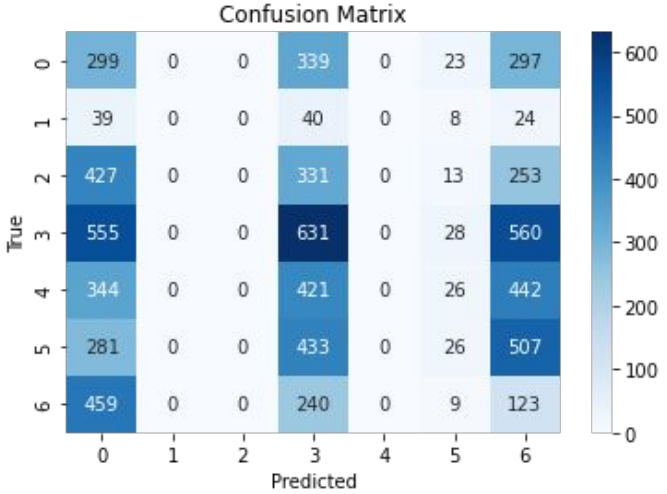
Cost	Training Accuracy	Test Accuracy
0.1	0.3811	0.3661
1.0	0.6181	0.4464
10.0	0.9770	0.4785



SVM

- Kernel Function:
  - Linear
  - Polynomial
  - RBF
  - Sigmoid
- Cost Parameters:
  - 0.1
  - 1.0
  - 10.0

Cost	Training Accuracy	Test Accuracy
0.1	0.1659	0.1503
1.0	0.1560	0.1256
10.0	0.1554	0.1344

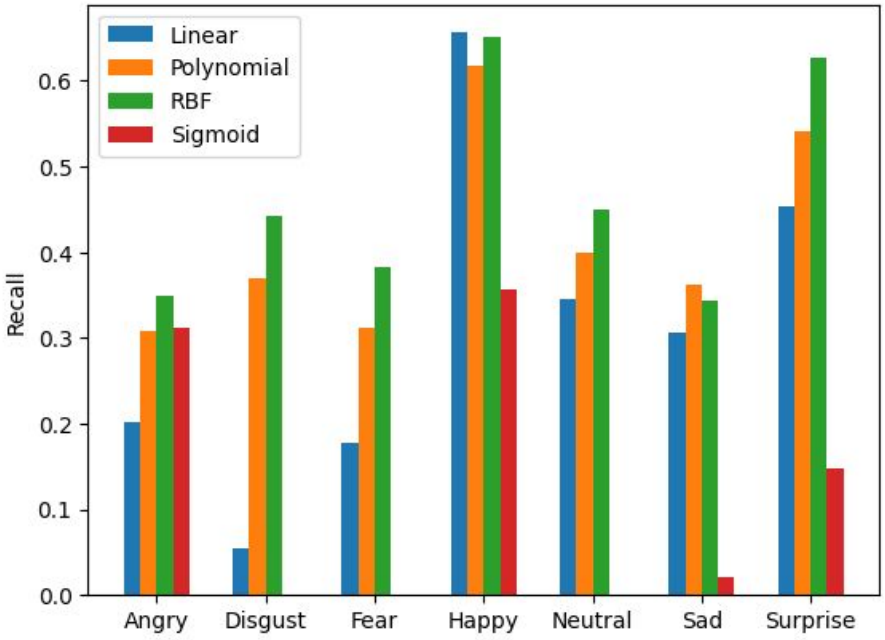


SVM

- Soft margin case: no need to test C=0
- Did not have time to test other regularization values
  - Training time was very long (N=28,000)

SVM

- RBF (C=10.0) had highest accuracy
- Happiness has highest TPR
- Sigmoid could not predict
  - Disgust
  - Fear
  - Neutral



Neural Networks

- Last model implemented was Neural Networks, specifically Convolutional Neural Networks (CNN)
  - CNN is type of Neural Network that uses convolutional layers for feature extraction
  - CNN is popular for image classification
- Basic architecture of our CNN
  - Different convolutional layers
  - ReLU activation function for convolutional layers
  - Last layer is a dense layer with number of neurons equal to the number of classes
    - Uses softmax activation function for multiple classifications

Convolutional Neural Network (CNN)

- Dropout Regularization
  - CNN doesn't often use L1 or L2 regularization
    - Because CNN already has built-in structure that acts as a form of implicit regularization.
  - Popular regularization technique is dropout
    - Randomly drops proportion of the neurons during each training epoch (sets them to 0)

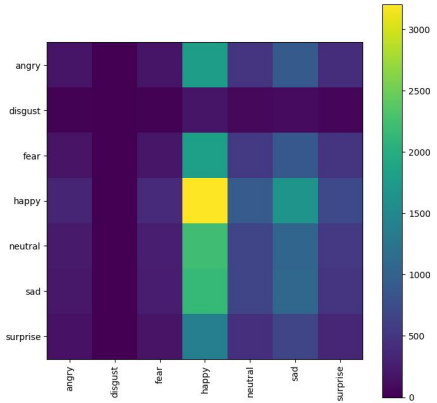
- Batch Normalization
  - Batch Normalization applied between the layers of a Neural Network instead of in the raw data
  - Normalizes the input to each layer of the network
    - Mean activation close to 0
    - Standard deviation close to 1
  - tf.keras.layers.BatchNormalization()

Convolutional Neural Network (CNN)

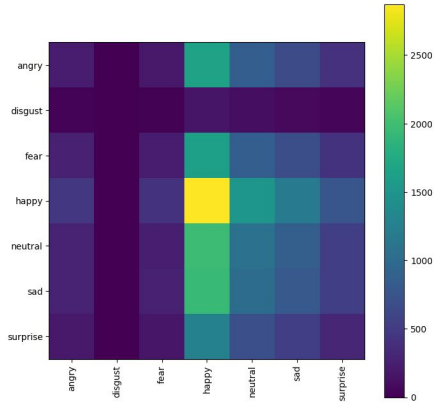
- Feature Transformation
  - Experimented with 1 to 3 Convolutional Layers

Layers	Training Accuracy	Testing Accuracy
1	0.3947	0.4267
2	0.4203	0.4521
3	0.4808	**0.5068**

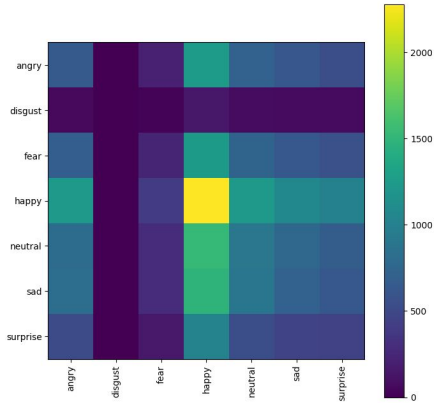
(1)



(2)



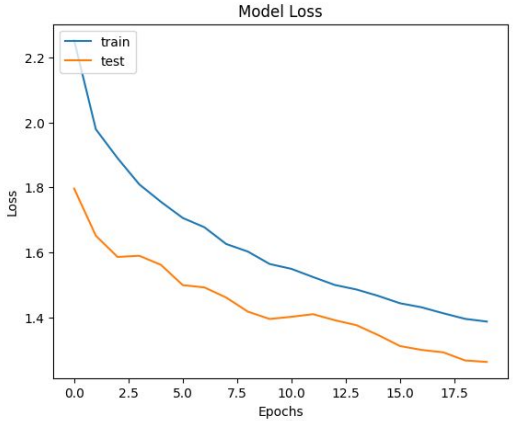
(3)

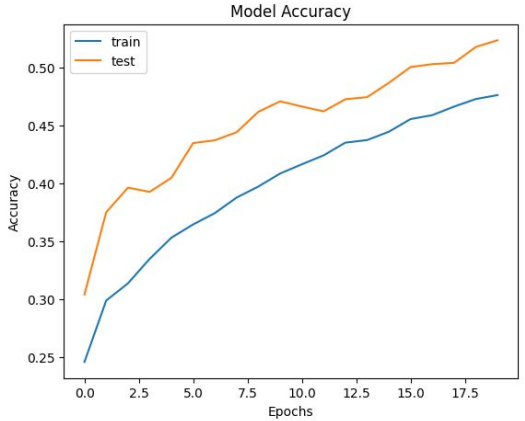


Convolutional Neural Network (CNN)

- Activation Function
  - ReLU
  - TanH
  - Sigmoid

Model	Training Accuracy	Testing Accuracy
Basic CNN	0.4808	0.5068
CNN w/ Regularization	0.4391	0.4521
CNN w/ Regularization & Batch Normalization	0.5123	**0.5235**

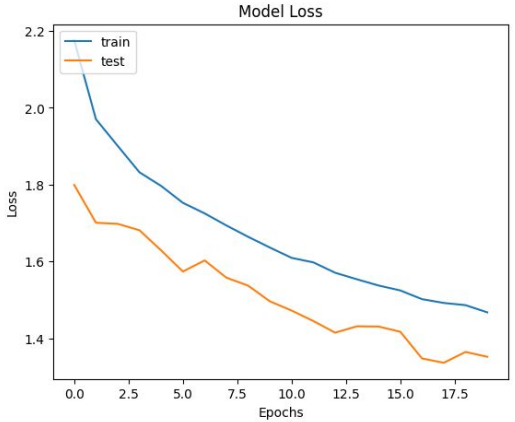


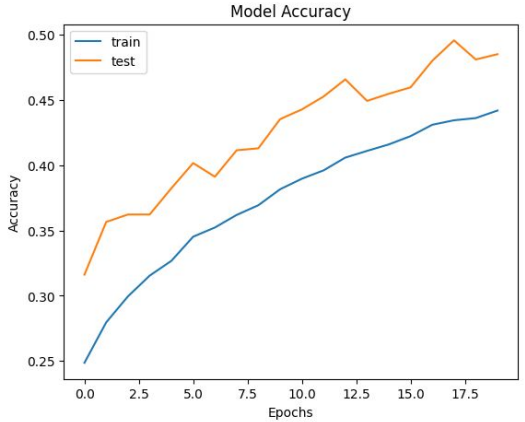


Convolutional Neural Network (CNN)

- Activation Function
  - ReLU
  - TanH
  - Sigmoid

Model	Training Accuracy	Testing Accuracy
Basic CNN	0.4518	0.4675
CNN w/ Regularization	0.41.71	0.4391
CNN w/ Regularization & Batch Normalization	0.4610	**0.4847**

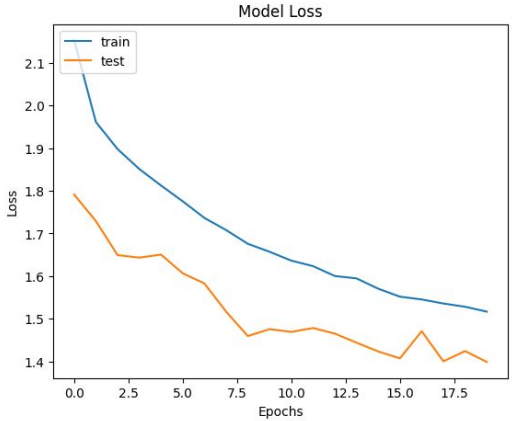


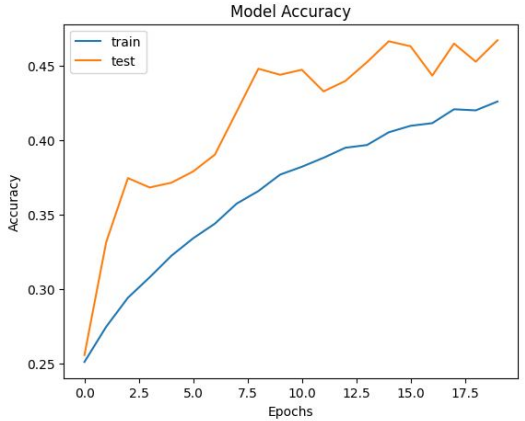


Convolutional Neural Network (CNN)

- Activation Function
  - ReLU
  - TanH
  - Sigmoid

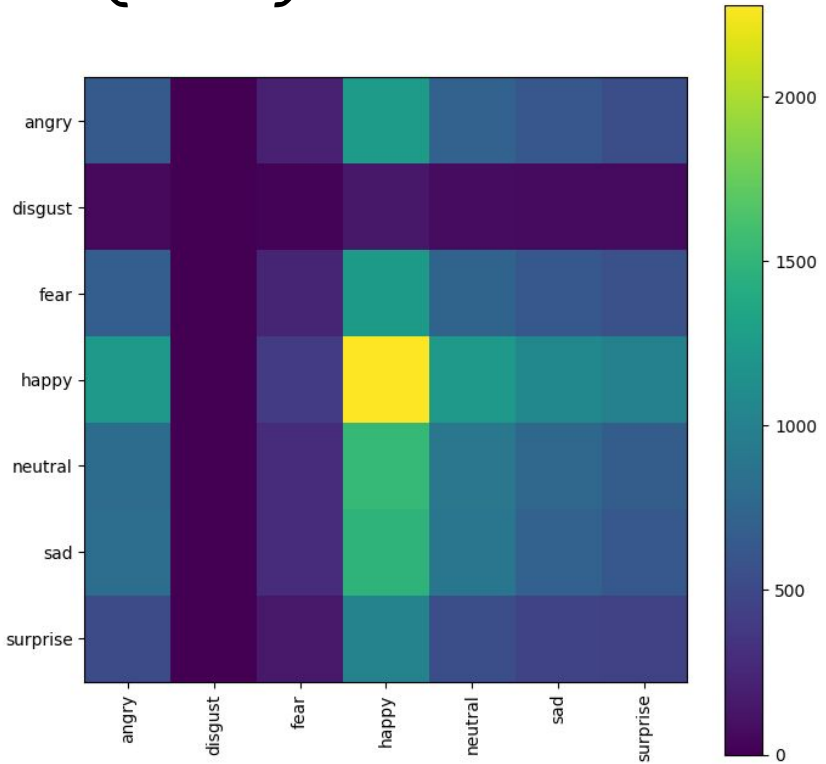
Model	Training Accuracy	Testing Accuracy
Basic CNN	0.3160	0.3383
CNN w/ Regularization	0.2513	0.2471
CNN w/ Regularization & Batch Normalization	0.4588	**0.4673**





Convolutional Neural Network (CNN)

- Best CNN Accuracy = 53.35%
  - With ReLU Activation
  - With Dropout Regularization
  - With Batch Normalization
- Important Notes:
  - Two methods to improve testing accuracy
    - Add more layers/filters (Not guarantee better results)
    - Increase number of epochs complexity when running each CNN model.
  - Both methods will increase time complexity



Confusion Matrix for Best CNN Model

Top 4 Model Results

Model	Testing Accuracy
CNN using ReLU, dropout regularization and basic normalization	53.35%
Basic CNN using ReLU	50.68%
CNN using tanh, dropout regularization and basic normalization	48.47%
SVM using RBF kernel function with cost hyperparameter C = 10.0	47.85%

Conclusion - Logistic Regression

- Highest Accuracy was 38.34%
  - With No Feature Transformation, L2 regularization & C = .1
- Lowest Accuracy was 32.93%
  - With Cubed Feature Transformation, L2 regularization & C = 10
- Future Improvements:
  - Experiment with different cost parameters

Conclusion - SVM

- Highest Accuracy was RBF (C=10.0)
  - Increasing C gave better accuracy -> overfitting
- Lowest Accuracy was sigmoid (C=10.0)
  - Increasing C gave worse accuracy -> underfitting
- Linear -> underfitting
- Polynomial -> overfitting
- Future Improvements:
  - Experiment with different cost parameters

Conclusion - Neural Networks

- Highest Accuracy was 53.35%
  - With ReLU Activation, dropout regularization & batch normalization
- Lowest Accuracy was 24.71%
  - With sigmoid Activation & dropout regularization
- Future Improvements:
  - Experiment with adding more layers
  - Experiment with more training epochs
  - Experiment with mixing activation functions

Conclusion

- Facial expression was more complex that what we tested, and we have way more to explore
- We are happy with 53%
- We are happy to apply our knowledge from class to something so important
- Future Improvements:
    - Experiment with other cost parameters for Logistic Regression & SVM
    - Experiment with adding more layers, training epochs to CNN
    - Experiment with mixing activation functions for CNN

Works Cited

- https://i0.wp.com/sefiks.com/wp-content/uploads/2018/01/kid-expressions-cover.png?resize=459%2C409&ssl=1
- https://www.baeldung.com/wp-content/uploads/sites/4/2020/10/multiclass-svm2-e1601952762246.png