Feature learning in facial expression recognition

2022 Spring DSE I2100

Applied Machine Learning and Data Mining

—

Professor: Michael Grossberg
By Chikako Olsen, Ivan Miller, Rabiul Hossain

Goals

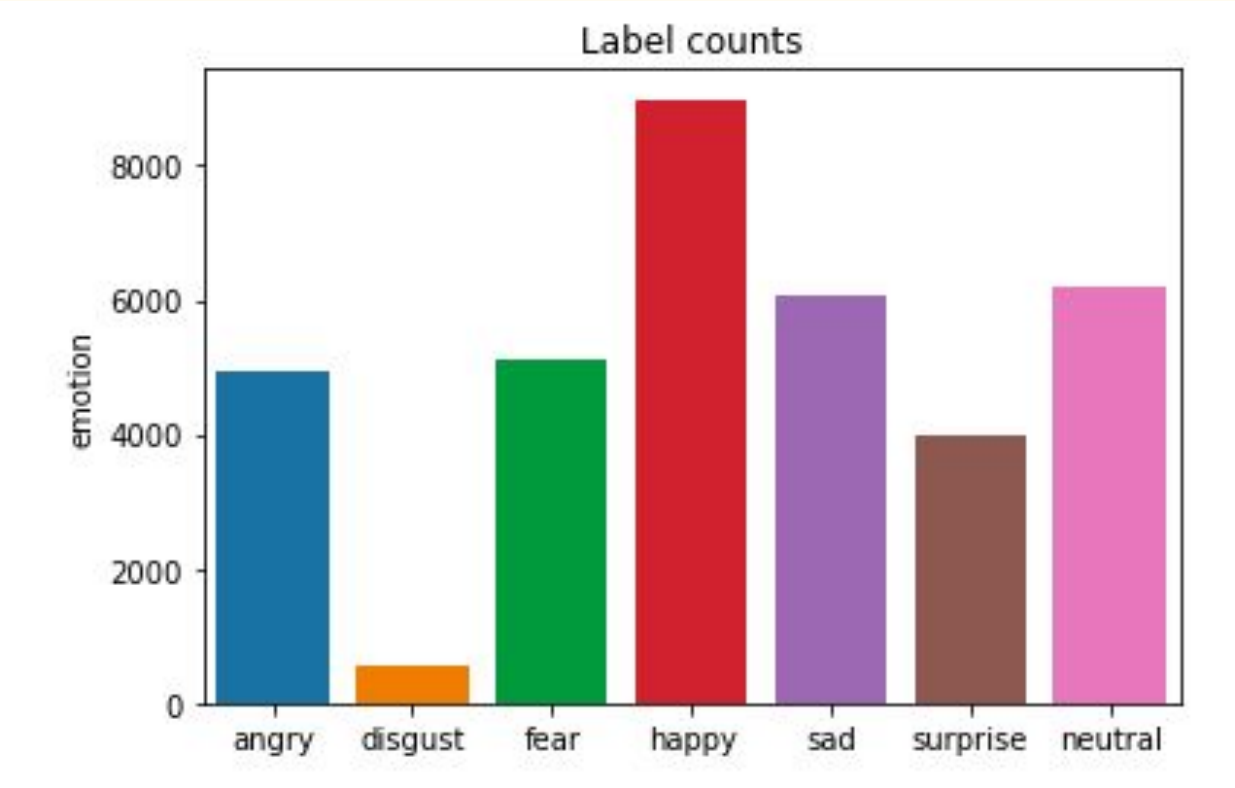
- Detect human emotion from face expression
- Find out which feature transformation works the best
- Find out which ML model gives the best accuracy
- Compare results achieved by classical ML methods with NN
- Look into activation of neurons in different layers of the NN

Dataset

The Data

- Kaggle
- 48x48 grayscale images of faces
- 36k images
- 7 classes of emotions

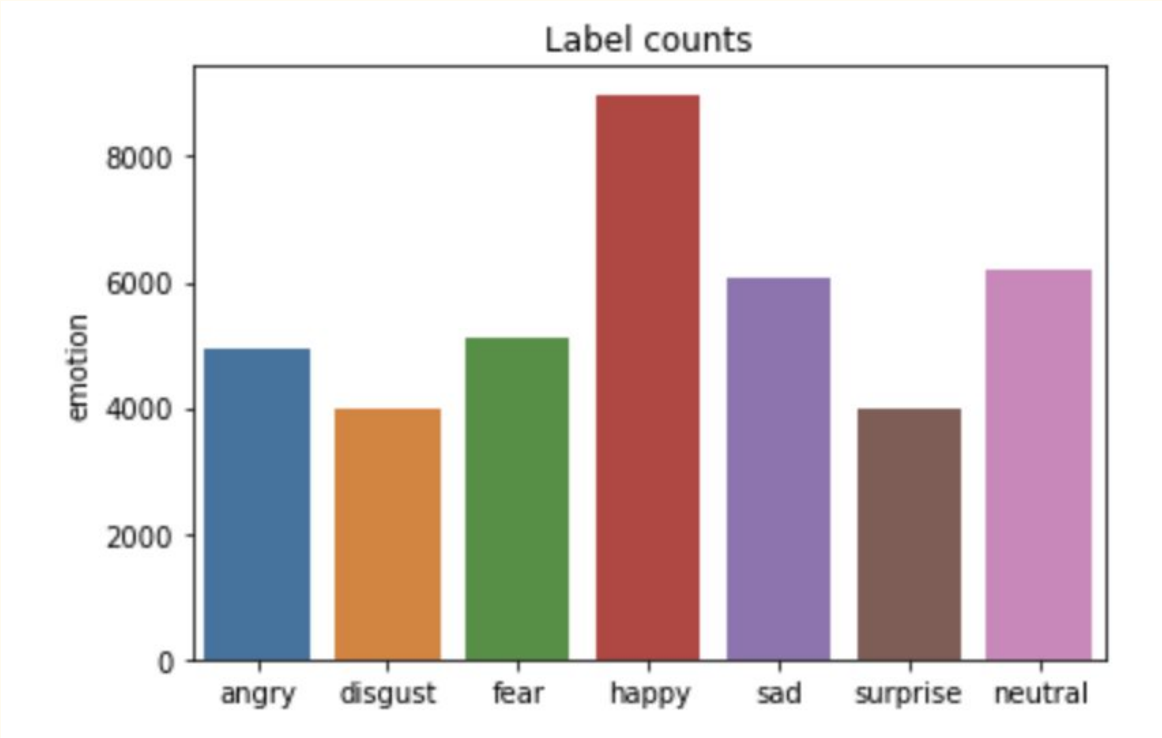
EDA


Label counts
emotion
8000
6000
4000
2000
0
angry
disgust
fear
happy
sad
surprise
neutral

Classic Machine Learning

Clean and Balanced Data

1. Deleted Black / White only images
2. Made disgust same amount as surprise label

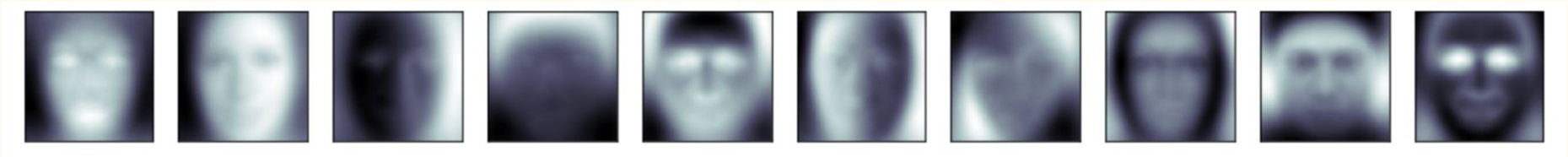


Feature Transformation

Technique we used

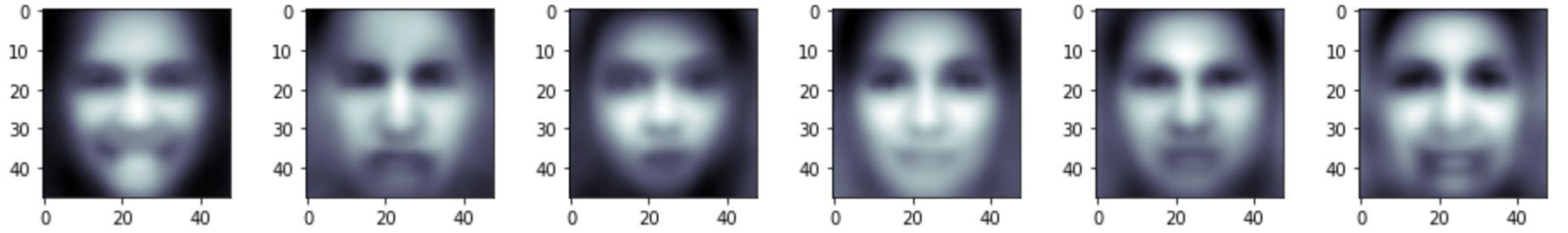
- EigenFace
- FisherFace
- Bag of Features
- HoG
- Affine transformation

EigenFace (PCA) n_component = 103 (90%)

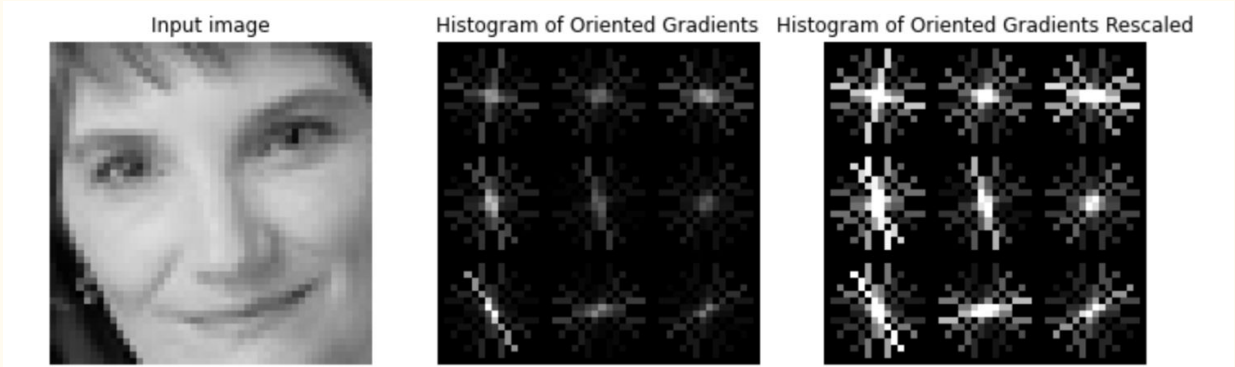


PCA Inverse

FisherFace (PCA + LDA)

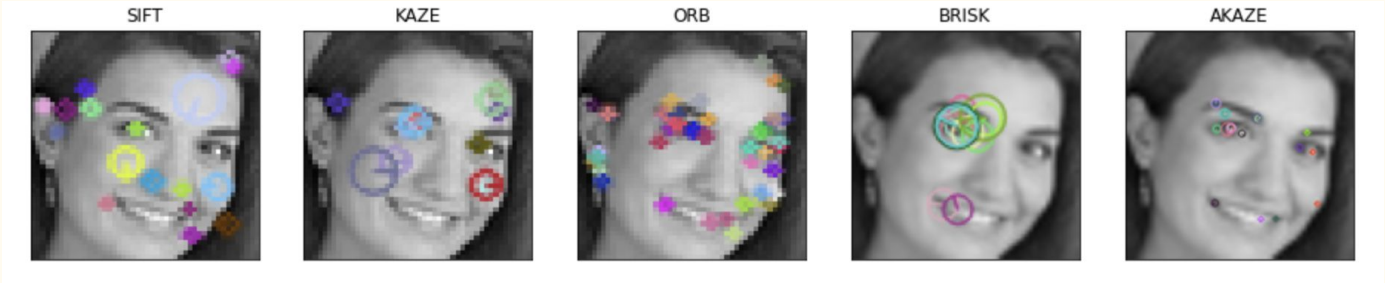


HoG (Histogram of oriented gradients)



Bag of Features (Bag of Visual Words)

1. Extract feature descriptors.

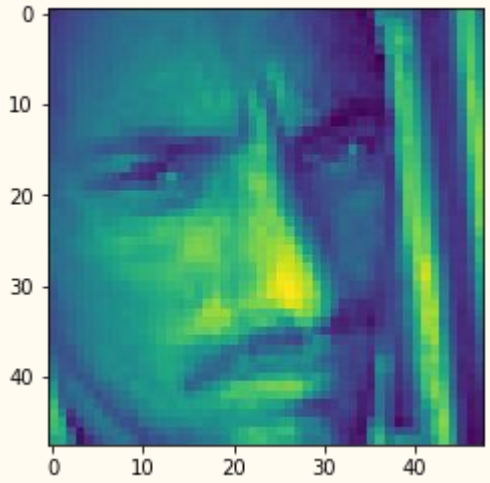


2. Apply the k-means clustering to the extracted feature descriptors to define the features (visual words).

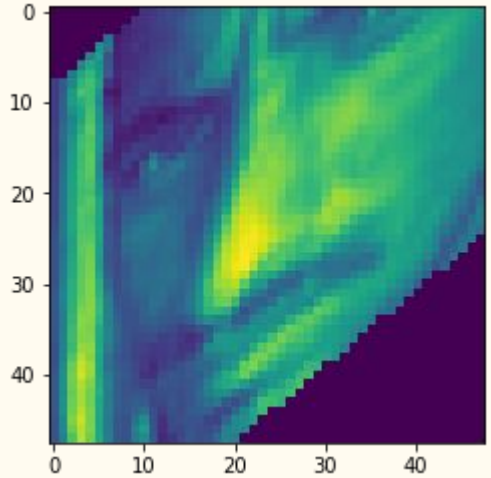
3. Use the bag of visual words to encode an image in an image set into a histogram of visual words.

Affine transformation

Original Image



Transformed Image



Machine Learning

Algorithms we used

- SVM
- KNN
- SGD
- Random Forest

Hyper parameters for comparison

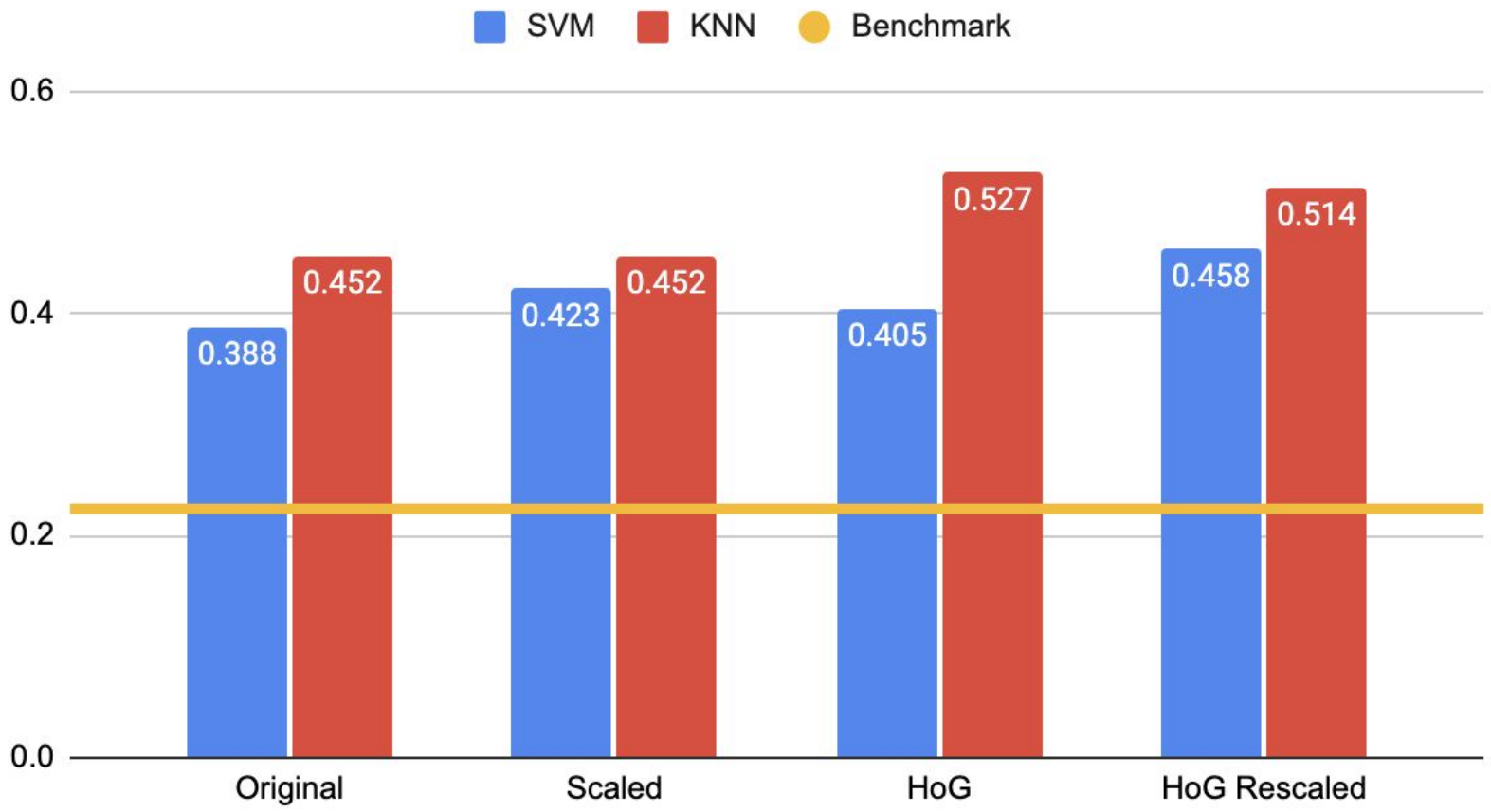
SVM parameters:

kernel = ‘rbf’, gamma = ‘auto’, C = 5, decision_function_shape=’ovo’

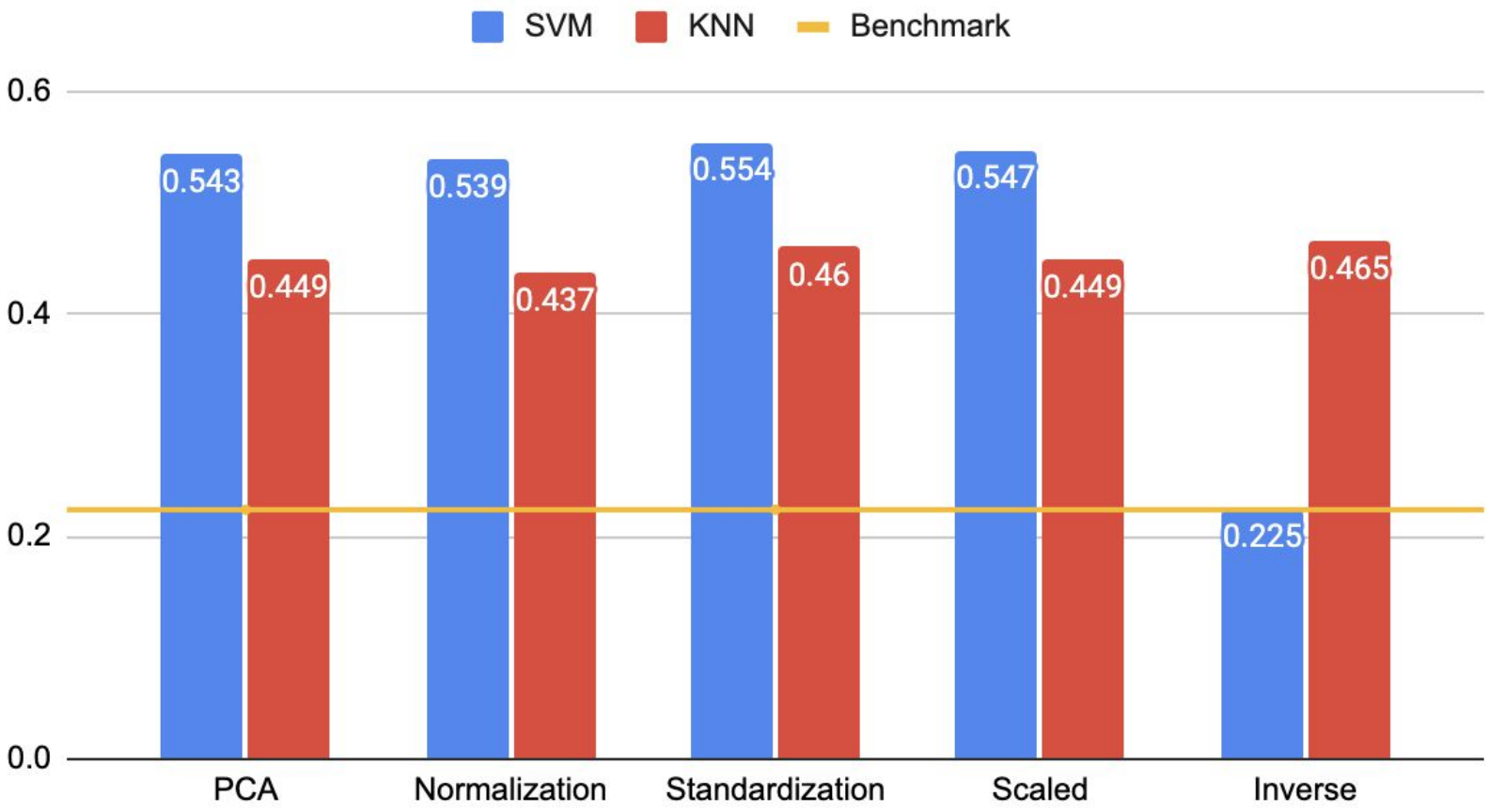
KNN parameters:

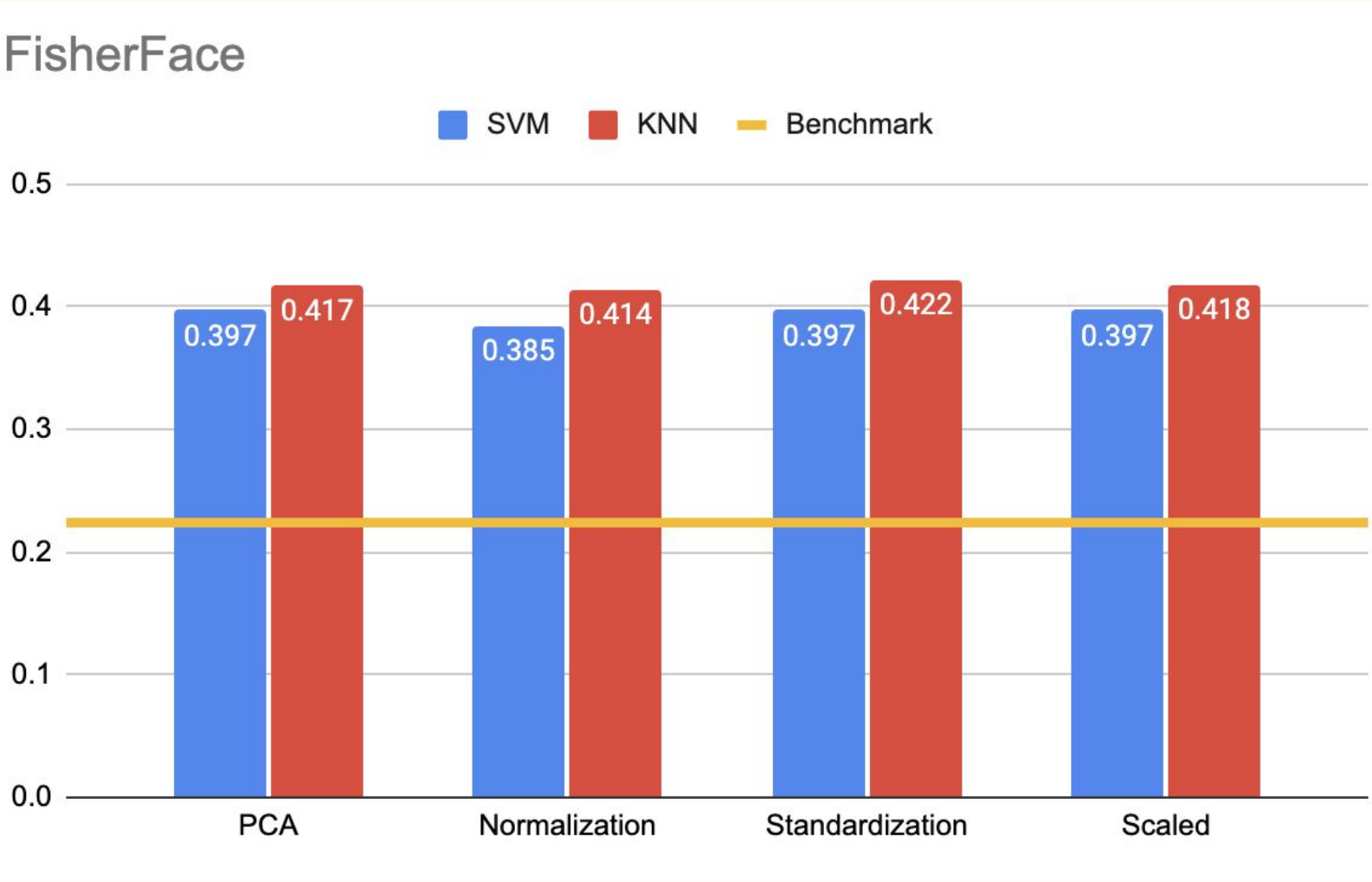
n_neighbors = 3, weights = ‘distance’, metric = ‘minkowski’

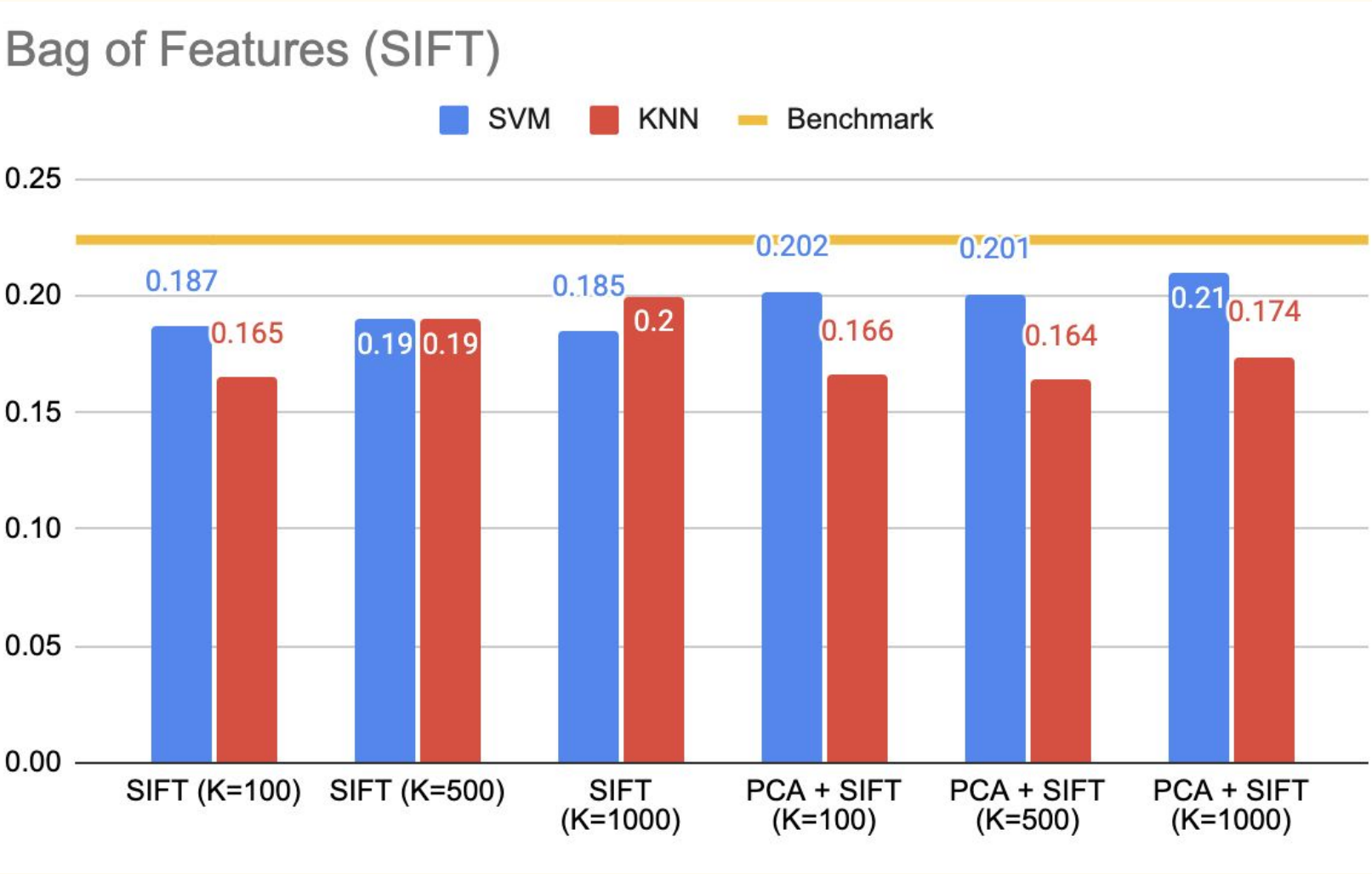
Original Data and HoG

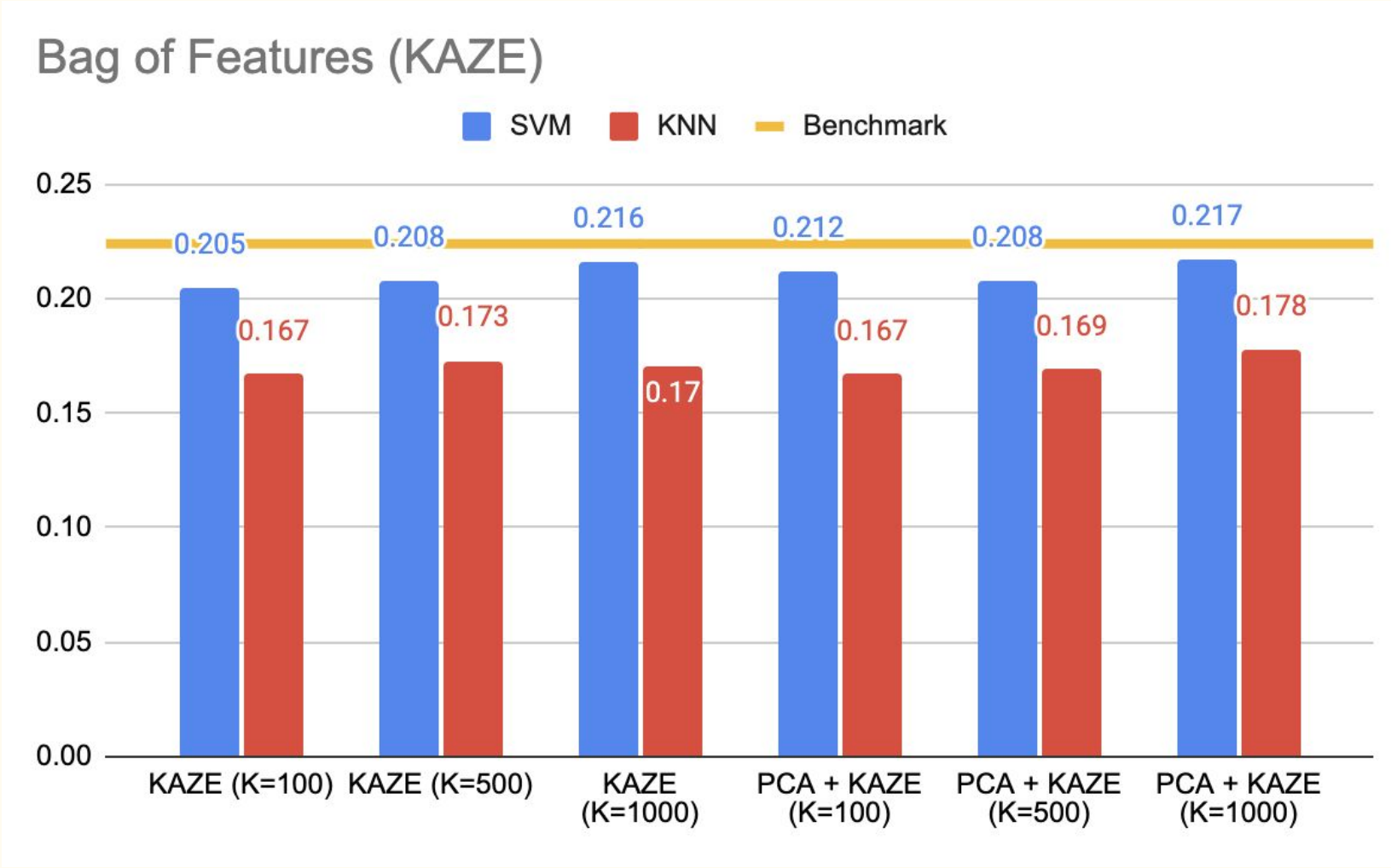


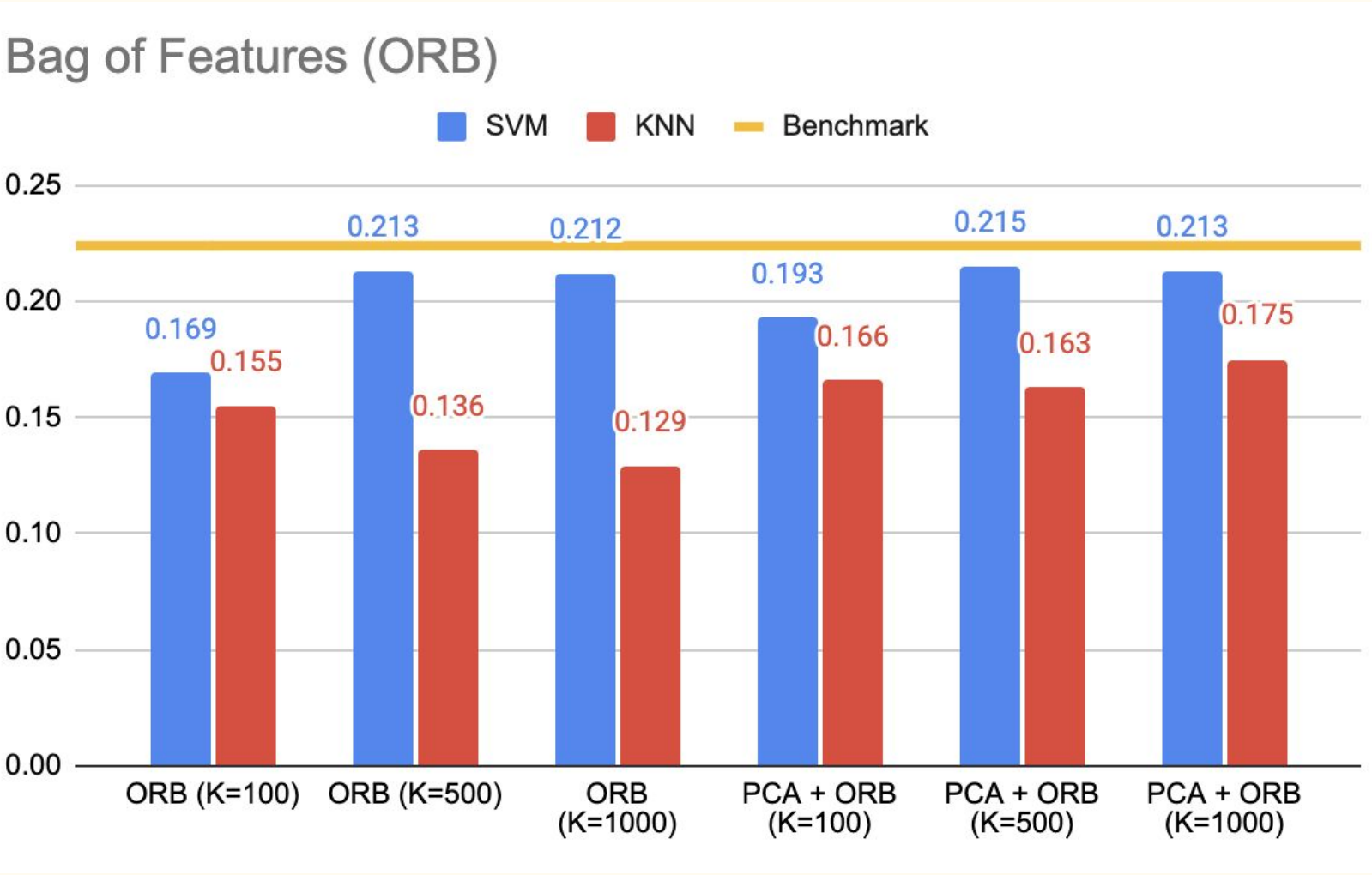
EigenFace


SVM
KNN
Benchmark
0.6
0.4
0.2
0.0
0.543
0.449
0.539
0.437
0.554
0.46
0.547
0.449
0.225
0.465
PCA
Normalization
Standardization
Scaled
Inverse

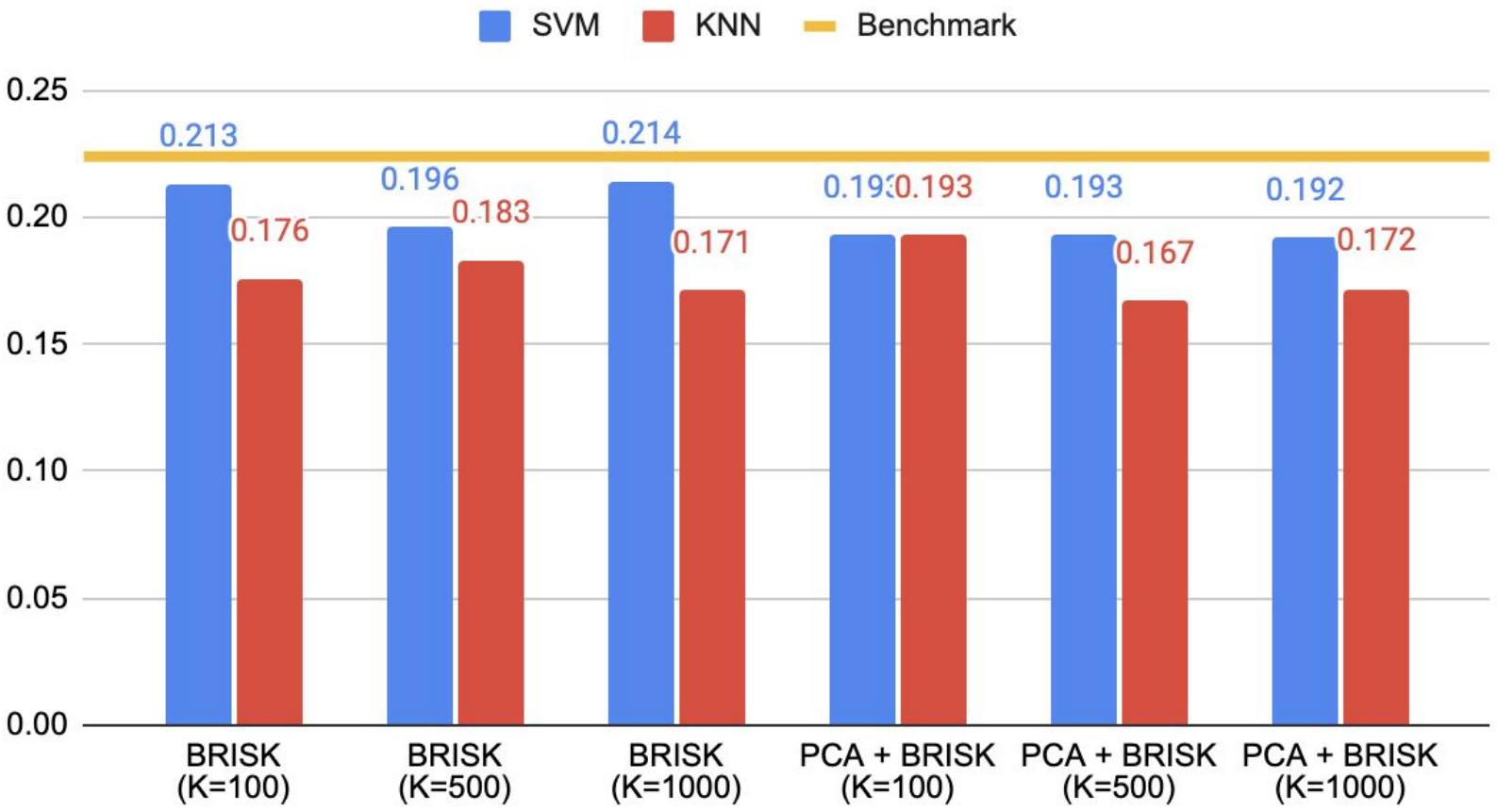

FisherFace
SVM
KNN
Benchmark
0.5
0.4
0.3
0.2
0.1
0.0
0.397
0.417
0.385
0.414
0.397
0.422
0.397
0.418
PCA
Normalization
Standardization
Scaled


Bag of Features (SIFT)
SVM
KNN
Benchmark
0.25
0.20
0.15
0.10
0.05
0.00
0.187
0.165
0.19
0.19
0.185
0.2
0.202
0.166
0.201
0.164
0.21
0.174
SIFT (K=100)
SIFT (K=500)
SIFT (K=1000)
PCA + SIFT (K=100)
PCA + SIFT (K=500)
PCA + SIFT (K=1000)

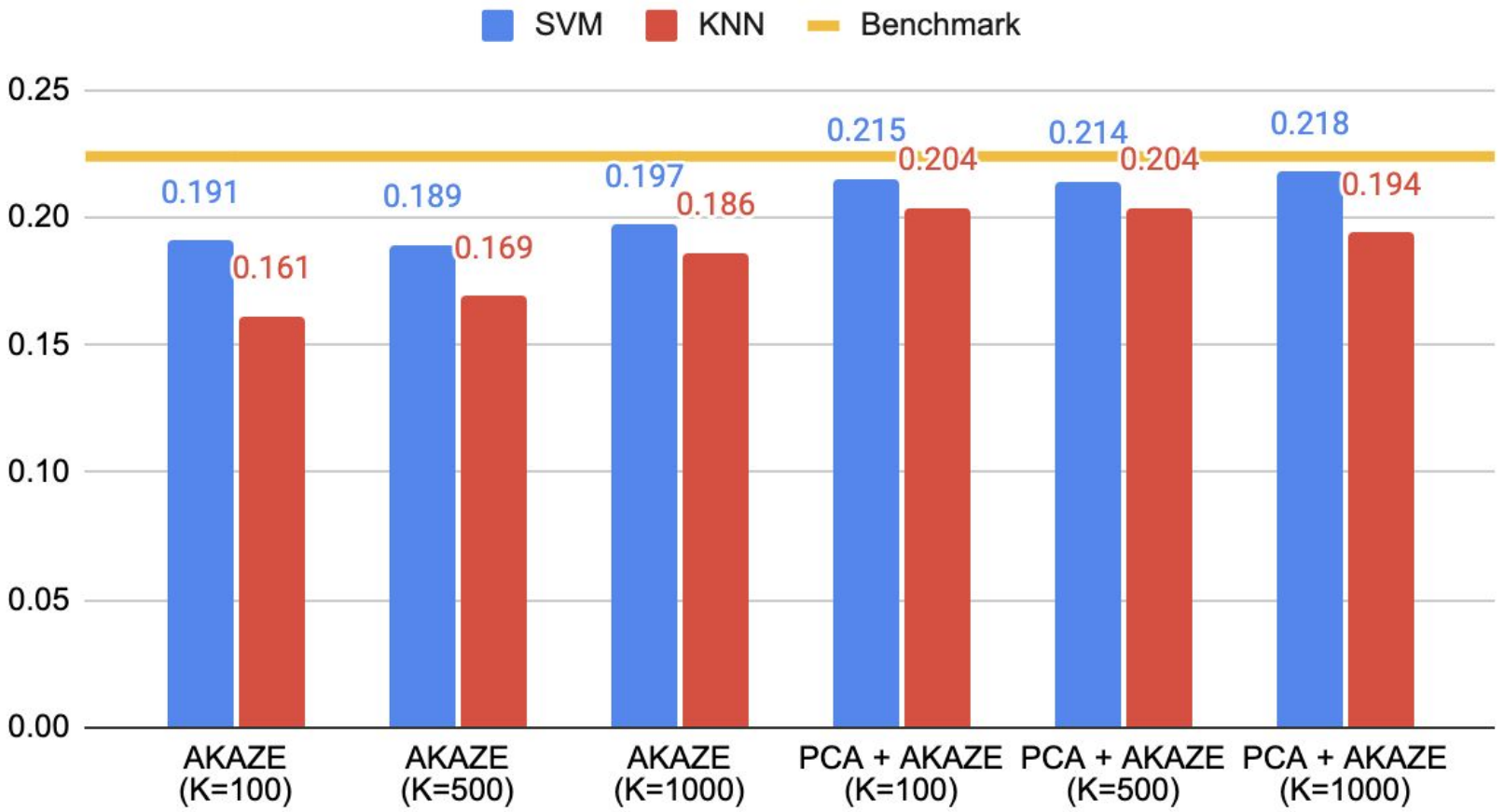

Bag of Features (KAZE)
SVM
KNN
Benchmark
0.25
0.20
0.15
0.10
0.05
0.00
0.205
0.208
0.216
0.212
0.208
0.217
0.167
0.173
0.17
0.167
0.169
0.178
KAZE (K=100)
KAZE (K=500)
KAZE (K=1000)
PCA + KAZE (K=100)
PCA + KAZE (K=500)
PCA + KAZE (K=1000)


Bag of Features (ORB)
SVM
KNN
Benchmark
0.25
0.20
0.15
0.10
0.05
0.00
0.169
0.155
0.213
0.136
0.212
0.129
0.193
0.166
0.215
0.163
0.213
0.175
ORB (K=100)
ORB (K=500)
ORB (K=1000)
PCA + ORB (K=100)
PCA + ORB (K=500)
PCA + ORB (K=1000)

Bag of Features (BRISK)

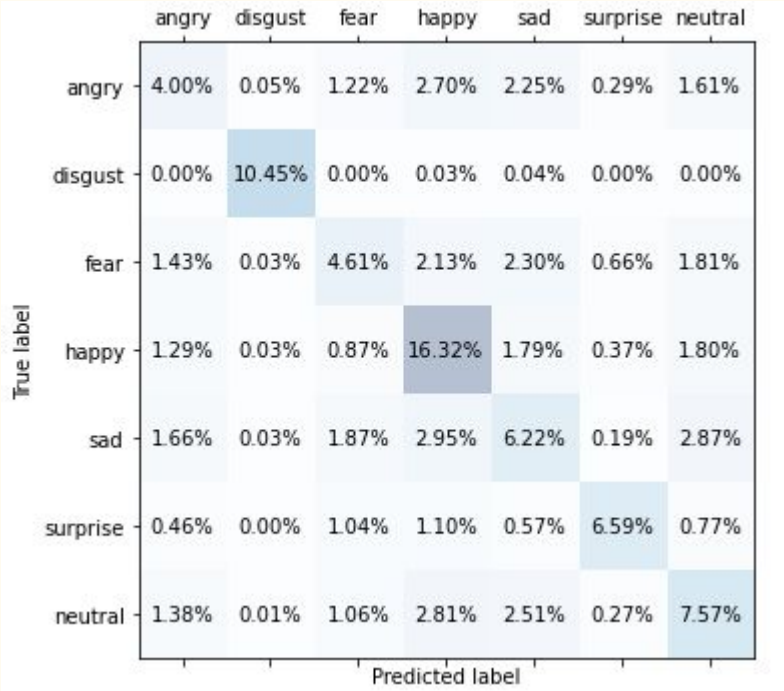


Bag of Features (AKAZE)

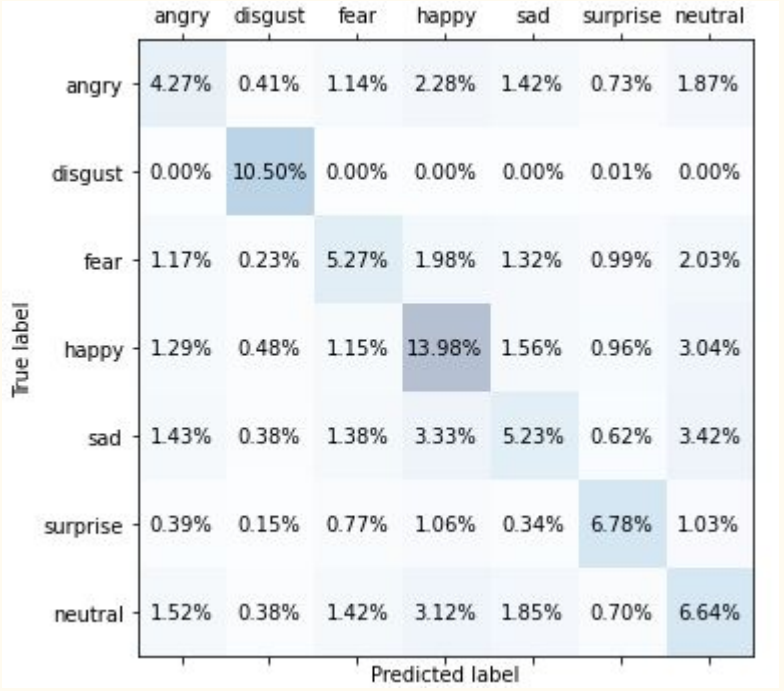


Confusion Matrix

SVM (PCA Standardization)



KNN (HoG)



Performance Metrics

SVM (PCA Standardization)

Precision: 0.577

Recall: 0.559

F1: 0.564

Accuracy: 0.558

KNN (HoG)

Precision: 0.532

Recall: 0.540

F1: 0.532

Accuracy: 0.527

Classification Report

SVM (PCA Standardization)

	precision	recall	f1-score	support
angry	0.39	0.33	0.36	957
disgust	0.99	0.99	0.99	830
fear	0.43	0.36	0.39	1024
happy	0.58	0.73	0.65	1774
sad	0.40	0.39	0.40	1247
surprise	0.79	0.63	0.70	831
neutral	0.46	0.48	0.47	1233

KNN (HoG)

	precision	recall	f1-score	support
angry	0.42	0.35	0.38	957
disgust	0.84	1.00	0.91	830
fear	0.47	0.41	0.44	1024
happy	0.54	0.62	0.58	1774
sad	0.45	0.33	0.38	1247
surprise	0.63	0.64	0.64	831
neutral	0.37	0.42	0.39	1233

Classification Report (continue)

SVM (PCA Standardization)

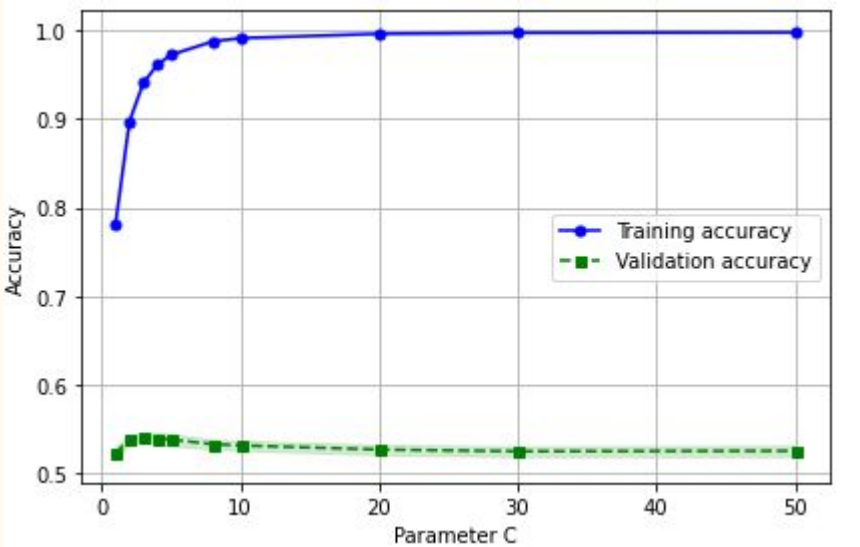
	precision	recall	f1-score	support
accuracy			0.56	7896
macro avg	0.58	0.56	0.56	7896
weighted avg	0.56	0.56	0.55	7896

KNN (HoG)

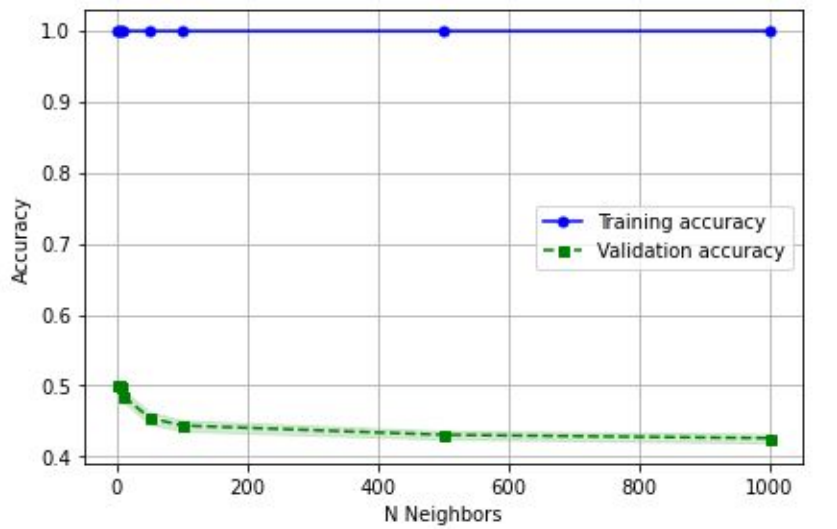
	precision	recall	f1-score	support
accuracy			0.53	7896
macro avg	0.53	0.54	0.53	7896
weighted avg	0.52	0.53	0.52	7896

Validation curve

SVM



KNN



SGD

SGD Parameters

Alpha: 0.1

Penalty: L2

max_iter: 5000

Loss: log

Results

1. Original dataset

   Accuracy: 27.1%

2. Affine transformation/Augmentation

   Accuracy: 15.55%

3. HoG Rescaled

   Accuracy: 45.29%

RandomForest

RandomForest Parameters

criterion: entropy

max_depth: 8

max_features: auto

n_estimators: 200

Results

1. Original dataset

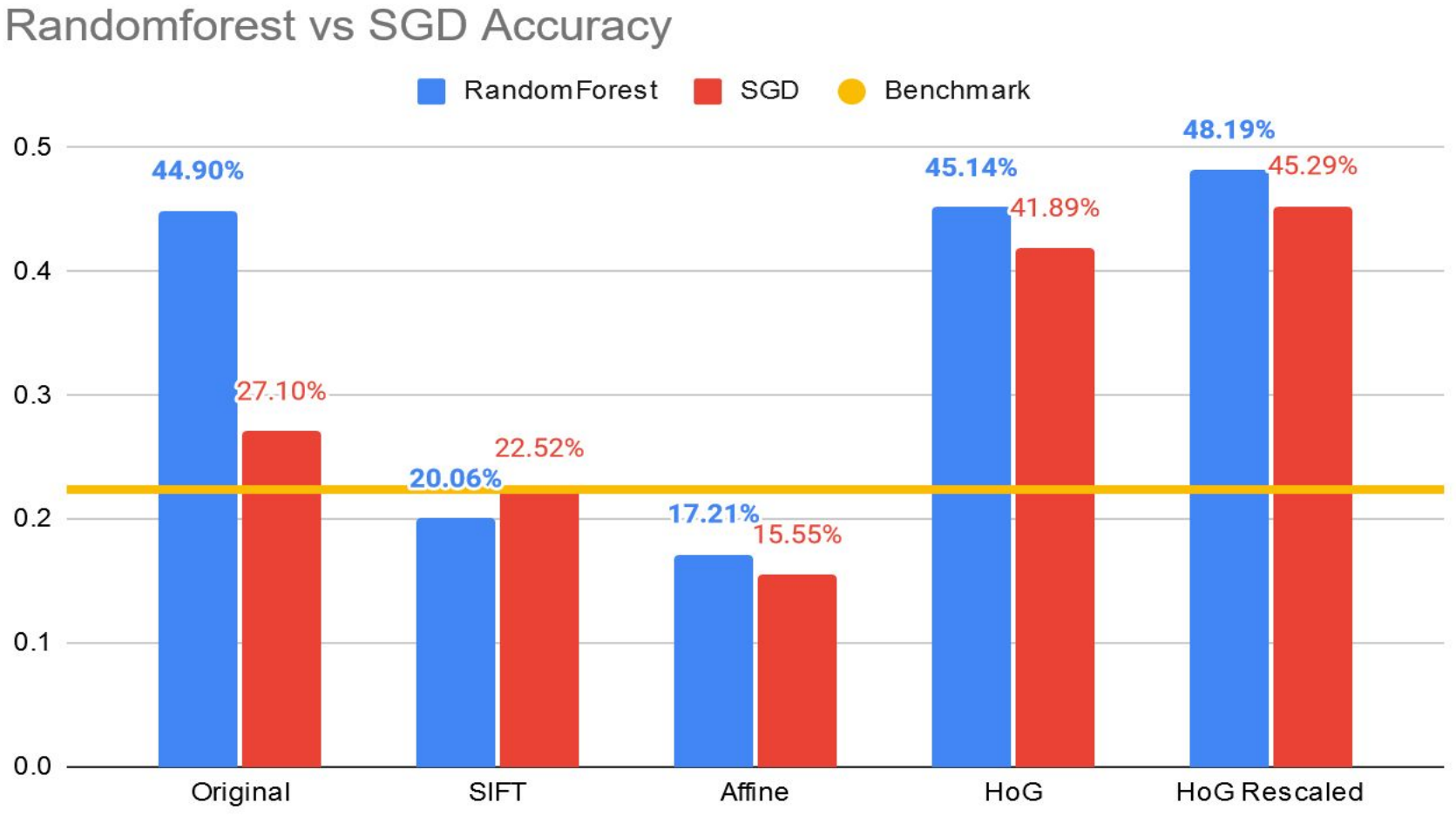
   Accuracy: 44.9%

2. Affine transformation/Augmentation

   Accuracy: 17.21%

3. HoG Rescaled

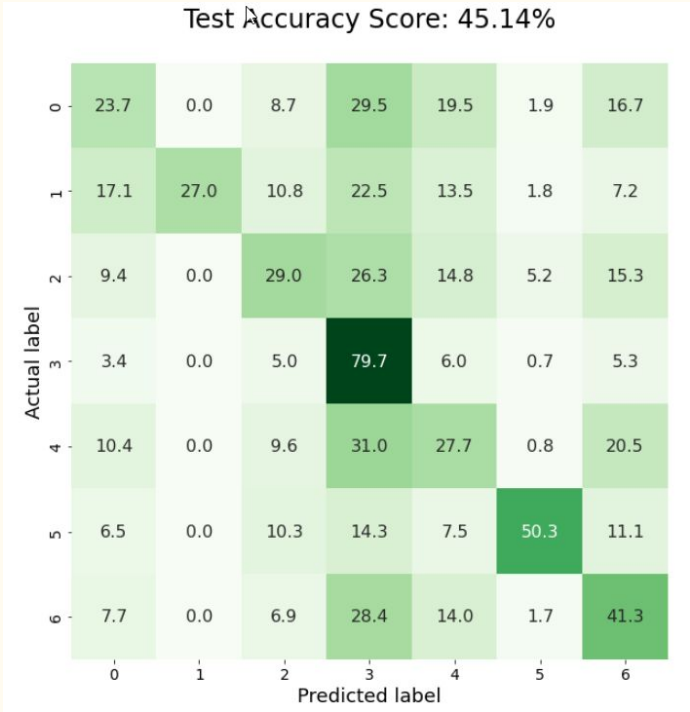
   Accuracy: 48.19%


Randomforest vs SGD Accuracy
RandomForest
SGD
Benchmark
0.5
0.4
0.3
0.2
0.1
0.0
44.90%
27.10%
20.06%
22.52%
17.21%
15.55%
45.14%
41.89%
48.19%
45.29%
Original
SIFT
Affine
HoG
HoG Rescaled

Confusion Matrix

SDG(HoG)

Randomforest (HoG)



Overall

- The best accuracy among BoF are below than benchmark.
- EigenFace and HoG were the best result.
- Alghouth highest accuracy 55.4% may seem not high, it is good result compare to benchmark.

Neural Network

RMSProp (Root Mean Squared Propagation)

Solves **vanishing gradients problem**

- Use a moving average of squared gradients to normalize them
- Divide the gradient by the root of this average

Adaptive learning rate

RMSProp Update Rule

$$v_t^w = \beta * v_{t-1}^w + (1 - \beta)(\nabla w_t)^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t^w + \epsilon}} * \nabla w_t$$

$$v_t^b = \beta * v_{t-1}^b + (1 - \beta)(\nabla b_t)^2$$

$$b_{t+1} = b_t - \frac{\eta}{\sqrt{v_t^b + \epsilon}} * \nabla b_t$$

Loss Function

Sparse Categorical Cross Entropy

CNN with TensorFlow

4 Sets of Convolutional and Pooling layers

- **Convolutional**:
  - 3x3 kernel size
  - RELU Activation
- **Pooling**:
  - 2x2

50% **Dropout** layer

- 2 **Dense** layers:
  - 512 Units, RELU
  - 256 Units, RELU
- **Dense**: 7 Units, Softmax

800 epochs

Model Summary

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 48, 48, 32)	320
max_pooling2d_12 (MaxPooling2D)	(None, 24, 24, 32)	0
conv2d_13 (Conv2D)	(None, 22, 22, 64)	18496
max_pooling2d_13 (MaxPooling2D)	(None, 11, 11, 64)	0
conv2d_14 (Conv2D)	(None, 9, 9, 128)	73856
max_pooling2d_14 (MaxPooling2D)	(None, 4, 4, 128)	0
conv2d_15 (Conv2D)	(None, 2, 2, 256)	295168
max_pooling2d_15 (MaxPooling2D)	(None, 1, 1, 256)	0

Model Summary

Layer (type)	Output Shape	Param #
dropout (Dropout)	(None, 1, 1, 256)	0
flatten (Flatten)	(None, 256)	0
dense (Dense)	(None, 512)	131584
dense_1 (Dense)	(None, 256)	131328
dense_2 (Dense)	(None, 7)	1799

Total params: 652,551
Trainable params: 652,551
Non-trainable params: 0

Image Augmentation in batches ~ 7s per epoch

- 40 degree random rotation
- 20% width/height shift
- Up to 20% zoom
- Up to 20% shear (stretch)
- Random horizontal flip

Image Augmentation: Rotation

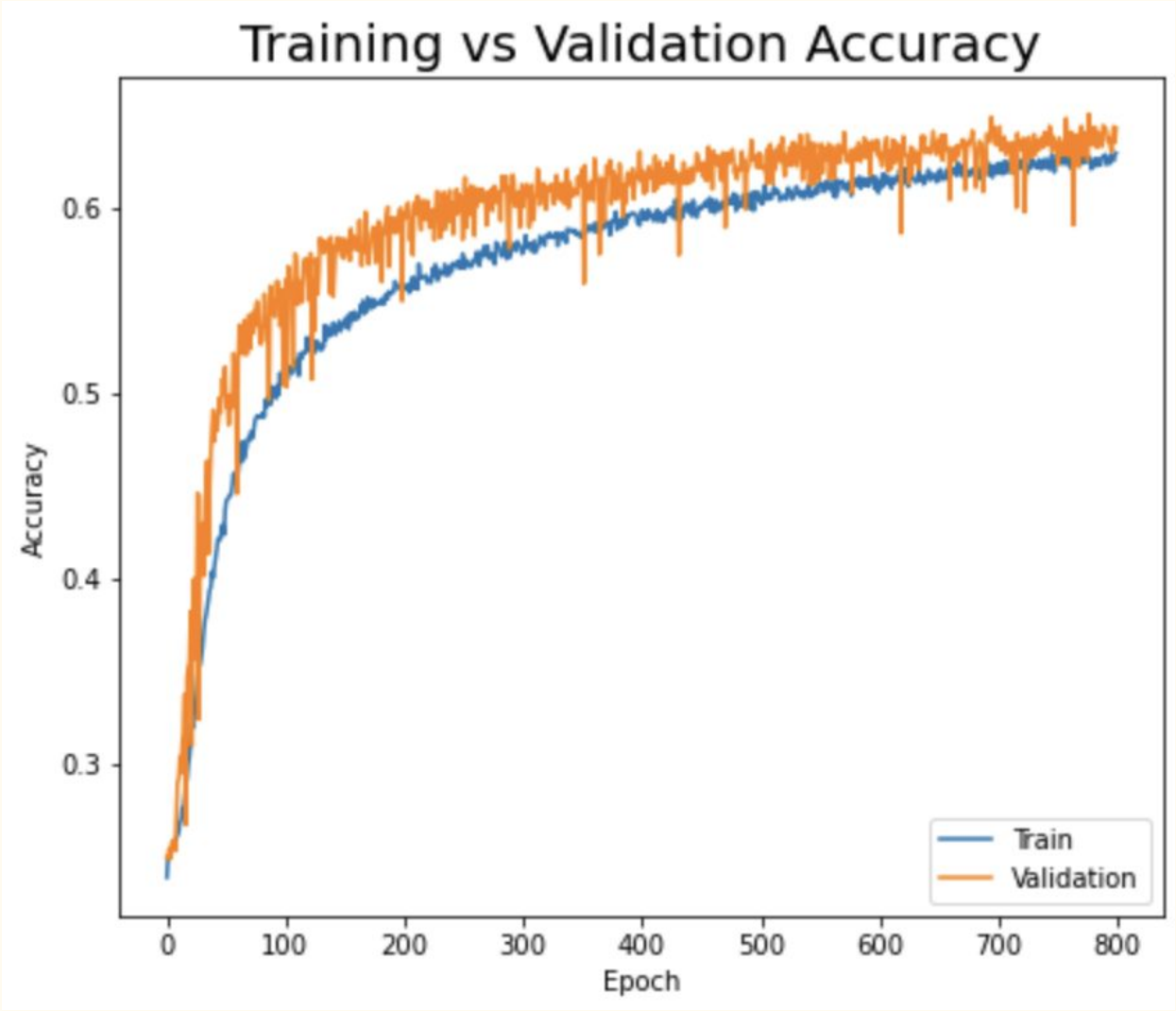
Image Augmentation: Width Shift

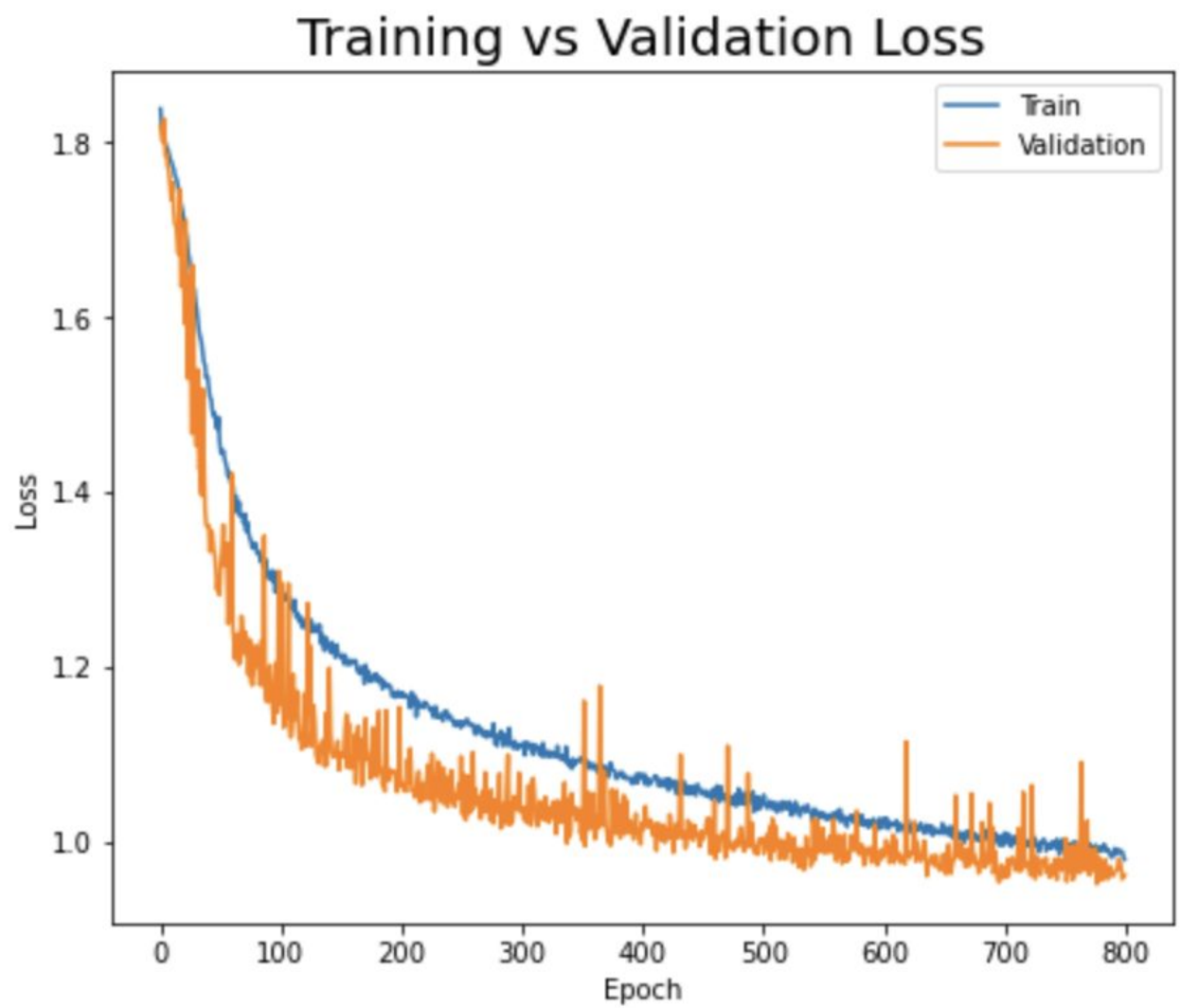
Image Augmentation: Height Shift

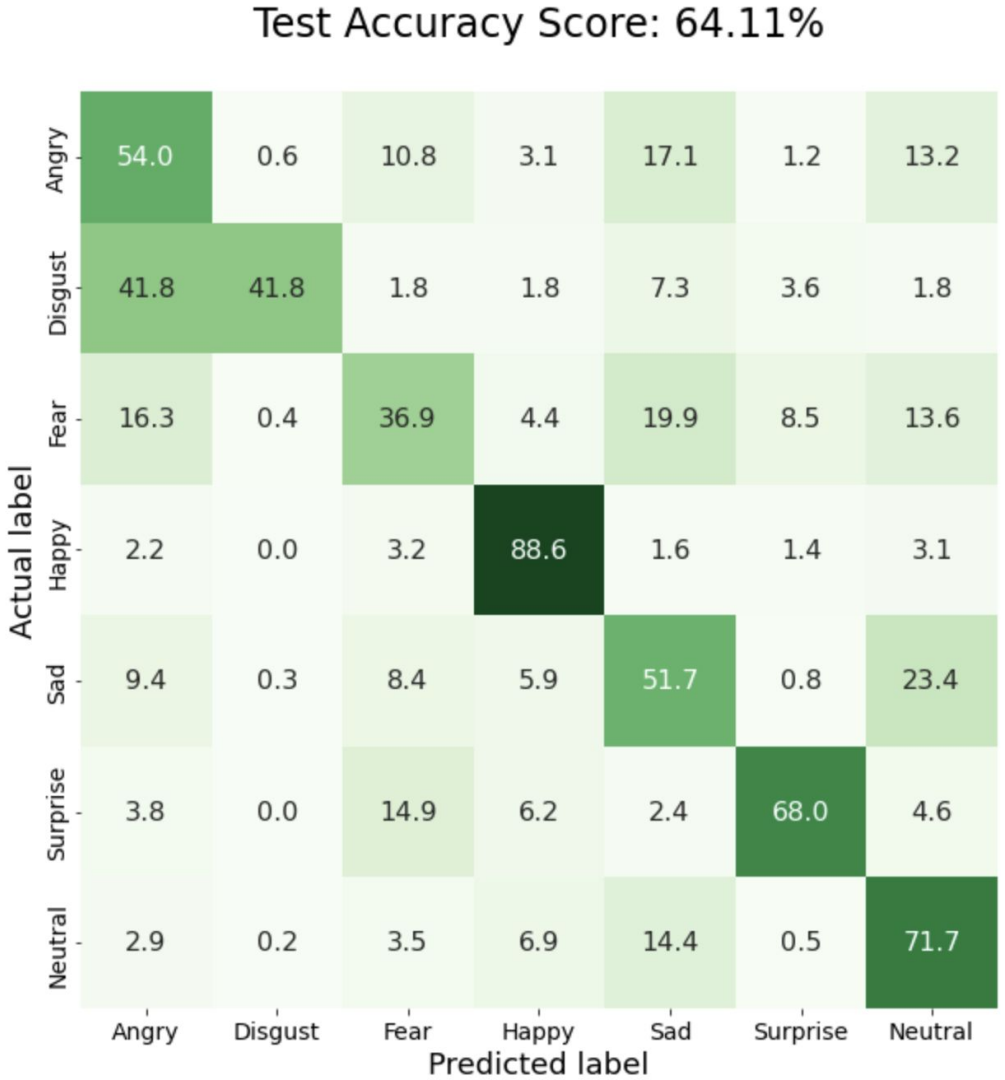
Image Augmentation: Zoom

Image Augmentation: Horizontal Flip

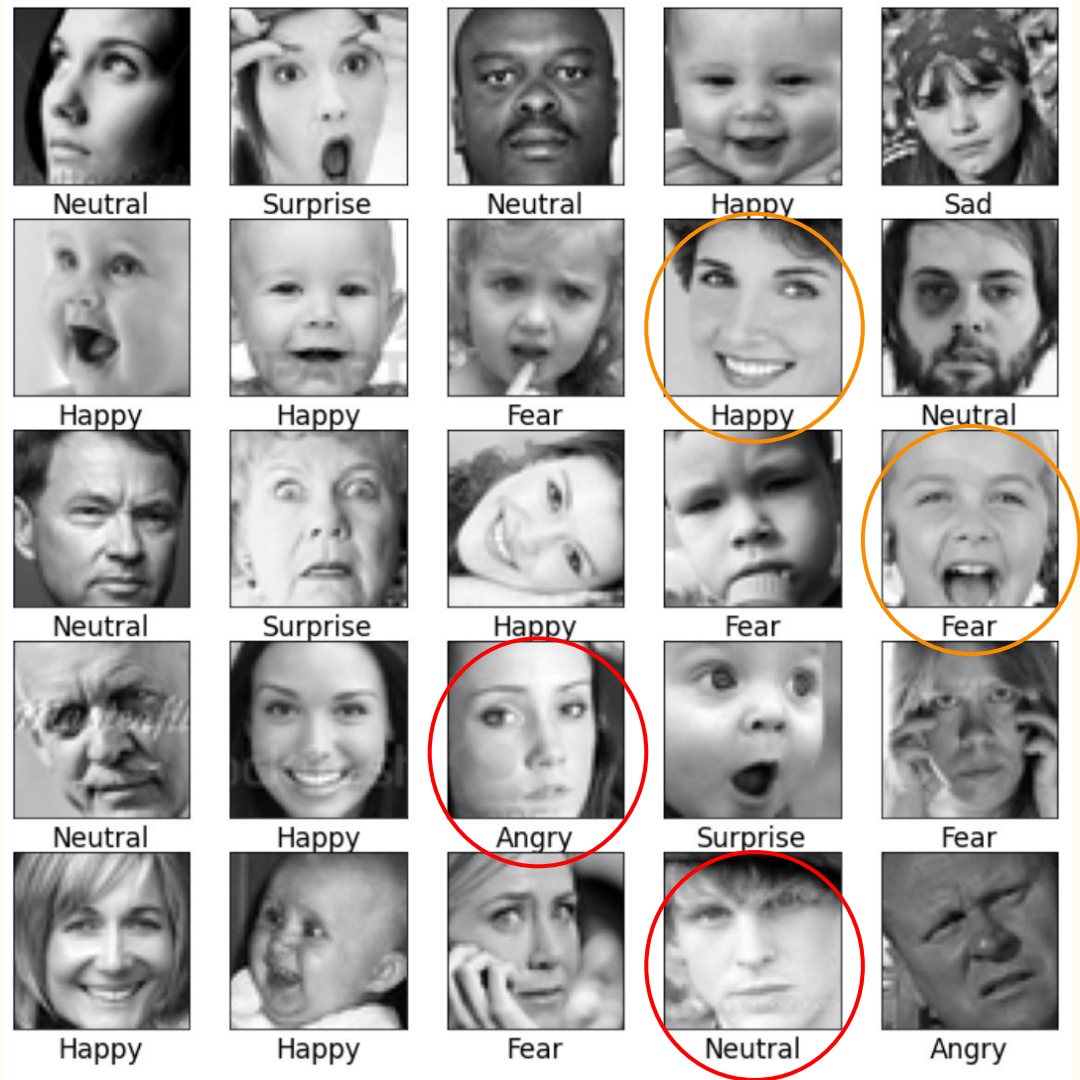
Image Augmentation: All Transformations


Training vs Validation Accuracy
Accuracy
0.3
0.4
0.5
0.6
0
100
200
300
400
500
600
700
800
Epoch
Train
Validation


Training vs Validation Loss
Train
Validation
Loss
1.8
1.6
1.4
1.2
1.0
0
100
200
300
400
500
600
700
800
Epoch



Diagonal elements represent the percent of labels for which the class predicted by the model was correct.

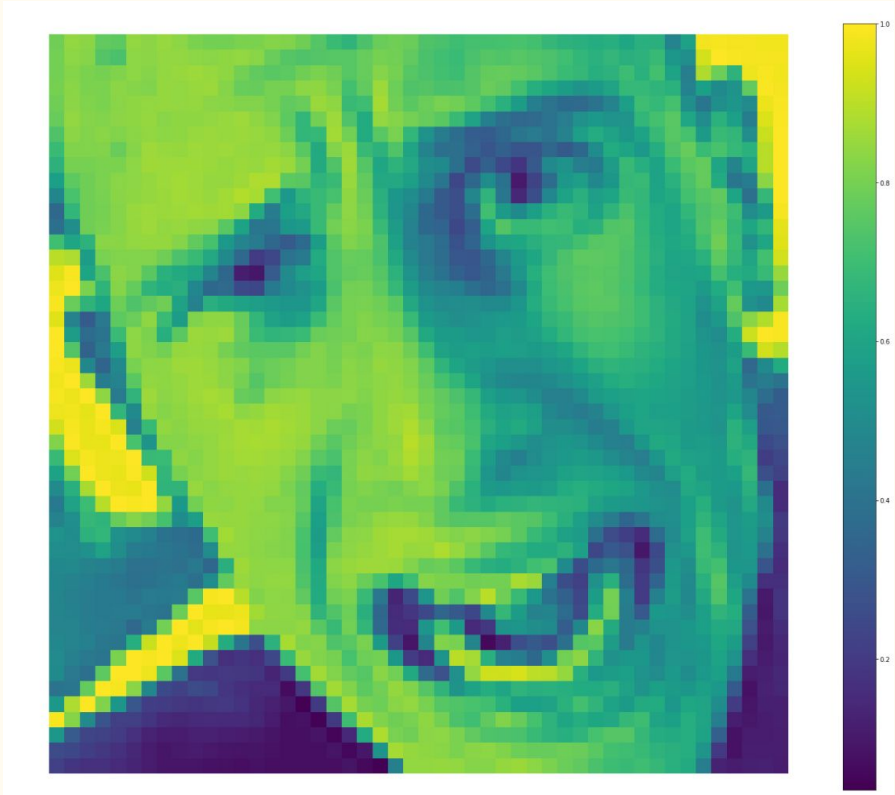


Top 5 models from Kaggle competition have accuracy between 65 and 71%

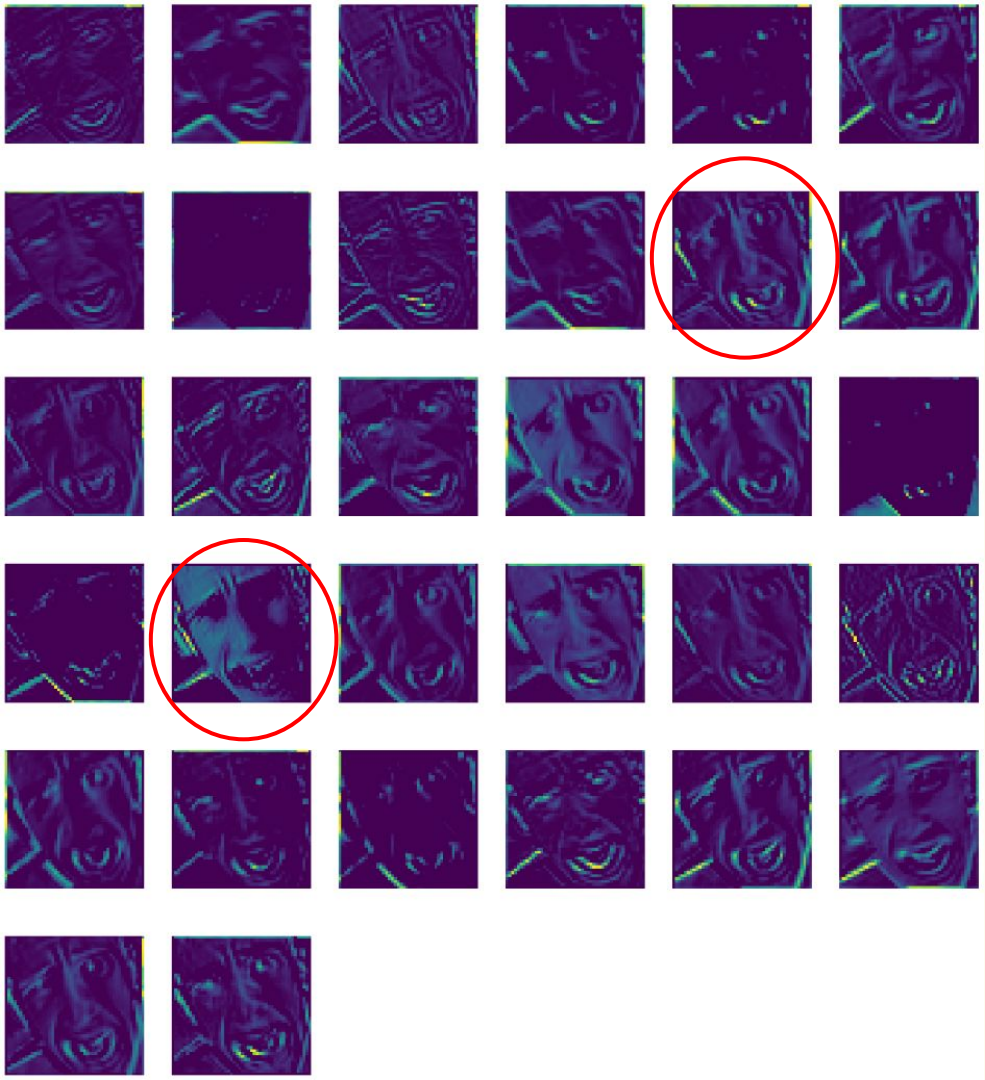
Learnings about Neural Networks:

- Data!!!
- Model design matters!
- Learn in batches

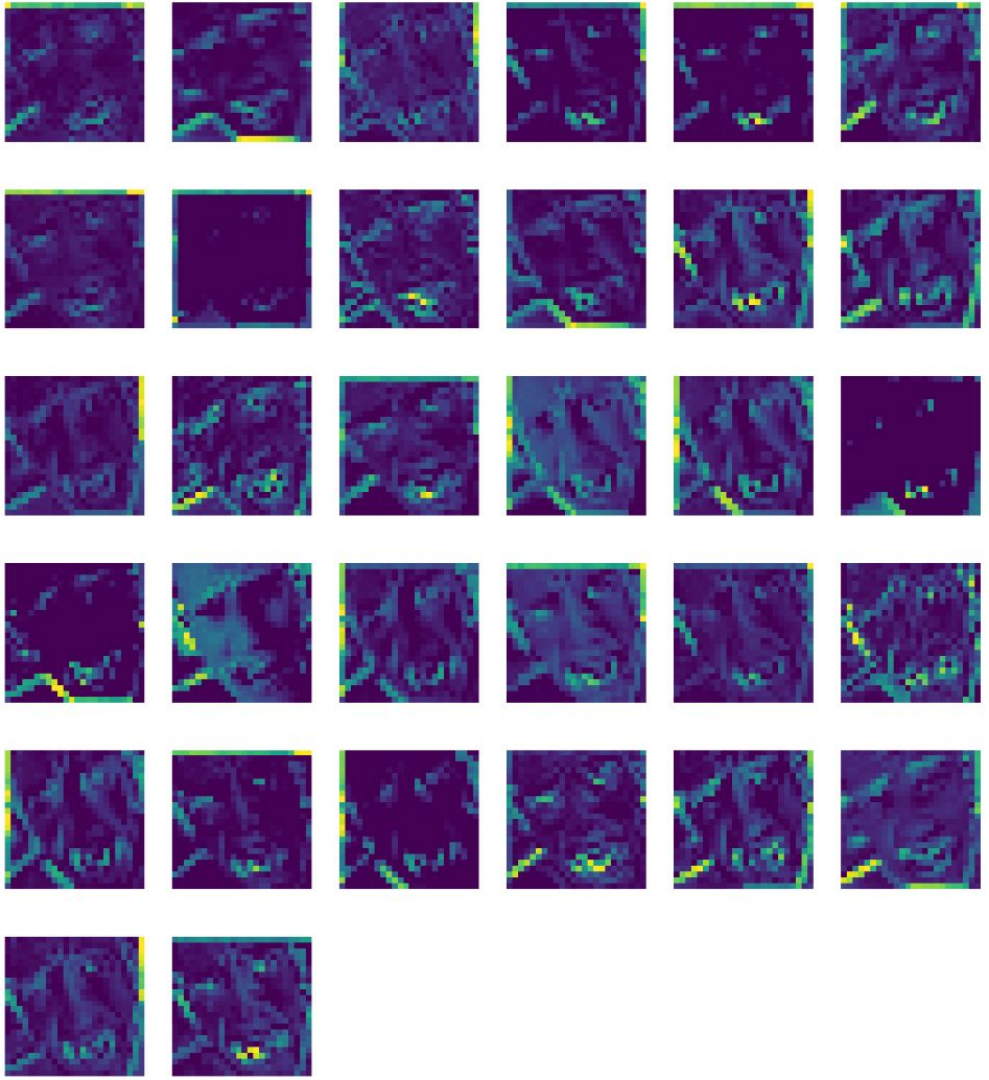
Activation Maps - Original Image - Fear



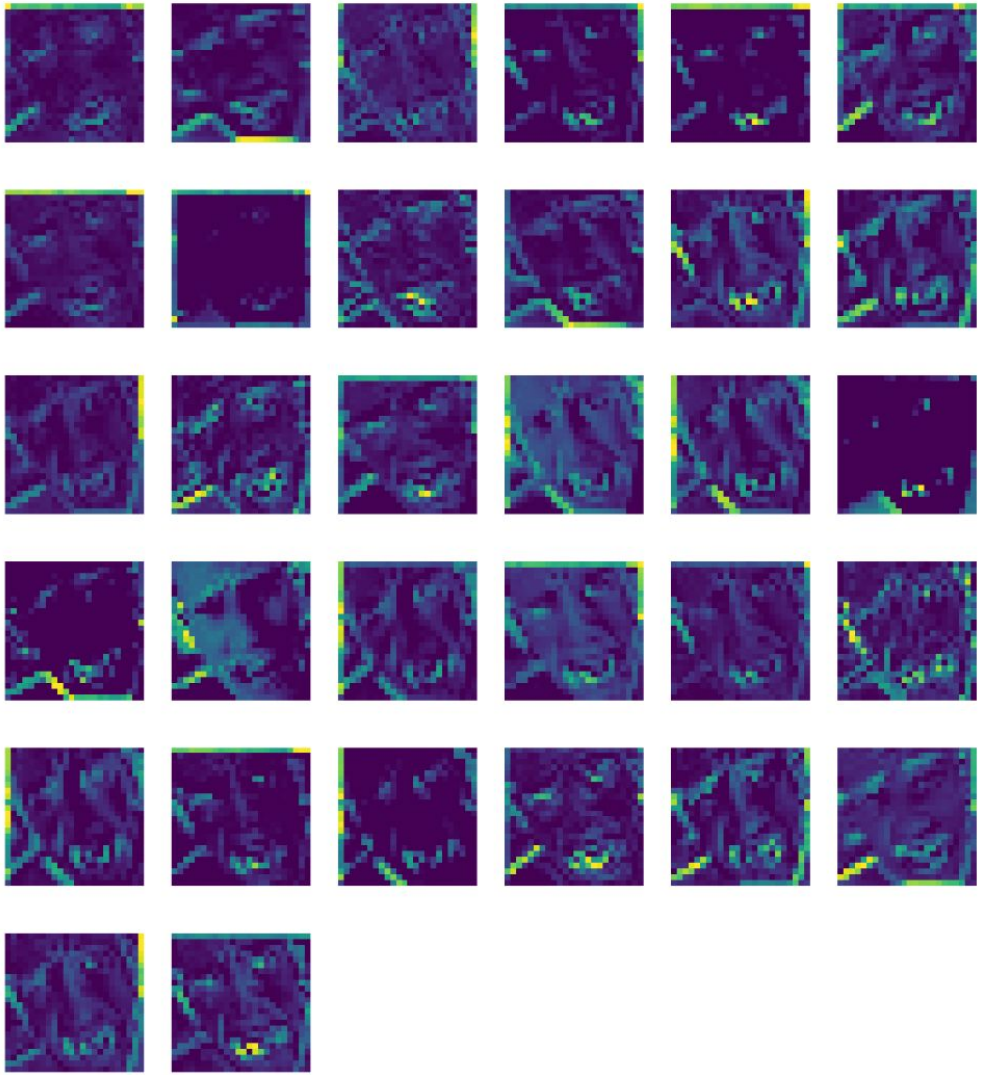
1st Convolutional Layer
(48 x 48)



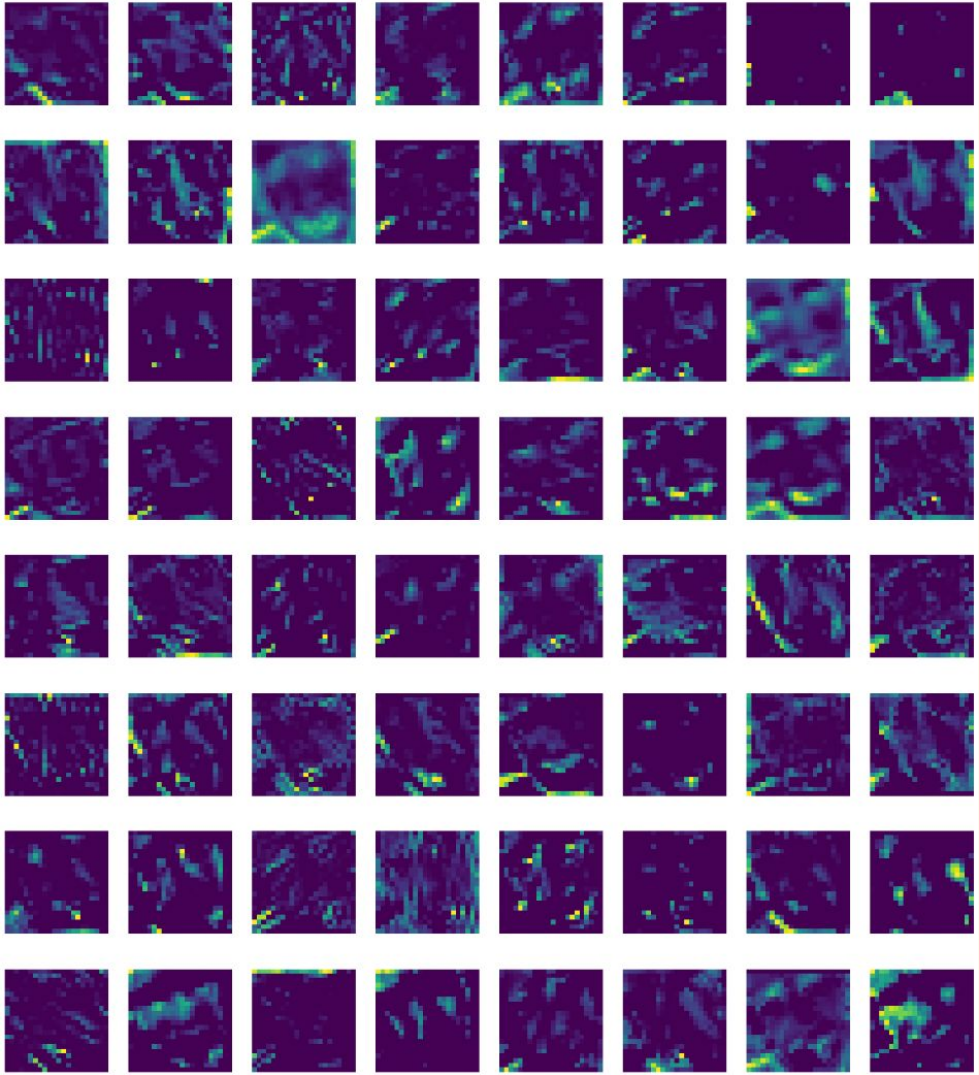
1st Pooling Layer
(24 x 24)



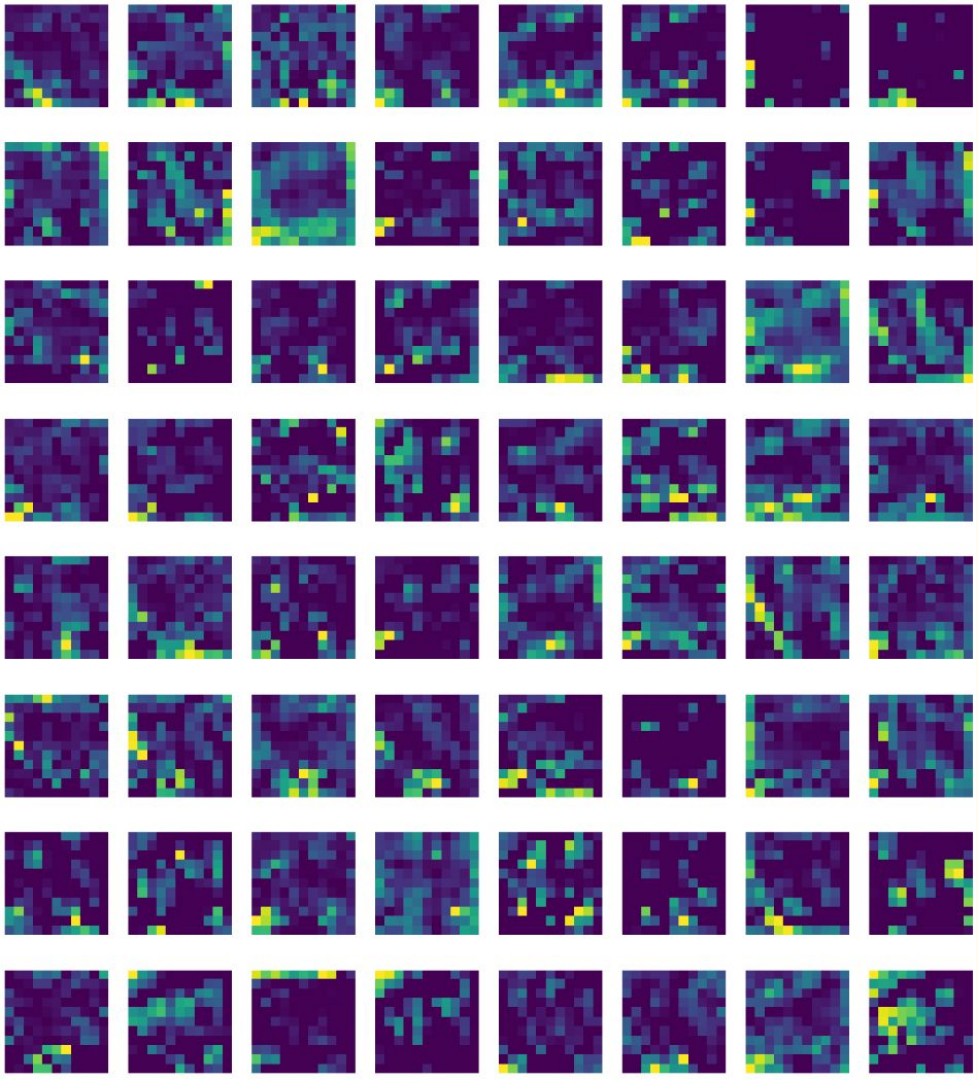
2nd Convolutional Layer
(22 x 22)



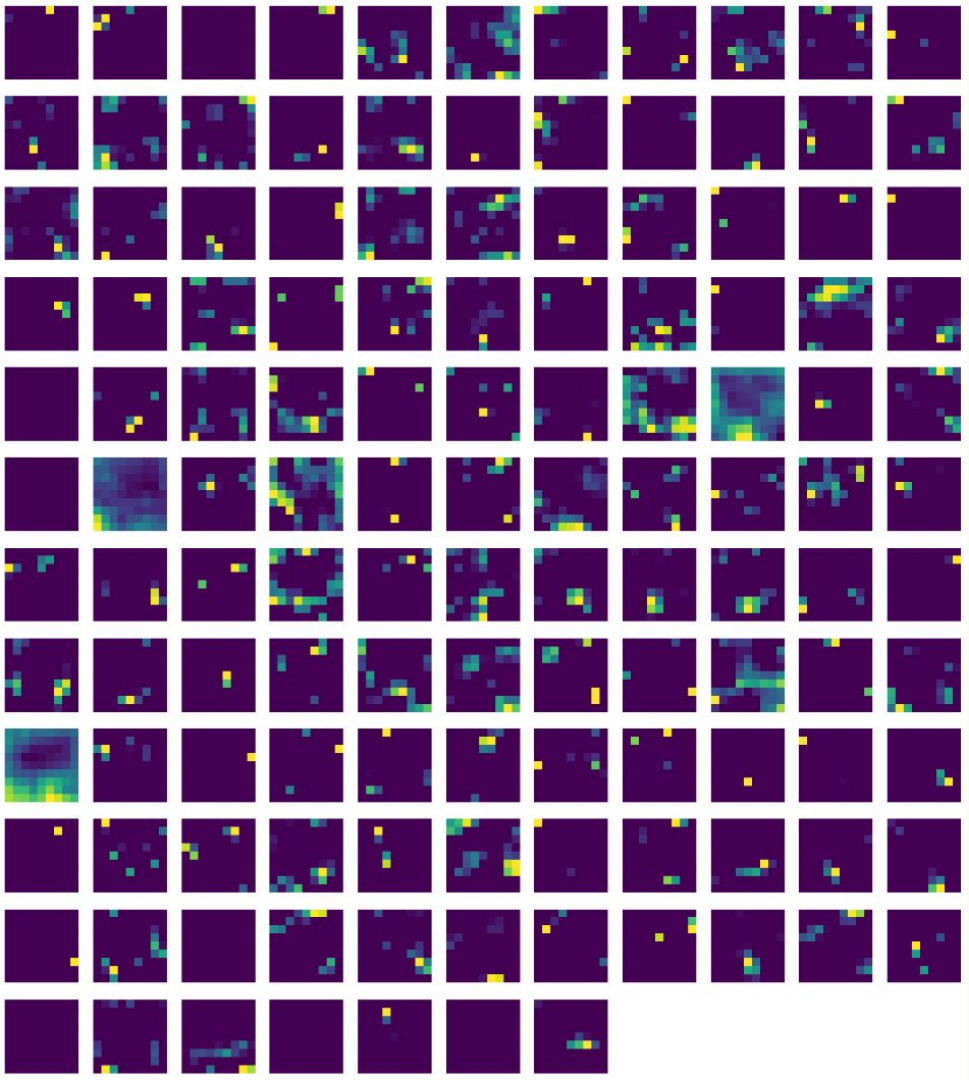
2nd Pooling Layer
(11 x 11)



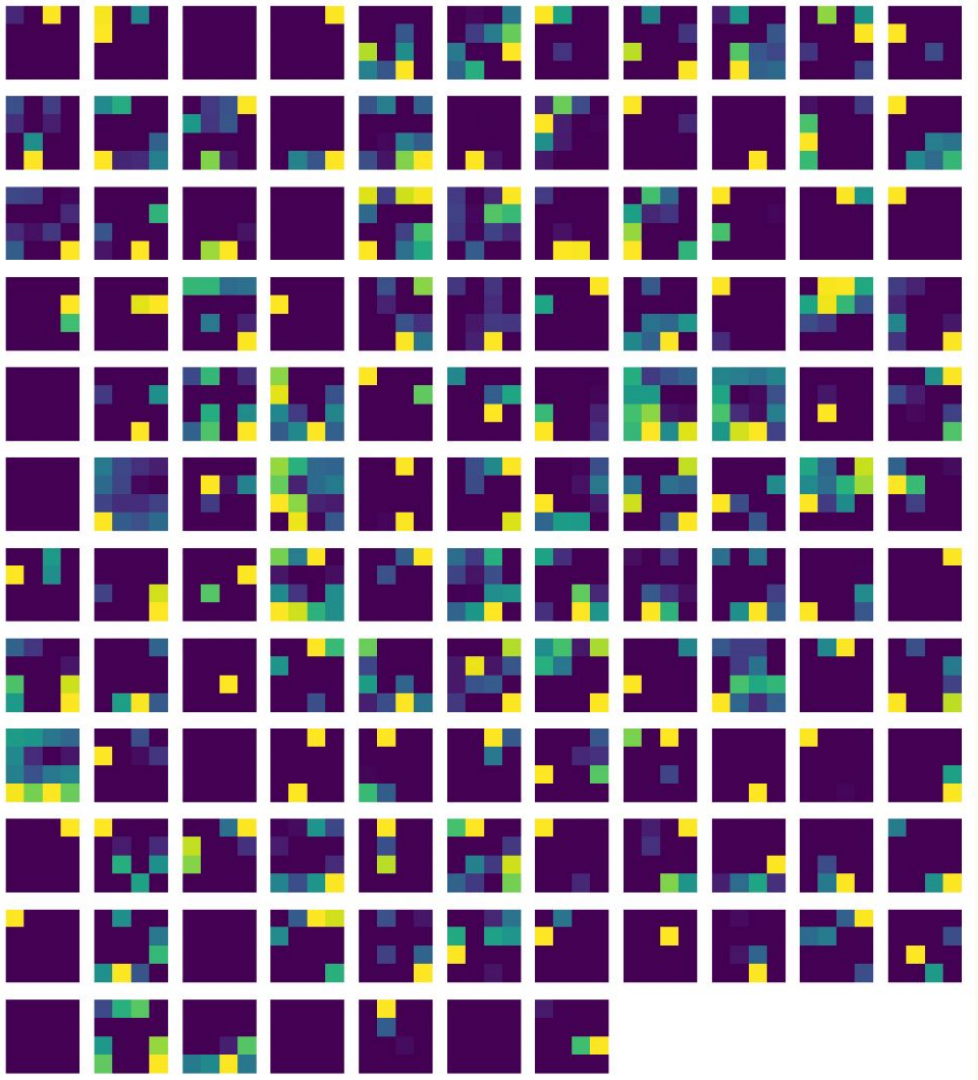
3rd Convolutional Layer
(9 x 9)



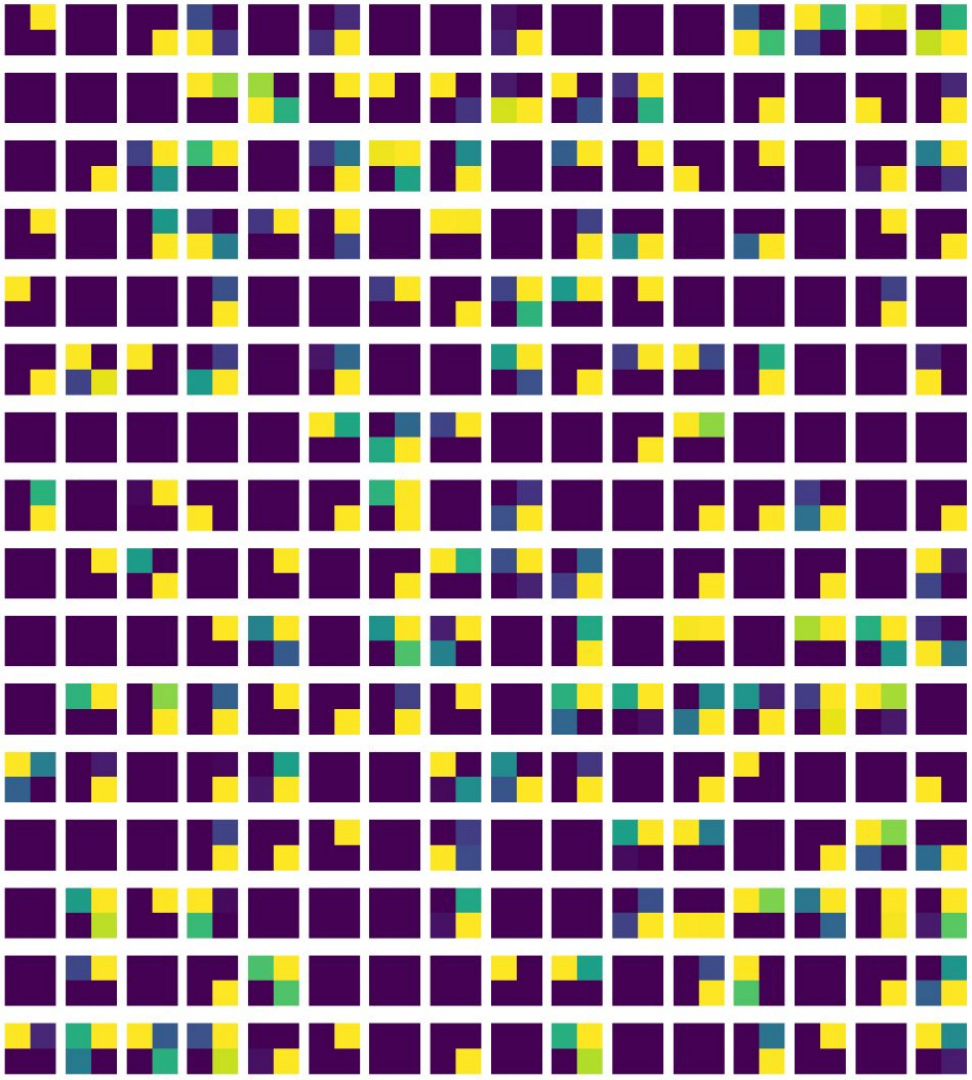
3rd Pooling Layer
(4 x 4)



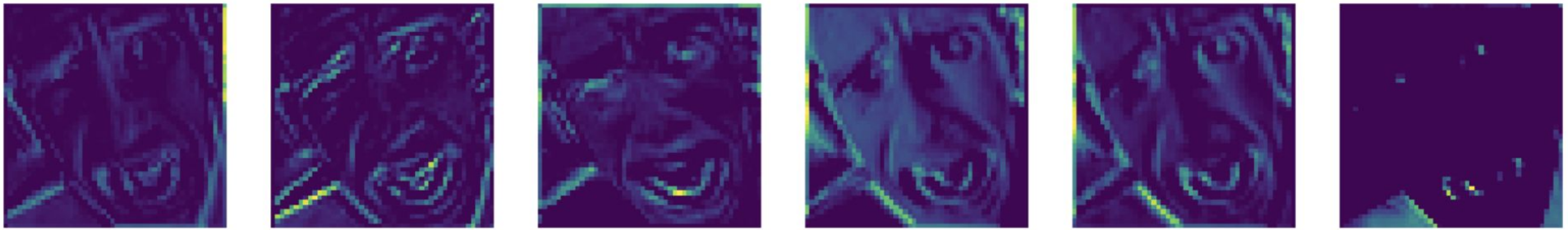
4th Convolutional Layer
(2 x 2)



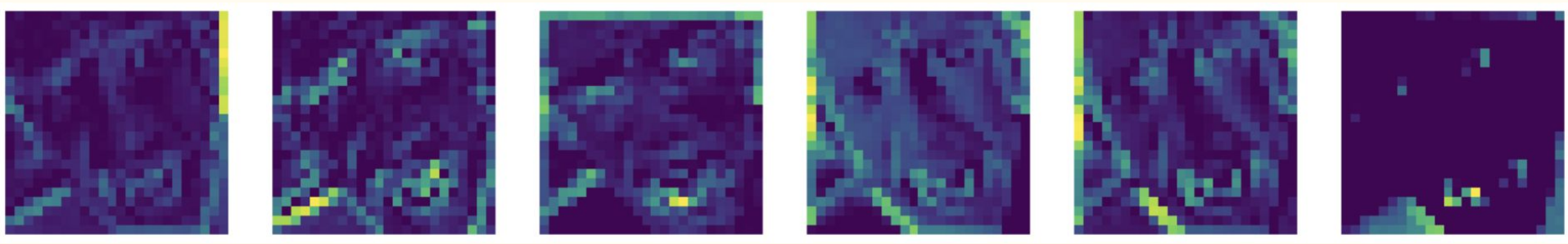
4th Pooling Layer
(1 x 1)



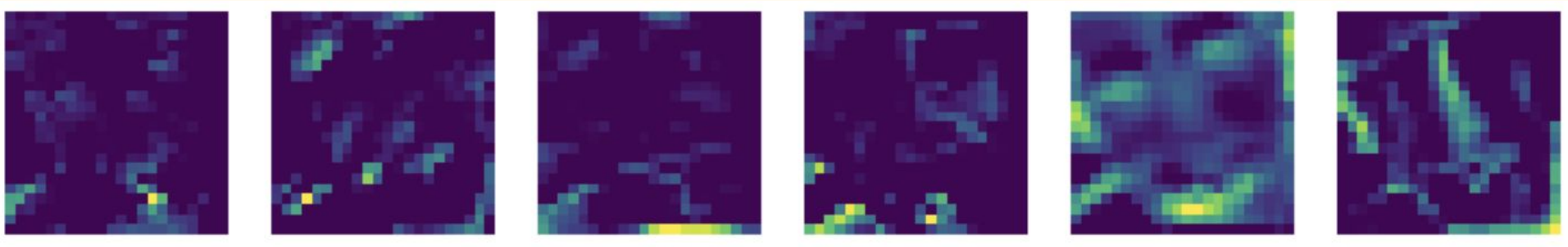
1st Convolutional Layer (48 x 48) - zoom



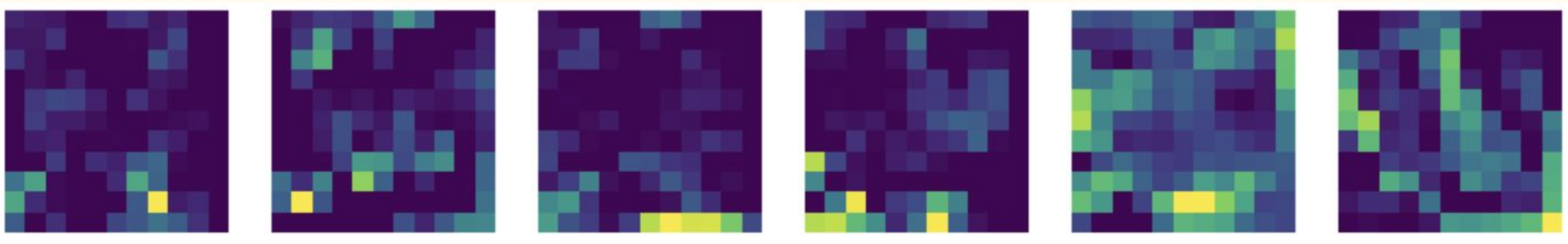
1st Pooling Layer (24 x 24) - zoom



2nd Convolutional Layer (22 x 22) - zoom



2nd Pooling Layer (11 x 11)



Demonstration

Contribution

chikakoto / facial-expression-recognition Public

Pin | Unwatch 2 | Fork 0 | Star 0

Code | Issues 3 | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

- Pulse
- Contributors
- Community
- Community Standards
- Traffic
- Commits
- Code frequency
- Dependency graph
- Network
- Forks

April 17, 2022 – May 17, 2022

Period: 1 month

Overview

4 Active pull requests

7 Active issues

4	0	4	3
Merged pull requests	Open pull requests	Closed issues	New issues

Excluding merges, **5 authors** have pushed **50 commits** to main and **68 commits** to all branches. On main, **24 files** have changed and there have been **27,143 additions** and **88 deletions**.

4 Pull requests merged by **3** people

Merging personal branch to main
#12 merged 2 days ago

Chikako

Thank you