CS 539: MACHINE LEARNING

DISTRACTED DRIVER DETECTION

Guidance:
Prof. Kyumin Lee
Team:
Vrushabh Desai
Rishabh Chadha
Rajendra Kante
Kratika Agrawal
Priyanka Benachamardi

MOTIVATION

INTRODUCTION

- **Goal:** To detect if the diver is **driving safe or performing any activity**
- Classify drivers' activity into 10 categories

c0: safe driving

c1: texting - right

c2: talking on the phone - right

c3: texting - left

c4: talking on the phone - left

c5: operating the radio

c6: drinking

c7: reaching behind

c8: hair and makeup

c9: talking to passenger

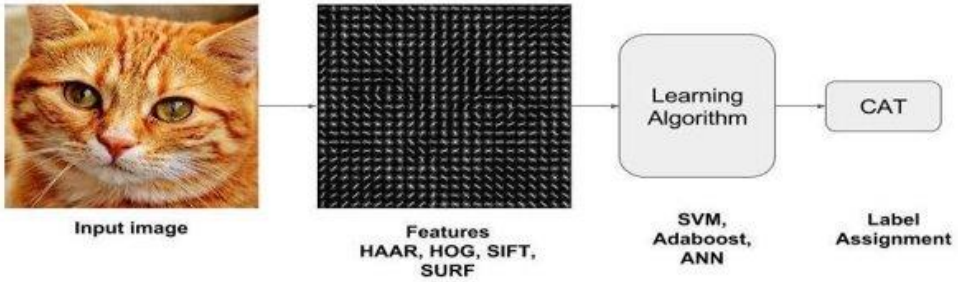
DATASET

- Distracted Driver Images from Kaggle State Farm Competition
- Includes 22.5k labelled images with approx. 2.2k images belonging to each class.
- 640x480 RGB images
- 70k unlabelled Images

FEATURE EXTRACTION TECHNIQUES

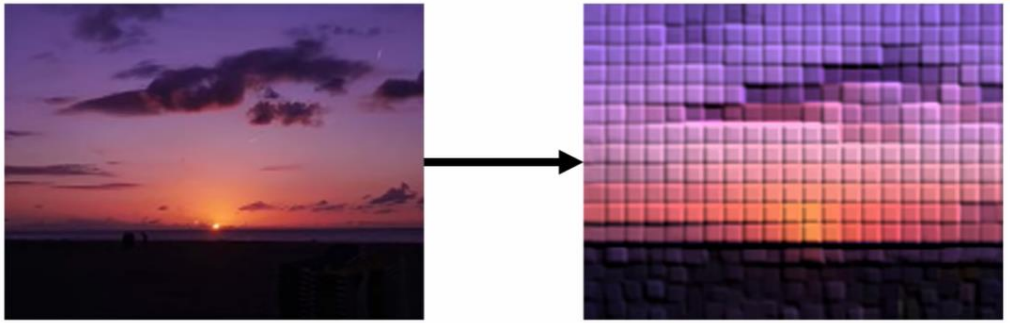
Image classification is best done by extracting features such as

- Pixel
- HOG
- Sobel
- Clustering



PIXELS AS FEATURES

- Resize image to 64x64 RGB image
- Stacked all the RGB pixels values in a single feature vector of size 64x64x3



CLUSTERING

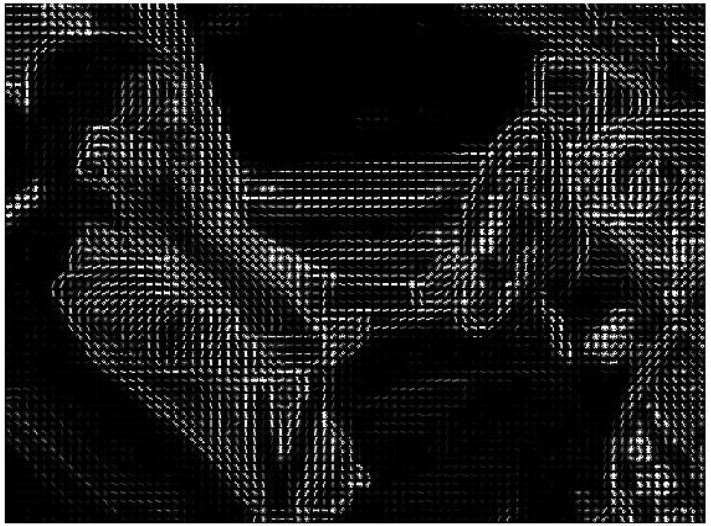
- Resize image to 64x64 RGB image
- Implement K-means clustering on each image with k=3 (in our case)
- Stack cluster features to create feature matrix

HOG FEATURE DESCRIPTOR

- Counts occurrences of gradient orientation in localized portions
- Stack HOG gradient features to generate a feature matrix

Input image

Histogram of Oriented Gradients

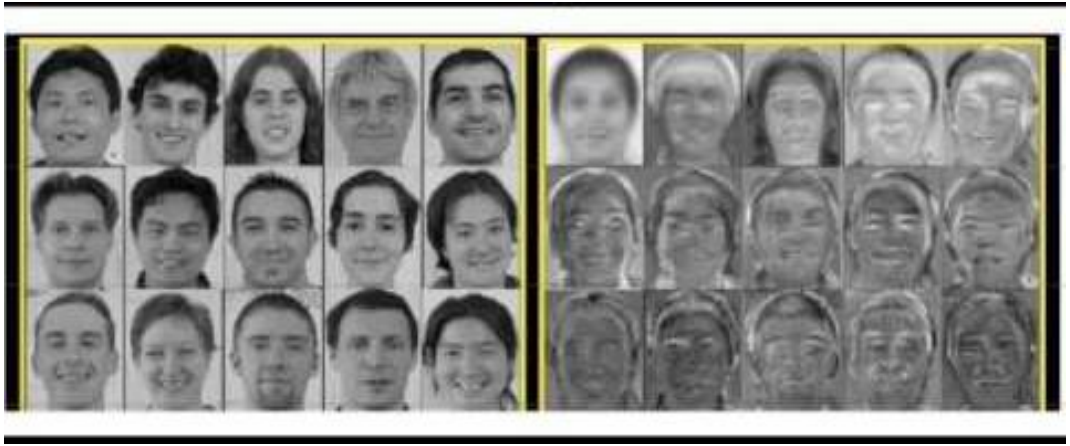


SOBEL EDGE DESCRIPTOR

- Resize image to 64x64 RGB image
- Obtained edges using Sobel gradient in X and Y direction
- Stacked object edges as feature vector

PRINCIPAL COMPONENT ANALYSIS

- The images are resized to 64 x 64 RGB images
- Individual pixels are used as features.
- Used PCA for dimensionality reduction
- Number of features reduced from 12,288 to 930 on retaining 95% variance.



ALGORITHM IMPLEMENTATION

- Split labelled data set into 75% Train, 15% Validation and 10% Test data.
- Used Scikit-learn to implement following algorithms:
  - Decision Tree
  - Support Vector Machine
  - Random Forest
- Implemented 2 Layer fully connected Neural Network
- Implemented Convolution Neural Network using Tensorflow

1. SUPPORT VECTOR MACHINE

- Implemented SVM on HOG features of the image
- Used kernels : Linear and Gaussian
- Varied value of "C" in order to obtain better accuracy

Train Data	Accuracy
64x64 RGB Image	74.6%
HOG FDs	79%

2. DECISION TREE

- Train Data: Gray images, RGB images, PCA features, Sobel Edges
- Tried Pruning

Image Resolution	Accuracy
Gray Scale Image	49.6%
PCA	80.2%
Sobel Egde Images	81.9%
64x64 RGB Image	87.1%

3. RANDOM FOREST

- Scikit-Learn Library
- Different number of trees
- Train data: Gray images, RGB images & K-means clustered images

Train data	Accuracy
64x64 Gray Image	78.1%
K-means clustered	77.3%
RGB Image	89.4%

4. 2-LAYERED NEURAL NETWORK

- Keras library
- Different combinations of number of neuron units
- Varied activation functions on layers

Image Resolution	Accuracy
64x64 Gray Scale Image	88.6%
64x64 RGB Image	92.7%

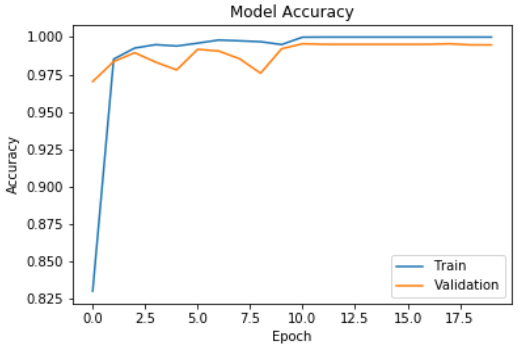
5. CONVOLUTION NEURAL NETWORK

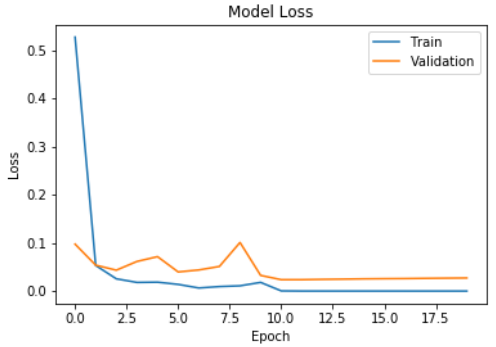
- Trained model on 100x100 sized RGB images data set using TensorFlow.
- Explored various hyper parameters such as:
  - Number of Kernels and Kernel Size in convolution layer
  - Number of Neurons in hidden layer
  - Number of Epochs
  - Optimizer: Adam and SGD
  - Learning Rate in SGD
  - Number of CNN Layers and Hidden Layers

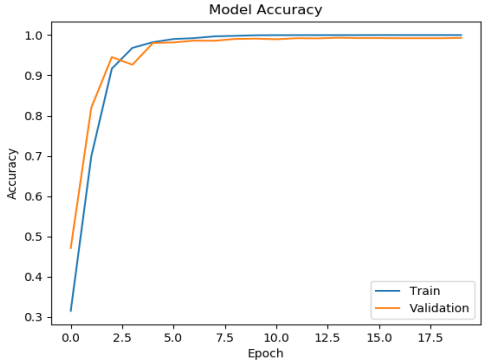
Hyperparameter	Parameter	Accuracy
No. of Convolution Kernel (Kernel Size = 3x3)	32	98.63%
	64	98.47%
Size of Kernel (No. of Convolution Kernel = 32)	3x3	98.63%
	5x5	97%
	9x9	96.4%

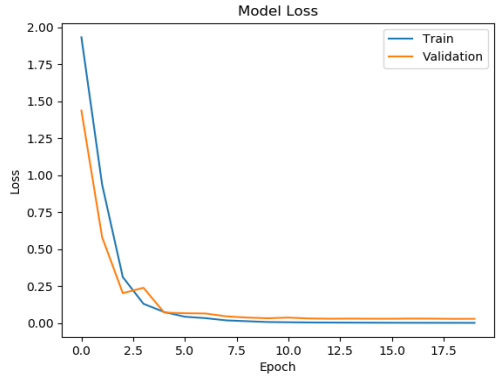
OPTIMIZER : ADAM VS SGD

- Test Accuracy:
  - Adam: 97.89%
  - SGD (Learning rate = 0.01): 97%
- Adam generalize poorly compared to SGD, but it converge faster compare to SGD



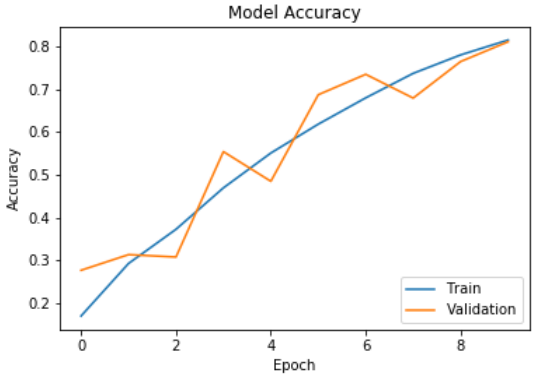




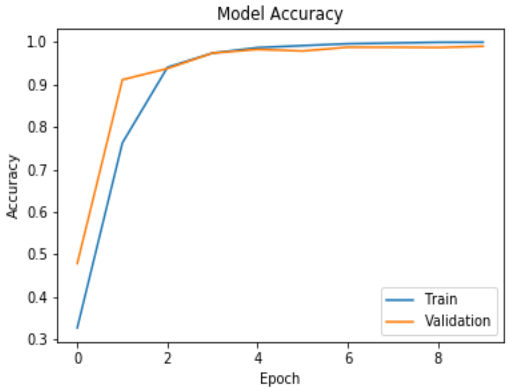


SGD: LEARNING RATE

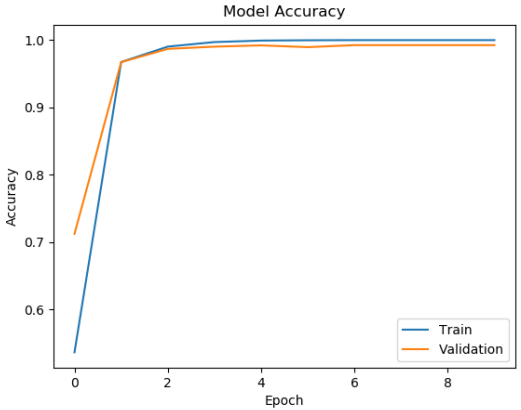
- All the model hyper parameter are constant varying learning rate.



Learning Rate: 0.001
Test Accuracy: 81%



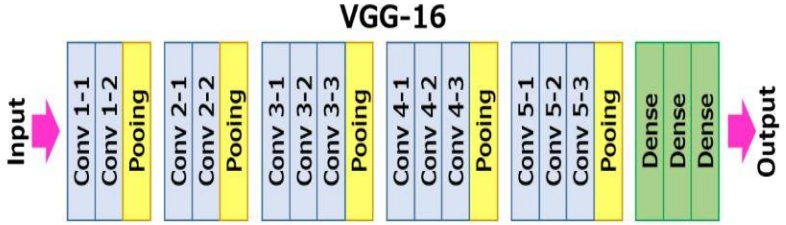
Learning Rate: 0.01
Test Accuracy: 96.8%

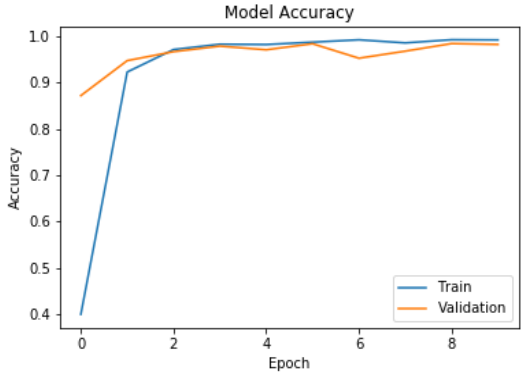


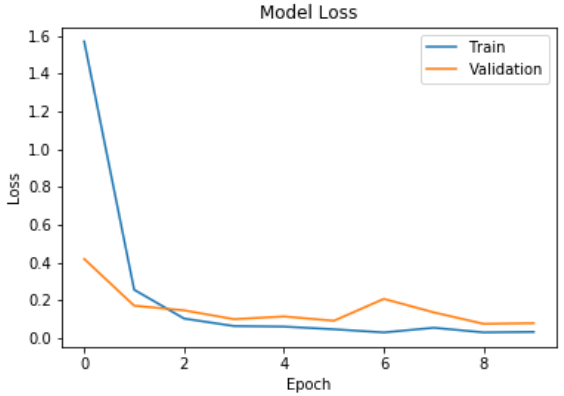
Learning Rate: 0.1
Test Accuracy: 97.32%

VGG16 ARCHITECTURE

- VGG16 is used for Deep Convolutional Networks for Large-Scale Image Recognition.
- Test Accuracy: 98.28%
- VGG16 model generalizes very well







EVALUATION

Model	Accuracy	Generalized Accuracy
Decision Tree	87.1%	41.6%
Random Forests	89.4%	43.8%
SVM	79.8%	60.2%
Neural Network	92.7%	42%
CNN	98.28%	64.3%

- The best results were obtained using a **CNN with VGG16 architecture**.

KEY TAKEAWAYS

- Feature extraction is an important step.
- Selection of model
- Good features are needed for Dimensionality Reduction
- Generalized accuracy depends on distribution of real-world data in comparison to test data.

FUTURE WORK

- Explore more feature extraction techniques
- Combine various feature vectors for Image Representation
- Implement modern Deep Neural Network Architectures
- Recurrent Neural Nets (RNN)

ANY QUESTION?