Vehicle Detection

and

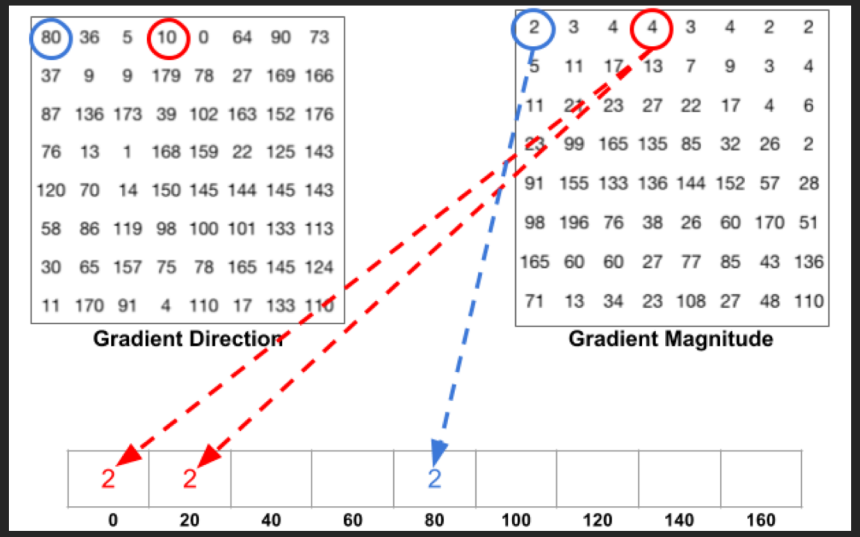
Make Model Classification

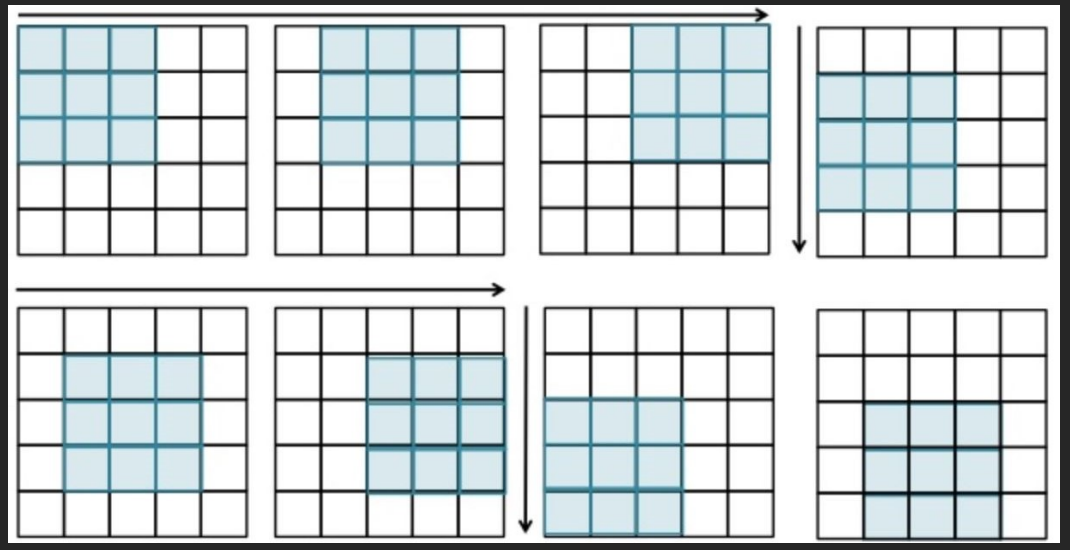
Problem Definition

- To Detect vehicles from an input image
- To Classify the Make, Model and Year of the detected vehicle

Methods

- Histogram of Oriented Gradient
- To pre-process the images into feature vectors
- Resize input images to 64 x 64. On every 8 x 8 cells inside the image, pixel value of gradient magnitude and orientation contribute to each cell's histogram with 12 bins.
- Perform 16 x 16 Block Normalization.
- Produce an 1 x 7056 vector per image.
- Get Help from Scikit-Image Implementation



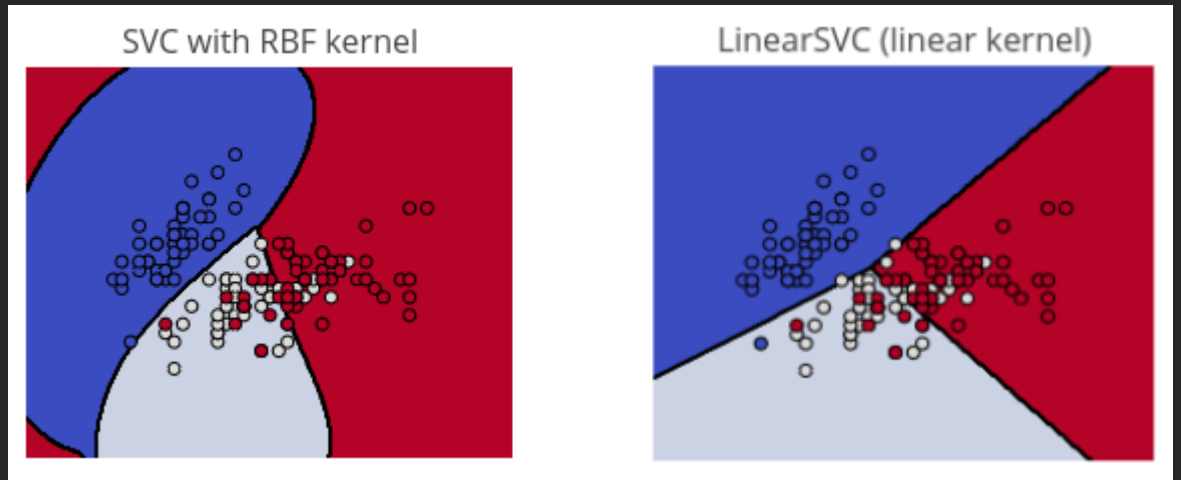


Left picture is Individual pixel value contribute to the histogram , right is visualisation of block normalization.

Methods

- Support Vector Machine (SVM)
- Trained with 7000 vehicle images and 7000 non-vehicle images from KITTI Dataset
- The SVM will find an optimal hyperplane that separates the two labels of HOG feature vectors with the maximum margin in high dimension
- Experiment different kernel functions : Radial (RBF) Kernel vs Linear Kernel
- Radial Kernel : Higher accuracy, slower to train
- Linear Kernel : Lower accuracy, faster to train, not so easy to be overfit
- Get help from Scikit-Learn implementation.

Radial Kernal (left) and Linear Kernel (right)



Methods

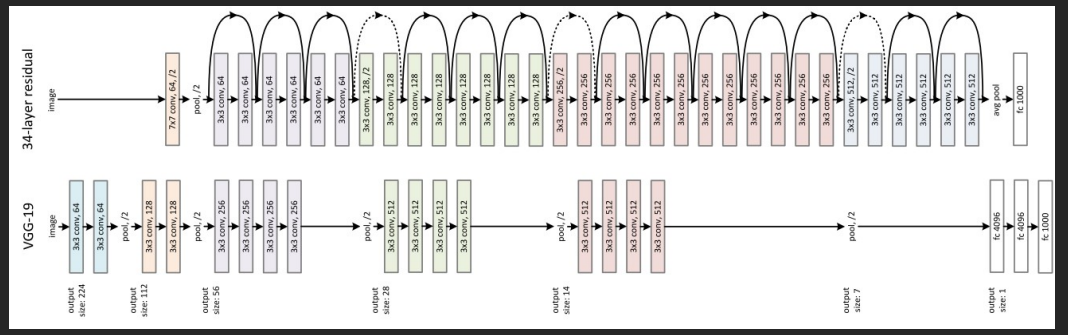
- Sliding Window
- It finds the region that it's going to be sliding around.
- Make a list of window positions, then checks each of those windows.
- Using the trained SVM to detect each window is vehicle or not.
- Store the coordinates for 'vehicle' windows as valid windows.
- Find the 'group' of valid windows, in other words, sort the interconnected windows into a group, find the whole window of the entire car.

Source: https://github.com/jaredjxyz/CarND-Vehicle-Detection

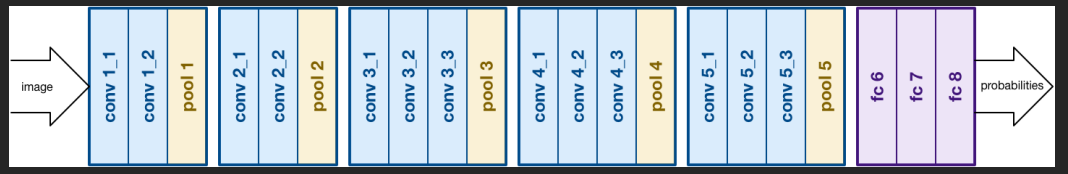


Methods

- Convolutional Neural Network (CNN)
- Trained with 6640 images from VMMR Dataset and Stanford Car Dataset.
- Consists several layers, including Convolutional Layers (Conv2D), Pooling Layers (MaxPooling2D), Rectified Linear Unit Layers (ReLU) and Fully Connected Layers.
- Convolutional Layers, Pooling Layers and Rectified Linear Unit Layers provides high-level feature to Fully Connect Layers for classification.
- Fine-Tuning vs Train From Scratch.
- Experiment fine-tuning with several design including VGG16, VGG19, ResNet etc.
- Experiment train-from-scratch with several design including VGGNet etc.
- We use Keras Library to implement our CNN



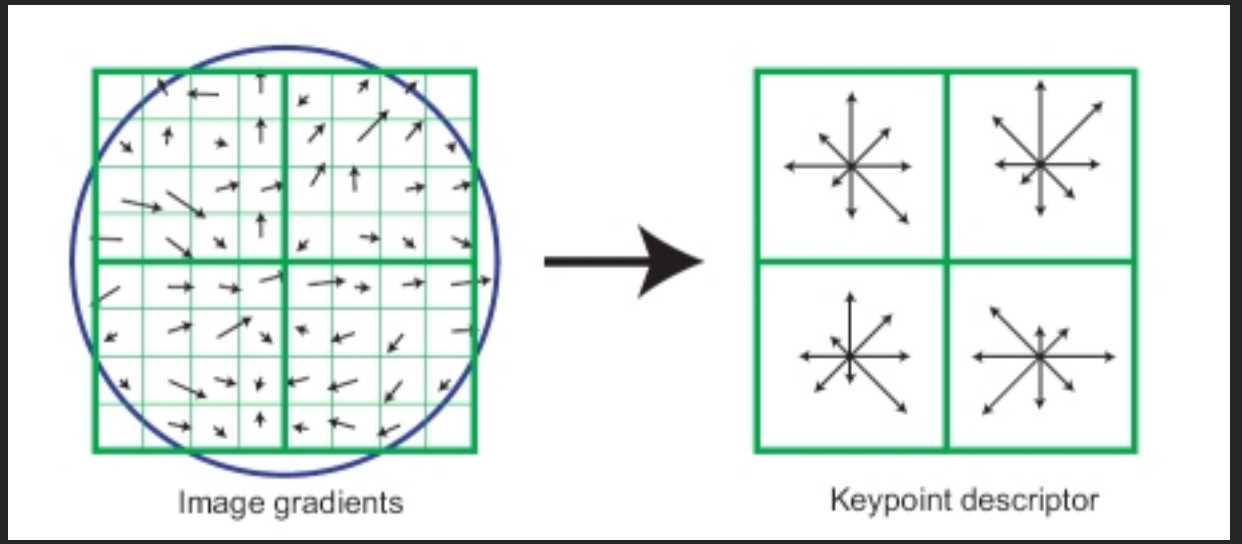
ResNet34 and VGG19



VGGNet Architecture

Methods - Alternative Approach

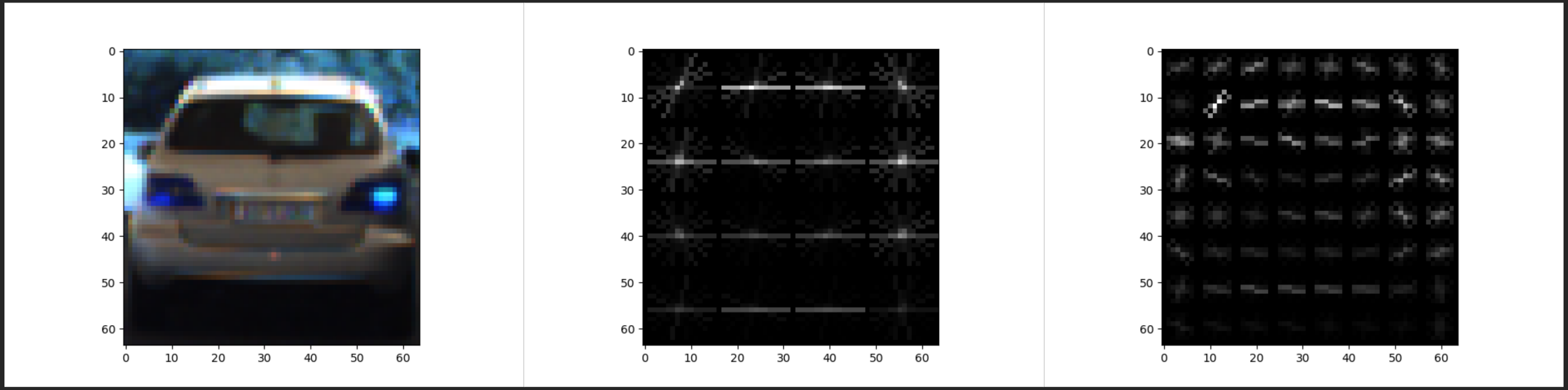
- SIFT Feature Matching and Homography Transform
- Compute SIFT feature for detected image, and matching the SIFT feature with some dataset images belong to various class.
- Compute Homography with RANSAC approach after matching, count inlier and classify the detected vehicle based on the highest percentage of inlier.



Example of SIFT Descriptor

Result

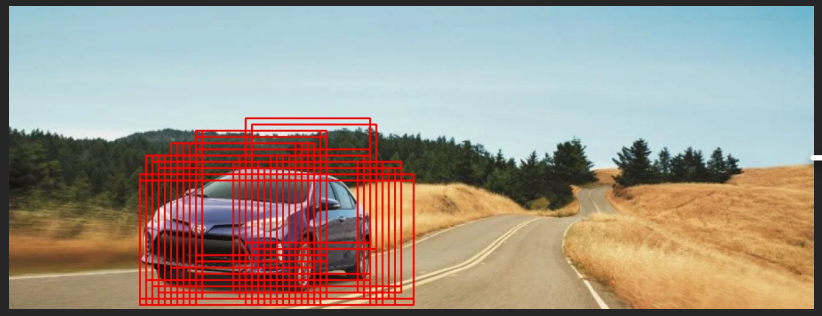
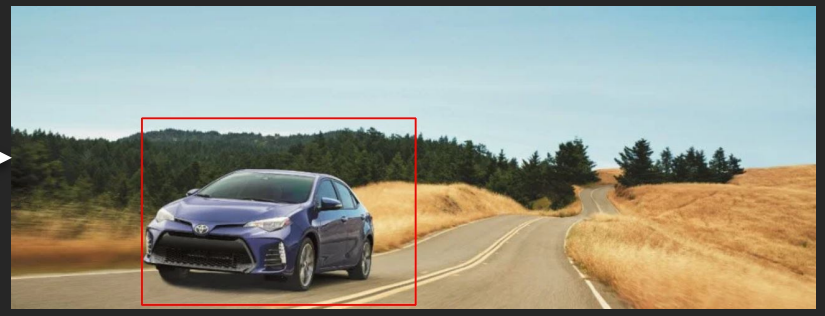
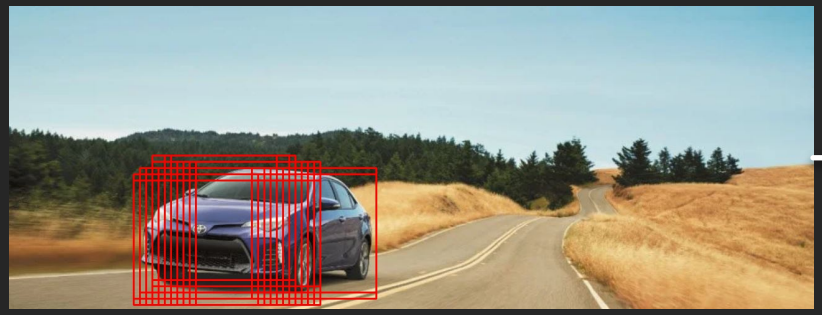
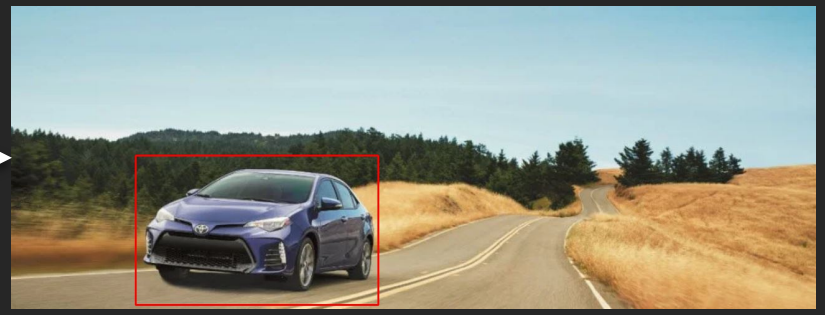
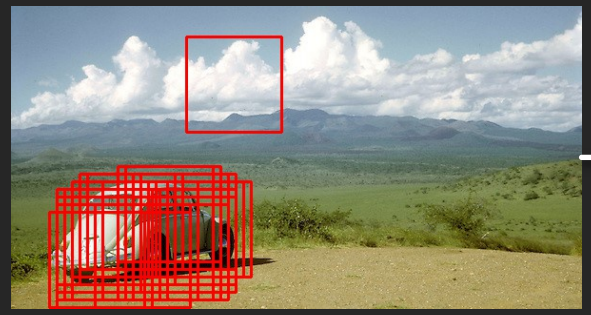
- Extraction of HOG feature in HSV space
- Comparison of different parameters: 1x972 vectors vs 1x7056 vectors
- Stored the computed vector for use of SVM
- SVM achieve higher accuracy when using 1x7056 vectors



Visualization of different sizes of HOG descriptor

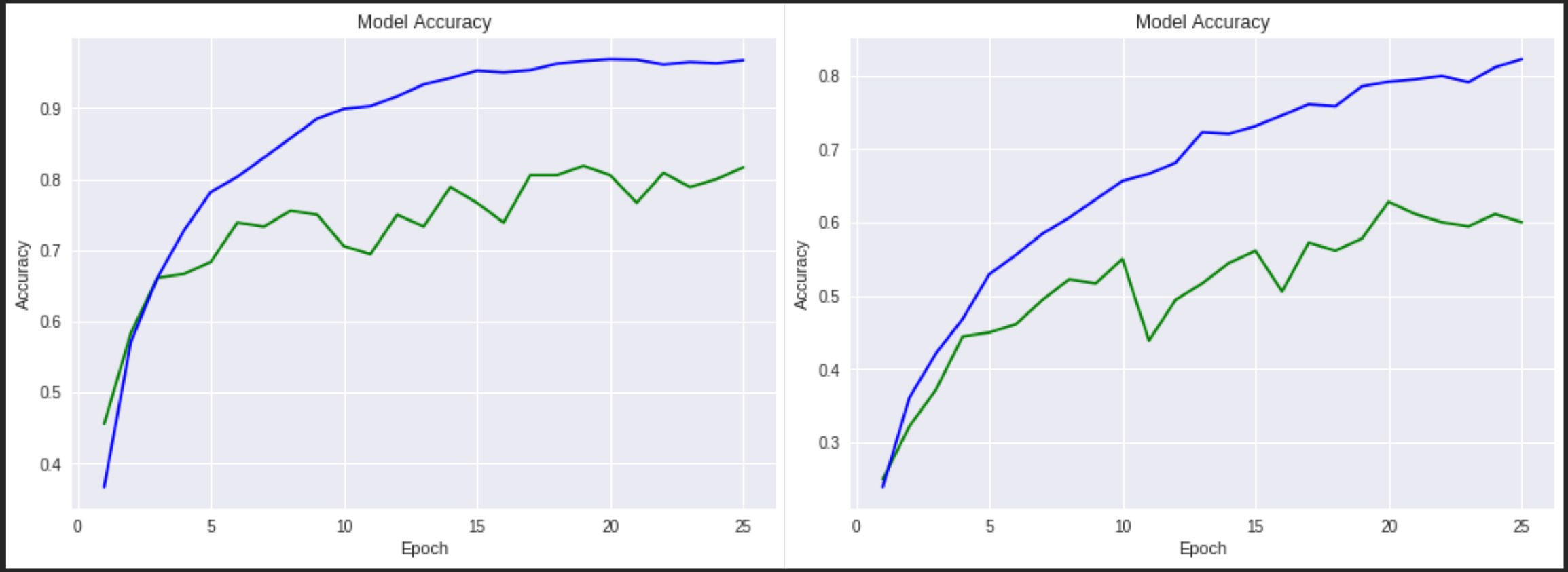
Results

- Trained Support Vector Machine with Linear Kernel achieve accuracy of 96%
- Trained Support Vector Machine with Radial Kernel achieve accuracy of 98%
- Sliding Window to detect vehicle, group detected windows and remove outlier


Results

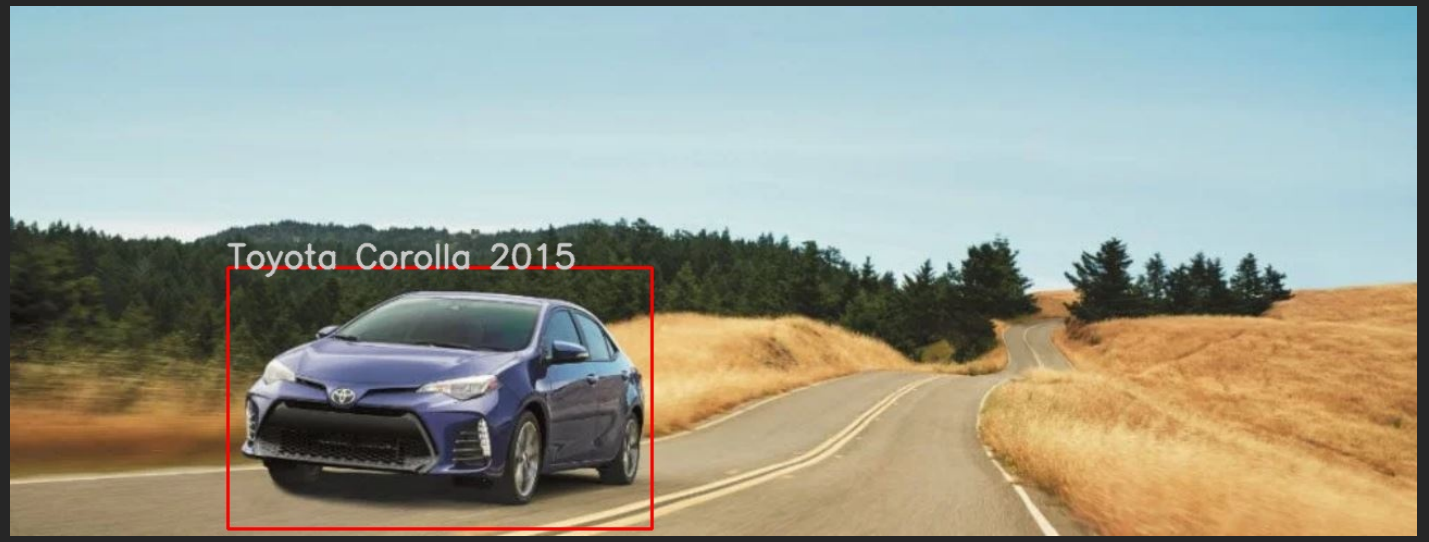
- Fine-Tuned CNN with VGG16 achieve higher accuracy compare to train-From-Scratched CNN with VGGNet Architecture.
- Classification accuracy also depends on detection accuracy (e.g if the detection result is good then classification is good vise versa)



Left : fine-tuned Train (blue) and Validation (Green) Accuracy, Right: Train From Scratch

Results

- Final Result of the whole process pipeline:



Method Comparison

- Various of Object Detection and Classification Model has been proposed and trained on the ImageNet dataset on high-level ontology (e.g classify cat, dog and human). Some of the example includes YOLO and SSD detector, where YOLO detector uses R-CNN like Neural Network.

  Link: https://pjreddie.com/darknet/yolo/

- HOG feature descriptor is widely used in face detection combine with support vector machine with similar approach as what we did in our project implementation.

  Link: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6077989

- Some of the Recognition Model uses other machine learning algorithm instead of CNN. One of the example is character recognition using KNN.

  Link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8079572

Knowledge Coverage

- **Image Processing and Feature Extraction :** Pre-process the image to HSV colour space and extract the HOG feature of the image into feature Vector. We also use SIFT feature in alternative approach of classification.

- **Support Vector Machine (Detection) :** Trained SVM with images use for detection purpose, as what we discussed in the class in lecture 4. We also experiment several different kernel function for SVM.

- **Convolutional Neural Network (Recognition) :** Trained CNN with images use for classification & recognition purpose as what we discussed in the class in lecture 14. We also experiment several different architecture in depth while training CNN.

- **Geometry:** In alternative approach, we used homography transform to find inlier of the matchings, and uses them for classification purpose.

Individual Rules

Yichong Guan :

1. Implement 'HOG_Processor.py' that process images into HOG feature vectors and experiment different parameter for the best effect.
2. Implement 'SVM_Classifier.py' and train the SVM with datasets. Experiment different kernel functions of the SVM.
3. Implement 'CNN_Classifier.py' and train the CNN with datasets. Experiment different architecture of CNN and different combination of optimizers and other parameters.
4. Some helper functions in 'utils.py'
5. Some pipelines of the main function.

Qiangyu Zheng :

1. CarFinder class in the 'Sliding_Windows.py', to find the valid windows and group them as the whole window for the entire car
2. SIFT class in the 'SIFT_Classifier.py', one experimented method to detect vehicle's type.
3. Some helper functions in 'utils.py'
4. Some pipelines of the main function.

Github link : https://github.com/EASONGUAN/DetectAndRecognize.git

To run the code, use Jupyter Notebook or Google Colab to run main.ipynb