CIS 520 Project: Segmentation and Analysis of road image

Archit Hardikar, Rohit Bhikule
University of Pennsylvania
12th December 2021

Abstract -

Road image analysis and data segmentation is important to the various applications of computer vision. With a sharp rise in autonomous vehicle industry, computer vision and perception are going to have an increased use. Thus, machine learning and more importantly Deep Learning is turning out to have a great potential. In this report, we have compared the performance of different machine learning models used for Multi-Class Image Classification from baseline deep learning neural network (DL NN) to hyper-parameter tuned Convolutional Neural Networks, and unsupervised learning methods like Auto-encoders. Besides this, a custom detection algorithm using OpenCV has been developed and compared with existing algorithms like YOLO (You Only Look Once).

Motivation –

Vehicle detection has a lot of applications including intelligent traffic management , highway traffic control, vehicle surveillance, obstacle detection/ collision aversion and autonomous driving. Because of the rampant installation of security cameras and onboard front/ back cameras, a vast database of traffic video footage and real-time vehicular data is widely available for analysis. Machine learning has wide application and potential for growth. Machine Learning and Computer Vision can act as tools which can supplement manual driving, provide stress free driving, and reduce the number of and severity of accidents. Accurate prediction and real time segmentation of the frames is important for taking decisions and maneuvering the vehicle. Lastly, improving the accuracy of existing models by tuning the hyperparameters gives better predictions.

Related Work –

In recent times, the use of deep convolutional networks (CNNs) has made great progress in the field of vehicle object detection [3]. CNNs can be used to learn image features and can perform image classification and bounding box regression [4]. The detection method can be generally divided into two categories. The two-stage method creates a bounding box around the object to be processed by using various different algorithms. After that, image classification is done by using convolutional neural networks. The other method by which this is achieved is the one-stage method. This method does not generate a candidate box.

In the two-stage method, Region-CNN (R-CNN) [5] achieves a special type of selective region search [6] in the image. A deeper structure of the neural network generally requires a long time for training. It also takes up a large amount of memory to process. Among the one-stage methods, the most important is the You Only Look Once (YOLO) [7] frameworks.

The YOLO [7] network divides the image into a fixed number of grids. Each grid is responsible for predicting objects whose center points are within the grid. YOLOv2 added the BN (Batch Normalization) layer, which makes the network normalizes the input of each layer and accelerate the network convergence speed. YOLOv2 uses a multi-scale training method to randomly select a new image size for every ten batches. Our vehicle object detection uses the YOLOv3 [2] network. Based on YOLOv2, YOLOv3 [2] uses logistic regression for the object category. The category loss method is two-class cross-entropy loss, which can handle multiple label problems for the same object. Moreover, logistic regression is used to regress the box confidence to determine if the IOU of the a priori box and the actual box is greater than 0.5. If more than one priority box satisfies the condition, only the largest prior box of the IOU is taken. In the final object prediction, YOLOv3 uses three different scales to predict the object in the image.

The traditional machine vision method has a faster speed when detecting the vehicle but does not produce a good result when the image changes in brightness, there is periodic motion in the

background, and where there are slow moving vehicles or complex scenes. Advanced CNN has achieved good results in object detection; however, CNN is sensitive to scale changes in object detection. The one stage method uses grids to predict objects, and the grid's spatial constraints make it impossible to have higher precision with the two-stage approach, especially for small objects. The two-stage method uses region of interest pooling to segment candidate regions into blocks according to given parameters, and if the candidate region is smaller than the size of the given parameters, the candidate region is padded to the size of the given parameters. In this way, the characteristic structure of a small object is destroyed, and its detection accuracy is low. The existing methods do not distinguish if large and small objects belong to the same category. The same method is used to deal with the same type of object, which will also lead to inaccurate detection. The use of image pyramids or multi-scale input images can solve the above problems, although the calculation requirements are large.

Dataset -

We are using CIFAR-10 and CIFAR-100 dataset (https://www.cs.toronto.edu/~kriz/cifar.html)

The CIFAR-10 [1] dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly selected images from each class.

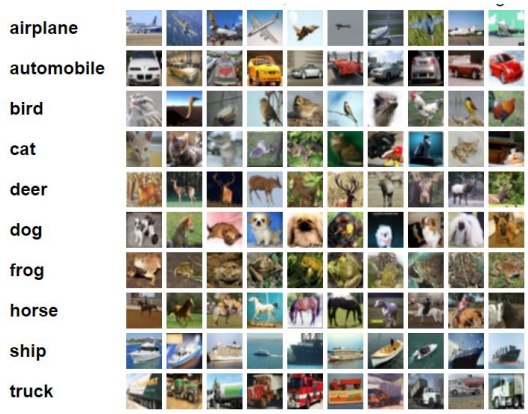

Figure 1: Ten images per class of dataset

The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

The roads have automobiles, trucks which account for 10000 images out of a total of 50000. We fit a baseline CNN model on CIFAR 100, before noting the observations and proceeding towards CIFAR 100. The results of the same will be discussed later in this report. 10000 images in CIFAR 100 account for 20% of the total data.

A more focused dataset of only a few classes (e.g automobiles, trucks, pedestrians, trees/ vegetation, roads) will improve the accuracy more, but securing such a dedicated dataset was a difficult task for training algorithms from scratch.

In CIFAR10, the classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

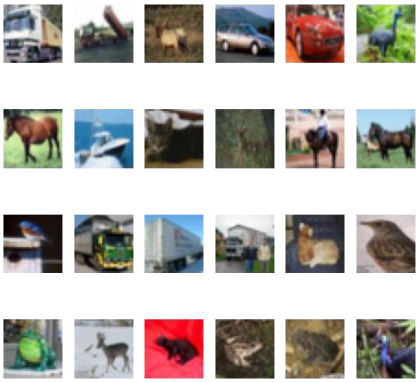

Figure 2: A sample of 24 random images from dataset

Problem Formulation -

We have formulated the task as a two part problem. 1) supervised classification problem using Baseline models, CNN, 2) Autoencoders. A lot of times, in the real world, the images taken are not clear, or they have outliers in them. At such times, Image classification doesn't perform efficiently with the

given data. In such cases, Autoencoders can be implemented to denoise the data and regenerate them. The final goal for autoencoder is to reconstruct the input image with minimum reconstruction loss. We first split our dataset into 50000 images training data and 10000 images test data. The test data was further divided into 70% test and 30% validation data. The final goal is to predict whether a real time road image contains car/ truck or not.

We preprocess the data by normalizing it. Each image is divided by a mean factor of 255 and standardize the images. All images are preprocessed to ensure that they aren't too dark/ doo bright etc. We will use the F1 score as our evaluation metric, which captures both precision and recall since we want to identify the classes correctly. Accuracy and true positives are of importance as we want to eliminate misclassifications. We will use different supervised machine learning models, as mentioned in the following section, and evaluate the models with the F1 score.

5. Methodologies–

5.1. Shallow CNN (Baseline)

The baseline model is set as the benchmark for the performance comparison. While training on CIFAR-100 [1], two Conv2D layers and four Dense layers have been used. The model is trained on CIFAR-100 dataset.

Shallow CNN converges faster. It takes lesser memory to execute.

5.2. CNN

CNN are especially useful for image recognition and image classification. A Convolutional Neural Network having Dense, MaxPooling, Convolution2D layers with their activations. It gives higher accuracy than the baseline CNN.

This has been trained on both CIFAR 100 and CIFAR 10. The results will be discussed subsequently.

5.3. CNN on grayscale images

A Convolutional Neural Network trained on grayscale data. It has layers like Dense, MaxPooling, Convolution2D layers with their activations.

5.4 Autoencoder (baseline)

A baseline model for autoencoder was created using only two convolutional layers for encoding, followed by an embedding layer and then the decoding part of the neural network.

5.5 Autoencoder

Maxpooling, Batch-normalization, Dropout was added to the above baseline model to improve the performance of the model. Skip connection was also added between the encoding model and the decoding model.

5.6 Pyramid Window Sliding technique

The best model trained above is used. The '.h5' file generated is loaded and processed further. By scaling the image and loading a window of the scaled image by using the ROI (Region of Interest), it is possible to run predictions on the window to check for matches. The window slides over the image through X and Y direction and identifies potential positive matches.

6. Experiments and Results, Discussions, Conclusions –

We evaluate the performance of all of the classification algorithms using the methods above. We tune the hyperparameters for each method and improve the accuracy.

6.1 Metric-

We choose F1 score as our evaluation metric across models because it captures both precision and recall.

$$F1 = \frac{2TP}{2TP + FP + FN}$$

In our case, TP means we correctly identify a image to what it actually is, FP means we identify it incorrectly as a positive. FN means we fail to identify an image correctly.

6.2. Hyperparameter Tuning

Before using the model and deploying it in the computer vision problem for identifying objects on the road, it is necessary to tune the hyperparameters.

6.2.1 Convolutional Neural Networks –

1) Initial shallow neural network trained on CIFAR 100 had the following structure-

```
model = Sequential()
model.add(Conv2D(input_shape=(32, 32, 3), kernel_size=(3, 3),
padding='same', strides=(2, 2), filters=32))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(1, 1),
padding='same'))
model.add(Conv2D(kernel_size=(3, 3), padding='same', strides=(2,
2), filters=40,activation='relu'))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(1, 1),
padding='same'))
model.add(Flatten())
model.add(Dense(180, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(150, activation='softmax'))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(100, activation='softmax'))

model.summary()
model.compile(loss='sparse_categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
```

The model yielded following accuracy curves.

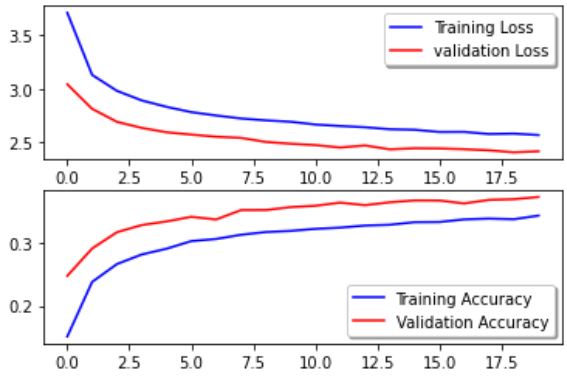

Figure 3: Accuracy curve

The train accuracy achieved after 20 epochs was approximately 37%. The main reason behind this is because of presence of 100 classes. We observed that the presence of large number of classes introduces such a large variance in the data.

2) Training the same shallow network on CIFAR10 dataset (which has 10 super-classes instead of 100 classes) gives us an accuracy close to 70% on 20 epochs.

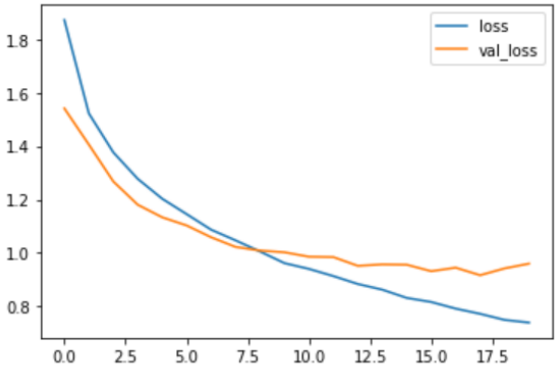

Figure 4: Validation loss and Train Loss

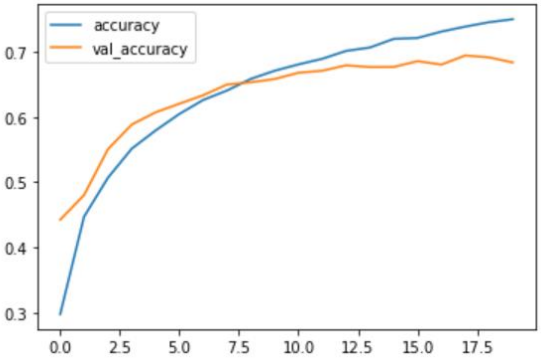

Figure 5: Validation error and Training error

	precision	recall	f1-score	support
0	0.67	0.76	0.71	1000
1	0.79	0.83	0.81	1000
2	0.53	0.65	0.58	1000
3	0.51	0.47	0.49	1000
4	0.59	0.68	0.63	1000
5	0.69	0.45	0.55	1000
6	0.76	0.77	0.76	1000
7	0.82	0.69	0.75	1000
8	0.72	0.86	0.78	1000
9	0.83	0.67	0.74	1000
accuracy			0.68	10000
macro avg	0.69	0.68	0.68	10000
weighted avg	0.69	0.68	0.68	10000

Figure 6: Precision vs Recall

```
array([[763,  18,  59,   8,  16,   2,  10,   4,  96,  24],
       [ 30, 834,  12,   6,   5,   2,   9,   3,  58,  41],
       [ 78,  11, 653,  46,  86,  36,  48,  20,  14,   8],
       [ 37,  17, 111, 466, 111, 110,  67,  24,  37,  20],
       [ 42,   7, 101,  49, 680,  12,  50,  39,  15,   5],
       [ 25,   9, 124, 216,  74, 452,  33,  40,  18,   9],
       [  7,   7,  78,  52,  56,  11, 767,   6,  13,   3],
       [ 27,   6,  62,  40, 118,  24,   6, 686,   9,  22],
       [ 69,  22,  19,   9,   4,   2,   3,   3, 859,  10],
       [ 58, 125,  22,  17,   9,   3,  14,   8,  70, 674]])
```

Figure 7: Confusion matrix

3) Training a deep neural network significantly improves the accuracy. The model starts performing significantly better and yields better results.

Following is the architecture used for the model-

```
model = Sequential()
model=keras.models.Sequential()
model.add(keras.layers.Conv2D(filters=128,kernel_size=(3,3),activ
ation='relu'))
model.add(keras.layers.BatchNormalization(momentum=0.8))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))
model.add(keras.layers.Conv2D(filters=64,kernel_size=(3,3),activa
tion='relu'))
model.add(keras.layers.BatchNormalization(momentum=0.8))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))\
model.add(keras.layers.Conv2D(filters=32,kernel_size=(3,3),activa
tion='relu'))
model.add(keras.layers.BatchNormalization(momentum=0.8))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(512,activation='softmax'))
model.add(keras.layers.Dense(10,activation='sigmoid'))
adam = optimizers.Adam(lr=0.001)
model.compile(loss='categorical_crossentropy',
              optimizer=adam,
              metrics=['accuracy'])
```

The model yielded a train accuracy of 78% in 20 epochs with signs of sharp accuracy increase. On training till 30 epochs, the train accuracy improves to 83.24%.
Following are the curves obtained for the same.

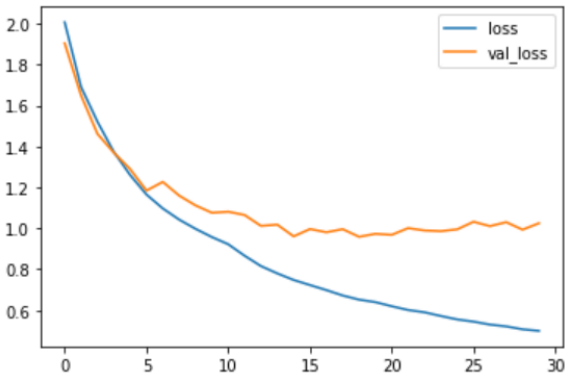

Figure 8: Loss curve

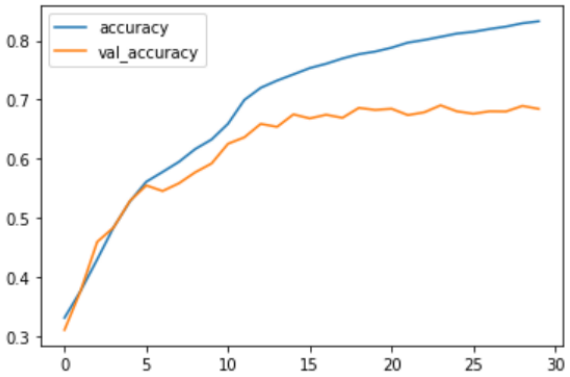

Figure 9: Accuracy curve

Making the neural net deep is able to extract all of the features from the 10 classes from the dataset. In addition to that, the architecture does not have dropouts or early stopping. Because there are close to 50000 images and 10 classes, the data has a lot of variance. Thus there is a low chance of the algorithm overfitting on 30 epochs. Also, the optimizer has been set to adam to help converge faster. SGD performs slower as compared to Adam.
Learning rate has been set to 0.001 or 1e-3. This learning rate is neither too fast, nor too slow. By hyperparameter tuning, it has been determined to be one of the optimal values for the given architecture.

	precision	recall	f1-score	support
0	0.78	0.71	0.74	1000
1	0.84	0.84	0.84	1000
2	0.55	0.65	0.59	1000
3	0.49	0.41	0.44	1000
4	0.71	0.64	0.67	1000
5	0.54	0.70	0.61	1000
6	0.79	0.77	0.78	1000
7	0.74	0.75	0.75	1000
8	0.82	0.81	0.82	1000
9	0.86	0.73	0.79	1000
accuracy			0.70	10000
macro avg	0.71	0.70	0.70	10000
weighted avg	0.71	0.70	0.70	10000

Figure 10: Precision vs Recall

```
array([[712,  16, 107,  25,   9,  13,  12,  11,  83,  12],
       [ 15, 841,  10,  12,   4,   8,  12,   8,  17,  73],
       [ 42,   3, 650,  60,  75,  75,  50,  28,  11,   6],
       [  7,   6,  97, 409,  39, 335,  50,  38,  11,   8],
       [  9,   0, 118,  59, 639,  47,  39,  83,   6,   0],
       [  6,   2,  72, 105,  38, 703,  17,  54,   2,   1],
       [  4,   1,  60,  86,  37,  24, 772,  11,   5,   0],
       [ 17,   1,  42,  36,  53,  88,   3, 751,   3,   6],
       [ 75,  24,  19,  18,   7,   6,  12,   7, 814,  18],
       [ 29, 107,  15,  29,   5,   9,   7,  22,  42, 735]])
```

Figure 11: Confusion matrix

6.2.2 Autoencoder

1) Autoencoder baseline model

Only two convolutional layers were used in the encoding part and subsequently two in the decoding part to train the autoencoder. The model was trained on 20 epochs and the following loss curve was obtained for the classification.

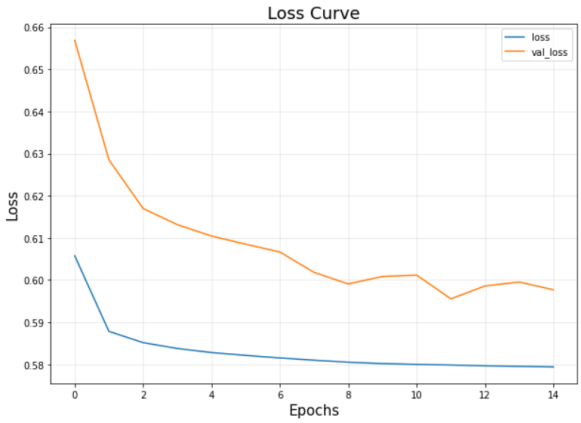

Figure 12: Loss curve for Autoencoder baseline model

2) Denoising Autoencoder

Three convolutional layers with Maxpool, dropout and batch normalization were used. Skip connection was also used between the encoding and the decoding layer.

Skip connections are used in convolutional and deconvolutional networks so that it restores the pieces of information which gets lost during training of the model.

As this is a denoising model, we have added gaussian noise to the input images, that is, thirty percent of the image array was randomized. Dropout of 30 percent was found as the optimal parameter. The size of the embedding layer was also varied and 128 was found to be the optimal size for it.

The model was trained for 20 epochs and following loss curve was obtained.

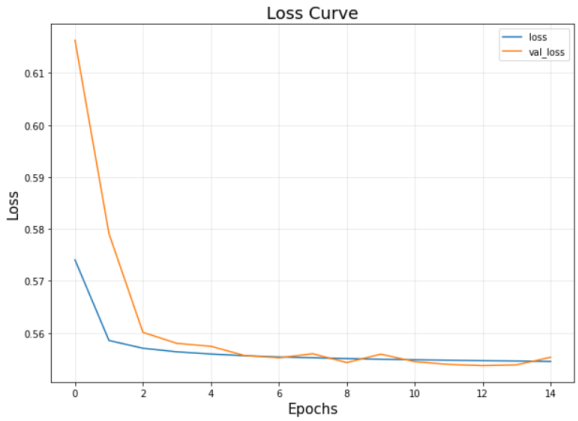

Figure 13: Loss curve for denoising autoencoder

The final comparison with the noisy images and the reconstructed images is given below. The first layer is the images with thirty percent gaussian noise. The bottom layer is the output reconstructed images.

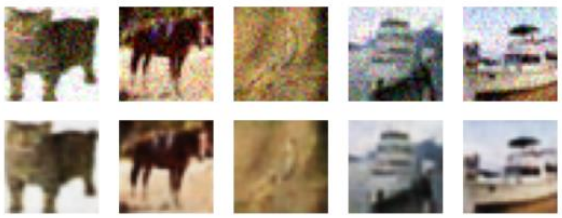

Figure 14: Comparison between noisy and denoised images

7 Sliding Window Computer Vision –

Similar to the YOLO[8] implementation discussed earlier in this report, a sliding window image classification algorithm has been developed to detect cars.

The algorithm takes an image as a input and outputs a processed image with high density rectangles in areas with cars. Refer to the image shown below.

Figure 15: Input image

The sliding window technique runs on the algorithm and divides the image into small parts. It runs the prediction on each image. Detection accuracy of cars was found to be around 94-97%.

The model used was the Deep Convolutional Neural Network mentioned earlier.
The output image is displayed below.

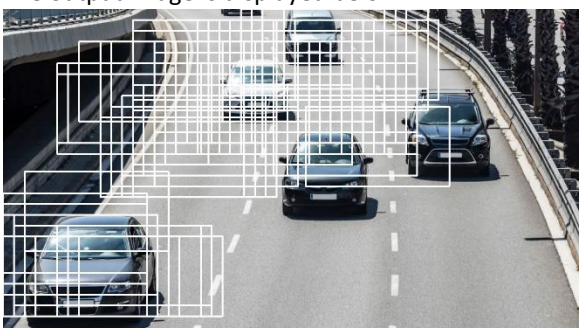

Figure 16: Output

The experimentally found regions where cars exist is demonstrated by high density rectangles. Each individual rectangle can be thought of as an instance where the algorithm detected a car as it moved further along the X and Y axes respectively. The top left region of the picture can be seen to have a few rectangles existing which don't have a car actually. This is mainly because of the FP (False Positivity).
As seen above, the precision and recall both are quite low. Having a 70% value means theres a higher chance of having a wrong classification.
Another challenge is that the dimensions of the images in CIFAR10 is 32*32*3. There is limited data and information stored in these images. Because they have been created for an academic purpose, their commercial use in such algorithms is very limited.

Further improvements in the algorithm have been discussed in the future scope.

Future Work –

As this is a largely research oriented field, a lot of work is being done in the area. An improvement that can be done to this project can be to create a dedicated dataset of pedestrians, trees and cars/ vehicles. Training the model on a smaller number of classes will reduce the variance (as seen in case of CIFAR 100 vs CIFAR 10 model train accuracies).
Another improvement is to improve the model architecture by making the network more deep and running the algorithm to 100 epochs. Decreasing the batch size will also improve the accuracy.
Decreasing the False Positivity rate will help increase the assurance of getting a correct classification. If the algorithm were to be used in a car obstacle detection/ onboard computer vision driving assist system, adding functionalities of signal detection, pedestrian detection and road paint/ road directions would be of utmost importance.
Removing multiple windows in the output image would help aide real time tracking of vehicles on the road.

Acknowledgments - We appreciate the assistance and guidance from Prof. Lyle Ungar and every teaching assistant throughout the semester.

References -

1] https://www.cs.toronto.edu/~kriz/cifar.html

2]https://etrr.springeropen.com/articles/10.1186/s12544-019-0390-4

3] https://en.wikipedia.org/wiki/Computer_vision

4] Zhao, Z.Q., Zheng, P., Xu, S.T., Wu, X. (2018). Object detection with deep learning: A review. arXiv e-prints, arXiv:1807.05511.

5] Girshick, R., Donahue, J., Darrell, T., Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In 2014 IEEE Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2014.81. IEEE.

6] Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders, A.W.M. (2013). Selective search for object recognition. International Journal of Computer Vision, 104(2), 154–171.

7] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You only look once: Unified, real-time object detection. In 2016 IEEE conference on computer vision and pattern recognition. https://doi.org/10.1109/cvpr.2016.91. IEEE, (pp. 779–788).

8] https://towardsdatascience.com/yolo-you-only-look-once-17f9280a47b0

9] Jiao Mao, Chunhua Shen, Yu-Bin Yang ,Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections Xiao-