Data Mining and Predictive Analytics (BUDT758T)

Project Title: Car Image Detection using Machine Learning

Team Members:

Sree Pradyumna Davuloori, LakshmiNarasimhan Narayanan, Sathya Anurag Siruguppa, Rohith V. Thomas, Navneet Sonak

ORIGINAL WORK STATEMENT

We the undersigned certify that the actual composition of this proposal was done by us and is original work.

	Typed Name	Signature
Contact Author	LAKSHMINARASIMHAN NARAYANAN	
	SREE PRADYUMNA DAVULOORI	
	SATHYA SIRUGUPPA	
	NAVNEET SONAK	
	ROHITH THOMAS	

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The primary goal of this project was to build an accurate and reliable car image detection model. Through the course of this endeavor we explored and understood the effectiveness of different machine learning algorithms on image data. We fitted various conventional models after some mandatory data preprocessing of the image dataset. By employing a dimensionality reduction technique, we were able to reduce the time and complexity of models and ultimately improve their performance.

II. DATA DESCRIPTION

The car image dataset was prepared by a few researchers at the University of Illinois Urbana-Champaign. All images consisted of side-views of various cars of different brands, sizes and colors. There were 1050 images in total of which 550 contained cars and the rest did not. The images were in grayscale (.pgm format) with a resolution of 40x100. Two images from the dataset are displayed below:

Dataset: https://cogcomp.cs.illinois.edu/Data/Car/

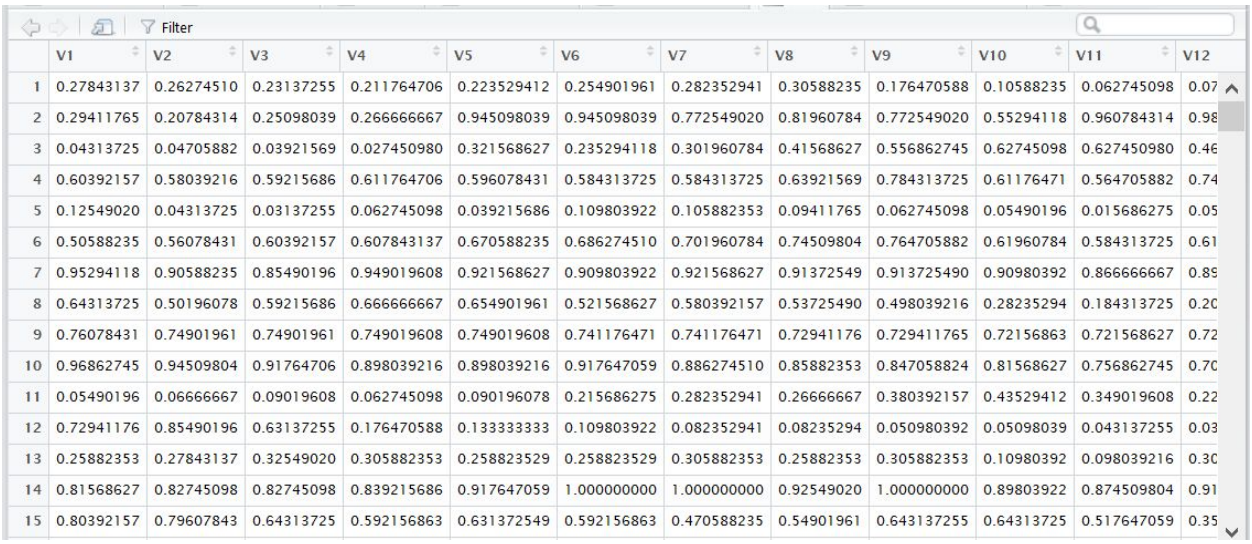
Few lines of Data

1050 rows of car images in pixel data.

4000 columns of pixel values.

Training data set: 80% of total set (840 images)

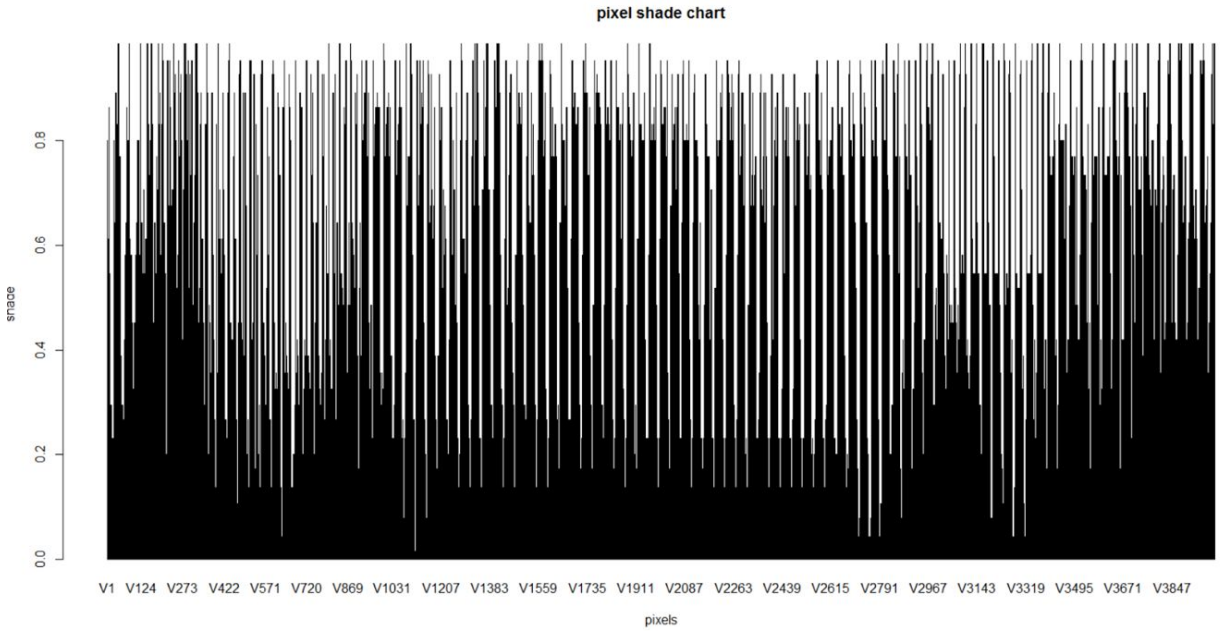
Testing data set: 20% of total set (210 images)



	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
1	0.27843137	0.26274510	0.23137255	0.211764706	0.223529412	0.254901961	0.282352941	0.30588235	0.176470588	0.10588235	0.062745098	0.07
2	0.29411765	0.20784314	0.25098039	0.266666667	0.945098039	0.945098039	0.772549020	0.81960784	0.772549020	0.55294118	0.960784314	0.98
3	0.04313725	0.04705882	0.03921569	0.027450980	0.321568627	0.235294118	0.301960784	0.41568627	0.556862745	0.62745098	0.627450980	0.46
4	0.60392157	0.58039216	0.59215686	0.611764706	0.596078431	0.584313725	0.584313725	0.63921569	0.784313725	0.61176471	0.564705882	0.74
5	0.12549020	0.04313725	0.03137255	0.062745098	0.039215686	0.109803922	0.105882353	0.09411765	0.062745098	0.05490196	0.015686275	0.05
6	0.50588235	0.56078431	0.60392157	0.607843137	0.670588235	0.686274510	0.701960784	0.74509804	0.764705882	0.61960784	0.584313725	0.61
7	0.95294118	0.90588235	0.85490196	0.949019608	0.921568627	0.909803922	0.921568627	0.91372549	0.913725490	0.90980392	0.866666667	0.89
8	0.64313725	0.50196078	0.59215686	0.666666667	0.654901961	0.521568627	0.580392157	0.53725490	0.498039216	0.28235294	0.184313725	0.20
9	0.76078431	0.74901961	0.74901961	0.749019608	0.749019608	0.741176471	0.741176471	0.72941176	0.729411765	0.72156863	0.721568627	0.72
10	0.96862745	0.94509804	0.91764706	0.898039216	0.898039216	0.917647059	0.886274510	0.85882353	0.847058824	0.81568627	0.756862745	0.70
11	0.05490196	0.06666667	0.09019608	0.062745098	0.090196078	0.215686275	0.282352941	0.26666667	0.380392157	0.43529412	0.349019608	0.22
12	0.72941176	0.85490196	0.63137255	0.176470588	0.133333333	0.109803922	0.082352941	0.08235294	0.050980392	0.05098039	0.043137255	0.03
13	0.25882353	0.27843137	0.32549020	0.305882353	0.258823529	0.258823529	0.305882353	0.25882353	0.305882353	0.10980392	0.098039216	0.30
14	0.81568627	0.82745098	0.82745098	0.839215686	0.917647059	1.000000000	1.000000000	0.92549020	1.000000000	0.89803922	0.874509804	0.91
15	0.80392157	0.79607843	0.64313725	0.592156863	0.631372549	0.592156863	0.470588235	0.54901961	0.643137255	0.64313725	0.517647059	0.35

III. DATA PREPARATION AND EXPLORATION

Initially each image was described by a matrix of size 40x100. We flattened out this matrix (row-wise) so that we have 4000 features(columns) that describe an image(row). For any image processing problem, the first and foremost exploration technique used is to plot the pixel values against the spatial arrangement of pixels. So the dataset contains 4000 features and 1050 rows.



Typically, for a text image, such a plot would give peaks followed by dips which characterize the presence of text(peaks) among the background (dips). Hence it would result in a cyclic multi-modal distribution. However, in this case we were unable to get such a definite pattern. The main reason for this could be due to the various colors which the cars possess and because of this we cannot define the area of the image to be containing a car or not based on the pixel values.

IV. RESEARCH QUESTION

Image classification is one of the biggest machine learning challenges and with the advancement in technology and faster processors it is being solved at a very brisk pace. We as a team wanted to explore working with unstructured data and analyze the effect of machine learning algorithms on image classification.

We chose this dataset to demonstrate the power of object detection which has endless applications in healthcare, e-commerce, manufacturing and transportation. With the advent of driverless cars/ self drive cars, object detection has put this sub section of the automotive industry to the front in this decade.

Successful object detection and identification by classifying each object found on the road as a building, vehicle or human will enable the safe implementation of self driving cars. A lot of research is going on this industry which can be seen in the form of google driverless cars and tesla motors. One important point to keep in mind is that richer the dataset of cars, the better is the identification and classification. Our findings are the result of our investigation into the basic and most crude development that these state-of-the-art technologies will build on.

V. MACHINE LEARNING MODELS

The models we considered for performing image classification are:

1. Logistic Regression
2. Random Forest
3. XGBoost

VI. RESULTS AND FINDING

Logistic Regression accuracy is very less it is 0.47, whereas the rest of the models have accuracies higher than this. It is because of curse of dimensionality. Logistic regression breaks down when the number of features is more than number of rows. Linear models partition the space into a single linear plane.

Mathematically, X'X becomes non-invertible when number of features is more than number of rows. It can also be viewed as in high dimensional spaces there is more than one plane that can be fitted to data, so the model does not find the optimal parameters and thus performs poorly. In these kind of cases we need to resort to regularization or dimensionality reduction.

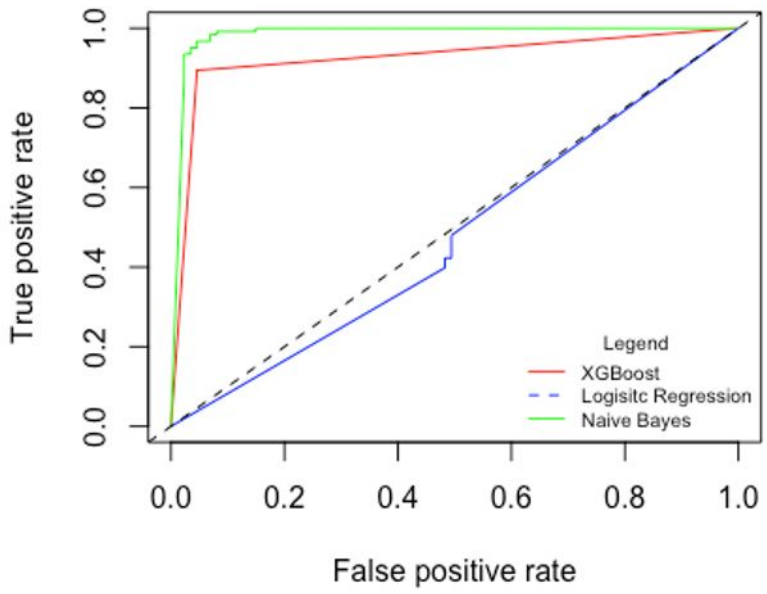
$$\hat{\beta} = (X'X)^{-1}X'y$$

ACCURACY SENSITIVITY AND SPECIFICITY

Model	Accuracy	Sensitivity	Specificity

Logistic Regression	0.47	0.44	0.51
Naive Bayes	0.92	0.97	0.89
XGBoost	0.92	0.87	0.97

ROC CURVES



In the ROC curve we can see that Naive Bayes and Random forest are doing better than the baseline model. But Logistic regression is performing poor than baseline model. Also, from the ROC curve we can see that 0.8 is the optimal cut-off for both Naive Bayes and Random forest. And taking this cut-off value will certainly increase accuracy of both the models.

PRINCIPAL COMPONENT ANALYSIS

The dataset is built in such a way that there are 4000 features, corresponding to each pixel of the image and 1050 observations corresponding to each image. Number of predictors are much larger than the number of observations, this is one of the reason why the logistic regression performs very poorly, giving an accuracy less than 50% which is worse than guessing randomly whether the image is a car or not. Logistic regression breaks when there are more predictors than there are observations.

The naïve bayes model and the XGBoost model perform quite well in giving a high accuracy, but take a lot of time to run. We also got thinking as to what can be done to reduce the number of predictors to a reasonable amount, somewhere closer to the number of observations.

We considered feature selection and dimensionality reduction for the purpose. Feature selection did not feel like a good to execute for this context. When working with image data, no pixel is unimportant and it is not a great to idea to narrow down to a few pixels which describe data. For example, some feature selection technique may narrow down to a list of pixels belonging to various areas in the car which best allow the model to identify whether it is a car or not. But for an unseen image, difference between colours of background and the car itself may not be high. For this reason, all the pixels of the image are important and we did not want to lose any of the predictors, which happens when we perform feature selection and narrow down the list of predictors.

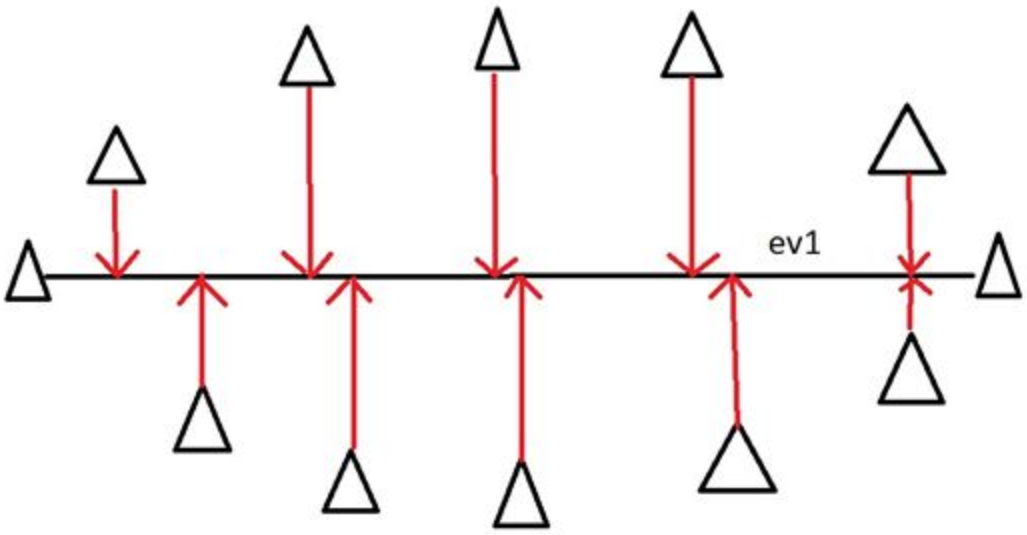
Having eliminated feature selection, we felt a dimensionality reduction technique like Principal Component Analysis was the right technique to use for many reasons. Some of the reasons we considered PCA include:

- Curse of dimensionality: With over 4000 features and only a quarter of that number of observations, we were struck by the curse of dimensionality. This occurs when the dimensional space of the features is so huge that any machine learning model will face difficulties dividing the features space distinctly enough so as to accurately predict the categorical target variable. Classifiers struggle to estimate decision boundaries as the feature space is very sparse.
- Constructing a correlation matrix showed that there were over 1000 features that had a correlation greater than 0.90. This meant that most of these features will not make significant contributions to the predictive power of any classifier and therefore do not add much value individually.
- Saving storage space and time of computation was also an important reason why we decided to go for PCA. Optimizing the time and storage complexity was important in adding business value to the problem we were tackling.

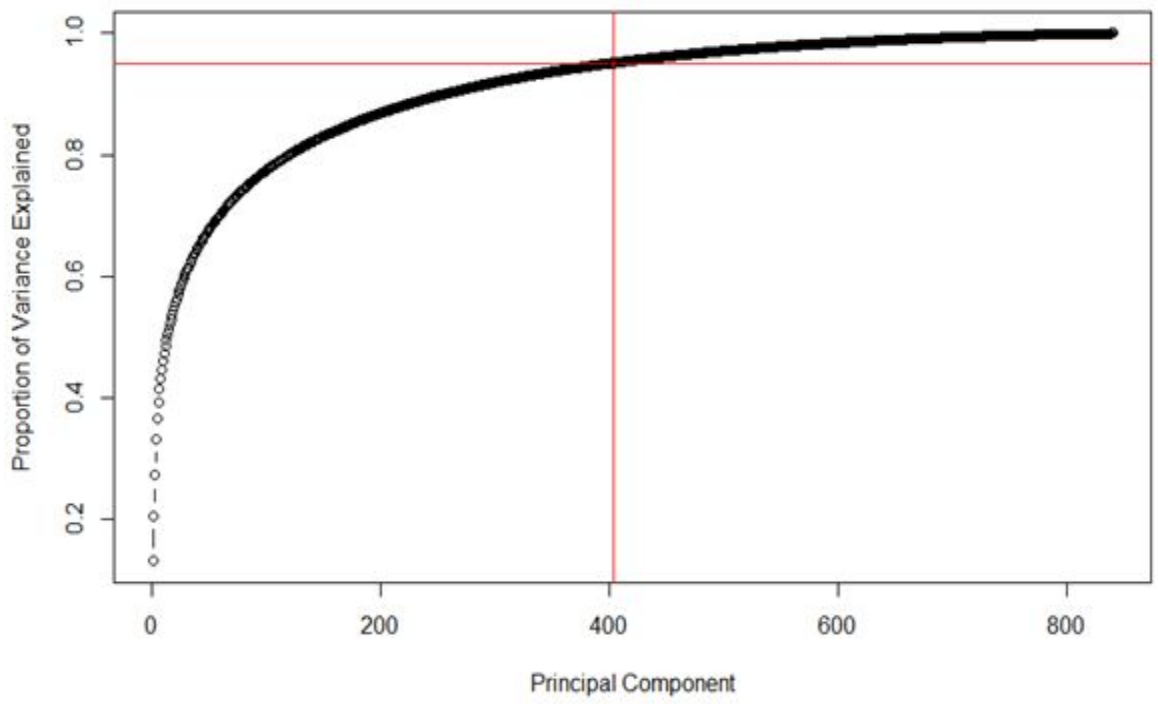
Brief overview of PCA:

The main idea behind PCA is to find a low dimensional set of axes that accurately summarize the data. It allows us to remove the redundancy introduced by highly correlated features without losing any information provided by them. It also helps tackle the problem of curse of

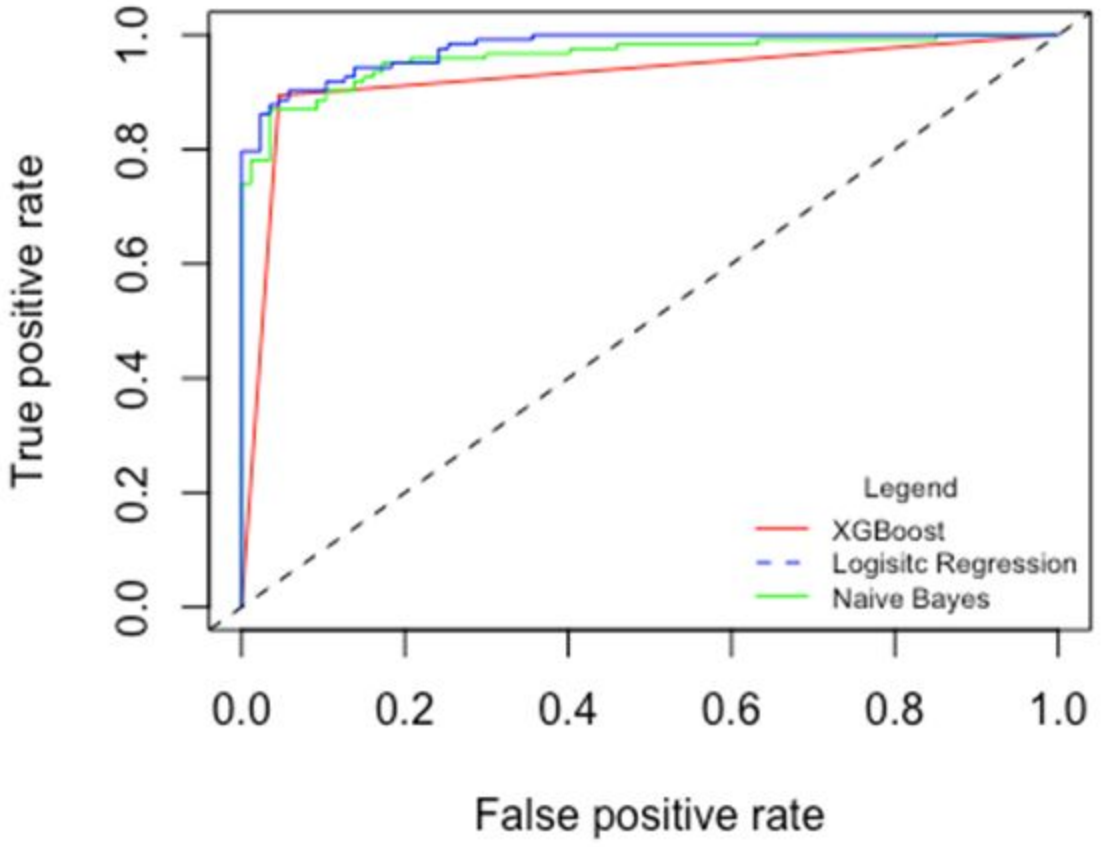
dimensionality and a high dimensional feature space. PCA tries to find a set of axes, called components, that are generally a linear combination of the already existing features, and these components explain a higher amount of variance in the target variable compared to that explained by these features individually. These components often do not have any explanatory meaning to us but contribute a lot in explaining the target variable.



The above image explains PCA succinctly. Consider each of the triangles to be one point. Projecting them onto the horizontal line means that data is very spread out and a large amount of variance could be explained. Principal component 1(PCA1) is always the component that explains the maximum amount of variance. PCA is not be confused with feature selection. The only similarity between them is that they allow us to narrow down the set of predictors used, the similarity ends there. PCA constructs a whole new set of features from the existing features, whereas feature selection just finds the most important ones and eliminates the others.



The graph shows the amount of variance explained versus the number of components. We chose 404 components as the optimal point. With 404 components, roughly 10% of the original number of features, we could still explain 95% of the variance in the target variable. This meant that we reduced the space and time complexity by almost 90%, found better features which have great relevance and still managed to explain 95% variance.



ROC Curve for model performance after implementing PCA

Accuracy of Logistic regression has improved drastically after PCA

Accuracy of XGBoost has also increased by considerable amount after PCA.

Accuracy, Sensitivity and Specificity

Model	**Accuracy**	**Sensitivity**	**Specificity**
Logistic Regression	0.91	0.87	0.94

Naive Bayes	0.81	0.59	0.96
XGBoost	0.94	0.91	0.97

DEEP LEARNING

As we saw earlier PCA improved accuracy considerably, but we were in the quest of increasing the accuracy of our models even more. When researching about this problem, we found a lot of literature has been written on the effectiveness of deep learning models for image datasets. So, we started exploring various deep learning algorithms and how to implement them. All the deep learning algorithms were implemented in Tensorflow in Python.

Feed forward Neural Network with Backpropagation:

A neural net has 3 layers namely the input layer, hidden layer and the output layer. In our algorithm we used 3 hidden layers. Input layer had 4000 input layer nodes, each hidden layer has 3000 hidden layer nodes and the output layer has 2 nodes. We used 2 nodes because of one-hot encoding. 1 epoch consists of a feed forward pass and a back propagation pass. We used 15 epochs totally. The cost function used at the output layer is Softmax cross entropy with logits and the optimization function used is Adam Optimizer. The training error rate goes to zero around 7 epochs. Accuracy of the model on the test set is 0.976 and the time taken by the model is 53.6s.

Recurrent Neural Network

In recurrent neural network the output of the previous hidden layer is passed as one of the inputs to the next hidden layer. This allows us to recognize patterns which were not visible earlier. Let's take an example : consider the word “HURRY”. H is followed by U, U is followed by R, R is followed by R, R is followed by Y. If asked what follows R it could be R or Y. But if we know that R was preceded by R, then the letter that follows R is Y. This is what recurrent neural networks helps us to achieve.

But there is one problem. The weights from previous hidden layers will keep on multiplying as we add more and more hidden layers, since the weights are multiplied with each other they could explode to infinity or vanish to zero. To overcome this problem we use LSTM (Long short

term memory) cells. When the weights come as input instead of the multiplicative process, an additive process is used.
So we used LSTM cells for the above purpose and similar to the previous neural network we used Softmax cross entropy with logits for Cost function and Adam optimizer as an optimization function. We ran it for totally 50 epochs and the training error did not go to zero even at the end of 50 epochs. Accuracy of this model was slightly on the lower side, it was around 0.924 whereas the time taken for the algorithm to run was mere 7.67s.

CONVOLUTIONAL NEURAL NETWORK (CNN)

Convolutional neural network also known as ConvNet is a type of feed forward artificial network. In case two-dimensional data like images, the input is in terms of patches. Each of the neurons works on patches of data. Each neuron identifies one of the features. Early layers of the neural net catch features like edge detection and in the successive layers in the case of object detection, the neurons of the network capture more complex data like shapes, texture and color.

Inception model

In image detection, the richer a dataset, the better are the results. To test Convolutional neural network, we used a pre-trained CNN model called inception model.
Inception model was developed by google using 100,000 images in 1000 distinct categories. But inception model is not specifically trained for side profile cars, which we specifically want to detect using this data set. Hence, we used inception model as our base, which is already trained to detect edges, shapes, texture and color.

Transfer learning

Transfer learning focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.
Modern object recognition models have millions of parameters and can take lot of time (sometimes even weeks) to completely train datasets. Transfer learning is a technique that bypasses a lot of this work by taking a fully-trained model for a set of categories like ImageNet, and retrains from the existing weights for new classes. In this example we'll be retraining the final layer from scratch with the image dataset of 1050 car images that we collected from UIUC (University of Illinois Urbana Champaign), while leaving all the others untouched.

Tensorflow

tf.Session - A session object encapsulates the environment in which Operation objects are executed and tensor objects are evaluated. Tf.session was used as a function to run a test image as an input to the model.
Bottleneck is an informal term used to refer to the topmost layer before classification is done. Because every image is reused multiple times during training and calculating each bottleneck takes a significant amount of time, it speeds things up to cache these bottleneck values on disk so they don't have to be repeatedly recalculated. By default, they're stored in the /tmp/bottleneck directory, and if you rerun the script they'll be reused so you don't have to wait for this part again.

Docker container

A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings. Available for both Linux and Windows based apps, containerized software will always run the same, regardless of the environment. Containers isolate software from its surroundings, for example differences between development and staging environments and help reduce conflicts between teams running different software on the same infrastructure.
We deployed the tensorflow inception model in a docker container.

Accuracy

The model was trained using the training image dataset and a small split from the same dataset was used as testing to calculate accuracy.
The inception model along with transfer learning was used for comparison rather than testing the performance of the model, as the base model.
Accuracy: ~97.5%

Commands used to run application on docker

docker run -it -v ~/tf_files:/tf_files gcr.io/tensorflow/tensorflow:latest-devel

python tensorflow/examples/image_retraining/retrain.py

--bottleneck_dir=/tf_files/bottlenecks
--model_dir=/tf_files/inception
--output_graph=/tf_files/retrained_graph.pb
--output_labels=/tf_files/retrained_labels.txt
--image_dir /tf_files/cars

python tf_files/label_image.py tf_files/example_image.jpg

VI. APPLICATIONS

1. Medicine

Medical imaging has a strong need for image detection. There is a strong need to not only automate the analysis, but also use and implement machine learning algorithms and deep learning techniques to automatically classify images faster and more accurately. Here we classified images based on object detection. Same logic can be applied for reading scan images and classifying particular patterns to be tumorous or not.

2. Transportation

Self-drive cars has become the buzzword of this decade. For these cars to be functional they need to be able to detect cars and objects to successfully navigate through roads. The need for objection and classification is high in this field and hence machine learning techniques demonstrated in this project are very useful for the same.

3. Ecommerce

Many leading companies in e-commerce like amazon and walmart are always in the look out of new technology for improving the shopping experience. Object detection can take open up new doors in the field of ecommerce.

Google recently introduced the ability to search for images by comparing them to others. By uploading an image or giving Google an image URL, it will show you where that image is used on the Web. During the recent London riots, a rumor spread on Twitter that a tiger had been set loose from London zoo. Searching using Google's tool revealed that the supposed photographic evidence came from a 2008 tiger escape in Italy and the streets of London were

free of wild animal. Image recognition technology can be also used in mobile applications to identify specific products.

4. Agriculture and manufacturing

As an example in Japan, a farmer used image detection to correctly identify the particular crop he wanted to harvest. Through this example, we would like to point out that lot of labor work can be automated through application of image detection and object detection. This will successfully reduce dependency on humans work and thus increase the efficiency and accuracy through the implementation of machine learning techniques.

VII. REFERENCES

1. Image Dataset from UIUC: https://cogcomp.cs.illinois.edu/Data/Car/
2. https://www.tensorflow.org/tutorials/image_recognition
3. https://arxiv.org/abs/1512.00567
4. https://github.com/tensorflow/tensorflow/issues/6589
5. http://sebastianruder.com/transfer-learning/index.html#usingpretrainedcnnfeatures
6. http://colah.github.io/posts/2014-07-Conv-Nets-Modular/
7. https://www.wikiwand.com/en/Logistic_regression
8. https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/
9. https://www.extremetech.com/extreme/189486-how-googles-self-driving-cars-detect-and-avoid-obstacles
10. https://www.wikiwand.com/en/Xgboost