Machine Learning: For Offending Drivers

Sapna Kapoor
Technical Director, NIC

Abstract
A serious and ever-growing threat, the distracted driving is an important element of any road safety plan. The evidence from various surveys and studies across the globe, clearly reflects that driver distraction is amenable to intervention and can be effectively dealt with. Unfortunately, in India, there is no data that is being currently captured at the crash site by authorities to record the connection between mobile phone usage and crashes. Some challenges impeding this issue include lack of sufficient data, insufficient convergence between key authorities and ineffectual enforcement.
The use of mobile phones while driving causes four types of mutually non-exclusive distractions; i.e visual, auditory, cognitive and manual/physical. While visual distractions cause drivers to look away from the roadway, manual distractions requires the driver to take their hands off the steering wheel, auditory distractions mask those sounds that are crucial for the driver to hear while driving and cognitive ones induce the driver to think about something other than driving. It has been established that distraction caused by mobile phone usage while driving, can deprecate driving performance, and have now joined alcohol and speeding as leading factors in fatal and serious injury crashes.
This paper strives to improve these alarming statistics, by testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviors. If we can capture the pictures of offending drivers in car seat (texting, eating, talking on the phone, makeup, reaching behind, etc), the goal is to predict the likelihood of what the driver is doing in each picture.

Keywords
Machine Learning, Distracted Driving, Image Detection, Semantic Segmentation

I. Introduction

According to a global survey, one in five car accidents is caused by a distracted driver. However, practices and models in some countries have not only yielded results, these can also be contextualized in the Indian super structure to bolster prevention efforts. Distracted drivers are about four times as likely to be involved in crashes as those who are focused on driving. Drivers who think they can multi-task are fooling themselves as a research shows 98 per cent are unable to divide their time without affecting performance.

From the dataset of 2D dash-board camera images, the challenge lies in to classifying each driver's behavior. The pictures are captured and the behavior of the driver is predicted by processing the images. Broadly we have classified them in following 10 classes:

1. Class-00: safe driving
2. Class-01: texting - right
3. Class-02: talking on the phone- right
4. Class-03: texting - left
5. Class-04: talking on the phone - left
6. Class-05: operating the radio
7. Class-06: drinking
8. Class-07: reaching behind
9. Class-08: hair and makeup
10. Class-09: talking to passenger

II. Capture the Image

Case : Driver Using mobile in left hand (Class-05)

Fig. 1: Offending Driver with Mobile

III. Approach

A. Pre-processing of Images

1. Image Depth

Counterintuitively, the model works better for grayscale images when compared to colour images. Hence, the image was read in grayscale mode as the processing is faster at the expense of losing information.

2. Rotation

It was observed that randomly rotating the images by ± 10 degrees improves the training accuracy.

3. Resizing the Image

Multiple image sizes were tried. It was found that smaller images will train significantly faster and possibly even converge quicker. For earlier models, image sizes of (24 x 32) or (48 x 48) in grayscale worked best. However, the model which is proposed in this report works the best on 64 x 64 sized images.Hence, image size of 64 x 64 was chosen for the purpose of this case study.

4. Normalization

After reading the data into a numpy array, it ensures the data has zero mean and unit variance(divide by 255). This makes it easier to search for optimization values when trying to minimize the loss function.

5. Localization: YOLO(You Only Look Once) [1]

Current detection systems repurpose classifiers to perform detection. To detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.
YOLO architecture is more like FCNN (fully convolutional neural network) and passes the image (nxn) once through the FCNN and output is (mxm) prediction. Thus the architecture is splitting the input image in mxm grid and for each grid generation 2 bounding boxes and class probabilities for those bounding boxes. The bounding box is more likely to be larger than the grid itself.

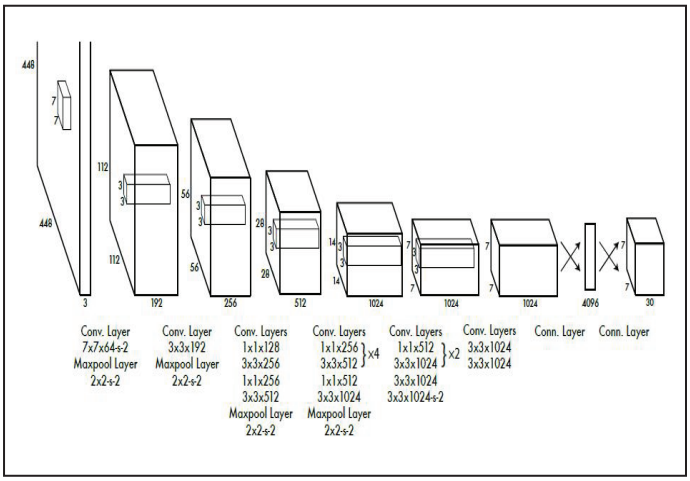

Fig. 2:

YOLO re-frame object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities and only looks once at an image to predict what objects are present and where they are.
YOLO is extremely fast. Since we frame detection as a regression problem we don't need a complex pipeline.
YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Fast R-CNN, a top detection method, mistakes background patches in an image for objects because it can't see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN.

It divides the input image into a S x S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and each box consists of 5 predictions: x, y (relative center), w, h (dimensions) and confidence. The model is implemented as a convolutional neural network. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates. They used the GoogLeNet model for image classification as the basis architecture. The final output of our network is the 7x7x30 (trained S=7 and B=2) tensor of predictions.

YOLO outperforms top detection methods like DPM and R-CNN by a wide margin. Since YOLO is highly generalizable it is less likely to break down when applied to new domains or unexpected input.

B. Convolutional Neural Network (Final Version): Inspired from VGG 16 NN

We have adopted the following neural network architecture:

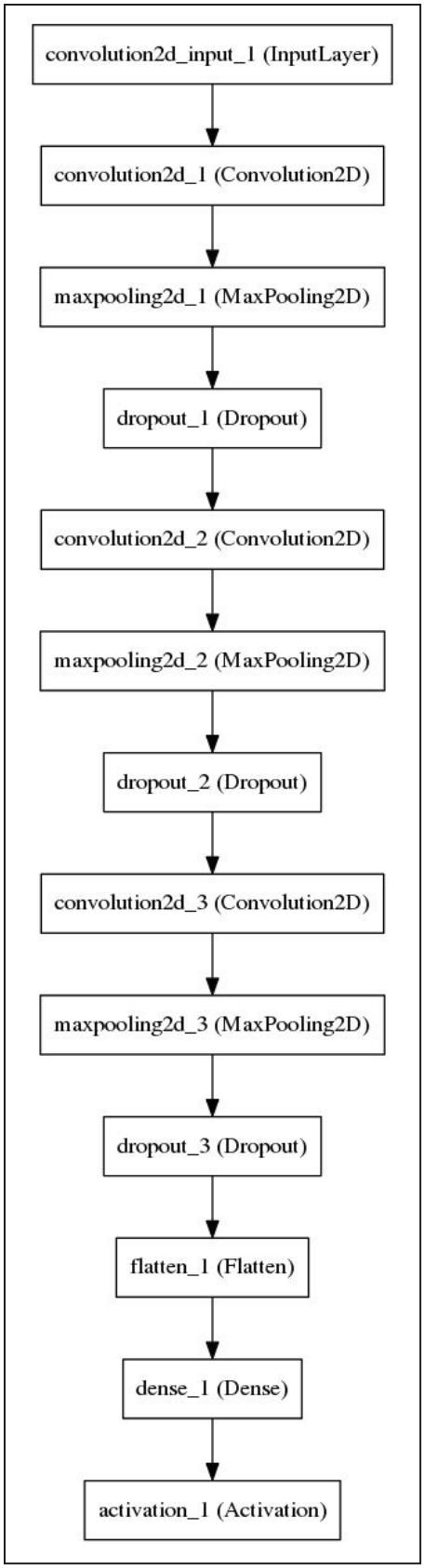

Fig. 3:

Input Layer: 64 X 64 X 1
First Convolutional Layer: 32 filters, 3 X 3 size
First Max Pooling Layer: 2 X 2 size
Dropout: 0.5

Second Convolutional Layer: 64 filters, 3 X 3 size
First Max Pooling Layer: 2 X 2 size
Dropout: 0.5

Third Convolutional Layer: 128 filters, 3 X 3 size
First Max Pooling Layer: 2 X 2 size
Dropout: 0.5

Fully Connected Layer: 9 Neurons
First Activation: Softmax

1. K-Fold Validation

The original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. There were two options:

- Split by drivers
- Split by classes

Split by drivers works best, as it makes the neural network independent of the driver (abstracts out the learning of individual drivers like clothes they are wearing, their body types etc. which are not useful to the problem). Thus, 26 fold validation is used for this problem.

A fully random split is useless, most convolutional neural networks seem to learn about individual drivers very well, so there is lots of overfitting to the Cross Validation set. It would seem to be quite easy to train a CNN to recognize the classes of a specific driver (with same clothes, hairstyle etc in same car).

C. Hyparameter Tuning

1. Optimizer used

Adam optimization algorithm has been used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. It is straightforward to implement, computationally efficient, has little memory requirements, only requires first-order gradients . The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. Also the Hyper-parameters have intuitive interpretation and typically require little tuning. Adam realizes the benefits of both Adaptive Gradient Algorithm and Root Mean Square Propagation. In addition to storing an exponentially decaying average of past squared gradients like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients.
Some of Adam's advantages are that the magnitudes of parameter updates are invariant to rescaling of the gradient, its step sizes are approximately bounded by the stepsize hyperparameter, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing. The proposed default value for:
beta1-The exponential decay rate for the first moment estimates is .9
beta2- The exponential decay rate for the second-moment estimates is .999
epsilon - number to prevent any division by zero and 10^{-8}

It shows empirically that Adam works well in practice and compares favorably to other adaptive learning-method algorithms [5-6].

2. Loss Function

The cross-entropy loss function was used as it takes into account the closeness of a prediction and is a more granular way to compute error [4].

3. Learning rate

In most of the models, lowering the learning rate from default values (0.1 in the case of SGD) helped improve the log loss error. Learning rate of 0.001 gave the best results for this model.

4. Batch Size

In general, the idea is larger the batch size, more representative is the batch of the training sample. Thus, intuitively larger batch size should yield better results. However, the following observations were made:

In this case, a smaller batch size actually helps! Smaller batch size means more gradient updates per epoch

- A very large batch size (say 64) did not yield very good results i.e the log loss error increased(overfitting). Also, larger batch size takes more time to process.
- A very small batch (say 8) further increases the log loss error.
- At batch size = 16 we hit the "sweet spot", where the log loss error is minimum.

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring careful parameter initialization and low learning rates, and makes it notoriously hard to train models with saturating non-linearities. This problem can be resolved by normalizing layer inputs. Batch Normalization draws its strength from making normalization a part of the model architecture and performing the normalization for each training mini-batch. It allows us to use much higher learning rates and be less careful about initialization. It also acts as a regularizer, in some cases eliminating the need for Dropout. It claim to achieve the same accuracy when applied to state-of-the-art image classification model with 14 times fewer training steps, which beats the original model by a significant margin [3].

5. Number of Epochs

An epoch is essentially a measure of the number of times all of the training vectors are used once to update the weights. Try to have a large number of epochs as only then does the optimizer converge to global minimum(about 100)

IV. Future Approach

A. Localization

Semantic Segmentation (Using Conditional Random Fields as Recurrent Neural Networks [2])

Semantic image segmentation involves assigning a label to each pixel in an image. There are significant challenges in adapting CNNs designed for high level computer vision tasks such as object recognition to pixel-level labeling tasks as traditional CNNs have convolutional filters with large receptive fields and hence produce coarse outputs when restructured to produce pixel-level labels and presence of maxpooling layers in CNNs further reduces the chance of getting a fine segmentation output. Markov Random

Fields (MRFs) and its variant Conditional Random Fields (CRFs) have observed widespread success in this area. The key idea of CRF inference for semantic labeling is to formulate the label assignment problem as a probabilistic inference problem that incorporates assumptions such as the label agreement between similar pixels. CRF inference is able to refine weak and coarse pixel-level label predictions to produce sharp boundaries and fine-grained segmentations. A CRF, used in the context of pixel-wise label prediction, models pixel labels as random variables that form a Markov Random Field (MRF) when conditioned upon a global observation. An energy is assigned to every configuration and depends on the CNN which predicts labels for pixels without considering the smoothness and the consistency of the label assignments. Minimizing this CRF energy yields the most probable label assignment x for the given image. Since this exact minimization is intractable, a mean-field approximation to the CRF distribution is used for approximate maximum posterior marginal inference.

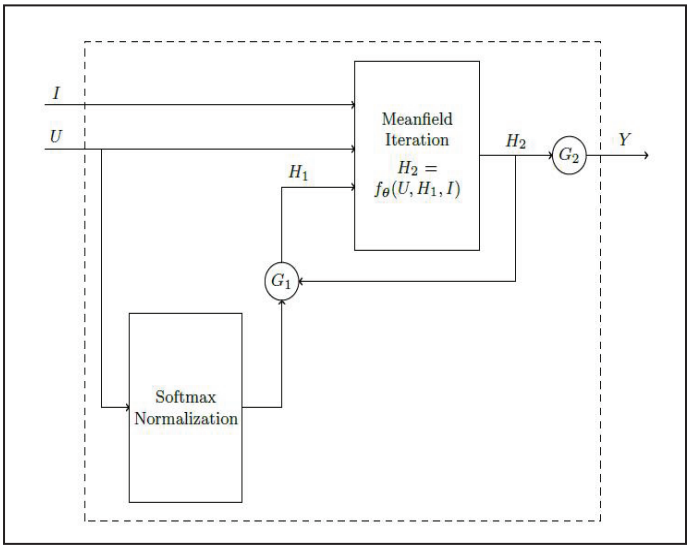

Fig. 4: This Iterative Algorithm was Restructured as RNN

The high-level architecture of the system is implemented using the popular Caffe deep learning library. The first part of the network was initialized using the publicly available weights of the FCN-8s network. Standard softmax loss function was used as error metric. Their system achieves a new state-of-the-art on the popular Pascal VOC segmentation benchmark.

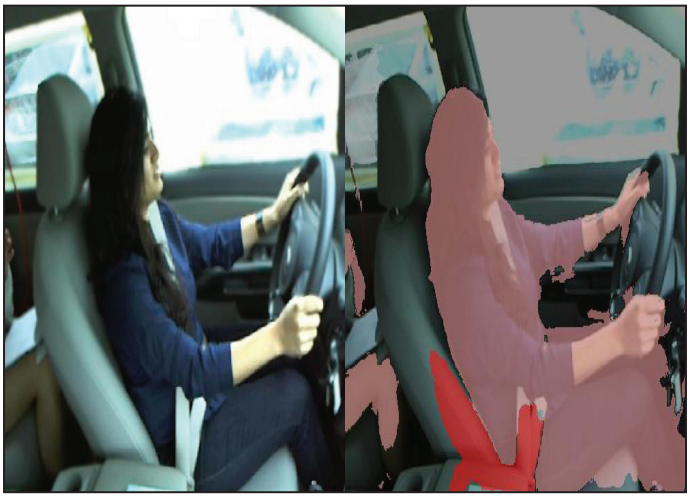

Fig. 5:

B. Real-time Face Pose Estimation

This is another technique which can be tried out.

C. Crop out Hands/Images

A potentially workable strategy would be to crop out many examples of hands and heads from the training examples (plus of course negative examples without hands or heads in them), create identifiers for hands and heads, then use those to pick out parts of the image to run a final stage image classifier on.

A similar issue might be worthwhile to predict camera angle from some sub-part of the image, by picking out fixed parts of the car. However, that might be possible with more old-school non-ML approaches, because there are some nice regular shapes to work from. i.e Find a head / hand in the image. Classify the hand/head.

D. Remove Outliers?

Some drivers are easy-to-predict, while some are hard-to-predict drivers when they are in the CV set. Easiest to predict, CV loss tends to match the training loss and accuracy is over 80% . Hardest to predict , with CV loss and accuracy is bad (consistently worse than 10%).

Case-1: If a dark skinned driver dressed with long black sleeves, about the same color as the car's, covering entire arms to the fingers. Then the network fail to identify arms, which is one of the strongest differentiators.

Case 2: A very huge bodied driver, distorts the geometry network learned based on all other subjects in the train set.
So, spotting such outliers and then training could improve the results.

V. Conclusion

The absence of robust data, specifically related to crashes caused due to the use of mobile phone is a serious flaw. Police data collection system across the country need to incorporate the use of mobile phone in crashes as a field to be compulsory filled by investigating officers. The Ministry of Road Transport and Highways, Government of India on February 21st, 2017 has released a revised Road Accident Data and Reporting Format recommended to be used by all State Governments and UT Administrations and it does have in Column 13, "Use of Mobile Phone" as one of the options to ascertain the type of traffic violation that contributed to the crash.
Setting in place an effective legislative framework along with a visible and sustained enforcement mechanism and collection of powerful data on the basis of this case study will definitely reduce the use of mobile phones while driving.

References

[1] Redmon J, Divvala S, Girshick R, Farhadi A (2015) You Only Look Once: Unified, Real-Time Object Detection. Arxivorg. Available: http://arxiv.org/abs/1506.02640 Accessed 10 May 2016.

[2] Zheng, Shuai, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, Philip H. S. Torr.,"Conditional Random Fields as Recurrent Neural Networks", IEEE International Conference on Computer Vision (ICCV) 2015.

[3] Ioffe Szegedy C (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Arxivorg. [Online] Available: http://arxiv.org/abs/1502.03167

[4] Why You Should Use Cross-Entropy Error Instead Of Classification Error Or Mean Squared Error For Neural Network Classifier Training (2013). James D McCaffrey. [Online] Available: https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training

[5] Kingma DBa J (2014) Adam: A Method for Stochastic Optimization. Arxivorg. [Online] Available: http://arxiv.org/abs/1412.6980

[6] Jason Brownlee (2017) : Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. [Online] Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/