Machine Learning Techniques for Distracted Driver Detection

Demeng Feng
Department of Materials Science and Engineering
Stanford University, Stanford, CA 94305, USA
dmfeng@stanford.edu

Yumeng Yue
Department of Materials Science and Engineering
Stanford University, Stanford, CA 94305, USA
yuey3@stanford.edu

1 Introduction

Distracted driving is a main factor that causes severe car accidents. It has been suggested as a possible contributor to the increase in fatal crashes from 2014 to 2018, and is a source of growing public concern [1]. It is revealed that different distraction activities have different risks of causing accident [2]. Thus, a proper recognition and categorization of distraction activities via images of drivers in their driving is important.

This project focuses on driver distraction activities detection via images using different kinds of machine learning techniques. Our goal is to build a high-accuracy model to distinguish whether drivers is driving safely or conducting a particular kind of distraction activity. The input of our model is images of driver taken in the car. We first preprocess these images to get input vectors, then use different classifiers (linear SVM, sofrmax, naive bayes, decision tree, and 2-layer neural network) to output a predicted type of distraction activity that drivers are conducting.

2 Related Work

Real-time image-based driver distraction detection is a popular topic in machine learning and computer vision field, and many models and algorithms are proposed and analyzed by researchers. Two main topics that researchers focus on are image preprocessing technique and classifying model selection. As for image preprocessing, HaQ *et al.* proposed that conducting feature extraction instead of directly conducting image flattening might, though not guaranteed, improve prediction accuracy [3].

As for the classifying model, one main approach is to use convolutional neural work (CNN)-based models. Liu *et al.* proposed a convolutional two-stream network with multi-facial feature fusion to detect distraction activities by combining static and dynamic features [4]. Eraqi *et al.* proposed a model based on ensemble of convolutional neural networks and shows that a weighted ensemble of classifiers using a genetic algorithm yields a better accuracy [5]. It is revealed that CNN-based models generally report high accuracy. Another major approach is to use Support Vector Machine (SVM). Osman *et al.* shows that both linear SVM classifier and nonlinear SVM classifier can reach reasonably high accuracy [6]. Comparing with CNN-based model, SVM model will not get the best accuracy, but the learning process is faster, and computational cost is lower than CNN. The third main approach is to use semi-supervised learning. Since the datasets obtained in this problem are likely to include many unlabelled images, using semi-supervised learning augments available data amount for training and can potentially improve prediction accuracy. Liu *et al.* proposed a semi-supervised model and revealed that with the additional unlabeled data, the semi-supervised learning methods improved the detection performance compared to the traditional supervised methods [7]. Besides these 3 main approaches, shallow neural network with feature extraction [8], random forest [6] , and K Nearest Neighbor [9] were also been explored.

3 Dataset and Image Preprocessing

The whole dataset contains 22,424 images categorized in 10 classes from *State Farm*®[10] (9 classes for different distraction activities and 1 class for safe driving), and Figure 1 shows illustrating images for each category. The size of each image is 640×480 pixels.

Each image needs to be preprocessed before going to the classifier, and the float chart of image preprocessing is shown in Figure 2. Each image is first converted into a high-dimensional $640 \times 480 \times 3$ matrix based on its RGB values on each pixel, then resized to a 64 $\times$ 64 $\times$ 3 matrix using CV2 in order to improve the computing efficiency of classifier, and finally flattened into a vector. For each flattened vector, a numerical label in the range of 0-9 is assigned based on the category it belongs. As for the whole training data, we shuffle the data and split them into the training set and the validation set by 80% / 20%, and pile each training example along the column axis. Thus, the final training set matrix $\boldsymbol{X}_{\textbf{train}}$ has the dimension (17939, 12288), the final training label vector $\boldsymbol{Y}_{\textbf{train}}$ has the dimension (17939,); the final validation set matrix $\boldsymbol{X}_{\textbf{val}}$ has the dimension (4485, 12288), the final training label vector $\boldsymbol{Y}_{\textbf{val}}$ has the dimension (4485,).

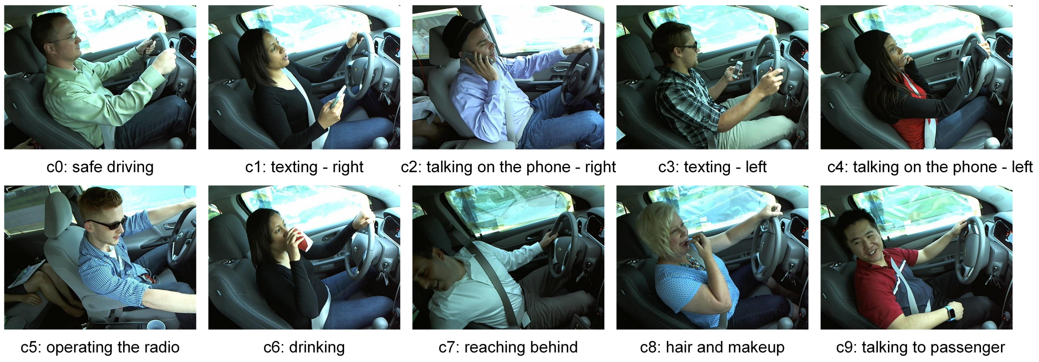

Figure 1: An illustration of 10 categories of driving activities considered here. c0 - c9 represent the label of each image in the dataset.

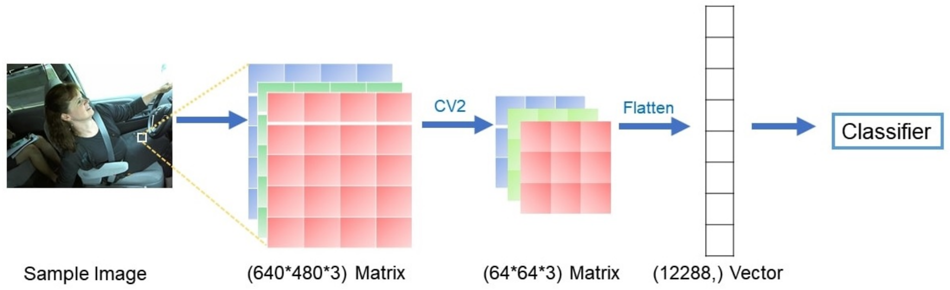

Figure 2: Float chart of image preprocessing

4 Methods

Below are 5 different models we considered for classification. For each one, the input is the flattened vector representing one image example we obtained after image preprocessing.

4.1 Linear SVM

For the baseline model, we choose the multi-class linear SVM(Support Vector Machine) method. The loss function is (we use L2 regularization here)[11]:

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i}\max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + 1) + \lambda||W||_2^2 \tag{1}$$

where x_i is the i-th training sample, y_i is the label of this training sample, j ranges from other label values other than y_i, and f is the score vector defined as $f(x, W) = Wx$, where W is the weight matrix. W can be updated via the gradient descent rule:

$$W := W - \alpha\frac{dL}{dW} \tag{2}$$

During prediction process, given an test example x, first calculate its score vector f, then the predicted class y of this example is

$$y = \underset{j}{\operatorname{argmax}}\, f(x, W)_j \tag{3}$$

where j is the j-th row of f, and j = 0, 1, ..., 9.

4.2 Softmax

Similar to linear SVM, another classifier we consider is softmax. The loss function is defined as follows (we use L2 regularization here):

$$L = \frac{1}{N}\sum_{i=1}^{N} -\log(\frac{\exp f(x_i, W)_{y_i}}{\sum_j \exp f(x_i, W)_j}) + \lambda ||W||_2^2 \tag{4}$$

where the definition of x_i, y_i, W and the subscript j are defined the same as in the previous section. The update rule of W is gradient descent rule as in equation (2), and the prediction rule is defined in equation (3).

4.3 Naive Bayes

The Naive Bayes classifier assumes that each element of the input vector be statistically independent. During training process, we first use training samples that are labelled c_k (k = 0, 1, ... 9) to fit the following probability distribution:

$$P(x_i \,|\, c_k) = \frac{1}{\sqrt{2\pi\sigma_{c_k}^2}} \exp\left(-\frac{(x_i - \mu_{c_k})^2}{2\sigma_{c_k}^2}\right) \tag{5}$$

where x_i is the element of each input vectors (i = 0, 1, ..., 12287). During prediction process, given an input vector $x = (x_0, x_1, ..., x_n)^T$, we can predict the class it belongs to use the following equation:

$$\hat{c_k} = \underset{c_k}{\text{argmax}}\, P(c_k) \prod_{i=0}^{n} P(x_i \,|\, c_k) \tag{6}$$

4.4 Decision Tree

Decision Tree is an important non-parametric method for image mining, This method uses several simple decision rules of image to provide a path for image classification [12]. Figure 3 shows an illustration of decision tree associated with this project for image classification. Here we use `DecisionTreeClassifier` built in sci-kit learn as the classifier.

4.5 Two-layer Neural Network

The last model we consider is a 2-layer neural network. Figure 4 shows the architecture of this neural network. There are 100 neurons in the hidden layer, and we use ReLU as the activation function for this layer. For the output layer, we use softmax activation function for final classification.

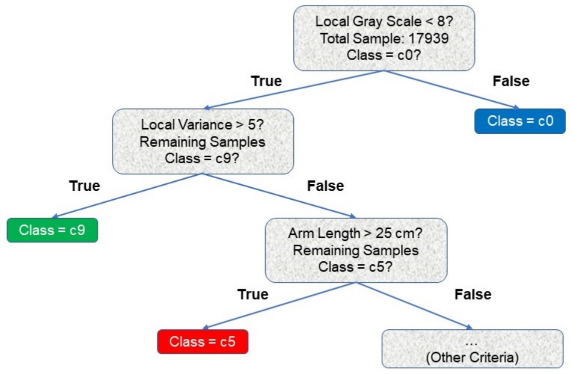

Figure 3: An illustration of decision tree for image classification

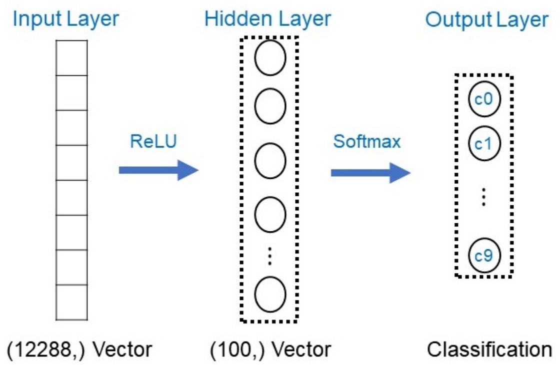

Figure 4: Architecture of this two-layer neural network

5 Experiments and Results

5.1 Experiment Setup

In our project, we adopted the starter code from the course Convolutional Neural Network for Visual Recognition (CS231N) [13] and completed the implementation of LinearSVM, Softmax and Two-Layer Net classifier on our own.

Also, we made some modification on top of that to match our project's need. For the Naive Bayes and Decision Tree classifier, we directly adopted their respective packages with default hyperparameters from the "Scikit-learn" package, which is a python version of machine learning package. Then we preprocessed our image samples and split them into 80% training and 20% validation, followed by feeding them into each classifier. In the softmax classifier, we trained 2000 iterations with 1×10^{-7} as learning rate and 1×10^{4} as regularization factor. In the Two-layer Net, we trained 2000 iterations as well with learning rate of 1×10^{-4}, regularization factor of 0.5 and learning rate decay rate of 0.9. In the Naive Bayes and Decision Tree classifiers, we turned on the option of cross-validation, so outcome represents the mean accuracy. The evaluation metric we used is the validation set accuracy, which is defined as the fraction of right class prediction throughout the validation set.[1]

5.2 Results and Model Evaluation

We choose SVM classifier as our baseline model. Figure 5 shows the results of SVM classifier. By tuning the 2 hyperparameters learning rate α, ranging from 8×10^{-9} to 1×10^{-8} and regularization coefficient λ ranging from 7×10^{4} to 9×10^{4}, we find that when $\alpha = 1 \times 10^{-8}$ and $\lambda = 9 \times 10^{4}$, SVM classifier has the best validation accuracy of 71.4% , and at this time the training set accuracy is 72.8% (see Figure 5(b) and 5(c) for results). Figure 5(a) shows the loss function vs. iteration number for this best SVM model. It should be noticed that the accuracy of SVM classifier is highly sensitive to the initialization of weight matrix *W*. Figure 5(d) shows the result when *W* is initialized differently and the range of 2 parameters are kept the same. Compare this with Figure 5(c), we can see that a bad initialization will drastically reduce the validation set accuracy. Thus, we set random seed to the classifier and used some smaller number for the weight matrix to try to eliminate this randomness.

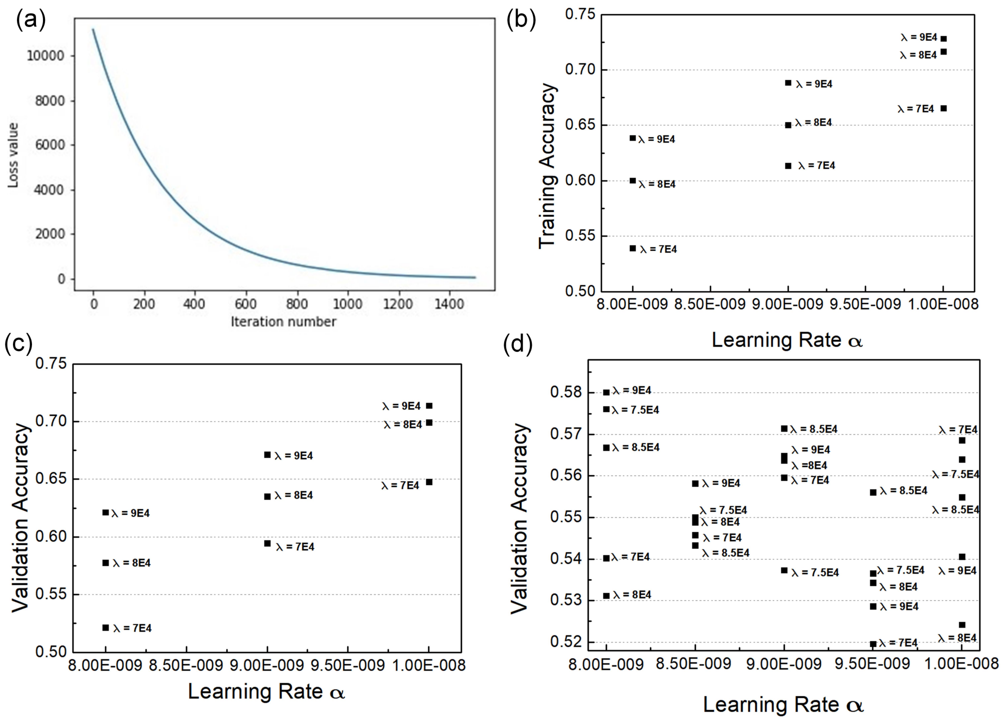

Figure 5: SVM Classifier Results: (a) Loss value vs. iteration number of the best result. (b) Training set accuracy of SVM models with different hyperparameter values. (c) Validation accuracy of SVM models with different hyperparameter values. (d) Validation accuracy of SVM models with "bad" weight initialization.

Table 1 shows the best accuracy obtained from different classifiers. Since we directly adopted the "Scikit-learn" packages, we could not obtain the training accuracy for the Naive Bayes and Decision Tree classifiers. Table 1 reveals these following things:

1. **Naive Bayes for Image Classification**: it is shown that except for Naive Bayes classifier, all classifiers give very good performance on this task, with accuracy higher than 70%. It is expected since Naive Bayes model generally does not do well in image classification task.

[1] All codes for these experiments are part of the repository: `https://github.com/jowe41/cs229`

2. **Overfitting**: Comparing training accuracy (if available) and validation accuracy differences of each classifiers, it can be concluded that these classifiers do not overfit, which indicates that tuning regularization coefficient successfully decreases the variance.

3. **Best Model**: In our experiments, the Two-layer Neural Network model shows the best validation accuracy of 92.2%, which matched our expectation that CNN-based models will have advantages on task of computer vision compared with others.

Classifiers	**Training Acc.**	**Evaluation Acc.**
Naive Bayes	N/A	54.99
Decision Tree	N/A	84.73
Linear SVM	72.82	71.39
Softmax	82.32	82.31
Two Layer Net	93.21	**92.24**

Table 1: Training and Validation Set Accuracy of Different Models

To analyze the best model in detail, we visualize the per class accuracy. The result, shown in Table 2 below, illustrates that the model performs relatively bad on "**Talking to the Phone-Left**" and "**Hair and Makeup**" classes, with accuracy in 70% - 80%. We examine all wrong predictions examples in these 2 classes to see the reason.

Among all the wrong predictions of "Talking on the Phone-Left" class, there are 75 examples that are wrongly predicted as "Texting-Left". From Figure 1 we can see these two activities are very similar: a driver raising his/her left hand with his/her phone but the position of the phone determines which class it belongs to. Thus, we think the model classifies some images to be "Texting-Left" is due to the similarity of these 2 activities, especially for the case where the drivers does not put the phone close to their ears but talk through speakers. The gesture of the drivers as well as the position of the phone make the model misclassified the image to be texting. This also explains why "Talking on the phone" classes have lower accuracy than "Texting".

As for the "Hair and Makeup" class, among all the wrong predictions, 19 of them are classified as "Drinking", 13 of them are classified as "Talking on the Phone - Right" and the rest of them are wrongly classified among all the other classes. By taking a close look of all misclassified images, we can find that most of the wrong predictions come from "Makeup" instead of "Hairing". Since finishing hair is a more unique gesture among all the classes (putting one's hand on top of the head), this result is expected. As for the "Makeup" behaviors, most of these involve the activities "raising right hands close to driver's face", which share the same gesture as "Talking on the Phone - Right" and "Drinking". For a small-size picture, it is hard for the classifier to exactly distinguish these 3 activities by the object that the driver is holding and the exact position of driver's right hand, and this hinders the model to have higher accuracy.

Class	**Acc.**	**Class**	**Acc.**
Save Driving	89.20	Operating the Radio	98.28
Texting - Right	97.80	Drinking	98.03
Talking on the Phone - Right	95.83	Reaching Behind	97.08
Texting - Left	98.37	Hair and Makeup	**78.97**
Talking on the Phone - Left	**71.24**	Talking to Passenger	96.20

Table 2: Per Class Accuracy on Two Layer Net

6 Conclusion and Future Work

In conclusion, we successfully implemented a Two-layer Neural Network model that has great performance on the distracted driver detection task and gives an evaluation accuracy of 92.24%. Still, the Convolutional Neural Network has many advantages on image classification tasks compared to traditional machine learning techniques. For the future improvement of these self-implemented models, we could applied more sophisticated weight initialization method like Xavier Initialization or KaiMing Initialization. Besides, the dataset also provides 77,000 test images without labels, so if these images are labelled or we use semi-supervised learning method, our models could be developed with these images. Finally, we could try out some existing packages from Pytorch or Tensorflow of more complex CNN-based models like ResNet or VGG.

7 Contribution

Y. Yue and D. Feng works together on experiments design, image preprocessing and model selection. Y. Yue also works on implementation of all models, and designs training and evaluation process. D. Feng also works on literature review and model evaluation analysis. Both D. Feng and Y. Yue work together on report writing and revising. All group members make equal contribution to this project.

References

[1] D. G. Kidd, N. K. Chaudhary, "Changes in the sources of distracted driving among Northern Virginia drivers in 2014 and 2018: A comparison of results from two roadside observation surveys", *Journal of Safety Research* **2019**, *69*, 131-138; `doi: 10.1016/j.jsr.2018.12.004`

[2] R.L. Olson, R.J. Hanowski, J.S. Hickman, J. Bocanegra. Driver Distraction in Commercial Vehicle Operations. *U.S. Department of Transportation, Washington, DC*. Data avaliable online since September 2009 on: `https://www.fmcsa.dot.gov/sites/fmcsa.dot.gov/files/docs/FMCSA-RRR-09-042.pdf`

[3] I. HAQ, S. Haryar, and J. Zhang, "Driver Drowsiness Detection Based on HOG and Gabor features", in *ICEMEH 2019: 2019 International Conference on Education, Management, Economics and Humanities*, **2019**, 277-282; `doi: 10.12783/dtssehs/iceme2019/29597`

[4] W. Liu, J. Qian, Z. Yao, X. Jiao, and J. Pan, "Convolutional Two-Stream Network Using Multi-Facial Feature Fusion for Driver Fatigue Detection", *Future Internet* **2019**, *11*, 115-127; `doi: 10.3390/fi11050115`

[5] H. Eraqi, Y. Abouelnaga, M. Saad, and M. Moustafa, "Driver Distraction Identification with an Ensemble of Convolutional Neural Networks", *Journal of Advanced Transportation* **2019**, 1-12; `doi: 10.1155/2019/4125865`

[6] O. Osman, M. Hajij, S. Karbalaieali, and S. Ishak, "A Hierarchical Machine Learning Classification Approach for Secondary Task Identification from Observed Driving Behavior Data", *Accident Analysis and Prevention* **2019**, *123*, 274-281; `doi: 10.1016/j.aap.2018.12.005`

[7] T. Liu, Y. Yang, G. Huang, Y. Yeo, and Z. Lin, "Driver Distraction Detection Using Semi-Supervised Machine Learning", *IEEE Transactions on Intelligent Transportation Systems* **2016**, *17*(**4**), 1108-1120; `doi: 10.1109/TITS.2015.2496157`

[8] M. Botta, R. Cancelliere, L. Ghignone, F. Tango, P. Gallinari, and C. Luison, "Real-time detection of driver distraction: random projections for pseudo-inversion-based neural training", *Knowledge and Information Systems* **2019**, 1-16; `doi: 10.1007/s10115-019-01339-0`

[9] M. Wali, M. Murugappan, and B. Ahmmad, "Wavelet Packet Transform Based Driver Distraction Level Classification Using EEG", *Mathematical Problems in Engineering* **2013**, 1-10; `doi: 10.1155/2013/297587`

[10] Dataset for *State Farm Distracted Driver Detection*. Available at: `https://github.com/Kaggle/kaggle-api`

[11] C. D. Manning, P. Raghavan, and Hinrich Schutze, "Introduction to Information Retrieval", *Cambridge University Press* **2009**, 344-346; available at `https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf`

[12] C. Agarwal, A. Sharma, "Image Understanding Using Decision Tree Based Machine Learning", *ICIMU 2011: Proceedings of the 5th international Conference on Information Technology & Multimedia* **2011**, 1-8; `doi: 10.1109/ICIMU.2011.6122757`

[13] Stanford University, Course CS 231N, Spring 2019, Assignment 1 starter code. Code Framework available at: `cs231n.github.io/assignments/2019/spring1819_assignment1.zip`.