Classification Models of Driving Distraction: Analysis and Comparison

Liuming Zhao, Taiming Zhang, Lingzi Guo

***Abstract*— Driving distraction has always been a driving safety issue since cars were invented. In order to deal with this problem, we aim to build a computer vision system to classify different driving distraction behaviors. In this project, different models have been implemented, such as fully connected neural network, basic CNN, VGG-16, and Inception-v4, to perform the detection of driving distraction.**

I. Introduction

During recent years, public concern about safe driving has reached unprecedented levels. Road traffic accidents have been regarded as one of the ten most frequent death causes in the world[1]. Distraction, a type of inattention, which has been defined as a slow response of a driver in recognizing the information required to drive safely due to some external factors[2]. It has been regarded as a major contributor to highway crashes. According to the National Highway Traffic Safety Administration(NHTSA), in 2015 alone, 3477 people were killed, and 391000 were injured in motor vehicle crashes involving distracted drivers[3]. Finding valid measures to detect distracted drivers has become a research focus. According to the Center for Disease Control and Prevention (CDC), distraction can be categorized into three types: visual (i.e. taking one's eyes off the road), cognitive (i.e. taking one's mind off driving) and manual (i.e. taking one's hands off the driving wheel) distractions[4]. In our project, we only focus on "manual distractions". They are usually accompanied by changes in hand position and could been effectively distinguished by posture estimation.

With the goal of detecting driver distractions, State Farm has initialized a Kaggle competition to encourage people design a driver posture classification system. They provided a dataset of 2D dashboard camera images which included ten classes of postures like texting, drinking and other behaviors(which we have used as algorithm input), and require people to classify each image into one of the ten classes. Our project presents a solution to the Kaggle competition by using different Convolutional Neural Network (CNN) algorithms: fully connected neural network, basic CNN, VGG-16 and Inception-v4 to output predicted class labels and the corresponding probabilities of what the driver is doing.

II. Related Work

Due to the public Kaggle competition, there are vast solutions to detect distracted drivers. As the most state-of-art algorithm, Convolutional Neural Network (CNN) has lead to impressive performance on visual recognition problems[5,6]. In our project, we use basic CNN model as baseline. Another popular model, VGG Net, which is a pre-trained deep Convolutional Networks (ConvNets)[7], could push the depth to 16-19 weight layers, called VGG-16 and VGG-19 separately[8]. Simonyan et al won the 2nd place using VGGNet on ILSVRC 2014. The third model is Inception-v4, which is proposed by Google in 2016, could achieve 3.08% top-5 error[9].

For the specific distracted driver detection task, most of the top teams on Kaggle LeaderBoard have tried ResNet and Inception models[10], then ensembled different models. In our project, we use basic CNN, VGG-16 and Inception-v4 models.

III. Dataset And Preprocessing

In this project, we will use the dataset provided by the State Farm in the Kaggle Challenge[10]. The dataset consists of images (640x480 pixels RGB) with different drivers' behaviors. Our task is to classify the input images into 10 classes that represent different drivers' behaviors: c0: safe driving, c1: texting - right, c2: talking on the phone - right, c3: texting - left, c4: talking on the phone - left, c5: operating the radio, c6: drinking, c7: reaching behind, c8: hair and makeup, c9: talking to passenger. The input images are divided into two groups: (1) training group that contains 22,400 images labeled with one of the 10 classes and (2) testing group that contains 79,727 unlabeled images. Apart from images, our dataset contains drivers' list in CSV form in which each of the driver is identified by the subject number along with their corresponding class and image numbers in the training group. In total, we have 26 different drivers among the training examples. We will use K-folds cross-validation method to split our training data into training set and validation set according to the driver in order to make sure that we do not overfit our training model by over-emphasizing on the features of drivers instead of their driving behaviors. The challenging part of our project is that it is very easy to overfit our training model due to limited number of drivers in our dataset. Because of high similarities between two images with the same driver, it is more likely for the training model to over-emphasize on the drivers' features instead of their driving behaviors.In our case, 21 drivers from 10 classes will be our training set and the other 5 drivers from 10 classes will be our validation set. Since there are approximately 2200 images in each class, we

do not have to worry about the imbalance problem of our data set.

Before feed our input image into our training mode, we resized the images from size 640x480x3 into 150x150x3 with both mean and variance normalized so that each pixel's value range from -0.5 to 0.5.

IV. METHOD

A. SOFTMAX REGRESSION

In this project, we aim to classify driver into 10 categories. Softmax regression would be our must-try method to approach this problem. Given the input data, it outputs a categorical distribution for each sample, which represents the probabilities of the outcomes for each category. Softmax regression is typically used in the output layer for neural network, which is responsible for classifying images to corresponding categories.

B. LOGARITHMIC LOSS FUNCTION

The logarithmic loss function will be applied to evaluate the overall performance of our model, which is given as follow:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij})$$

where N is the number of images in the test set, M is the number of image class labels, y_{ij} is 1 if observation i belongs to class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j. This equation will be used to evaluate our outcomes throughout our project.

C. FULLY CONNECTED NEURAL-NETWORK

The first baseline model that we used in our project is fully connected neural network model for MNIST handwritten digit detection, which has been reported to have a test accuracy 0.9761[11]. The overall architecture of fully connected neural network consists of 5 layers. The final layer is the softmax layer that contains 10 nodes representing 10-classe classifications in our dataset. The input layer contains 67500 input nodes after the resizing our input image (150x150x3) into one single vector with the size of 67500x1. The detailed architecture implementation is shown below:

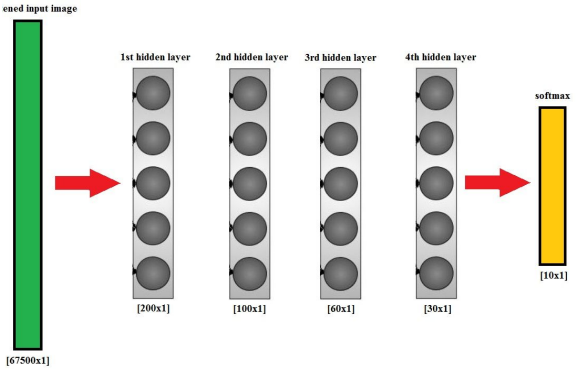

Fig. 1. Fully-connected Neural-Network

D. BASIC CONVOLUTIONAL NEURAL-NETWORK

Convolutional Neural-Network has been reported to be an ideal paradigm for image recognition and classification. It sometimes is referred as "black box", which can discover complex feature relations by back-propagation. We plan to adapt our basic CNN model from MNIST handwritten digit detection [12], which consists of two stages, each of which has a convolutional layer and a max pooling layer, and finally a softmax output layer is applied to operate classification. The convolution layers will achieve feature detection from original images, while the pooling layer will down-sample the original image for faster execution. Finally, a fully connected layer (softmax layer) is used to classify images to different categories. In addition to the layers mentioned above, we also include 3 dropout layers (2 of them right after each max pooling layer and one after the flatten layer) in order avoid any potential overfit problems due to limited number of input images in our dataset. The structure of our basic CNN model is illustrated in Fig.2. In order to accelerate our training process, we will use Google Cloud with 8 CPU plus 52 GB RAM to train our basic CNN models, and plan to use GPU to train advanced CNN models if desired. In our model, we set the training epoch to be 10 and batch size to be 32 to reduce the noise from a small amount of images.

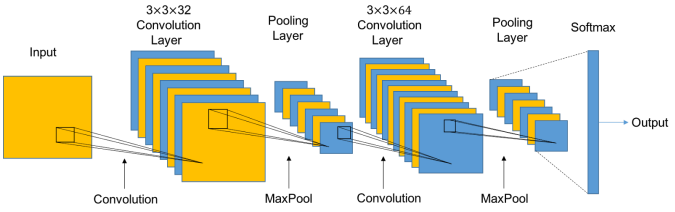

Fig. 2. Basic CNN architecture

E. VGG-16 MODEL

The first pre-trained model for image recognition and classification we decide to use is VGG-16, which is developed by Visual Geometry Group at Oxford University[7]. VGG-16 model consists of 5 hidden layers, each of which has several convolution layers and a max pooling layer as shown in Fig.3. Since training VGG-16 from scratch would be very time consuming, we decided to stick with the pre-trained model by setting it untrainable, while only training the output fully connected layer. To achieve this, given the input data, we only run the forward propagation through hidden layers once to get the output. Then, we keep training the output fully connected layer by using the output from hidden layers, which saves us a great amount of time and resources.

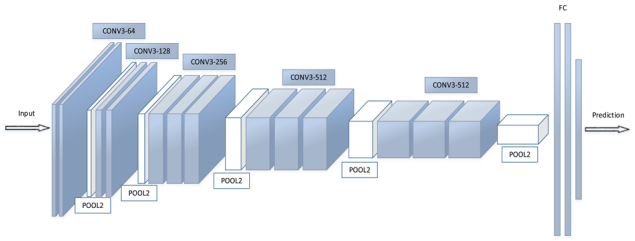

Fig. 3. VGG-16 architecture

F. *Inception-v4*

Another pre-trained model that we use for image classification task is Inception-v4. This model is the state-of-art architecture which was proposed by Google in 2016[10]. With the introduction of TensorFlow, the Inception-v4 model could be implemented in simple architecture. The ensemble consist of one Inception-v4 and three Inception-ResNet-v2 could achieve 3.08% top-5 error on the test set of ImageNet. Similar to VGG-16, we remove the output layer, leading to 8x8x1536 encoding for each image. We then feed the encoding for each image to our newly-created 10-class classifier. We implement our code using Keras library with tensorflow as backend. Pre-trained inception V4 model was adapted from Somshubra Majumdar's github[11].

G. *K-nearest Neighbors Algorithm*

By examining the test set, we observed that images are highly correlated because they are taken from the in-car cameras[9]. Even though images from the same driver differ subtly, our model might still predict them in different categories. Therefore, we introduce the K-nearest Neighbors Algorithm (K-nn) that gets the most k relevant neighbors in terms of Euclidean pixel difference. We can calculate the average probability among the neighbors and overwrite their probabilities. This serves as a noise reduction and greatly reduces the effect of the outliers.

Fig.4. Similar images from test set using K-nn algorithm

Fig.4. shows an example of similar images taken from test set by using K-nn algorithm. These images must belong to the same category, while our model might predict them inaccurately or assign different probabilities.

V. Experiment Result And Error Analysis

A. *Fully Connected Neural Network*

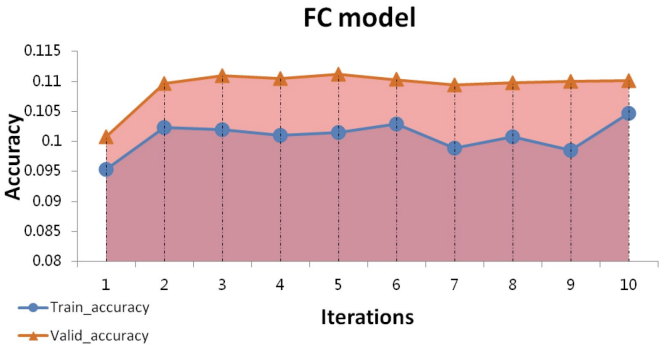

Fig.5. Train vs validation accuracy for fully connected neural network

Fig.5 shows the training and validation set accuracies for fully connected neural network. We trained our model with Adam optimizer with 0.001 step size. After 10 epochs, our model converges with around 10% validation accuracy. Fully connected Neural Network model implies high bias (around 10% accuracy for both sets) to our image classification task with kaggle score 6.8. We can observe that fully connected neural network is not good enough to classify drivers' behaviors. In fact, several precise classification of our image requires our model to detect more subtle details such as gestures and positions of drivers' hands, angle of their face, ect. By flattening image pixels into 1-D array will certainly not detect these details and thus our fully connected network failed to grasp important drivers' features. Since our fully connected neural network exhibits high bias, deeper and more complex model should be proposed in our next step.

B. *Basic Convolutional Neural Network*

In order to grasp subtle 2D features of the image, we build basic CNN model to solve this problem. The validation accuracy that we achieve for basic CNN model is about 60% with a log loss of 1.32 for the validation, which is much better than fully connected neural network. We plot the training and validation set accuracies in Fig.6.

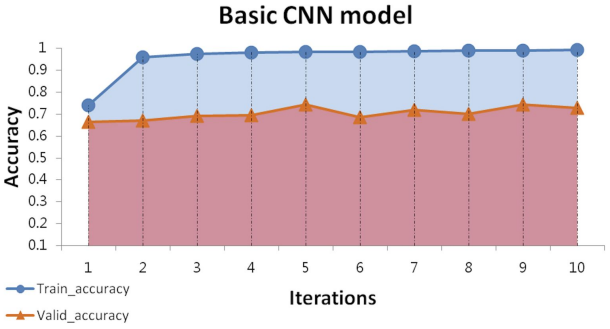

Fig 6. Training and validation accuracy for basic CNN

It is worth noticing that, among 10 training epochs, the training accuracy increase rapidly to around 99%, while the validation set accuracy remains at 74%. This is a clear evidence of overfitting. In order to further illustrate the performance of our basic CNN model, we also include the confusion matrix as shown below.

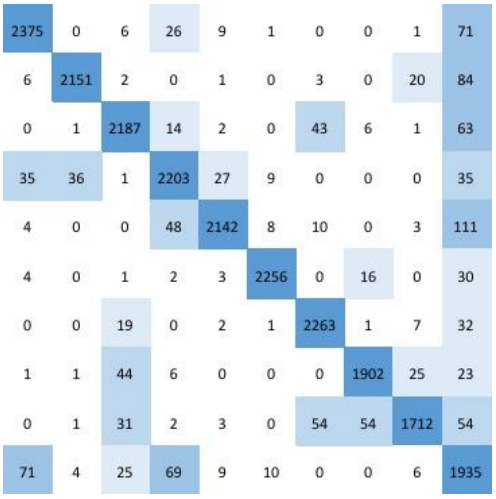

Fig. 7. Confusion matrix for basic CNN model

As we can observe from the confusion matrix(Fig.7), our basic CNN model is very bad at predicting c9 type image, which is labeled as "talking to passenger". Several images from other classes are mistakenly labeled as c9.

C. *VGG-16 Model*

We setup our pre-trained VGG-16 model with the last output layer removed. We first run a forward propagation through the VGG-16 model and save the results for future use. Then, we add and train the output layer with batch size of 64, learning rate 0.001, epoch of 15, and dropout of 0.8. The dropout layer will randomly drop 80% of the neurons during the training process for each iteration, and proves significantly reduce the overfitting issue in our experiments. The model converges quickly in about 10 epoches and then stabilize throughout the training process. We plot the accuracy for both training and valid set in Fig. 8, and the maximum accuracy in validation set is 85% with a log loss of 0.45.

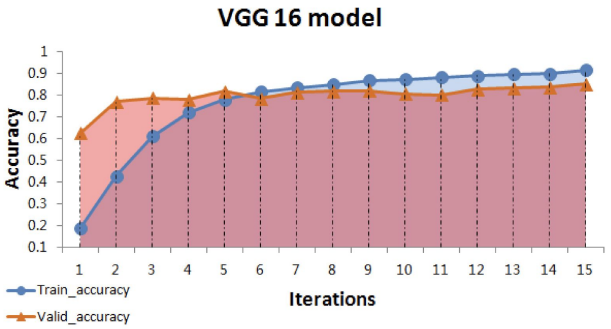

Fig. 8. Training and validation accuracy for VGG-16

The confusion matrix is shown in Fig. 9. We can conclude that our model will falsely classify images from other classes as "c0: safe driving". If we extract some images that are mislabelled as c0 in Fig. 10, some of the images cannot be even distinguished at a human performance level. One reason of mislabeling might be the incorrect focus of features. One of misclassified features might be the opening of drivers' mouth.

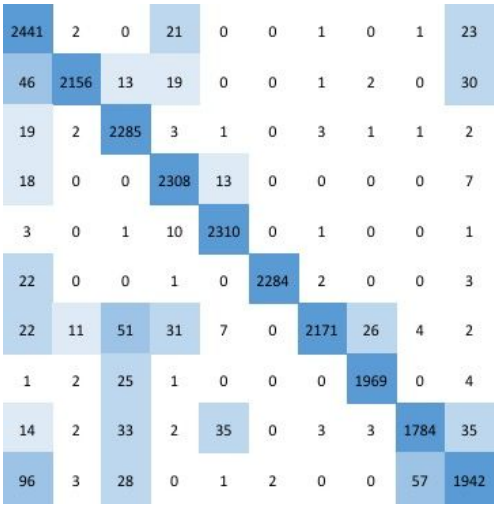

Fig. 9. Confusion matrix for VGG-16 model

Fig. 10. Images mislabelled as "safe driving"

In order to address on this phenomenon, We decide to apply K-nn to reduce the chance of misclassification, which is already discussed previously. We get the nearest k neighbors (7 neighbors in our case) in terms of pixel differences, calculate the weighted average, and overwrite each neighbor's probability with the average probability. As discussed in previous section, K-nn will obtain images with small Euclidean distance according to pixel values instead of just focusing one feature of the image. Fig. 6 provides an example of highly correlated images in test set. Thus, K-nn method can effectively solve the misclassification problem as VGG-16 did and improves the overall accuracy.

D. Ensembled Model: VGG-16+Inception v4+K-nn

Similar to VGG-16, we replace Inception v4's output layer with a 10 class softmax layer. We train the Inception v4 model with batch size of 64, learning rate of 0.001, and training epoch of 20. The predicted result is then averaged with that of VGG-16. Additionally, we include K-nn to update our result from VGG-16 and Inception v4. We achieve 88% accuracy and 0.45 logarithmic loss. Our kaggle score also has a dramatic drop from 0.64 to 0.50. We also include the confusion matrix as shown in Fig.11 to better illustrate our final result.

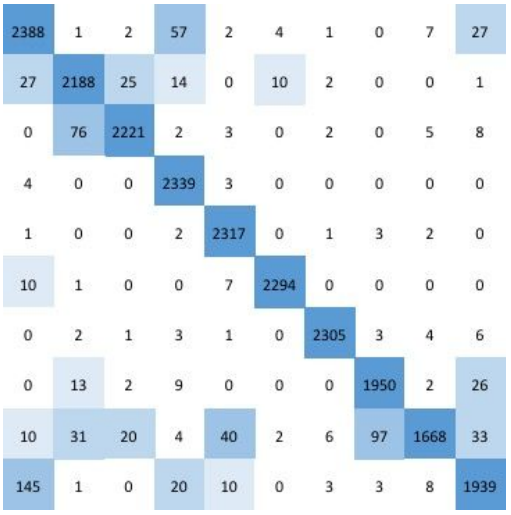

Fig. 11. Confusion matrix for VGG+Inception ensemble model

According to the confusion matrix above, we can observe that the mislabeling of c0 and c9 has been mitigated. Thus, the ensembled model is beneficial in solving mislabeling of c0 and c9 classes.

VI. Conclusion and Future Work

The table below summarizes the overall performance for each of our models. The best result that we have achieved is the ensembled models with Kaggle Score of 0.5067(Fig.12), which corresponds to top 20% among all submissions.

Table 1. Model performance summary

Models	Validation Accuracy	Validation Loss	Kaggle Score
FC	11.4%	6.1	6.8
Basic CNN	74.38%	1.2081	1.32
VGG-16	85.01%	0.4954	0.64
VGG-16 + Inception-v4 +KNN	88.65%	0.4521	0.50

Fig.12. Final Kaggle score

One of pre-trained models that we would like to try in the future will be ResNet-50, which has been shown having good performance in classifying distracted driver behaviors[14]. Another step that we would like to try in the future is data augmentation including color shifting and rotating. Since the test images are highly correlated and our training data set is small compared with test data set, it is beneficial to try data augmentation to avoid overfitting problem.

References

[1] World Health Organization: Global Status Report on Road Safety (2015), from http://www.who.int/violence_injury_prevention/publications/road_traffic/en/

[2] Stutts, J. C., Reinfurt, D. W., Staplin, L., & Rodgman, E. A. (2001). The role of driver distraction in traffic crashes.

[3] Distracted Driving. NHTSA. Retrieved October 20, 2017, from https://www.nhtsa.gov/risky-driving/distracted-driving

[4] U. D. o. H. &. H. Services, "Distracted Driving," 2016

[5] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2921-2929).

[6] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., & Oliva, A. (2014). Learning deep features for scene recognition using places database. *In Advances in neural information processing systems* (pp. 487-495).

[7] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint* arXiv:1409.1556.

[8] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[9] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In AAAI (pp. 4278-4284).

[10] Kaggle Competition: State Farm Distracted Driver Detection: https://www.kaggle.com/c/state-farm-distracted-driver-detection/data

[11] Sopyla, K. Tensorflow MNIST Convolutional Network Tutorial. Github repository. https://github.com/ksopyla/tensorflow-mnist-convnets

[12] Lee, H. Convolutional Neural-Network for MNIST, Github repository, https://github.com/hwalsuklee/tensorflow-mnist-cnn

[13] Majumdar, J. Inception-v4 Github repository, https://github.com/titu1994/Inception-v4

[14] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In AAAI (pp. 4278-4284).

[15] Felix Yu. (n.d.). Retrieved December 15, 2017, from https://flyyufelix.github.io/2016/10/11/kaggle-statefarm.html

Contribution

Each group member shares the fair amount of contribution to the project. Specifically, Liuming Zhao pre-processed the images and implemented the basic CNN model. Taiming Zhang and Lingzi Guo assisted on training and optimizing CNN model. Taiming Zhang is responsible for implementing VGG-16. Lingzi Guo selected and designed the next three models. All members provided critical feedback and helped shape the project and manuscript.