Distracted Driving Recognition: Classifying Safe and Unsafe Driving With Deep Learning

James Carroll, Aakarshan Dhakal
Department of Computer Science
Stanford University
carrollj@stanford.edu, adhakal@stanford.edu

Abstract

Distracted driving is an epidemic in the United States; every day about 1,000 people are injured and nine are killed from car accidents involving distracted driving [1]. Thanks to a data set collected from a State Farm Machine Learning Competition, we have access to dash-cam footage of safe and unsafe driving. Our project aims to use this dataset to help identify distracted driving with computer vision. In this project, we used three convolutional neural network models - a four-layer simple CNN, a pre-trained VGG-16 model [2], and a pre-trained Inception-v3 model [3] - to distinguish between safe and distracted driving. Overall, our models were unable to consistently recognize some distractions such as drinking or applying makeup, but we were able to identify whether or not drivers were texting or talking on the phone while driving with high accuracy.

1 Introduction

According to the National Highway Traffic Safety Association (NHSTA), distracted driving occurs when a driver engages in some activity that diverts their attention from the road [4]. Some common activities include using a mobile device, reaching for something in the car, applying makeup, or even talking to another passenger. The implications of distracted driving are clear: it is extremely dangerous for both the driver and other drivers on the road. Unfortunately, in some cases (about 3,450 per year), the consequences are fatal [4].

We believe that computer vision can be applied to this problem to help alleviate the consequences. Placing cameras within a car that alert drivers when distracted driving is detected will hold them accountable and strongly encourage safe driving habits. We experimented with three CNN models to try detecting distracted driving from still images. The input to our models were images of drivers who were either driving safely or engaging in one of nine distracting activities (outlined further in the 'Dataset' section). We then used a four-layer CNN modified from the class GitHub repository [5], a pre-trained VGG-16 network, and a pre-trained Inception-v3 network to predict whether a driver was safe or distracted. Initially, we output 10 different labels, one for safe driving and the other nine for the different distracting activities. However, our models were not good at handling 10 output classes. We ran further experiments where our model only classified 3 activities (safe driving, texting with right hand, talking with right hand) and 5 activities (safe driving, texting with either hand, talking with either hand). We were able to obtain much higher accuracy with these limited output classes.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

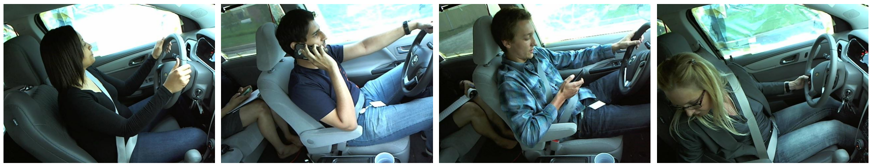
2 Related work

Deep learning researchers have been retraining Google's deep learning model called Inception-v3 [6] for image classification tasks. Andre Esteva and Brett Kuprel used Inception-v3 to classify different types of skin cancer [7]. They used Inception-v3 CNN architecture that was pre-trained on approximately 1.28 million images (1,000 object categories) from the 2014 ImageNet Large Scale Visual Recognition Challenge, and trained it on their dataset using transfer learning. Their CNN achieved performance on par with all tested experts across both tasks, demonstrating an artificial intelligence capable of classifying skin cancer with a level of competence comparable to dermatologists [7].

VGGNet is another widely used convolutional neural network model. Karen Simonyan and Andrew Zisserman's work on VGGNet has had an important contribution to large scale image recognition [8]. Through a evaluation of networks of increasing depths using an architecture with very small (3x3) convolutions filters, they were able to show significant improvements can be achieved by pushing the depth to 16-19 weight layers.

3 Dataset

Our dataset came from a Kaggle competition hosted by State Farm [9]. The dataset included 22,424 labeled images of drivers and 79,726 unlabeled test images (which were unlabeled as part of the competition). There were 26 drivers in total, with each driver appearing multiple times in each of the 10 image classes. The 10 classes were: 0) safe driving, 1) texting with right hand, 2) talking on the phone with right hand, 3) texting with left hand, 4) talking on the phone with left hand, 5) operating the radio, 6) drinking, 7) reaching behind, 8) fixing hair or makeup, 9) talking to a passenger. Below are examples of some of our dataset:



(a) Safe driving (b) Talking on phone (c) Texting (d) Reaching behind

Figure 1: Examples from State Farm dataset.

Since we were limited to using just the 22,424 labeled images, we decided to only use a training and validation set to accommodate this relatively small amount of data. As part of our data preprocessing, we randomly selected 3 of the 26 drivers to compose our validation set, so that drivers would not appear in both the training and validation sets. When we considered all 10 output classes, our training/validation split was 19,542 images to 2,882 images. For 5 classes, it was 10,265 images to 1,480 images. For 3 classes, it was 6,188 images to 885 images. The original images were 640x480, but to speed up training, we input 128x96 images to our four-layer CNN. Since both the VGG-16 and Inception-v3 models take in square images as input, we changed the resolution of our data to be 480x480 when testing theses models.

We initially had trouble with over-fitting the training set in our four-layer CNN, so we attempted data augmentation. Our augmented dataset included 5 new images for every original image (one crop, two rotations, two saturation changes) and randomized these image modifications (e.g. one of the rotations was randomized to be between 5 and 10 degrees).

4 Methods/Experiments

We began by using the CS230 project starter code with to come up with our baseline model. This model consisted of 4 convolutional layers, followed by two fully connected layers. Each convolutional layer included a 3x3 same convolution, a ReLU activation, and a 2x2 maxpool. The loss we used for this model was softmax cross-entropy. We modified the model's code to change the image resolution from square to 4:3 aspect ratio. When we first ran the model, we noticed a high over-fitting of the training data as there was big variance between the training set accuracy and development set accuracy. To fix this problem, we first tried data augmentation. We added code that included 5 augmented images for each image in the training dataset using two rotations, two saturation changes and one crop. When this did not yield good results, we also added code to experiment with L2 regularization and dropout. Our model still over-fit the training data, as training accuracy was above 99%, but the maximum development set accuracy we achieved was 78.6%.

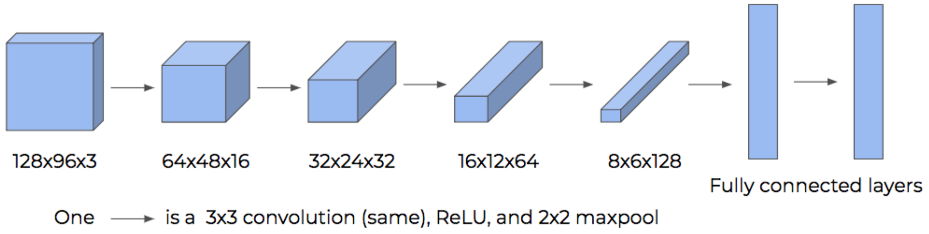


Figure 2: Basic 4-layer CNN model architecture

We thought of simplifying our task by classifying just three out of ten classes from our dataset. The classes we chose to classify were safe driving, texting with the right hand, and talking on the phone with the right hand. Testing this smaller dataset with the combination of data augmentation, L2 regularization, and dropout, yielded much better results, with the training set accuracy over 99% and the development set accuracy reaching a high of 97.3%.

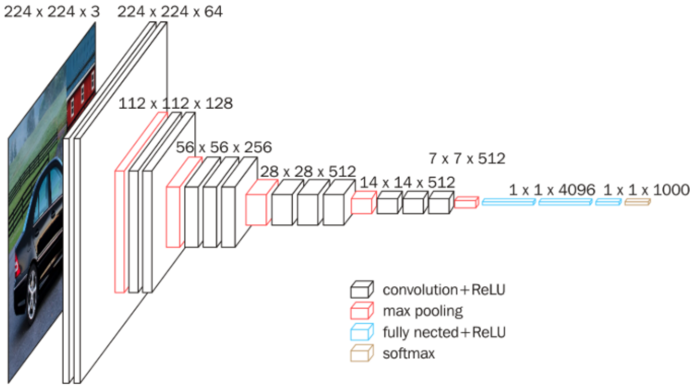


Figure 3: Pre-trained VGG-16 model architecture[8]

We then experimented with transfer learning to solve our over-fitting problem. We obtained a pre-trained VGG-16 model and continued to classify just three classes. Our loss function was still softmax cross-entropy for this model. We modified the code to fit our data's file structure layout, plus added code to save weights and log progress. We then retrained the last fully connected layer for 10 epochs, then trained the entire model for 10 epochs. With this new approach, our training set accuracy was still in the high 90s, but the maximum development set accuracy was just 83.6%. Due to computational limitations (we did not have access to GPUs), we were not able to run the VGG-16 model on all 10 classes in a reasonable amount of time.

We also tested a pre-trained Inception-v3 model on three classes. Again, the loss function we used was softmax cross-entropy. We modified the code to fit our data file structure and added code to calculate and log the validation set loss/accuracy. We also created scripts to parse through the resulting logs and extract the loss/accuracies at each training step. We retrained the last fully connected layer of this model and added a final softmax activation with the correct number of output classes. We initially ran the model on randomly-selected batches of 100 images for 10000 steps and obtained training set accuracies in the high 90s and development set accuracy of 93.0%. When we ran the model on full size batches for 500 epochs, our development set accuracy dipped to 92.1%. In general, we found that running the model on randomized small batches yielded a better validation accuracy then using full epochs.

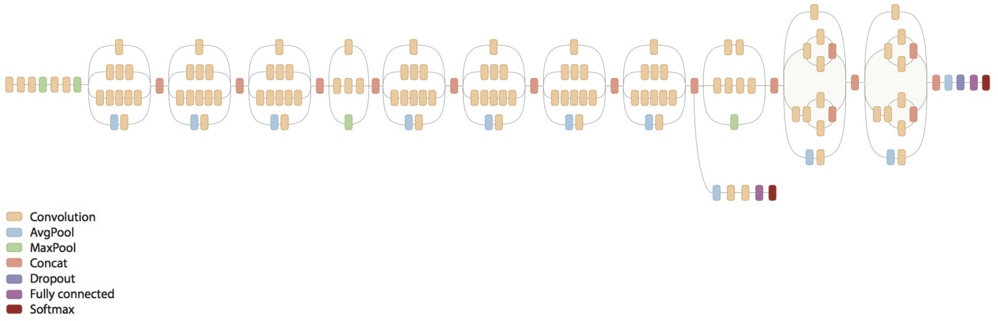


Figure 4: Pre-trained InceptionV3 model architecture [6]

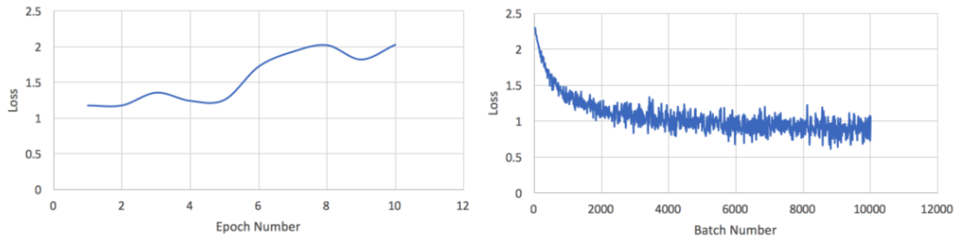
The Inception-v3 model was much faster than VGG-16 (because it caches results from the entire model, minus the last layer), so we tried classifying more than just three classes. We first attempted to classify all ten classes, to compare to the baseline model. When we tried running the model on small batches of 100 images for 10000 steps, we got training set accuracy in the mid 90s, but the development set accuracy was just 73.5%. We also ran the model on just five classes (texting/talking on the phone with either hand and safe driving), to build a phone-distraction detector, an application which we thought would still be pretty impactful. Running the model with data augmentation and random batches of 100 images for 10000 steps, we achieved training set accuracy in the high 90s and development set accuracy of 87.4%.

5 Results

Overall, our models did not produce the greatest results when attempting to classify all 10 classes, but did well when attempting to classify a subset of the classes. As mentioned before, we were only able to run the VGG-16 model on three classes (safe driving, texting/talking with right hand). Below are comparisons of the three models' performances on these three classes.

Model	Train Accuracy	Dev Accuracy	Loss	Epoch Runs
Basic CNN	99.9%	97.3%	0.074	10
VGG-16	98.9%	83.6%	-	20
Inception-v3	97.5%	92.1%	0.427523	500

When classifying 10 classes, both our baseline model and Inception-v3 models overfit the training set quite a bit. The validation loss was still very high at the end of the training process. These results are presented below:



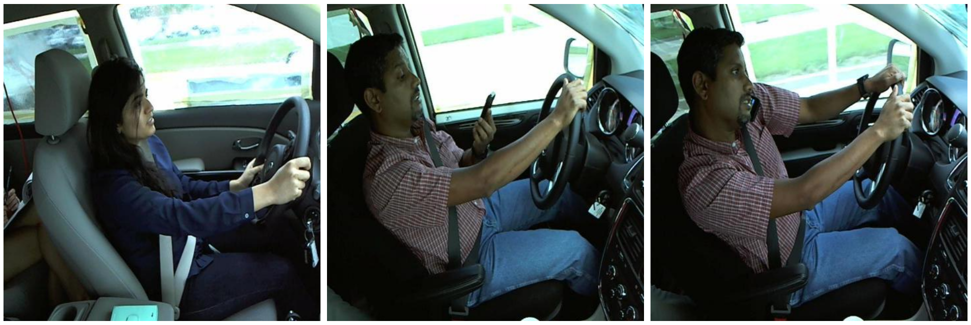
(a) Basic CNN

(b) Inception-v3

Figure 5: Loss considering 10 classes

We performed error analysis on our Inception-v3 model's mislabeled images when considering all 10 classes. The most common mistake by far was classifying talking to passengers as safe driving.

We believe that this error, and several other errors, were due to the fact that it is difficult even for a human to distinguish between some of the classes. For instance, in some still images, it is difficult to tell if a driver is paying attention to the road, or talking to someone. In some images, important classification-related items (such as a phone) are obscured from the camera view. Some of these incorrectly labeled images are presented below as examples:



(a) Label: Safe driving
Truth: Talking to passenger)

(b) Label: Texting
Truth: Talking on the phone

(c) Label: Safe driving
Truth: Talking on phone

Figure 6: Incorrectly labeled images

Since it is difficult for humans to classify these images at first glance, we were not surprised our model got these image labels incorrect.

We finally shifted our focus to identifying just 5 classes (safe driving, talking/texting with either hand) to build a phone distraction detector. With augmented data, we achieved a relatively low validation loss:

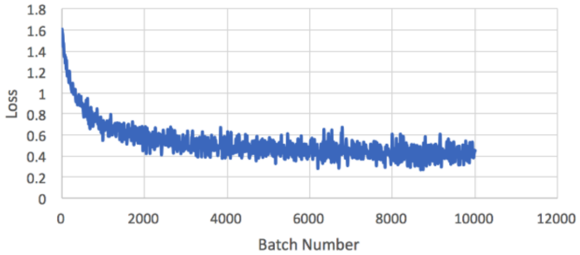


Figure 7: Inception-v3 Model - Loss considering 5 classes

6 Conclusion

The margin between our training set accuracy and development set accuracy was lower and our models over-fit the training set less when we ran our models on just three classes for the basic four layer CNN and VGG-16 models and on five classes for the Inception-v3 model. This margin was much higher when we ran our models on all ten classes. We have two explanations for this. First, accurately classifying all ten classes is a harder problem than accurately classifying just three or five classes. Second, we were limited by time and computational resources to run the basic four layer CNN and VGG-16 models on all ten classes. Of the three models we used, we found the Inception-v3 model to be the best balance of speed and accuracy. Since the Inception-v3 model ran much faster than the other two, we were able to run it on all ten classes for more iterations. Our results were still not as good when we ran the Inception-v3 model on ten class compared to when we ran it on five classes. While our models were not able to achieve high accuracy on all 10 classes, we were able to build a pretty good phone distraction detector. In the future, we look forward to accessing better computational resources to test the VGG-16 model on all 10 classes, as well as run all of our models for much longer with different hyperparameters.

7 Contributions

We both contributed equally to the project. We both worked on the basic four layer CNN model. Aakarshan focused on VGG-16 model, while James worked on Inception-v3 model.

8 References

[1] "Motor Vehicle Safety." Centers for Disease Control and Prevention, Centers for Disease Control and Prevention, 9 June 2017, www.cdc.gov/motorvehiclesafety/distracted_driving/

[2] Moindrot, Oliver. "Example TensorFlow Script for Fine-Tuning a VGG Model (Uses Tf.contrib.data)." GitHub, gist.github.com/omoindrot/dedc857cdc0e680dfb1be99762990c9c/.

[3] d'Almeida, Wisdom. "Image-Classification-Transfer-Learning." GitHub, github.com/wisdal/Image-classification-transfer-learning/blob/master/retrain.py.

[4] "Distracted Driving" NHTSA, NHTSA, 23 Apr. 2018, www.nhtsa.gov/risky-driving/distracted-driving.

[5] CS 230 Deep Learning. "Hand Signs Recognition with Tensorflow." GitHub, github.com/cs230-stanford/cs230-code-examples/tree/master/tensorflow/vision.

[6] "Train Your Own Image Classifier with Inception in TensorFlow." Google AI Blog, 9 Mar. 2016, ai.googleblog.com/2016/03/train-your-own-image-classifier-with.html.

[7] Esteva, Andre, et al. "Dermatologist-Level Classification of Skin Cancer with Deep Neural Networks." Nature, vol. 542, no. 7639, 2017, pp. 115–118., doi:10.1038/nature21056.

[8] Simonyan, Karen and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." CoRR (2014). arXiv. Web. 4 Jun 2018, http://arxiv.org/abs/1409.1556.

[9] "State Farm Distracted Driver Detection | Kaggle." Countries of the World | Kaggle, www.kaggle.com/c/state-farm-distracted-driver-detection/data.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going Deeper with Convolutions." CoRR (2014). arXiv. Web. 3 Jun 2018, arxiv.org/abs/1409.4842.