Driver Distraction Classification Using Transfer Learning on Computer Vision

James Lai
School of Continuing Studies
York University
Toronto, Canada

Lingling Zhang
School of Continuing Studies
York University
Toronto, Canada

Iman Lau
School of Continuing Studies
York University
Toronto, Canada

L. Yang
School of Continuing Studies
York University
Toronto, Canada

***Abstract*—When drivers are distracted, they are less likely to see potential issues on the road, which can cause accidents. Using computer vision and transfer learning, we aim to classify driver status. Our research applied transfer learning using pre-trained models and ensemble learning techniques to build a meta-model. In the task of classifying driver status into 10 categories - safe driving and 9 different types of distractions, our final model was able to reach a loss score of 0.20022 on Kaggle (top 4.4%).**

Keywords—Distracted Drivers, Computer Vision, Deep Learning, Neural Networks, Transfer Learning, Ensemble Learning

I. Introduction

This paper documents the methodologies and results for our submissions to the State Farm Distracted Driver Detection competition hosted on Kaggle.

Driver distraction is a major concern for road safety. To detect driver inattention, different techniques have experimented on different types of data in previous research.

As early as 2001, [1] applied neural networks in the detection of phone-use related driver distraction. In 2007, [2] used neural networks to track a driver's eye movement in image sequences to learn the normal behavior for each driver.

Another popular technique is the support-vector machine (SVM). In 2007, [3] applied rule-based and SVM classification methods to detect a driver's visual and cognitive workload by fusing stereo vision and lane tracking data. In the same year, [4] applied SVMs to develop a real-time approach for detecting cognitive distractions based on drivers' eye movements and driving performance data.

In 2011, [5] implemented long short-term memory (LSTM) recurrent neural networks in the detection of driver inattention based on driving and head-tracking data. The research was able to get an accuracy of up to 96.6%.

With new developments in deep learning methodologies on computer vision, advances in recognizing drivers' driving status are possible. Classification can be performed on images that capture not only the drivers' heads and faces but also their poses and environment of their sitting areas.

To achieve this, we obtained data from a Kaggle competition, which has 10 classes of driving statuses: safe driving and 9 different types of distractions. We implemented transfer learning and ensemble learning methods in this research.

II. Methodology

A. Data Collection and Exploration

a) Data Collection: The data set was obtained from a Kaggle competition called State Farm Distracted Driver Detection[1]. State Farm included a disclaimer that the images were collected in a controlled environment, where distracted drivers were not really driving.

b) Data Exploration: The data contained 22,424 images in the training set and 79,726 images in the test set.

Classes. The training set was labeled into 10 classes and the test set was not labeled. Indexes of the 10 classes are detailed in Table I.

The training set came with a driver_imgs_list.csv file of 3 attributes: subject (driver ID); classname (class index); img (unique image ID). There was no missing value in this file.

TABLE I. Distribution of Classes in Training Set

Class Index	Class	Count
c0	safe driving	2489
c3	texting - left	2346
c4	talking on the phone - left	2326
c6	drinking	2325
c2	talking on the phone - right	2317
c5	operating the radio	2312
c1	texting - right	2267
c9	talking to passenger	2129
c7	reaching behind	2002
c8	hair and makeup	1911

[1] https://www.kaggle.com/c/state-farm-distracted-driver-detection

In the training set, the 10 classes were relatively evenly distributed as shown in Table I. Therefore, no downsampling or upsampling treatment was needed in the training process.

Images. We manually examined some images in the training and test data sets. The images seemed to be taken from the same angle: the upper right corner the car. The driver's head (usually the right side of the face), arms, hands, body, and seat area were captured by the image. Fig 1 shows some sample images from the training set.

Fig. 1. Sample images. (Row one from left to right: class C0 and C3; Row two from left to right: class C4 and C6)

Subject. By analyzing the distribution of subjects (driver ID) in the training set with the driver_imgs_list.csv file, we found that images in the training set were taken with 26 drivers. The number of images of individual driver ranged from 346 to 1237.

However, the test set does not contain a CSV file with subject information. Therefore, this classification task was subject-independent. Furthermore, the training and test sets were split between the drivers, meaning that one driver would only appear on either the train or test set.

B. Machine Learning Pipeline

The machine learning pipeline we built for this problem is described in 2 phases. In Phase 1, pre-trained models from the ImageNet were downloaded and fine-tuned on the training data set. In Phase 2, ensembling was applied using models trained in Phase 1 so as to improve the models' generalization capability on unseen data.

a) Phase 1. Pre-trained Models and Data Processing

Pre-trained Model Selection. There are a number of pre-trained models and weights available in the Keras framework which can be downloaded and fine-tuned. We selected pre-trained models based on two criteria: First, the models should have relatively high scores in ImageNet's annual contest - Large Scale Visual Recognition Challenge (LSVRC). Second, the models should have a relatively small amount of tunable parameters. The first criterion was to ensure the pre-trained weights learned enough from the ImageNet database while the second one was to allow us to go with smaller GPU memory which would be more economical.

As a result, InceptionV3 [6] from the keras.applications.inception_v3 library and ResNet50 [7] from the keras.applications.resnet50 library were selected. Images were downscaled to 299 x 299 pixels and 224 x 224 respectively as we applied the two pre-trained models.

Optimizer Selection. Stochastic gradient descent (SGD) and Adam were selected as optimizers.

Pre-trained Model-Optimizer Combinations. In order to compare the performance of both of the pre-trained models and both of the optimizers with the other factor controlled, we designed 3 pairs of Pre-trained Model-Optimizer combination:

Pair 1: InceptionV3 + SGD.

Pair 2: ResNet50 + SGD.

Pair 3: ResNet50 + Adam.

All 3 pairs were trained for 30 epochs and a (starting) learning rate of 0.0001.

Image Augmentation. Image augmentation on the training set was performed randomly on the fly during the model training process. For one particular training image, 1 augmented image was generated.

Cross-Validation. Five-fold cross-validation was implemented in the training process. It served 2 functions in the pipeline: model evaluation and submodels generating.

The submodels trained on different cross-validation folds were saved with an aim that we could opt to ensemble them in phase 2.

Models Trained on Full Data Set. After cross-validation, pre-trained networks were used to train models on the full training data set.

Therefore, each time we trained models with the pre-trained networks, a total of 6 base models would be generated (5 submodels from cross-validation + 1 model trained on full data set).

b) Phase 2. Ensembling

We implemented the following procedures as we ensembled the base models obtained from phase 1.

Image Augmentation. For one particular test image, 5 random augmented images were generated using the same image augmentation method performed in phase 1. Then each base model would generate a prediction for each of the 5 augmented images. Those 5 predictions were given the same

weight and averaged out to serve as the prediction of this particular image.

K-Nearest Neighbors (KNN). After image augmentation, KNN was applied to find each test image's closest neighbors. We applied this technique based on our observation and assumption from the training data set - the images have been taken continuously in time domain. For one particular test image, its prediction was averaged with the predictions of its neighbors. We expected that KNN could reduce noise resulted from overfitting.

KNN was also performed on the training set so as to evaluate the accuracy of KNN on this task and to decide on the number of neighbors should be chosen for averaging the prediction of test images.

We evaluated KNN accuracy on the training set based on the following rule: for a particular training image, we would consider one of its neighbors was predicted correctly only if both the neighbor's 'subject' (driver ID) and its class index match with that of the training image. Table II lists KNN's accuracy on the first N-nearest neighbors. For example, the accuracy of the "first 2" nearest neighbors stands for the number of instances in the training data whose both the first and the second nearest neighbors were correctly predicted.

Averaging Ensemble. Our ensembling strategy was designed in a way that the effectiveness of five machine learning and deep learning techniques - pre-trained model, optimizer, image augmentation, KNN and ensembling can be examined with the other four techniques controlled. As a result, dozens of attempts were tried as documented in Table IV and Table VI. Along our experiment path, we observed 2 submission candidates.

Candidate 1 was obtained by averaging out the predictions from all base models trained by *Pair 1* and *Pair 2* of the Pre-trained Model-Optimizer Combination, with image augmentation and KNN treatment applied.

Candidate 2 was obtained by ensembling *Candidate 1* with the best-performed models trained by *Pair 3* of the Pre-trained Model-Optimizer Combinations.

In summary, in *Candidate 1*, one single test image would be averaged from a total of 300 predictions (12 models * 5 augmented images * 5-nearest neighbors); in *Candidate 2,* one single test image would be derived by averaging from even more predictions.

C. Model Evaluation

We first evaluated the two fine-tuned pre-trained models using cross-validation to compare their performance. After that, we documented the public and private leaderboard scores to present the effectiveness of each ensembling technique we used in the project.

III. Results

A. Cross-validation results on training data

The loss and accuracy of *Pair 1* and *Pair 2* of the Pre-trained Model-Optimizer Combination with randomly augmented images during the cross-validation process are shown in Table III.

Results of *Pair 3* of the Pre-trained Model-Optimizer Combination are listed in Table V. Note that in our first run of *Pair 3,* we observed that the cross-validation loss variance was high. This was particularly obvious comparing CV0 with CV1. Therefore, Try 2 of *Pair 3* was performed. It was obvious that CV0 from *Pair 3* Try 1 and CV2 and CV4 from *Pair 3* Try 2 yielded very good results and they can be applied in ensembling.

B. Ensembling

Results on the test set measured by the 5 dimensions are listed in Table IV and Table VI.

Our final attempt - *Candidate 2* achieved a Kaggle private leaderboard score of 0.20022, which ranked 63 out of the total 1440 submissions (top 4.4%).

TABLE II. KNN Accuracy on Training Set by Neighbours

First N Nearest Neighbors	Accuracy of All Neighbors Predicted Correctly (Percentage)
first 1	1.000000
first 2	0.996878
first 3	0.992062
first 4	0.986577
first 5	0.980200
first 6	0.973065
first 7	0.965305
first 8	0.956029
first 9	0.947021
first 10	0.939083
first 11	0.931056

IV. Discussions

Overall we achieved good results with our method, which we can see when we examine the results in Table IV and Table VI. The ensembling techniques we applied improved the performance of image classification models.

A. Pre-trained models

In general, ResNet50 performed a little bit better than InceptionV3 when all other dimensions being equal. ResNet50 was the winner of image classification in ILSVRC 2015 while InceptionV3 was the first runner up in the same year. Our tests

TABLE III. CROSS-VALIDATION RESULTS OF MODELS TRAINED FROM INCEPTIONV3 AND RESNET50 WITH SGD ON TRAINING SET

Pair Index	Model Optimizer	Result	CV0	CV1	CV2	CV3	CV4	mean	std
Pair 1	InceptionV3 SGD	train_loss	0.004741	0.004954	0.006126	0.004986	0.004745	**0.005110**	**0.000518**
		train_acc	0.999219	0.999275	0.998885	0.999275	0.999108	0.999153	0.000147
		val-loss	0.021353	0.022775	0.019593	0.026735	0.021053	**0.022302**	**0.002436**
		val-acc	0.993541	0.995097	0.993977	0.993531	0.993528	0.993935	0.000606
Pair 2	ResNet50 SGD	train_loss	0.007156	0.006682	0.006466	0.006349	0.005758	**0.006482**	**0.000455**
		train_acc	0.998439	0.998327	0.998662	0.998718	0.998830	0.998595	0.000185
		val-loss	0.019599	0.022456	0.021639	0.026064	0.022333	**0.022418**	**0.002091**
		val-acc	0.994209	0.994428	0.994200	0.993977	0.994421	0.994247	0.000167

TABLE IV. MODEL RESULTS AND RANKING ON KAGGLE (I)

Model Optimizer		Training Method		Test Data Preprocessing		InceptionV3 SGD		ResNet50 SGD	
Group	Index	Training set	No. of Model	Augmentation	KNN	Private Board Score	Public Board Score	Private Board Score	Public Board Score
1	1 - A	cv0	1	N	N	0.48396	0.65042	0.44329	0.55615
	1 - B	cv0	1	Y	N	0.40052	0.52621	0.35607	0.46736
2	2 - A	full	1	N	N	0.43700	0.59318	0.47368	0.45343
	2 - B	full	1	Y	N	0.36239	0.44067	0.36663	0.39720
3	3 - A	cv0, cv1, cv2, cv3, cv4	5	N	N	0.34215	0.47330	0.35722	0.40510
	3 - B	cv0, cv1, cv2, cv3, cv4	5	Y	N	0.32889	0.43142	0.31650	0.36871
4	4 - A	cv0, cv1, cv2, cv3, cv4, full	6	N	N	0.33477	0.46454	0.35183	0.39345
	4 - B	cv0, cv1, cv2, cv3, cv4, full	6	Y	N	0.32322	0.42092	0.31386	0.36483
	4 - C	cv0, cv1, cv2, cv3, cv4, full	6	Y	First 5	0.28109	0.37298	0.27728	0.32967
		Training Method		Test Data Preprocessing		InceptionV3 SGD + ResNet50 SGD			
		Training set	No. of Model	Augmentation	KNN	Private Board Score Ranking (Top%)		Public Board Score Ranking (Top%)	
5	5 - A	cv0, cv1, cv2, cv3, cv4, full	12	Y	First 5	0.26251 152 10.6%		0.32938 216 15.0%	

TABLE V. CROSS-VALIDATION RESULTS OF MODELS TRAINED FROM 2 RUNS OF RESNET50 WITH ADAM ON TRAINING SET

Pair Index	Model Optimizer	Result	CV0	CV1	CV2	CV3	CV4	mean	std
Pair 3 Try1	ResNet50 Adam	train_loss	0.002614	0.162013	0.009556	0.006231	0.028656	**0.041814**	**0.060769**
		train_acc	0.999052	0.967442	0.995931	0.998328	0.991194	0.990389	0.011798
		val-loss	0.016337	0.253276	0.026917	0.024897	0.046944	**0.073674**	**0.090360**
		val-acc	0.995100	0.953644	0.993308	0.995539	0.987280	0.984974	0.015941
Pair 3 Try 2	ResNet50 Adam	train_loss	0.013737	0.015116	0.001446	0.005611	0.010234	**0.009229**	**0.005092**
		train_acc	0.996208	0.996934	0.999554	0.997993	0.997269	0.997592	0.001137
		val-loss	0.023318	0.027315	0.017609	0.025737	0.017620	**0.022320**	**0.004048**
		val-acc	0.993764	0.993314	0.996877	0.993308	0.995983	0.994649	0.001490

TABLE VI. MODEL RESULTS AND RANKING ON KAGGLE (II)

Model Optimizer		Training Method	Test Data Preprocessing		ResNet50 Adam Try 1 CV0		ResNet50 Adam Try 2 CV2		ResNet50 Adam Try 2 CV4	
Group	Index	No. of Model	Augmentation	KNN	Private Board Score	Public Board Score	Private Board Score	Public Board Score	Private Board Score	Public Board Score
6	6 - A	1	Y	First 5	0.22537	0.22082	0.23779	0.24154	0.30331	0.29786
	6 - B	1	Y	First 10	0.22006	0.21145	0.22799	0.23202	0.29710	0.28737
					CV0 + Candidate 1		**CV2 + Candidate 1**		**CV4 + Candidate 1**	
7	7 - A	13	Y	First 10	0.21654	0.24366	0.21798	0.25559	0.25535	0.27775
					CV0 + CV2 + Candidate 1					
					Private Board Score Ranking (Top%)		**Public Board Score Ranking (Top%)**		NA	
8	8 - A	14	Y	First 10	0.20022 63 (4.4%)		0.21994 103 (7.2%)			
					CV0 + CV2 + CV 4 + Candidate 1					
9	9 - A	15	Y	First 10	**Private Board Score**: 0.23013 | **Public Board Score**: 0.23067					

proved the usefulness of residual learning, which tries to learn some residual instead of trying to learn some features.

B. Optimizer

Comparing cross validation results of *Pair 3* (Table V) with that of *Pair 2* (Table III), performance of model with SGD as optimizer seemed to be more stable than that of Adam. However, while training with Adam, we obtained some submodels with particular good result. That maybe due to the nature of Adam.

C. Image Augmentation

In Groups 1 to 4, all models in Experiment B out-performed models in Experiment A, which showed the effectiveness of image augmentation. In almost all experiments, our models' performance on Kaggle's private leadership board was better than that of public leadership board. This demonstrated that our models were well generalized. Image augmentation played a key role in preventing overfitting.

D. Ensemble Learning

Ensembling was shown to be able to resolve overfitting in our experiments. Before ensembling, most single models seemed a bit over-fitted when we compare the validation loss with the training loss listed in Table III. After we conducted ensembling, private score continued to be lower than public score.

We also observed that as we added on the number of models from Groups 1 and 2 (single model) to Group 5 (12 models ensembled), model performance continued to improve.

E. KNN

KNN greatly boosted our models' performance. This was demonstrated when comparing Experiment 4-B with Experiment 4-C: by applying KNN to the same model, loss on private leadership score decreased by 13.03%.

F. Transfer Learning

Applying transfer learning yielded satisfactory results. Using pre-trained models also reduces the computation time.

V. Conclusions

Distracted driving is hazardous to drivers as well as cyclists and pedestrians. We conducted 5-dimensional experiments to understand the effectiveness of different machine learning and deep learning technologies (ensembling, KNN, image augmentation, different pre-trained models and optimizers) in image classification to determine whether a driver is driving safely or is distracted in 9 categories.

Using transfer learning, image augmentation, ensembling, and KNN, our final model reached top 4.4% among all submissions on Kaggle, which is a laudable achievement. Improvements, as always, can be made, although they would likely increase the demand for computational resources.

There are three action items that we think may further improve the ranking.

First, we only took a simple averaging of all predictions as the ensembling method, which is equivalent to giving the same weight to each prediction. It would be interesting to see the performance in each class by each of the models, and then to manually assign a weight to predictions from each model based on that observation in the performance.

Second, we only used 2 pre-trained models, even though it is possible to train more and ensemble them to further reduce overfitting. While we attempted to train a DenseNet121, it ran into memory issues, potentially due to the setup.

Finally, the fact that some submodels trained with Adam as optimizer yielded extremely good results was an interesting discovery. Further investigation on the performance of Adam might lead to improvement of our final submission.

Acknowledgment

We would like to thank our ML1020: Machine Learning at Scale course instructor, Karthik Kuber, for his guidance.

References

[1] D. de Waard, K. A. Brookhuis, and N. Hernandez-Gress, "The feasibility
of detecting phone-use related driver distraction," Int. J. Vehicle Des., vol. 26, no. 1, pp. 85–95, 2001.

[2] T. D'Orazio, M. Leo, C. Guaragnella, and A. Distante, "A visual approach
for driver inattention detection," Pattern Recognit., vol. 40, no. 8, pp. 2341–2355, Aug. 2007.

[3] Kutila, M., Jokela, M., Markkula, G., & Rué, M. R. (2007, September). Driver distraction detection with a camera vision system. In 2007 IEEE International Conference on Image Processing (Vol. 6, pp. VI-201). IEEE.

[4] Y. Liang, M. L. Reyes, and J. D. Lee, "Real-time detection of driver cognitive distraction using support vector machines," IEEE Trans. Intell. Transp. Syst., vol. 8, no. 2, pp. 340–350, Jun. 2007.

[5] Wollmer, M., Blaschke, C., Schindl, T., Schuller, B., Farber, B., Mayer, S., & Trefflich, B. (2011). Online driver distraction detection using long short-term memory. IEEE Transactions on Intelligent Transportation Systems, 12(2), 574-582.

[6] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

[7] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[8] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).