Driver Distraction Image Classification

A Five-Dimensional Evaluation of Machine Learning and Deep Learning Techniques

James Lai, Lingling Zhang, Iman Lau, L. Yang

Final standing

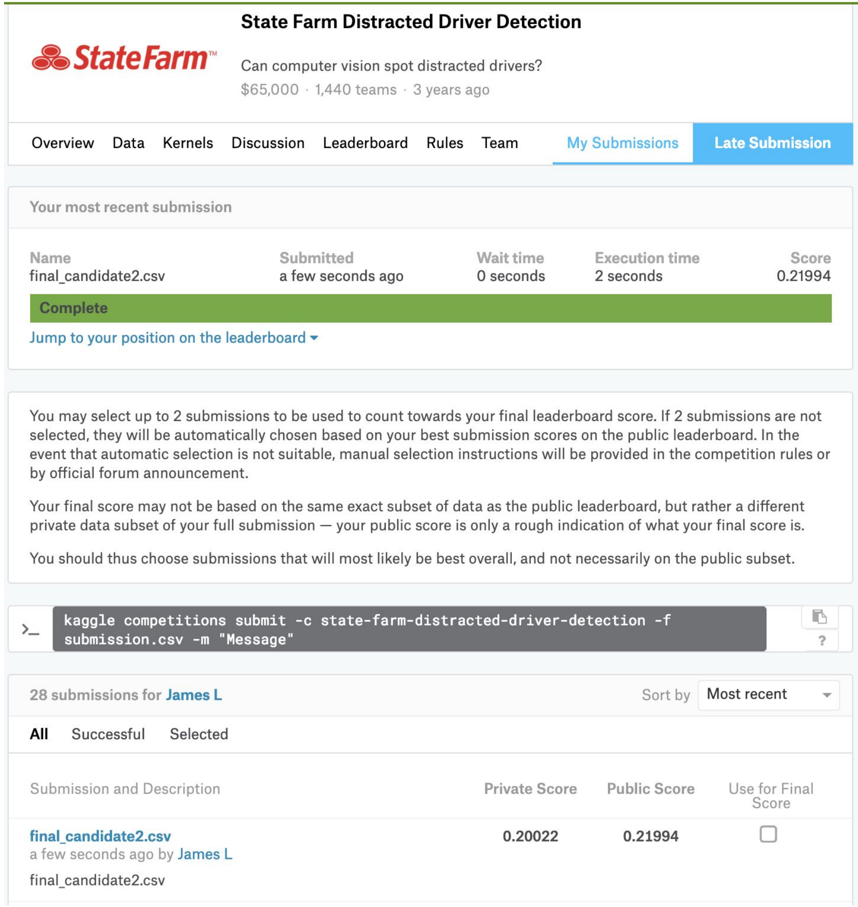
Ensemble of
12 models by SGD + 2 models by Adam

- Private Leaderboard score 0.20022

  63 out of 1440 teams (Top 4.4%)

- Public Leaderboard score 0.21994

  103 out of 1440 teams (Top 7.2%)



Outline

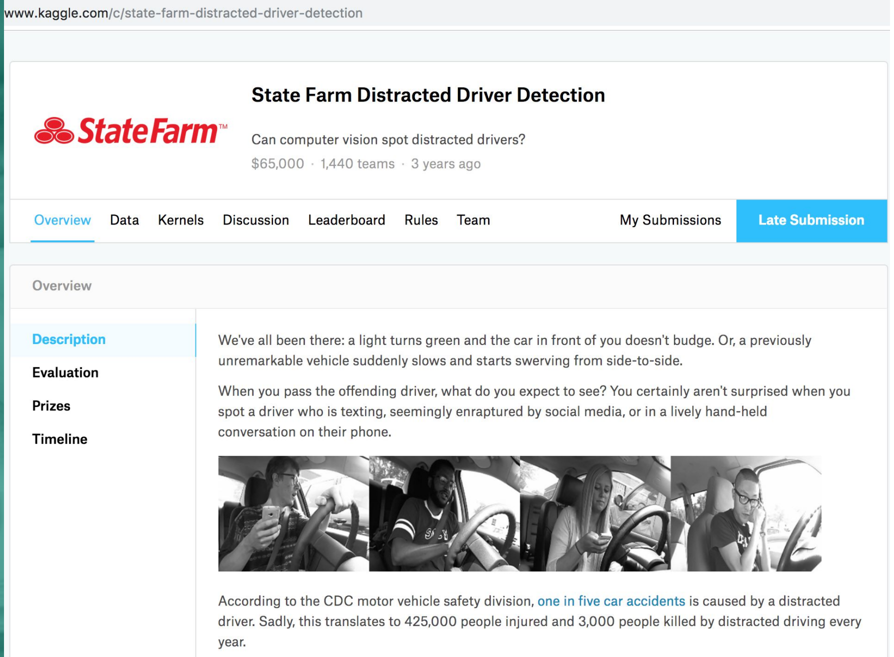
Problem Statement

Distracted driving is a major concern for road safety. Distracted drivers are less likely to see potential issues on the road, and they can become issues themselves, leading to accidents.

Kaggle Competition

The data set was obtained from a Kaggle competition called State Farm Distracted Driver Detection.

Training dataset contained 22,424 images where 79,726 images in the test set.


www.kaggle.com/c/state-farm-distracted-driver-detection

StateFarm™

State Farm Distracted Driver Detection

Can computer vision spot distracted drivers?

$65,000 · 1,440 teams · 3 years ago

Overview Data Kernels Discussion Leaderboard Rules Team My Submissions Late Submission

Overview

Description

Evaluation

Prizes

Timeline

We've all been there: a light turns green and the car in front of you doesn't budge. Or, a previously unremarkable vehicle suddenly slows and starts swerving from side-to-side.

When you pass the offending driver, what do you expect to see? You certainly aren't surprised when you spot a driver who is texting, seemingly enraptured by social media, or in a lively hand-held conversation on their phone.

According to the CDC motor vehicle safety division, one in five car accidents is caused by a distracted driver. Sadly, this translates to 425,000 people injured and 3,000 people killed by distracted driving every year.

10 Classes

The training set was labeled into 10 classes and relatively evenly distributed,

Therefore, no Downsampling or Upsampling treatment was needed in the training process.

Index	Class
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

Data

- Training: 22,424 images, 1 csv file

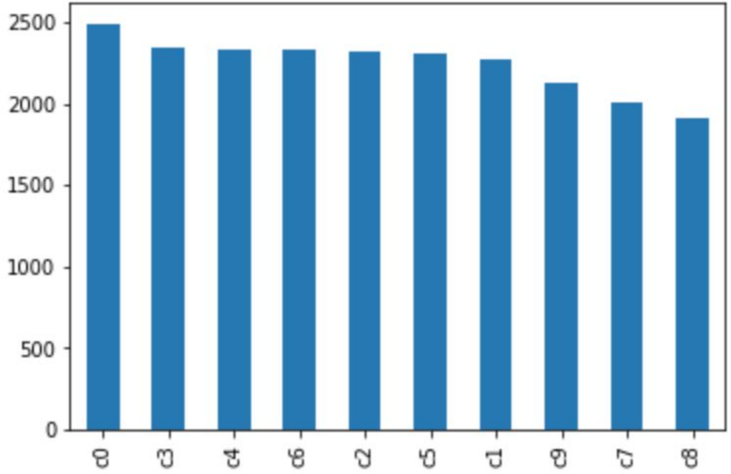
subject	classname	img
p002	c0	img_44733.jpg
p002	c0	img_72999.jpg
………..	………..	………..
p012	c1	img_45632.jpg
p012	c1	img_26825.jpg
……..	……..	……..
p014	c9	img_73160.jpg
p014	c9	img_17772.jpg
………..	………..	………..

- Test: 79,726 images

Data

Training: Classes

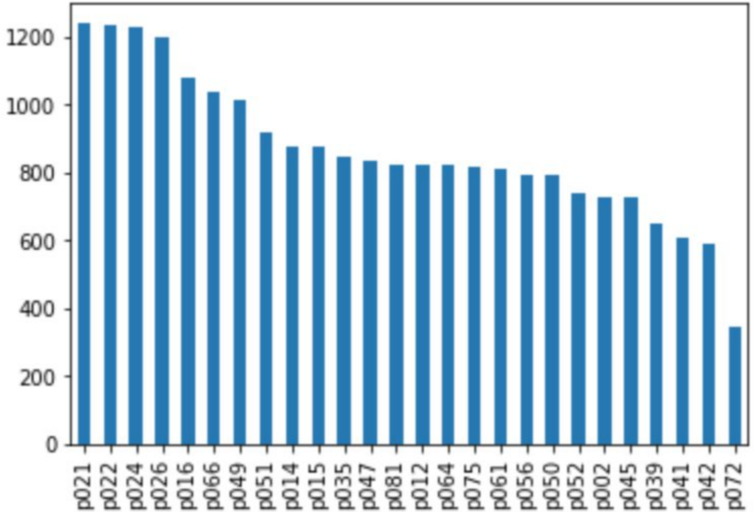
- 10 classes relatively evenly distributed

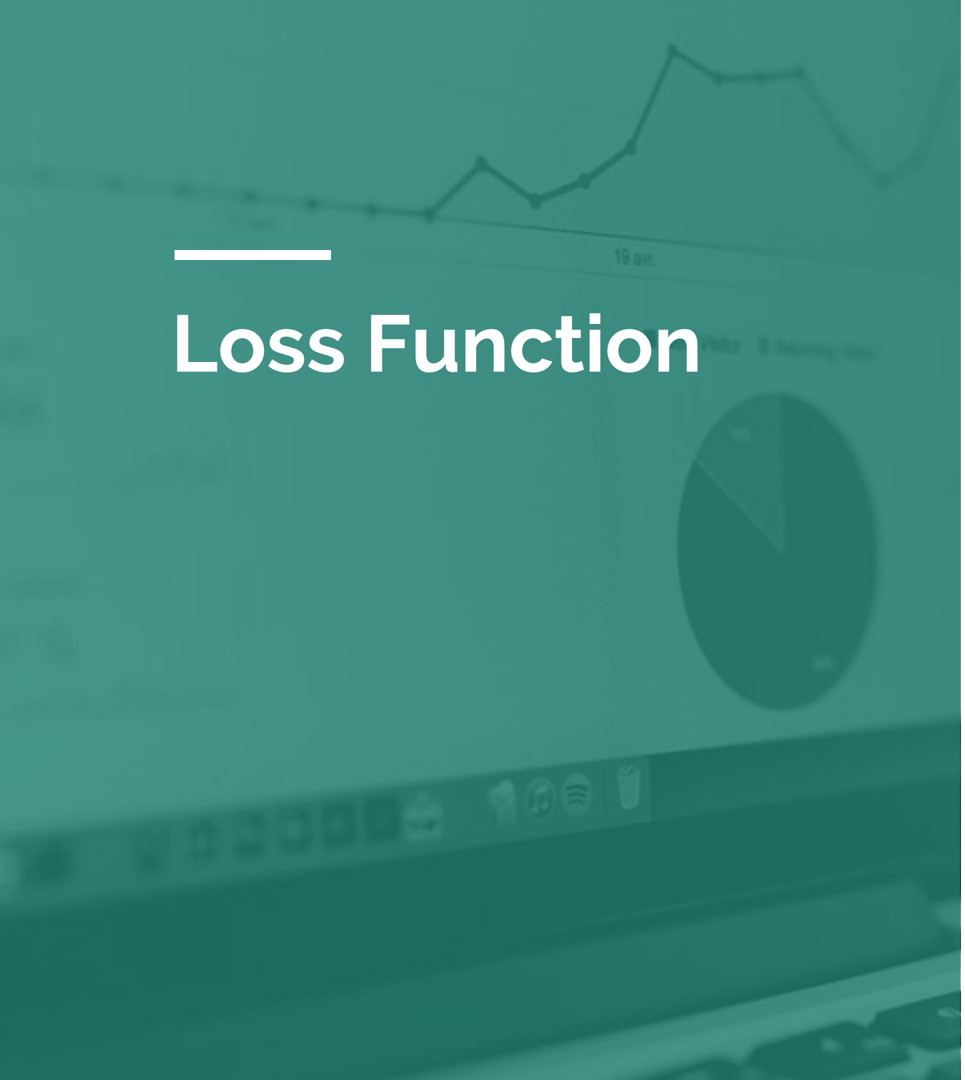


Data

Training: Drivers

- 26 drivers, whose images ranged from 346 to 1237





Loss Function

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

Where:

- N is the number of images in the test set
- M is the number of image class labels
- y_{ij} is 1 if an observation *i* belongs to class *j*
- p_{ij} is the predicted probability that an observation *i* belongs to class *j*

Machine Learning Pipeline

Machine Learning Pipeline

1. Fine-tune pretrained models - InceptionV3 and ResNet50 - on the training set.
    a. Train K models from K-fold cross-validation, plus one additional model trained on whole dataset. Images in the training set is randomly augmented during training. Adam optimizer is also evaluated with ResNet50.
    b. Compare cross-validation loss and accuracy
2. Apply various ensembling technique to reduce overfit.
    a. Ensemble K+1 models trained on 1st step.
    b. Ensemble predictions generated from randomly augmented images.
    c. Ensemble predictions from K nearest neighbours.
    d. Ensemble the two group of models - InceptionV3 and ResNet50.

The “ensembling” mentioned here is simple averaging, giving equal weights to predictions from all kinds.

Pre-trained Weights & Models

- We choose ResNet50 and InceptionV3. These two models have relatively small amount of parameters and high score on image-net competition.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159

*This table is taken from Keras's website.

Cross-Validations

- We use 5-fold cross validations to compare the performance between InceptionV3 and ResNet50.
- However, we keep the 5 models resulting from the 5-fold cross validations. They are treated as slightly different models and we will ensemble their outputs.
- An additional model is trained on the whole dataset for both InceptionV3 and ResNet50.

Image Augmentation

- The following augmentations are used in this project: rescale, shear, zoom, rotation, width Shift, height Shift, horizontal Flip
- During the models' training phase, image augmentations are randomly applied to the image in training set.
- When generating predictions for the test set, 5 augmented images are generated from a single test image. Predictions for the 5 images are averaged out as an ensembling method.
    - We also compare the performance without augmentation for test set, see later discussion.

K Nearest Neighbor

- For a single image in the test set, we used KNN to find K nearest neighbours from within the test set.
  - The image are first converted to a (40x30x3) image. Then a KNN model based on L2 distance is fit on the test dataset.
  - We also tried using the output from the second-to-last layers of InceptionV3, but we ran out of CPU memory when fitting the dataset using KNN (KNN is an in-memory algorithm).
- Predictions for the K nearest neighbours are averaged to produce a single prediction, as an ensembling technique.

The underlying assumptions are the test images are taken continuously. And the ones that are adjacent in time domain can help remove the noise from the model overfitting the training set.

K Nearest Neighbor

We also evaluate how accurate the KNN algorithm based on 40x30x3 images would be.

- Accuracy is defined as the percentage of rows where **both the subject and class** match with target image.

  For example, for “first 2” (nearest neighbors)’s accuracy, the accuracy is the percentage of rows where both the 1st and 2nd nearest neighbors belong to the same subject (a subject is the person) and the same class (the class is the person’s behaviour in the image, driving safe/looking at phones, etc.)

- The first neighbor always matches with target image. The accuracy drops to 93% when we look at 11 nearest neighbors.

KNN Accuracy on Training Set by Neighbors

Nearest Neighbors	Accuracy of All Neighbors Predicted Correctly
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

K Nearest Neighbor

- In this project we evaluated using either 5 nearest neighbors or 10 nearest neighbors, and choose based on the results' private and public leaderboard score.

Ensembling Workflow

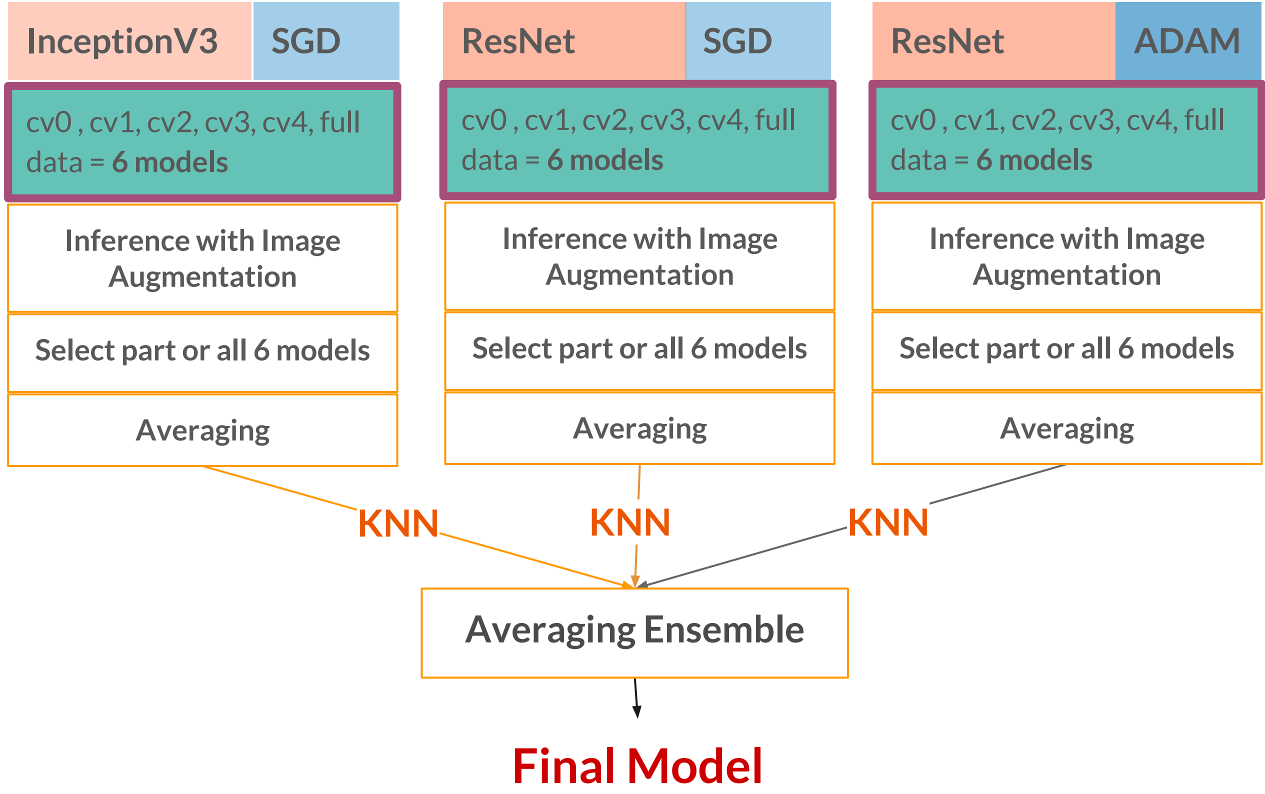
“Ensembling” is simple averaging in this project.

- 1st level: 6 models * 5 augmented images = 30 predictions. We can opt to use portions or all 30 predictions, based on cross-validation scores.
- 2nd level: 5 nearest neighbors
- 3rd level: Averaging different groups of models (InceptionV3, ResNet50)

All predictions are given the same weight.

The 2nd and 3rd level need not perform forward propagation using the model. For a single image, forward propagation needs to carry on for 6*5*3=150 times.

Inference may be too slow for real time application.



Results and Discussion

Cross-Validation Results on Training set

		CV0	CV1	CV2	CV3	CV4	mean	std
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
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

Overfitting: Validation loss was around 4 times higher than training loss.
Ensembling may reduce overfitting

Kaggle Scores on Test Set

Measuring Results in Five Dimensions

- InceptionV3 vs. ResNet50
- Single model vs. Ensembled models
- Augmentation vs. No augmentation
- KNN vs. No KNN
- SGD vs. Adam

Model Optimizer		Training Method		Test Data Preprocessing		InceptionV3 SGD		ResNet50 SGD	
Grp	Index	Training set	# of Model	Augmentation	KNN	Private Board	Public Board	Private Board	Public Board
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	Y	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	Y	0.26251 152 10.6%	0.32938 216 15.0%

Candidate 1

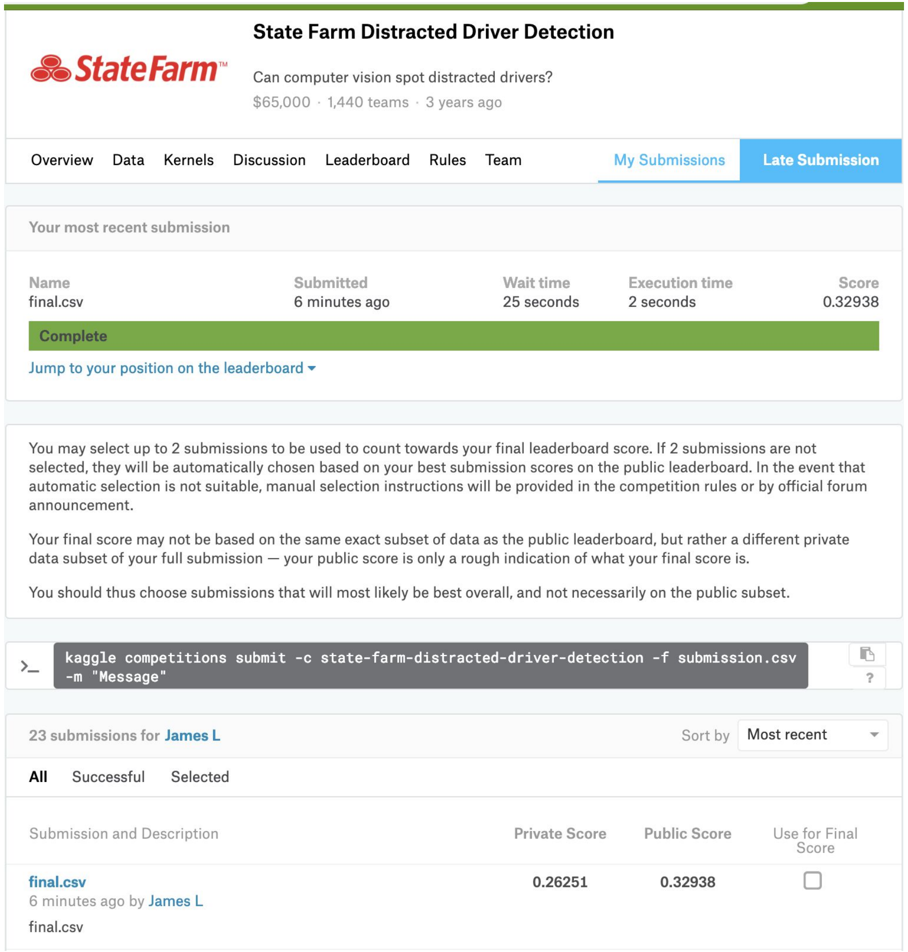
12 models

- Private Leaderboard score 0.26251

  152 out of 1440 teams (Top 10.6%)

- Public Leaderboard score 0.32938

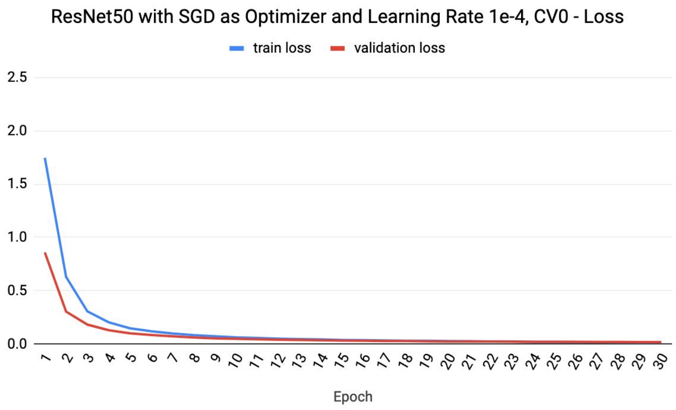
  216 out of 1440 teams (Top 15.0%)

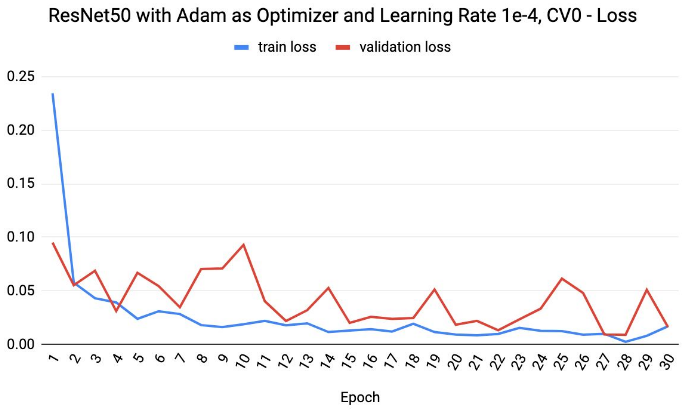


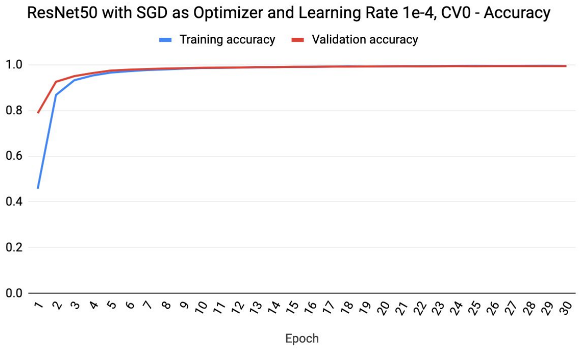
Optimizer Comparison: SGD vs. Adam

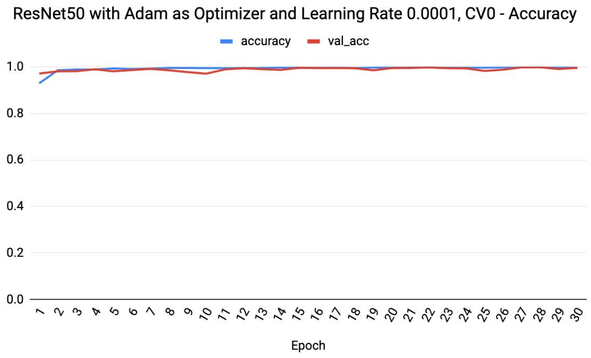
SGD

Adam


ResNet50 with SGD as Optimizer and Learning Rate 1e-4, CV0 - Loss
train loss
validation loss
2.5
2.0
1.5
1.0
0.5
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Epoch


ResNet50 with Adam as Optimizer and Learning Rate 1e-4, CV0 - Loss
train loss
validation loss
0.25
0.20
0.15
0.10
0.05
0.00
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Epoch


ResNet50 with SGD as Optimizer and Learning Rate 1e-4, CV0 - Accuracy
Training accuracy
Validation accuracy
1.0
0.8
0.6
0.4
0.2
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Epoch


ResNet50 with Adam as Optimizer and Learning Rate 0.0001, CV0 - Accuracy
accuracy
val_acc
1.0
0.8
0.6
0.4
0.2
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Epoch

Adam produced some super good models during cross validation

		CV0	CV1	CV2	CV3	CV4	mean	std
ResNet50 Adam Try 1	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

Adam produced some super good models during cross validation

		CV0	CV1	CV2	CV3	CV4	mean	std
ResNet50 Adam Try 1	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
ResNet50 Adam Try 2	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

Training Method	Test Data Preprocessing		ResNet50 Adam Try 1 CV0		ResNet50 Adam Try 2 CV2		ResNet50 Adam Try 2 CV4	
# of Model	Augmentation	KNN	Private Board	Public Board	Private Board	Public Board	Private Board	Public Board
1	Y	First 5	0.22537	0.22082	0.23779	0.24154	0.30331	0.29786
1	Y	First 10	0.22006	0.21145	0.22799	0.23202	0.29710	0.28737
			CV0 + 12 Models*		CV2 + 12 Models*		CV4 + 12 Models*	
13	Y	First 10	0.21654	0.24366	0.21798	0.25559	0.25535	0.27775
			CV0 + CV2 + 12 Models*				NA	
			Private Board Score Ranking (Top%)		Public Board Score Ranking (Top%)			
14	Y	First 10	0.20022 63 (4.4%)		0.21994 103 (7.2%)			
			CV0 + CV2 + CV 4 + 12 Models*					
			0.23013 Private ; 0.23067 Public					

* 12 models refer to the 12 models (InceptionV3, ResNet50) using SGD optimizer

Candidate 2

12 models by SGD +
2 models by Adam

Training Method	Test Data Preprocessing		ResNet50 Adam Try 1 CV0		ResNet50 Adam Try 2 CV2		ResNet50 Adam Try 2 CV4	
# of Model	Augmentation	KNN	Private Board	Public Board	Private Board	Public Board	Private Board	Public Board
			0.22537	0.22082	0.23779	0.24154	0.30331	0.29786
			0.22006	0.21145	0.22799	0.23202	0.29710	0.28737
			CV0 + 12 Models*		CV2 + 12 Models*		CV4 + 12 Models*	
			0.21654	0.24366	0.21798	0.25559	0.25535	0.27775
			CV0 + CV2 + 12 Models*				NA	
			Private Board Score Ranking (Top%)		Public Board Score Ranking (Top%)			
14	Y	First 10	0.20022 63 (4.4%)		0.21994 103 (7.2%)			
			CV0 + CV2 + CV 4 + 12 Models*					
			0.23013 Private ; 0.23067 Public					

* 12 models refer to the 12 models (InceptionV3, ResNet50) using SGD optimizer

Candidate 2

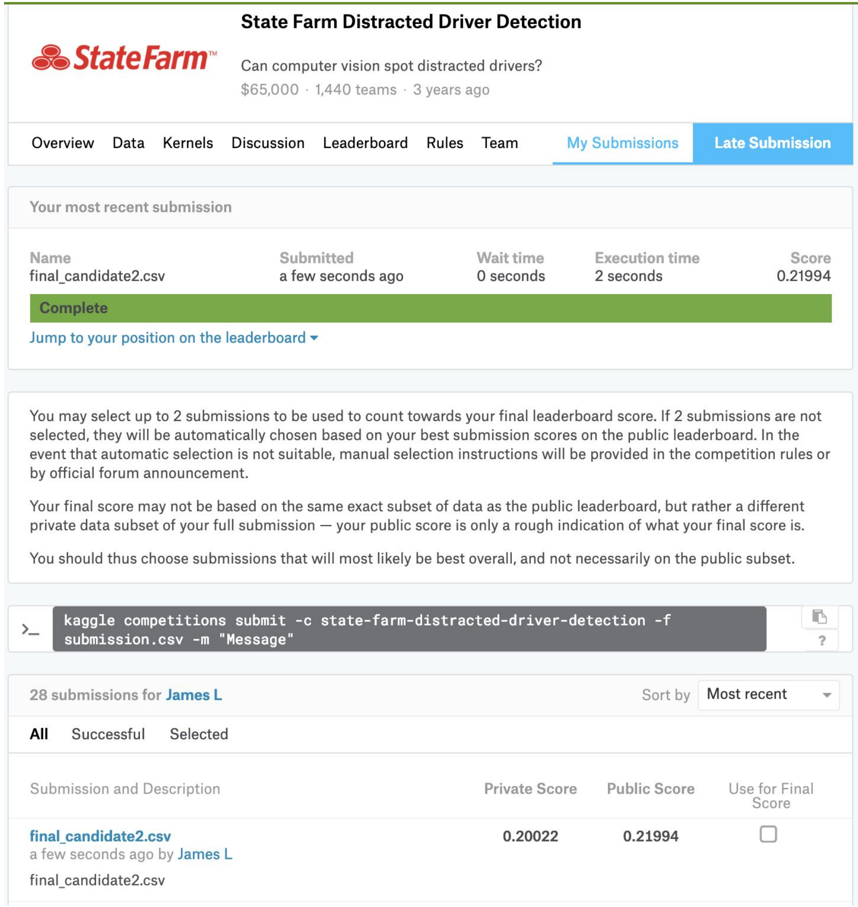
12 models by SGD + 2 models by Adam

- Private Leaderboard score 0.20022

  63 out of 1440 teams (Top 4.4%)

- Public Leaderboard score 0.21994

  103 out of 1440 teams (Top 7.2%)



Summary

What Matters in Improving Performance

- Transfer Learning using Pre-trained models
  - ResNet50 performs a little better than InceptionV3 in terms of loss and accuracy based on cross-validation
- Ensembling
  - Ensemble k models from k-fold cross-validation
  - Ensemble predictions on randomly augmented images
  - Ensemble based on KNN
  - Ensemble two groups of models - InceptionV3 and ResNet50

Future Work

- Averaging predictions to ensemble → Examine each class in each model and manually assign a weight to predictions
- 2 pre-trained models → More models to further reduced overfitting
- For the 12 models trained with SGD, select the best performing models based on cross-validation score and use them to ensemble, instead of ensemble them all

Bonus

Google Cloud Cluster

- ➔ A Dataproc cluster cannot train one model faster, but we could have used it to train multiple models at a time
- ➔ PySpark could be used to create the submission, but...
- ➔ Dataproc cluster with GPU on master and slaves is expensive!! Very easy to spend over $700 in 5 days
- ➔ Google Cloud Platform provides a one-time “courtesy goodwill credit” if you make the same mistake and spend too much

DenseNet121 Memory Requirement

- ➔ DenseNet is composed of multiple densely connected blocks
- ➔ It is extremely memory hungry and consumes more memory than a ResNet because of the number of connections and the backpropagation
- ➔ When attempting to train it, had to reduce batch size and increase learning rate to not run out of memory

Conclusion

Key Points

- Distracted drivers cause problems
- Reached top 4.4% of Kaggle competition!
- Transfer learning is a great way to start off
- There’s always room for improvement - lots of things we could try in the future
- We can also look more into why the model is doing so well

Questions?