Focus on the road

Ennio Nasca - nascaennio@gmail.com

April 27, 2020

Contents

1 Introduction

1.1 Project Overview

According to the World Health Organization, road traffic injuries cause more than 1M deaths worldwide every year[8]. In particular, among the different causes that may lead to car accidents (e.g. speeding and driving under the use of alcohol/drugs) distracted driving is one of the most common. It suffices to say that, according to the WHO, drivers that use their phone are 4 times more likely to get involved in an accident.

These are shocking numbers, that not only make you pause and reflect, but need to be a call to action, because, being able to detect if a driver is absent-minded or, even worst, with a hand on their phone may save real lives.

Among the many companies that tried to come up with useful and innovative solution *StateFarm*[1] — an insurance company — decided to create a Kaggle competition that required participants to create new algorithms that were able to correctly asses whether or not a driver was driving safely. As of today, the dataset is still available at the competition link free for use.

The SateFarm distracted driver detection dataset[2] consists of more than 20k annotated images. Images can be classified among 10 different categories that represent the different types of distractions (e.g. talking to the passenger or hair and makeup) that a driver can have.

1.2 Problem Statement

The goal of this project is to define a machine learning model that is able to recognize the activity done by the driver. The task can be formulated as a multi-classification problem with 10 different classes. The find a suitable solution for this problem we will start by defining a baseline through the use of well known neural network architectures such as VGG19[6] or MobilenetV2[5].

Considering the peculiarity of the problem, we will also try to extend the set of features used to classify an image using information extracted through Human Pose Estimation[3]. This may result particularly useful when it comes to discriminate the hand with which the driver is performing the action (e.g. texting with the left or right hand).

[1] `https://www.statefarm.com`

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

[3] `https://en.wikipedia.org/wiki/Articulated_body_pose_estimation`

The goal of HPE is to localize of human joints, a.k.a. keypoints (e.g. elbows, wrists, hands, etc) in images or videos. Neural Networks trained for HPE on dataset such as COCO[3] and MPII[1] can act as feature extractor and provide crucial information useful to better understand people, and in particular drivers. Finally, we will extend the predictions generated by the baseline, which rely on visual features, with the predictions that rely solely on human pose estimation and combine the two using an ensemble method.

1.3 Metrics

This project treats the activity recognition problem as a supervised multi-label classification problem. For this project we decided to assess the goodness of a model adopting the same metric used to evaluate submissions on Kaggle, i.e. multi-class logarithmic loss. The formula is then:

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

where N is the number of images in the test set, M is the number of image class labels, log is the natural logarithm, 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

2 Analysis

2.1 Data Exploration

The data that will be used in this project is the *Distracted Driver Detection* dataset provided by State Farm. Images are collected with dashboard cameras mounted on different vehicles and picture 26 different drivers that perform 10 different types of actions. Detailed information related to the dataset dimensionality and base statistics are reported in table 1.

Categories	**Units**
image size	(640, 480)
training images	22424
testing images	79726
drivers	26
classes	10
class name	'safe driving','texting right', 'texting left' 'talking on the phone left','operating the radio' 'reaching behind', 'drinking','talking to passenger 'talking on the phone right', 'hair and makeup'

Table 1: Dataset information

From a quick analysis on some sample images1 we can easily see infer two important insights: 1) The position of the dashboard camera can change between drivers 2) Images sharing the same driver and class may be very similar to each other. In particular, from this last insight, we can anticipate the use of image augmentation to prevent overfitting.

Figure 1: Four random images taken from the StateFarm Dataset showing three different drivers and four different classes. From left to rigth: operating the radio, texting left, safe driving and talking to the passenger.

2.1.1 Exploratory Visualization

To obtain further insights on the dataset some data exploration is carried out to identify the underlying data distribution and spot a potential class imbalance.

The data exploration step is performed both for drivers and classes. First, we focus on the drivers and plot the number of images taken for each one of them. From figure 2 we can see that images are evenly distributed among drivers, except for driver *p072*, with an average of 862 images per driver and a standard deviation of 214.

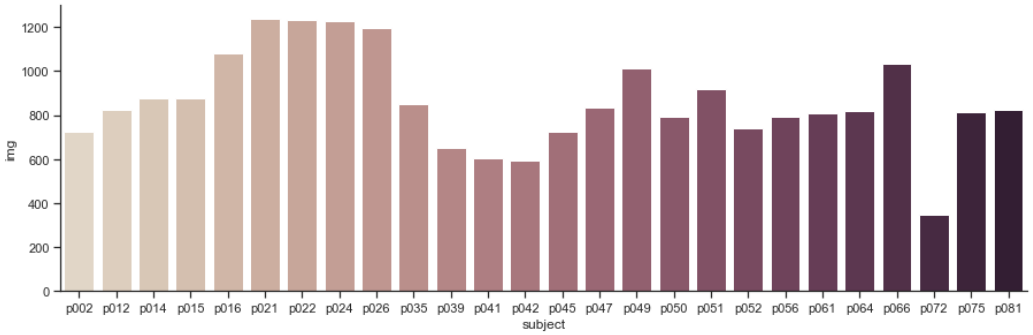


Figure 2: Image distribution per driver. On the x axis, the id of the driver, on the y axis the number of images for that driver.

The same analysis is performed on image labels with the results shown in figure 3. From this figure, we can easily see that we are working with a balanced dataset with an average of more than 2000 images per class. The most common class is *safe driving*, directly followed by the two *texting* ones.

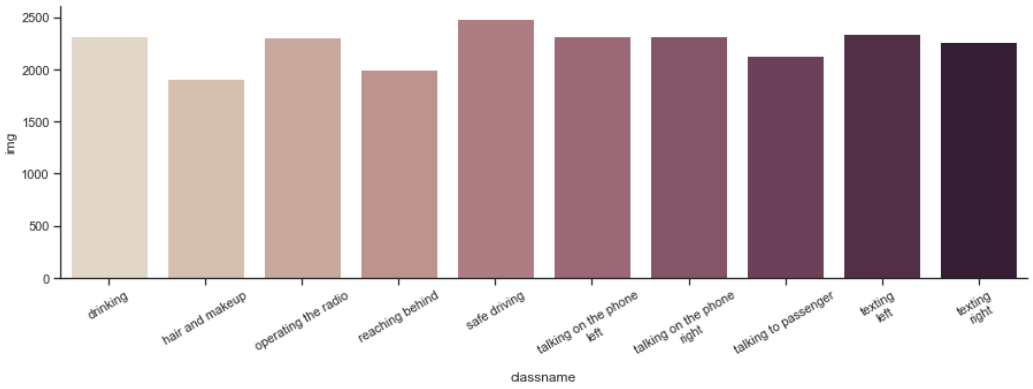


Figure 3: Image distribution per label. On the x axis, the class labels, on the y axis the number of images for that label.

Finally, we can perform a deeper analysis on the combination of drivers and class labels. In figure 4 we can see a heatmap showing the number of images in the dataset for each pair of *(driver, class label)*. From the figure, we can see that 4 drivers have more than double the images for almost all classes. This imbalance may lead a CNN model to overfit on the driver. For this reason, we will put inplace a sampling strategy that will only take a fixed number of images for each class trying to reduce the bias that having less driver may introduce.

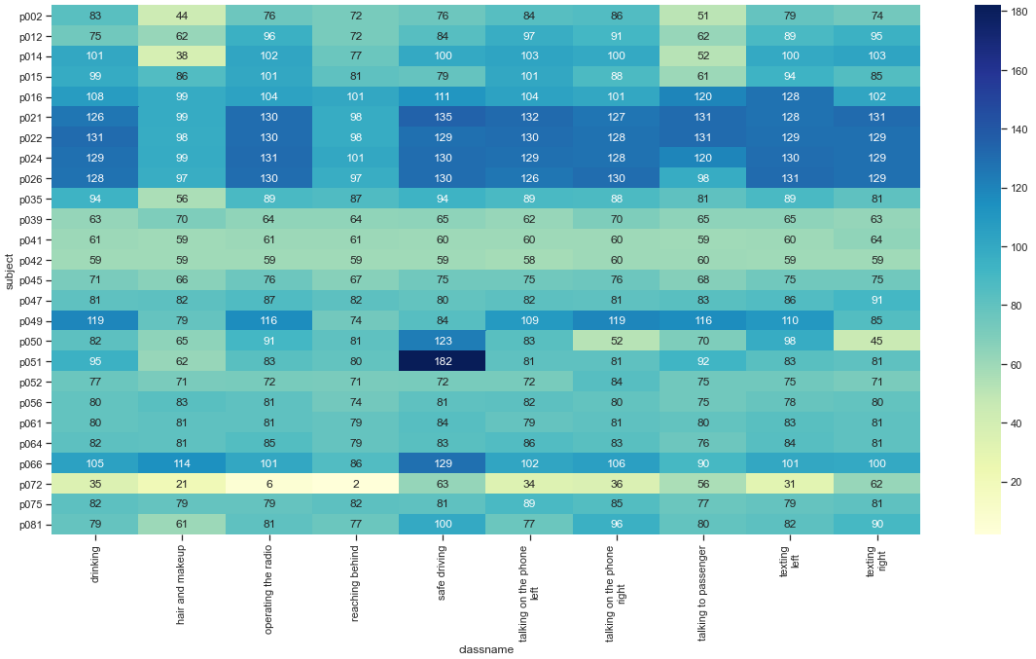


Figure 4: Heatmap showing the number of images for each *(driver, class label)* pair.

2.2 Algorithms and Techniques

In this project we will rely on two families of models, namely Convolutional Neural Networks[4] and Extreme Gradient Boosting[5].

The former is a Deep Learning algorithm which can take an image as input, assign importance, i.e. learnable weights and biases, to various aspects/objects in the image and be able to discriminate one from the other.

[4] `https://en.wikipedia.org/wiki/Convolutional_neural_network`

[5] `https://en.wikipedia.org/wiki/Gradient_boosting`

As a starting point for this project we will use the VGG-19[6] pretrained ImageNet[6] model with additional batch normalization[7] to prevent overfitting.

The latter is a Machine Learning algorithm suitable for regression and classification problems. Gradient boosting works by constructing an ensemble of weak prediction models, typically decision trees, and combining their prediction to generate a final output. Over the past few year algorithms like XGBoost and CatBoost have been used in many winning solutions for Kaggle competitions. These algorithms are particularly suitable when working with heterogeneous data coming from different source, which will be our case when combining HPE with CNN.

To asses the goodness of our models we will compare our performance against the entries in the Kaggle competition. This results in a good benchmark since more than 1400 groups have taken part in the competition. Therefore, comparing our solution against their score on the public and private leaderboards will allow us to identify in which top % our model ranks.

[6] `http://www.image-net.org`

[7] `https://en.wikipedia.org/wiki/Batch_normalization`

3 Methodology

3.1 Data Preprocessing

For this project, we carry out two different types of preprocessing, namely sampling and augmentation.

The rationale behind the former is that few drivers have more than double the number of images for certain classes, and this may lead a CNN model to overfit towards those drivers. In order to limit this issue, we implement a sampling strategy that takes only a subset of all the images in each class.

The latter preprocessing step is image augmentation, is a crucial step to avoid overfitting when working with images. In our case, to images are randomly rotated in a fixed angle that doesn't distort too much the image, resulting in an unrealistic dashboard camera picture.

Additionally, when working with pretrained PyTorch models we need to normalize the data using the same normalization required by ImageNet models[8].

To speed up computation, images are resized from (640, 480) to (298, 224) preserving the aspect ratio.

Finally, to assess the performance of our models the whole dataset is divided into train, validation and test set. The first two sets are built using the sampling process described before and contain 5k images each while the test set contains the remaining 12k images.

3.2 Implementation

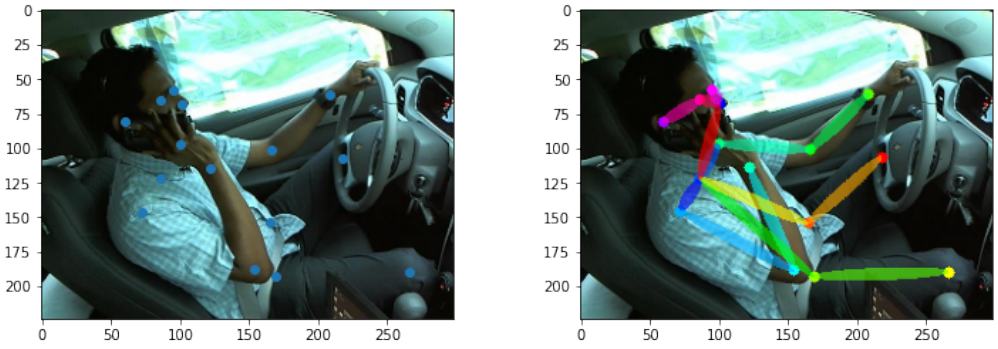
The implementation process can be divided into three separate steps that require the use of pretrained PyTorch models and CatBoost gradient boosting to train both the classifier that relies on human pose estimation and the final ensemble method.

VGG-19 The first step is to train a CNN that receives as input an image and gives as output a vector of class probabilities. Our CNN of choice is a VGG-19 CNN with batch normalization that uses pretrained ImageNet weights, available in Pytorch, as starting weights. Because the ImageNet task requires the network to predict 1k classes we need to change the final layer with one with only 10 classes. Additionally, given the fact that we want to have probabilities as output we will add a Softmax layer on top of the final predictions. The model is trained for various epochs using Adagrad optimizer

[8] `https://pytorch.org/docs/stable/torchvision/models.html`

with a learning rate of $1e^{-3}$ and a weight decay of $1e^{-4}$ while monitoring the validation loss to perform early stopping.

OpenPose The second step is to train a model that receives as input a vector of keypoints coordinates and gives as output a vector of class probabilities. To extract relevant body parts from the images we rely on a pretrained Human Pose Estimation model called OpenPose[2]. The original Caffe[9] architecture is ported to Pytorch to allow the development of the project using a single deep learning framework. From each image, 18 body keypoints are extracted according to the COCO output format[10]. An example is reported in figure **??**, where both the keypoints coordinates and the reconstructed human pose are depicted.



(a) (b)

Figure 5: (a) Keypoints and (b) Reconstructed human pose for a sample image

In our case some body parts, such as ankles, may often be missing due to occlusion in the image, nonetheless relevant limbs such as forearms and head are never hidden. With the extracted keypoints we can train a CatBoost classifier that relies solely on this information to predict the correct class probabilities. The CatBoost optimal parameters shown in table 2 are obtained from a grid search over 81 different combinations.

[9] `https://caffe.berkeleyvision.org`
[10] `https://github.com/ArtificialShane/OpenPose/blob/master/doc/output.md`

Parameter	Value
iterations	1000
learning rate	0.02
depth	6
min data in leaf	5
l2 leaf reg	1

Table 2: Best CatBoost parameters found through grid search for the classifier that uses keypoints features

Ensemble Due to the orthogonality of the two previous approaches we can further improve our performance by combining the predictions with an ensemble. The goal of this new CatBoost model is to weight the different class probabilities predicted in the previous steps and output a unified vector of probabilities. Similar to the previous step, the optimal parameters are shown in table 3 are found through grid search.

Parameter	Value
iterations	100
learning rate	0.1
depth	6
min data in leaf	5
l2 leaf reg	3

Table 3: Best CatBoost parameters found through grid search for the ensemble classifier

4 Results

4.1 Model Evaluation and Validation

In this project we are able to perform two different types of evaluation, a local one using a test set and an "online" one thanks to the Kaggle leaderboard. To perform the former we assess our performance against a test test containing images that haven't been used neither for training nor for validation. The latter is achieved by loading a well formatted submission file on the Kaggle competition website.

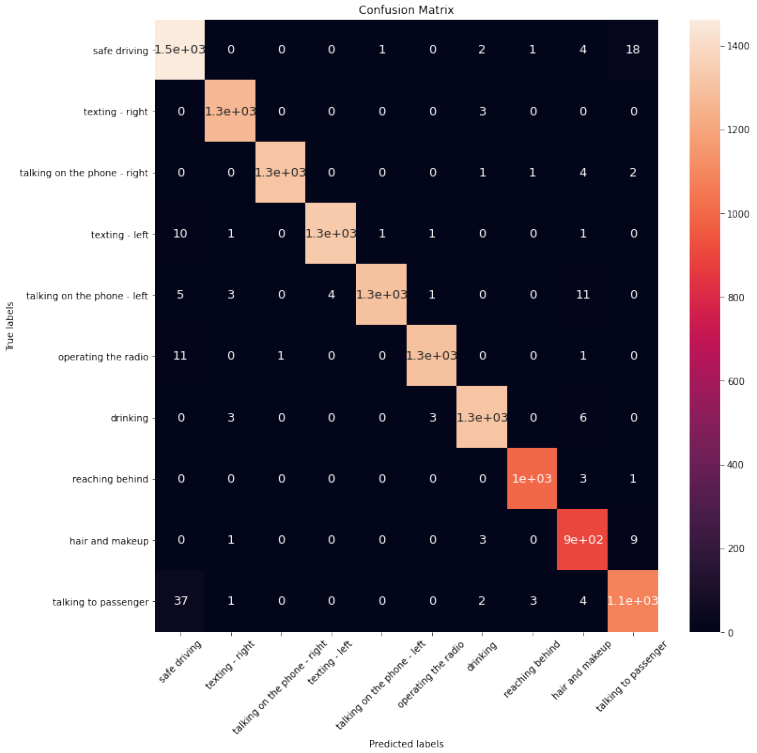
Local Evaluation For the first type of evaluation we train our models on a train set with 5k images while performing early stopping using a validation set of the same size. The test set on the other hand contains around 12k images.

For each of the three models described in the previous chapter we use two different visualization that can help in the evaluation process, the first one is the classical confusion matrix[11], the second type of visualization is a boxplot[12] that can help us better understand not only which images are correctly classified but also how certain our model is about its predictions, in fact to build this visualization we take the probability that our model assigns to the correct class and compute the quartile ranges.

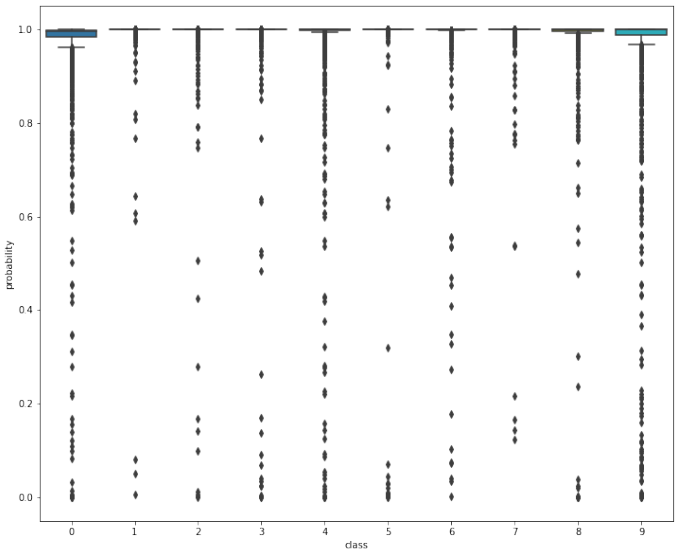
In figure 6, 7 and 8 we report the combination of confusion matrix and box plot for each model. Looking at the confusion matrix we can see which classes are misclassified the most. In general, the two worst one are *safe driving* and *talking to the passenger*.

Moreover, it is interesting to notice how the model that relies only on human pose has some difficulties in distinguishing between *hair and makup* and *drinkin* which in a human pose can be attributed to a slightly different height of the hand w.r.t. the face and hence it is very difficult to detect. The same problem is also testified by the box plot which shows a lower first quartile for class 8, i.e. *hair and makup*.

[11] `https://en.wikipedia.org/wiki/Confusion_matrix`
[12] `https://en.wikipedia.org/wiki/Box_plot`

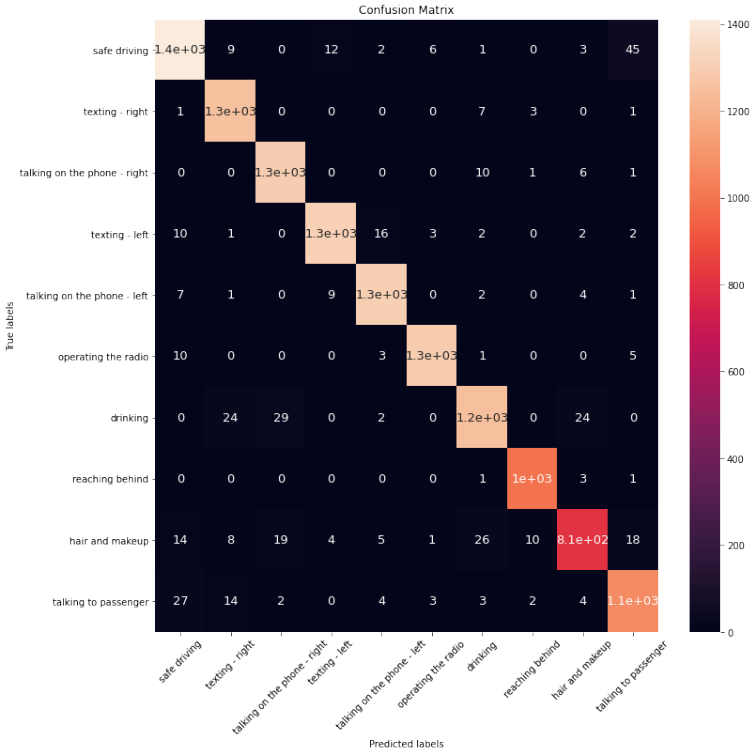


(a)

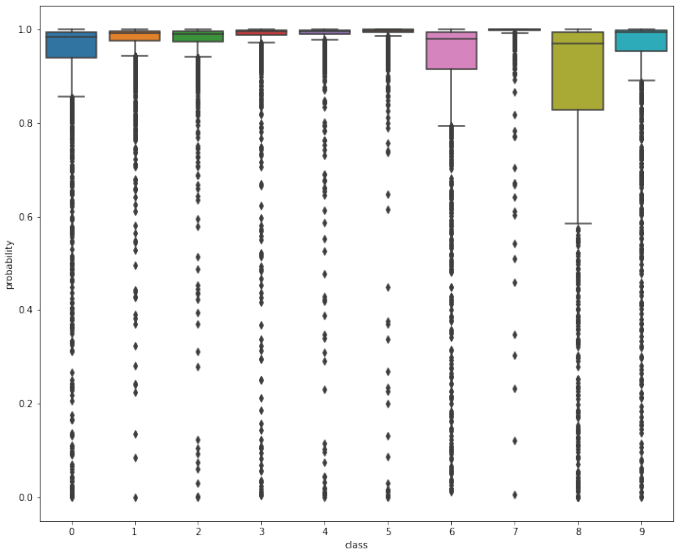


(b)

Figure 6: (a) Confusion matrix and (b) Probability box plot for the VGG19 model trained using the augmented images.

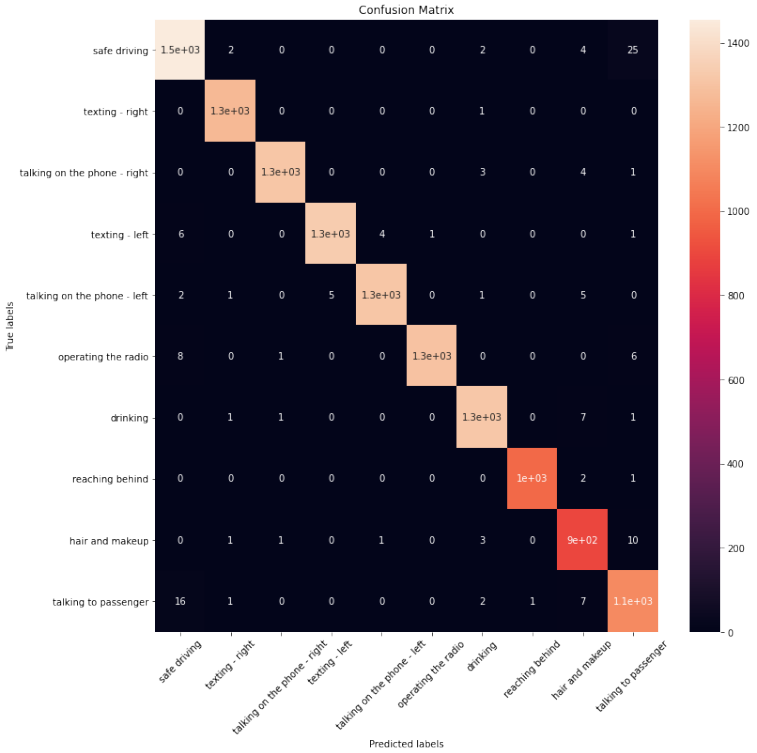


(a)

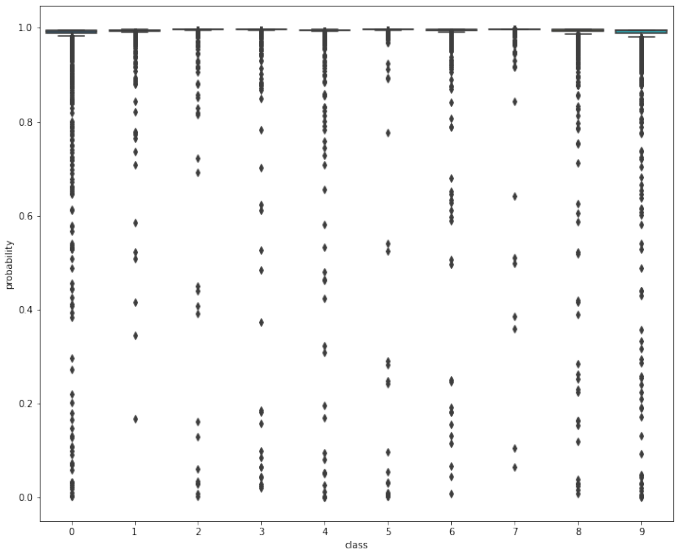


(b)

Figure 7: (a) Confusion matrix and (b) Probability box plot for the CatBoost model built using the features extracted with the OpenPose model.



(a)



(b)

Figure 8: (a) Confusion matrix and (b) Probability box plot for the final ensemble model trained using predictions generated by the two previous models.

Kaggle Leaderboard Kaggle offers the possibility to score a late submission against the unlabelled test data which consists of more than 77k images. Uploading a well formatted csv file, where each row is an image with the 10 class probabilities. We uploaded a csv file for each one of our models and report in table 4 the score obtained by each one of them in both the private and public leaderboard. The public leaderboard is calculated with approximately 19% of the test data while the private leaderboard is calculated with the remaining 81% of the test data. As expected the CNN models performs quite good by itself while the CatBoost model trained using keypoints performs slightly worsts and shows a higher variance between the two leaderboards. Finally, the ensemble method is able to combine two different sets of features and improve performance while also decreasing the variance of the two models.

Model Name	**Public Leaderboard**	**Private Leaderboard**	**Final Score**
VGG19	0.3744	0.3799	0.3788
OpenPose + CatBoost	0.6565	0.6080	0.6172
Ensemble	**0.3196**	**0.3192**	**0.3192**

Table 4: Multi-class log loss score obtained on the Kaggle competition divided by public, private and final leaderboard.

5 Conclusions

The main objective of this project was to detect distracted drivers from images, along with the type of distraction, among a set of 10 different classes. The project focused on extending the set of visual features extracted by a CNN with orthogonal features related to human pose estimation extracted with OpenPose. In the end, throughout the use of the StateFarm dataset, we were able to train a model that ranked among the top 14% of the Kaggle leaderboard for the associated competition.

If we consider this project as a starting point, there are several directions in which it could be extended.

First of all, we could further improve our performance by trying to improve our sampling strategy in a way that it takes into consideration the data distribution and sample not at random but following a proper probability distribution extracted from the dataset.

Additionally, we chose to focus only on body pose keypoints, but the family of keypoints extracted from an image can be extended to other body parts such as face and hands, which may further improve the model ability to understand the type of distraction, for example the position of the mouth can help to discriminate whether or not the driver is talking to the passenger or is driving safely.

Finally, another option could be to investigate different CNN architectures to train an image based classifier. New state of the art architectures, namely ResNetXt[4] and FixEfficientNet[7], could lead to a significative improve in performance.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693, 2014.

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[4] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

[5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[7] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy: Fixefficientnet. *arXiv preprint arXiv:2003.08237*, 2020.

[8] World Health Organisation. Road traffic injuries, 2020. `https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries`, Last accessed on 2020-04-14.