Distracted Driver Detection

Luca Pessina
luca.pessina@studenti.unipd.it

Filippo Santin
filippo.santin@studenti.unipd.it

Abstract

In this project, after having well analyzed a dataset about Distracted Driver Detection with a definite exploration and showing some statistical graphs, we tested different Convolutional Neural Networks: a pre-trained model VGG16 and three models (of different complexity) that we created with optimal parameters after some experiments.

Trying to find the best values of learning rate and dropout, we aimed to reach the best accuracies of Action Recognition for the presumed most skillful model displaying our results and suggesting further studies.

1. Introduction

Distracted driving is one of the leading causes of car accidents. Recently, the number of road accidents has been increased worldwide. According to the World Health Organization (WHO) survey [1], 1.3 million people worldwide die in traffic accidents each year, 150 people per hour, making them the eighth leading cause of death and an additional 20-50 millions are injured or disabled.

Some actions carried out by drivers while driving such as using mobile phone, switching to a favorite radio station, eating or drinking may danger road safety and could cause accidents since they require driver's attention. Therefore, it is essential to monitor and analyze the driver's behavior during the driving time to detect the distraction and mitigate this larger number of road accidents.

Road safety can be considered the result of the state of three fundamental components: the driver, the vehicle, and the road. Distracted driving certainly falls under the first component, and consequently it is the human that needs to be acted upon for the purpose of risk reduction.

Driving distractions can be divided into three categories:

- Visual distractions (taking your eyes off the road)
- Manual distractions (taking your hands off the wheel)
- Cognitive distractions (thinking about something else while driving).

A first approach can be to use the sensors embedded in a vehicle. Some cars have hundreds of physical sensors onboard that describe the operation of the internal subsystems of the car. Some of the example sensor readings are speed, revolutions per minute, throttle position, and fuel capacity.

Instead, in our work we focus on a approach based on vision where the manual behaviours of the driver are captured by a camera placed inside the car. In this way we take a large number of pictures so that we can analyze the postures. Arm gestures, the presence of objects near the human, the movement of the head and other information help us to categorize the wrong actions. In a nutshell, we can consider Action Recognition as the aim of the work. In particular, machine learning is concerned with the development of models that enable a system to gain the necessary knowledge based on these images.

We present Convolutional Neural Networks based approach for this problem. We also attempt to reduce the computational complexity and memory requirement while maintaining good accuracy which is desirable in real time applications [2].

2. Related work

Image classification is a well known problem in computer vision. Explicitly, in the last 10 year with the coming of new and more powerful processor, the complexity of the models increased leading to a noticeable rise in model performance. The performance of the classification for the ImageNet dataset is a reference for the state of the art in this field. From the 2015 when the ResNet network outperformed humans, several models have been implemented that exploit different techniques. Particular trends in designing CNN for image classification involve non-linearity, dropout regularization, deeper neural networks, batch normalization, multi-level features extraction, more connections between the layers. In recent years optimized models of EfficientNet reached a top1 accuracy around 90% and a top5 around 88% with hundreds of trainable parameters.

Furthermore, this process of drivers detection has been linked to prevent the Driver Drowsiness, topic about which a great number of scientists have worked and are working to help automakers to create smart cars. Lots of papers deal

Figure 1. Save driving

Figure 2. Drinking

Figure 3. Talk passenger

Figure 4. Using radio

with deep learning techniques to extract numeric features from images, which are introduced into a fuzzy logic-based system afterwards. The accuracy obtained is around 65% accuracy over training data, and 60% accuracy on test data. In any case, the fuzzy logic-based system avoids raising false alarms and reaches a specificity of 93% [3]. Similar topics can be found in [5].

3. Dataset

The provided dataset has images of 26 different drivers, each subject is either simply driving (no distractions) or is distracted in some way. For example in particular each image is associated to one of the following driving action: safe driving, texting using the right hand, talking on the phone using the right hand, texting using the left hand, talking on the phone using the left hand, operating on the radio, drinking, reaching behind, hair and makeup, and talking to a passenger. This dataset is taken from a Kaggle competition of the State Farm insurance that what to insure their customers by testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviors [4]. We found out these images are correlated as they are produced as frames from a video clip here. Images taken from a single video segment could be put into the same category, although if judged individually, a few of them might be labelled differently. Anyway, in the following figures (1-6) we report some examples of the images used for the classification with the relative class as caption.

There are 22424 train images and 79725 test images overall. We have divided the images in the train folder into training, validation and test sets so that we could train and evaluate our models. Specifically, there are 15135 training images (67%), 5046 validation images (23%) and 2243 labeled test images (10%).

Figure 5. Reach behind

Figure 6. Texting right

3.1. Data exploration

The dataset is fairly balanced across the ten different classes, in particular each class has between 2500 (safe driving) and 1900 samples (hair and makeup). If we consider the images per driver, we have some differences: some subjects show a double number of images respected to others. This fact will not impact the performance of the classification. In terms of kind of images per driver also in this case we do not highlight particular dissimilarities across the subjects, all have more or less the same proportion of images per class. Each image has a resolution of 640x480 BGR and each of them will be re-scaled in a 64x64 size.

We implemented a small functional code to show an heatmap in order to understand where the user's attention lingers. It is a sort of representation of data in the form of a map in which data values are represented as colors. The greatest advantage of this tool lies in the visual representation of the data itself, which immediately provides at a glance a complete overview of how the user uses all the elements that are made available to them. Especially, in our case the arm movement can help to realize what the driver is doing (7).

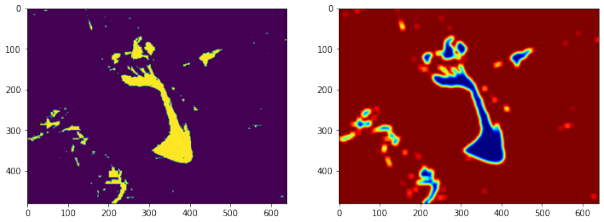

Figure 7. Heatmap example - talk on the phone right

3.2. Pre-processing

In addition, we want to underline the use of *IAMREAD_COLOR* to read pictures. It means that images are always converted to the 3 channel BGR color image. The most classic tool is definitely RGB which stands for Red Green Blue. Most often, like in our case, it is stored in a structure with Blue occupying the least significant "area", Green the second least, and Red the third least (8).

This color space equips us with what seems to be a natural representation of images in the contest of our task. Since

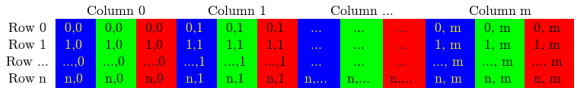

Figure 8. BGR structure

our goal is to identify specific actions inside the car, each colored object makes more detail-oriented in the recognition respect to a system characterised by grayscale.

4. Methods

To carry out the multiclass classification of the images we studied different CNN (Convolutional Neural Networks) models, a type of artificial neural network widely used from image recognition and classification. Convolutional neural networks are composed of multiple layers of artificial neurons, in particular a set of convolutional, pooling, normalization and dense layers. The first layers usually extract the most basic features and going deeper into the network's architecture we can identify more complex features. This kind of models need a lot of data to train and, to avoid overfitting, some regularization techniques are usually implemented. In particular the dropout technique is used to prevent overfitting by ignoring some randomly chosen neurons during training. In this way the network can learn more robust features and help to improve the generalization error. To choose the right parameters, fraction of neurons deactivated, we performed different experiments and selected the best one.

4.1. Models overview

We studied three different methods, the first two models are built and trained from scratch while for the third we used a transfer learning technique for a VGG16 model. For all the three models we used the same pipeline and used the same training, validation and test data as reported in 3. The models have all different architecture in terms of number of layers. For all the methods we used the *soft-max function* as activation function to perform the multi-class classification that will provide, for each images, the probability to belong to each class. This function is usually used in the final layer because converts the scores to a normalized probability distribution. As loss function, for all the models, we used the categorical cross entropy. The formula is reported in (1) where $\hat{y}_i$ is the scalar value of the model output, y_i is the corresponding value and N is the output size.

$$Loss = -\sum_{i=1}^{N} y_i * log(\hat{y}_i) \tag{1}$$

As example in figure (9), we give an account of the architecture of one of the models implemented. For the basci model, model 2 and VGG16 we used the Adam optimizer, a batch size of 40 and for 10 epochs. For the model 1 we used the RMSProp as optimizer. As well, this optimizer tries to dampen the oscillations, in particular it takes away the need to adjust learning rate doing it automatically. As metric, we used the accuracy that is a stable and reliable metric in this case as long as the classes are well balanced in terms of instances for class. For all the models we applied the BGR and the greyscale images, we highlight only the results for the BGR case as they offer better achievements and stability. The basic model has about 77.000 trainable parameters, the model 1 around 4.5 million while the model 2 around 8.8 million. We trained all this models from scratch but also tried to build a model with less trainable parameters exploiting the transfer learning technique as we show in the following section.

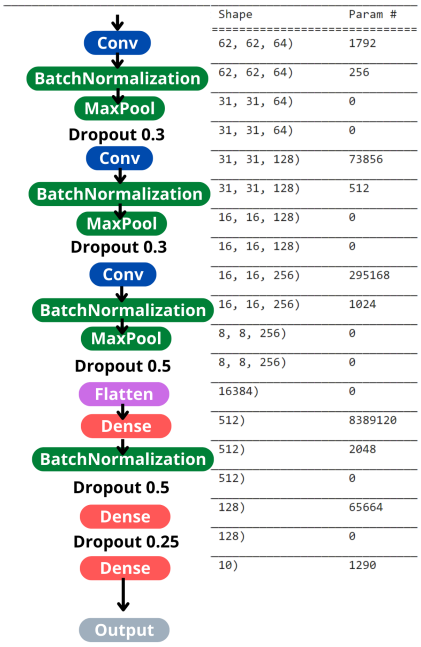

Figure 9. Second CNN model

4.2. Transfer Learning

Transfer learning is a technique that has the goal to use the information and knowledge learned from a particular task, usually manipulating a huge amount of data, to solve another but similar task. Particularly, a model that exploits transfer learning consists in a pre-trained network that is used as base for the full model. There are attached some specialized layers that will be trained and used for the detail we are trying to solve, typically fully connected layers. In transfer learning, the weights of the pre-trained layers are not modified but are applied only to extract the basic features for the new task. We implemented a transfer learning

model in Keras using a pre-trained VGG16 network. This famous network consists in 5 blocks with a total of 19 layers and we adopted the weights from the training using the ImageNet dataset composed by more that 14 million images belonging to 1000 different classes. Then we added the final layer for our task composed by two dense layers. The first one with a *Relu* activation function and the last, that will compute the classification of the images into 10 different classes, using a *Softmax* function. The final model has around 15 million parameters but about 500.000 of them are trainable. Also in this case we performed the same procedure and analysis carried out for the previous models. The model was trained for 10 epochs using a batch size or 40 samples, the *categorial crossentropy* as loss function and the *accuracy* as metric.

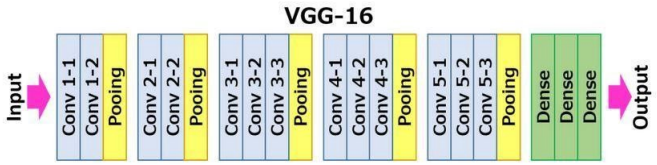

Figure 10. VGG16 architecture

5. Experiments

As we mentioned before we trained the models for 10 epochs using a batch normalization of 40 and different kind optimizers reaching quite good performance for each class. We report the main results only for the model 2 that is the most stable and performing, we also carried out a verification process about the dropout value and learning rate and finally compared the models in terms of speed and performance.

5.1. Main Results

All the models reach a good overall accuracy between 96% and 99%. The best model in terms of accuracy is the model 2 that reaches a 0.995% in the test set composed by around 2200 images. The plot of the accuracy fig.(11) and the loss fig.(12) of the models are posted. The validation accuracy is slightly bigger than the training one and the loss function smaller. We can state why. First of all, maybe the variance of the images is not so big and they are very similar each others since are all drivers; the dropout rate increases the generalization of the model and affects the training loss, but not in the validation and test ones. Besides, the training loss that Keras displays is the average of the losses for each batch of training data, over the current epoch. Because the model is changing over time, the loss over the first batches of an epoch is generally higher than over the last batches. On the other hand, the validation loss for an epoch is computed using the model as it is at the end of the epoch, resulting in a lower loss.

Dropout experiments			
Dropout	Train	Validation	Test (macro)
0.2	0.994	0.991	0.995
0.4	0.985	0.994	0.996
0.6	0.928	0.974	0.978
0.8	0.409	0.331	0.623

Table 1. Accuracies based on different dropout values

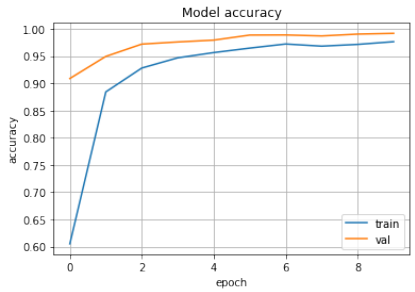

Figure 11. Accuracy CNN model

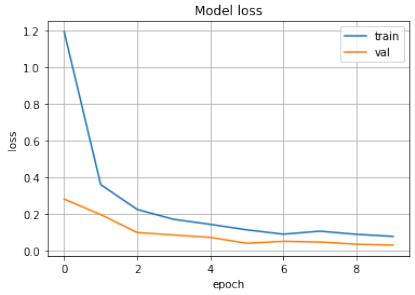

Figure 12. Loss CNN model

We studied the CNN model 2 deeper and we observed the behaviour of different dropout rates $(0.2, 0.4, 0.6, 0.8)$, we trained the model and plotted the train and validation accuracy. It is possible to notice that when the rate is high, the model is not able to perform the classification and it is in overfitting. However, While decreasing the value the performance increases and also the overfitting phenomenon decreases. We selected the dropout values used in the model 2, previous results, trying different combinations and looking for the best accuracy and minimizing the overfitting. For completeness we report the results in table (1). We report the *macro average* that compute the metric independently fr each class and then take the average.

We also carried out some experiments on the learning rate using the Adam optimizer and the CNN model 2 with the optimized dropout rates. It is interesting to see how, arising the learning rate parameters, the optimization is less stable and, if it is too small (0.0001), the learning process

Learning rate experiments			
Learn rate	Train	Validation	Test (macro)
0.0001	0.957	0.985	0.986
0.0005	0.987	0.994	0.999
0.001	0.985	0.994	0.997
0.005	0.960	0.987	0.991
0.01	0.938	0.976	0.988

Table 2. Accuracies based on different learning rates

is very slow. The best learning rate parameters seem to be 0.001. Since the training dataset is not huge the learning rate does not afflict the time in a sensible way. Also in this case, we report the result in table (2).

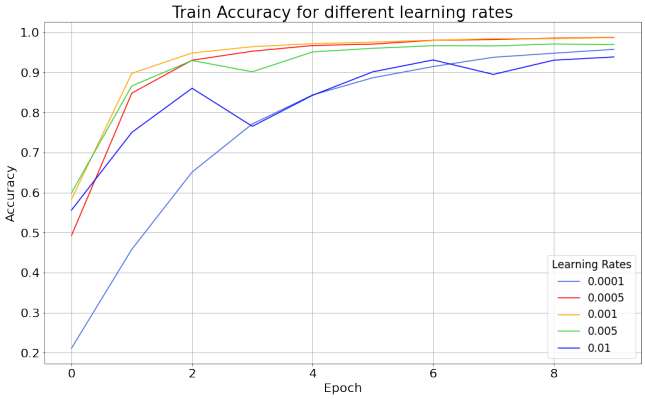

Figure 13. Different learning rate for Second Model CNN

For completeness we report in figure (14) the results of the second CNN model per class. Clearly, the class that is classified worse is the "Talking to passenger" maybe because the driver does not have any particular object (like drink bottle or phone), but only the head slightly tuned, which is an equivocal movement.

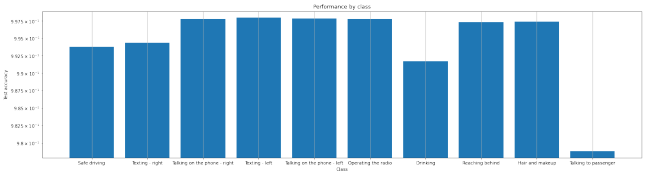

Figure 14. Test accuracy by class for the second CNN model

5.2. Comparison among the models

If we compare the different models we do not notice a big difference in performance (apart from the first baseline model). It can be interesting a reasoning about the achievement per class and a possible production of the model. We plotted the train accuracy for each model and it is possible to witness in figure (15) how the VGG16 "learn faster", indeed in the first epoch of training reach a train accuracy of 0.72 when the other models only around 0.60. This is due to the fact that the model is pre-trained, therefore it has some knowledge about basic features before starting the training action.

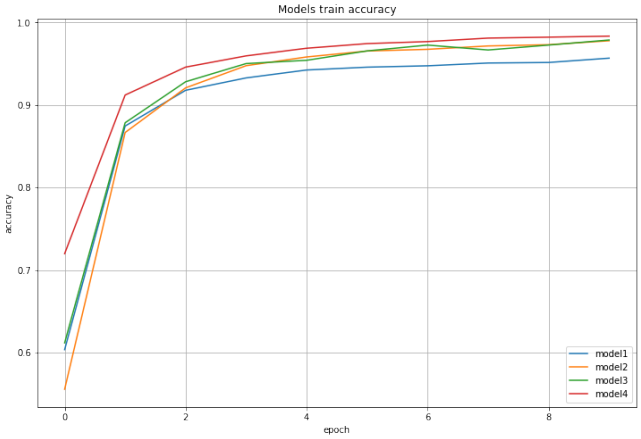

Figure 15. Train accuracy for the all the models

On the other hand when the number of epochs increases the models trained from scratch outperform the VGG16 because, given the largest number of parameters, can gain more specific features. If we think to train a model with a larger dataset, maybe with more classes and elements, the transfer learning technique is the most promising. The table (3) shows important output values, summarizing our analyzes.

Model	Train	Validation	Test
Basic CNN	0.956	0.987	0.986
Model 1	0.977	0.986	0.987
Model 2	0.978	0.991	0.995
VGG16	0.983	0.965	0.960

Table 3. Accuracy of the models evaluated in each sample

5.3. Particular case

Investigating the test set and the prediction of the models, it is interesting to look at outcomes and to try to deduce some peculiar behaviours. In figure (16) we report a image that is incorrectly classified by all the models. The true class is "safe driving", while for the models we achieve:

- Texting - left (Basic model)
- Texting - right (CNN1)
- Texting - right (CNN2)
- Hair and makeup (VGG16)

The main reason is that the driver is driving with only the left hand, while for the majority of the "safe driving" images the driver uses two hands. And the free hand is the right, so the predicted classes, in particular text because this hand is placed low.

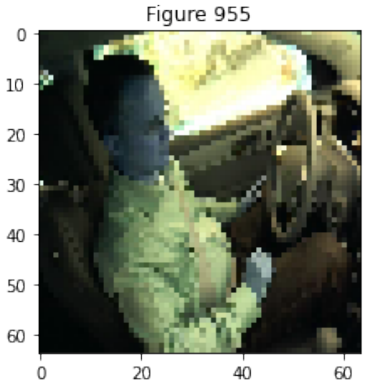

Figure 16. Wrongly classified image

6. Conclusions

With the development of this project we took our steps towards the deep and complex task of driver distraction detection. We understood the importance of computer vision in trying to prevent car accidents due to reckless behavior by people and we believe that a device with a camera installed in cars that tracks the movements of the driver and alerts them can help in general.

The resolution of colored images and the study of heatmaps are the main subjects of the exploration of our dataset. Starting with one of the simplest CNN model we worked in order to add layers and regularizations with a logic (Global average pooling, Dropout, Batch normalization) to extrapolate parameters and specific information to improve it. Besides, using a pretrained model like VGG16 model lets us to obtain always good outcomes.

We consider our results to be optimal since the accuracy is so high, always close to 1. However, the topic is too important and it would be nice to reach perfection in order to save people's lives. For this reason, we think that the part of Action recognition based only on the Image tools alone cannot be enough. More in-depth work is needed, where we can add for example Video action or speech recognition. The code is available at this notebook.

References

[1] *Info drivers accidents*. 2018. URL: `https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries`

[2] *A review of action recognition based on Convolutional Neural Network*. 2021. URL: `https://iopscience.iop.org/article/10.1088/1742-6596/1827/1/012138/pdf`

[3] *Driver Drowsiness Detection by Applying Deep Learning*. 2021. URL: `https://www.mdpi.com`

[4] *Distraction driver dataset*. 2016. URL: `https://www.kaggle.com/competitions/state-farm-distracted-driver-detection`

[5] Omerustaoglu, Furkan and Sakar, C Okan and Kar, Gorkem (2020), Distracted driver detection by combining in-vehicle and image data using deep learning, Elsevier