Machine Learning Engineer Nanodegree

Capstone Project

Jordan Milushev
November 10th, 2018

I. Definition

Project Overview

Distracted driving is one of the leading causes of auto accidents in the US today. According to NHTSA, in 2016 alone, there were 3,450 lives lost due to distracted driving (https://www.nhtsa.gov/risky-driving/distracted-driving). Distracted driving may be caused by many factors, including but not limited to conversations with passengers in the car, conversation on the phone, adjusting radio, navigation or other controls in the car, eating, texting, checking e-mail, drowsiness, etc. Passengers should always remind the driver to keep their eyes on the road and hands on the wheel. A lot of times, drivers are alone in the car, and in the case of teenager drivers, having one or more teenagers in the car may further increase the teenage driver distraction.

This project will provide a solution that would hopefully be able to improve these statistics, by allowing real-time feedback to drivers or/and insurance companies how focused they are on the task of driving.

The project is based on a Kaggle competition sponsored by State Farm (https://www.kaggle.com/c/state-farm-distracted-driver-detection).

The data for the solution is used from that Kaggle competition (https://www.kaggle.com/c/state-farm-distracted-driver-detection/data). The data is not provided for public use and can only be used for the purpose of the competition. It can be obtained by logging into the Kaggle web-site, accepting the rules of the competition, after which the data can be downloaded.

Problem Statement

The accidents caused by distracted driving can easily be reduced or eliminated altogether if this behavior can be minimized. Usually, every one of us knows the risk associated with distractions during driving but still engage in this behavior without even noticing. If there was a way for the environment in the car to change when the driver becomes distracted, it could serve as a reminder to the driver to correct himself or herself. For example, if the driver becomes distracted, the radio could start playing or turn its volume up. If the driver starts texting, there could be a sound signal or voice warning played. In addition, insurance companies could reward good behavior by lowering premiums for drivers who do not engage in distracted driving. This project will propose a solution by creating an algorithm that would process frames from a dashboard camera and determine which category from above can best describe the current state of the driver. The idea is to analyze the driving behavior in real time, and when unwanted behavior is detected, signal a warning appropriately. The latter part is not part of the proposed solution, which only addresses the driver category inference part.

In order for the solution to be successful, the accuracy of the image classification should be high (false positives in this case could cause this feature to be turned off by the driver since it would only increase the distraction of the driver even further). Today, cars do not have the compute or RAM capabilities of servers used in online systems. In order to be a useful feature, the category inference should be able to be performed in less than 2-3 seconds since the snapshot was taken. Therefore, a model with high accuracy, low Ram and CPU requirements and a high speed of inference would be ideal for this solution.

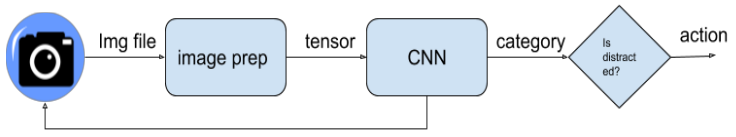


Fig 1.

As shown in figure 1., the end solution would be able to accept image frames of the driver from a dashboard camera, scale the image, convert it into a tensor, pass it through the model, which would output the state of the driver as one of the 10 categories described above. The category is passed to another logical unit, which will execute an action if the category is one of the distracted driver ones.

This project will focus on choosing an image classifier model and training it on the data provided in the Kaggle competition sponsored by State Farm.

The model will be trained and validated with the dataset provided. For that purpose, the provided training set will be split into two 80% / 20%. (training / validation) sets. The performance of the model will be determined with the help of the validation data set.

Metrics

Models will be evaluated by submitting the test dataset predictions to the Kaggle competition, which will calculate the multi-class logarithmic loss.

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} Yi \log(Pij) ,$$

Where N is the number of images in the test set, M is the number of image class labels (10), log is the natural logarithm, Yij is 1 if observation i belongs to class j and 0 otherwise, and Pij is the predicted probability that observation i belongs to class j.

II. Analysis

Data Exploration

The dataset consists of driver images taken from a dashboard camera divided into a training (22,424 images) and test (79,726 images) sets. All images have the same size – 480 x 640 pixels. The training images are categorized and labeled in one of the following categories:

- 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

The train and test data are split on the drivers such that one driver can appear in only one of the data sets.

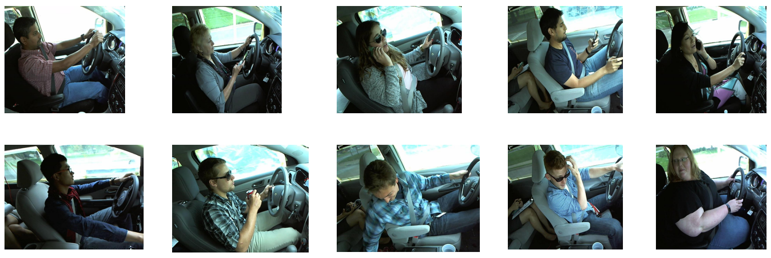
The file driver_imgs_list.csv associates an image from the training set with its class name (c0 – c9) and driver in the following format:

subject – the driver (p002, p003, etc)

classname – the class (c0 – c9)

img – the image file name, i.e. img_44733.jpg

An example of the images in the training set is shown below.



Analysis was made on the number of drivers in the dataset, the distribution of pictures based on drivers, distribution of images per category per driver.

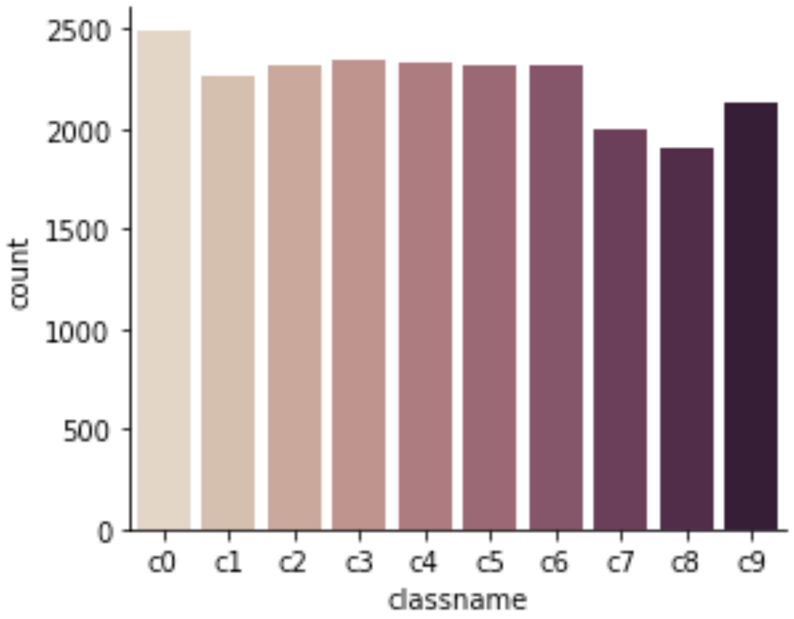


Fig 2

Fig 2 shows the distribution of categories without taking consideration of driver. The number of images in each category is similar to each other. There are two categories (c7 and c8) that stand out with the least images.

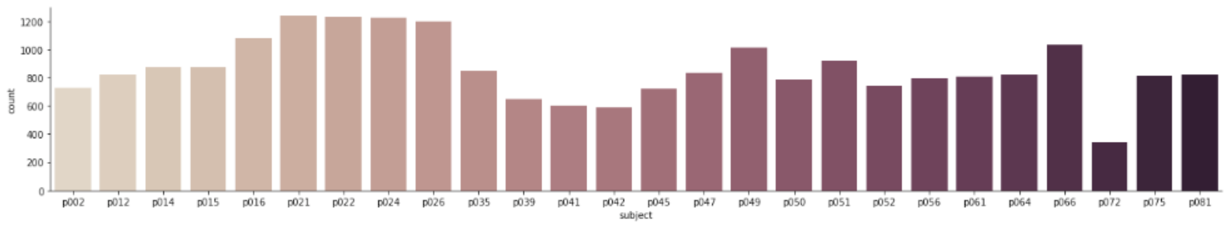


Fig 3

When the drivers are taken in consideration (fig. 3) we can see that category distribution is not as uniform. There are some drivers (p041, p042, p072) with significantly lower count of images than other drivers, while (p021, p022, p024, p026) have more images then the rest.

The distribution if image category per driver is shown in fig. 4 below. It shows that category distribution is not uniform. There are some drivers (p050 and p051) that have evenly distributed categories of images, but the count in them is significantly lower than it is for the rest of the drivers. Other drivers like p014 have categories (c8, c9) with much fewer images then the rest of the categories for those drivers

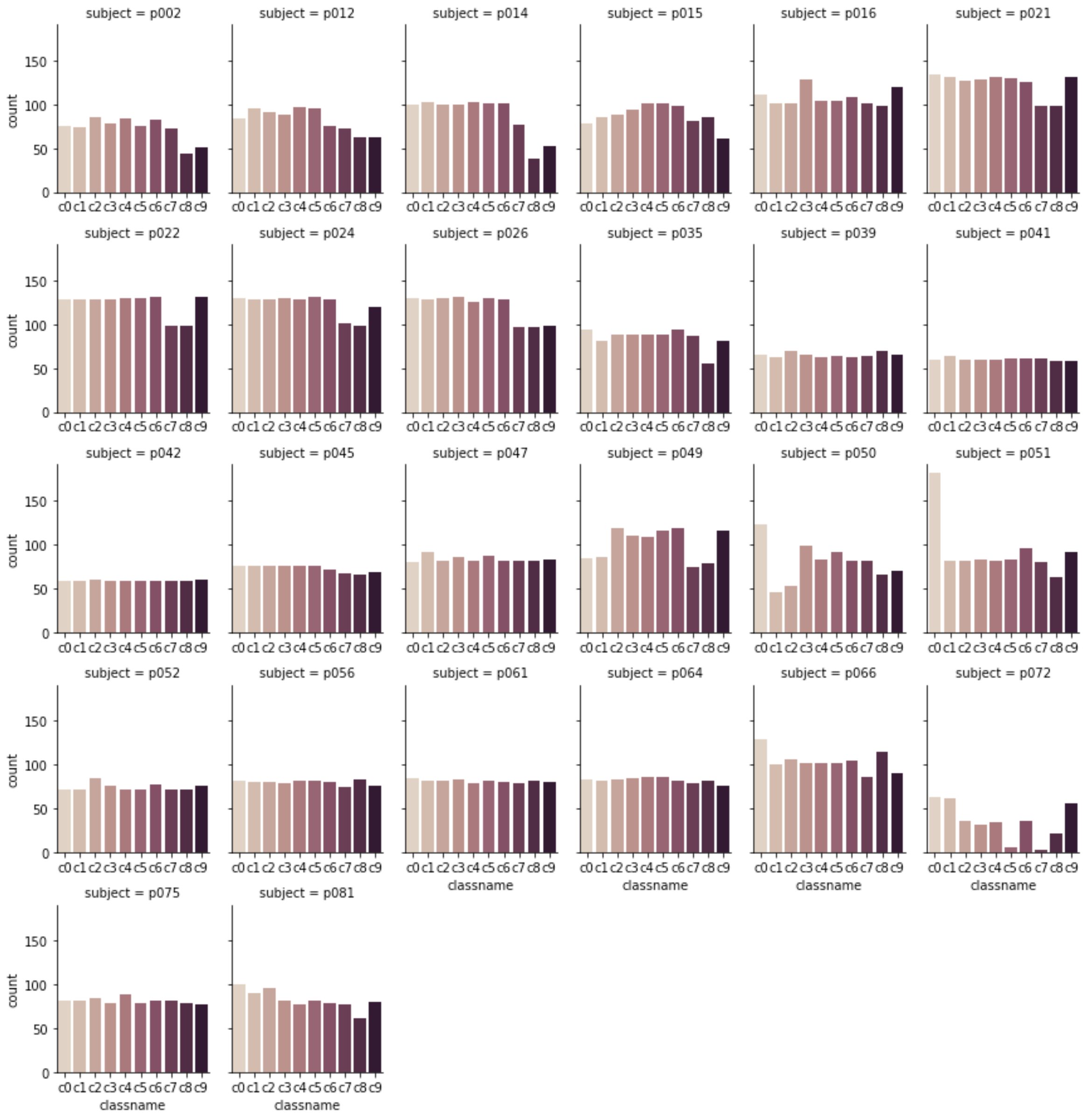


Fig 4

Overall, the training set has sufficient representation for each category and can be used for the purpose of training the image classification model.

Algorithms and Techniques

The proposed solution is an image classification problem, solutions for which in the last 5-7 years have proven that the application of deep learning models can result in high accuracy classifications. With the help of the ImageNet competition, multiple architectures have been created in the last 5-7 years, which have been reducing the classification error while improving the training speed.

Today, Keras provides easy access to pre-trained algorithms on the ImageNet set (https://keras.io/applications/):

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	99 MB	0.749	0.921	25,636,712	168
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

The ImageNet competition is a multi-class classification, which gives a probability for an image to be classified as each one of the classes. If the highest predicted probability for a category happens to be what the image category truly is, then it is considered Top-1

Accuracy. If, the true category is in the top 5 predicted classes, then it is considered a Top-5 Accuracy.

The table also contains the number of parameters, number of hidden layers and size of the model. These characteristics of the model are indicative of the computational complexity.

Models with higher accuracy and lower computational complexity are preferable.

The idea for the proposed solution is to select one of the above pre-trained models and optimize for the image set described for this problem. Based on that criteria, MobileNetV2 (M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.C. Chen, Google Inc., MobileNetV2: Inverted Residuals and Linear Bottlenecks. arXiv:1801.04381v3 [cs.CV] 2 Apr 2018) has been chosen.

MobileNetV2 is a convolutional neural network architecture optimized for fewer operations and less parameters, but provides high accuracy for image classification, detection and segmentation. It uses depthwise separable convolution as building blocks and uses linear bottlenecks between the layers and shortcut connections between the bottlenecks. Just with any CNN, the intuition is that the initial layers learn low level concepts, such as pixels, lines and with every subsequent layer the concepts become more complex, culminating to image categories on the output.

The following parameters can be tuned in order to improve the performance of the model:

1. batch size – number of samples presented at a time to the neural network.
2. Learning rate – to what degrees are the weights in the network with respect to the gradient of the loss function.
3. epochs – number of learning iterations over the whole training set
4. optimizer – the optimizer used to train the model
5. dropout rate – select the percent up to which neurons from the FCL will not be used
6. activation for fully connected layers (FCL) – the activation function used for the FCLs.
7. number of neurons for the fully connected layers
8. at what layer to start training the pre-trained model with the images from our dataset. Edge cases are start at the beginning or start after the top of the architecture, or anywhere in between.
9. Alpha – the width multiplier in the MobileNetV2 paper
10. Depth multiplier for the depthwise convolution, also from the MobileNetV2 paper

Benchmark

The proposed solution is essentially a solution for an existing Kaggle competition from two years ago. Because of that the competition leaderboard provides a great benchmark for the purpose of this project. The closer the score from the created model in this solution is to the leader on the leader board, the better. In order to be able to compare the results on the other side of the scale, a score of the prediction on the test dataset using uniform random probabilities will be utilized. The score achieved by the solution in this project needs to be better than the one achieved by the uniform random prediction. Once this is achieved, the score will be rated on how close it is to the leader, as shown in table 1 below.

	Private	Public
Top score	0.08739	0.08689
75th percentile	0.57999	0.58258
50th percentile	1.65041	1.50701
25th percentile	1.97218	1.78658
random	2.82342	2.82009

Tab. 1

The private score board is calculated with approximately 69% of the test data and the public score board is calculated with 31% of the data.

III. Methodology

Data Preprocessing

The images from the training and test datasets have the same size – 480 x 640 pixels. By default, the MobileNetV2 implementation in Keras (http://keras.io) expects the input data to be a tensor of (n, 224, 224, 3), where 224 x 224 is the size of each of the RGB channels.

The pre-processing steps are:

1. Convert image file to tensor
    a. Load the image from file and resize to 224 x 224 pixels

   b. Convert the image to a 4D tensor with shape (1, 224, 224, 3)
   c. Normalize tensor values in range of [0.0, 1.0] (by dividing each pixel value by 255.0)
2. Create two tensors – one containing all training images and another with all test images
3. Create a tensor with size (n, 10) for the target variable by converting the true labels of the training images to one-hot vectors
4. Split the tensors into train and validation datasets, as well as of the target tensor – making sure that no dataset contains any drivers from the other dataset.
5. Shuffle the records in the tensors
6. Save the resulting 6 tensors to files, so that the pre-processing step does not need to be performed multiple times, but rather loaded from disk after the initial processing.

At the end of the pre-processing steps, there is a 18,047 image training tensor and a 4,377 image validation tensor with corresponding target value tensors.

Because of the low number of drivers in the training set image augmentation will be used during the training process in order to reduce the effects of overfitting.

Implementation

1. Define the architecture of the network
   The architecture of the original MobileNetV2 network is used up to it's top.
   A global average pooling layer is added
   A sequence of two fully connected layers (FCLs), each followed by a dropout layer are added
   A fully connected layer with 10 neurons and softmax activation is added to calculate the categorical cross entropy for each category

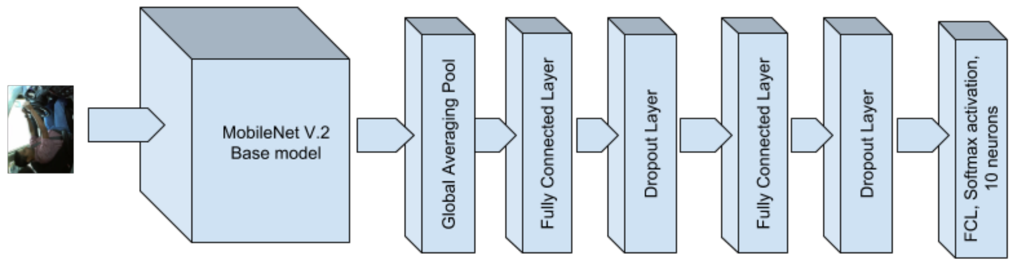


Fig. 5

2. Using scikit-learn grid search, and a small fraction of the training set (500 images) perform grid search on the parameters listed above

Parameter Group	Parameter	Value ranges	Optimal Value determined
1	Optimize	‘Adam’, ‘RMSprop’	‘Adam’
2	FCL activation function	‘relu’, ‘tanh’, ‘linear’	‘tanh’
2	FCL dropout rate	0.2, 0.3, 0.4	0.4
2	Number of neurons	512, 1024	512
3	batch size	16, 32, 64	16
3	Number of epochs	7, 15	7
3	Learning rate	0.0001, 0.001	0.0001
4	alpha	0.35, 0.50, 0.75, 1.0, 1.3, 1.4	0.75

Tab. 2

In order to reduce the number of permutations of the parameters, the grid search is performed with 4 parameter groups (the grouping is shown in table 2 above) in the order of their group number and uses the optimal parameters determined by the grid search in the previous parameter groups

3. Using the optimal values from the grid search, perform training on the whole dataset using augmented images in attempt to reduce overfitting. Load the optimal weights into the model
4. Using the pre-processed training set and the trained model, infer the categories for each image from the training set
5. Generate the submission file and submit to the Kaggle competition in order to get the value for the loss.
6. Compare the loss value with the benchmark results as defined above.

Refinement

Number of network layers to include in the training

The chosen model – MobileNetV2 is pretrained on the ImageNet dataset (http://image-net.org/about-overview) , which consists of 12 million images from 1000 categories. The goal was to utilize the knowledge gained from training on this dataset and transfer it to our problem. The choice that had to be made was what part of the weights in the base model should be included in the optimization calculation. In all cases, the weights from the trained model are loaded as default to the weights in the model.

- Freeze weights from all network layers in the base model.
- Include weights from all network layers in the training
- Choose a layer after the input layer and train the layers after that

The choice was to optimize the weights from all layers, since the nature of the images is quite different from the ImageNet dataset. Using the pretrained on ImageNet weights proved to reduce the number of epochs to less than 10.

Overfitting issues

Overfitting was an issue in training – while the training loss decreased for every epoch, the validation loss decreased in the first few epochs, after which it increased. This most probably is due to the quantity of images in the input data. The model was created and trained on a dataset 1000 times larger than the one we are training on.

In order to improve on that, image augmentation was utilized. The best results were achieved with the following parameters:

- Random shift horizontally: 25%
- Random shift vertically: 25%
- Rotation range: 20 degrees

The alpha parameter (width multiplier) thins the network when it is less than 1.0. This parameter was optimized with the help of grid search and the optimal value of 0.75 was used to train the model.

IV. Results

Model Evaluation and Validation

The training set was divided in two parts – approximately 80% training set and 20% validation set. Drivers were either part of the training set or the validation set so that cross training does not occur. The training set was used to train the model, while the validation set was used to score the trained model. This is done in order to produce a generalized model and not a model that is highly accurate for the input data it is trained on (overfit).

Models that produced the best validation score were used to infer the test dataset and the results of that scored for the final score.

Justification

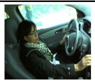
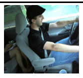
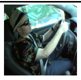
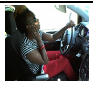
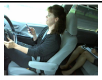
Using this approach, a model was created to determine the state of a driver based on a snapshot taken from a dashboard camera. The average time for inferring the category of an image is 10ms on a computer with a NVIDIA GK210 GPU. This result, even when accounting for the less capable hardware in vehicles, meets the speed requirement of 2-3 seconds.

The model achieved the following best scores on the training set:

Private score: 0.56259

Public Score: 0.48554

According to the benchmark shown in table 1, those scores are better than 75% of the scores submitted by the participants in the Kaggle competition.

Image	Safe Driving	Texting right	Phone right	Texting left	Phone left	Operating radio	Drinking	Reaching behind	Hair and makeup	Talking to passenger
	0.000080	0.000034	0.000029	0.000290	0.000005	0.999201	0.000005	0.000206	0.000007	0.000143
	0.005001	0.000128	0.000131	0.000282	0.000015	0.992926	0.000050	0.000019	0.000602	0.000846
	0.988371	0.008919	0.000097	0.000356	0.000016	0.000239	0.000066	0.000039	0.000076	0.001821
	0.000505	0.000032	0.034328	0.000071	0.001507	0.000244	0.027703	0.000019	0.935432	0.000158
	0.001516	0.002960	0.000014	0.991738	0.002669	0.000049	0.000526	0.000118	0.000189	0.000221

Tab 3.

Table 3 shows five of the training images and the corresponding predictions for each category. The last two of the five images are not 100% accurate, but correctly predict that the driver is not performing safe driving.

V. Conclusion

Loss Function Visualization

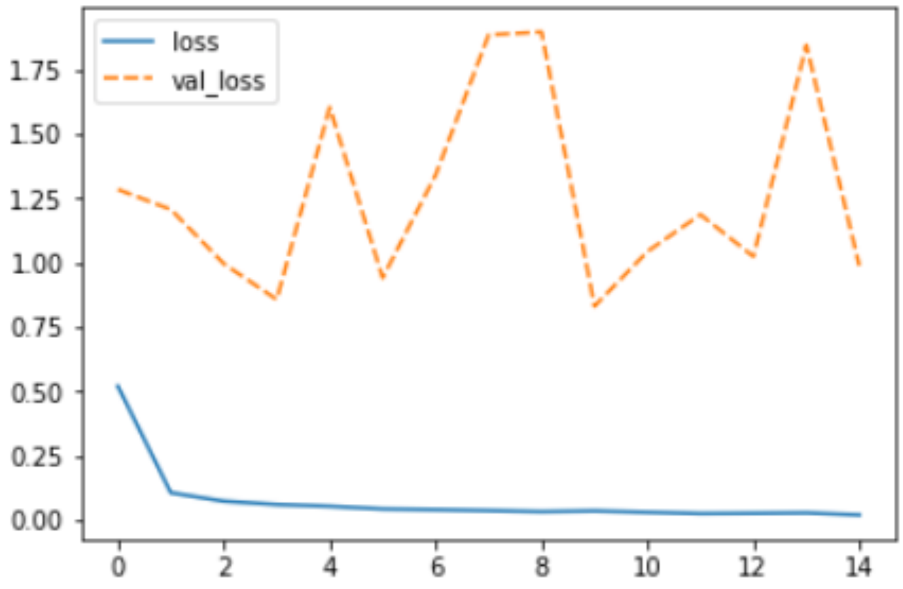


Fig. 6

Figure 6 visualizes the curves for the loss function during training set and validation. The training set curve is descending quite nicely as the epochs increase, which shows that the learning rate has been chosen well. When we look at the curve of the loss function on the validation set, we can see that the loss always is greater than the training loss and does not come close to it during the training process. This shows that the model is overfitted. As discussed earlier, aside of adding more sample to the training set, the only remedy is to augment the existing dataset by introducing variation to the images.

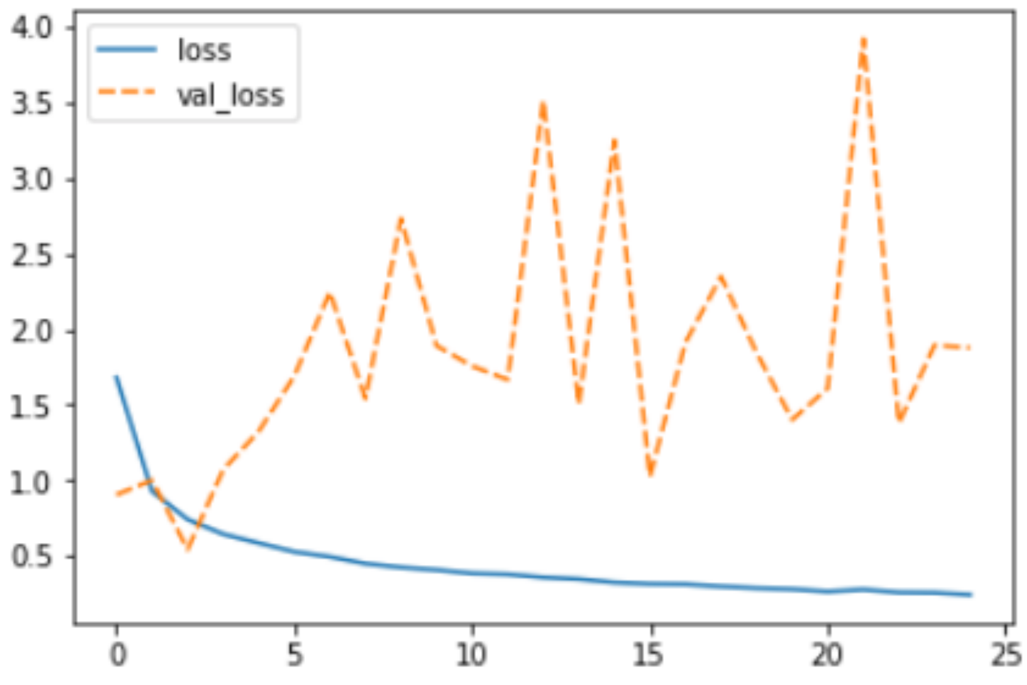


Fig. 7

Fig 7 shows the training and validation loss function after increasing the image augmentation as follows:

width shift range: 65%

height shift range: 65%

rotation range: 50 degrees

We see that the training loss now is higher than before, but the validation loss is smaller or equal to the training loss in the first three epochs, after which it gets larger and behaves similar to Fig. 6

This shows to prove that the model can benefit a lot from a larger dataset, since image augmentation only helps to an extent.

Reflection

During this project a dataset of images was used to create a deep neural network model to classify each image into 1 of 10 categories. A pretrained model (MobileNetV2) was used which was then trained on the provided dataset. The hyper parameters were optimized using grid search and image augmentation was utilized in an attempt to reduce overfitting. Overall the achieved result was better than 75% of the participants in a Kaggle competition taken place about 2 years earlier.

This project was interesting to me because it allowed to get acquainted with the various models created in the last 5-7 years for computer vision. It also underlined the importance of good data for creation of meaningful models. While there are datasets as ImageNet created, they can only be used as a starting point when creating a model for a specific real-world need, such as radiology, skin cancer detection, or even distracted driving detection. For each of these cases, a dataset should be created with sufficient number of samples from the studied domain, in order to create a model with high accuracy. A lot of innovation and cooperation is happening today and will continue to happen in finding ways of acquiring the required labeled images in a period of time allowing to achieve breakthrough products like the CT Scanners did during the 1970s.

Improvement

In the driver images there is a lot of information that is not relevant to the state of the driver – for example, the gender, or face characteristics (there are just 26 drivers in the dataset). One way for improvement is to extract the relevant information and train only on that – for example, if the resolution of the images is reduced, the irrelevant details would blend with the background and the relevant features, such as face direction, hands position will have a stronger weight.

Collecting more labeled images would help to achieve a more generalized model.

Performing the grid search on groups of parameters lowered the computational time but performing the optimization on all of the parameters at the same time could lead to more optimized values.

Performing the grid search on the whole dataset would also lead to finding even more optimal hyper-parameters.

I am positive that utilizing methods as described above would achieve a model with higher accuracy.