**Babeș-Bolyai University**

**Faculty of Mathematics and Computer Sciecne**

**Computer Vision and Deep Learning.**

**Drivers’ Distraction Detection.**

**Can Computer Vision Spot Distracted Drivers?**

**Students:** Zalum Camelia-Luciana

Ureche Florin-Alexandru

Umanschii Ianec

**Group:** 937

Cluj-Napoca, Romania

2021

Introduction

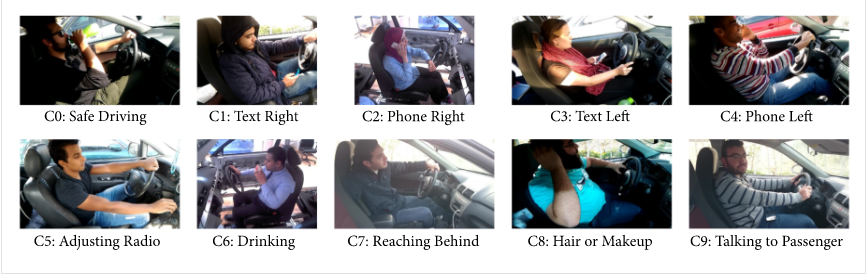
Convolutional Neural Network (**CNN**) is a deep [neural network](https://www.sciencedirect.com/topics/chemical-engineering/neural-network) originally designed for image analysis. Recently, it was discovered that the CNN also has an excellent capacity in sequent data analysis such as [natural language processing](https://www.sciencedirect.com/topics/computer-science/natural-language-processing). CNN always contains two basic operations, namely convolution and pooling. The [convolution operation](https://www.sciencedirect.com/topics/computer-science/convolution-operation) using multiple filters is able to extract features (feature map) from the data set, through which their corresponding spatial information can be preserved. The pooling operation, also called subsampling, is used to reduce the [dimensionality](https://www.sciencedirect.com/topics/computer-science/dimensionality) of feature maps from the convolution operation. Max pooling and [average pooling](https://www.sciencedirect.com/topics/computer-science/average-pooling) are the most common pooling operations used in the CNN. Due to the complicity of CNN, ReLu is the common choice for the [activation function](https://www.sciencedirect.com/topics/computer-science/activation-function) to transfer gradient in training by [backpropagation](https://www.sciencedirect.com/topics/computer-science/backpropagation).

According to the CDC motor vehicle safety division, [one of five car accidents](http://www.cdc.gov/motorvehiclesafety/distracted_driving/) is caused by a distracted driver. Sadly, this translates to 425,000 people injured and 3,000 people killed by distracted driving every year.

Our aim is to build a CNN that will be able to detect drivers’ distractions while driving. At fixed periods of time, a photo of the driver will be taken in order to categorize his behavior in 10 classes:

1. Safe-driving
2. Text right
3. Phone Right
4. Text left
5. Phone left
6. Adjusting radio
7. Eating/Drinking
8. Reaching something behind
9. Hair or Makeup
10. Talking to passenger

The goal is to predict the likelihood of what the driver is doing in each picture.



Dataset

The dataset images were taken from Kaggle (<https://www.kaggle.com/c/state-farm-distracted-driver-detection/data>). This dataset contains 22424 images categorized in 10 classes.

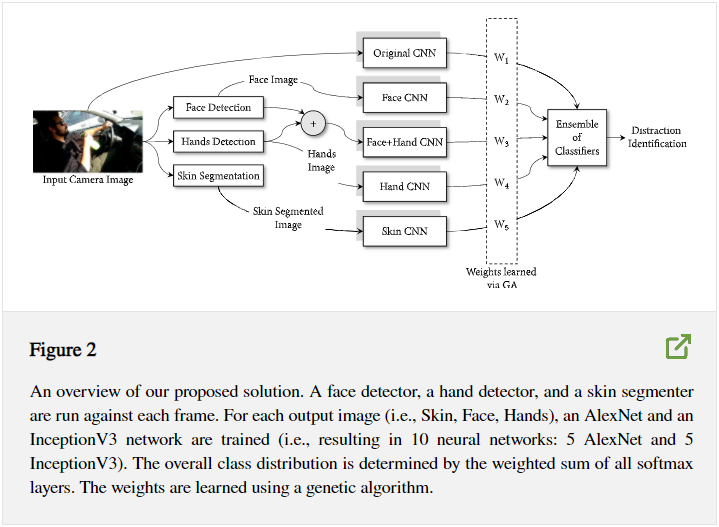


Other methods

# Driver Distraction Identification with an Ensemble of Convolutional Neural Networks

#### Proposed Method

Our proposed solution consists of a genetically weighted ensemble of convolutional neural networks. The convolutional neural networks are trained on raw images, skin-segmented images, face images, hands images, and “face+hands” images. On those five images sources, we train and benchmark an AlexNet network, an InceptionV3 network, a ResNet network having 50 layers, and a VGG-16 network. We fine-tune a pretrained ImageNet model (i.e., transfer learning) for these networks. Then, we evaluate a weighted sum of all networks’ outputs yielding the final class distribution using a genetic algorithm. The system overview is shown in Figure [2](https://www.hindawi.com/journals/jat/2019/4125865/fig2/).

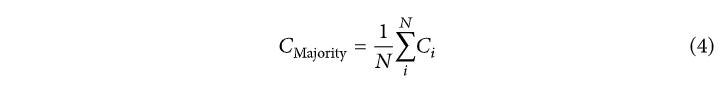


##### **Convolutional Neural Network**

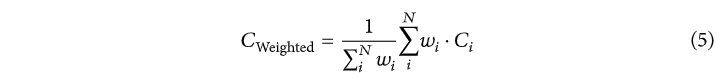
For distracted driver posture classification, we trained and benchmarked different neural networks architectures: an AlexNet, an InceptionV3, a ResNet network having 50 layers, and a VGG-16 network. Each network is trained on 5 different image sources (i.e., raw, skin, face, hands, and face+hands images).

We trained our AlexNet models from scratch. We did not use a pretrained model. As for InceptionV3, we performed transfer learning. We fine-tuned a pretrained model on the distraction postures. We removed the “logits” fully connected layer and replaced it with 10-neuron fully connected layer (i.e., corresponding to 10 driving postures). For all of our models, we used a gradient descent optimizer with an initial learning rate of 0.01. The learning rate decays linearly in each epoch with a step of (0.01-0.0001)/epochs. We trained the networks for 30 epochs. In each epoch, we divide the training dataset into minibatches of 50 images each.

Each classifier produces a class probability vector (i.e., output of the “softmax” layer), C1...CN, such that Ci is a vector having 10 probabilities (for 10 distraction classes) and N is the number of classifiers. In a majority voting system as in ([4](https://www.hindawi.com/journals/jat/2019/4125865/#EEq4)), it is assumed that all experts (i.e., classifiers) can equally contribute to a better decision by taking the unweighted sum of all classifier outputs.



However, that is not usually a valid assumption. In a weighted voting system as in ([5](https://www.hindawi.com/journals/jat/2019/4125865/#EEq5)), we assume that classifiers do not contribute equally to the ensemble and that some classifiers might yield higher accuracy than others. Therefore, there is a need to estimate the weights of each classifier’s contribution to the ensemble. Reference presents a variety of methods to estimate the weights. We opted to use a genetic algorithm (i.e., a search-based method).



In our genetic algorithm, a chromosome consists of N genes that correspond to the weights w1…wN . Our fitness function evaluates the Negative Log Likelihood (NLL) loss over a 50% random sample of the population. This helps prevent overfitting. Our population consists of 50 individuals. In each iteration, we retain the top 20% of the population and use them as parents. Then, we randomly select 10% of the remaining 80% of the population as parents. In other words, we have 30% of the population as parents. Now, we randomly mutate 5% of the selected parents. Finally, we cross-over random pairs of the parents to produce children until we have a full population (i.e., with 50 individuals). We ran the above procedure for only 5 iterations in order to avoid overfitting. We selected the chromosome with the highest fitness score (test against all data points, not 50%).

#### Conclusion

Distracted driving is a major problem leading to a striking number of accidents worldwide. Its detection is an important system component in semiautonomous cars. In this paper, we presented a robust vision-based system that recognizes distracted driving postures. We collected a novel publicly available distracted driver dataset that we used to develop and test our system. Our best model utilizes a genetically weighted ensemble of convolutional neural networks to achieve a 90% classification accuracy. We aim to provide a baseline performance for future research to benchmark against. We also showed that a simpler model (only using AlexNet) could operate in real-time and still maintains a satisfactory classification accuracy. Face, hands, and skin detection proved to improve classification accuracy in our ensemble. However, in a real-time setting, their performance overhead is much higher than their contribution.

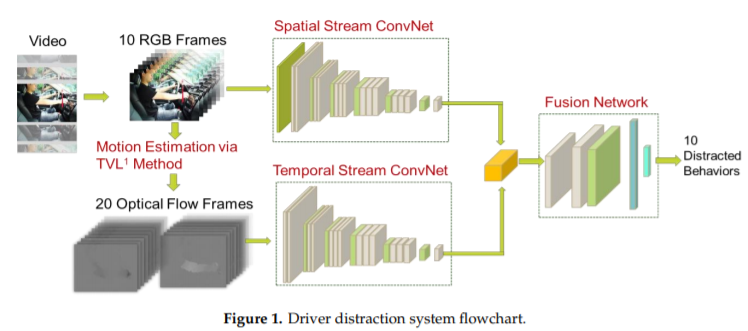
In a future work, we need to devise a better face, hands, and skin detector. We would need to manually label hand and face proposals and use them to train a Fast-RCNN (or, any other object detector) to localize both faces and hands in one shot and evaluate it against our existing CNN-based localization method.

<https://www.hindawi.com/journals/jat/2019/4125865/>

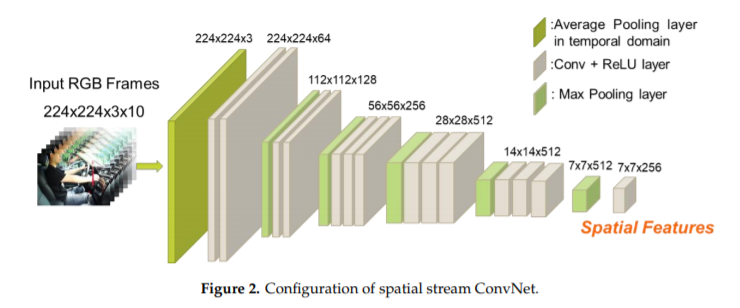
1. Driver Behavior Analysis via Two-Stream Deep Convolutional Neural Network

**Proposed Method**

For action recognition, temporal information is important, as well as spatial information. Motivated by work using temporal information, we proposed an architecture of two-stream convolutional networks for distracted detection as shown in Figure 1. The network is composed of three sub-networks: spatial stream ConvNet, temporal stream ConvNet, and a fusion network. Spatial stream ConvNet and temporal stream ConvNet are used to extract the spatial and temporal features, respectively, and the different features are integrated in the fusion network. By benefitting from transfer learning, the spatial stream ConvNet was designed based on the famous network configuration, VGG-16, and the pre-trained model on the ImageNet dataset can be applied. The average pooling in the temporal dimension is firstly performed on 10 consecutive RGB images, and the result is passed to the following layers of the spatial stream ConvNet. Via convolution operators performed in the convolutional layers, the spatial features are obtained from the feature map in the last convolutional layer. On the other hand, instead of using the famous 3D ConvNet that would suffer from large parameters and overfitting when there is lack of a large training dataset, the temporal features are obtained by extracting the motion information via the TVL1 optical flow from video frames and then a stack of consecutive flow images capturing vertical and horizontal motion information are input to the temporal stream ConvNet. Then, the temporal features are obtained from the feature map in the last convolution layer of temporal stream ConvNet. In the end, a fusion network, consisting of two convolutional layers and two fully connected layers, was designed to integrate the spatial and temporal features to classify 10 distracted behaviors.

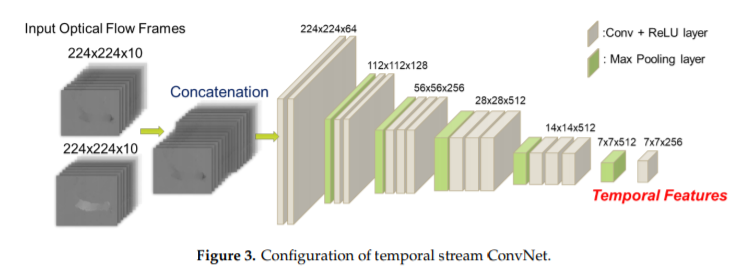


The convolutional neural network (CNN) plays an important role in deep learning models. In recent years, many studies relied on famous network architectures such as AlexNet, VGG-16, and GoogleNet. These networks usually consist of convolutional layers and pooling layers, followed by few fully connected layers. Some studies showed that better accuracy can be achieved using a deeper network. By considering the balance of performance and computation efficiency and the promising results, the spatial stream ConvNet was based on the network configuration of VGG-16 [21], which was designed for image classification and was proven to effectively extract the features of images layer by layer. Figure 2 shows the network configuration of spatial stream ConvNet. The network input is a stack of 10 consecutive RGB frames, each of which was resized to 224 × 224 pixels. By benefitting from the transfer learning of using the pre-trained model on the large-scale dataset, averaging pooling of the temporal direction is firstly performed on these input frames, and the resulting map with the size of 224 × 224 × 3 pixels is obtained. Following the average pooling layer, the network is composed of 13 convolution layers, while five max pooling and the ReLU activation function are set in each layer. Unlike other CNN models which set different kernel size in convolutional layers, a small kernel size of 3 × 3 pixels is set in all convolution layers which keeps the scale-invariant feature transform after convolution by using the same padding mechanism. The max pooling can help to extract the feature information of a larger area. Although the configuration of the VGG-16 is simple and effective, the large number of parameters (140 million) on fully connected layers is the main problem. This leads to high cost for the training and test process. Hence, in our work, unlike the original VGG-16 configuration, the fully connected layer was not used in the proposed spatial stream ConvNet. Here, a feature map with the size of 7 × 7 × 256 pixels from the last convolution layer is extracted as the spatial features and further processed in the following fusion network.



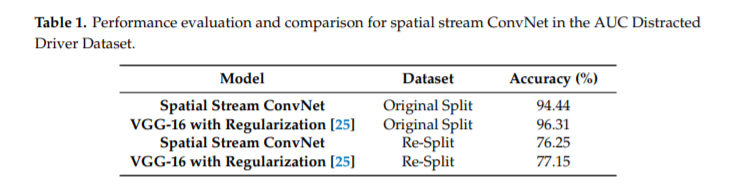
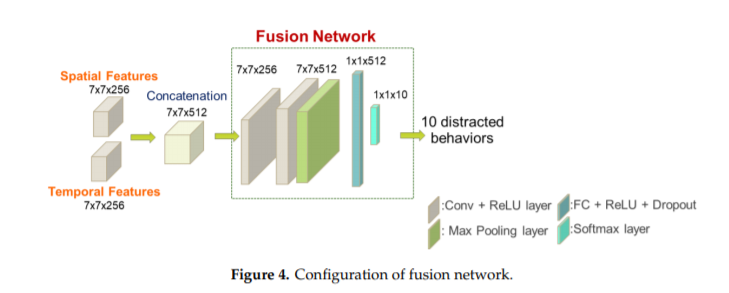
Since the parameters of CNN models are large, transfer learning, using the pre-training model to initialize the network, is applied in the training process. For driver distracted behavior analysis, the hand movements are the main changes. Hence, rather than using the pre-trained model on the ImageNet, the spatial stream ConvNet was pre-trained on the dataset that contains the videos of N (N = 10 in our study) actions containing obvious hand movements from thUCF-101 dataset [53]. Additionally, data augmentation, including cropping, rotating, horizontal flipping, and shifting, was performed to enlarge the size of dataset. Note that, in order to obtain the pre-trained model with the ability to extract discriminative features, a softmax layer with the size of 1 × 1 × N was added after the last convolutional layer, which was removed in the fine-tuned process. The network was trained by stochastic gradient descent with a learning rate of 0.0001, decay rate of 10−6, and momentum value 0.9. The batch size and number of epochs were set to 32 and 100, respectively.

Although distracted behavior analysis was studied in recent works only spatial information was considered while temporal information was discarded. In previous studies of action recognition, many network configurations were designed to integrate the spatial and temporal information, for example, 2D ConvNets + LSTM, 3D ConvNet, and two-stream network. Instead of using the famous 3D ConvNet [43,44] that would suffer from large parameters and overfitting when there is lack of a large training dataset, the two-stream network was used to design the temporal stream ConvNet. In order to extract the motion information, TVL1 optical flow with default parameters in OpenCV is firstly applied to obtain vertical and horizontal flow frames between two consecutive frames. Then, the optical frames of two directions are concatenated, and a stack of 20 flow images with the size of 224 × 224 × 20 pixels are input to the temporal stream ConvNet. Figure 3 shows the configuration of the temporal stream ConvNet. Following the input layer, the network is composed of 13 convolution layers with a kernel size of 3 × 3 pixels, while five max pooling and the ReLU activation function are set in each layer. Note that, in our work, unlike the original VGG-16 configuration, the Appl. Sci. 2020, 10, 1908 8 of 14 fully connected layers were not used in the proposed temporal stream ConvNet. Here, a feature map with the size of 7 × 7 × 256 pixels from the last convolution layer is extracted as the temporal features and fused with the spatial features in the following fusion network.

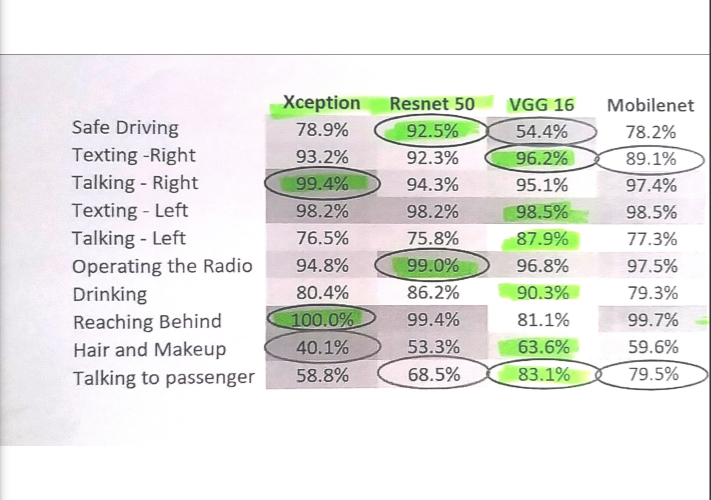


In order to train the temporal stream ConvNet, the pre-trained dataset, consisting of videos of N (N = 10) actions involving hand motions from the UCF-101 dataset, as used in the training process of the spatial stream ConvNet, was applied. Data augmentation involving random cropping and horizontal flipping was also performed to enlarge the dataset. The vertical and horizontal motion information was estimated between two frames via TVL1 optical flow. Additionally, in order to reduce the impact caused by the camera movement, the mean flow frame of each direction is calculated, and the flow frames were subtracted from the corresponding mean frame. For the pre-training process, a softmax layer with the size of 1 × 1 × N was added after the last convolutional layer, which was removed in the fine-tuned process, and the hyper-parameters were the same as used in the spatial stream ConvNet. The pre-trained model was then obtained, which was used to initialize the temporal stream ConvNet in the training process of the whole network.

After the spatial and temporal stream ConvNet, the spatial and temporal feature maps with a size of 7 × 7 × 256 pixels are obtained. In other to fuse different modalities, score-level and feature-level fusion are common methodologies. Since the dimension of features is high, i.e., 25,088, rather than using either the manual-defined weights or weights obtained via the optimization method, e.g., genetic algorithm (GA), at the score level, a fusion network was designed to fuse features in the feature level. Figure 4 shows the configuration of the fusion network. Two kinds of feature maps are concatenated in the third dimension, and then the resulting feature map with a size of 7 × 7 × 512 pixels is input to the following convolutional layer. Since the CNN model as used in the spatial and temporal stream ConvNet has promising ability of feature extraction, the fusion network was not designed with deep layers in order to reduce the risk of overfitting. In our study, the fusion network consisted of two convolutional layers, as well as one pooling layer and two fully connected layers with sizes of 1 × 1 × 512 and 1 × 1 × 10, respectively. The kernel size of the first and second convolutional layers was 1 × 1 and 3 × 3 pixels, and the ReLU activation function was applied to both convolutional layers. The classification result of 10 distracted behaviors is obtained in the softmax layer.



Experiments



Architecture

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_1 (Conv2D) (None, 224, 224, 64) 1792

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D) (None, 224, 224, 64) 36928

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 112, 112, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_3 (Conv2D) (None, 112, 112, 128) 73856

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_4 (Conv2D) (None, 112, 112, 128) 147584

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 56, 56, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_5 (Conv2D) (None, 56, 56, 256) 295168

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_6 (Conv2D) (None, 56, 56, 256) 590080

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_7 (Conv2D) (None, 56, 56, 256) 590080

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_3 (MaxPooling2 (None, 28, 28, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_8 (Conv2D) (None, 28, 28, 512) 1180160

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_9 (Conv2D) (None, 28, 28, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_10 (Conv2D) (None, 28, 28, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_4 (MaxPooling2 (None, 14, 14, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_11 (Conv2D) (None, 14, 14, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_12 (Conv2D) (None, 14, 14, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_13 (Conv2D) (None, 14, 14, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_5 (MaxPooling2 (None, 7, 7, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 25088) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 4096) 102764544

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 4096) 16781312

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 10) 40970

=================================================================

Total parameters: 134,301,514

Trainable parameters: 134,301,514

Non-trainable parameters: 0

Proposed Method

Our proposed solution consists of a genetically weighted ensemble of convolutional neural networks. The convolutional neural networks are trained on raw images from the dataset. We’ve trained and benchmarked a VGG-16 Network.



VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

**Disadvantages of VGG 16:**

1. It is very slow to train (the original VGG model was trained on Nvidia Titan GPU for 2-3 weeks).
2. The size of VGG-16 trained ImageNet weights is *528* MB. So, it takes quite a lot of disk space and bandwidth that makes it inefficient.

The input to the network is an image of dimensions (224x224x3). The first two layers have 64 channels of 3x3 filter size and same padding. Then after a max pool layer of stride (2, 2), two layers which have convolution layers of 128 filter size and filter size (3x3). This followed by a max pooling layer of stride (2, 2) which is same as previous layer. Then there are 2 convolution layers of filter size (3x3) and 256 filters. After that there are 2 sets of 3 convolution layer and a max pool layer. Each has 512 filters of (3x3) size with same padding. This image is then passed to the stack of two convolution layers. In these convolution and max pooling layers, the filters we use are of the size 3x3 again. There is a padding of 1-pixel (same padding) done after each convolution layer to prevent the spatial feature of the image. After the stack of convolution and max-pooling layer, we got a (7x7x512) feature map. We flatten this output to make it a (1x1x 25088) feature vector. After this there are 3 fully connected layer, the first layer takes input from the last feature vector and outputs a (1x4096) vector, second layer also outputs a vector of size (1x4096) but the third layer output a 10 channels for 10 classes which we have to compute, then after the output of 3rd fully connected layer is passed to softmax layer in order to normalize the classification vector. All the hidden layers use ReLU as its activation function. ReLU is more computationally efficient because it results in faster learning and it also decreases the likelihood of vanishing gradient problem.

For optimization we’ve used *The****Adam optimization algorithm.* It** is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. When introducing the algorithm, the authors list the attractive benefits of using Adam on non-convex optimization problems, as follows:

* Straightforward to implement
* Computationally efficient
* Little memory requirements
* Invariant to diagonal rescale of the gradients
* Well suited for problems which are large in terms of data and/or parameters
* Appropriate for non-stationary objectives
* Appropriate for problems with very noisy/or sparse gradients
* Hyper-parameters have intuitive interpretation and typically require little tuning

Adam is a popular algorithm in the field of deep learning because it achieves good results fast. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods.

**Parameters:**

**Alpha**: Also referred to as the learning rate or step size. It is the proportion that weights are updated (e.g. 0.001). Larger values (e.g. 0.3) results in faster initial learning before the rate is updated. Smaller values (e.g. 1.0E-5) slow learning right down during training

**beta1**: The exponential decay rate for the first moment estimates (e.g. 0.9).

**beta2**: The exponential decay rate for the second-moment estimates (e.g. 0.999). This value should be set close to 1.0 on problems with a sparse gradient (e.g. NLP and computer vision problems).

**Epsilon**: Is a very small number to prevent any division by zero in the implementation (e.g. 10E-8). Further, learning rate decay can also be used with Adam.

For loss we’ve used Cross-Entropy Loss Function

where M is the number of classes (e.g. 10 in our case),  is the model's prediction for that class (e.g. the output of the softmax for class i) and yi is the correct prediction. Due to the fact that the labels are one-hot encoded and y is a 10×1 vector of 1 and 0. Thus, out of the whole sum only one term will actually be added: the one with yi=1.

Bibliography

1. Lecture Notes from *Computer Vision and Deep Learning Class*, 2020-2021

*2. Andrew Ng’s courses* from [Coursera.org](coursera.org)

3. Xiangyu Zhang, Jianhua Zou, Kaiming He and Jian Sun, *Accelerating Very Deep Convolutional Networks for Classification and Detection (*<https://arxiv.org/abs/1505.06798>)

4. Vision-based Driver Behaviour Analysis - Chao Yan School of Electrical Engineering, Electronics & Computer Science University of Liverpool(https://core.ac.uk/download/pdf/80777305.pdf)

5. Driver Distraction Identification with an Ensemble of Convolutional Neural Networks (https://www.hindawi.com/journals/jat/2019/4125865/)

6. Adam: A Method for Stochastic Optimization (https://arxiv.org/pdf/1412.6980)