Comparative study for Drivers Drowsiness Detection using Deep Learning Models

Ashwanthika Umasankar[1], Anusha Vobbilireddy[2], Devendher Mandala[3]
[1]University of Texas at Arlington, TX
[2]University of Texas at Arlington, TX
[3]University of Texas at Arlington, TX

Abstract

Increased demand for mobility has necessitated faster transportation growth in recent years. Automobiles are an essential way of commuting for most people. Although the automobile has altered people's lifestyles and made routine activities more convenient, it is also connected to a range of negative consequences, such as road accidents. As we know, falling asleep while driving is very dangerous not only for the driver but also for others. Numerous accident cases have been reported due to the driver falling asleep while driving for maybe even a second. The most disturbing issue is that drowsy driving isn't merely falling asleep behind the wheel. When a driver is not paying full attentionto the road, drowsy driving can be as simple as a momentary state of unconsciousness. Our approach to this issue is to create a detection system that identifies the key aspects of drowsiness and sends out an alert before it's too late utilizing image detection and Deep Learning. We have used three Deep Learning methodologies for the implementationof our project. They are:

- CNN using Keras.
- VGG16 using Keras.
- CNN using NumPy.

Introduction

Drowsiness, commonly referred to as sleepiness, is a biological state in which the body is making a transition from an awake to a resting state. A driver may lose focus at this point and will be unable to perform the necessary actions while driving. If proper rest is not taken by the driver before driving and if they are driving for long hours it might result in drowsy driving. As we know, the person driving is in charge of the traffic and its safety. In addition to the passengers in the car, the driver is accountable for himself and to the other people surrounding him. Drowsiness is a common human feature that many individuals overlook when it comes to their safety. Road accidents have become very common and the reason is an increase in the number of vehicles on the road.

Driver drowsiness is the main cause of road accidents all over the world and there is enough evidence to support this. These tragedies have motivated many re- searchers throughout the world to look at methods for detecting and alerting drowsiness early on. There are evident indications that a person is drowsy, such as yawning frequently, inability to keep eyes open, and the head swaying forward. This study describes a set of deep learning approaches that can be used to create accurate and reliable judgments for detecting driver drowsiness. Our model focuses on detecting drowsiness the person is unable to open their eyes while driving. The rest of this paper is organized as follows: Section I has Dataset description. Section II has Project Description. Section III talks about the analysis we have done and Finally, conclusions are provided in Section V.

Section I: Dataset Description

We have taken our dataset from Kaggle. The dataset contains images of drowsiness-related data. Our dataset considers the movement of eyes which helps in identifying whether a person's eyes are open or closed. If the person's eyes are closed for a long duration then we can conclude that the person is drowsy. We have images taken during different times of the day. The images are categorized into Open Eyes, Closed Eyes. The categorized data has two labels, open eyes that contains 700 images and closed eyes contain 699 images.

Section II: Project Description

Description:

The steps involved in building the model are:

- Exploringthe data: Extract facial features.
- Train the model using CNN/VGG16 that identifies facial features like closed eyes or opened mouth indicating distraction with the driver.
- Test the model which involves converting videos into images that will help us to classify whether the person is drowsy or not.
- Obtain accuracy and plot the metrics.
- Test the model by using haarcascade that identifies the facial features such as the left eye and right eye. Post which it will identify whether the eyes are open or close

Convolutional neural networks: A Convolutional Neural Network (ConvNet/CNN) is a Deep Technique that can take in an image as input, assign value (learnable weights and biases) to distinct aspects/objects in the image, and distinguish one from the other. When compared to other classification methods, the amount of pre-processing required by a ConvNet is significantly less. While basic approaches require hand-engineering of filters, ConvNets can learn these filters/characteristics with enough training.

A ConvNet usually has 3 types of layers:

1. Convolutional Layer (CONV)
2. Pooling Layer (POOL)
3. Fully Connected Layer (FC)

The ConvNet's goal is to compress the images into a format that is easier to process while preserving elements that are important for obtaining a decent prediction. To construct our ConvNet model, we have utilized max pooling, zero padding, and a stride value of zero.

Max Pooling: is a down sampling (pooled) feature map created by calculating the maximum value for sections of a feature map? After a convolutional layer, it's commonly used. It adds a small amount of translation invariance, which means that slight changes in the image will not have a substantial impact on the values of most pooled outputs.

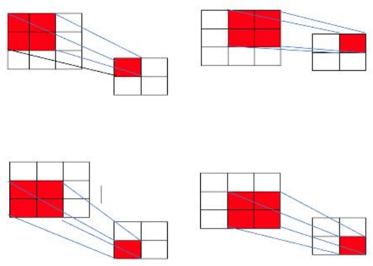


Figure 2.1

Stride: Stride is a component of convolutional neural networks, which are neural networks tailored for image and video compression. Stride is a filter parameter neural network that controls the amount of movement in an image or video. If the stride of a neural network is set to 1, for example, the filter will move one pixel (or unit) at a time. Because the size of the filter has an impact on the encoded output volume, the stride is frequently set to a whole number rather than a fraction or decimal.

Padding: Padding is a concept used in convolutional neural networks to describe how many pixels are added to an image when it is processed by the CNN kernel. If the padding in a CNN is set to zero, for example, every pixel value added will have the value zero. If the zero padding is set to one, a one-pixel border with a pixel value of zero will be added to the image.

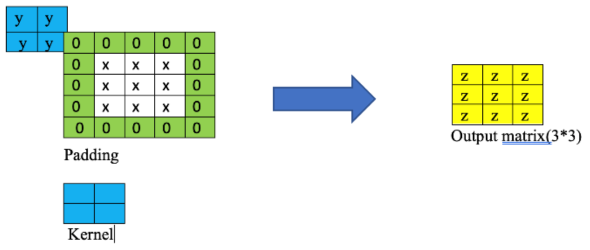


Figure 2.2

Activation function: In a neural network, an activation function specifies how the weighted sum of the input is transformed into an output from a node in a layer. The activation function is sometimes known as a "transfer function." The activation function is referred to as a "squashing function" if its output range is restricted. Many activation functions are nonlinear, and this "nonlinearity" in the layer or network design is referred to as "nonlinearity." The activation function chosen has a significant impact on the neural network's capabilities and performance, and different activation functions may be utilized in different portions of the model. The same activation function is used by all hidden levels. The activation function used by the output layer differs from that used by the hidden layers, and it is determined by the type of prediction required by the model. We employed RelU and Sigmoid, which are two different forms of activation functions used in neural networks. A node that implements this activation function is referred to as a rectified linear activation unit. Often, networks that use the rectifier function for the hidden layers are referred to as rectified networks.

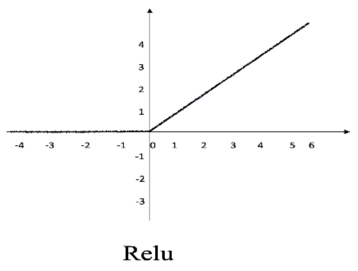


Figure 2.3

Because its domain is the set of all real numbers and its range is the set of all real numbers, the sigmoid function is also known as a squashing function (0, 1). As a result, the output is always between 0 and 1 if the function's input is either a very large negative number or a very large positive number.

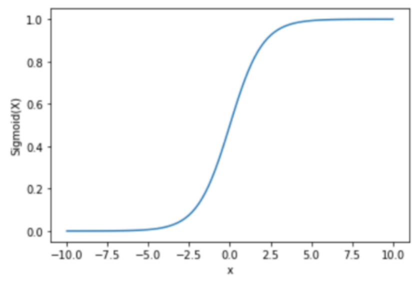


Figure 2.4

Fully Connected: Fully Connected Layers are the network's final layers. The output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer, is the input to the fully connected layer. The final (and any) Pooling and Convolutional Layer produces a three-dimensional matrix, which can be flattened by unrolling all of its values into a vector. We first convert all the images into grey scale, because basic CNN doesn't accept colored images.

- We have used are using the batch size of 128, so that all the images divide according to batch size.
- We are taking the input data as image and it is reprocessed by zooming, flipping and re-scaling it. We have defined our model with 1 input layer, 2 hidden layers, fully connected layers and one output layer.
- We are using Conv2D for convolution and we are taking input size as 224*224.
- Maxpooling is done on the input which is taken from the previous layer.

- For the hidden layers we have used the activation function” ReLU’ and for the output layer we have used” SoftMax” activation function
- We have used” categorical cross entropy” loss function and ”Adam” as an optimizer function.

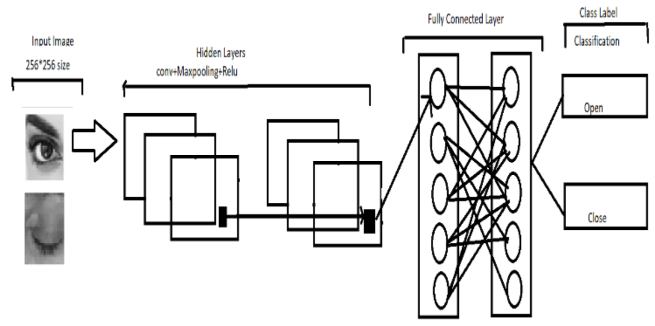


Figure 2.5

VGG 16: The VGG16 architecture is a convolutional neural network (CNN). It is regarded as one of the best vision model architectures ever created. The most distinctive feature of VGG16 is that, rather than having a huge number of hyper-parameters, they focused on having 3x3 filter convolution layers with a stride 1 and always used the same padding and max-pool layer of 2x2 filter stride 2. Throughout the architecture, the convolution and max pool layers are arranged in the same way. It has two FC (completely connected layers) in the end, followed by a softmax for output. The 16 in VGG16 relates to the fact that it contains 16 layers with different weights. This network is quite huge, with approximately 138 million (estimated) parameters.

In our model:

- We are re-sizing all the images to 224 x 224.
- We are taking the training dataset, and preprocessing it by, rescaling, zooming and flipping the images.
- We are taking 13 convolutional layers and we downsample them by using 5 max pooling layers and 3 fully connected layers used in connecting them all.
- After several layers we flatten it.
- We have used ”categorical cross entropy” loss function and ”Adam” as an optimizer function.

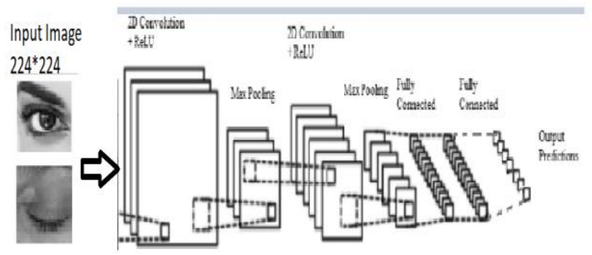


Figure 2.6

CNN using Np-array: Our primary goal is to build a CNN, based on the architecture shown in the illustration above and test its capabilities on the MNIST image dataset. This time, however, we won’t use any of the popular DL frameworks. Instead, we will take advantage of NumPy — a powerful but low-level library for linear algebra in Python. Here,

- We are converting our images to gray scale then we are resizing them to 48*48 pixels. We then convert all the images to a dictionary format which contains image pixel arrays, its actions i.e., closed/open eye and repository variable that says what type of our data is train/test. Model
- We have used convolution layers with forward and backward propagation.
- We have used maxpooling layer for down sampling data.
- We have used SoftMax activation function .

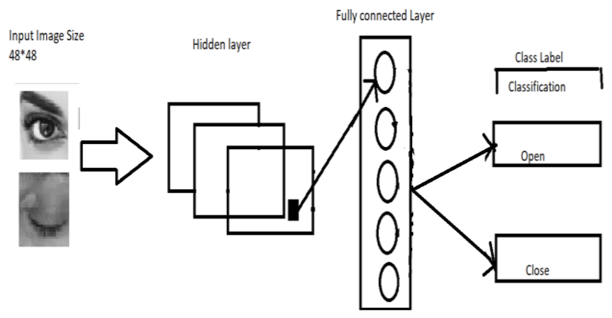


Figure 2.7

Difference between your project and the main projects of your references: Our project is a comparative study. We are comparing the model we have created using NumPy with the models that are built using inbuilt Deep learning libraries and frameworks. The existing papers does not propose a comparative study with a model that was built using np arrays.

Difference in Performance between your project:

1. **Report on test data using CNN using NumPy:**
   - We get the accuracy score: 0.4954128440366973
   - Confusion Matrix:

```
[[  1 108]
 [  2 107]]
```

   - Classification report:

```
              precision    recall  f1-score   support

      Closed       0.33      0.01      0.02       109
        Open       0.50      0.98      0.66       109

    accuracy                           0.50       218
   macro avg       0.42      0.50      0.34       218
weighted avg       0.42      0.50      0.34       218
```

2. **Report on the test data for VGG16 using Keras:**

- We get the accuracy score :0.944954128440367
- Confusion Matrix:
  [[104   5]
   [7 102]]
- Classification report**:**

```
         precision    recall  f1-score   support

   Closed       0.94      0.95      0.95       109
     Open       0.95      0.94      0.94       109

  accuracy                          0.94       218
 macro avg       0.95      0.94      0.94       218
weighted avg     0.95      0.94      0.94       218
```

3. **Report on the test data for CNN using Keras**

- We got the accuracy score : 0.8761467889908257
- Confusion Matrix:
  [[82 27]
   [0 109]]
- Classification report:

```
         precision    recall  f1-score   support

   Closed       1.00      0.75      0.86       109
     Open       0.80      1.00      0.89       109

  accuracy                          0.88       218
 macro avg       0.90      0.88      0.87       218
weighted avg     0.90      0.88      0.87       218
```

Section III: Analysis

What did I do well ?

We have built 3 models and the model that we built using **NumPy** without using keras framework gives us good accuracy. The other two models built using Keras framework are also performing good as shown in the classification report.

What could I have done better?

- Including more facial features for the detection such as yawn, head movements identification in the model to detect the drowsiness more accurately.
- By implementing more layers in our CNN using nparry model to get a better accuracy.

Future scope

Our future scope is to integrate all the models into the rea world.

Section IV: Results

Results for CNN and VGG16 using keras:

Once the training of the model is complete, we can go ahead with testing. For testing the model, we import the saved train model into the test file. We have used haarcascade for image identification in our test model. Haarcascade helps us in identifying facial features and extracting them. We use it to identify the eyes from a person's face. Upon executing the test file, a video recorder opens that checks whether the persons eyes are open/ close and it computes a score. If the person's eyes are closed for a count >=16 it concludes that the persons eyes are closed indicating that the driver has fallen asleep and it rings an alarm and captures a picture.

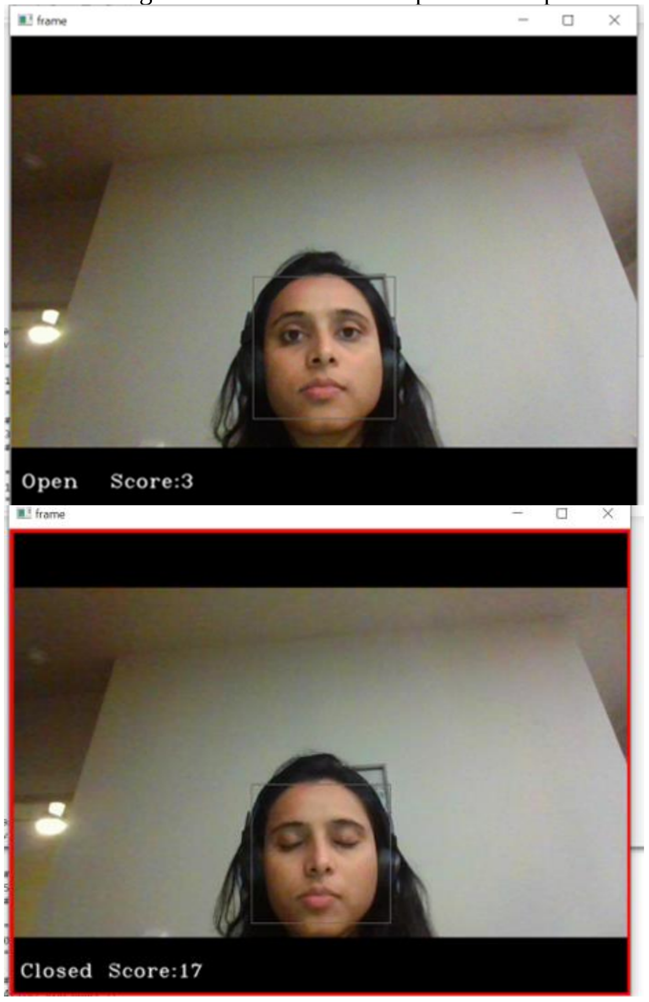
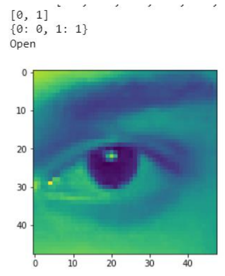


Figure 4.1

Results for CNN using NumPy:

Our test file predicts whether the given test image is open or closed.



```
[0, 1]
{0: 0, 1: 1}
Open
```

Figure 4.2

Section IV: Conclusion

Thus, our comparative study on three models for driver's drowsiness using open/closed eye dataset reveals that, VGG 16 performs good with 96% accuracy as opposed to CNN built with keras that produces 87% and CNN using NumPy that produces 50% accuracy.

References

1. https://www.kaggle.com/code/adinishad/driver-drowsiness-using-keras
2. https://data-flair.training/blogs/python-project-driver-drowsiness-detection-system/
3. https://en.wikipedia.org/wiki/Driver_drowsiness_detection
4. https://pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/
5. https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/
6. https://en.wikipedia.org/wiki/Convolutional_neural_network
7. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
8. https://sci-hubtw.hkvisa.net/10.1007/s00521-020-05209-7
9. https://scihubtw.hkvisa.net/10.1109/ACCESS.2019.2936663
10. https://towardsdatascience.com/drowsiness-detection-with-machine-learning-765a16ca208a
11. https://sci-hubtw.hkvisa.net/10.1007/s11042-018-6378-6
12. https://www.researchgate.net/profile/U-Srinivasulu-Reddy/publication/338251837_Deep_CNN_A_Machine_Learning_Approach_for_Driver_Drowsiness_Detection_Based_on_Eye_State/links/5f012fc192851c52d619b0cb/Deep-CNN-A-Machine-Learning-Approach-for-Driver-Drowsiness-Detection-Based-on-Eye-State.pdf
13. https://ieeexplore.ieee.org/document/8014793
14. https://www.sciencedirect.com/science/article/pii/S1877050918304137
15. https://www.mdpi.com/2076-3417/11/18/8441
16. https://www.hindawi.com/journals/cin/2020/7251280/
17. https://www.hindawi.com/journals/scn/2021/5383573/
18. https://www.hindawi.com/journals/jat/2020/8851485/
19. https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/
20. https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/
21. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
22. https://deepai.org/machine-learning-glossary-and-terms/padding#:~:text=What%20is%20Padding%20in%20Machine,will%20be%20of%20value%20zero
23. https://paperswithcode.com/method/max-pooling
24. https://medium.com/ai-techsystems/driver-drowsiness-detection-using-cnn-ac66863718d
25. https://github.com/opencv/opencv/tree/master/data/haarcascades
26. https://towardsdatascience.com/fine-tuning-pre-trained-model-vgg-16-1277268c537f
27. https://www.geeksforgeeks.org/vgg-16-cnn-model/
28. https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c
29. https://www.mygreatlearning.com/blog/introduction-to-vgg16/