Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12386 p-ISSN : 2541-044X

Drowsy Detection in the Eye Area using the Convolutional Neural Network

Alessandro Benito Putra Bayu Wedha[1)], Ben Rahman[2)*], Djarot Hindarto[3)], Bayu Yasa Wedha[4)]
[1)] Bina Nusantara (BINUS ASO), Indonesia
[2,3,4)] Fakultas Teknologi Komunikasi dan Informatika, Universitas Nasional, Indonesia
[1)] alessandro.wedha@binus.ac.id, [2)] ben.rahman@civitas.unas.ac.id,
[3)]djarot.hindarto@civitas.unas.ac.id,
[4)] bayu.yasa@civitas.unas.ac.id

Submitted : Apr 1, 2023 | **Accepted** : Apr 14, 2023 | **Published** : Apr 29, 2023

Abstract: Detection of a drowsy driver is an important aspect of driving safety. For this reason, it is necessary to have technology to carry out early detection before fatigue occurs. Mainly focused on driver fatigue that occurs at night. Analysis can be done quickly and accurately. These conditions can be sent via data so that they can be monitored and analyzed in real time. The results of the analysis can be sent by communication via the internet network. In addition, it functions as an early warning and can be used as logging or records that can be stored. This research does not discuss data communication but makes a prototype for detecting sleepy drivers. Prototype created using the Convolutional Neural Network Algorithm. The detection area is in the eye and testing is carried out with the brightness level of the light. In this study, building a prototype to detect signs of driver fatigue using the Convolutional Neural Network algorithm. The detection area used is in the eye, by testing at different light brightness levels. The dataset used in this study consists of a series of eye images, which are divided into two classes, namely open eyes, and closed eyes. After conducting the training process on Convolutional Neural Network, we get results of detection accuracy reaching 90%.

Keywords: Convolutional Neural Network; Close Eyes; Detection Drowsy; Fatigue; Open Eyes; Prototype

INTRODUCTION

In recent years, driver fatigue has become a significant problem for road safety. The World Health Organization (WHO) estimates that around 1.35 million people die each year because of road traffic accidents (Mwale et al., 2023), (Chen & James, 2022), with driver fatigue being a factor in many of these accidents. In this research, have investigated ways to detect and prevent driver fatigue, especially when driving at night, to try to reduce the occurrences as claimed above. Traffic accidents caused by sleepy drivers are one of the main causes of road accidents. This occurs when the driver is unable to maintain concentration and focus while driving, causing fatigue and eventually drowsiness. This condition can affect the driver's ability to make decisions and respond to road situations, which can eventually result in a fatal accident. Therefore, it is important to understand the dangers of drowsy driving and take appropriate precautions, such as getting enough rest and avoiding driving for too long. Several detection methods for drowsiness are as follows:

*name of corresponding author

1. Pupil Movement Detection, method utilizes eye-tracking technology to detect pupil movement in the driver's eye (Cheng et al., 2022). By measuring eye movements, the system can determine whether the driver is sleepy or not.
2. Head Movement Detection, method utilizes a camera to observe the driver's head movement (Zheng et al., 2022). If drivers experience fatigue or drowsiness, their head movements will become unstable or often experience a significant reduction.
3. Steering Pressure Detection, method utilizes the pressure sensor on the steering wheel to detect whether the driver is holding the steering wheel firmly or not. If the pressure generated is insufficient, the system may conclude that the driver is tired or drowsy.
4. Sound Analysis This method utilizes voice recognition technology to measure whether the driver makes certain sounds indicating fatigue or sleepiness. Examples include the sound of sighs or heavy breathing sounds.
5. Body Movement Detection This method utilizes the camera to record the driver's body movements (Wong et al., 2019). The system will identify abnormal movements and link them to driver fatigue or drowsiness.

The underlying method is the use of Convolutional Neural Networks (CNN) to detect driver fatigue based on changes in their facial expressions. This approach assumes that fatigue affects facial expressions, and these changes can be detected and used as predictors of driver fatigue. One study that investigated the effects of mental fatigue on attention used Event-Related Potentials (ERPs) and found that mental fatigue can impair cognitive performance. This suggests that using CNN (Saputra, Hindarto, & Santoso, 2023) to detect changes in facial expressions may be a promising approach for detecting driver fatigue. Another study explored the relationship between fatigue and safety, focusing on the transportation industry. The study found that fatigue is a significant crash risk factor, and that fatigue management programs are effective in reducing crash risk. The proposed CNN-based driver fatigue detection system would involve using cameras to capture facial expressions of drivers while they are driving. The images will be processed by the CNN (Muhammad Hammad Saleem, Johan Potgieter, 2016), which will be trained to detect changes in facial expressions associated with fatigue.

In addition to object detection using a Convolutional Neural Network (CNN), there are several other deep learning methods that can be used for image processing and computer vision tasks. Some of them are: Recurrent Neural Networks (RNNs), (Song et al., 2023), (Ajitha et al., 2022) are a type of neural network architecture that allow information to flow forward and backward through the network, so that they can be used to process sequential data such as text or time. RNNs can be used to recognize handwriting, predict the next word in a sentence, and so on. Generative Adversarial Networks (GAN) (Martín et al., 2023) is a type of neural network architecture which consists of two models, namely the generator and the discriminator, which function to generate realistic new data. GANs (Wedha et al., 2022) can be used to create synthetic images, improve image quality, and so on. Autoencoders are a type of neural network architecture that are used to re-represent data, usually with the aim of improving data quality or reducing data dimensionality. Autoencoders can be used to perform data compression, image denoising, and so on. Long Short-Term Memory (LSTM) is a type of RNN designed (Aseeri, 2023), (Ajitha et al., 2022) to solve the problem of long-distance dependencies, that is, dependencies between data over longer distances in a data sequence. LSTM can be used to recognize human activity from sensor data, translate natural language, and so on. Deep Belief Networks (DBNs) are a type of neural network architecture consisting of multiple layers of Restricted Boltzmann Machines (RBMs) that are trained in stages. DBNs can be used to classify data and perform data dimension reduction. The combination of several deep learning methods can be used to solve various tasks in the field of image processing and computer vision, such as object detection, facial recognition, image classification, and so on.

The explanation regarding the introduction raises Research Questions (RQ) as follows:
How is the performance of the driver fatigue or drowsiness detection system using the Convolutional Neural Network (CNN) on the image from the camera in the car in real-time situations? (RQ 1).
Can the CNN model accurately discriminate between drowsy and non-drowsy drivers based on visual facial features and head movements? (RQ 2).

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12386 p-ISSN : 2541-044X

LITERATURE REVIEW

Research on the topic of image detection using Convolutional Neural Networks has been carried out by many previous researchers. Review of Star Hotels Using Convolutional Neural Network (Sze et al., 2022), this research uses Convolutional Neural Network to detect five-star hotels to 1-star hotels, using hotel images. The Convolutional Neural Network method does not only process images, but combines it with the text classification method, so that it can be used as an image-based hotel review. Comparison of Convolutional Neural Network and Artificial Neural Network for Rice Detection (Suherman et al., 2023) . This research detects disease in rice using the Convolutional Neural Network, the results are quite good, reaching 90%. Comparison of Accuracy in Detecting Tomato Leaf Disease with Google-Net VS EfficientNetB3 (Saputra, Hindarto, Rahman, et al., 2023). This research detects tomato diseases by comparing the Google-Net algorithm and the EfficientNetB3 algorithm. From some of the descriptions of previous research, it turns out that there are still quite a lot of trending topics for detection with the Convolutional Neural Network algorithm, considering that the Convolutional Neural Network algorithm approach in detection and classification is simple and very powerful. This research aims to detect sleepy images using the Convolutional Neural Network algorithm.

METHOD

Convolutional Neural Network is a type of Deep Learning algorithm used to process image and video data. CNN works by processing input data through a series of layers, which consist of several convolution and pooling layers, which are then followed by more complex layers, such as the fully connected layer and the output layer.

Here are some of the basic components in the CNN method:

Convolutional layer: The convolutional layer is used to extract visual features from the input data by multiplying the filter kernel to a defined input area. The filter kernel slides across the input area, thus extracting certain patterns from the input data. Each convolution layer consists of several different filter kernels to extract more complex features. The convolutional layer is the first layer in the CNN architecture which is useful for extracting visual features from input images or videos. The convolution layer works by multiplying the filter kernel to a defined image input area and producing a smaller feature matrix that represents the visual features of the image. Kernel filter is a small matrix that is used to extract certain patterns from the input image. The filter kernel moves horizontally and vertically across the input image and multiplies the pixels in that area to produce values that represent certain patterns. By using several different filter kernels on a single convolution layer, the model can extract many kinds of visual features from the input image, such as edges, lines, corners, and more complex shapes. After the filter kernel is applied to the input image, its output values are calculated by performing a convolution operation between the filter kernel and the pixels in the input image. These values are then processed by an activation function, such as ReLU (Rectified Linear Unit), to remove negative values and add nonlinearity to the output. This allows the model to learn more complex patterns. One of the main advantages of the convolution layer is its ability to extract visual features in a local and translation-invariant manner. That is, the features found in one part of the image can be found in another, different part of the image without the need to do special training at each position in the image. After the convolution layer has been executed, its output will be used as input to the next pooling layer, which aims to reduce the dimensions of the output from the convolution layer by taking the maximum value (max pooling) or average value (average pooling) from the defined input area.

Layer pooling: Layer pooling is used to reduce the output dimension of the convolution layer by taking the maximum value (max pooling) or average value (average pooling) from the defined input area. These speeds up the computing process and helps prevent overfitting. The pooling layer is the second layer in the CNN architecture which is used to reduce the output dimension of the convolution layer. The goal is to extract the most important information from the visual features found in the previous layer and maintain a translation-invariant output. There are two types of pooling layers that are commonly used in the CNN architecture: max pooling and average pooling. In max pooling, the maximum value of the defined input area will be taken as the output. Whereas in average pooling, the average value of the defined input area will be taken as the output. Basically, max pooling and average pooling perform down sampling operations on the output of the previous layer. By reducing the output

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12386 p-ISSN : 2541-044X

dimension of the previous layer, the model can process information more efficiently and reduce the risk of overfitting the training data. For example, suppose the output of the previous convolution layer has dimensions of 10 x 10 x 64. If we use max pooling with a kernel filter size of 2 x 2 and stride 2, then the output dimensions of the pooling layer will be 5 x 5 x 64. This process will repeat at each layer of convolution and pooling on the CNN architecture (Ismail et al., 2023). Another advantage of using the pooling layer is its ability to maintain a translation-invariant output. That is, the visual features found in certain parts of the image can be found in other parts of the image without the need for special training at each position in the image. This allows the model to learn visual features that are more generic and can be applied to various positions in the image.

Activation function: After each convolution and pooling layer, an activation function such as ReLU (Rectified Linear Unit) is used to remove negative values and add nonlinearity to the output. This allows the model to learn more complex patterns. Activation function is a mathematical function used to add non-linearity to the output of the convolution layer and fully connected layer in the CNN architecture. This activation function is applied to each neuron in that layer to determine whether the neuron will be active or not. Without the activation function, the output of each layer would consist only of a linear combination of its inputs, which would eventually produce an output equal to its inputs. By adding an activation function for each neuron, the model can learn more complex patterns and improve generalizability of training and testing data.

Several types of activation functions that are often used in the CNN architecture include:

1. ReLU (Rectified Linear Unit): This activation function is the most widely used function in the convolution layer and fully connected layer in the CNN architecture. The function is simple, that is, it produces an output of 0 if the input is negative, and it produces the same output as the input if the input is positive.
2. Sigmoid: The sigmoid activation function produces an output that is in the range 0 to 1. This function is often used in the output layer of logistic regression models and in some cases binary classification.
3. Tanh (Hyperbolic Tangent): The tanh activation function produces an output that is in the range -1 to 1. This function is like the sigmoid but has a more symmetrical output and can produce negative values.
4. Softmax: The softmax activation function is often used in the output layer of multiclass classification models. This function produces output in the form of probabilities for each possible class, with the sum of the probabilities for all classes equal to 1.

Choosing the right type of activation function is very important in designing an accurate and efficient CNN architecture. Apart from these types of activation functions, there are also several other types of activation functions that can be used, such as Leaky ReLU, ELU, and Swish.

Fully connected layer: The fully connected layer consists of many neurons that are fully connected to the neurons in the previous layer. This layer is used to process features that have been extracted from the previous convolution and pooling layers. The fully connected layer is the last layer in the CNN architecture which is used to connect the output from the previous layer to the desired output, such as the predicted class in classification or the predicted value in regression. This layer is also often referred to as the output layer or dense layer. In the fully connected layer, every neuron in that layer is connected to every neuron in the previous layer. In the CNN architecture, the output of the previous layer is usually a high-dimensional vector. Therefore, at the fully connected layer, the vector will be converted into a shorter vector with dimensions that match the desired output. This conversion process is carried out through the matrix multiplication operation between the input vector and the weight matrix. Each neuron in the fully connected layer has a different weight and will determine its contribution to the final output. As in the convolution and pooling layers, the fully connected layer also uses an activation function to add non-linearity to the output. The activation functions commonly used at the fully connected layer are ReLU or softmax, depending on the type of task to be completed. The fully connected layer is an important component in the CNN architecture to enable the model to learn complex relationships between the visual features found in the previous layer and the desired output. However, too many neurons in the fully connected layer can cause overfitting of the training data, so choosing the right number of neurons must be done carefully.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12386 p-ISSN : 2541-044X

Output layer: The output layer consists of neurons that generate predictions based on input from the previous fully connected layer. In binary classification tasks, the output layer usually uses the sigmoid activation function, whereas in multiclass classification, the output layer usually uses the softmax activation function. The output layer is the last layer in a neural network, which produces output from the model based on the input provided. The output layer is very important in tasks such as classification, regression, and object detection, which is why the output layer in the CNN architecture is also very important. In a binary classification task, the output layer usually consists of one neuron with a sigmoid activation function, which produces a probability value between 0 and 1 for the positive class. Whereas in a multiclass classification task, the output layer usually consists of several neurons, each representing a probability for each possible class, with the softmax activation function. In a regression task, the output layer usually consists of a single neuron without an activation function, which produces a continuous predictive value. The output value in regression can be in the form of integer numbers, real numbers, or in the form of high-dimensional vectors, depending on the type of task at hand. The output layer on the CNN architecture can also be used to determine the location and size of objects in the image. In an object detection task, the output layer usually consists of several neurons each representing a probability for each possible object class, as well as coordinates for determining the location and size of that object in the image. The output layer is the last layer in the CNN architecture, which generates the final prediction based on the given input. The CNN training process is carried out by minimizing the error between the output produced by the model and the appropriate label. Therefore, careful selection of the activation function and the number of neurons in the output layer is necessary to ensure accurate model performance for the task at hand.

In CNN model training, the input data is provided along with the appropriate labels. The model output is then compared with the original label to calculate the error or loss function, which is used to adjust the model parameters through the backpropagation algorithm. With continuous iteration, the CNN model can gradually optimize parameters and improve accuracy in image or video classification tasks. CNN has proven effective in various computer vision applications, such as image classification, object detection, image segmentation, and facial recognition.

OpenCV (Open-Source Computer Vision Library) is a library of open-source software used to process and analyze images and videos. OpenCV can be used for a variety of applications, such as digital image processing, computer vision and pattern recognition. This library was developed by Intel in 1999 and since then it has become a very popular open-source project. OpenCV has many features, such as face detection, motion detection, object recognition, text recognition and many more. This library is also equipped with various image processing algorithms, such as mathematical operations, edge detection, image segmentation, noise removal, and many more. OpenCV can be used in a variety of programming languages, including C++, Python and Java. OpenCV can also be integrated with various platforms and hardware, such as Raspberry Pi, iOS, and Android. Due to its open-source nature, many developers and researchers use OpenCV to develop innovative and effective computer vision and image processing solutions. OpenCV is also widely used in industry, including in industrial security, surveillance, and control.

OpenCV can be used to detect objects using an object detection technique called the Haar Cascade Classifier. The Haar Cascade Classifier is a machine learning algorithm that can be trained to detect certain objects in images by identifying special features in these objects. The object detection process with the Haar Cascade Classifier involves several stages, namely:

1. Model training, object detection model is trained using a dataset of desired object images. At the training stage, the Haar Cascade Classifier will study the special features of the object and create a model to detect the object in the image.
2. Feature detection, trained model will be used to detect special features in the image being analyzed. These features can be lines, edges or dots in the image which can help in detecting the desired object.
3. Object detection, After the special features are detected, the Haar Cascade Classifier will use the trained model to recognize the desired object based on these features. If an object is detected, it will be identified by a rectangular box indicating the object's location in the image.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023
DOI : https://doi.org/10.33395/sinkron.v8i2.12386
e-ISSN : 2541-2019
p-ISSN : 2541-044X

OpenCV also has a built-in function to detect objects with the Haar Cascade Classifier. This function allows the user to use a trained model to detect certain objects in the image. To use this function, the user must enter the trained model and the image to be analyzed as input, and the function will return the location of the detected object in the image.

RESULT

This part of the result describes the experience used in detecting sleepiness using the Convolutional Neural Network. Starting from the dataset downloaded from Kaggle, it consists of four folders, namely Closed_eyes having 726 pictures, Open_eyes having 726 pictures, Yawn having 725 pictures, no_yawn having 723 pictures.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 143, 143, 256)     7168
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 71, 71, 256)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 69, 69, 128)       295040
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 34, 34, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        73792
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 32)        18464
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 1568)              0
_________________________________________________________________
dropout (Dropout)            (None, 1568)              0
_________________________________________________________________
dense (Dense)                (None, 64)                100416
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 260
=================================================================
Total params: 495,140
Trainable params: 495,140
Non-trainable params: 0
```

Figure 1. Model Sequential for Detection Drowsy

Figure 1, before conducting training on the model, the parameters of the Convolutional Neural Network (CNN) are set model on Keras using the Sequential API. This model is a Convolutional Neural Network (CNN) which consists of several layers, namely:

1. First Layer: Conv2D with 256 filters, kernel size (3,3), ReLU activation function, and input_shape (145,145,3), which indicates that the input image has a size of 145x145 pixels with 3 color channels (RGB).
2. Second Layer: MaxPooling2D with pool size (2,2), which serves to downsampling the feature map resulting from the previous layer.
3. Third Layer: Conv2D with 128 filters, kernel size (3,3), and ReLU activation function.
4. Fourth Layer: MaxPooling2D with pool size (2,2).
5. Fifth Layer: Conv2D with 64 filters, kernel size (3,3), and ReLU activation function.
6. Sixth Layer: MaxPooling2D with pool size (2,2).
7. Seventh Layer: Conv2D with 32 filters, kernel size (3,3), and ReLU activation function.
8. Eighth Layer: MaxPooling2D with pool size (2,2).

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12386 p-ISSN : 2541-044X

9. Ninth Layer: Flatten, which functions to convert 3D feature maps into 1D vectors.
10. Tenth Layer: Dropout with a rate of 0.5, which serves to reduce overfitting of the model.
11. Eleventh Layer: Dense with 64 units and ReLU activation function.
12. Last Layer: Dense with 4 units and softmax activation function, which produces classification output with 4 classes.

This model is designed to process an input image measuring 145x145 pixels with 3 color channels and produces a classification output with 4 classes. In the training process, this model will update the weights and biases for each layer so that it can produce accurate output.

```
history = model.fit (train_generator, epochs=50, validation_data=test_generator, shuffle=True, validation_steps = len(test_generator))

Result
Epoch 1/50
43/43 [==============================] - 422s 10s/step - loss: 1.1787 - accuracy: 0.4595 - val_loss: 0.8622 - val_accuracy: 0.6747
Epoch 2/50
43/43 [==============================] - 412s 10s/step - loss: 0.6066 - accuracy: 0.7506 - val_loss: 0.6524 - val_accuracy: 0.7180
Epoch 3/50
43/43 [==============================] - 408s 9s/step - loss: 0.4214 - accuracy: 0.8411 - val_loss: 0.3382 - val_accuracy: 0.8668
.......
.......
.......
.......
Epoch 49/50
43/43 [==============================] - 239s 6s/step - loss: 0.0934 - accuracy: 0.9710 - val_loss: 0.0949 - val_accuracy: 0.9706
Epoch 50/50
43/43 [==============================] - 239s 6s/step - loss: 0.1165 - accuracy: 0.9562 - val_loss: 0.0817 - val_accuracy: 0.9619
```

Figure 2. Training process

Figure 2. Convolutional Neural Network (CNN) training algorithm or Convolutional Neural Network (ANN) is a machine learning process that involves optimizing weight and bias parameters in order to produce a CNN model that can classify objects in images. In the training process, the CNN model is supplied with training data in batch form. Each batch consists of several images that are processed by CNN, which then generates predictions. Next, the predicted value is compared with the actual label on the training data, and then the loss function value is calculated which represents the model prediction error. In CNN, the optimization process is carried out using backpropagation and gradient descent algorithms. This algorithm calculates the gradient of the loss function against the model parameters, namely weights and bias, and then optimizes these values to produce a better model. During the training process, the value of the loss function is calculated for each batch of training and validation data to measure model performance on data that has not been seen before. The CNN training algorithm uses techniques such as dropout and L2 regularization to prevent overfitting and increase model generalization. In the end, the CNN model generated by this training algorithm can be used to classify images on data that has never been seen before. The results of the training reached accuracy: 0.9562 and loss: 0.1165.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12386 p-ISSN : 2541-044X

DISCUSSIONS

How is the performance of the driver fatigue or drowsiness detection system using the Convolutional Neural Network (CNN) on the image from the camera in the car in real-time situations? (RQ 1)

The performance of a driver fatigue or drowsiness detection system using the Convolutional Neural Network (CNN) on images from in-car cameras in real-time situations may vary depending on several factors, such as the accuracy of the CNN model used, the quality of the detected image, and the environment around the car . But in general, driver fatigue or drowsiness detection systems using CNN have shown quite good results in several studies. Convolutional Neural Network (CNN) on images from in-car cameras in real-time situations is one method for detecting fatigue or drowsiness in car drivers using deep learning algorithms. This method works by utilizing the image from the camera in the car and processing it through a neural network that has been previously trained to recognize patterns that indicate fatigue or drowsiness in drivers, such as slow head movements or frequent eye blinks. In general, this method requires several steps, such as image acquisition from the camera, image pre-processing, feature extraction using CNN, classification, and output to the alarm system. When a fatigue or drowsiness detection system operates in a real-time situation, all steps must be carried out in a very fast time to be able to provide the fastest possible response. In this method, CNN is used to extract features from the camera image and classify whether the driver is sleepy or not. CNN is a type of neural network that consists of many layers and is specifically designed to process image or image data by taking advantage of their ability to recognize patterns and features in images. When CNN is trained using a sufficiently large and varied dataset, it can recognize specific patterns and features associated with fatigue or sleepiness in drivers and produce accurate output in a very short time. The output results can then be used to provide early warning to the driver through an alarm system or other notifications to avoid possible accidents caused by fatigue or drowsiness while driving. Several studies have produced CNN models that can recognize signs of fatigue or drowsiness in drivers with high accuracy, such as measuring the movement of the driver's head or eyes. However, several studies have also found several obstacles in implementing driver fatigue or drowsiness detection systems using CNN in real-time situations, such as limited processing speed and limited computational resources. In addition, the accuracy of identifying signs of fatigue or drowsiness in drivers can also be influenced by several other factors, such as mental fatigue and stress. Thus, a driver fatigue or drowsiness detection system using CNN on images from in-car cameras in real-time situations has great potential to improve driving safety, but several factors need to be considered that can affect the system's performance.

Detection of fatigue or drowsiness in car drivers using an image-based detection system from car cameras is very dependent on ambient light. This is because the detection system uses camera images as input and the information that can be retrieved from these images is strongly influenced by the ambient lighting conditions. In good lighting conditions, such as during the day, enough light enters the camera so that the resulting image is bright and clear. This allows the detection system to more accurately detect slow head movements or frequent eye blinks that indicate fatigue or drowsiness in a driver. However, in poor lighting conditions, such as at night, very little light enters the camera, so the resulting image tends to be dark and difficult for detection systems to analyze. This can cause errors in detecting head movements or eye blinks associated with fatigue or drowsiness in drivers. Therefore, detecting fatigue or drowsiness in drivers at night becomes more difficult and inaccurate when compared to good lighting conditions such as during the day. To overcome this problem, there are several solutions that can be done. One way is to improve the quality of lighting around the camera, for example by adding additional lights in the car or changing the camera angle to get better light. In addition, the use of camera technology that is more sensitive to low light can also help improve detection accuracy in poor lighting conditions. In addition, the use of additional sensors that can assist in detecting fatigue or drowsiness in drivers such as heart rate sensors, body temperature sensors, or body movement sensors can be a more effective alternative solution in poor lighting conditions. These sensors can provide additional information that can help identify fatigue or drowsiness in drivers more accurately and are independent of ambient lighting conditions.

Can the CNN model accurately discriminate between drowsy and non-drowsy drivers based on visual facial features and head movements? (RQ 2)

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Number 2, April 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i2.12386 p-ISSN : 2541-044X

Convolutional Neural Network (CNN) models can be used to accurately distinguish between drowsy and non-drowsy drivers based on visual facial features and head movements. CNN is a type of machine learning algorithm used to process and study image or video data. To discriminate between sleepy and not sleepy drivers, CNNs can be trained with image or video datasets of sleepy and not sleepy drivers, which include visual facial features and head movements. CNN can then learn important patterns and features from the image or video to recognize signs of a drowsy driver. However, to improve model accuracy, it is necessary to collect large and representative datasets, and to undergo careful image processing and analysis. In addition, a good model also requires the right parameters and architecture, as well as effective and efficient training techniques.

Therefore, developing a CNN model that is accurate in discriminating between drowsy and non-drowsy drivers requires in-depth research and development as well as collaboration between experts in the fields of machine learning, computer vision and road safety.

To develop a CNN model that is accurate in discriminating between drowsy and non-drowsy drivers, several stages and careful decision making are required. Here are some common stages in developing a CNN model:

1. Dataset collection and preparation: The dataset should include images or videos of sleepy and non-drowsy drivers, with visual facial features and head movements. The dataset must be large and representative, with a balance between positive and negative classes. The dataset also needs to be processed and normalized to improve image quality and reduce noise.
2. CNN model creation: The CNN model must be built taking into account the appropriate parameters and architecture for the given classification task. This includes the number of CNN layers, filter size, number of neurons, activation functions, pooling layers, and so on.
3. CNN model training: CNN models must be trained using prepared datasets, with effective and efficient training techniques, such as stochastic gradient descent, backpropagation, dropout, and others. Models should be monitored regularly to avoid overfitting or underfitting.
4. Evaluation of the CNN model: The CNN model must be evaluated using validation and test datasets, using appropriate evaluation metrics such as accuracy, precision, recall, F1 score, ROC curve, and others.
5. CNN model improvement: Based on the evaluation results, the model should be improved by adjusting the parameters and architecture, improving the dataset, or introducing new techniques in training.
6. CNN model implementation: Once the CNN model is considered sufficiently accurate and stable, the model can be implemented on an automotive system, such as a car or truck, to identify drowsy drivers and provide appropriate preventive measures.

In developing an accurate CNN model, it is also necessary to pay attention to several things such as:

1. Availability and quality of datasets that include a variety of signs of drowsy drivers, such as fatigue, decreased concentration, sudden head movements, and others.
2. Selection of the appropriate model architecture, considering the complexity of the classification task, the size of the dataset, and the available computational resources.
3. Use of effective training techniques, such as data augmentation, early stopping, and batch normalization to avoid overfitting and increase convergence speed.
4. Careful evaluation of the model, avoiding bias and differences in distribution between the training data and the validation data.
5. Application of a safe and responsible model, considering factors such as privacy, security, ethics and applicable regulations.

The development of a Convolutional Neural Network model that is accurate and reliable in discriminating between drowsy and non-drowsy drivers could provide major benefits for road safety, by reducing.

*name of corresponding author

CONCLUSION

In conclusion, driver fatigue is a serious problem affecting road safety, especially at night. The use of the Convolutional Neural Network algorithm to detect changes in facial expressions associated with fatigue is a promising approach to address this problem. By training a Convolutional Neural Network algorithm to detect fatigue in real-time, it is possible to warn drivers when they are fatigued, reducing the risk of accidents and saving lives. To train the Convolutional Neural Network algorithm, a data set of images of people with varying levels of fatigue will be used. The data sets will be labeled according to the degree of fatigue indicated in each image, allowing CNN to learn to detect different levels of fatigue. The Convolutional Neural Network algorithm will then be tested using images of the driver taken while driving at night, to see if it can accurately detect driver fatigue.

REFERENCES

Ajitha, A., Goel, M., Assudani, M., Radhika, S., & Goel, S. (2022). Design and development of Residential Sector Load Prediction model during COVID-19 Pandemic using LSTM based RNN. *Electric Power Systems Research*, *212*(October 2021), 108635. https://doi.org/10.1016/j.epsr.2022.108635

Aseeri, A. O. (2023). Effective RNN-Based Forecasting Methodology Design for Improving Short-Term Power Load Forecasts: Application to Large-Scale Power-Grid Time Series. *Journal of Computational Science*, *68*(February), 101984. https://doi.org/10.1016/j.jocs.2023.101984

Chen, Z., & James, J. (2022). Put your FTSE down: Wealth shocks and road traffic collisions. *Social Science and Medicine*, *314*(April), 115488. https://doi.org/10.1016/j.socscimed.2022.115488

Cheng, S., Ping, Q., Wang, J., & Chen, Y. (2022). EasyGaze: Hybrid eye tracking approach for handheld mobile devices. *Virtual Reality and Intelligent Hardware*, *4*(2), 173–188. https://doi.org/10.1016/j.vrih.2021.10.003

Ismail, W. N., Alsalamah, H. A., Hassan, M. M., & Mohamed, E. (2023). AUTO-HAR: An adaptive human activity recognition framework using an automated CNN architecture design. *Heliyon*, *9*(2), e13636. https://doi.org/10.1016/j.heliyon.2023.e13636

Martín, A., Hernández, A., Alazab, M., Jung, J., & Camacho, D. (2023). Evolving Generative Adversarial Networks to improve image steganography. *Expert Systems with Applications*, *222*(February), 119841. https://doi.org/10.1016/j.eswa.2023.119841

Muhammad Hammad Saleem, Johan Potgieter, K. M. A. (2016). Plant Disease Detection and Classification by Deep Learning. *Nature*, *29*(7553), 1–73. http://deeplearning.net/

Mwale, M., Mwangilwa, K., Kakoma, E., & Iaych, K. (2023). Estimation of the completeness of road traffic mortality data in Zambia using a three source capture recapture method. *Accident Analysis and Prevention*, *186*(July 2022), 107048. https://doi.org/10.1016/j.aap.2023.107048

Saputra, A. D., Hindarto, D., Rahman, B., & Santoso, H. (2023). *Comparison of Accuracy in Detecting Tomato Leaf Disease with GoogleNet VS EfficientNetB3*. *8*(2), 647–656.

Saputra, A. D., Hindarto, D., & Santoso, H. (2023). Disease Classification on Rice Leaves using DenseNet121, DenseNet169, DenseNet201. *Sinkron*, *8*(1), 48–55. https://doi.org/10.33395/sinkron.v8i1.11906

Song, S., Chen, J., Ma, L., Zhang, L., He, S., Du, G., & Wang, J. (2023). Research on a working face gas concentration prediction model based on LASSO-RNN time series data. *Heliyon*, *9*(4). https://doi.org/10.1016/j.heliyon.2023.e14864

Suherman, E., Hindarto, D., Makmur, A., & Santoso, H. (2023). Comparison of Convolutional Neural Network and Artificial Neural Network for Rice Detection. *Sinkron*, *8*(1), 247–255. https://doi.org/10.33395/sinkron.v8i1.11944

Sze, E., Santoso, H., & Hindarto, D. (2022). *Review Star Hotels Using Convolutional Neural Network*. *7*(1), 2469–2477.

Wedha, B. Y., Karjadi, D. A., Wedha, A. E. P. B., & Santoso, H. (2022). Style Transfer Generator for Dataset Testing Classification. *SinkrOn*, *7*(2), 448–454. https://doi.org/10.33395/sinkron.v7i2.11375

*name of corresponding author

Wong, S. T., Belant, J. L., Sollmann, R., Mohamed, A., Niedballa, J., Mathai, J., Street, G. M., & Wilting, A. (2019). Influence of body mass, sociality, and movement behavior on improved detection probabilities when using a second camera trap. *Global Ecology and Conservation*, *20*, e00791. https://doi.org/10.1016/j.gecco.2019.e00791

Zheng, T., Glock, C. H., & Grosse, E. H. (2022). Opportunities for using eye tracking technology in manufacturing and logistics: Systematic literature review and research agenda. *Computers and Industrial Engineering*, *171*(February), 108444. https://doi.org/10.1016/j.cie.2022.108444

*name of corresponding author