IJIRMET ISSN (Online): 2456-0448
International Journal of Innovative Research in Management, Engineering, and Technology
Vol. 8, Issue 6, June 2023

SIGN LANGUAGE PREDICITION USING CNN

[1] Mr.E.Dilipkumar,[2] Mr.S.Raja
[1] Department of MCA, Dhanalakshmi Srinivasan college of engineering and technology
[1] savitiris.mca@dscet.ac.in,[2] rajas.@yahoo.com.

INTRODUCTION

Helmet detection is a computer vision task that involves detecting whether a person is wearing a helmet or not. This task is typically performed using machine learning algorithms that are trained on a large dataset of images that contain people with and without helmets.
The purpose of helmet detection is to improve safety measures in various industries such as construction, manufacturing, and sports. Wearing helmets is essential for preventing head injuries, and automating the process of helmet detection can help reduce the risk of accidents and injuries.
There are several approaches that can be used for helmet detection, including deep learning algorithms such as convolutional neural networks (CNNs) and object detection algorithms such as You Only Look Once (YOLO). These algorithms can be trained on a dataset of labeled images to learn how to detect helmets in new images.

Overall, helmet detection is an important task in many industries, and it has the potential to improve safety and prevent accidents and injuries. The use of machine learning algorithms can automate the process of helmet detection and help ensure that safety measures are followed.

OBJECTIVES OF THE PROJECT

The objectives of a project focused on helmet detection can vary depending on the specific application and context. However, some common objectives of such a project may include:
Improving Safety: The primary objective of a helmet detection project is to improve safety in industries such as construction, manufacturing, and sports. By automating helmet detection, it is possible to ensure that safety measures are being followed, and reduce the risk of accidents and injuries.
Real-time Detection: Another objective of a helmet detection project may be to achieve real-time detection of helmets. This can be important in applications such as sports analysis or video surveillance, where timely detection is critical.
High Accuracy: The accuracy of the helmet detection algorithm is another important objective of the project. A high accuracy algorithm will ensure that helmets are being detected correctly, and false alarms are minimized.
Scalability: A helmet detection project may also aim to create a scalable solution that can be deployed in various settings and industries. This can involve optimizing the algorithm's performance for different hardware platforms and improving its efficiency to handle large volumes of data.
Overall, the objectives of a helmet detection project are to improve safety, achieve real-time detection, ensure high accuracy, create a scalable solution, and develop a user-friendly interface.

PROJECT DESCRIPTION

In this Project Work, a Non-Helmet Rider detection system is built which attempts to satisfy the automation of detecting the traffic violation of not wearing helmet and extracting the vehicles' license plate number. The main principle involved is Object Detection using Deep Learning at three levels. The objects detected are person, motorcycle at first level using YOLOv3, helmet at second level using YOLOv3, License plate at the last level the number plate detect and sent the email

SYSTEM ANALYSIS

INTRODUCTION

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization. Once the software requirement have been analyzed and specified the software design involving three technical activities-design, coding, implementation and testing that are required to build and verify the software.

SYSTEM STUDY

System study in helmet detection involves understanding the technical aspects of detecting whether a person is wearing a helmet or

not. This can be achieved through the use of various algorithms, and computer vision techniques.
The system should be designed in a way that it can accurately detect whether a person is wearing a helmet or not, even in different lighting conditions and angles. The following steps can be taken to perform a system study in helmet detection:

Identify the requirements: Determine the specific requirements for the system, such as the types of helmets to be detected, the lighting conditions, the camera angles, and the accuracy needed.
Select the appropriate sensors: Choose the appropriate sensors, such as cameras, thermal imaging sensors, or depth sensors, based on the identified requirements.
Develop the algorithm: Develop an algorithm that can detect whether a helmet is present or not based on the data collected from the sensors.
Train the system: Train the system using a large dataset of images with and without helmets to improve its accuracy.

Test the system: Test the system in different lighting conditions and angles to determine its accuracy and reliability.
Optimize the system: Optimize the system based on the testing results to improve its accuracy and performance.
Implement the system: Implement the system in real-world applications, such as in traffic monitoring systems or sports safety systems.
Overall, a system study in helmet detection involves understanding the technical aspects of the system, selecting the appropriate sensors, developing an algorithm, training and testing the system, optimizing it, and implementing it in real-world applications.

EXISTING SYSTEM

In this existing we are taking algorithm as support vector machine it will take hyper plane to divide data into test and train like input data and database. It will give moderate output using region of interest and moving object but we cant detect exact are of the face to detect the helmet.

- Moving object.
- Region of interest
- Support vector machine.

DISADVANTAGES

- It has limit data set
- Only single motor cycle identified

PROPOSED SYSTEM

A proposed system for helmet detection can be developed using computer vision techniques and deep learning algorithms. The system would involve the following steps:
Data Collection: The first step in developing a helmet detection system would be to collect a dataset of images that contains people wearing helmets and people not wearing helmets. The dataset should be diverse and include images captured under different lighting conditions, camera angles, and backgrounds.
Image Pre-processing: The collected dataset would then be pre-processed to remove any noise or unwanted artifacts from the images. This could involve techniques such as resizing, normalization, and histogram equalization.
Object Detection: The next step would be to use a deep learning algorithm such as YOLO (You Only Look Once) or Faster R-CNN (Region-based Convolutional Neural Network) to detect the presence of helmets in the images. These algorithms work by identifying regions of interest in the image and then classifying them as either containing a helmet or not.
Post-processing: Once the helmets have been detected, the system can perform post-processing to refine the results and eliminate any false positives. This could involve techniques such as non-maximum suppression and thresholding.
Integration: The final step would be to integrate the helmet detection system into an application or platform that can be used in real-world scenarios. This could involve developing a mobile application that uses the device's camera to capture images and detect helmets in real-time, or integrating the system into a surveillance camera system to monitor a specific area for helmet usage.
Overall, a helmet detection system based on computer vision and deep learning algorithms could help to promote safety and prevent accidents in a variety of settings, from construction sites to sports events is shown in figure 2.4

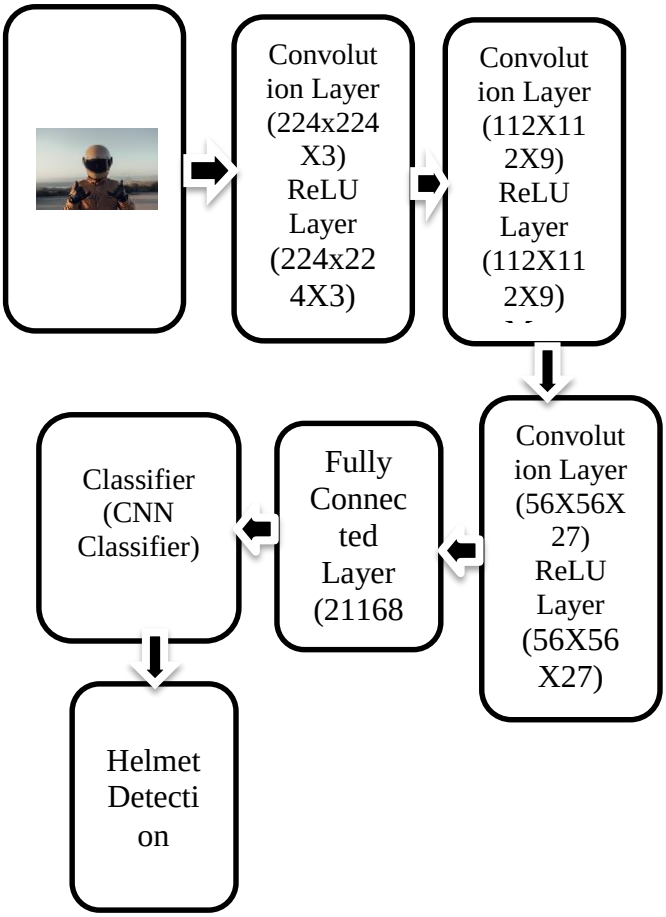


Figure 2.4 Proposed System

MODULE DESCRIPTION

The project includes six processing modules, they are:

- Data Collection
- Data Annotation
- Data Preprocessing
- Training
- Evaluation
- Inference

DATA COLLECTION

Data collection is a crucial step in developing a helmet detection system, as the accuracy of the system will depend on the quality and quantity of the data used to train the model. Here are some considerations for data collection in helmet detection

DATA ANNOTATION

Data annotation is the process of adding labels to the data to make it useful for machine learning algorithms. In helmet detection, data annotation involves identifying the regions of interest (i.e., the helmets) in the images and labeling them accordingly. Here are some

considerations for data annotation in helmet detection

DATA PREPROCESSING

Data preprocessing is an important step in developing a helmet detection system using YOLO (You Only Look Once), which is a popular deep learning algorithm used for object detection. Here are some considerations for data preprocessing in helmet detection using YOLO

TRAINING

Training a helmet detection system using YOLOv3 (You Only Look Once version 3) involves the following steps:

- Preparing the dataset
- Configuring the yolov3 architecture
- Preparing the pre-trained weights
- Training the modelevaluating the model
- fine-tuning the model
- Deploying the model

EVALUATION

Evaluating the performance of a helmet detection model trained with YOLOv3 involves several metrics that can be used to assess its accuracy. Here are some metrics that can be used for evaluating the performance of the model

INFERENCE

To create an interface for a helmet detection system based on YOLOv3, you can use a graphical user interface (GUI) toolkit such as Tkinter, PyQt, or wxPython. Here are some basic steps for creating a GUI interface for helmet detection

SYSTEM DESIGN AND DEVELOPMENT

INTRODUCTION

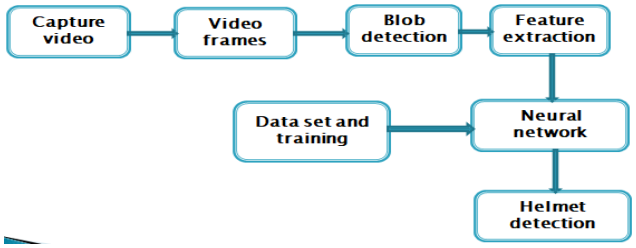
The system analysis follows system design with two major aspects namely, i)Logical design and ii) physical design. Following the design, the development of the system based on coding design using the chosen software technologies is carried out.

LOGICAL DESIGN

The logic design of the system is conceived and represented using some standard design elements such as algorithmic procedures, Block diagram, sequence diagram, Acitivity diagram etc.

BLOCK DIAGRAM

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks.



Block Diagram

Block Diagram is figure 3.2.1 describe the over work of the project. First to input the video the video is convert to frames the next step to using yolov3 algorithm using Blob detection. The object is detect and next step using CNN algorithm to preprocessing the image and final the helmet found or not

PHYSICAL DESIGN

The physical design is the actual design of database tables, form design, design of input and output forms and finally the code design.

DATASET

This dataset was obtained from online.(http://www.kaggle.com). The dataset images in shown in figure 3.4

ISSN (Online): 2456-0448
International Journal of Innovative Research in Management, Engineering, and Technology
Vol. 8, Issue 6, June 2023

S. NO.	IMAGE NUMBER	TYPE	SIZE
1	BikesHelmets0	PNG File	406 KB
2	BikesHelmets1	PNG File	664 KB
3	BikesHelmets2	PNG File	725KB
4	BikesHelmets3	PNG File	517 KB
5	BikesHelmets4	PNG File	593 KB
6	BikesHelmets5	PNG File	584KB

Table 3.4 Data Set Image Format Table

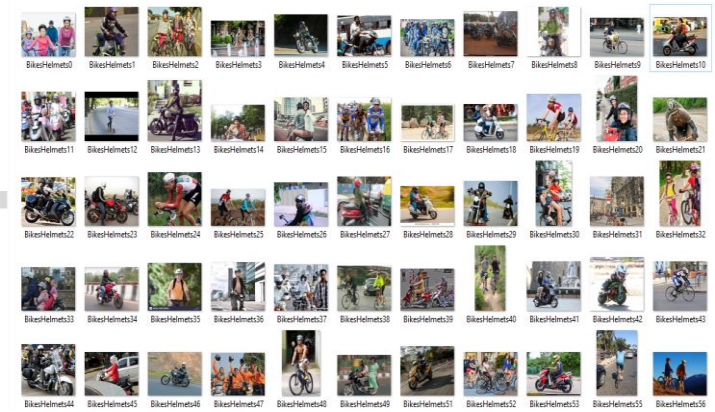


Figure 3.4 Dataset images

MODULES

The project includes six processing modules, they are:

- Data Collection
- Data Annotation
- Data Preprocessing
- Training
- Evaluation

- Inference

DATA COLLECTION

Data collection is a crucial step in developing a helmet detection system, as the accuracy of the system will depend on the quality and quantity of the data used to train the model. Here are some considerations for data collection in helmet detection:

IMAGE SOURCE: The first step in data collection is to determine the source of the images. Images can be captured using surveillance cameras, smart phones. The input video is shown in Figure 3.5.1 In Appendices

DIVERSITY: It is important to collect a diverse range of images that include different lighting conditions, camera angles, and backgrounds. This will help to ensure that the model is robust and can accurately detect helmets in a variety of scenarios.

ANNOTATION: Once the images have been collected, they need to be annotated to identify the regions of interest (i.e., the helmets) in the images. This can be done manually by humans, or using automated tools such as bounding box annotation software.

QUANTITY: The quantity of data collected is also important, as a larger dataset can help to improve the accuracy of the model. It is recommended to have at least a few thousand images in he dataset.

BALANCE: It is important to balance the dataset between positive and negative examples. In other words, there should be an equal number of images with and without helmets to ensure that the model does not become biased towards one class.

PRIVACY: It is important to respect privacy laws when collecting data, especially if the images are captured in public areas. It is recommended to obtain consent from individuals who are captured in the images, or to blur their faces to protect their privacy.

Overall, data collection is an important step in developing a helmet detection system, and care should be taken to ensure that the dataset is diverse, balanced, and annotated accurately.

DATA ANNOTATION

Data annotation is the process of adding labels to the data to make it useful for machine learning algorithms. In helmet detection, data annotation involves identifying the regions of interest (i.e., the helmets) in the images and labeling them accordingly. Here are some considerations for data annotation in helmet detection:

BOUNDING BOXES: The most common form of annotation in object detection is bounding box annotation. This involves drawing a box around the region of interest (i.e., the helmet) and labeling it as such. Bounding box annotation is a relatively simple and straightforward method, and it can be done manually by humans or using automated tools.

ACCURACY: It is important to ensure that the annotations are accurate and consistent across the dataset. Inaccurate annotations can lead to a decrease in the accuracy of the model and make it difficult to train the algorithm effectively.

CLASS IMBALANCE: As mentioned earlier, it is important to balance the dataset between positive (helmets) and negative (no helmets) examples. This helps to prevent the model from becoming biased towards one class and improves its ability to accurately detect helmets in a variety of scenarios.

OCCLUSION: In some cases, helmets may be partially occluded by other objects, such as clothing or accessories. In these cases, it is important to label only the visible parts of the helmet to avoid confusion during training.

ANNOTATION TOOLS: There are many annotation tools available, ranging from simple tools such as Microsoft Paint to more advanced tools such as Labelbox or CVAT (Computer Vision Annotation Tool). The choice of annotation tool will depend on the specific requirements of the project.

Overall, data annotation is a critical step in developing a helmet detection system. Careful consideration should be given to the accuracy and consistency of the annotations, as well as the balance between positive and negative examples in the dataset.

DATA PREPROCESSING

Data preprocessing is an important step in developing a helmet detection system using YOLO (You Only Look Once), which is a popular deep learning algorithm used for object detection. Here are some considerations for data preprocessing in helmet detection using **YOLO:**

Image Resizing: YOLO works best with images that are a fixed size. Therefore, it is important to resize all the images in the dataset to the same dimensions. The recommended size for YOLO is 416x416 pixels.

Image Normalization: It is recommended to normalize the pixel values in the images to a common range, such as [0,1] or [-1,1]. This can help to improve the stability of the model during training.

Image Augmentation: Data augmentation can help to increase the size and diversity of the dataset, which can help to improve the accuracy of the model. Common data augmentation techniques include random cropping, flipping, rotation, and color jittering.

Label Encoding: In YOLO, the labels for each object in the image consist of the object's class and its bounding box coordinates. Therefore, it is important to encode the labels in a format that can be used by YOLO, such as the YOLOv3 format.

Dataset Splitting: It is recommended to split the dataset into training, validation, and testing sets. The training set is used to train the model, the validation set is used to monitor the model's performance during training, and the testing set is used to evaluate the final performance of the model.

Data Normalization: It is important to normalize the dataset to ensure that the distribution of the data does not affect the performance

of the model. This can be done by subtracting the mean and dividing by the standard deviation of the dataset.
Overall, data preprocessing is an important step in developing a helmet detection system using YOLO. The goal of data preprocessing is to ensure that the data is in a format that can be used by the YOLO algorithm and to improve the accuracy and stability of the model during training.

TRAINING

Training a helmet detection system using YOLOv3 (You Only Look Once version 3) involves the following steps:

PREPARING THE DATASET: The first step in training a YOLOv3 model for helmet detection is to prepare the dataset. This involves collecting a dataset of images containing people wearing and not wearing helmets, and annotating the images with bounding boxes around the helmets. The dataset should be divided into training, validation, and test sets.

CONFIGURING THE YOLOV3 ARCHITECTURE: YOLOv3 is a neural network architecture that consists of multiple layers. To configure the architecture, you will need to modify the configuration file for the YOLOv3 model. The configuration file contains information such as the number of classes, the dimensions of the input images, and the hyper parameters for the training process is shown in figure in 3.5.4 Yolov3 network architecture in Appendices

PREPARING THE PRE-TRAINED WEIGHTS: YOLOv3 can be trained from scratch, but it is more common to use pre-trained weights as a starting point. Pre-trained weights are learned weights from a model that has already been trained on a large dataset. You can download pre-trained weights from the official YOLO website or other sources.

TRAINING THE MODEL: To train the YOLOv3 model, you will need to run the training script with the configuration file, pre-trained weights, and dataset. The training process involves optimizing the weights of the model to minimize the loss function. During training, the model will be evaluated on the validation set, and the weights with the best validation loss will be saved.

EVALUATING THE MODEL: Once the model has been trained, you can evaluate its performance on the test set. This involves running the model on the test images and calculating metrics such as precision, recall, and F1 score.

FINE-TUNING THE MODEL: If the performance of the model is not satisfactory, you can fine-tune the model by adjusting the hyper parameters or by training it for more epochs.

DEPLOYING THE MODEL: Finally, once you have a trained and evaluated model, you can deploy it to a production environment to detect helmets in real-time.

Overall, training a YOLOv3 model for helmet detection requires careful preparation of the dataset, configuration of the YOLOv3 architecture, and training and evaluation of the model. With proper training and tuning, a YOLOv3 model can achieve high accuracy in helmet detection.

EVALUATION

Evaluating the performance of a helmet detection model trained with YOLOv3 involves several metrics that can be used to assess its accuracy. Here are some metrics that can be used for evaluating the performance of the model:

PRECISION: Precision measures the fraction of true positive detections among all the detections. It is defined as the ratio of the number of correctly detected helmets to the total number of detected helmets. A high precision score indicates that the model is able to accurately detect helmets and avoid false positives.

RECALL: Recall measures the fraction of true positive detections among all the actual helmets in the images. It is defined as the ratio of the number of correctly detected helmets to the total number of actual helmets. A high recall score indicates that the model is able to detect most of the helmets in the images.

F1-SCORE: F1-score is the harmonic mean of precision and recall. It is a measure of the model's overall accuracy and takes into account both false positives and false negatives.

MEAN AVERAGE PRECISION (mAP): mAP is a widely used metric for evaluating object detection models, including helmet detection models. It measures the average precision across all classes in the dataset. A high mAP score indicates that the model is able to accurately detect helmets in a variety of images.

Intersection over Union (IoU): IoU measures the overlap between the predicted bounding boxes and the ground truth bounding boxes. It is defined as the area of the intersection between the two boxes divided by the area of their union. A high IoU score indicates that the predicted bounding box is close to the ground truth bounding box.

These metrics can be computed by running the trained model on a test dataset and comparing the predicted bounding boxes with the ground truth bounding boxes. It is important to use a diverse set of test images to ensure that the model is able to generalize to new and unseen data. By evaluating the model using these metrics, you can assess its accuracy and identify areas for improvement.

INFERENCE

To create an interface for a helmet detection system based on YOLOv3, you can use a graphical user interface (GUI) toolkit such as Tkinter, PyQt, or wxPython. Here are some basic steps for creating a GUI interface for helmet detection:

Design the user interface: The first step is to design the user interface for the application. You can use a GUI designer tool to create

ISSN (Online): 2456-0448
International Journal of Innovative Research in Management, Engineering, and Technology
Vol. 8, Issue 6, June 2023

the layout of the interface, including buttons, labels, and input fields.
Add a file chooser: The application should allow the user to choose the input image or video file for detecting helmets. You can add a file chooser dialog box to the interface to enable the user to select the file.
Process the input file: Once the user has selected the input file, the application should load the file and pass it to the helmet detection

model for processing. You can add a progress bar to the interface to show the progress of the detection process.
Display the output: After the model has processed the input file, the application should display the output to the user. You can display the output in a new window or in the same window as the input file. The output should include the original image or video with bounding boxes around the detected helmets.
Add configuration options: You can add configuration options to the interface to allow the user to customize the detection process. For example, the user may want to set the confidence threshold or select a specific YOLOv3 configuration file.
Add logging and error handling: Finally, you should add logging and error handling to the application to help diagnose any problems that may occur during the detection process. You can add log messages to the interface or write them to a log file.
By creating a GUI interface for a helmet detection system, you can make it easier for users to use the system and customize the detection process. The interface can also provide feedback to the user about the progress of the detection and any errors that occur.

FINAL OUTPUT

The Bikers will not wear helmet to detect that person and sent the person and bike image to outlook mail is shown in figure 3.6.6.

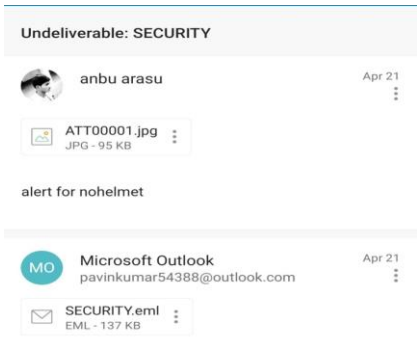


Figure 3.6.6 outlook output

TESTING AND IMPLEMENTATION

4.1 SOFTWARE TESTING

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team. System testing is important because of the following reasons: System testing is the first step in the Software Development Life Cycle, where the application is tested as a whole. The application is tested thoroughly to verify that it meets the functional and technical specifications. The application is tested in an environment that is very close to the production environment where the application will be deployed. System testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

UNIT TESTING

Unit testing is a software testing technique that involves testing individual units or components of a software system in isolation to ensure they are functioning correctly. For helmet detection, unit testing can involve testing individual functions or modules of the software that are responsible for detecting helmets. Here are some steps you can follow for unit testing in helmet detection:
Identify the functions or modules: Identify the specific functions or modules of the software that are responsible for detecting helmets. For example, this may include modules for image processing, object detection, or neural network inference.
Define test cases: Define a set of test cases that cover the expected inputs and outputs for each function or module. Test cases should include both valid and invalid inputs to ensure that the software handles all scenarios correctly.
Write unit tests: Write unit tests for each function or module that test its functionality in isolation. Use testing frameworks such as Pytest or Unittest to automate the testing process and to assert the expected outputs for each test case.
Run tests: Run the unit tests to ensure that each function or module is functioning correctly and that all test cases pass. Debug and fix

any issues that arise during testing.
Refine and iterate: Refine the unit tests and test cases as needed to cover additional scenarios or edge cases. Iterate and rerun the tests to ensure that the software is functioning correctly in all scenarios.

Unit testing in helmet detection involves testing individual functions or modules of the software to ensure that they are functioning correctly. By following the steps outlined above, you can identify the specific functions or modules to test, define test cases, write unit tests, run the tests, and refine and iterate as needed to ensure that the software is functioning correctly.

SYSTEM IMPLEMENTATION

The title of this project is Helmet Detection for Motor Vehicle Department using Deep Learning. I am using yolov3 and CNN Algorithms in my project.

CONVOLUTION NEURAL NETWORK

A convolutional neural network (CNN) is a type of neural network that is specifically designed for image processing tasks such as image classification, object detection, and image segmentation. The name "convolutional" refers to the mathematical operation of convolution that is used in the network.

In a CNN, the input image is passed through a series of convolutional layers, which extract features from the image by applying a set of filters or kernels. Each filter slides over the input image, performing a convolution operation to produce a feature map. These feature maps are then passed through activation functions such as ReLU to introduce non-linearity into the network.

The output of the convolutional layers is then passed through pooling layers, which down sample the feature maps by taking the maximum or average value of a given window size. This reduces the dimensionality of the feature maps and makes the network more efficient.

Finally, the output of the pooling layers is flattened and passed through one or more fully connected layers, which perform the final classification or regression task based on the extracted features.

One of the key advantages of CNNs is their ability to automatically learn and extract features from the input image, without the need for manual feature engineering. This makes them particularly well-suited to image processing tasks, where the features of interest may be complex and difficult to define explicitly.

CNNs have been used in a wide range of applications, including image classification, object detection, facial recognition, and medical image analysis. They have also been used in combination with other types of neural networks, such as recurrent neural networks, to perform tasks such as video classification and natural language processing.

CNN is inspired by the biological phenomenon of the animal visual cortex, which shows the connectivity pattern between different neurons. CNN has a wide range of applications such as image processing, video processing, speech processing, and natural language processing. CNN consists of four significant steps, such as convolution layer, rectified linear unit (ReLU), maximum pooling layer, and fully connected layer. The architecture of the CNN single layer is shown in Fig.4.2.1

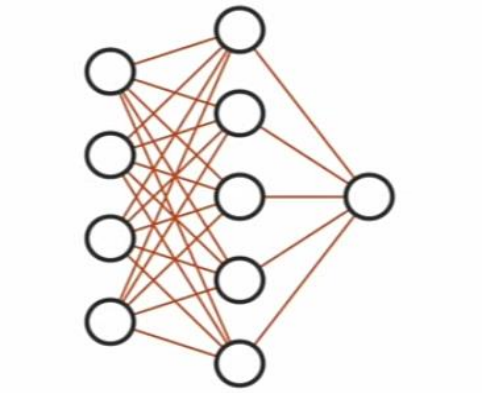


Figure 4.2.1The architecture of the CNN Layer

ReLU LAYER

In the convolution layer, an image is multiplied by filter kernel, which may have some negative values. These negative values bring the non-linearity in the image. The non-linearity is then removed by using the rectified linear unit layer by converting all the negative values to zero usingwhere, IReLUis the ReLU layer image and Iconvis the convolutional layer image. The ReLU layer output size is 224X224X3, which is equal to the size of the convolution layer output. The output of the ReLU layer is given to max pooling layer

.

MAX POOLING LAYER

Pooling is used for the minimization of the computing cost by reducing the dimension of the ReLU layer output. Pooling is also used to retain the position and rotational invariant features of an image. There are two types of pooling methods: Maximum and Average pooling. In maximum pooling, the maximum value of the given window is selected, while in average pooling, the average value of the window is selected . Unlike average pooling, maximum pooling suppresses the noise in the image along with feature reduction. Pooling is also used to maintain the localization of the shape of the local object in the image. The window size of 2×2 is selected for maximum pooling, which reduced the size to exactly half of the ReLU layer (56×56×27). The larger window size for the maximum pooling may result in the fine local information of the image is shown in figure 4.2.3.

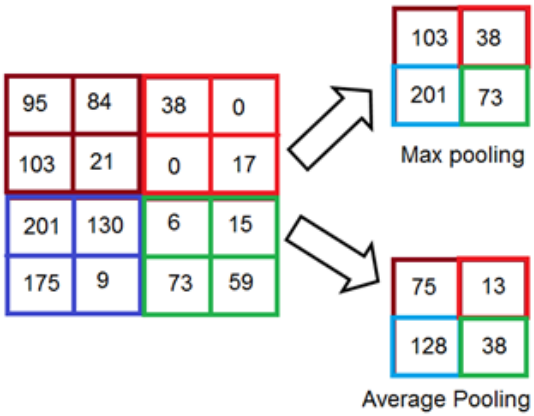


Figure 4.2.3 Max Pooling Layer

HELMET DETECTION USING CNN

In helmet detection, CNNs are used to detect helmets in images and videos. The CNN architecture used in helmet detection typically consists of convolutional layers, pooling layers, and fully connected layers.

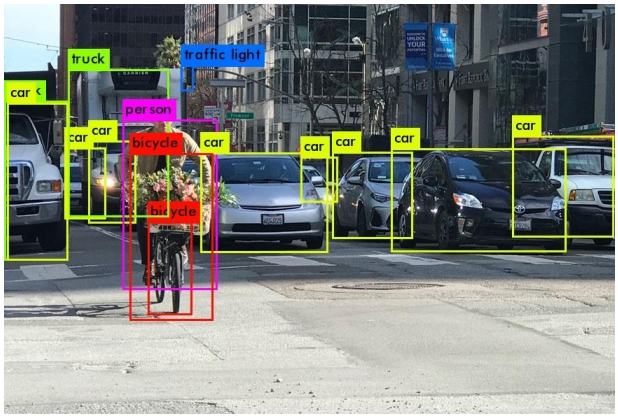
The convolutional layers extract features from the input image using convolution operations, where a set of filters or kernels slide over the image to produce a feature map. The pooling layers down sample the feature maps by taking the maximum or average value of a given window size. This reduces the dimensionality of the feature maps and makes the network more efficient.

The fully connected layers perform the final classification or regression task based on the extracted features. In the case of helmet detection, the final task is to classify whether a helmet is present in the image or not.

To train the CNN for helmet detection, a dataset of images and corresponding labels is required. The dataset consists of images of people with and without helmets, along with corresponding labels indicating whether a helmet is present or not. This dataset is used to train the CNN using supervised learning techniques, such as back propagation and gradient descent.

Once trained, the CNN can be used to detect helmets in new images and videos by passing the input through the network and analyzing the output. The output of the network typically consists of bounding boxes and class labels for each detected object in the input image or video frame.

Overall, the use of CNNs in helmet detection allows for accurate and efficient detection of helmets in images and videos, making it a useful tool for applications such as safety monitoring and surveillance.

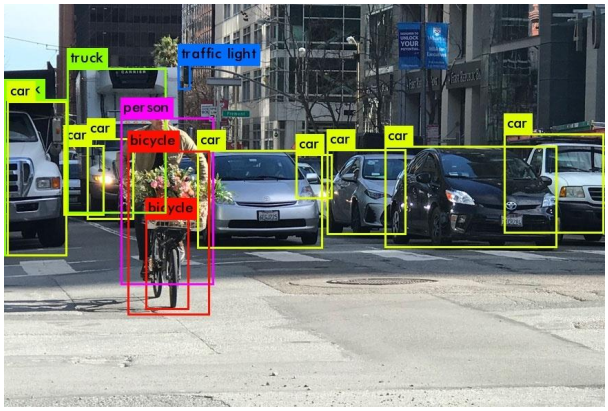


4.2.5 YOLOV3 ALOGORITHM

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. The YOLO machine learning algorithm uses features learned by a deep convolutional neural network to detect an object.

Versions 1-3 of YOLO were created by Joseph Redmon and Ali Farhadi, and the third version of the YOLO machine learning

ISSN (Online): 2456-0448
International Journal of Innovative Research in Management, Engineering, and Technology
Vol. 8, Issue 6, June 2023



algorithm is a more accurate version of the original ML algorithm. The first version of YOLO was created in 2016, and version 3, which is discussed extensively in this article,

was made two years later in 2018. YOLOv3 is an improved version of YOLO and YOLOv2.

YOLO is a Convolutional Neural Network (CNN) for performing object detection in real-time. CNNs are classifier-based systems that can process input images as structured arrays of data and recognize patterns between them (view image below). YOLO has the advantage of being much faster than other networks and still maintains accuracy. It allows the model to look at the whole image at test time, so its predictions are informed by the global context in the image. YOLO and other convolutional neural network algorithms "score" regions based on their similarities to predefined classes. High-scoring regions are noted as positive detections of whatever class they most closely identify with. For example, in a live feed of traffic, YOLO can be used to detect different kinds of vehicles depending on which regions of the video score

highly in comparison to predefined classes of vehicles is shown in figure 4.2.4

YOLOV3 WORK

YOLOv3 (You Only Look Once version 3) is an object detection model that uses convolution neural networks to detect objects in images or video frames. It is an improvement over the earlier versions of YOLO, with higher accuracy and faster processing times.

The YOLOv3 model works by dividing an image into a grid and predicting bounding boxes and class probabilities for each cell in the grid. It uses a series of convolutional neural networks (CNNs) to extract features from the image and then predicts the objects in the image based on those features.

YOLOv3 uses convolutional neural network framework and includes a number of technical enhancements, including feature concatenation, multi-scale training, and a new classification loss function. It also includes some optimizations for GPU processing, which allows it to run much faster than other object detection models.

Overall, YOLOv3 is a highly effective object detection model, capable of detecting a wide range of objects with high accuracy and speed. It has become a popular choice for a variety of computer vision tasks, including surveillance, self-driving cars, and augmented reality applications. The Yolov3 working flow in shown in figure 4.2.5

ISSN (Online): 2456-0448
International Journal of Innovative Research in Management, Engineering, and Technology
Vol. 8, Issue 6, June 2023

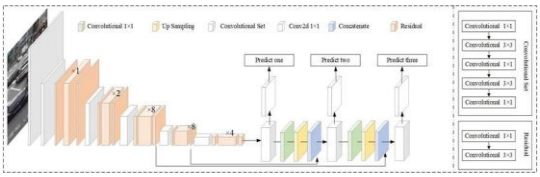


Figure 4.2.6 Yolov3 work Flow

DATA SET COLLECTION

YOLOv3, the dataset collection process involves gathering and labeling images that will be used to train the object detection model. Here are the basic steps involved in collecting a dataset for YOLOv3:

Determine the objects you want to detect: The first step in dataset collection is to determine the objects that you want to detect using YOLOv3. This will help you focus your efforts on finding and labeling images that contain these objects.

Gather images: Once you know what objects you want to detect, you need to start gathering images that contain those objects. You can use various sources such as online image repositories or capture images using a camera or Smartphone.

Label images: After gathering images, the next step is to label them. Labeling involves drawing bounding boxes around the objects you want to detect in each image. There are various tools available that can help with this task such as LabelImg, VoTT, or RectLabel.

Organize the dataset: Once all the images have been labeled, organize them into a directory structure that YOLOv3 can read. Each image should have a corresponding text file with the same name containing the object labels and their coordinates.

Train the model: After the dataset has been collected and organized, you can use it to train the YOLOv3 model. During training, the model will learn to detect the objects in the images based on the labeled data.

Test and evaluate the model: Once the model has been trained, it's important to test and evaluate it to ensure that it can accurately detect the objects in new images. This involves feeding the model new images and comparing its predictions to the ground truth labels.

HELMET-NONHELMET_CNN.H5

Helmet-nonhelmet_cnn.h5 is likely a file that contains a trained Convolutional Neural Network (CNN) model that can classify images as either containing a helmet or not.

CNNs are a type of deep learning model commonly used in image classification tasks. They work by analyzing the image data through multiple layers of filters and extracting features that are relevant to the classification task.

In this case, the helmet-nonhelmet_cnn.h5 model has likely been trained on a dataset of images of people wearing helmets and not wearing helmets, and it has learned to recognize patterns and features that distinguish between the two categories. The .h5 file extension is often used to indicate that the file contains a saved Keras model, a popular deep learning framework for Python.

To use this model, you could load it into a Python script or notebook and feed it new images to classify. The output would likely be a probability score for each class (helmet or non-helmet) based on the input image.

YOLOV3-CUSTOM.CFG

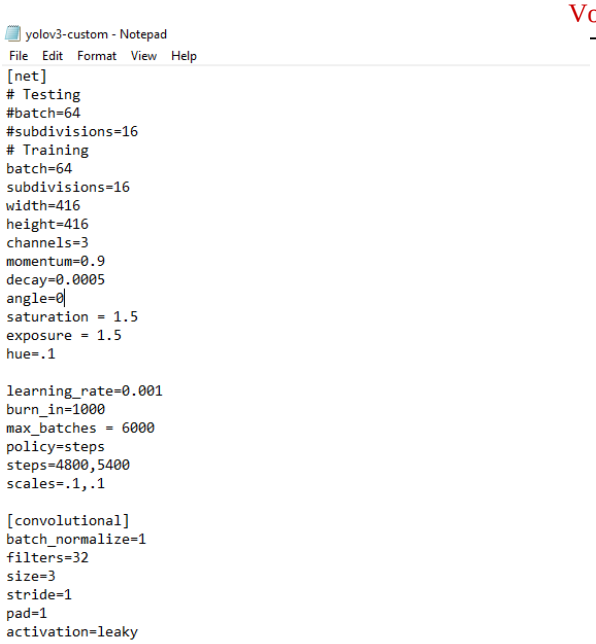
yolov3-custom.cfg is a configuration file for a custom-trained YOLOv3 object detection model.

YOLO (You Only Look Once) is a popular real-time object detection algorithm that can detect objects in images and videos. YOLOv3 is the third version of this algorithm and includes improvements such as a deeper network architecture, feature pyramid networks, and more accurate bounding box predictions.

The yolov3-custom.cfg file contains the architecture and settings for a YOLOv3 model that has been customized for a specific object detection task. This file includes various parameters such as the number of classes to detect, the input and output sizes of the network, the number and size of filters used in each layer, and other hyperparameters that affect the model's performance.

To use this configuration file, you would typically pair it with a custom dataset of labeled images and use it to train the YOLOv3 model. The model will learn to detect the objects specified in the yolov3-custom.cfg file and output bounding boxes around them in new images or videos. After training, the

ISSN (Online): 2456-0448
International Journal of Innovative Research in Management, Engineering, and Technology
Vol. 8, Issue 6, June 2023



configuration file can also be used to fine-tune or adjust the model's performance. The Yolov3 cfg file in shown on figure 4.2.9 yolov3.cfg file

yolov3.cfg file

RESULT& DISCUSSON

In this project the system is built to automatically detect the motor vehicle riders who are not wearing helmets and their vehicles with no number plates. The Proposed model detected through real time video. The yolov3 and CNN algorithm and some computer vision techniques help in attain good accuracy for detection of helmets and number plates. But, they needed to be punished only detection is not sufficient. So, the transport officers can take better actions against the riders. Our System detect the helmet and number plates from the real time video and give the result is interpreted the form of the images i.e, it marks the green square against the helmet and red square against the number plate is shown in figure 4.2.10 and figure 4.2.11 in Appendices

REPORT

In the proposed project five train videos are given has input and the report is generated as shown in table 4.3.1.

Table 4.3.1 Helmet and Non-Helmet Persons Details

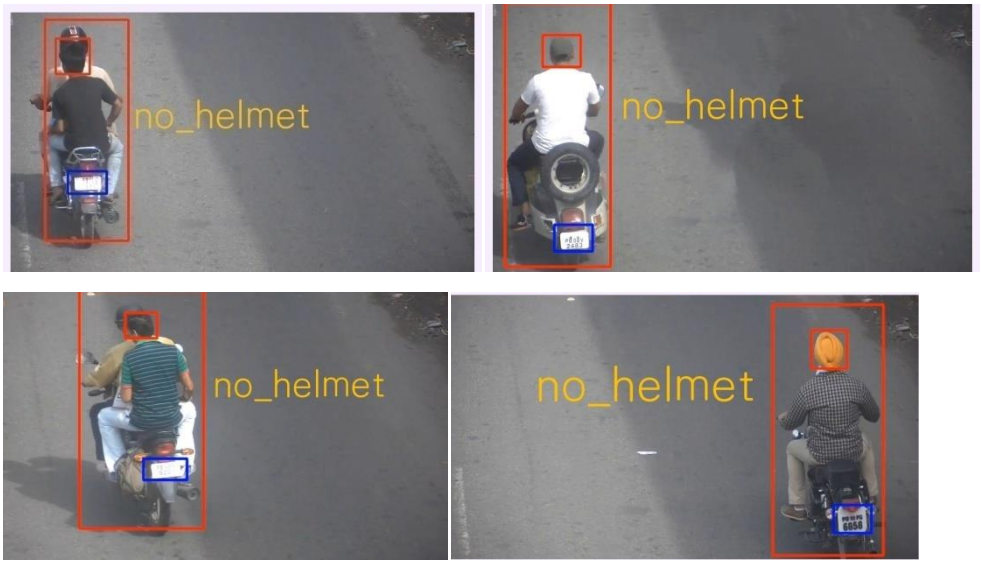
S.NO	Video Name	Helmet	Non-Helmet
1	video1	11	27
2	video2	12	19
3	video3	12	16
4	video4	14	23

5	video5	12	14

The table in reference for each video the total number of person vehicles wearing helmet or not wearing helmet. The sample images are shown in figure 4.3.2.
It helps the motor vehicle department to easily identify the person in the further manner and the remedial measure

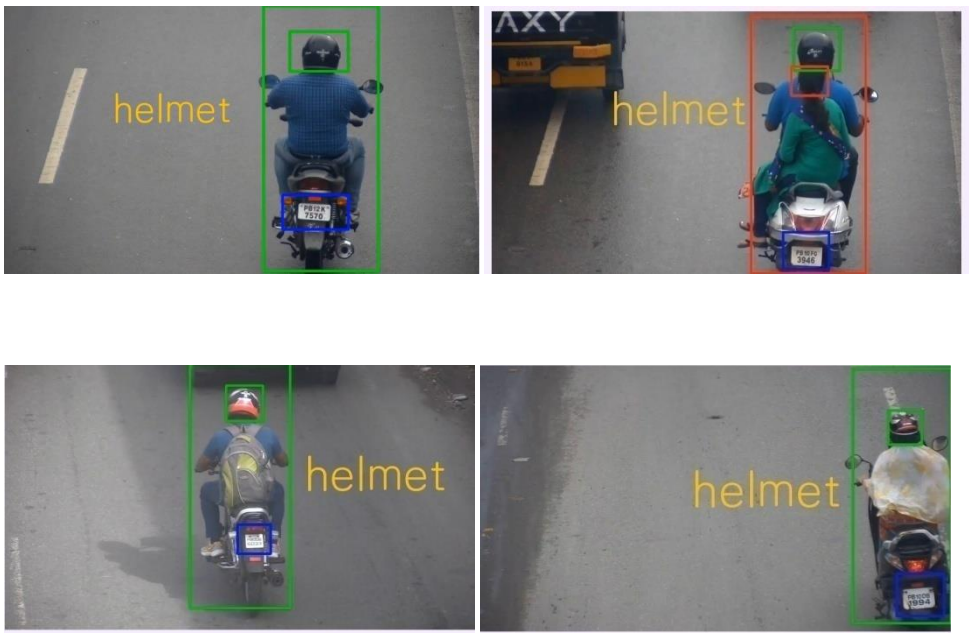
Video Name: video1.mp4
Video1.mp4 helmet wear Person Images

Video1.mp4 Non - Helmet Person Images



Video Name: video2.mp4

Video2.mp4 Helmet wear Person Images



Video2.mp4 Non - Helmet Person Images

CONCLUSION & FUTURE ENHANCEMENTS

CONCLUSION

This Project "Helmet Detection Using Deep Learning" is successfully analyzed, designed developed tested and implemented. The YOLO helmet wearing detection model is proposed in this Project. The modeling capacity of the network on the dependencies between distinct points in the image is strengthened by incorporating the attention mechanism into the YOLOv3 to substantially develop the model's feature demonstration ability and take out accurate features.. We created a dataset and ran several assessment tests on it to check the performance of these proposed strategies. The experimental outcomes suggest that the strategies presented in this research develop the YOLO network's performance, making it an outstanding key for the helmet-wearing recognition method in real-world circumstances.

FUTURE ENHANCEMENTS

There are several potential future enhancements for helmet detection using YOLOv3:

Improved accuracy: YOLOv3 is a powerful object detection algorithm, but it can still benefit from improvements in accuracy. One way to achieve this is through the use of more advanced neural network architectures or training techniques.

Real-time tracking: Currently, YOLOv3 can detect helmets in a single image, but it can be enhanced to track helmets in real-time video footage. This would be especially useful for applications such as monitoring construction sites or sporting events.

Integration with other sensors: Helmet detection technology can be enhanced by integrating it with other sensors, such as thermal cameras or lidar sensors. This would enable the system to detect helmets in low-light or poor visibility conditions.

Customization for specific applications: YOLOv3 can be customized to suit specific applications. For example, the algorithm can be trained to recognize different types of helmets or to detect helmets in specific locations.

Integration with AI-based decision-making systems: Helmet detection technology can be integrated with AI-based decision-making systems to automate safety responses based on the detection of a helmet or the absence of a helmet. This could include triggering an alarm or alerting safety personnel to a potential hazard.

Integration with cloud computing: YOLOv3 can be enhanced by integrating it with cloud computing platforms, which can enable real-time processing of large amounts of data. This would enable the system to detect helmets more quickly and accurately.

Object recognition in complex environments: YOLOv3 can be further enhanced to recognize helmets in complex environments, such as crowded areas or areas with complex backgrounds. This could involve incorporating advanced algorithms for object recognition and segmentation.

Overall, there is significant ptential for future enhancements of helmet detection technology using YOLOv3. These improvements can help make work environments and recreational areas safer for individuals and promote the prevention of accidents and injuries.

REFERENCES

[1]. L. Xie, T. Ahmad, L. Jin, Y. Liu, and S. Zhang, “A new CNN-based procedure for multi-directional car license plate detection,” IEEE
Trans. Intell. Transp.Syst., vol.19,no.2,pp.507–517,Feb.2018.
[2]. Dasgupta, M., Bandyopadhyay, O., & Chatterji, S. (2019). Automated Helmet Detection for Multiple Motorcycle Riders using CNN, 2019 IEEE Conference on Communication Technology and Information.
[3]. Image processing and segmentation techniques for vehicle's plate recognition. , 2020 IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS).
[4]. Shrivastava, S., Singh, S. K., Shrivastava, K., & Sharma, V. (2020). CNN based on Automated Vehicle Registration Number Plate Recognition System. 2020 2nd International Conference on progress in Computing, Communication Control and Networking (ICACCCN)
[5]. Luvizon, Diogo Carbonera; Nassu, Bogdan Tomoyuki; Minetto, Rodrigo (2016). This is a Video-Based System for Vehicle Speed Measurement in Urban Roadways. IEEE Transactions on Intelligent Transportation Systems.
Journal of Pharmaceutical Negative Results ¦ Volume 13 ¦ Special Issue 10 ¦ 2022 3740
[6]. SaquibNadeemHashmi, Kaushtubh Kumar, Siddhant Khandelwal, Dravit Lochan, Sangeeta Mittal,(2019) “Real Time License Plate Recognition from the Video Streams using Deep Learning”, International Journal of Information Retrieval Research, Volume 9 Issue 1 January-March 2019.
[7]. Hendry and Rung-Ching Chen, (2019) “Automatic License Plate Recognition via sliding-window darknetYOLO deep learning”, Image and Vision Computing.
[8]. Dariusz Laskowski,Piotr Lubkowski,(2017) “Assessment of Quality of Identification of Data in Systems of Automatic Licence Plate Recognition” In: Mikulski J. (eds) Smart Solutions in Today’s Transport. TST 2017.
[9] New Ni Kyawf, G R Sinhaf, Khin Lay Mon, (2018) “License Plate Recognition of Myanmar on Vehicle Number Plates A Critical Review," IEEE 7th Conference on Consumer Electronics.
[10] Suresh, Anukul Patil, Abhishek Kashyap, Ankit Jaiswal, Saksham Sharma(2018)