INTERNSHIP REPORT

On

INTELLIGENT DRIVING ASSISTANT SYSTEM(IDAS)

By

YERRAPOTHU PAVANI

Regd. No.: BU22CSEN0300214

PATHVISION INNOVATIONS PVT

LTD INTN3444

(Duration: 05-05-2024 to 04-07-2024)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Gandhi Institute of Technology and

Management (DEEMED TO BE A

UNIVERSITY) BENGALURU, KARNATAKA,

INDIA SESSION:2020-2024

Acceptance/Offer Letter from Industry

athvision

Date: 01 May 2025

To
Yerrapothu Pavani
Address:- Kammavaripalem(V),Bn Kandriga (M), Chittoor (D),
Andhra Pradesh, 517213
Email: pavaniyerrapothu6@gmail.com

Subject: INTERNSHIP OFFER LETTER

Dear **Yerrapothu Pavani,**

You have been **selected** to join **PathVision Innovations Private Limited** as an **Intern – Algorithm Development** for a 2-month technical internship. Your role is aligned with our active **Research and Development** projects.

This opportunity is designed not as a training program, but as an active engagement with real-world challenges that require your effort, consistency, and willingness to learn and contribute. You will be working with a team that values innovation, discipline, and mission-first thinking.

Internship Details

- **Start Date:** 05 May 2025
- **End Date:** 04 July 2025
- **Internship Duration:** 2 Months
- **Working Mode/Location:** Offline - GITAM Bengaluru Campus
- **Stipend:** Unpaid (learning-oriented engagement)

Your Journey with us – Overview & Structure

During your course of employment, you will be learning and delivering value real-time every day.

Up to 14 Days: Induction & Training:

You will undergo an immersive training to develop a problem-solving mindset that combines math, and technology with design thinking. You will be mentored to elevate your mindset, skillset, and you will be groomed for current industry challenges and real-world problems. **At the end of the program,** you will be equipped with an experimentative, learning-driven and interdisciplinary approach to problem-solving and you will be allocated in different real-world problem solving projects.

Expected Contributions

- Participate actively in your assigned domain and contribute to ongoing product or research development.
- Complete milestone-based tasks, submit progress reports, and attend evaluations.
- Work collaboratively with fellow interns or research/development leads.
- Maintain proper documentation and professional discipline.

Certificate Issued After Internship

Date: 04 July 2025

To
Yerrapothu Pavani
GITAM Regd No: BU22CSEN0300214
Department: CSE AI&DS
Email: pavaniyerrapothu6@gmail.com

INTERNSHIP COMPLETION CERTIFICATE

This is to certify that **Ms. Yerrapothu Pavani**, from **CSE (Artificial Intelligence & Machine Learning), GITAM Deemed to be University, Bengaluru Campus, has completed a 2-month technical Internship as Intern - Algorithm Development at PathVision Innovations Pvt. Ltd., from 05 May 2025 to 04 July 2025.** During this internship, she contributed to the design and development of algorithms used in driver assistance system. She worked with OpenCV, Python, and logic formulation strategies in cross-functional environments. Her work reflected qualities of discipline, learning agility, collaboration, technical competence, and aligned with startup-grade product development goals.

Performance Rating: Excellent
We appreciate her contribution and wish her continued success in upcoming professional journey.

Certificate ID: PV-GITAM-05/2025-AL-04
Internship Location: GITAM Deemed to be University, Bengaluru Campus

Yatheendra Nath Reddy Devasani
Founder & CEO
PathVision Innovations Pvt. Ltd.

PathVision Innovations Private Limited
Registered Address: 7/4686-10, Srinivasa Nagar, Proddatur, Andhra Pradesh, 516360
Email: pathvision.queries@gmail.com | **Phone:** +91 76720 76329
CIN: U46521AP2025PTC117317

GITAM

CERTIFICATE

This is to certify that the mini project report entitled **"Real-Time Seatbelt Usage Detection"** is a bonafide record of work carried out by **YERRAPOTHU PAVANI(BU22CSEN0300214)** submitted in partial fulfillment of requirement for the award of degree of **Bachelor of Technology in Computer Science and Engineering**.

Dr. Monica Ravishankar
Assistant Professor
Department of CSE,
GST, Bengaluru

CERTIFICATE

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose consistent guidance and encouragement crowned our efforts with success.

We consider it our privilege to express our gratitude to all those who guided us in the completion of the project.

We express our gratitude to Director Prof. **Basavaraj Gundappa Katageri** for having provided uswith the golden opportunity to undertake this project work in their esteemed organization.

We sincerely thank **Dr. Y. Vamshidhar,** HOD, Department of Computer Science and Engineering, Gandhi Institute of Technology and Management, Bengaluru for the immense support given to us.

We express our gratitude to our project guide **Dr. MONICA RAVISHANKAR, ASSISTANT PROFESSOR,** Department of Computer Science and Engineering, Gandhi Institute of Technology and Management, Bengaluru, for their support, guidance, and suggestions throughout the project work.

Student Name	Registration No.
YERRAPOTHU PAVANI	BU22CSEN0300214

Contents

Title	Page.No
Internship	I
Acceptance/Offer Letter from Industry	II
Certificate Issued After Internship	III
Certificate	IV
Acknowledgement	v
Contents	VI
Abstract	VII
INTRODUCTION	1
FOUNDATIONS IN ALGORITHM DEVELOPMENT	2-3
DATASET COLLECTION & PREPROCESSING	4-7
MODEL ARCHITECTURE & TRAINING PROCESS	8-11
TOOLS WORKED ON	12-15
RESULTS	16-18
CHALLENGES FACED	19-21
CONCLUSION	22

Abstract

This internship report presents the work undertaken during a two-month internship at PathVision Innovations Pvt. Ltd., where the primary focus was the development of a machine learning algorithm to detect seatbelt usage from static images. The project was part of a larger initiative toward enhancing vehicle safety through automated driver monitoring systems.

In the initial phase of the internship, significant time was dedicated to learning key technologies such as TensorFlow, MATLAB, and OpenCV, which laid the foundation for the practical implementation that followed. Subsequently, work progressed into designing and training a Convolutional Neural Network (CNN) model capable of identifying whether a person is wearing a seatbelt. The model was trained using a custom dataset containing labeled images of drivers with and without seatbelts.

The dataset underwent thorough preprocessing, including resizing, normalization, and augmentation techniques such as rotation, zoom, and horizontal flipping to improve generalization. The model architecture consisted of multiple convolutional layers with batch normalization and dropout regularization, followed by dense layers with a sigmoid output for binary classification. The training process employed techniques like early stopping and class balancing to ensure robustness.

The final model achieved a test accuracy of 96.55% and demonstrated strong performance in detecting seatbelt usage, with very few false negatives and no false positives. These results indicate the model's potential for integration into real-time vehicle surveillance systems. The internship experience offered valuable hands-on exposure to deep learning pipelines, data management, model optimization, and the broader scope of computer vision applications in road safety.

1. INTRODUCTION

In recent years, advancements in artificial intelligence and computer vision have played a significant role in enhancing road safety and vehicle automation. One of the key areas of innovation lies in the development of intelligent driver monitoring systems, which include seatbelt detection, drowsiness detection, and distraction monitoring. These systems aim to reduce accidents and promote compliance with traffic safety regulations.

This report presents the work carried out during a two-month internship at PathVision Innovations Pvt. Ltd., where the primary objective was to develop a machine learning- based solution for seatbelt usage detection. The task involved building a robust Convolutional Neural Network (CNN) capable of distinguishing between images where the driver is wearing a seatbelt and those where they are not. The system was trained to perform under varying lighting conditions, poses, and occlusions to ensure real-world usability.

The internship began with foundational learning in TensorFlow, MATLAB, and OpenCV, which provided the technical skills required for algorithm development. This was followed by a systematic pipeline involving dataset preparation, image preprocessing, data augmentation, model training, and evaluation. The dataset used included real-world images annotated with seatbelt usage labels and required handling of class imbalance and noisy inputs.

The development of such a model is not only a practical implementation of deep learning techniques but also contributes toward solving a real-world problem in the automotive domain. Ensuring that drivers wear seatbelts is a critical aspect of traffic law enforcement, and automating this detection process can assist in implementing safety measures more effectively.

Through this internship, both technical and analytical skills were enhanced. Exposure to end-to-end machine learning pipelines, along with real-world debugging and model

optimization challenges, helped bridge the gap between academic learning and practical application. The project demonstrated how modern AI tools can be used to build scalable safety solutions, contributing positively to the ongoing transformation in smart transportation systems.

2. Learning Phase: Foundations in Algorithm Development

2.1 Overview

The initial phase of the internship was dedicated to strengthening foundational knowledge in algorithm development, with a focus on tools and concepts that would later be applied in practical implementation. This phase primarily involved completing guided coursework and tutorials on TensorFlow, MATLAB, and general algorithm design strategies.

During the first two weeks, I invested time in building a strong understanding of:

- Basic Python-based data structures and algorithm patterns,
- Matrix operations and simulation modeling using MATLAB,
- Machine learning workflows using TensorFlow,
- The principles of neural networks and activation functions,
- Loss functions, optimizers, and backpropagation.

2.2 TensorFlow Learning

TensorFlow, an open-source framework developed by Google, was at the core of this learning phase. I followed introductory tutorials and hands-on notebooks to get comfortable with its syntax and functionalities.

- Tensor operations and broadcasting

- Model definition using Sequential() and Functional APIs
- Compilation and training using model.fit()
- Using callbacks like EarlyStopping and ModelCheckpoint
- Visualizing metrics with TensorBoard

This knowledge became crucial when designing, training, and evaluating the CNN for seatbelt detection later in the project.

2.3 MATLAB Introduction

MATLAB was used to simulate algorithms and understand image processing at a mathematical level. Topics like grayscale conversion, edge detection (Sobel and Canny), and noise reduction were explored.

Sample experiments included:

- Gaussian blurring and median filtering
- Segmentation techniques using thresholds
- Visualization of image histograms

Understanding these algorithms in MATLAB helped build intuition before applying them using OpenCV in Python.

2.4 Python for Algorithmic Thinking

Apart from library-specific learning, I also spent time on algorithm design and optimization, focusing on:

- Logic formulation and problem decomposition

- Writing modular and reusable code
- Time and space complexity analysis
- Unit testing and debugging strategies

3. Dataset Collection, Labeling & Preprocessing

3.1 Dataset Overview

The success of any machine learning project, especially in computer vision, depends largely on the quality and quantity of the dataset used for training and validation. For this project, the dataset consisted of images of car drivers—some wearing seatbelts and some not—captured under varied real-world conditions including different lighting, angles, and occlusions.

The dataset was stored and managed using Google Drive and accessed via Google Colab during training.

The CSV file contained two columns:

1. **Filename** – the name of the image file (e.g., img_0123.jpg)
2. **Label** – binary classification:
    - 1 → Seatbelt present (Driver is wearing a seatbelt)
    - 0 → Seatbelt absent (Driver is not wearing a seatbelt)

Each image had a corresponding label which served as the ground truth during supervised learning.

3.2 Data Cleaning & Handling Faulty Images

During the data loading phase, faulty or unreadable images were detected using OpenCV's image reading function cv2.imread(). If the function returned None, it meant the image was corrupted or could not be read due to incompatible format or missing files.

A simple condition was added in the data loading loop:

```
if img is None:
    continue
```

This ensured that such images were skipped automatically without breaking the pipeline.

3.3 Dataset Splitting

To train a model effectively and assess its performance fairly, the dataset was split into three subsets:

- **Training Set (80%)**
  Used to train the model. The largest portion of the dataset.
- **Validation Set (10%)**
  Used during training to fine-tune parameters and detect overfitting.
- **Test Set (10%)**
  Used only after training to evaluate the final performance of the model.

The splitting was done using train_test_split() from sklearn.model_selection with shuffle=True to ensure randomness.

3.4 Image Resizing

For consistent input dimensions across all images and to optimize computational efficiency, all images were resized to **128×128 pixels** using the following code:

resized_img = cv2.resize(img, (128, 128))

This resolution is small enough to enable fast training while retaining enough visual detail for the CNN to learn relevant features.

3.5 Normalization

After resizing, the pixel values were normalized to a range of [0, 1] by dividing by 255. This helped improve training speed and model convergence by ensuring all inputs had the same scale.

normalized_img = resized_img / 255.0

3.6 Data Augmentation

To increase the dataset size virtually and improve the model's generalization, augmentation techniques were applied. These were especially helpful due to limited training data.

Augmentation techniques included:

- **Random Rotation**: ±10 degrees
- **Random Zoom**: up to 10%
- **Horizontal Flip**: 50% probability

Implemented using TensorFlow's

ImageDataGenerator: train_datagen =

ImageDataGenerator(

rescale=1./255,

rotation_range=10,

zoom_range=0.1,

```
    horizontal_flip=True,
)
```

3.7 Label Distribution

An imbalance was noticed: more images had drivers wearing seatbelts compared to not wearing. To handle this:

- **Class weights** were computed and applied during training.
- This ensured the model did not become biased toward the majority class.

3.8 Dataset Exploration and Visualization

Before training the model, it was essential to understand the distribution and quality of the data. This was done by visually inspecting sample images and plotting class counts using Python libraries like Matplotlib and Seaborn.

Example code to plot class distribution:

```
import seaborn as sns
import pandas as pd

df = pd.read_csv('/content/drive/MyDrive/combined_seatbelt.csv')
sns.countplot(x='label', data=df)
```

This helped in identifying whether the dataset was significantly skewed toward one class and if additional preprocessing or class weighting was needed.

4. Model Architecture & Training Process

4.1 Overview

After dataset preparation and preprocessing, the next critical step was the development of a Convolutional Neural Network (CNN) model for binary classification — detecting whether a driver is wearing a seatbelt or not. CNNs are well-suited for this task due to their powerful feature extraction capabilities and performance in image classification problems.

The model was designed from scratch using **TensorFlow** and **Keras**, ensuring flexibility and control over each layer, parameter, and regularization method. The architecture was kept deep enough to capture complex spatial features, yet optimized for efficiency to train on standard hardware.

4.2 CNN Architecture Design

The final model comprised **four convolutional blocks** followed by a dense classification head. Here's the layer-wise breakdown:

Convolutional Blocks (Feature Extractors)

Each block followed a structure of:

- **Conv2D layer** with ReLU activation and 'same' padding
- **BatchNormalization** layer for faster convergence and stability
- **MaxPooling2D** to downsample spatial features

Block	Filter Size	Kernel Size	Pool Size
1	32	3x3	2x2
2	64	3x3	2x2

3	128	3x3	2x2
4	256	3x3	2x2

Each convolutional block increased depth while reducing spatial dimensions, allowing the network to learn increasingly abstract and complex features at deeper layers.

Dense Head (Classifier)

- **Flatten Layer** to convert feature maps into a 1D vector
- **Dense Layer** with 256 neurons and ReLU activation
- **Dropout (0.5)** to prevent overfitting by randomly disabling 50% of the neurons during training
- **Output Layer** with 1 neuron and **sigmoid activation** for binary classification

```
model.add(Dense(1, activation='sigmoid'))
```

4.3 Compilation Details

The model was compiled using:

- **Loss Function**: BinaryCrossentropy() — ideal for binary classification
- **Optimizer**: Adam() — adaptive learning rate with momentum for faster convergence
- **Evaluation Metric**: accuracy

```
model.compile( loss='binary_
   crossentropy',
   optimizer='adam',
   metrics=['accuracy']
)
```

4.4 Training Configuration

- **Epochs**: 20
- **Batch Size**: 16
- **Validation Split**: 10% from training data
- **EarlyStopping**: Stops training if validation loss does not improve for 5 consecutive epochs
- **ModelCheckpoint**: Automatically saves the best weights based on validation accuracy
- **Class Weights**: Used to address class imbalance (computed from training labels)

These callbacks were implemented using TensorFlow's Keras API to ensure optimal training:

```
from keras.callbacks import EarlyStopping, ModelCheckpoint
```

4.5 Training Execution

Training was conducted on **Google Colab** with GPU acceleration enabled. The model converged around the 13th epoch, showing a strong upward trend in training accuracy and decreasing loss. The **training accuracy exceeded 96%**, with a validation accuracy of around 94%, indicating good generalization with minimal overfitting.

Real-time plots of loss and accuracy were visualized using matplotlib to monitor training behavior and detect anomalies.

4.6 Summary of Parameters

Parameter	Value
Image Size	128x128
Batch Size	16
Epochs	20
Optimizer	Adam
Loss Function	Binary Crossentropy
Accuracy Achieved	96.55%
Regularization	Dropout +BatchNorm

4.6 Summary of Parameters

The final architecture and training configuration were the result of several experimental iterations. Minor adjustments in hyperparameters, such as batch size, filter count, and dropout rates, were tested to find the optimal setup that achieved high accuracy without overfitting.

The model maintained a good balance between complexity and performance. Unlike very deep networks that require huge datasets and computational power, this custom CNN achieved high accuracy with moderate depth and well-chosen architectural components.

5. TOOLS WORKED ON

In the course of this internship, several industry-standard tools, libraries, and platforms were used for data handling, model development, visualization, and documentation. Each tool contributed to a specific part of the machine learning pipeline, from data preprocessing to model evaluation.

5.1 Python

Python was the core programming language used throughout the internship. Its readability, extensive libraries, and strong support for scientific computing make it ideal for machine learning projects.

Key features leveraged:

- Efficient handling of arrays and matrices using NumPy
- Visualization of training metrics and sample predictions using Matplotlib and Seaborn
- File handling for loading CSVs and image directories

5.2 TensorFlow & Keras

TensorFlow, along with its high-level API Keras, was the primary framework used to build, train, and evaluate the deep learning model.

Key functionalities used:

- Model building via Sequential API
- Data generators for batch processing and augmentation
- Callbacks for EarlyStopping and saving best model weights

- GPU acceleration using Colab runtime
- Evaluation metrics and loss visualization

5.3 OpenCV

OpenCV (Open Source Computer Vision Library) was used for image handling and preprocessing tasks. It enabled efficient manipulation of pixel-level data and real-time image transformations.

OpenCV utilities used:

- Reading and resizing images (cv2.imread, cv2.resize)
- Normalizing images
- Grayscale conversion
- Visualizing images for debugging

purposes Example:

```
img = cv2.imread('image.jpg')
img_resized = cv2.resize(img, (128, 128))
img_normalized = img_resized / 255.0
```

5.4 Google Colab

Google Colab served as the primary development environment. It provided access to free GPUs and an easy-to-use interface to run Jupyter notebooks directly in the browser.

Benefits experienced:

- Free GPU for accelerated training
- Integration with Google Drive for data storage

- Easy collaboration and notebook sharing
- Visualization and logging capabilities using inline plots

5.5 Pandas & NumPy

- **Pandas**: Used for reading the label CSV, merging metadata, and analyzing dataset structure.
- **NumPy**: Essential for manipulating image arrays and performing matrix operations during normalization and batch formation.

5.6 Matplotlib & Seaborn

These libraries were used for visualization of:

- Accuracy and loss curves during training
- Sample predictions
- Confusion matrix
- Class distribution and performance comparison

Visualizations played a key role in understanding model behavior and debugging performance issues.

5.7 Sklearn (Scikit-learn)

Scikit-learn was used for:

- Splitting the dataset (train_test_split)
- Computing class weights

- Generating evaluation metrics (precision, recall, F1-score)
- Drawing the confusion matrix for test predictions

5.8 MATLAB

In the early learning phase of the internship, MATLAB was used to simulate basic image processing tasks. While not directly used in the final model, it helped gain mathematical understanding of filters, edges, and grayscale transformations.

Experiments included:

- Edge detection using Sobel and Canny
- Histogram visualization
- Gaussian noise filtering

Together, these tools formed a cohesive ecosystem that enabled the successful development of a high-performing seatbelt detection system. Exposure to such a comprehensive set of technologies helped build technical depth and real-world implementation experience.

6. RESULTS

The results section highlights the performance of the developed Convolutional Neural Network (CNN) model on the seatbelt detection task. After weeks of data preparation, model tuning, and training, the final model demonstrated excellent classification accuracy and robustness across diverse test cases.

6.1 Model Performance Overview

The model was evaluated on the **test dataset (10%)**, which was never seen during training or validation. Key performance metrics were calculated to assess its generalization capabilities.

These results indicate that the model was able to accurately identify both classes (seatbelt and no seatbelt), with minimal misclassifications.

6.2 Confusion Matrix Analysis

A confusion matrix was generated to analyze true positives, true negatives, false positives, and false negatives.

	Predicted: Seatbelt	**Predicted: No Seatbelt**
Actual: Seatbelt	48	2
Actual: No Seatbelt	0	50

Interpretation:

- **True Positives (TP)**: 48 cases where seatbelts were correctly detected.
- **True Negatives (TN)**: 50 cases where no seatbelt was correctly identified.
- **False Positives (FP)**: 0 — Excellent result.

- **False Negatives (FN)**: Only 2 — in cases with poor lighting or partially obscured belts.

6.3 Visual Results

The model's predictions were visualized by overlaying the results on the test images. Example outcomes:

- **Correct Detection (Seatbelt Present):**
  Clear diagonal stripe across torso identified; confidence > 95%
- **Correct Detection (No Seatbelt):**
  Clean background, no belt visible; model marked class 0 with high certainty
- **Incorrect Detection (False Negative):**
  Belt was partially occluded by hand or bag; model failed to detect

Sample Output Code Snippet:

```
prediction = model.predict(image_input)
label = 'Seatbelt' if prediction >= 0.5 else 'No Seatbelt'
```

6.4 Loss and Accuracy Curves

Training and validation accuracy/loss were plotted to ensure stable learning. Results showed:

- Steady increase in accuracy over epochs
- Decrease in both training and validation loss
- No signs of severe overfitting

This indicated a well-balanced model that generalized well to unseen data.

6.5 Model Robustness

The model demonstrated robustness across:

- Multiple backgrounds (car interiors, low contrast)
- Different driver poses
- Varying brightness and shadow conditions
- Real-world quality images (not studio-lit or overly clean)

Even in the presence of noise or partial occlusion, predictions remained accurate most of the time.

6.6 Limitations Observed

Despite its high accuracy, a few limitations were noted:

- **False Negatives** in dim lighting or when the seatbelt was covered
- Model sensitivity to image clarity and camera angle
- Occasional misclassifications with accessories resembling seatbelts (e.g., sling bags)

6.7 Overall Assessment

Achieving a **96.55% accuracy**, the CNN-based model proved effective for binary seatbelt classification. It performs particularly well in real-world imagery and meets the standard for potential integration into automated vehicle surveillance systems.

7. CHALLENGES FACED AND SOLUTIONS

During the development of the seatbelt detection system, several technical and practical challenges were encountered. These challenges provided opportunities to explore new solutions, improve model performance, and deepen understanding of machine learning workflows.

7.1 Dataset Imbalance

One of the early issues was the imbalance between the number of images labeled as “seatbelt” and “no seatbelt.” The majority of samples depicted drivers wearing seatbelts, while only a small portion showed drivers without seatbelts. This imbalance risked biasing the model toward the dominant class, reducing reliability in real-world detection scenarios.

Solution:
To address this, class weights were calculated and passed into the model during training. This forced the model to give equal importance to minority class samples, improving the model’s ability to detect “no seatbelt” cases effectively.

7.2 Image Quality and Variability

The dataset included images with poor resolution, varying camera angles, partial occlusions, and inconsistent lighting. In many cases, the seatbelt was barely visible or hidden by objects such as bags, hands, or clothing. These inconsistencies made feature extraction challenging for the CNN model.

Solution:
Faulty or unreadable images were removed. Data augmentation techniques such as zooming, flipping, and rotation were applied to simulate varied conditions. This helped the model generalize better and learn to detect the presence of seatbelts in more diverse scenarios.

7.3 Overfitting in Initial Training

In the early training stages, the model showed signs of overfitting — very high training accuracy but significantly lower validation accuracy. This suggested the model was learning to memorize the training set rather than generalize to unseen data.

Solution:
Regularization strategies were employed:

- Added Dropout layers to prevent neuron co-adaptation
- Used BatchNormalization to stabilize learning
- Implemented EarlyStopping to halt training when validation performance plateaued
- Augmented the data to reduce dependency on specific patterns

7.4 Limited Computational Resources

Since the model was trained on Google Colab with limited session durations and memory constraints, training deep models with large batch sizes or high resolution became a bottleneck. There were also restrictions on GPU availability during peak usage hours.

Solution:
The image size was fixed to 128×128 to reduce memory footprint. Model depth was kept optimal—not too shallow to underfit, but not so deep that training would exceed Colab limits. Additionally, session checkpoints were regularly saved to avoid data loss.

7.5 Ambiguity in Ground Truth Labels

In a few images, the distinction between seatbelt and no-seatbelt was unclear, even to the human eye. Some images showed drivers with straps that looked like seatbelts but were

actually bag slings or seat covers. These images may have introduced noise into the model's learning.

Solution:
Such samples were reviewed manually. In future, an automated labeling validation step could be introduced or a region-of-interest (ROI) detection mechanism can be added to localize and analyze only the chest/shoulder area of the driver.

7.6 Real-Time Implementation Gaps

Although the model performed well on static images, real-world applications require real-time processing of video streams with frame-by-frame inference.

Solution:
While not implemented during the internship, future plans include converting the trained model into a lightweight version (e.g., TensorFlow Lite or ONNX) and testing it on edge devices like Raspberry Pi or NVIDIA Jetson Nano for real-time inference.

7. CONCLUSION

The internship at PathVision Innovations Pvt. Ltd. provided an enriching and hands-on experience in the field of computer vision, specifically focusing on algorithm development for real-world applications. The primary task of designing and building a Convolutional Neural Network (CNN)-based model for seatbelt detection allowed me to explore the complete machine learning pipeline — from data acquisition and preprocessing to model training, evaluation, and optimization.

Over the course of two months, I developed not only technical expertise but also problem-solving abilities and a better understanding of practical constraints faced in industrial AI projects. The internship began with a strong emphasis on building foundational skills through self-paced learning in **TensorFlow**, **MATLAB**, and **OpenCV**. These initial weeks proved vital in preparing for the subsequent project development phase.

The core project, seatbelt detection, addressed a critical real-world challenge in automotive safety. The CNN model was trained on a custom dataset, implemented with batch normalization, dropout, and data augmentation to improve accuracy and robustness. With an impressive **96.55% test accuracy**, the model showed reliable performance in identifying seatbelt usage from static driver images, demonstrating its potential for deployment in real-time surveillance systems.

Through the journey, I encountered and overcame challenges such as class imbalance, limited data variability, and edge cases with occluded seatbelts. These hurdles deepened my understanding of model generalization and optimization techniques. The internship also helped in strengthening my ability to document progress, communicate findings effectively, and present results with clarity.

In conclusion, this internship served as a significant stepping stone in my academic and professional development. It gave me real-world exposure to machine learning systems in action, taught me to work in a focused and deadline-driven environment, and reinforced the importance of structured thinking and continuous learning in the field of artificial intelligence.