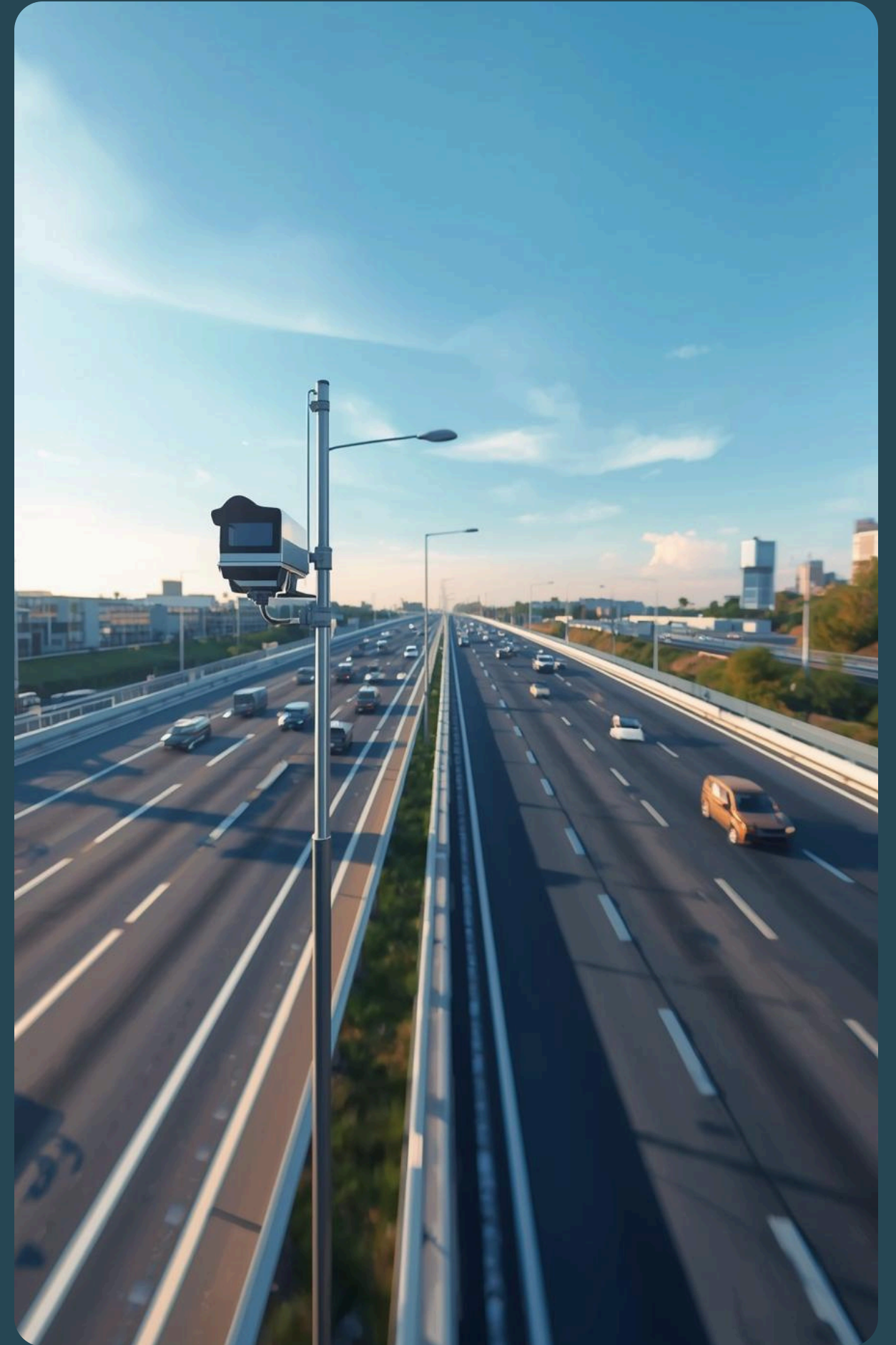
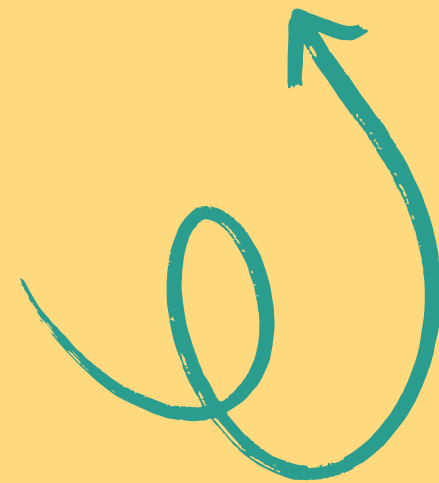


Machine-learning based Fire Detection for Highways



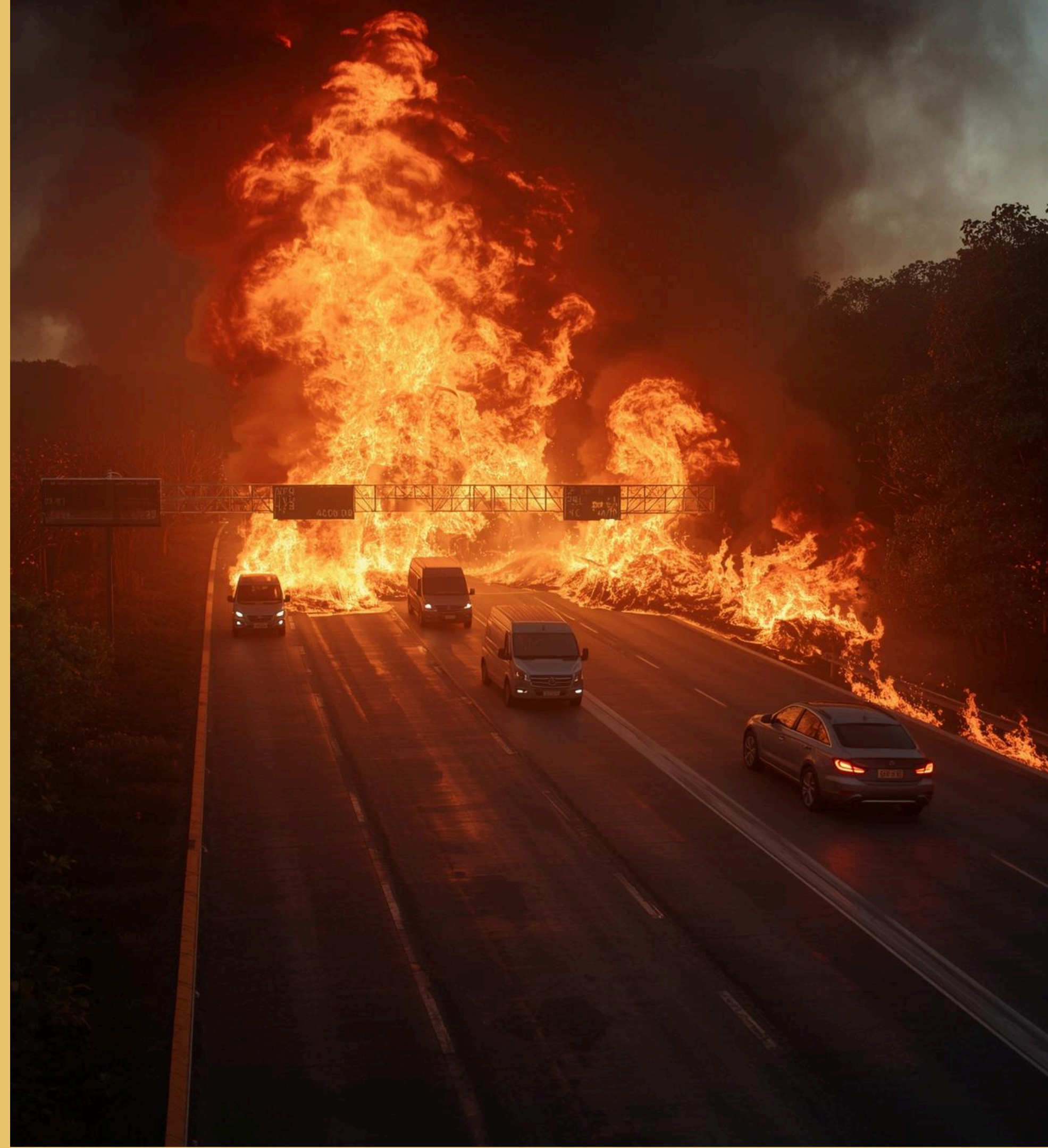
Importance and Motivation



The Escalating Danger of Highway Fires

Highway fires pose a significant threat to vehicles, the environment, and public safety, necessitating innovative solutions like AI detection systems to enhance early warning and prevention measures.

In fire safety, every second counts. Early detection during the initial "white smoke" or "small flame" stage is critical for successful containment.



Limitations of Traditional Methods

Physical Latency: Traditional sensors (smoke/heat) operate passively.

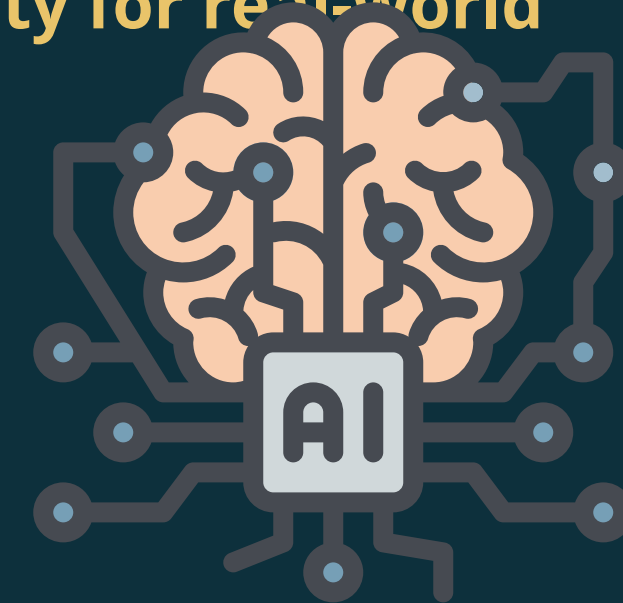
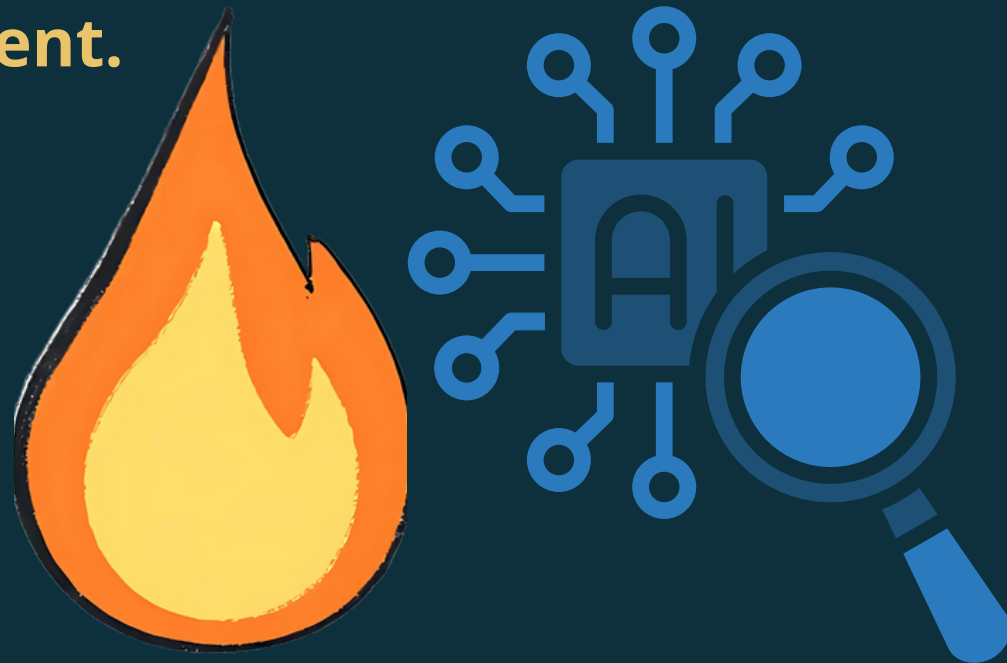
Environmental Constraints: They are less effective in open spaces

False Positives: Traditional sensors are prone to false alarms



Objectives of the System

- Develop a real-time fire hazard detection system using camera video streams.
- Train deep learning models, specifically Convolutional Neural Networks(CNNs) using the TensorFlow framework, to classify frames containing fire versus non-fire scenes by using MobileNetV2 model.
- Integrate automatic alert mechanisms, including notifications and audible alarms.
- Evaluate system performance in terms of accuracy, latency, and feasibility for real-world deployment.



Methodology of Detection



Phase 1: Data Acquisition and Analysis

Comprehensive Approach to Detection

Data Acquisition Data learning

Collection: Gather and aggregate high-quality datasets for "Fire" and "Non-fire" classes from open sources or self-recorded footage.

Pictures learning: Divided into 3 categories, each with their own tasks: Train, Test, Val.

Preprocessing & Augmentation

Apply techniques like rotation, flipping, brightness adjustment, and noise addition to diversify the data, helping the model generalize better and prevent overfitting



Phase 2: Model Training

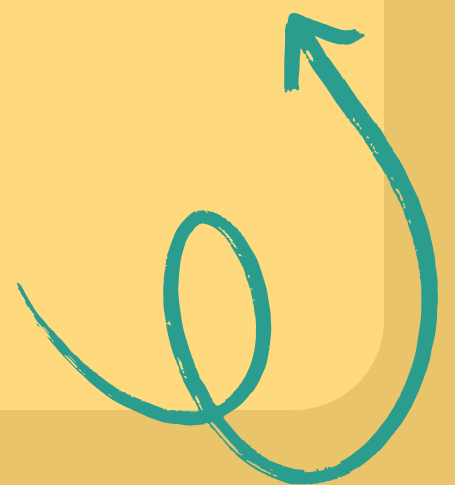
- CNN (Basic): Used as a baseline model for comparison and for static image classification tasks.
- TensorFlow-based Deep Learning Models: TensorFlow is selected as the primary framework for model development and training with MobileNetV2 model.

Training Environment: Use Python along with specialized libraries



Phase 3: System Integration

- **Develop a software application to ingest video streams.**
- **Embed the trained model weights into the application to perform inference on individual video frames.**
- **Post-processing Logic: Filter out noise and establish a confidence threshold to validate fire detection events.**



Phase 4: Alert Mechanism Design

- Hardware: Interface the computer with external speakers.
- Software: Program the functionality to play an alert sound instantly when the state variable switches to "Fire Detected".



Results of Our Research



Performance Metrics of AI System

The AI fire detection system demonstrated **impressive accuracy** and reliability, significantly reducing false alarms while improving response times, ensuring timely interventions that enhance highway safety and protect valuable resources.



Data Training

Process of Learning

Dataset & Image Classification

Collect and label images to create a binary dataset.

Image Preprocessing

Resize, normalize, and augment images to improve generalization.

Model Training

Train a CNN classifier using MobileNetV2 as a feature extractor.

Evaluation & Model Saving

Monitor performance and save the trained model for deployment.

Dataset & Image Classification

- Collect fire and non-fire images
- Label images for binary classification (fire : 0 vs. no fire : 1)
- Load images from directory-based datasets
- Assign class labels based on folder structure



Dataset & Image Classification

```
# Library for deeplearning, Tensorflow. tf is the abbreviation for tensorflow,  
# used for: building models, training, computing tensor  
import tensorflow as tf  
# import the layers to build training models  
from tensorflow.keras import layers, models  
# import training models, existing data on the internet MobileNetV2  
from tensorflow.keras.applications import MobileNetV2  
# import functions for reading images and modifying images  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
# Resizing images to 224x224 to be compatible with MobileNet  
IMG_SIZE = (224, 224)  
# Defining amount of images put into the model every update  
BATCH_SIZE = 32
```



Dataset & Image Classification

```
# change image data to the type we want
# Change pixel from [0-225] to [0-1]
# Rotate randomly images from +- 20
# Zoom in/out max +- 20%
# Flip images randomly left to right
datagen = ImageDataGenerator(rescale = 1./255, rotation_range = 20, zoom_range = 0.2, horizontal_flip = True)

# Folder for training
# Folder contains training data
# Resize to the same size: 224x224
# Each batch contains 32 images
# Binary means using for 2 layers which is fire/no_fire
train_gen = ImageDataGenerator(rescale = 1./255).flow_from_directory("dataset/train", target_size = (224, 224), batch_size = 32, class_mode="binary")

Found 3431 images belonging to 2 classes.

# Main folder containing data

val_gen = ImageDataGenerator(rescale = 1./255).flow_from_directory("dataset/val", target_size = IMG_SIZE, batch_size = BATCH_SIZE, class_mode = "binary")
```

Binary classification →

```
print(train_gen.class_indices)
```

```
{'fire': 0, 'no_fire': 1}
```

Process of Learning

Dataset & Image Classification

Collect and label images to create a binary dataset.

Image Preprocessing

Resize, normalize, and augment images to improve generalization.

Model Training

Train a CNN classifier using MobileNetV2 as a feature extractor.

Evaluation & Model Saving

Monitor performance and save the trained model for deployment.

Image Preprocessing

- Resize images to 224×224 (RGB)
- Normalize pixel values to $[0, 1]$
- Apply data augmentation:
 - Rotate ($\pm 20^\circ$)
 - Zoom ($\pm 20\%$)
 - Flip horizontally

Preprocessing ensures consistent inputs, while augmentation exposes the model to varied versions of images so it can correctly detect fire in new, unseen situations.



Image Preprocessing

```
# change image data to the type we want  
# Change pixel from [0-225] to [0-1]  
# Rotate randomly images from +- 20  
# Zoom in/out max +- 20%  
# Flip images randomly left to right  
datagen = ImageDataGenerator(rescale = 1./255,  
                             rotation_range = 20,  
                             zoom_range = 0.2,  
                             horizontal_flip = True)
```

Process of Learning

Dataset & Image Classification

Collect and label images to create a binary dataset.

Image Preprocessing

Resize, normalize, and augment images to improve generalization.

Model Training

Train a CNN classifier using MobileNetV2 as a feature extractor.

Evaluation & Model Saving

Monitor performance and save the trained model for deployment.

Model Training

- Build CNN-based image classification model
- Use MobileNetV2 as feature extractor
- Reuse pretrained ImageNet weights
- Freeze base model weights
- Configure training:
 - Optimizer: Adam
 - Loss: Binary Cross-Entropy
 - Metric: Accuracy
- Train for 10 epochs (one complete pass over all training images)
- Monitor validation data to track generalization

Model Training

Base model setup

```
# Initialize the model MobileNet
# Using existing data on imagenet
# Dense layers data are not used because only 2 layers exist
# 224x224, RGB images (3 colours)
base_model = MobileNetV2(weights= 'imagenet',
                          include_top = False ,
                          input_shape = (224,224,3)
                          )

base_model.trainable = False
```

Custom classifier + training config

```
# Training model
# Changed to feature map
# Relu is a nonlinear training relationship with 128 neurons
# Each batch we randomly have 50% neuron turned off
# 1 neuron output 0 or 1 with 'fire' : 0, 'no-fire' : 1
model = models.Sequential([base_model,
                           layers.GlobalAveragePooling2D(),
                           layers.Dense(128, activation='relu'),
                           layers.Dropout(0.5),
                           layers.Dense(1, activation='sigmoid'),
                           ])

# Machine Learning process configuration
# Weight updating algorithm
# ADAM = Adaptive Moment Estimation
# Function evaluating error between predictions and reality
# Evaluation metric
model.compile(optimizer = 'adam',
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])
```

Model Training

Model: "sequential"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dense (Dense)	(None, 128)	163,968
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

Total params: 2,422,081 (9.24 MB)

Trainable params: 164,097 (641.00 KB)

Non-trainable params: 2,257,984 (8.61 MB)

Extracts high-level visual features from the image (edges, shapes, patterns).

Compresses feature maps into a single feature vector.

Learns task-specific patterns to separate fire vs no-fire.

Reduces overfitting by randomly disabling neurons during training.

Outputs the probability for binary classification.

Model Training

```
# Starting training
# 1 epoch = model finishing examining pictures, 10-20 is appropriate
history = model.fit(train_gen, validation_data = val_gen, epochs = 10)
```

```
Epoch 1/10
108/108 ————— 87s 756ms/step - accuracy: 0.9735 - loss: 0.0722 - val_accuracy: 1.0000 - val_loss: 0.0020
Epoch 2/10
108/108 ————— 56s 519ms/step - accuracy: 0.9985 - loss: 0.0065 - val_accuracy: 1.0000 - val_loss: 0.0014
Epoch 3/10
108/108 ————— 53s 486ms/step - accuracy: 0.9997 - loss: 0.0031 - val_accuracy: 1.0000 - val_loss: 2.5969e-04
Epoch 4/10
108/108 ————— 50s 466ms/step - accuracy: 1.0000 - loss: 0.0020 - val_accuracy: 1.0000 - val_loss: 1.0177e-04
Epoch 5/10
108/108 ————— 50s 464ms/step - accuracy: 0.9994 - loss: 0.0022 - val_accuracy: 1.0000 - val_loss: 9.5699e-05
Epoch 6/10
108/108 ————— 54s 501ms/step - accuracy: 0.9997 - loss: 0.0013 - val_accuracy: 1.0000 - val_loss: 1.1874e-05
Epoch 7/10
108/108 ————— 52s 480ms/step - accuracy: 1.0000 - loss: 4.4712e-04 - val_accuracy: 1.0000 - val_loss: 2.3666e-05
Epoch 8/10
108/108 ————— 55s 506ms/step - accuracy: 1.0000 - loss: 2.8264e-04 - val_accuracy: 1.0000 - val_loss: 1.7640e-05
Epoch 9/10
108/108 ————— 51s 474ms/step - accuracy: 1.0000 - loss: 2.0444e-04 - val_accuracy: 1.0000 - val_loss: 9.5540e-06
Epoch 10/10
108/108 ————— 51s 468ms/step - accuracy: 1.0000 - loss: 1.6149e-04 - val_accuracy: 1.0000 - val_loss: 1.0864e-05
```

Process of Learning

Dataset & Image Classification

Collect and label images to create a binary dataset.

Image Preprocessing

Resize, normalize, and augment images to improve generalization.

Model Training

Train a CNN classifier using MobileNetV2 as a feature extractor.

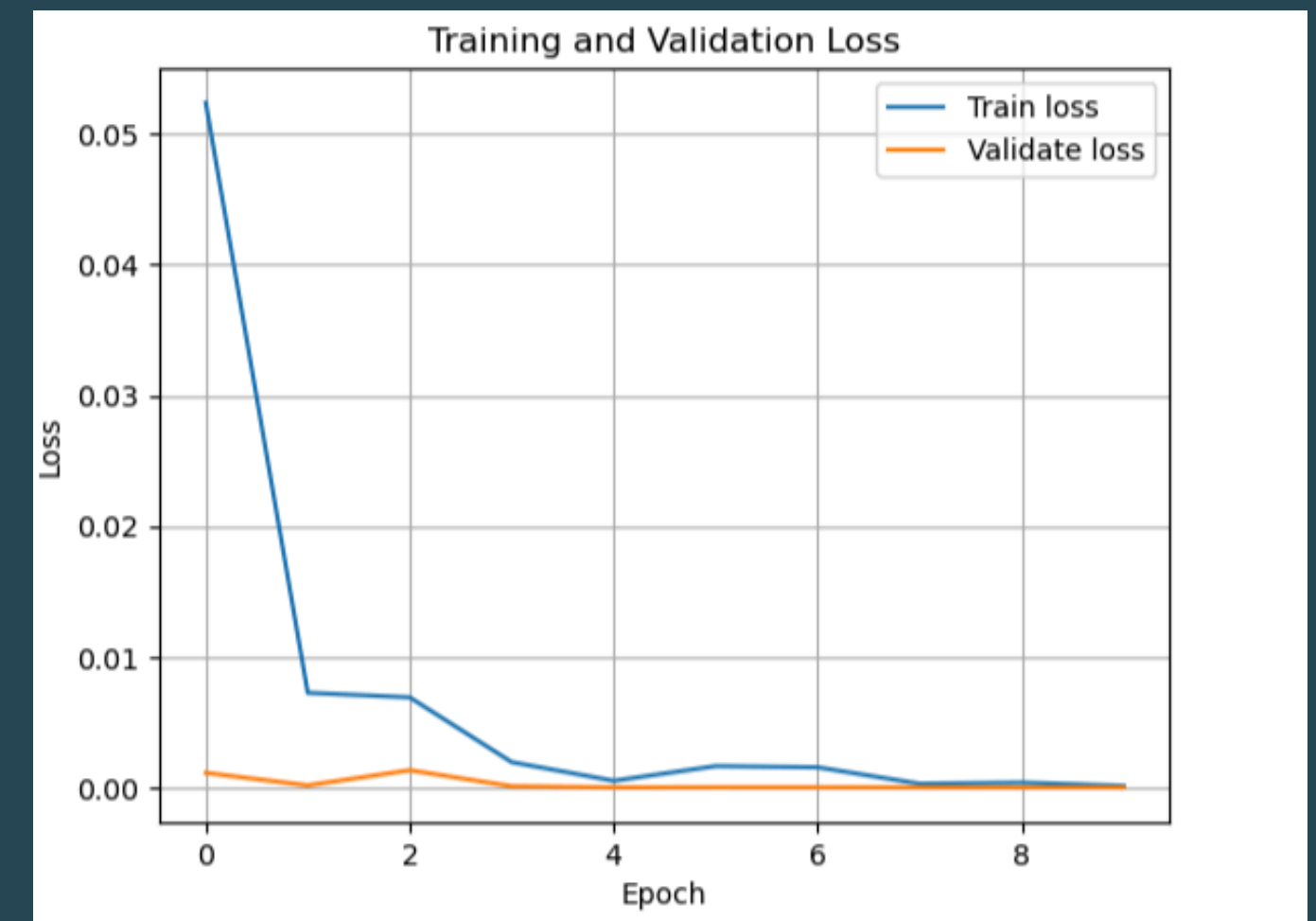
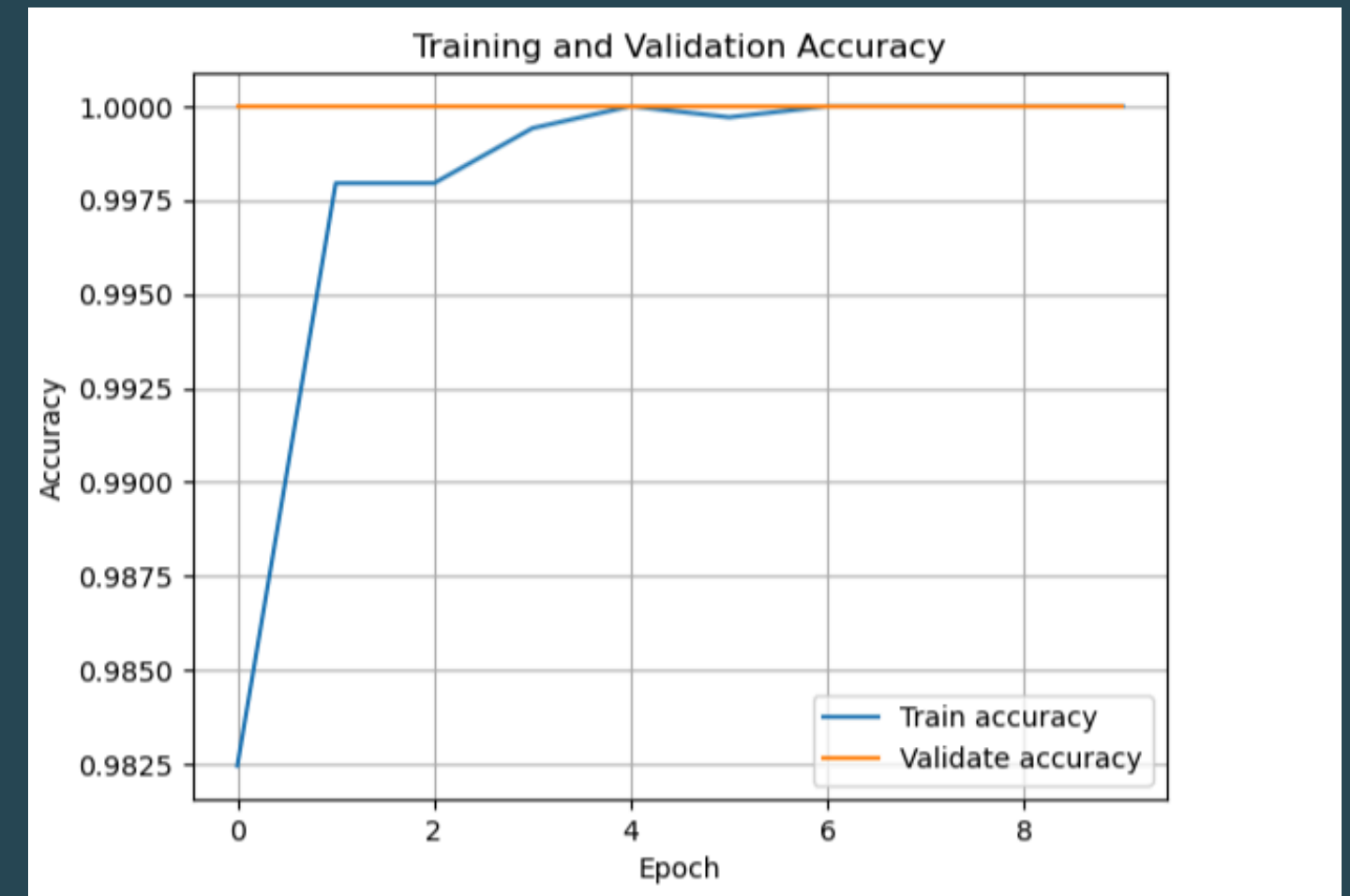
Evaluation & Model Saving

Monitor performance and save the trained model for deployment.

Evaluation & Model Saving

- Track training and validation accuracy per epoch
- Analyze training and validation loss
- Visualize learning curves to detect overfitting
- Save final model in .keras format
- Prepare model for deployment or further fine-tuning

```
# Save model post-training  
model.save('fire_detection_model_v1.1.keras')
```



Product Demonstration

AI vs Human Detection Accuracy

Conclusion and Future Directions



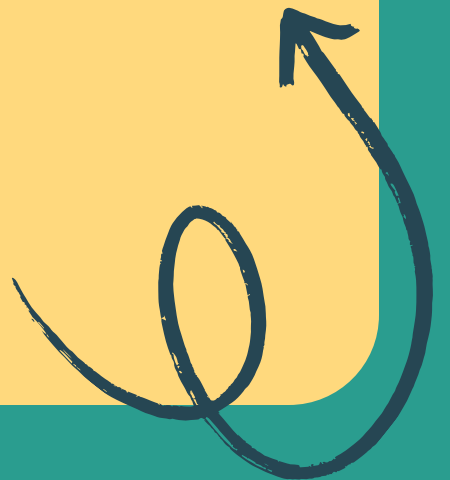
Future Uses of AI Technology

As we advance, AI can enhance highway safety through **predictive analytics** for maintenance and traffic management, ensuring efficient operations and fostering a safer environment for all road users.



Understanding the multiple advantages of our system

The AI fire detection system on highways offers significant **economic savings** by reducing damage costs and minimizing disruption to traffic. Additionally, it plays a crucial role in **environmental protection** by preventing fires that can lead to habitat destruction and air pollution. Enhanced **safety** for drivers and emergency responders is another key benefit, as timely alerts allow for quicker reactions to potential hazards. This multi-faceted approach not only safeguards lives but also preserves valuable resources, ensuring a more sustainable and efficient highway infrastructure for the future.



System Benefits of AI Detection



01

Early Detection

Implementing AI technology results in **earlier identification** of potential highway fires, allowing for faster intervention and significantly reducing the likelihood of widespread damage and danger.

02

Rapid Response

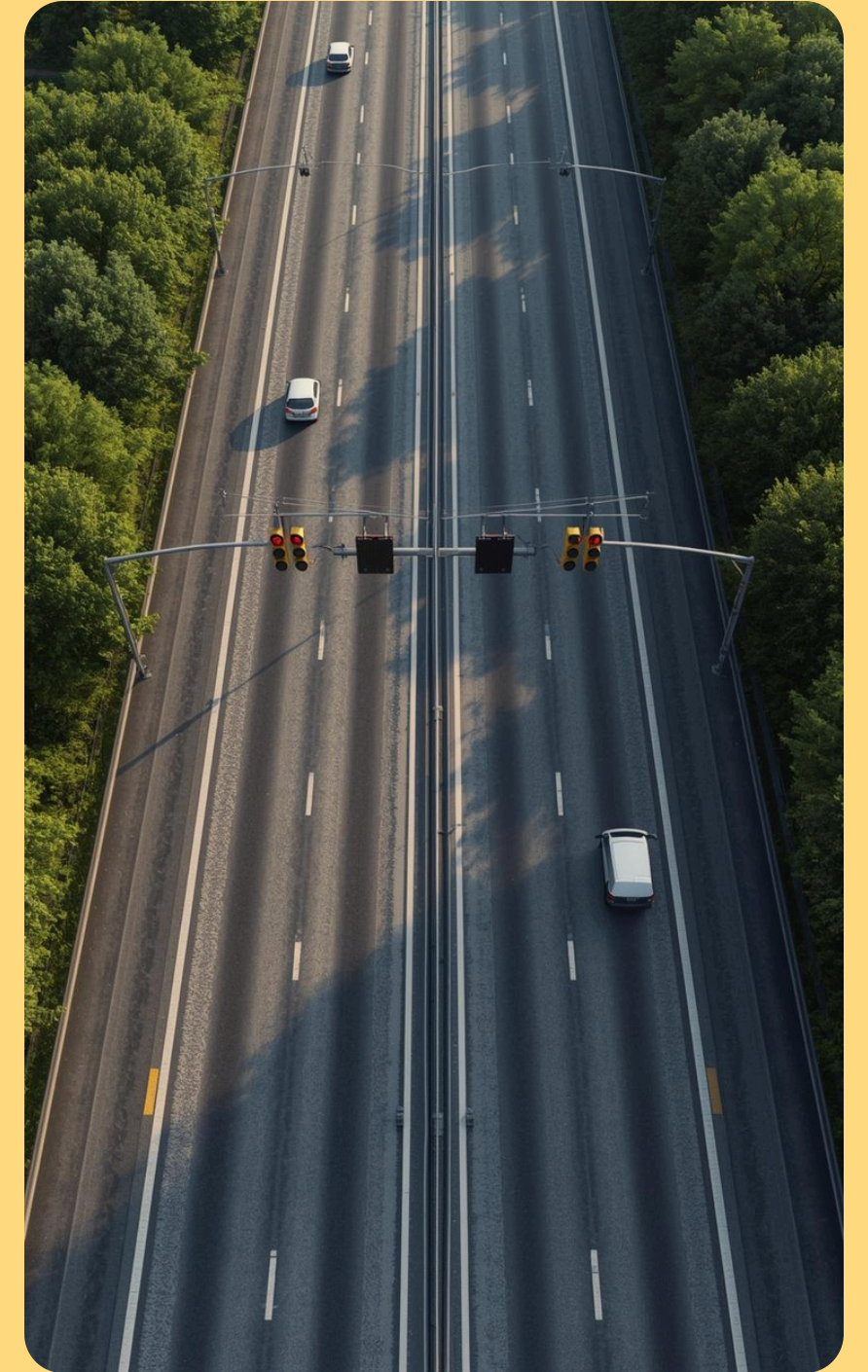
An AI-driven system facilitates a **swift emergency response**, enabling authorities to deploy resources immediately, thus minimizing both human and environmental impacts during fire incidents.

Future Enhancements in Highway Fire Detection



Traffic Integration

Enhanced coordination between highways and detection systems.



Predictive Analysis

Utilizing data to forecast potential fire risks.

Implementation Timeline

Initial Research



The project began with extensive research on existing fire detection methods, identifying gaps where AI technology could enhance response times and improve overall highway safety measures.

Prototype Development



A prototype AI fire detection system was developed, integrating advanced algorithms and thermal imaging technology, designed to accurately detect and analyze potential fire hazards on highways.

Testing Phase



The system underwent rigorous testing in controlled environments to evaluate its effectiveness in various scenarios, ensuring reliable performance before deployment on actual highway networks for safety.

Full Deployment



The AI fire detection system was successfully deployed across key highway networks, providing real-time monitoring and immediate alerts, significantly enhancing emergency response capabilities and highway safety.

Summary and Acknowledgments

In conclusion, the AI fire detection system enhances **highway safety** and **environmental protection**, ensuring faster responses to incidents and minimizing damage, ultimately leading to a safer travel experience for all road users.





A Safer Future

Saving Lives and Resources