IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Improving Object Detection And Classification For Autonomous Vehicle In Adverse Weather Conditions

Mayara Pavani

Master of technology

JNTUH

Abstract: Fundamental building blocks in the field of autonomous vehicle perception, which has seen significant progress in recent years, are object detection and classification. These innovations are essential for giving cars the ability to sense and comprehend their environment, which promotes safe navigation and well-informed decision-making. Even with these advancements, present-day perception algorithms sometimes struggle to remain dependable under harsh weather situations like dimly lit rooms, intense sunlight, reflecting glare, or obscuring fog. These circumstances have the potential to seriously impair perception system performance, raising safety concerns. As we move closer to the objective of complete autonomy, it is critical that we create a robust perception system—a system that can confidently manage these various and demanding external circumstances. With the help of such a system, autonomous cars will be able to function consistently safely and precisely, even in unpredictable real-world situations. Achieving this degree of perception system complexity is not just a goal, but also a prerequisite for the development of fully autonomous cars that have the potential to completely transform the way we travel.
Our study uses the CNN method, which is a powerful tool for deep learning, to take on this difficulty head-on. The CNN is being trained to be exceptionally good at identifying cars in bad circumstances, such as fog and rain. The objective is to provide our system with the capacity to surpass the constraints of hazy images and accomplish high-fidelity, real-time vehicle recognition.

This results in a notable advancement in AV safety, guaranteeing their ability to confidently traverse even the most difficult weather situations. We're laying the groundwork for a time when autonomous cars will be able to drive on roads with greater safety and efficiency in all weather conditions by fusing the capabilities of deep learning and WDS. In this research, we are detecting vehicles in inclement weather by utilizing an algorithm based on YOLO.

INTRODUCTION

Vehicle detection is a critical component in traffic monitoring and forms the basis of sophisticated visual surveillance systems. The development of AI-heavy autonomous or self-driving automobiles has evolved dramatically with the advent of sophisticated sensors, powerful GPUs, and state-of-the-art deep learning algorithms. A new trend in the automotive industry is the endeavor by technological advancements to radically change how cars function and interact with their surroundings.
One of the main challenges facing these sensor-based detection systems, though, is using them in bad weather. Severe fog, heavy rain, snowstorms, dust storms, and low light levels can all significantly degrade image quality by obstructing visibility.

Because of the decreased visibility, there may be a higher risk of traffic accidents due to the sensors' inability to accurately detect vehicles. Reducing these issues requires the development of efficient picture enhancement techniques that can highlight distinguishing characteristics or improve visual clarity in challenging circumstances. Vehicle tracking performance can be significantly enhanced. Along with aiding in identification, better image quality increases the overall dependability and safety of autonomous driving systems.

Thus, in this study, we use deep neural networks to provide weather identification in four different weather circumstances (snowy, sandy, overcast, and rainy) by using the capabilities of our high-performance commodity machine's Nvidia GPU and transfer learning techniques. In order to detect weather, we develop a deep learning framework that examines and contrasts the output of three well-known, effective deep Convolutional neural networks (CNNs): CNN, ResNet-50, and EfficientNet-b0, Yolo. The proposed model is generally sensitive and precise enough to be used with autonomous vehicles to help them make the appropriate choices, especially in bad weather.

1. LITERATURE SURVEY

Reference DAWN: Vehicle Detection in Adverse Weather Nature Authors for this article are Mourad A. Kenk, Mahmoud Hassaballah

1. **LiDAR (Light Detection and Ranging) Systems**:

Advantages: Due to accurate distance measurements, it has high accuracy in 3D mapping and object detection.

- It also functions well in a variety of lighting conditions, including low light and complete darkness.
- Offers comprehensive point cloud information for strong perception.
- Is able to identify items farther away.

Drawbacks:

- Expensive: LiDAR systems may be too expensive for widespread use.
- Restricted resolution: Sparse point clouds can make it difficult to see fine details.
- Sluggish in the face of unfavorable weather (heavy rain, fog, snow).
- Wear and tear can occur in mechanical components, or moving parts.

2. **Radar (Radio Detection and Ranging) Systems**:

Advantages:

- Works well in adverse weather conditions (rain, fog, etc.).
- Long-range detection capability.
- Relatively lower cost compared to LiDAR.
- Robust and reliable.

Disadvantages:

- Coarse resolution: Limited ability to distinguish fine details.
- Difficulty in detecting stationary objects.
- Vulnerable to interference from other radar systems.
- Cannot provide detailed 3D information[1].

3. **Camera Systems**:

Advantages:

- Cost-effective and widely available.
- High-resolution images for visual perception.
- Useful for lane detection, traffic sign recognition, and object classification.
- Works well in daylight conditions.

Disadvantages:

- Prone to poor visibility in adverse weather (rain, fog, etc.).
- Sensitive to lighting changes (shadows, glare).
- Limited depth perception without additional sensors.

4. **Ultrasonic Sensors**:

Advantages:

- Low cost and simple design.
- Used for short-range obstacle detection (parking, close proximity).
- Works well in parking scenarios.
- Reliable for detecting nearby objects.

Disadvantages:

- Limited range (typically a few meters).
- Low accuracy and resolution.
- Susceptible to other ultrasonic sources' interference.
- Not suitable for high-speed driving[2].

Vehicle Detection in Foggy Weather using Deep Learning.

1. **Advantages**:
   - **Improved Safety**: Accurate vehicle detection in foggy conditions enhances safety by preventing collisions and enabling timely responses.
   - **Real-Time Capability**: Deep learning-based methods can process images rapidly, providing real-time detection even in challenging weather.
   - **Adaptability**: These models can learn from large datasets, adapting to varying foggy conditions.
   - **Reduced Reliance on External Sensors**: When visibility is poor, relying solely on LiDAR or radar may be insufficient. Deep learning can complement sensor-based approaches.
2. **Disadvantages**:
   - **Data Limitations**: High-quality labeled data for foggy conditions is scarce, affecting model performance.
   - **Complexity**: Deep learning models require substantial computational resources and expertise for training and optimization.

- **False Positives**: Fog-induced noise may lead to false positives, impacting reliability.
- **Generalization Challenges**: Models trained on specific foggy conditions may struggle in novel scenarios.

Detection in Adverse Weather Conditions for Autonomous
Vehicles via Deep Learning
Authors for ref are Qasem Abu Al-Haija * , Manaf Gharaibeh and Ammar Odeh

1. **Advantages**:
   - **High Accuracy**: Deep learning models, such as the ones evaluated in your paper (SqueezeNet, ResNet-50, and EfficientNet-b0), achieve remarkable accuracy rates (e.g., 98.48%, 97.78%, and 96.05%). Accurate weather detection is crucial for safe autonomous driving.
   - **Real-Time Processing**: These models can provide rapid inference, which is essential for real-time decision-making in autonomous systems.
   - **Performance on Combined Datasets**: Combining datasets (e.g., MCWDS2018 and DAWN2020) allows for better generalization across different weather conditions.
   - **GPU Acceleration**: Leveraging GPUs for inference enables efficient processing, contributing to shorter response times.
   - **Novel Use of DAWN2020 Dataset**: Your work introduces the DAWN2020 dataset to deep learning applications, expanding the available resources for weather classification.
2. **Disadvantages**:
   - **Limited Standardization**: Lack of a universally accepted weather dataset makes it challenging to compare models consistently.
   - **Processing Time**: While your lightweight architecture performs well, evaluating it on various devices is essential to assess processing time and adaptability.
   - **Object Detection Scope**: Extending the system to detect other objects (e.g., signs, mailboxes) requires additional experimentation and adaptation

Object Detection in AdverseWeather for Autonomous Driving
through Data Merging and YOLOv8 and authors for ref are
Debasis Kumar * and Naveed Muhammad

1. **Objective**:
   - The study aimed to improve object detection accuracy in severe weather by merging data from open-source datasets (DAWN and ACDC).
2. **Dataset Preparation**:
   - Individual weather images (fog, rain, snow, night, sand) were collected and annotated.
   - Data was split into training (70%), validation (20%), and testing (10%) sets.
   - Merged datasets (MERGED) were created by combining training, validation, and testing images.
3. **Data Augmentation and Training**:
   - Various data augmentations were applied to choose the best-augmented version.
   - Custom weights were trained using the merged datasets.
4. **Performance Outcomes**:
   - MERGED dataset outperformed individual datasets in object detection.
   - Training on custom datasets improved accuracy, especially with more relevant images.
   - MERGED dataset consistently performed well across all subsets.
5. **Insights and Future Directions**:
   - Further enhancements can be achieved by adding more diverse datasets.
   - Larger image sizes or longer training epochs may enhance outcomes.
   - Consideration of environmental factors (e.g., sun glare) is important.
   - Regular weather datasets could complement severe weather data for accurate detection.
6. **Funding and Conflicts**:
   - The research was supported by the European Social Fund.

Perception and sensing for autonomous vehicles under adverse weather
Conditions: A survey being conducted

1. **Surveyed Influence**:
- The study investigated how adverse weather affects five major ADS sensors.
- Sensor fusion solutions were explored to mitigate weather-related challenges.

2. **Core Solution: Perception Enhancement**:
- Various machine learning and image processing techniques, including de-noising, were analyzed.
- Additional methods for classification and localization were discussed.

3. **Research Trends**:
- The direction is toward robust sensor fusion, sophisticated networks, and advanced computer vision models.
- Candidates for future ADS sensors include FMCW LiDAR, HDR cameras, and hyper spectral cameras.

4. **Challenges and Limitations**:
- Lack of relevant weather datasets remains a limitation.
- 1550 nm LiDAR faces difficulties.
- Strong light and contamination need further research.

5. **Prospects and Future Work**:
- V2X (Vehicle-to-Everything) and IoT technologies hold promise for weather research.
- Snow perception enhancement and point cloud processing are areas for improvement.

- Efforts toward sensor robustness will advance adverse weather research.

6. **Declaration of Competing Interests**:

- The authors declare no financial conflicts or personal relationships influencing their work

Detection in Adverse Weather Conditions for Autonomous Vehicles via Deep Learning<u>**And authors for reference are QasemAbuAl-Haija* ,ManafGharaibeh andAmmarOdeh**</u>

1. Proposed System:
   - An intelligent independent weather detection system was developed using deep learning.
   - Three CNN models (SqueezeNet, ResNet-50, and EfficientNet-b0) were evaluated.
   - The system classified six weather classes: cloudy, rainy, snowy, sandy, shine, and sunrise.
2. Model Performance:
   - All models achieved remarkable accuracy rates (e.g., 98.48%, 97.78%, and 96.05%).
   - The proposed model performed well, with short inference times using GPU acceleration.
3. Combined Datasets:
   - The study combined DAWN2020 and MCWCD2018 datasets to create six weather classes.
   - DAWN2020 was used for the first time in deep learning applications.
4. Comparison with Other Models:
   - Compared models used either the four-class MCWCD2018 dataset or subsets.
   - The goal is accurate weather detection with minimal inference delay for autonomous systems.
5. Future Directions:
   - Testing the lightweight architecture on various devices to evaluate processing time and accuracy.
   - Extending the system to detect other street objects (signs, mailboxes, street lighting) for real-world usability.
6. Author Contributions and Funding:
   - The authors' roles and funding sources were specified.

3. METHODOLOGY

a) Proposed Work:

1. **Choice of Algorithms**:

- We plan to utilize several powerful algorithms for object detection:
  - **YOLO (You Only Look Once)**: A real-time object detection system that creates a grid out of the image and forecasts class probabilities and bounding boxes for every grid cell.
  - **VGG16**: a deep convolutional neural network (CNN) architecture renowned for its potent ability to extract features
  - **Efficient Net B0**: a productive CNN design that strikes a compromise between performance and model size
  - **ResNet-50**: a 50-layer residual neural network that aids in reducing the training-related vanishing gradient issue
- These algorithms will work together to identify objects accurately.

2. **Evaluation Metrics**:

- To assess the performance of our models, we'll calculate the following metrics:
  - **Loss Function**: Measures the difference between predicted and actual values during training.
  - **Accuracy**: The proportion of correctly predicted instances.
  - **Recall Score**: The ratio of true positive predictions to the total actual positives.
  - **Precision**: Precision is the ratio of all anticipated positives to the number of real positive predictions.
- These metrics help us understand how well our models are performing.

3. **Algorithm Comparison**:

- We'll compare the performance of different algorithms to identify the most effective one.
- Factors such as accuracy, speed, and robustness will guide our choice.

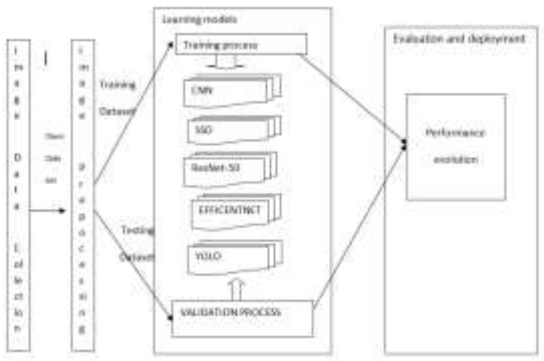
4. **Enhancing Image Quality**:

- To improve detection, we'll preprocess images using filters:
  - **Sobel Filter**: Enhances edges by detecting intensity gradients.
  - **Canny Edge Filter**: Identifies edges with high precision.
- These filters enhance object boundaries, aiding detection.

5. **Considering Environmental Conditions**:

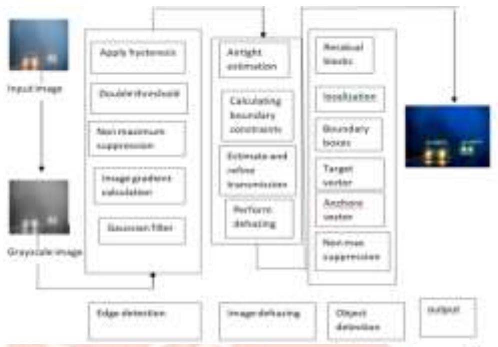
- We'll account for various weather scenarios:
  - **Sand**: Objects partially buried in sand may be harder to detect.
  - **Snow**: Reduced visibility due to snow accumulation.
  - **Fog and Rain**: Adverse weather conditions affect sensor data quality.
- Our models will adapt to handle these challenges.

b) System Architecture:



The flowchart represents the typical workflow for developing machine learning models, specifically focusing on image-based models. Here's a breakdown:

1. **Image Data Collection**:
   - The left side of the flowchart begins with data collection. Two columns labeled "Training Dataset" and "Testing Dataset" indicates the data used for model training and evaluation.
   - These datasets are essential for training and validating the machine learning models.
2. **Learning Models**:
   - In the center, we have a list of neural network architectures commonly used for image recognition tasks:
     - **CNN (Convolutional Neural Network)**
     - **SSD (Single Shot Multibox Detector)**
     - **ResNet-50**
     - **EFFICIENTNET**
     - **YOLO (You Only Look Once)**
   - These models are connected to the "Training process" box.
3. **Training Process**:
   - The training process involves feeding the data to the neural networks, allowing them to learn patterns and features from the images.
   - The trained models are then evaluated using validation data.
4. **Validation Process**:
   - The box labeled "VALIDATION PROCESS" represents the evaluation stage. Here, the trained models are rigorously tested to assess their performance.
   - Metrics such as accuracy, precision, recall, and F1-score are used to evaluate how well the models generalize to unseen data.
5. **Evaluation and Deployment**:
   - On the right side, we see the "Evaluation and deployment" column.
   - After successful validation, the best-performing model can be deployed for real-world use.



1. **Image Preprocessing: -** The input image is where the process starts.
   - Grayscale conversion of the image makes further processing easier.
   - A Gaussian filter is used to detect edges. This draws attention to the image's regions with notable intensity changes.
   On the edges, non-maximum suppression is used. By keeping only the strongest edges, this phase aids in the thinnest possible.
2. **Canny Edge Detection: -** One prominent method for locating edges in an image is canny edge detection.
3. **Image Dehazing**: This stage attempts to enhance visibility within the image, particularly in bad weather.
   - Methods for dehazing consist of:
   - Estimating Atmospheric Light: Calculating how much light the atmosphere scatters.
   - Transmission Estimation: Determine the percentage of light that passes through the foggy medium and reaches the camera.
   - Refining Transmission: Improving the estimation of

transmission even further.
- Image Dehazing: improving the image by lowering haze through the use of improved transmission.
- Estimating Residuals: Finding information that remains after the image has been dehazed.

4. **Object Localization**: - Localizing objects in the image is the focus of this step.
   - Around identified objects, bounding boxes are created.
   The position and size of each object are represented by a target vector that is formed.
   - Anchor generation facilitates the boundary box alignment with the real objects.
5. **Non-Maximum Suppression (NMS):** - NMS is used to eliminate unnecessary bounding boxes following localization.
   - Overlapping bounding boxes are discarded and only the most certain ones are kept.
6. **Result**: - The identified objects from the original input image are displayed in the final result.

c) Dataset:

DAWN Dataset:

To sum up, the DAWN dataset is an important tool for developing applications related to Intelligent Transportation Systems (ITS). It is made up of actual images taken in inclement weather, such as sand, rain, snow, and fog. With DAWN, researchers can test vehicle detection and classification algorithms in a variety of difficult traffic settings and get accurate annotations for different kinds of vehicles and people. This dataset closes a significant gap in real-world picture analysis for autonomous vehicles and safety applications.

.

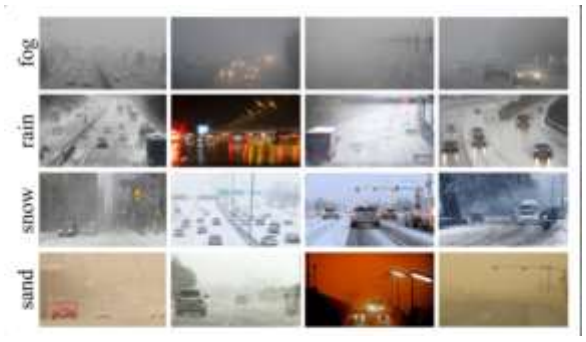


Fig 2 Dataset

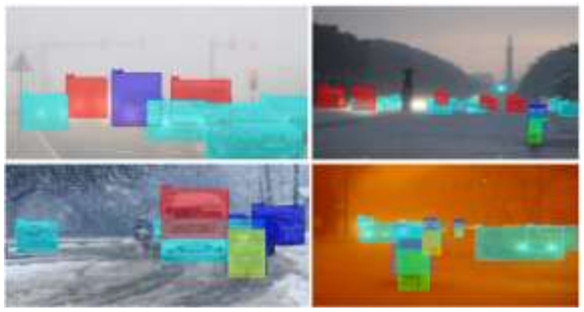
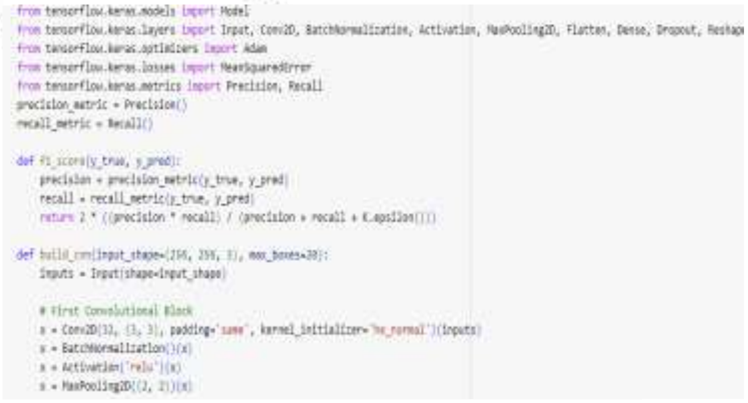


Fig 3 : Examples of annotations in DAWN dataset. The dataset is annotated using LabelMe into 7,845 total bounding boxes of five types (e.g., car, truck, motorcycles, and bicycles) and person for cyclist/pedestrian

d) Algorithms:

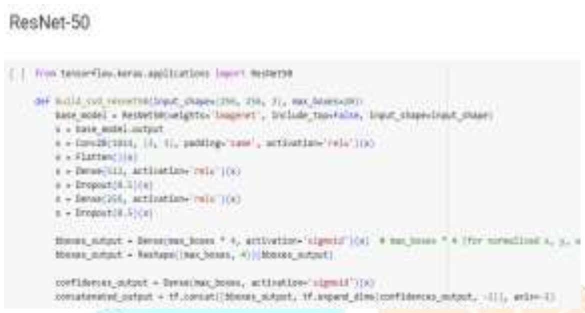
1. **Convolutional Neural Networks (CNNs)**:
   - **Purpose**: CNNs are specialized deep learning algorithms designed for tasks like image classification, object detection, and segmentation.
   - **How They Work**:
     - Convolutional layers are used by CNNs to extract features from incoming images.
     - These layers apply filters (kernels) to detect patterns like edges, textures, and shapes.
     - Pooling layers down sample feature maps, reducing spatial dimensions.
     - Fully connected layers perform classification based on extracted features.
   - **Applications**: Image recognition, autonomous vehicles, medical imaging, and more.



```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense, Dropout, Reshape
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.metrics import Precision, Recall
precision_metric = Precision()
recall_metric = Recall()

def f1_score(y_true, y_pred):
    precision = precision_metric(y_true, y_pred)
    recall = recall_metric(y_true, y_pred)
    return 2 * ((precision * recall) / (precision + recall + K.epsilon()))

def build_cnn(input_shape=(256, 256, 3), max_boxes=20):
    inputs = Input(shape=input_shape)

    # First Convolutional Block
    x = Conv2D(32, (3, 3), padding='same', kernel_initializer='he_normal')(inputs)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = MaxPooling2D((2, 2))(x)
```

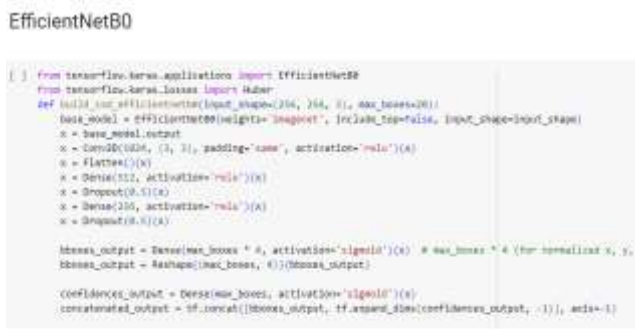
2. **ResNet-50 (Residual Network)**:
   - **Purpose**: ResNet-50 is architecture for deep neural networks.

- **Key Features**:
  - It has 48 Convolutional layers, 1 max pooling layer, and 1 average pooling layer.
  - ResNet-50 uses residual blocks, allowing training of ultra-deep networks.
  - Squeeze-and-excitation blocks enhance feature representation.
- **Applications**: Image classification, transfer learning, and feature extraction.


ResNet-50

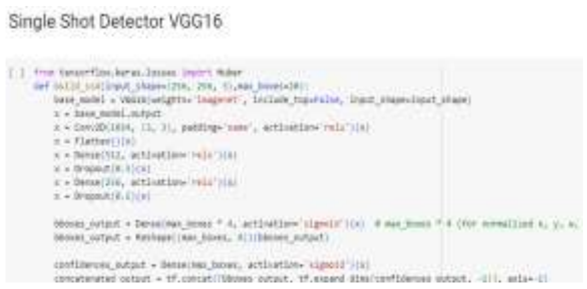
```
from tensorflow.keras.applications import ResNet50

def build_ssd_resnet50(input_shape=(256, 256, 3), max_boxes=20):
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=input_shape)
    x = base_model.output
    x = Conv2D(1024, (3, 3), padding='same', activation='relu')(x)
    x = Flatten()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(256, activation='relu')(x)
    x = Dropout(0.5)(x)

    bboxes_output = Dense(max_boxes * 4, activation='sigmoid')(x)  # max_boxes * 4 (for normalized x, y, w
    bboxes_output = Reshape((max_boxes, 4))(bboxes_output)

    confidences_output = Dense(max_boxes, activation='sigmoid')(x)
    concatenated_output = tf.concat([bboxes_output, tf.expand_dims(confidences_output, -1)], axis=-1)
```

3. **EfficientNet B0**:
- **Purpose**: EfficientNet is a scalable CNN architecture.
- **Features**:
  - It uniformly scales depth, width, and resolution using a compound coefficient.
  - EfficientNet-B0 is based on inverted bottleneck residual blocks (similar to MobileNetV2).
  - It uses fewer parameters to attain great precision.
- **Applications**: Image classification, transfer learning, and efficient model deployment.


EfficientNetB0

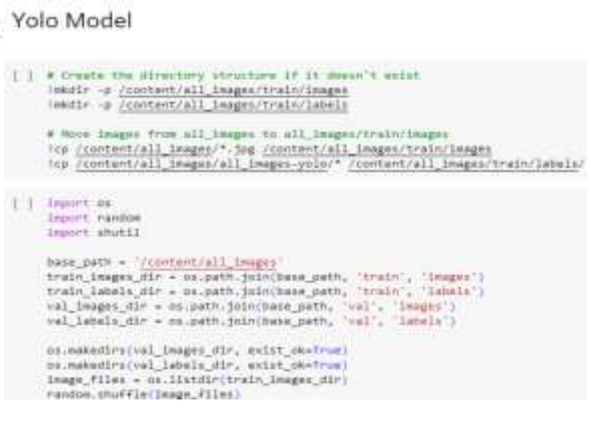
```
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.losses import Huber
def build_ssd_efficientnetb0(input_shape=(256, 256, 3), max_boxes=20):
    base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=input_shape)
    x = base_model.output
    x = Conv2D(1024, (3, 3), padding='same', activation='relu')(x)
    x = Flatten()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(256, activation='relu')(x)
    x = Dropout(0.5)(x)

    bboxes_output = Dense(max_boxes * 4, activation='sigmoid')(x)  # max_boxes * 4 (for normalized x, y,
    bboxes_output = Reshape((max_boxes, 4))(bboxes_output)

    confidences_output = Dense(max_boxes, activation='sigmoid')(x)
    concatenated_output = tf.concat([bboxes_output, tf.expand_dims(confidences_output, -1)], axis=-1)
```

4. **SSD (Single-Shot Detector)**:
- **Purpose**: SSD is a real-time object detection algorithm.
- **How It Works**:
  - SSD predicts bounding boxes and class probabilities simultaneously.
  - It divides the input image into a grid and predicts objects within each grid cell.
  - Efficient and accurate for detecting multiple objects.
- **Applications**: Real-time object detection in videos, surveillance, and robotics.


Single Shot Detector VGG16

```
from tensorflow.keras.losses import Huber
def build_ssd(input_shape=(256, 256, 3),max_boxes=20):
    base_model = VGG16(weights='imagenet', include_top=False, input_shape=input_shape)
    x = base_model.output
    x = Conv2D(1024, (3, 3), padding='same', activation='relu')(x)
    x = Flatten()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(256, activation='relu')(x)
    x = Dropout(0.5)(x)

    bboxes_output = Dense(max_boxes * 4, activation='sigmoid')(x)  # max_boxes * 4 (for normalized x, y, w,
    bboxes_output = Reshape((max_boxes, 4))(bboxes_output)

    confidences_output = Dense(max_boxes, activation='sigmoid')(x)
    concatenated_output = tf.concat([bboxes_output, tf.expand_dims(confidences_output, -1)], axis=-1)
```

5. **YOLO (You Only Look Once)**:
- **Purpose**: YOLO is another real-time object detection algorithm.
- **Key Features**:
  - YOLO predicts bounding boxes and class probabilities in one pass.
  - It uses a single CNN for both localization and classification.
  - Efficient and accurate, suitable for real-time applications.
- **Applications**: Real-time object detection, surveillance, and self-driving cars.


Yolo Model

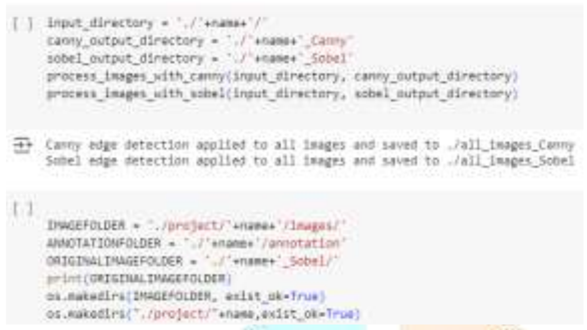
```
# Create the directory structure if it doesn't exist
!mkdir -p /content/all_images/train/images
!mkdir -p /content/all_images/train/labels

# Move images from all_images to all_images/train/images
!cp /content/all_images/*.jpg /content/all_images/train/images
!cp /content/all_images/all_images-yolo/* /content/all_images/train/labels/
```

```
import os
import random
import shutil

base_path = '/content/all_images'
train_images_dir = os.path.join(base_path, 'train', 'images')
train_labels_dir = os.path.join(base_path, 'train', 'labels')
val_images_dir = os.path.join(base_path, 'val', 'images')
val_labels_dir = os.path.join(base_path, 'val', 'labels')

os.makedirs(val_images_dir, exist_ok=True)
os.makedirs(val_labels_dir, exist_ok=True)
image_files = os.listdir(train_images_dir)
random.shuffle(image_files)
```

6. **Sobel Filter and Canny Edge Detection**:
- **Purpose**: These are image processing techniques for edge detection.
- **How They Work**:

- Sobel filter computes gradient magnitude in x and y directions.
- Canny edge detection identifies edges by finding local maxima in gradient magnitude.
- Both methods enhance object boundaries.

o **Applications**: Preprocessing for object detection, computer vision tasks, and feature extraction.



```
[ ] input_directory = './'+name+'/'
    canny_output_directory = './'+name+'_Canny'
    sobel_output_directory = './'+name+'_Sobel'
    process_images_with_canny(input_directory, canny_output_directory)
    process_images_with_sobel(input_directory, sobel_output_directory)

    Canny edge detection applied to all images and saved to ./all_images_Canny
    Sobel edge detection applied to all images and saved to ./all_images_Sobel

[ ]
    IMAGEFOLDER = "./project/"+name+'/images/'
    ANNOTATIONFOLDER = "./"+name+'/annotation'
    ORIGINALIMAGEFOLDER = './'+name+'_Sobel/'
    print(ORIGINALIMAGEFOLDER)
    os.makedirs(IMAGEFOLDER, exist_ok=True)
    os.makedirs("./project/"+name,exist_ok=True)
```

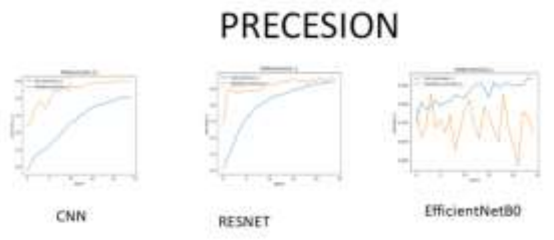
3. EXPERIMENTAL RESULTS

Precision: The precision of a model is the percentage of its positive predictions that come true.

o It is calculated as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

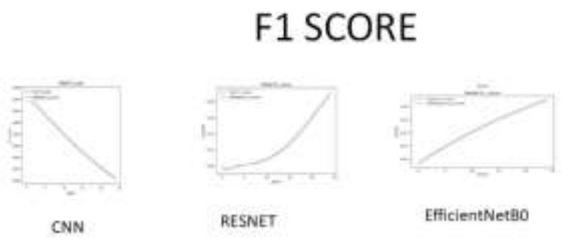
A high precision means that there are fewer erroneous positive predictions made by the model.



4. **Recall (Sensitivity)**:
   - o Recall (also known as sensitivity or true positive rate) measures how many of the actual positive instances were correctly predicted by the model.
   - o High recall indicates that the model captures most of the positive instances.
5. **F1-score**:
   - o The F1-score combines precision and recall into a single metric.

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

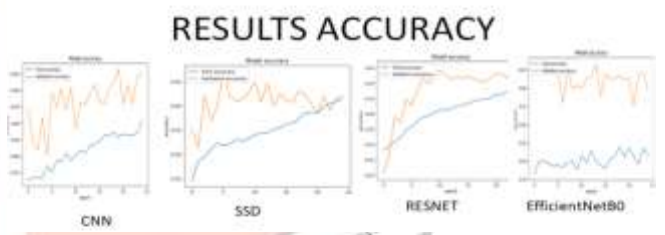
   - o F1-score balances precision and recall, especially when class distribution is imbalanced.



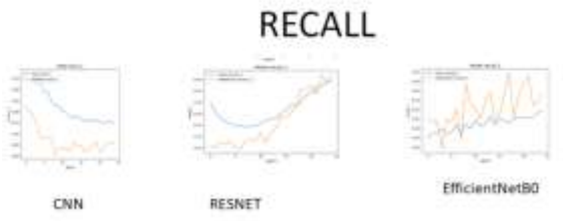
6. **Accuracy**: Accuracy gauges how well the model predicts things overall..

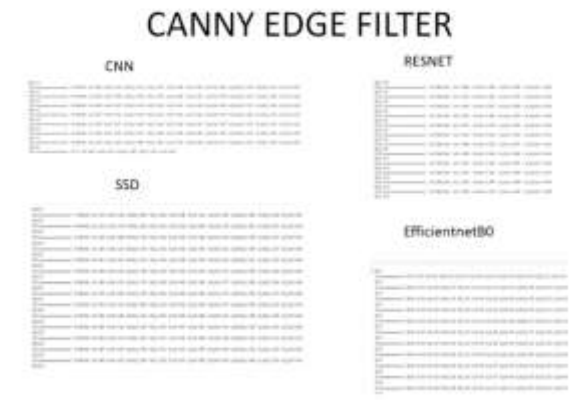
$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

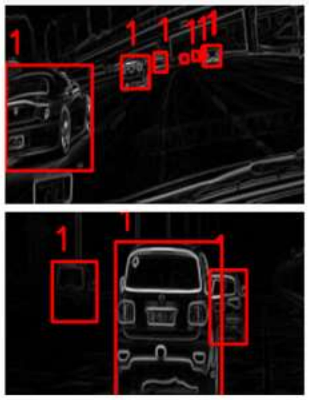
   - o While accuracy is essential, it can be misleading when classes are imbalanced.

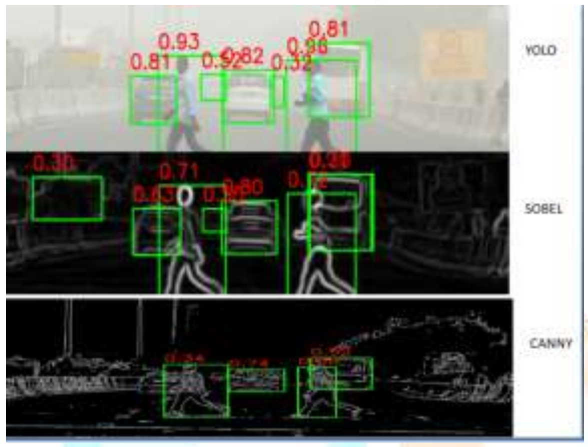


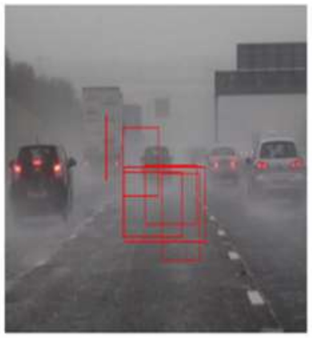
7. **Loss Function**:
   - o A loss function quantifies the difference between predicted values and actual ground truth labels.
   - o During training, the model aims to minimize this loss.
   - o Common loss functions include:
     1. **Mean Squared Error (MSE)**: Used for regression tasks.
     2. **Binary Cross-Entropy (Log Loss)**: Used for binary classification.
     3. **Categorical Cross-Entropy**: Used for multi-class classification. The choice of loss function depends on the problem type and model architecture

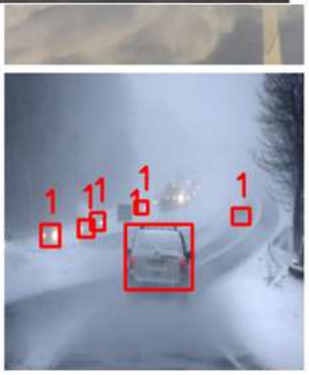


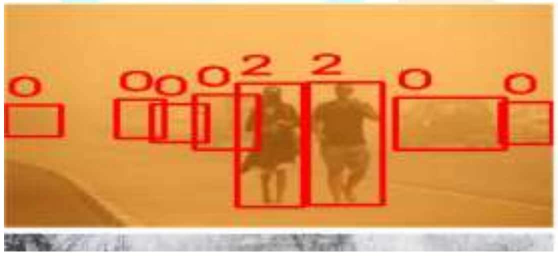


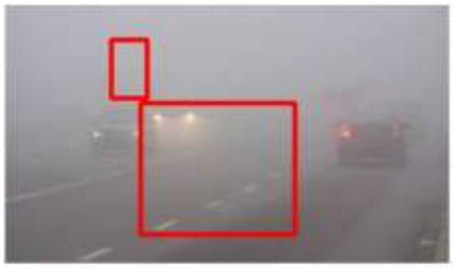


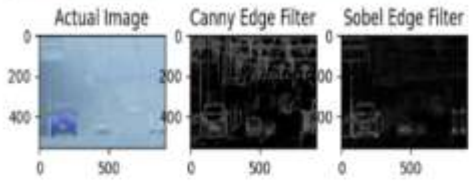












5. CONCLUSION

The proposed work aims to enhance object detection models, particularly in the context of adverse weather conditions. Researchers have experimented with various techniques, including edge detection preprocessing layers, to improve model performance. Initially, all models started with a training accuracy of approximately 0.2447. Over time, the accuracy increased, peaking around 0.3281. This upward trend suggests that the models effectively learn from the training data. The addition of an edge detection preprocessing layer significantly improved training accuracy. Accuracy increased from 0.2138 to 0.3730, demonstrating the effectiveness of this enhancement. Edge detection likely helps models focus on relevant features and boundaries. In normal trials, validation accuracy increased from 0.3335 to 0.3798. Edge detection experiments also showed improvement, with validation accuracy rising from 0.2760 to 0.3615. These trends indicate that the models generalize better to unseen data. Regular training consistently improved accuracy and loss measures across all models. The addition of edge detection further enhanced overall performance. However, recall (sensitivity) showed some fluctuations, suggesting room for improvement. Researchers can focus on: Class Imbalances: Addressing any skewed class distributions to improve recall. Hyper parameter Tuning: Fine-tuning model parameters for better performance .Complex Structures: Investigating more intricate neural network architectures. Regularization Techniques: Exploring methods to prevent over fitting. Wider Assessment Metrics: Considering additional evaluation metrics beyond accuracy. Cross-validation and broader assessment metrics can provide a deeper understanding of model generalizability. Robustness to diverse environmental conditions (beyond edge detection) remains a critical area of exploration. The proposed enhancements, especially edge detection preprocessing, contribute to better object detection models. As models continue to improve, their ability to categorize objects in challenging scenarios becomes more reliable. Continued research will refine these methods, ultimately advancing autonomous systems' safety and reliability.

Remember, this work lays the groundwork for more effective object detection, benefiting applications like autonomous vehicles and surveillance systems!

6. FUTURE SCOPE

Some potential future directions and areas of exploration related to enhancing object detection in adverse weather conditions:

1. **Multi-Modal Fusion**: Investigate methods that combine information from multiple sensors (such as cameras, lidar, radar, and thermal imaging) to improve object detection robustness. Fusion techniques can enhance performance under challenging weather scenarios.
2. **Dynamic Adversarial Training**: Explore adversarial training techniques that specifically target adverse weather conditions. By generating synthetic weather variations during training, models can learn to be more resilient to real-world weather challenges.
3. **Transfer Learning with Weather-Specific Pretraining**: Pretrain object detection models on weather-specific datasets (e.g., rainy, foggy, snowy) before fine-tuning on the target dataset. This approach leverages domain-specific knowledge and can lead to better generalization.
4. **Temporal Context Modeling**: Incorporate temporal context by considering sequential frames. Recurrent neural networks (RNNs) or 3D convolutional networks can capture motion patterns and improve object detection accuracy in dynamic weather conditions.
5. **Attention Mechanisms for Weather-Aware Features**: Design attention mechanisms that dynamically emphasize relevant features based on weather conditions. These mechanisms can adaptively adjust feature weights during inference.
6. **Real-Time Weather Estimation**: Develop lightweight weather estimation models that predict weather conditions (e.g., rain intensity, fog density) from camera images. These estimates can guide object detection models to adjust their behavior accordingly.
7. **Robustness Metrics Beyond Accuracy**: Consider non-functional requirements such as latency, energy efficiency, and reliability. Evaluate models not only based on accuracy but also on their ability to operate efficiently in real-world scenarios.
8. **Edge and Fog Computing**: Optimize object detection algorithms for edge devices (e.g., onboard vehicle systems). Efficient inference on resource-constrained hardware is crucial for practical deployment.

REFERENCES

1. Debasisss Kumar * and Naveed Muhammad B. Object Detection in Adverse Weather conditions for Autonomous Driven through Data Merging and YOLOv8
2. Vehicle Detection in Foggy Weather using Deep Learning. Miss. Nimse Shweta[1], Miss. Phaphale Apeksha[2], Miss. Thorat Mayuri[3],Miss. Walave Prajakta[4]Prof. SThanekar[5]
3. Yuxiao Zhang a,∗, Alexander Carballo b,c,d, Hanting Yang a, Kazuya Takeda Perception and detection and sensing for autonomous vehicles under adverse weather conditions: A survey
4. Tsung-Yi, S. B. L. B. R. G., " Microsoft COCO: Common Objects in Context", 2025
5. Perception and sensing for autonomous vehicles under adverse weatherconditions: A surveyYuxiao Zhang a,∗, Alexander Carballo b,c,d, Hanting Yang a, Kazuya Takeda a,c,d
6. Detection and classification in Adverse Weather Conditions for Autonomous Vehicles via Deep Learning Qasem Abu Al-Haija * , Manaf Gharaibeh and Ammar Odeh

7. **Image-Adaptive You look only once for Object Detection in Adverse Weather Conditions**: Liu, W., Ren, G., Yu, R., Guo, S., Zhu, J., & Zhang, L.
8. Heinzler, R., Piewak, F., Schindler, P., Stork, W., 2020. Convolutional nueral network -based lidar point cloud de-noising in adverse weather. IEEE Robot. Autom. Lett. 5 (2), 2514–2521.
9. Hill, D.J., 2015. Assimilation of weather radar and binary ubiquitous sensor measurements for quantitative precipitation estimation. J. Hydroinform. 17 (4), 598–613.
10. Hill, C.J., Hamilton, B.A., 2017. Concept of Operations for Road Weather Connected and Automated Vehicle Applications. Report, United States. Federal Highway Administration.
11. Hjelkrem, O.A., Ryeng, E.O., 2017. Driver and his behaviour data linked with vehicle, weather, road surface, and daylight data. Data Brief 10, 511–514.
12. Fritsche, P., Kueppers, S., Briese, G., Wagner, B., 2016. LiDAR sensorfusion in low visibility environments. In: International Conference about informatics in Control, Automation and Robotics. ICINCO, pp. 30–36.

Dataset Link:

- Dawn dataset : 'https://prod-dcd-datasets-cache-zipfiles.s3.eu-west-1.amazonaws.com/766ygrbt8y-3.zip'