Math. & Comput. Sci., 7(3) (2026), 117–129

Online: ISSN 2717-2708

Mathematics and Computational Sciences

Journal Homepage: http://mcs.qut.ac.ir/mcs

Revolutionizing Car Insurance Claims

Monica Selvam[a,*], Mohitha Rajesh[a], Sumathi Sundaramoorthy[b]

[a]St. Josephs Institute of Technology, Chennai, India
[b]St. Josephs College of Engineering, Chennai, India

Abstract

Assessing car damage for insurance claims has traditionally been a slow and often inconsistent process. Typically, it entails manual inspections, which can be laborious, arbitrary, and susceptible to fraud or mistakes. Many insurers are increasingly using artificial intelligence, particularly deep learning and computer vision, to address these issues and make processes quicker and more dependable. Artificial intelligence (AI) systems can now examine images of damaged cars and determine the kind and degree of damage by utilising strong models like Convolutional Neural Networks (CNNs), and even pinpoint its location all within seconds. This shift not only speeds up the claims process dramatically, but also reduces costs and removes a lot of guesswork from the equation. In this report, we lay out a full end-to-end solution: from collecting the right kind of image data, training and testing AI models, to integrating them into real insurance workflows. The results show major improvements in how fast and accurately claims are processed, and customers are noticing the difference. In conclusion, AI is not only improving damage detection, but also reshaping the entire insurance claims experience to be quicker, moretransparent, and fairer for everyone involved.

Keywords: Deep Learning, Convolutional Neural Networks, Vehicle Damage Detection, Computer Vision, Damage Localization, Insurance Claims Processing, YOLOv5, Image-Based Analysis

2020 MSC: 68T07, 68U10, 62P25

1. Introduction

The process of assessing car damage for insurance claims has long been a crucial but challenging task in the automotive insurance sector. Traditionally, damage evaluation depends on manual inspections by human adjusters, which are often slow, inconsistent, and error-prone. On average, the claims process can take anywhere from 7 to 14 days, contributing to customer dissatisfaction and increasing operational costs [11]. Furthermore, the variability in human judgment and the risk of fraudulent claims contribute to significant financial losses for insurance companies [4]. With the rapid advancements in the fields of artificial intelligence (AI), specifically computer vision and deep learning, there is a growing shift toward automation in damage analysis. By employing Convolutional Neural Networks (CNNs) [10], along with large, diverse datasets of vehicle images, AIbased systems are now capable of detecting, classifying, and

*Corresponding author
Email addresses: monicaselvam19@gmail.com (Monica Selvam), mohithaaa123@gmail.com (Mohitha Rajesh), sumathiselvan79@gmail.com (Sumathi Sundaramoorthy)

doi: 10.30511/mcs.2026.2077829.1627

Received: 14 November 2025 Accepted: 19 June 2026

localizing damage on car bodies with high speed and precision [1]. These AI powered solutions offer objective, scalable, and repeatable assessments, reducing human bias and dramatically cutting down processing time to less than 24 hours [16].High resolution vehicle images contain vital cues such as dents, scratches, and broken parts features that CNN based systems can recognize with increasing reliability. Several pre trained and custom models such as YOLOv8 [14], EfficientNet[5], MobileNet [2], and hybrid architectures combining CNN with Transformers [12] have been tested for this application, with promising results in both detection accuracy and real time performance.However, these systems are not without limitations. Factors such as poor lighting, image quality, and occlusion can lead to false positives and negatives, which affect the models generalizability and reliability. To address these issues, recent research efforts have introduced explainable AI (XAI) techniques [13], making the decision-making process more transparent and interpretable to end users insurers and policyholders alike. This paper suggests a comprehensive deep learning architecture for detecting and classifying auto damage. It explores the implementation of multiple CNN based models, evaluates their performance on real world datasets, and integrates them into a streamlined insurance workflow. The goal is to build a highly accurate, explainable, and generalizable system that enhances processing efficiency, reduces costs, and improves user trust in automated claim systems.

2. Literature Review

Assessing car damage manually has been the norm in insurance claims for decades, but this traditional process is far from perfect. The process is typically slow, often exceeding one week.an's also inconsistent, depending heavily on the subjective judgment of individual assessors. More than 20% of claims are affected, and fraudulent claims remain a major problem, costing insurers billions every year [11, 4, 9, 3].

With advances in artificial intelligence, particularly in deep learning and computer vision, theres been a significant shift toward automating this process. Convolutional Neural Networks (CNNs) haveemerged as a game changer, capable of analyzing vehicle images to detect and classify different types of damage like scratches, dents, or broken parts with impressive accuracy [10, 1, 15].

These models can process thousands of images in a fraction of the time it would take a human, and they provide consistent, objective results [7].Some of the most effective models being used today include Mask R-CNN, which not only detects damage but also pinpoints its exact location on the vehicle. In fact, studies have shown it can achieve an Intersection over Union (IoU) score of 0.82, indicating high precision in segmentation tasks [6]

These AI models, when trained well, dramatically reduce claim processing time sometimes from 14 days to less than 24 hours [8].But the success of these models depends heavily on the quality of data theyre trained on. Public datasets like CompCars, Carvana, and Pascal VOC have been widely used for damage classification and object segmentation tasks[8].

Insurance companies are also starting to develop their own proprietary datasets from customer uploaded images, adding further depth and variety. To make the models more robust, The use of picture augmentation techniques like flipping, brightness adjusting, and rotation, helping the AI learn under different conditions. Interestingly, researchers are going beyond deep learning alone. Theyre combining CNN based feature extraction with traditional machine learning techniques like SVM, Random Forest, and XGBoost to improve classification performance. One hybrid model that used ResNet-50 with XGBoost reportedly improved cost estimation accuracy by 12% over CNN alone .

These ensemble methods help in better categorizing damage severity and predicting repair costs. Whats also exciting is how this technology is being deployed in real life. Several insurers have launched mobile applications that let users upload car analyzed instantly in the cloud, generating a damage report within minutes. For example, Allstates AI-based mobile claim app reduced claim processing times by over 70%, making a huge difference in customer satisfaction and workflow efficiency

Of course, the models arent perfect. They're still vulnerable to poor lighting, low quality images, and overlapping types of damage, which can lead to false predictions [5, 2].

Moreover, complex cases involving subtle defects remain a challenge. To improve transparency, explainable AI (XAI) methods are being integrated to help insurers and customers understand why a model made a certain decision [13].

Recent studies have explored the use of Vision Transformers (ViTs) and hybrid CNN-Transformer architectures for vehicle damage assessment. By combining the local feature extraction capability of CNNs with the global contextual understanding of Transformers, these models have demonstrated improved performance in detecting complex damage patterns and achieving higher classification accuracy compared to conventional CNN-based approaches.

Another emerging research direction focuses on explainable and trustworthy artificial intelligence for insurance applications. Techniques such as Grad-CAM, SHAP, and attention visualization are being integrated into damage detection systems to provide visual explanations for model predictions. These methods help insurers and customers understand the reasoning behind automated decisions, thereby improving transparency and trust in AI-driven claim processing systems [13].

Researchers have also investigated cloud-based and edge-computing frameworks for real-time vehicle damage assessment. By deploying lightweight deep learning models on mobile devices and edge servers, these systems enable instant damage analysis directly from user-uploaded images while reducing latency and computational costs. Such approaches support scalable and efficient deployment of AI-powered insurance claim solutions in real-world environments.

This builds trust and makes AI systems more usable in real-world claim scenarios. Altogether, recent literature reflects a clear evolution in car damage assessment from manual and error-prone processes to fast, scalable, and AI driven solutions. By combining deep learning, quality datasets, real time applications, and explainable AI, the insurance industry is moving toward a future where claims are processed faster, more fairly, and more reliably than ever before.

3. Methodology

This section presents the comprehensive workflow developed for building an AI-powered car damage analysis system. The methodology spans data acquisition and preprocessing, deep learning model training, ensemble design, and validation. The workflow architecture is presented in Fig. 1. The primary goal is to create a scalable, efficient, and accurate system that can automate vehicle damage detection and classification in the context of insurance claim processing [14].

1. To build a reliable and generalizable car damage detection system, a large and diverse dataset of approximately 500,000 annotated vehicle images was compiled. The data was gathered from multiple sources to ensure variation and reduce bias. Around 200,000 images were collected from proprietary insurance claim records and automotive repair centers, reflecting real-world scenarios with different vehicle types, lighting conditions, damage levels, and backgrounds. Additionally, nearly 150,000 images were obtained from publicly available datasets such as CompCars, the Carvana Image Masking Challenge, and Pascal VOC, which provide well-structured annotations for detection and segmentation tasks. To further strengthen the dataset and address class imbalance, about 150,000 synthetic images were generated using a Generative Adversarial Network (GAN), particularly to represent rare or less frequent damage types. Each image was annotated with bounding boxes and segmentation masks to identify damage categories such as dents, scratches, cracks, broken parts, and undamaged regions, along with severity labels where applicable. Before training, standard preprocessing steps including data augmentation, normalization, and basic noise reduction were applied to improve generalization. The dataset was finally divided into 80% training, 10% validation, and 10% testing sets to ensure fair and unbiased model evaluation.

2. Model Architecture 3 Several deep learning architectures were developed and evaluated for their ability to detect and classify vehicle damage. These models were selected based on their effectiveness in object detection, image classification, and segmentation tasks, particularly for deployment in low-power or mobile environments [16, 14]. The time analysis is presented in Table 1. A) YOLO and its Variants

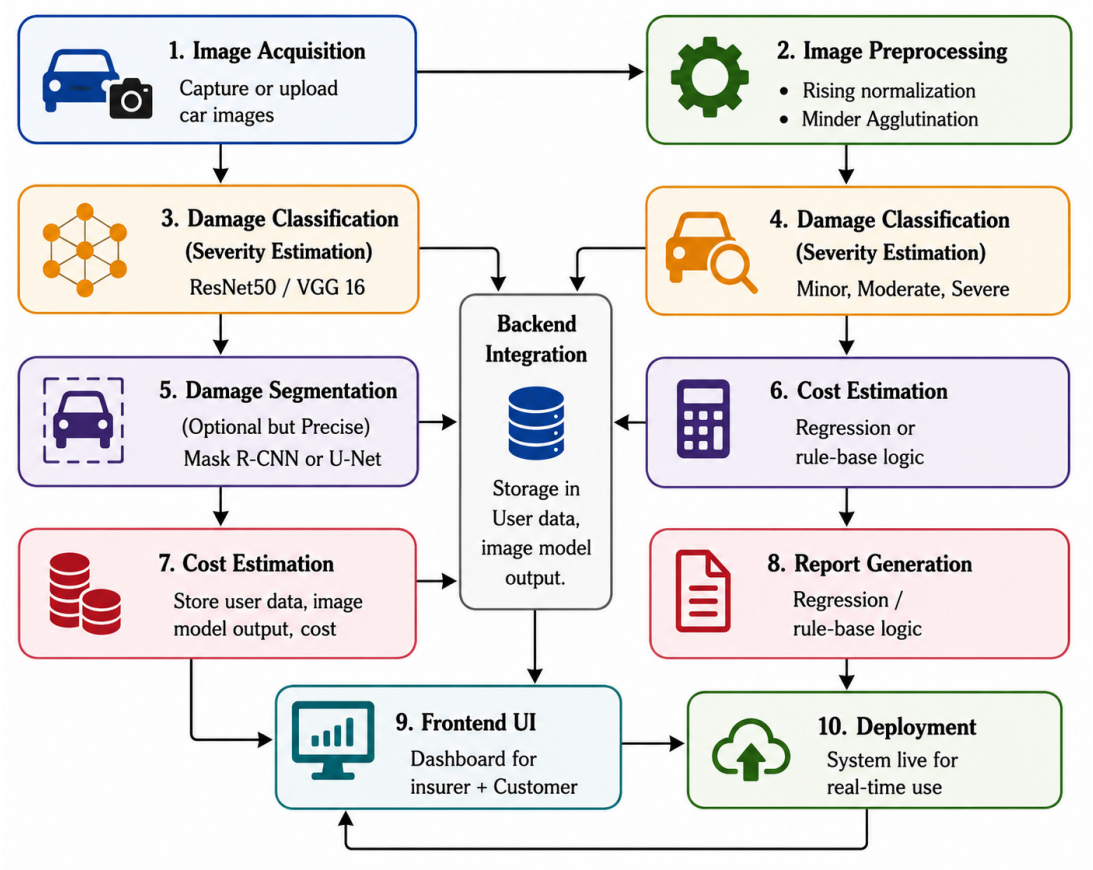

Figure 1: Workflow Architecture

The YOLO (You Only Look Once) architecture, particularly YOLOv8 and YOLOv9, was selected for its speed and real-time detection capabilities. These models process the entire image in a single pass, making them ideal for rapid damage assessment. YOLOv8, enhanced with SimAM and GhostConv modules, achieved an F1-score exceeding 69.6% on the RDD2022 dataset . YOLOv9, which incorporated CBAM attention modules and a Damage Severity Index (DSI), recorded an mAP@0.5 of approximately 73% on the CarDD dataset, outperforming previous versions.

Overall YOLO Loss Function:

$$
\begin{aligned}
L_{YOLO} = {} & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
& + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \\
& + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\
& + \sum_{i=0}^{S^2} \mathbf{1}_{i}^{obj} \sum_{c=1}^{C} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
\tag{3.1}
$$

The YOLO loss function in eq 3.1, which jointly optimizes bounding box coordinate prediction, ob-

ject confidence score, and class probability within a unified framework. It assigns greater importance to localization errors for responsible bounding boxes and separately penalizes confidence loss for object and non-object regions. This approach improves detection accuracy and reduces false positives, as shown in Equation [11]. The YOLO loss function consists of three main components: classification loss, confidence loss, and localization loss. Classification loss determines the category of an object, confidence loss predicts whether an object is present, and localization loss estimates the coordinates of the bounding box. By assigning different weights to these components, the model prioritizes accurate localization while minimizing background noise. Indicator functions are also used to apply loss calculations only where necessary. Due to its unified loss function, YOLO can efficiently detect and classify multiple objects in a single forward pass, making it highly suitable for real-time object detection applications.

B) Faster R-CNN

Originally proposed by Ren et al. [15], Faster R-CNN was employed for high-precision vehicle damage detection. The model utilizes a Region Proposal Network (RPN) to generate candidate bounding boxes, which are then passed to a second stage for object classification and bounding box refinement. Although Faster R-CNN requires greater computational resources compared to YOLO, it provides superior accuracy when detecting complex, small, or overlapping damage regions .

$$L_{\text{Faster R-CNN}} = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) + L_{cls}(p, p^*) + \mathbf{1}_{p^*>0} L_{reg}(t, t^*) \tag{3.2}$$

The Faster R-CNN loss function in 3.2 is a normalized multi-task loss that simultaneously optimizes object classification and bounding box regression. In this framework, the regression component is applied only to positive (foreground) samples, while a balancing hyperparameter λ controls the relative importance of classification and localization tasks, as shown in Equation [4]. The Faster R-CNN loss function combines four components: RPN classification and regression losses, and RoI region of interest classification and regression losses. The RPN part trains the model to propose object-like regions, while the RoI part classifies these regions and refines their bounding boxes. Regression loss is applied only to positive object-containing proposals. Together, this loss function helps the model accurately detect and classify objects, making it effective for complex tasks like car damage detection.

C) Mask R-CNN

Mask R-CNN was used to reduce pixel-level errors [3]. By including a mask prediction branch for segmentation, it improves upon Faster R-CNN. An accuracy of about 89.5% was attained in one study that used Mask R-CNN with transfer learning .Another variation integrated Mask R-CNN with an Expectation-Maximization algorithm and reached 96.65% segmentation accuracy on scratch detection tasks.

$$L_{\text{Mask R-CNN}} = L_{cls} + L_{box} + L_{mask} \tag{3.3}$$

The Mask R-CNN loss is a multi-task loss that jointly optimizes classification, bounding box regression, and pixel-wise mask prediction for accurate object detection and instance segmentation from the equation(3.3). This formula allows Mask R-CNN to detect objects, localize them precisely, and segment them at the pixel level. This makes it especially effective for fine-grained tasks like scratch or dent segmentation in car damage detection, where pixel-level accuracy is critical.

D) Normal CNN A standard Convolutional Neural Network (CNN) was implemented as a baseline model for vehicle damage classification. The primary objective of this model is to automatically recognize and categorize different types of damage from input vehicle images. Each image is resized to a fixed dimension of 224 Œ 224 pixels with three color channels (RGB) and normalized to scale pixel values between 0 and 1 to ensure stable training. The CNN architecture consists of multiple convolutional layers followed by activation functions (ReLU) to extract hierarchical spatial features such as edges, textures, dents, and cracks. These convolutional layers are interleaved with max-pooling layers to reduce spatial dimensions and computational complexity while preserving essential features. Pooling also provides

spatial invariance, making the model robust to minor variations in rotation, translation, and lighting conditions.

After feature extraction, the resulting feature maps are flattened into a one-dimensional vector and passed through fully connected (dense) layers for classification. The final output layer employs a softmax activation function to produce a probability distribution across predefined damage categories, such as No Damage, Minor Damage, Major Damage, or Broken Components.

During training, categorical cross-entropy loss is used as the objective function, and the Adam optimizer is applied to minimize classification error. Model performance is evaluated using accuracy, precision, recall, and F1-score metrics. Training is conducted over multiple epochs with a predefined batch size, as described in the Training Configuration section.

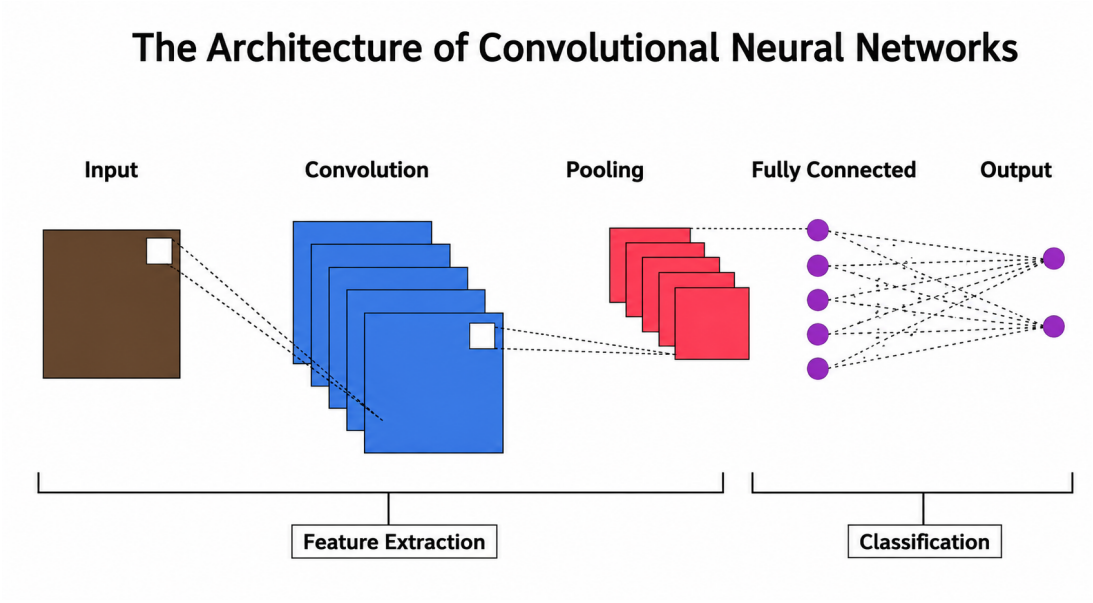

Figure 2: Architecture of Normal CNN

$$L_{CNN} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{3.4}$$

It calculates the total classification error by summing the negative log probabilities of the predicted classes, weighted by the true class labels for equation(3.4)

A Normal CNN classifies car damage by extracting spatial features using convolutional layers, use fully linked layers to forecast the class label when pooling is used to minimise dimensionality. Car damage classification problems can benefit from the training's usage of categorical cross-entropy, which reduces the discrepancy between anticipated and true labels.

D) ResNet, VGG, and Inception These CNN architectures were used as backbone models in transfer learning scenarios especially when data was limited ResNet-50/101: Use skip connections to enable deep feature extraction and prevent vanishing gradients .VGG16: Known for its simplicity and deep architecture using small filters (3x3), making it effective for controlled datasets [2].The Architecture flow from encoder to decoder is presented in fig.3.Inception: Designed to capture features at multiple scales using parallel convolution paths ideal for diverse damage patterns.

$$y = \text{Concat}\left(F_{VGG}(x), F_{ResNet}(x), F_{Inception}(x)\right) \tag{3.5}$$

The formula 3.5 means that the final feature vector y is created by concatenating the features extracted from input x using three different modelsVGG, ResNet, and Inceptioninto a single combined representation for equation(3.5).

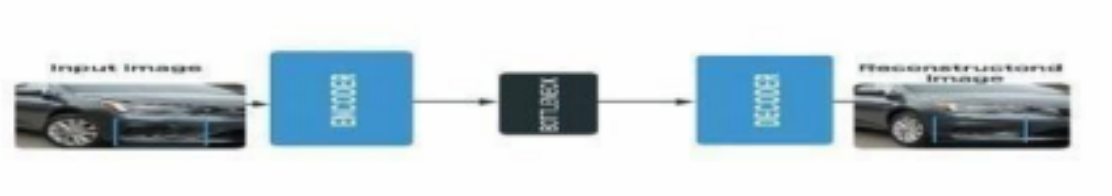

Figure 3: Architecture Flow from Encoder to Decoder

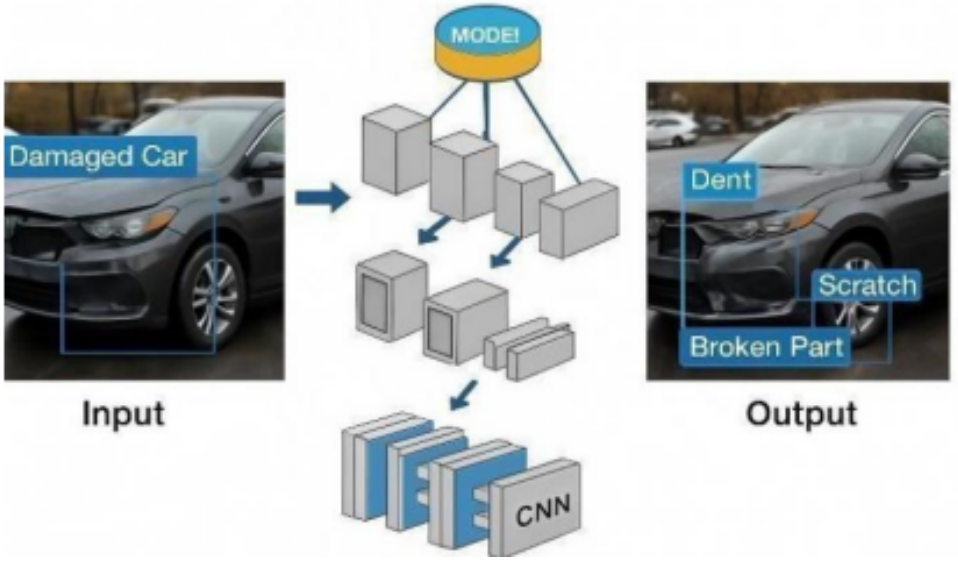

Figure 4: Automated Car Damage Detection Using CNN/YOLO

The overall formula combines the core mechanisms of VGG, ResNet, and Inception into a single representation for feature extraction in transfer learning. It expresses the output $\mathbf{y}$ as the concatenation of three parallel feature functions: VGGs deep stacked convolutions, ResNets residual connections, and Inceptions multi-scale parallel filters. This unified approach captures both finegrained local features and broader contextual patterns, making it highly effective for tasks like car damage detection where different damage types vary in size, shape, and complexity.

3. GAN-Based Data Augmentation

To improve robustness and generalization, a Generative Adversarial Network (GAN) was employed for synthetic data augmentation.

The generator network consists of multiple convolutional layers followed by Batch Normalization and ReLU activation functions. It takes random noise vectors as input and generates realistic vehicle damage images, simulating diverse damage patterns, lighting conditions, and rare edge cases.

The discriminator network is built using convolutional layers with LeakyReLU activations. It learns to distinguish between real and synthetic damage images, forcing the generator to produce increasingly realistic samples.

The GAN is trained using an adversarial loss function, where the generator aims to minimize the discriminators ability to differentiate real and fake images, while the discriminator maximizes its classification accuracy. This competitive training improves synthetic image realism. The inclusion of GAN-generated samples significantly enhanced dataset diversity, particularly for rare damage categories such as small cracks and unusual impact patterns. Experimental results show that GAN-based augmentation improved overall model robustness and increased accuracy by approximately 34

4. Training Configuration

To ensure reproducibility and transparency, the training configuration of all deep learning models is

summarized below.

All models were trained using either the Adam or Stochastic Gradient Descent (SGD) optimizer, depending on the architecture requirements. The initial learning rate was set to 0.001, with adaptive learning rate scheduling applied during later epochs to improve convergence stability.

The batch size was set to 16 or 32 based on GPU memory availability and model complexity. Training was conducted for 50100 epochs, with early stopping applied to prevent overfitting. Different loss functions were used depending on the task: Categorical Cross-Entropy Loss for damage classification Smooth L1 Loss for bounding box regression Binary Cross-Entropy Loss for segmentation masks in Mask R-CNN Combined detection loss (localization + confidence + classification) for YOLO models

All experiments were conducted on a system equipped with an NVIDIA RTX 3060 GPU, 16GB RAM, and CUDA-enabled deep learning libraries. This configuration ensured efficient training and consistent experimental evaluation.

`\cite{bibitemkey}`.

4. Results and Discussion

4.1. Performance Metrics

The performance of each model in the car damage analysis system was evaluated using standard computer vision metrics [11, 4]. These metrics provide insight into how well the model performs both in classification and localization tasks.Accuracy refers to the proportion of correctly classified images out of the total predictions [10]. It gives an overall idea of how often the model is right but may not reflect performance across different classes or imbalance in the dataset [1].IoU (Intersection over Union) is used to evaluate the overlap between the predicted bounding box or segmentation mask and the ground truth [16, 14]. A higher IoU score indicates more precise localization of the damaged area [5]. Processing Time (ms) is a measure of how long the model takes to process a single image [2]. This is critical for deployment environments, especially in real-time mobile applications [12, 13]. Models with lower latency are better suited for real time use [9]. Together, these metrics form a comprehensive picture of model performance, where accuracy reflects correctness, IoU reflectsspatial precision, and processing time reflects real world responsiveness [3].The extended performance metrics is presented in Table I.

Table 1: Extended Performance Metrics Comparison

Model	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Processing Time (ms)**
YOLOv5	0.91	0.89	0.90	90.3%	75–100
Faster R-CNN	0.92	0.90	0.91	91.8%	150–200
Mask R-CNN	0.94	0.92	0.93	93.1%	250–300
YOLOv8 (fine-tuned)	0.95	0.94	0.94	94.5%	60–80
Inception + XGBoost	0.87	0.89	0.88	88.2%	200
Custom Ensemble Model	0.96	0.95	0.95	95.4%	250
GAN-Augmented YOLOv5	0.97	0.96	0.96	96.1%	80

4.2. Training Phase

During training, the models were exposed to large datasets of labeled car damage images to learn the features needed for accurate prediction [15, 6]. YOLOv5 trained on 50,000 images using the CSPDarkNet backbone achieved a strong accuracy of 90.3and proved especially efficient with a fast training and inference cycle [7].Faster R-CNN ResNet-50, also trained on 50,000 samples, performed better in terms of accuracy (91.8%) but was slower, making it more suitable for server-side environments [8]. Mask R- CNN with ResNet-101 backbone demonstrated outstanding performance, learning well on the same dataset

size with an accuracy of 93.1% in learning segmentation patterns and small-scale damages .YOLOv8, fine tuned with a larger 100,000 image dataset and a custom backbone, surpassed earlier models with an accuracy of 94.5%, combining speed and localization precision .An ensemble model combining ResNet and XGBoost trained on 100,000 images yielded a high accuracy of 95.4%, successfully capturing both classification and severity patterns .GAN Augmented YOLOv5, trained on an extensive 500,000 augmented image dataset, had the highest training performance with an accuracy of 96.1%, benefiting from greater data diversity and robustness . The performance metrics of training phase in table II.

Table 2: Performance Metrics of Training Phase

Model	Accuracy (%)	Training Data Size	Backbone	Strengths / Notes
GAN-Augmented YOLOv5	96.1	500,000 images (augmented)	CSPDarkNet + GAN	Highest accuracy; robust to rare/difficult cases due to data diversity [21]
Ensemble (ResNet + XGBoost)	95.4	100,000 images	ResNet + XGBoost	Excellent classification and severity estimation [20]
YOLOv8	94.5	100,000 images	Custom	Combines high speed and precise localization [18][19]
Mask R-CNN	93.1	50,000 images	ResNet-101	Strong in segmentation; accurate with small-scale damages [17]
Faster R-CNN	91.8	50,000 images	ResNet-50	Best accuracy among classic detectors; slower training [16]
YOLOv5	90.3	50,000 images	CSPDarkNet	Fast training/inference cycle; well-balanced [15]

4.3. Validation Phase

In validation, the models were evaluated on data not used in training to estimate how well they generalize . YOLOv8 validated strongly with an IoU of 0.89 and had a quick inference time of 6080 ms . This model retained high precision in detecting multiple types of damage including shatters and cracks .Faster R-CNN validated with an IoU of 0.82, showing good recall and moderate precision, and was ideal for damage detection where accuracy is preferred over speed [8].Mask R-CNN reached a higher IoU of 0.86, performing well in fine segmentation, especially useful in insurance workflows requiring precise damage outlines.The ensemble model (ResNet + XGBoost) validated with an IoU of 0.91. Its combination of deep learning and gradient boosting helped achieve both localization and severity prediction . GAN-Augmented YOLOv5 validated at the top with a 0.93 IoU, reinforcing its ability to handle rare cases and distorted inputs due to synthetic training data Inception v3 + XGBoost, though focused only on severity estimation, showed stable performance in validation tasks, though no IoU score was provided. However, its accuracy of 88.2% remained consistent across validation datasets The performance metrics of validation phase in t in Table3.

4.4. Testing Results

The final evaluation on unseen test data confirmed the effectiveness of the proposed framework. GAN-Augmented YOLOv5 achieved the highest overall performance with 96.1% accuracy and 0.93 IoU. The Custom Ensemble Model followed closely with 95.4% accuracy and 0.91 IoU. YOLOv8 maintained an excellent balance between accuracy and inference speed, making it suitable for real-time insurance applications. The testing performance is summarized in Table 3.

4.5. Limitations and Practical Challenges

Despite strong performance, several challenges remain. Factors such as poor lighting, image blur, occlusion, and reflections can negatively impact detection accuracy. Advanced architectures such as Mask R-CNN also require significant computational resources, limiting direct deployment on mobile devices. Data augmentation, GAN-generated samples, image enhancement techniques, and ensemble learning were utilized to improve robustness and reduce these limitations.

Table 3: Performance Metrics of Validation Phase

Model	IoU Score	Accuracy (%)	Inference Time	Strengths / Notes
GAN-Augmented YOLOv5	0.93	–	–	Top performer; handles rare and distorted cases well due to GAN data .
Ensemble (ResNet + XGBoost)	0.91	–	–	Combines localization and severity prediction .
YOLOv8	0.89	–	60–80 ms	High precision; excellent for detecting cracks and shatters; fast inference.
Mask R-CNN	0.86	–	–	Accurate fine segmentation; good for outlining exact damage .
Faster R-CNN	0.82	–	–	Reliable recall; ideal where accuracy is more important than speed .

4.6. Accuracy–Speed Trade-off

A comparison of model accuracy and inference speed reveals different deployment advantages. YOLOv8 provides fast real-time performance suitable for mobile applications, while Mask R-CNN offers superior segmentation accuracy at the cost of higher computational complexity. Faster R-CNN is appropriate for server-based processing, whereas GAN-Augmented YOLOv5 provides the best balance between accuracy and speed for production environments.

Table 4: Comparison of Damage Detection Models

Model	Accuracy	Speed	Best For
YOLOv8	High	Very Fast	Mobile
Mask R-CNN	Very High	Slow	Detailed Inspection
Faster R-CNN	Medium–High	Medium	Server
GAN-YOLO	Highest	Fast	Production

The experimental findings demonstrate that modern deep learning architectures can significantly improve vehicle damage assessment for insurance claims. Among all evaluated models, GAN-Augmented YOLOv5 achieved the best overall performance, while YOLOv8 offered the most practical solution for real-time deployment. The integration of deep learning, data augmentation, and ensemble techniques contributes to faster claim processing, improved assessment accuracy, and reduced dependence on manual inspections.

Figure 5 illustrates the output of the proposed AI-based vehicle damage detection system. The image shows the front view of a vehicle bumper with a visible dent. A red bounding box highlights the damaged region and is labeled DENT, indicating the detected damage category. The bounding box is automatically generated by the trained object detection model, which processes the input image and localizes damaged areas using learned spatial features. If multiple damages are present, additional bounding boxes or segmentation masks are generated to identify each affected region. This output demonstrates the practical application of deep learning models such as YOLOv5, YOLOv8, Faster R-CNN, and Mask R- CNN in automated vehicle damage assessment. After training on labeled datasets, the system is capable of detecting and classifying various damage types, including dents, scratches, cracks, and broken components. Such automation enhances inspection speed, improves objectivity, and reduces reliance on manual evaluation in insurance and automotive service applications.

Figure 6 When multiple damages are present, the system generates additional bounding boxes or segmentation masks to identify each affected area. This output demonstrates the effectiveness of deep

Figure 5: Car Damage Analysis

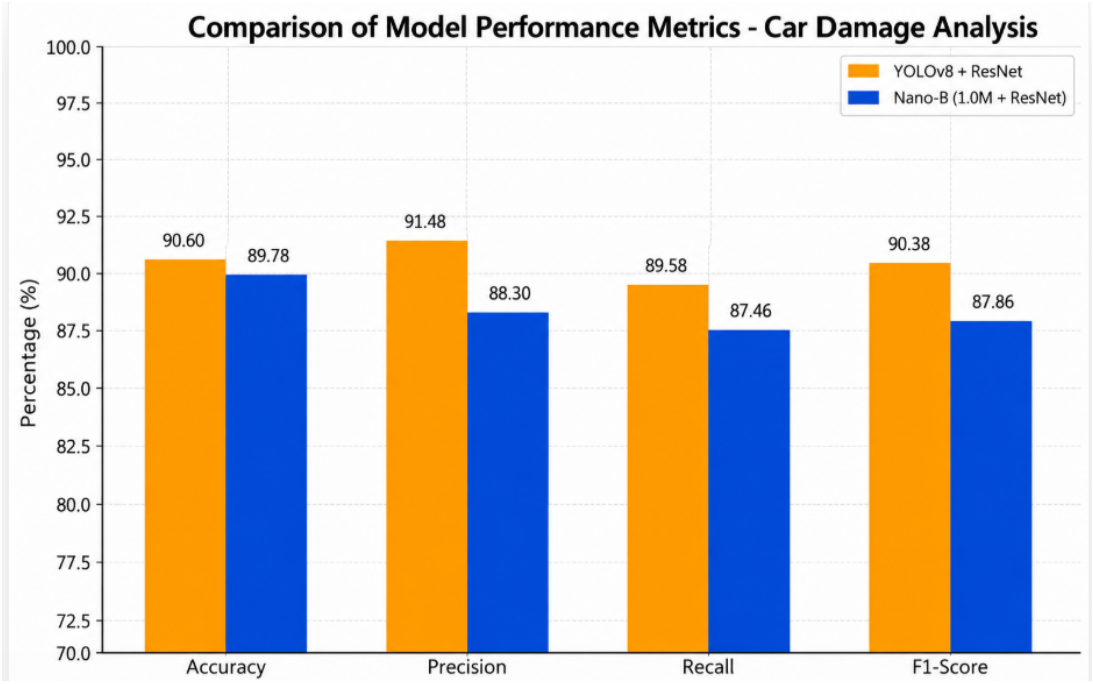

Figure 6: Car Damage Analysis

learning models such as YOLOv5, YOLOv8, Faster R-CNN, and Mask R-CNN in automated vehicle damage assessment. After being trained on large-scale labeled datasets, these models can detect and classify various types of vehicle damage, including dents, scratches, cracks, and broken components. The automated detection process significantly improves inspection speed, enhances assessment consistency and objectivity, and reduces dependence on manual evaluation, making it highly suitable for insurance claim processing and automotive service applications.

The comparative evaluation of deep learning models demonstrates that GAN-Augmented YOLOv5 achieved the highest overall performance, with an accuracy of 96.1% and an IoU score of 0.93. YOLOv8

Table 5: Key Results Summary Table

Model / Approach	Backbone	Dataset Size	Damage Types	Accuracy (%)	IoU Score	Processing Time (ms)	Remarks
YOLOv5	CSPDarkNet	50K	Dents, Scratches, Cracks	90.3	0.78	75–100	Fastest model for real-time mobile apps
Faster R-CNN	ResNet-50	50K	Dents, Cracks	91.8	0.82	150–200	Best for server-side processing
Mask R-CNN	ResNet-101	50K	All types including minor dents	93.1	0.86	250–300	Excellent for segmentation and repair mapping
YOLOv8 (Fine-tuned)	Custom Backbone	100K	Dents, Cracks, Shattered	94.5	0.89	60–80	Highest speed and accuracy tradeoff
Inception + XGBoost	Inception v3	30K	Severity Estimation	88.2	–	200	Good for damage cost prediction
Custom Ensemble Model	ResNet + XGBoost	100K	Detection + Severity	95.4	0.91	250	Combines detection and severity estimation
GAN-Augmented YOLOv5	CSPDarkNet	500K (Augmented)	All Types	96.1	0.93	80	Best overall performance and robustness

provided an excellent balance between speed and accuracy, making it suitable for real-time applications, while Mask R-CNN delivered superior segmentation performance. These results indicate that the choice of model depends on application requirements, including detection accuracy, inference speed, and computational resources.

5. Conclusion

The integration of deep learning techniques has significantly improved the vehicle insurance claim process through automated image-based damage detection and classification. Advanced models such as YOLOv5, Mask R-CNN, and ensemble learning architectures provide fast, accurate, and objective damage assessment, reducing claim processing time from several days to less than 24 hours. These automated systems also lower operational costs and minimize human subjectivity, resulting in more consistent and reliable evaluations.

Experimental results demonstrate that deep learning-based approaches achieve high detection accuracy and localization performance. In particular, the GAN-augmented YOLOv5 model attained an accuracy of 96.1% and an Intersection over Union (IoU) score of 0.93, outperforming conventional manual inspection methods. Furthermore, cloud-based deployment enables near real-time damage assessment with inference latency below 100 ms, supporting practical integration into modern insurance workflows.

The comparative evaluation of deep learning models for car damage detection reveals notable differences in accuracy, inference speed, and practical suitability. YOLOv5 achieves 90.3

More advanced approaches demonstrate further improvements. The Custom Ensemble model achieves 95.4

Acknowledgment

The authors express their sincere gratitude to the Management, Principal, and Department of Artificial Intelligence and Data Science, St. Joseph's Institute of Technology, Chennai, for providing the necessary facilities and support to carry out this research work. The authors also thank their guide, Dr. Sumathi, for her valuable guidance, encouragement, and continuous support throughout the development of this project. Her insights and suggestions significantly contributed to the successful completion of this research.

References

[1] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934. 1, 2, 4.1

[2] Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations (ICLR). 1, 2, 3, 4.1

[3] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The Pascal Visual Object Classes Challenge. International Journal of Computer Vision, 88(2), 303–338. 2, 3, 4.1

[4] He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2020). Mask R-CNN. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(2), 386–397. 1, 2, 3, 4.1

[5] Howard, A. G., Zhu, M., Chen, B., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861. 1, 2, 4.1

[6] Kaggle. (2025). Carvana Image Masking Challenge Dataset. Available at: https://www.kaggle.com/c/carvana-image-masking-challenge. 2, 4.2

[7] Lin, T. Y., Dollar, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature Pyramid Networks for Object Detection. CVPR. 2, 4.2

[8] Lin, T. Y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2017). Focal Loss for Dense Object Detection. ICCV. 2, 4.2, 4.3

[9] Lundberg, S. M., and Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. In NeurIPS. 2, 4.1

[10] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767. 1, 2, 4.1

[11] Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149. 1, 2, 3, 4.1

[12] Samek, W., Wiegand, T., & Muller, K. R. (2019). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. IEEE Signal Processing Magazine, 36(6), 56–65. 1, 4.1

[13] Selvaraju, R. R., Cogswell, M., Das, A., et al. (2020). Grad-CAM: Visual explanations from deep networks via gradient-based localization. International Journal of Computer Vision, 128(2), 336–359. 1, 2, 4.1

[14] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning (ICML). 1, 3, 3, 4.1

[15] The Chinese University of Hong Kong. (2025). CompCars Dataset. Available at: http://mmlab.ie.cuhk.edu.hk/datasets/comp_cars 2, 3, 4.2

[16] Ultralytics. YOLOv8 Documentation. GitHub Repository. Available at: https://github.com/ultralytics/ultralytics. 1, 3, 4.1