Journal of Advanced Research in Natural and Applied Sciences
dergipark.org.tr/tr/pub/jarnas
Open Access

ISSN: 2757-5195 11 (3) (2025) 210-223 doi.org/10.28979/jarnas.1651526

A New Deep Learning Based Damage Classification Pipeline for Car Glass

Ömer Küllü[1], Anıl Çelik[2], Pınar Ece Aktan Hatipoğlu[3], Mehmet Kıvılcım Keleş[4]

Article Info
Received: 04 Mar 2025
Accepted: 30 Jun 2025
Published: 30 Sep 2025
Research Article

Abstract— Intelligent damage evaluation is a popular domain of applied Artificial Intelligence (AI) in the insurance industry. Car glass damage is a common insurance claim. Accurately identifying the nature of glass damage is essential, as it directly affects the decision between repair and replacement, which has different safety and cost implications. This distinction is crucial to ensure vehicle safety and effective claim processing. This paper proposes a deep learning (DL) enhanced approach to reduce the claim process time by automatically detecting damage in car glass. The primary objective of this study is to assess whether glass damage is suitable for repair or necessitates total replacement. Owing to variations in the image zoom levels for damaged glass, our method incorporates a branched pathway to handle these differences effectively. Our pipeline employs both classification and segmentation methods. The proposed approach begins by filtering the images. Based on the results, the image is fed directly to the damage classification or subjected to further processing before classification. Furthermore, ensemble and fusion techniques were employed to improve overall efficiency. The ensemble-based strategy achieved an F1 score of 0.99 in damage classification, outperforming other approaches.

Keywords — *Deep learning, damage classification, insurance, ensemble learning*

1. Introduction

Recently, Artificial Intelligence (AI) applications have had a substantial impact on the insurance industry. Cases that include car damage assessment, document analysis, and fraud detection are popular domains for the application of AI in the insurance industry [1–5]. Insurance companies handle large numbers of daily claims. Manually reviewing each claim is inefficient. Consequently, firms seek solutions to streamline their claims process.

In this paper, we propose a pipeline that employs classification and segmentation techniques to detect damage to car glass, a common issue in insurance claims. Accurately predicting the type of damage is essential, as the distinction is not just a matter of safety but also has financial implications. Additionally, the ability to accurately detect the type of glass damage is essential for preventing insurance fraud, wherein service providers may recommend more costly replacements than repairs in situations where repairs would be adequate to ensure the safety of drivers. Insurance fraud can lead to unnecessary costs and inflated claims, which may affect all drivers' premiums. AI-based solutions can help insurers

[1]okullu@anadolusigorta.com.tr; [2]acelik@anadolusigorta.com.tr; [3]ehatipoglu@anadolusigorta.com.tr;
[4]kkeles@anadolusigorta.com.tr (Corresponding Author)
[1,2,3,4]Anadolu Sigorta, İstanbul, Türkiye

ensure that repairs are made where appropriate. The proposed pipeline consists of multiple stages. First, a filtering model is utilized to categorize images into two groups, namely, Car and Close-up, to differentiate between the zoom levels at which damaged glass may be observed (Figure 1). Multiple vehicle components are discernible in the Car category, whereas the Close-up displays only the glass damage and a limited surrounding area. Following the initial classification, images categorized as Close-up are fed directly to the glass damage classification. Due to the various angles from which Car images can be captured, additional processing is required. A classifier was employed to identify the angles; based on its output, a segmentation model tailored for that angle was utilized. The segmentation result was then cropped to isolate the regions containing glass damage, and the cropped images were used for classification. Example images from the Car and Close-up classes are shown in Figure 1.

Figure 1. Example images of the Car and the Close-up classes, as utilized in the car filter model

Finally, ensemble and feature fusion strategies and individual learners were employed to improve damage classification efficiency. The rest of the paper is organized as follows. First, a brief review of the literature on damage classification is provided. Subsequently, an overview of the proposed approach is presented, and each step is explained in detail. As the study concludes, the experimental results are discussed, and suggestions for future research are offered.

2. Related Works

Van Ruitenbeek and Bhulai [6] proposed a pipeline to detect damage using an object detection model that grouped the detected regions into 12 categories. Patil et al. [7] proposed a Convolutional Neural Network (CNN) for various car damage types. Owing to the lack of a publicly available dataset, they investigated the efficiency of the transfer learning and data augmentation techniques. Li et al. [8] employed Mask Region-Based Convolutional Neural Networks (MaskRCNN) to detect vehicles. The detected vehicles were then fed into a classifier to detect the damage. Kyu and Woraratpanya [1] proposed a pipeline with Visual Geometry Group 16 (VGG-16) and VGG-19 to detect the location and severity of damage individually. Parhizkar and Amirfakhrian [2] proposed a DL-based strategy for detecting cars and damaged parts. They used multiple CNN architectures for two different tasks. The first architecture is utilized to identify 10 parts of the car, and the second is used to recognize damaged regions. Additionally, they implemented Local Binary Pattern (LBP) [9] and Local Directional Number (LDN) [10] methods, which are algorithms used for local feature extraction to improve the accuracy of the pipeline. Zhang et al. [11] proposed a damage detection algorithm based on an enhanced MaskRCNN and transfer learning. Mallios et al. [12] proposed a pipeline to predict the damage severity using car images and structured data. In their pipeline, they used an object detector to determine the location of a car. The detected car was then fed into several DL-based feature extractors. They designed their DL-based flow to handle multiple inputs, meaning that different images belonging to the same car helped collectively determine the damage severity. Dwivedi et al. [13] designed a car damage

detection pipeline that can be used to process insurance claims. In the pipeline, they detected the damaged region using the third version of the You Only Look Once (YOLOv3) [14] and used a CNN to classify the damaged region into eight categories. Owing to a lack of publicly available datasets, Wang et al. [15] collected and shared high-quality image datasets. In addition, they proposed Deformable Convolutional Networks + (DCN+) [16], an improved version of DCN.

3. Proposed Method

Accurately detecting damage to car glasses is challenging. Damage detection performance is influenced by the complexity of the damage and the conditions under which it is observed. Small cracks can be detected, and reflections may be misinterpreted as damage. Furthermore, the distance between the vehicle and the camera can affect the detection accuracy. Our proposed solution addresses these challenges by ensuring that only a localized region surrounding glass damage is evaluated. The workflow of the proposed approach is illustrated in Figure 2.

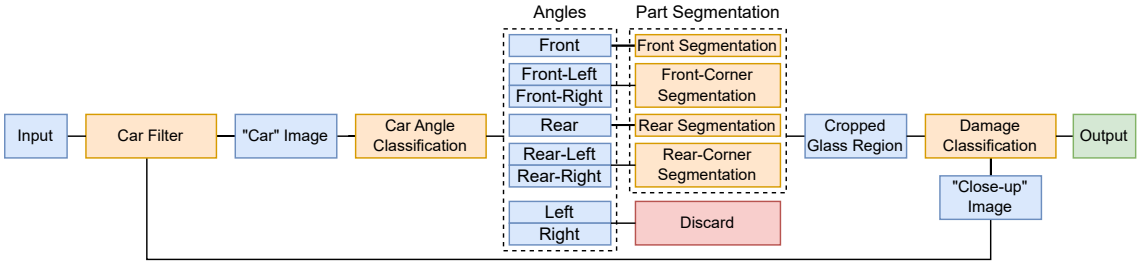

Figure 2. Workflow of the approach

3.1. Car Filter

A filter model was first employed in the proposed pipeline to classify images as Car or Close-up. Close-up images are used for glass damage classification, whereas Car images undergo additional processing. As the car-filtering approach, EfficientNetV2 [17] was selected. EfficientNet [18] is an architecture that aims to optimize width, depth, and resolution to achieve better performance with less computational power. The V2 architecture was improved on the original with fine-tuned neural architecture search, non-uniform parameter(Width, Depth, and Resolution) scaling, new layers, and multi-stage training. EfficientNetV2 begins with a 3×3 convolution block, followed by the FusedMBConv and MBConv blocks. Finally, the architecture is concluded with a 1×1 convolution, pooling, and a fully connected layer. EfficientNetV2's architecture is shown in Figure 3.

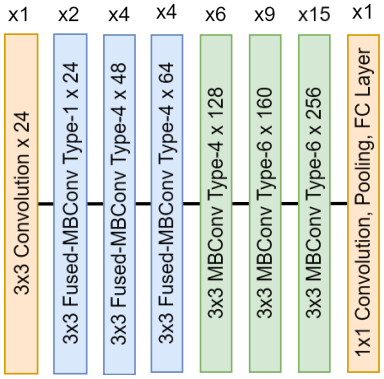

Figure 3. EfficientNetV2 architecture

3.2. Car Angle Classification

Images classified as Car are likely to be captured from various perspectives, resulting in differing viewpoints. To improve the accuracy of subsequent steps, Car images were classified according to the angles at which they appeared. In car angle classification, angles were grouped under eight classes: Front, Front-Left, Front-Right, Rear, Rear-Right, Rear-Left, Left, and Right. After classification, images with Left and Right labels were discarded, as they are unlikely to include front or back glass. Finally, as the main architecture, EfficientNetV2 was employed. Example images from discarded classes are shown in Figure 4.

Figure 4. Discarded images from "Left" and "Right" classes

3.3. Part Segmentation

In part segmentation, a specialized segmentation approach is applied for each angle identified in the previous step to isolate the glass region. Figure 5 shows each segmentation model, with angles that pair and the parts they cover.

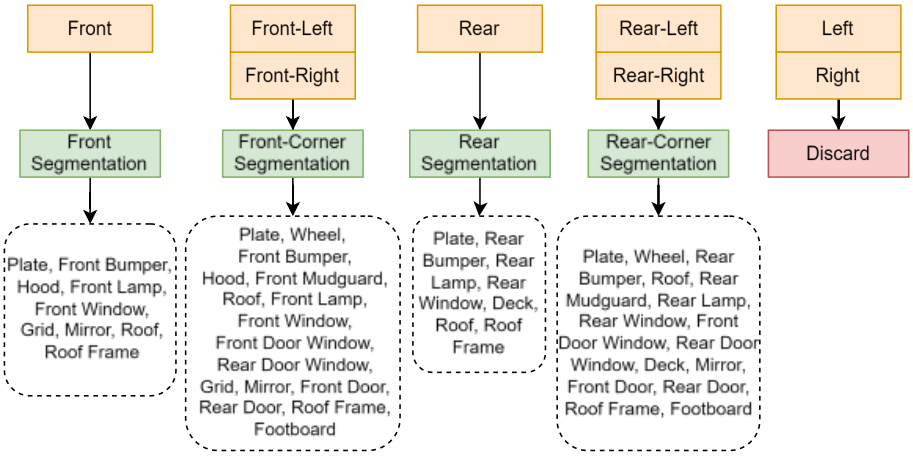

Figure 5. Segmentation models and angle pairs

As the segmentation architecture, U-Net [19] was selected. U-Net is an image segmentation approach that utilizes an encoder/decoder setup with skip connections. Encoder blocks extract high-level features while downsampling the image. Each encoder block consists of 2, 3×3 convolution layers. Each convolution layer is followed by a rectified linear unit activation function (ReLU). The output of each encoder block is connected to the corresponding decoder block with skip connections. At the end of the encoder block, a 2×2 max pooling operation is applied to reduce computational load. Encoder blocks reduce the resolution of the input image with repeated application of convolution operations, inadvertently leading to information loss. Skip connections transfer features from encoder blocks to decoder blocks, allowing U-Net to retain essential information. Decoder blocks upsample features from encoder blocks with deconvolution operation. Each decoder block consists of 2×2 deconvolution, merged with the skip connection from the matching encoder block. Then, 2, 3×3 convolution layers are used, followed by the ReLU function. Finally, 1x1 convolution is applied to the output of the last decoder block with the sigmoid activation function, generating the segmentation mask. U-Net architecture is shown in Figure 6.

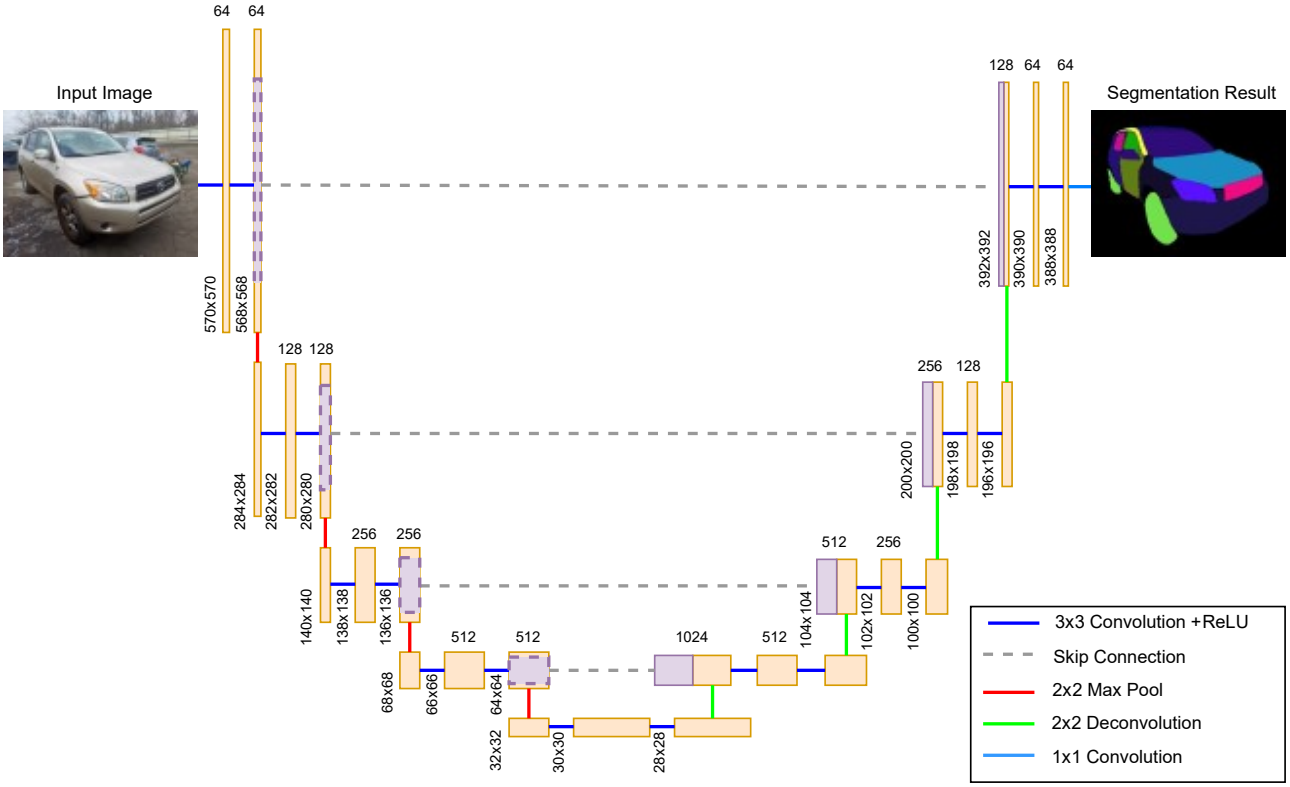

Figure 6. U-Net architecture

3.4. Damage Classification

The final step of the pipeline involves damage classification. Images classified as 'Close-up' are utilized directly, whereas for Car images, the glass regions cropped from the previous segmentation step were employed. In addition to EfficientNetV2, DenseNet [20], MobileNetV2 [21], ResNet50 [22], and BEiT were [23] evaluated to increase the diversity of classifiers and support the development of ensemble and fusion strategies for damage classification. DenseNet is an architecture that reuses features by connecting each layer to all the previous layers inside dense blocks. The dense design promotes cooperation between the layers, making the features interconnected. The interconnected layer design has multiple benefits besides feature-level collaboration: reduced risk of encountering vanishing gradient problems, parameter efficiency during training, and improved generalization performance. DenseNet begins with 7×7 convolution followed by 3×3 pooling, then four dense blocks with transitional layers between each are added. Finally, the architecture is concluded with 7×7 global average pooling and a fully connected layer. The DenseNet's architecture is shown in Figure 7.

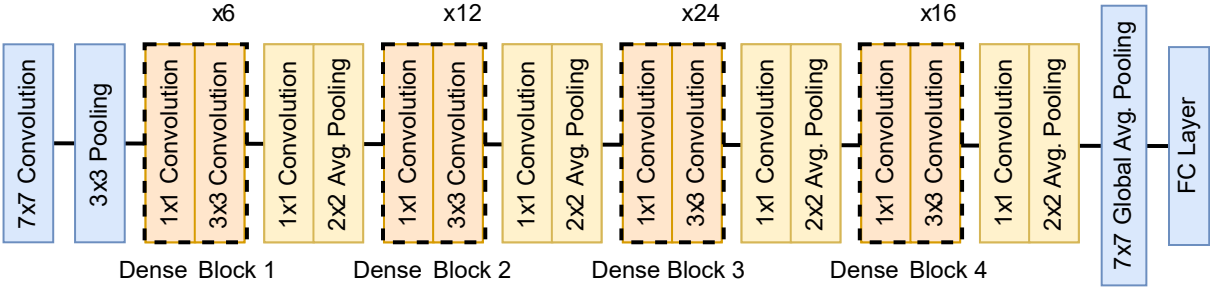

Figure 7. DenseNet architecture

MobileNet [24] is an architecture that employs depthwise separable convolutions to balance computational efficiency and accuracy. Compared to EfficientNet and EfficientNetV2, the differences between MobileNet and MobileNetV2 are more subtle, such as utilizing the swish activation function instead of ReLU. MobileNetV2 begins with a 2×2 convolution followed by seven bottleneck blocks; then, architecture is finished with 1×1 convolution, 7×7 average pooling, and 1×1 convolution, again. MobileNetV2's architecture is shown in Figure 8.

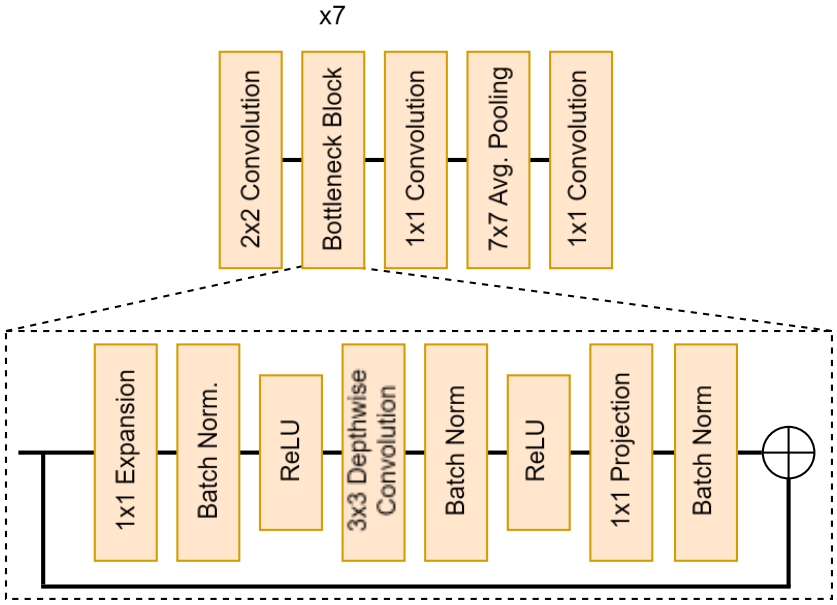

Figure 8. MobileNetV2 architecture

ResNet50 is a CNN-based architecture that employs residual blocks. Skip connections in residual blocks allow deeper networks to be established by allowing gradients to bypass layers, mitigating the vanishing gradient problem. ResNet50 begins with a 7×7 convolution followed by 3×3 max pooling. The architecture proceeds with four bottleneck blocks before finalizing with 1×1 average pooling and a fully connected layer. The architecture of ResNet50 is shown in Figure 9.

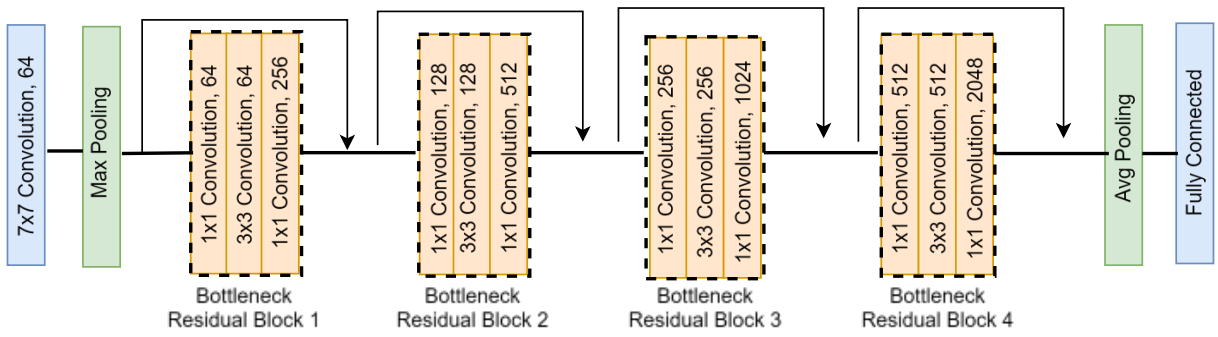

Figure 9. ResNet50 architecture

Bidirectional Encoder representation from Image Transformers (BEiT) is a transformer-based architecture that leverages cooperation between image patches and visual tokens. BEiT begins by dividing the image into patches and generating a matrix of visual tokens. These patches undergo further processing within the transformer architecture while the visual tokens reconstruct the image. Within the transformer framework, image patches are randomly masked to predict the corresponding visual tokens. The BEiT architecture is designed to capture both local and global contexts through this cooperative mechanism. By masking patches and predicting tokens, BEiT encourages the model to understand spatial relationships across the image. The architecture of BEiT is shown in Figure 10.

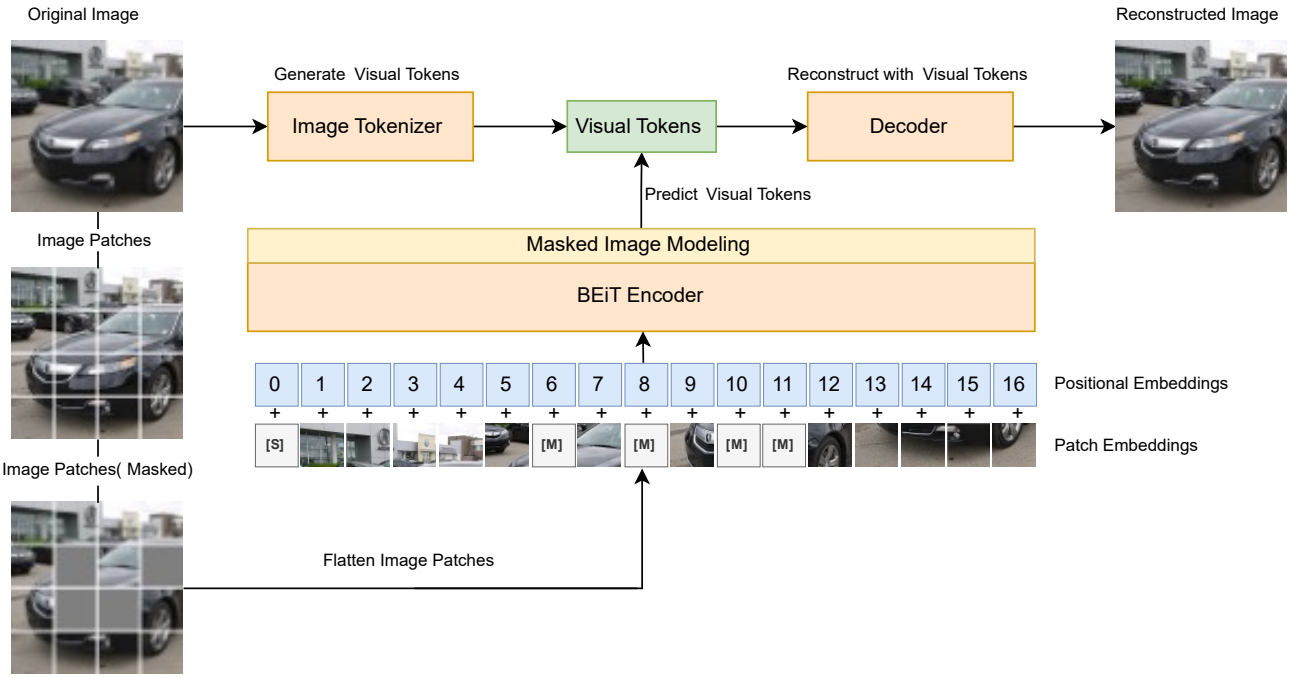

Figure 10. BEiT architecture

3.5. Ensemble

Ensemble methods constitute a strategy that integrates multiple learners in a complementary manner to enhance generalization performance. Although individually accurate, ensemble methods promote the effective collaboration of different architectures to leverage their strengths. The approach herein selected a weighted average of the learners as the primary ensemble strategy. Weighted averaging integrates the outputs of multiple learners, wherein each architecture's prediction is assigned a weight. The final prediction was calculated as the average of the individual learners' predictions. The optimal weights for averaging were determined using a grid search.

3.6. Fusion Strategy

Finally, along with the ensemble setup, a fusion strategy is employed to evaluate learners' collaboration at the feature level. Our fusion strategy is based on late feature extraction, which allows learners to develop fully before cooperating. Concatenation and summation were utilized as the fusion methods. Concatenation appends features together, whereas summation adds features in an element-wise manner. Finally, an attention layer was used to select informative features and suppress less desirable features before fusion. Resnet50, MobileNetV2, EfficientNetV2, and DenseNet were evaluated as architectures.

4. Dataset

All experiments were conducted on our internal insurance dataset, which consisted of damaged car images taken by insurance clients and experts. The following subsections explain the datasets used for each task in detail.

4.1. Car Filter Dataset

The car filter dataset consists of 33767 images. The dataset was split into three classes: Car, Close-up, and Not-Car. The class distributions are listed in Table 1.

Table 1. Distribution of images in the car filter dataset

Class	Train	Validation	Test	Total
Car	9571	1197	1196	11964
Close-up	6300	350	350	7000
Not-Car	10362	2221	2220	14803
Total	26233	3768	3766	33767

4.2. Car Angle Dataset

The car angle dataset consists of 113577 images. Dataset is split into eight classes: Front, Rear, Left, Right, Front-Right, Front-Left, Rear-right and Rear-Left. The class distributions are listed in Table 2. Example images for each class are shown in Figure 11.

Table 2. Distribution of images in the car angle dataset

Class	Train	Validation	Test	Total
Front	13082	4185	400	17667
Front-Right	12962	2634	400	15996
Front-Left	13364	3022	400	16786
Rear	11368	2110	400	13878
Rear-Right	10835	2344	400	13579
Rear-Left	11433	2335	400	14168
Left	9856	780	400	11036
Right	9297	770	400	10467
Total	92197	18180	3200	113577

Figure 11. Example images in the car angle dataset: (**a**) Front (**b**) Rear (**c**) Front-Left (**d**) Rear-Left (**e**) Left (**f**) Front-Right (**g**) Rear-Right (**h**) Right

4.3. Car Part Segmentation Dataset

The car part segmentation dataset consisted of 1316 images. Dataset is split into four classes: Front, Front-Corner, Rear, Rear-Corner. The class distributions are listed in Table 3. Example images for each class are shown in Figure 12.

Table 3. Distribution of images in the car part segmentation dataset

Angle	Train	Validation	Test	Total
Front	259	30	29	318
Front-Corner	285	36	35	356
Rear	221	26	25	272
Rear-Corner	290	40	40	370
Total	1055	132	129	1316

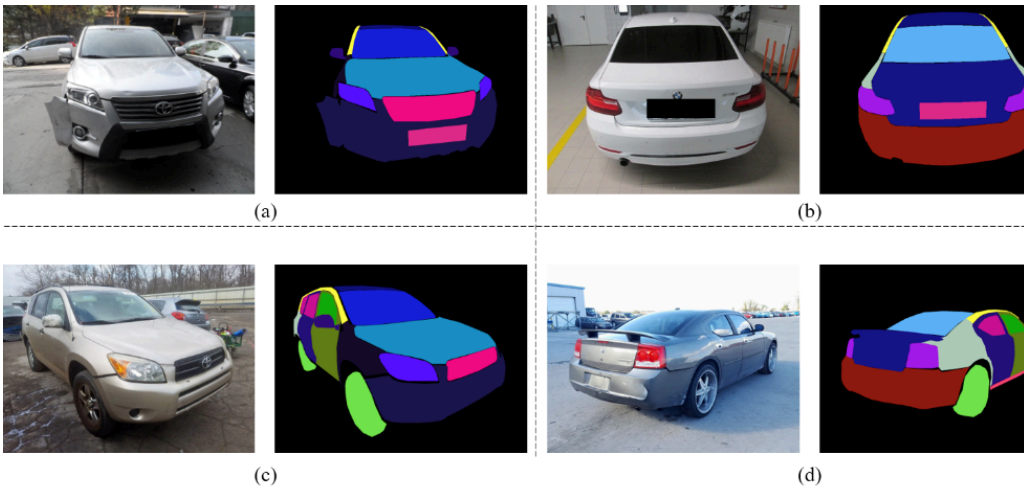

Figure 12. Example images in the car part segmentation dataset. The image and its mask are shown side by side. Each image is an example for a class. (**a**) Front, (**b**) Rear, (**c**) Front-Corner, (**d**) Rear-Corner

4.4. Car Glass Damage Dataset

The car glass damage dataset consisted of 21,530 images. The dataset was split into five classes: No-Damage, No-Glass, Repair, Replace and Other. The class distributions are listed in Table 4. Examples of each class are presented in Figure 13.

Table 4. Distribution of images in the car glass damage dataset

Class	Train	Validation	Test	Total
No-Damage	4116	459	680	5255
No-Glass	1889	118	207	2214
Repair	662	100	396	1158
Replace	4760	327	488	5575
Other	5122	876	1330	7328
Total	16549	1880	3101	21530

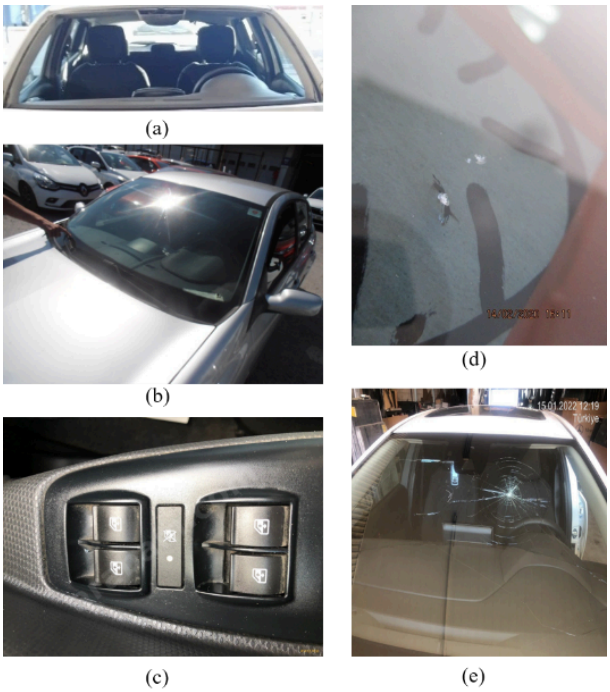

Figure 13. Example images in the car glass damage dataset. (**a**) No-Glass, (**b**) No-Damage, (**c**) Other, (**d**) Repair, (**e**) Replace

5. Results and Discussion

5.1. Experimental Setup

The computer configuration used for training and testing the DL models was the Intel Core i9-9900KF (3.60 GHz) processor, 64 GB DDR4 RAM, and a single RTX 2080TI graphics card. Experiments were conducted using the TensorFlow framework.

5.2. Experiments

Experiments were conducted for each phase of the damage classification process. The results of the damage classification stage are presented in Table 5.

Table 5. Results for stages preceding damage classification

Task	Precision	Recall	F1 Score	Accuracy	IoU
Car Filter	0.91	0.95	0.93	0.96	-
Car Angle	0.86	0.86	0.86	0.86	-
Front Segmentation	-	-	-	0.94	0.72
Front-Corner Segmentation	-	-	-	0.94	0.73
Rear Segmentation	-	-	-	0.97	0.73
Rear-Corner Segmentation	-	-	-	0.93	0.72

The car filter and angle classification stages achieved F1 scores of 0.93 and 0.86, respectively, providing reliable inputs for the subsequent phases of the pipeline. The segmentation models' accuracies and IoUs were comparable. ResNet50, EfficientNetV2 (b2 and b7), MobileNetV2, DenseNet, and BEiT were used for damage classification. The damage classification results are listed in Table 6.

Table 6. Results for the damage classification model

Architecture	Input Size	Precision	Recall	F1 Score	Accuracy	Inference Time
ResNet50	224	0.85	0.84	0.84	0.87	0.05
EfficientNetV2 (b2)	480	0.88	0.88	0.88	0.90	0.08
EfficientNetV2 (b7)	224	0.85	0.86	0.85	0.88	0.14
MobileNet	224	0.85	0.84	0.84	0.87	0.03
MobileNetV2	224	0.85	0.84	0.84	0.87	0.04
DenseNet	480	0.87	0.87	0.87	0.89	0.12
BEiT	256	0.86	0.85	0.86	0.88	0.08
BEiT	384	0.85	0.86	0.85	0.88	0.30
BEiT	512	0.89	0.88	0.88	0.90	1.70

ResNet50, EfficientNetV2 (b7), MobileNet, and MobileNetV2 demonstrated inferior performance in the F1 score relative to the alternatives. Except for EfficientNetV2 (b7), the architectures demonstrated the shortest individual inference times. EfficientNetV2 (b2), DenseNet, and BEiT (256×256) achieved strong F1 scores, with their respective inference times showing minimal deviation compared with architectures using a 224×224 input size. Unfortunately, BEiT (384×384) and BEiT (512×512) did not achieve an optimal trade-off between F1 score and inference time.

In our ensemble setup, EfficientNetV2 (b2), DenseNet, and BEiT (256×256) were initially evaluated because of their optimal balance between F1 score and inference time. The results of the ensemble strategy are listed in Table 7.

Table 7. Ensemble results

Architecture	Precision	Recall	F1 Score	Accuracy	Inference Time
EfficientNetV2 (b2), DenseNet, BEiT	0.89	0.89	0.89	0.91	0.77
EfficientNetV2 (b2), DenseNet, MobileNetV2	0.89	0.89	0.89	0.91	0.24

The EfficientNetV2 (b2), DenseNet, and BEiT (256×256) achieved high F1 scores throughout the experiments. However, because of the relatively high inference time of the setup and considering the potential application of the proposed approach to mobile devices, which is widely favored by experts and customers in insurance claims, BEiT was replaced with MobileNetV2 to develop an alternative solution. Interestingly, the combination of EfficientNetV2(b2), DenseNet, and MobileNetV2 matches the performance of the previous setup, yielding the same F1 score. Finally, EfficientNetV2(b2), DenseNet, MobileNetV2, and ResNet50 were evaluated for feature fusion using concatenation, summation, and an attention layer. The results of the fusion strategy are listed in Table 8.

Table 8. Fusion results

Architecture	Fusion Method	Input Size	Dense Layer	F1 Score	Inf. Time
EfficientNetV2(b2), DenseNet	Concat.	480	64	0.87	0.14
EfficientNetV2(b2), DenseNet, MobileNetV2	Concat.	224	64	0.79	0.11
EfficientNetV2(b2), DenseNet, MobileNetV2	Concat.	480	64	0.86	0.16
EfficientNetV2 (b2), DenseNet, ResNet50	Concat.	480	64	0.82	0.25
EfficientNetV2 (b2), DenseNet, ResNet50	Concat.	480	128, 64	0.71	0.25
EfficientNetV2 (b2), DenseNet, ResNet50	Sum.	480	128, 64	0.82	0.25
EfficientNetV2 (b2), DenseNet, ResNet50	Attention + Concat.	480	128, 64	0.82	0.25
EfficientNetV2 (b2), DenseNet, ResNet50	Attention + Concat.	480	64	0.82	0.25
EfficientNetV2 (b2), DenseNet, ResNet50	Attention + Sum.	480	64	0.80	0.25
EfficientNetV2 (b2), DenseNet, MobileNetV2	Attention + Concat.	480	256,128,64	0.84	0.16
EfficientNetV2 (b2), DenseNet, MobileNetV2	Attention + Sum.	480	256,128,64	0.80	0.17

Feature fusion did not exceed the F1 score obtained from the ensemble configurations. EfficientNetV2 (b2) and DenseNet individually demonstrated equal or superior performance in the F1 score and inference time compared to their combined performance through concatenation. Concatenation produced superior results compared to other techniques, suggesting that the selective mechanism of the attention layer did not perform as effectively as anticipated. In our approach, the ensemble of EfficientNetV2 (b2), DenseNet, and MobileNetV2 emerged as the most optimal solution for damage classification because the marginal increase in inference time is negligible.

6. Conclusion

In this study, we proposed a DL-based framework for detecting car glass damage, an essential task in processing insurance claims. In pursuit of an optimal solution for damage classification, we investigated various methodologies, including weighted average ensemble techniques and late feature fusion. Finally, we evaluated our solutions regarding the F1 score and inference time. The experimental results showed that EfficientNetV2 (b2), DenseNet, and MobileNetV2 are the optimal solutions for mobile-intensive environments in insurance claim processing. In the future, researchers can generate examples of synthetic car glass damage using the synthetic image-generation approach to model the damaged area and its surrounding pixels to build new examples.

Author Contributions

The first author: Conceptualization, Methodology, Software, Data curation, Writing - Original Draft. The second author: Methodology, Writing - Original Draft, Writing - Review and Editing. The third author: Methodology, Writing - Original Draft, Writing - Review and Editing. The fourth author: Writing - Review and Editing, Supervision. They all read and approved the final version of the paper.

Conflicts of Interest

All the authors declare no conflict of interest.

Ethical Review and Approval

No approval from the Board of Ethics is required.

References

[1] P. M. Kyu, K. Woraratpanya, *Car damage detection and classification*, in: Proceedings of the 11th International Conference on Advances in Information Technology, Bangkok, 2020, pp. 1–6.

[2] M. Parhizkar, M. Amirfakhrian, *Car detection and damage segmentation in the real scene using a deep learning approach*, International Journal of Intelligent Robotics and Applications 6 (2) (2022) 231–245.

[3] A. Guha, D. Samanta, *Hybrid approach to document anomaly detection: An application to facilitate RPA in title insurance*, International Journal of Automation and Computing 18 (1) (2021) 55–72.

[4] B. Benedek, C. Ciumas, B. Z. Nagy, *Automobile insurance fraud detection in the age of big data – a systematic and comprehensive literature review*, Journal of Financial Regulation and Compliance 30 (4) (2022) 503–523.

[5] H. Xia, Y. Zhou, Z. Zhang, *Auto insurance fraud identification based on a CNN-LSTM fusion deep learning model*, International Journal of Ad Hoc and Ubiquitous Computing 39 (1-2) (2022) 37.

[6] R. Van Ruitenbeek, S. Bhulai, *Convolutional neural networks for vehicle damage detection*, Machine Learning with Applications 9 (2022) 100332.

[7] K. Patil, M. Kulkarni, A. Sriraman, S. Karande, *Deep learning based car damage classification*, in: 2017 16th IEEE International Conference on Machine Learning and Applications, Cancun, 2017, pp. 50–54.

[8] L. Li, K. Ono, C.-K. Ngan, *A deep learning and transfer learning approach for vehicle damage detection*, The International FLAIRS Conference Proceedings 34 (1) (2021) 6 pages.

[9] T. Ojala, M. Pietikäinen, D. Harwood, *A comparative study of texture measures with classification based on featured distributions*, Pattern Recognition 29 (1) (1996) 51–59.

[10] A. Ramirez Rivera, J. Rojas Castillo, O. Oksam Chae, *Local directional number pattern for face analysis: Face and expression recognition*, IEEE Transactions on Image Processing 22 (5) (2013) 1740–1752.

[11] Q. Zhang, X. Chang, S. B. Bian, *Vehicle-damage-detection segmentation algorithm based on improved mask RCNN*, IEEE Access 8 (2020) 6997–7004.

[12] D. Mallios, L. Xiaofei, N. McLaughlin, J. M. D. Rincon, C. Galbraith, R. Garland, *Vehicle damage severity estimation for insurance operations using in-the-wild mobile images*, IEEE Access 11 (2023) 78644–78655.

[13] M. Dwivedi, H. S. Malik, S. N. Omkar, E. B. Monis, B. Khanna, S. R. Samal, A. Tiwari, A. Rathi, *Deep learning-based car damage classification and detection*, in: N. N. Chiplunkar, T. Fukao (Eds.), Advances in Artificial Intelligence and Data Engineering, Vol. 1133 of *Advances in Intelligent Systems and Computing*, Springer Nature Singapore, Singapore, 2021, pp. 207–221.

[14] J. Redmon, A. Farhadi, YOLOv3: An incremental improvement (2018), https://arxiv.org/abs/1804.02767, Accessed 3 Mar 2025.

[15] X. Wang, W. Li, Z. Wu, *CarDD: A new dataset for vision-based car damage detection*, IEEE Transactions on Intelligent Transportation Systems 24 (7) (2023) 7202–7214.

[16] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable convolutional networks (2017), https://arxiv.org/abs/1703.06211, Accessed 3 Mar 2025.

[17] M. Tan, Q. V. Le, EfficientNetV2: Smaller models and faster training (2021), https://arxiv.org/abs/2104.00298, Accessed 3 Mar 2025.

[18] M. Tan, Q. V. Le, EfficientNet: Rethinking model scaling for convolutional neural networks (2019), https://arxiv.org/abs/1905.11946, Accessed 3 Mar 2025.

[19] O. Ronneberger, P. Fischer, T. Brox, *U-Net: Convolutional networks for biomedical image segmentation*, in: N. Navab, J. Hornegger, W. M. Wells, A. F. Frangi (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Vol. 9351 of *Lecture Notes in Computer Science*, Springer International Publishing, Cham, 2015, pp. 234–241.

[20] G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger, Densely connected convolutional networks (2016), https://arxiv.org/abs/1608.06993, Accessed 3 Mar 2025.

[21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: Inverted residuals and linear bottlenecks (2018), https://arxiv.org/abs/1801.04381, Accessed 3 Mar 2025.

[22] K. He, X. Zhang, S. Ren, J. Sun, *Deep residual learning for image recognition*, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 2016, pp. 770–778.

[23] H. Bao, L. Dong, S. Piao, F. Wei, BEiT: BERT pre-training of image transformers (2021), https://arxiv.org/abs/2106.08254, Accessed 3 Mar 2025.

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: Efficient convolutional neural networks for mobile vision applications (2017), https://arxiv.org/abs/1704.04861, Accessed 3 Mar 2025.