Car Damage Detection AI Project
SE3508

Umut Can ORAL / 210717030
Can Deniz SALMAN / 220717045

June 2025

Contents

1 Introduction

In this project, we present a deep learning-based system for automatic car damage detection and classification. The objective is to design a multitask learning architecture that can identify both the type of car damage and the view angle (front or rear) from an image. We experimented with two state-of-the-art convolutional neural networks: ResNet50 and DenseNet121.

2 Dataset Preparation

2.1 Source and Structure

The original dataset was sourced from Kaggle (samwash94/comprehensive-car-damage-detection). It initially consisted of 6 classes:

- F_Normal, F_Crushed, F_Breakage
- R_Normal, R_Crushed, R_Breakage

Each class name encodes two different attributes: the view of the image (F=Front, R=Rear) and the type of damage (Normal, Crushed, Breakage).

2.2 Multitask Label Reformulation

To better generalize and simplify the problem, we reformulated the dataset as a multitask classification problem with two labels:

1. **Damage Type:** {Normal, Crushed, Breakage} (3 classes)
2. **View Type:** {Front, Rear} (2 classes)

This approach has two major advantages:

- It reduces the complexity of the output space from 6 to a combination of 3 + 2 classes.
- It allows the model to independently learn two orthogonal features: damage and perspective.

The class splitting logic and label assignment were handled through a custom PyTorch Dataset class named MultiTaskCarDataset.

2.3 Data Augmentation

To mitigate overfitting and improve model generalization, we employed a variety of data augmentation techniques:

- RandomHorizontalFlip
- RandomRotation(±10 degrees)
- ColorJitter (brightness, contrast, saturation)
- RandomResizedCrop to 128x128

2.4 Dataset Split

We used a stratified 80/20 split:

- 80% for training
- 20% for validation

3 Model Architectures

3.1 ResNet50

The ResNet50 model was modified to include two classification heads:

- One head for damage prediction
- One head for view angle prediction

Dropout and linear layers were added after the backbone to build the dual-output model.

3.2 DenseNet121

Similarly, we used DenseNet121 as a second architecture. Its dense connectivity structure allowed for better feature reuse. The model was adapted with two fully connected output heads. DenseNet121 also demonstrated superior generalization in detailed damage classification, benefiting from its capacity to preserve and combine low-level features. Furthermore, DenseNet121 consistently produced better results in both tasks compared to ResNet50 and was selected as the final model for integration into the user interface (UI) system.

4 Training Procedure

- **Loss:** CrossEntropyLoss (separately for damage and view)
- **Optimizer:** Adam with learning rate scheduling
- **Batch Size:** 32
- **Early Stopping:** Patience = 3
- **Epochs:** 30

Training and validation losses were tracked separately for both outputs.

5 Evaluation and Results

5.1 Metrics

Classification reports were generated for both tasks. Performance was measured using accuracy, precision, recall, and F1-score. DenseNet121 showed a slight improvement over ResNet50.

Table 1: Damage Classification Accuracy Comparison

Model	Damage Accuracy	View Accuracy
ResNet50	74%	95%
DenseNet121	93%	98%

5.2 Confusion Matrices

The following confusion matrices were plotted to analyze model behavior:

- **Damage Confusion Matrix**

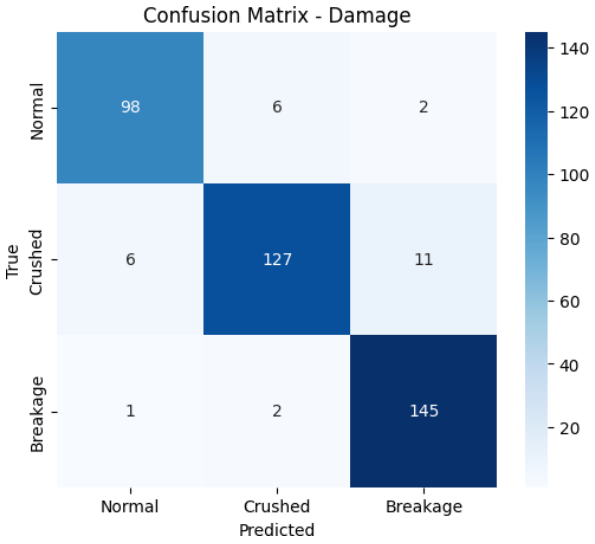

Figure 1: Confusion Matrix for Damage Classification

- **View Confusion Matrix**

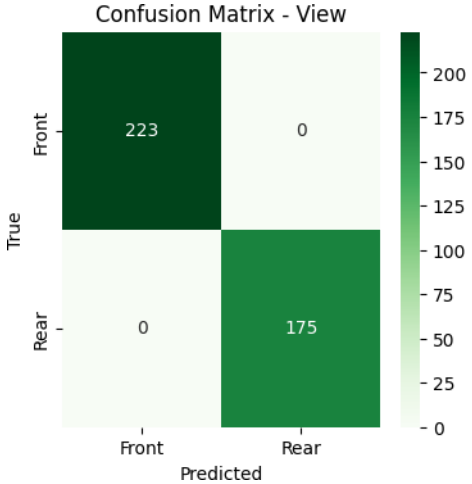

Figure 2: Confusion Matrix for View Classification

5.3 Loss Curve

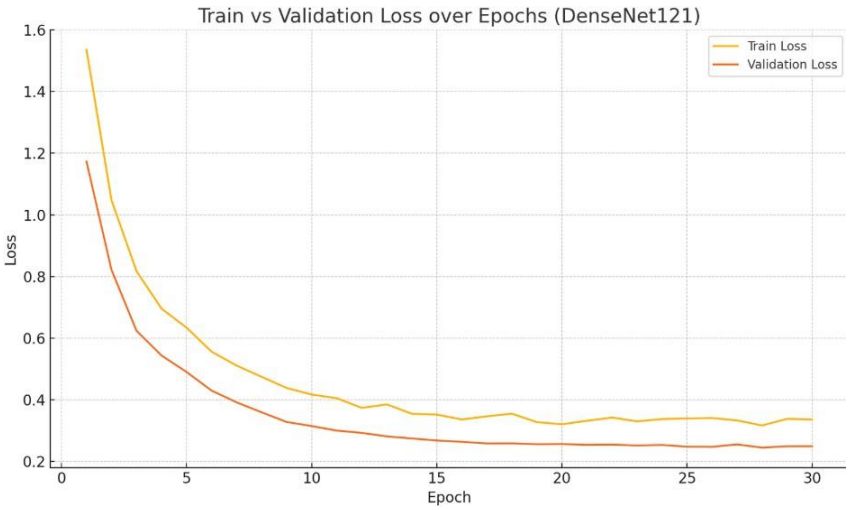

Figure 3: Train vs Validation Loss for DenseNet121

6 Challenges Faced

6.1 Overfitting

During the initial training phase, the models achieved very high training accuracy but significantly lower performance on the validation set — a classic indication of overfitting. This was primarily due to the relatively limited size and variability of the dataset, which caused the models to memorize specific training samples instead of learning generalized features.

To mitigate this issue, several strategies were implemented:

- nhanced Data Augmentation: Random horizontal flips, rotations, and color jitter transformations were applied during training to artificially increase the diversity of input data. These transformations helped the models to better generalize by introducing spatial and photometric variations.

- Dropout Regularization: Dropout layers were added to the fully connected heads of both ResNet50 and DenseNet121. This technique randomly deactivates a fraction of neurons during each training iteration, forcing the model to learn redundant and robust representations rather than relying on specific neuron paths.

- Dropout Regularization: Dropout layers were added to the fully connected heads of both ResNet50 and DenseNet121. This technique randomly deactivates a fraction of neurons during each training iteration, forcing the model to learn redundant and robust representations rather than relying on specific neuron paths.

6.2 Dataset Issues

Several challenges were encountered in the dataset that directly impacted model performance and training stability:

- Class Imbalance: Although the dataset included six predefined classes (e.g., front_crushed, rear_normal), the distribution across these classes was not uniform.
    - Low-Resolution and Noisy Images: Some of the images in the dataset were of low visual quality, either due to compression artifacts, motion blur, or suboptimal lighting. These issues limited the models' ability to detect fine-grained damage patterns, especially in borderline cases between normal and crushed conditions.
    - Inconsistent Labeling: In several instances, images that were visually similar were placed under different folder labels, or vice versa. For example, mildly damaged vehicles were sometimes labeled as normal and other times as crushed. These inconsistencies introduced noise into the learning process, making it harder for the model to learn a clear decision boundary.

7 Conclusion

By converting a 6-class flat classification into a dual-label multitask problem, we were able to improve the performance and generalization of the models. DenseNet121 performed best overall, and the trained models were exported for integration into a downstream UI system.