Final Report: Car Damage Detection and Classification

Shangyu Chen Yuqi Yan Jake Beinart Andrew Zhao

Abstract

The final report presents several models for determining vehicle damage and classifying the region of the primary damaged components. We evaluated conventional pre-trained neural network classification models, and conducted an analysis of their distinct performance characteristics. Additionally, we discuss the limitations of the current models and outline our future research endeavors.

1. Introduction

Our project is driven by the need for accurate vehicle damage assessment in the fleet management and insurance industries. Leveraging our team's expertise in data science and machine learning, we aim to develop robust models using curated datasets from platforms like Kaggle.

Our approach is innovative in several ways. We move beyond binary classification to include multiple classification tasks, categorizing damages into frontal, side, and rear categories for deeper insights. We integrate deep learning models such as DenseNet121, MobileNet-v2, ResNet-50, and CLIP to improve classification accuracy and generalization. Additionally, advanced pre-processing techniques are employed to optimize model performance and avoid overfitting.

Our project comprises two primary tasks: binary classification to distinguish between non-damaged and damaged cars, and multiple classification to identify the type and location of damages (front, side, rear). For the binary classification task, we compare the performance of DenseNet121, MobileNet-v2, ResNet-50, and CLIP against a baseline model. We conduct rigorous evaluations and metrics comparison to assess the models' effectiveness in accurately classifying vehicle damages.

In the multiple classification task, we utilize ResNet-50 and CLIP models to categorize damages into different classes. Additionally, we manually relabel datasets to ensure high-quality training data for multi-class classification, emphasizing the importance of data relevance and accuracy in our methodologies.

2. Related Work

2.1. MobileNet-v2

MobileNet-v2 [5] is a neural network architecture designed for mobile and embedded devices, with a focus on efficiency and speed. Also, depending on the size and number of our dataset, MobileNet-v2 is a good choice.

In contrast to the deeply separable convolution used in MobileNet-v1, MobileNet-v2 employs inverted residuals and linear bottlenecks. And more than ResNet, MobileNet-v2 has some of the same mechanisms to prevent gradient descent as ResNet, but reduces the computational complexity.

2.2. DenseNet121

DenseNet121 is a deep learning model that performs well in tasks such as image classification due to its unique structure where each layer is connected to every other layer. This helps the model retain important information throughout the network and prevents data from being lost in transit (Figure 1).

Combining migration learning and data augmentation can significantly improve the performance of DenseNet 121, especially when only a small number of training samples are available. Migration learning allows a model to utilize knowledge gained from models previously trained on large datasets to help it perform well on new, smaller datasets [2]. Data augmentation increases the diversity of the training data through techniques such as rotating or scaling images, which helps the model generalize better and avoid overfitting.

In summary, DenseNet 121 combines with these techniques to provide a powerful solution for image analysis, even when data is limited [1]. This makes it very effective in a variety of applications such as medical imaging and real-time analytics.

2.3. ResNet50

ResNet is a CNN architecture that has been foundational to modern computer vision. Although it has been around for years, continues to be a popular choice for a range of tasks. The network is composed of convolutional residual blocks that make use of "shortcut" connections to allow training of

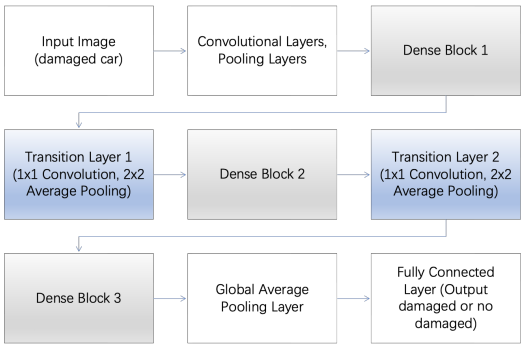

Figure 1. Structure of DenseNet 121 model

much deeper networks and avoiding the vanishing gradient problem. The version used in our project (ResNet-50) provides a good balance between depth and performance and makes use of a deeper residual block called a bottleneck block.

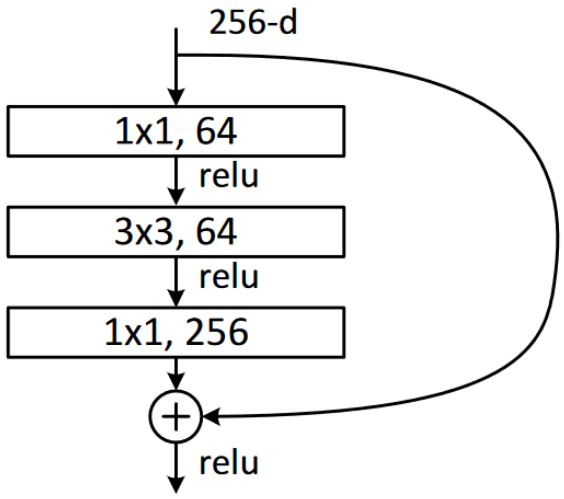

Figure 2. Bottleneck block used in ResNet-50 [3]

One reason for ResNet's continued use is how long it has been in favor, and thus many pre-trained options exist to take advantage of transfer learning. The existence of pre-trained, relevant, and numerous weights was a big factor in guiding our use of ResNet50 in this project.

2.4. CLIP

CLIP (Contrastive Language–Image Pre-training) [4], developed by OpenAI, represents a significant advancement in the field of AI, enabling models to understand images in conjunction with natural language descriptions. This technology leverages a large-scale dataset of images paired with textual descriptions to learn visual concepts from natural language supervision.

CLIP's training approach is based on contrastive learning, where the model is trained to predict which caption from a set of captions matches a given image. This is achieved through a dual-encoder architecture, consisting of a text encoder and an image encoder. The text encoder processes textual descriptions into embeddings, while the image encoder transforms images into visual embeddings. The goal is for the embeddings from corresponding text-image pairs to be closer in the embedding space compared to non-matching pairs.

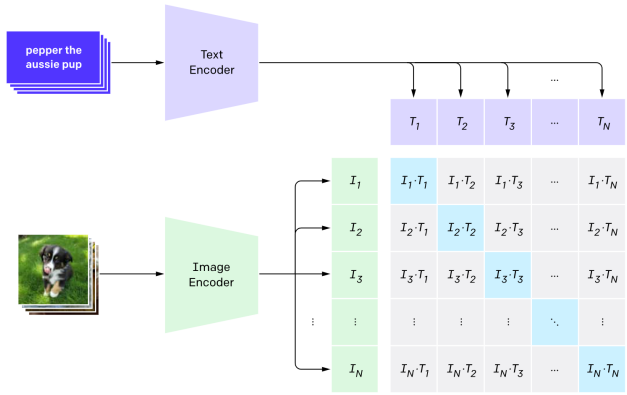

Figure 3. CLIP Model

In our project, we will use the CLIP model to compare its performance with other models. We believe this captures the difference between pre-trained large models and fine-tuned classical models.

3. Approach

3.1. Data Augmentation

We randomly replicated 20% of the data in the training set and subjected them to the following operations with a probability of 50% each: (1) Gaussian blurring, (2) cropping, (3) rotation, and (4) flipping.

3.2. Dataset Splitting

The Kaggle dataset has been divided into training set and test set. In order to ensure the stability of the training results, we fixed the random number seed and divided the training set into training set and test set again with the ratio of 4:1, so that we could find the best model in the training epoch.

The number of datasets is shown in the table:

	Training	Validating	Testing
Damaged	736	184	230
Whole	736	184	230

3.3. Binary Classification

For binary classification, we first define our own artificial CNN as baseline, and then select some existing classifica-

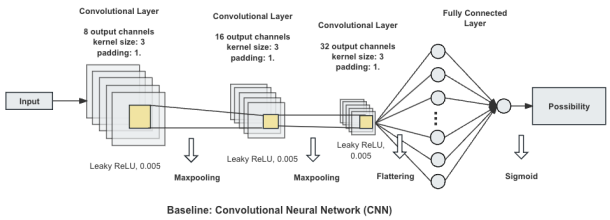

Figure 4. Baseline Model

tion models - MobileNet-v2, DenseNet121 and ResNet50.

3.3.1 Baseline Model

As with homework, for a common binary image classification problem, we typically build a convolutional neural network (CNN) to train and classify. The structure shows as Figure 4.

For the input, we set 3 convolution layers with leaky ReLu as activation function. After this, we use fully connected layer and sigmoid function to calculate the possibility that the car is damaged.

3.3.2 MobileNet-v2

We use pretrained MobileNetV2. For the MobileNet-v2 building, we basically used the structure of the original MobileNet-v2, but after parameter scanning, we set the rate of the dropout to 0.15 and set two outputs for each input, and the scores of the two outputs were compared to determine if the vehicle was damaged.

After our scanning, we use cross entropy loss and use Adam as optimizer (learning rate is set to be 0.001) for 20 epochs of training and take the epoch with maximum validation accuracy as the result.

Among 460 test dataset, our final accuracy with MobileNet-v2 was 94.78%.

3.3.3 DenseNet121

To accommodate binary classification, we replaced the final fully connected layer with one that has two output neurons and set a discard rate of 0.05 to minimize overfitting. In addition, we initialized the model with pre-trained weights to leverage existing knowledge, accelerate early learning, and improve performance.

During training, we used a cross-entropy loss function and a stochastic gradient descent (SGD) optimizer with a learning rate of 0.001 and a momentum of 0.9. Momentum helps the optimizer to converge in the right direction faster and reduces oscillations during the training process, which improves the efficiency and stability of the model. Throughout the training process, the model was validated on an independent test set to evaluate its performance on unseen data to ensure good generalization.

The model performed well in the tests. The accuracy of the model was consistently 94.57% in independent tests.

3.3.4 ResNet50

We opted to use ResNet50 weights from PyTorch pre-trained on the ImageNet-1k dataset (IMAGENET1K_V2). The dataset contains several classes that make it attractive for transfer learning with respect to our vehicle-related task: sports cars, passenger cars, and trucks are all represented individually.

Several transforms were used on the data to combat overfitting and take advantage of the pre-trained weights: random crop, random horizontal/vertical flips, and normalization to the ImageNet mean and standard deviation.

Several hyperparameter options were also explored, but SGD for the optimizer, a learning rate of 0.001, and a batch size of 32 produced the best performance, though other configurations also produced a similar classification accuracy on the validation set.

Our final accuracy with ResNet50 was 93.91%.

3.3.5 CLIP

We also use CLIP to classify vehicle images as either damaged or intact. Images are encoded using a dual-encoder framework—one for text and one for images. This setup enables the model to compare image embeddings against text embeddings derived from descriptions like ”damaged car” and ”whole car.”

The evaluation results indicated a distinct difference in classification accuracy between damaged and whole cars. The model achieved an accuracy of 64.43% for damaged vehicles and 99.91% for whole vehicles, culminating in an overall accuracy of 82.17%. These results highlight the effectiveness of CLIP in distinguishing between severely differing conditions but also suggest a need for further tuning to improve the detection of subtle damages.

After this, we altered the textual descriptions associated with each class. For damaged cars, we shifted from a generic label to a more expressive description: ”damaged car, auto-destructive art.” For undamaged cars, the label was changed to ”a car, clean, precisionism.” These descriptions were designed to emphasize specific traits and contexts that enhance the distinctiveness recognized by the model. We achieved notable improvements: accuracy for damaged cars increased to 91.91% and for undamaged cars to 93.39%, with an overall accuracy rising to 92.65%. These results demonstrate that precise, evocative language can greatly influence CLIP’s ability to discern between nuanced visual categories, emphasizing the importance of tailored textual descriptions in training visual models.

3.4. Detailed part classification

Based on the performance of the above models, we find that the three traditional pre-trained neural networks perform similarly.

In binary classification, three pretrained models all demonstrated good performance. We selected ResNet50 for further modifications to meet the requirements of multiclass classification. Since the original dataset did not include assessments of specific vehicle damage locations, we manually labeled the vehicle location damages.

Based on Section 3.3's ResNet50, we made the following modifications: the dropout rate was retained, hyperparameter scanning was conducted, and the output was changed to three units passed through a sigmoid function. Additionally, we utilized the CLIP model for comparison.

We compared three metrics: (1) the accuracy of predictions for each specific part; (2) whether the damage to each part was accurately predicted for a single image; (3) whether the main damage part was correctly predicted, meaning whether the part with the highest prediction score matched the labeled damage location. Finally, we plotted the ROC curve based on changes in the threshold.

4. Results

4.1. Binary Classification

For the binary classification problem, the performance of each model is illustrated in the confusion matrices as shown in Figure 5, 6, 7 and 8.

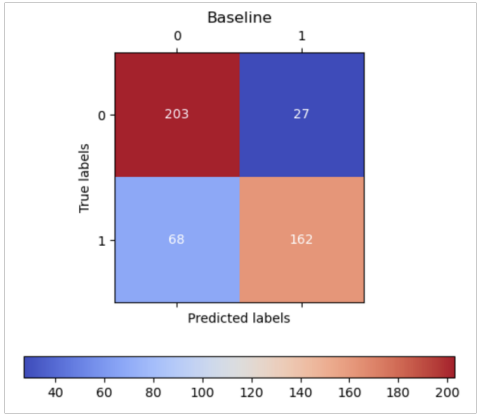

Figure 5. Baseline Model Performance

Compare the accuracy, shown on Table 1

For the pretrained, mature models, they all perform well in solving this classification problem.

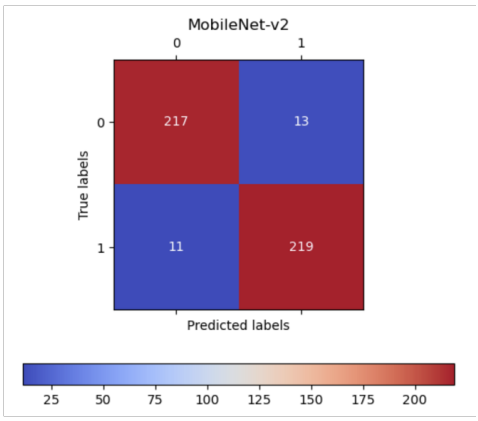

Figure 6. MobileNet-v2 Performance

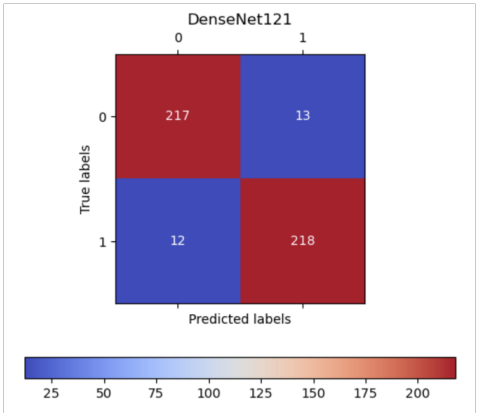

Figure 7. DenseNet121 Performance

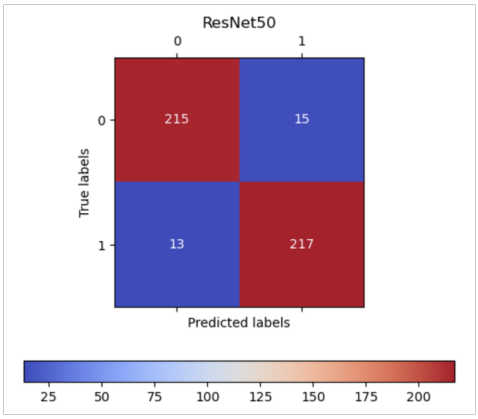

Figure 8. ResNet50 Performance

4.2. Detailed part classification

4.2.1 ResNet50

We set the threshold at 0.5, and out of a total of 460 data points, 330 were completely accurate in predicting whether each part of the car was damaged, achieving an accuracy

Model	Test Accuracy	Best Val. Acc.
Baseline	79.35%	74.20%
MobileNet-v2	94.78%	90.43%
DenseNet121	94.57%	97.07%
ResNet50	93.91%	94.95%
CLIP	82.17%	/

Table 1. Accuracy for binary classification

rate of 65.87%. The average accuracy rates for predictions of the three individual parts were 84.8%, while the accuracy for predicting the primary damaged part in the multiclass classification problem reached 72.9%.

The ROC plot is shown on Figure 9. Based on the curvature of the ROC curve, the model still performs well overall.

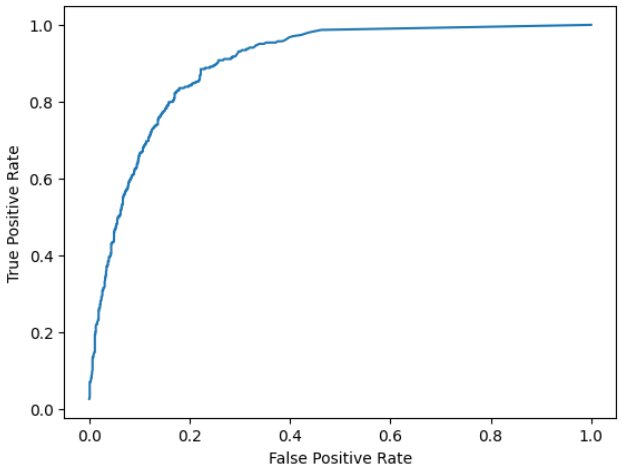

Figure 9. ROC Plot for ResNet50 predicting

4.2.2 CLIP

Performance by Damage Type:

- Front Damage: The model achieved an accuracy of 66.26% in identifying front damage. This indicates a moderate level of effectiveness, suggesting that while the model can generally recognize front damage, there is room for improvement

- Side Damage: With an accuracy of 58.35%, the model's performance on side damage is the lowest among the three types. This lower accuracy could be attributed to the possible complexity and less distinctive features of side damages as captured in images, or insufficient training data representing various forms of side damage.

- Back Damage: The model performed best in identifying back damage, with an accuracy of 71.91%. This suggests that the model is more adept at recognizing features associated with back damage or that the textual descriptions used for back damage align well with the visual cues present in the images. This high performance relative to other categories might indicate more consistent or distinctive features that are easier for the model to learn.

The overall accuracy for all types of damage combined is 40.00%. This relatively low overall accuracy indicates that while the model can identify specific types of damage with reasonable accuracy, its ability to accurately classify all three types in the same image is limited. This could be due to overlapping features between different damage types or challenges in the model's capacity to simultaneously associate multiple textual descriptions with a single image effectively.

5. Discussion and Conclusions

While we had some success using CLIP for binary classification, we noticed that results could vary wildly based on choosing the right prompt for the classes. The table below shows how even small changes to the prompt can swing the results. CLIP is pre-trained, but trying to semantically fit the best class labels becomes a sort-of training task itself. We can use tools like CLIP Interrogator to investigate certain tokens that might work well in our dataset, but might not be obvious class labels: for example, "auto-destructive art" and "precisionism" bring our accuracy up dramatically.

Class Labels	Damage Accuracy	Intact Accuracy	Overall Accuracy
"damaged car", "undamaged car"	1.65%	**100.00%**	50.83%
"damaged vehicle", "undamaged vehicle"	37.83%	98.43%	68.13%
"damaged car", "whole car"	65.22%	99.74%	82.48%
"damaged car", "intact car"	84.78%	83.74%	84.26%
"car with dents, damage, scratches, or other blemishes", "car"	**96.52%**	22.35%	59.43%
"damaged car, auto-destructive art", "a car, clean, precisionism"	91.91%	93.39%	**92.65%**

Figure 10. Comparison of CLIP Prompts

This issue with CLIP extends to multi-class classification. How we chose to label "front damage", "side damage", and "back damage" is likely different from the way CLIP conceptualizes those tokens. Since CLIP never learned from our labeled training data, evaluating against it for multi-class classification proved to be difficult. Even using the same CLIP Interrogator trick from binary classification is more challenging because the specificity of CLIP's descriptions doesn't neatly map onto our labels. This explains why we had notably better success using ResNet for multi-class classification.

While popular CNN models performed better for the task laid out in this report, we believe CLIP could still be the better option if the task was tweaked slightly. For example, CLIP excels at describing a wide range of specific vehicle damage, and, from exploration with CLIP Interrogator, may

perform better if classification was performed by specific vehicle part (hood, front bumper, etc.) or the specific type of damage (dent, scratch, etc.)

Overall, this project gave our group a good understanding of the successes and limitations of pre-trained models. Though our results were good for binary classification, multi-class classification was more of a challenge. Even so, the methods used in this project will be tested and potentially implemented into a corporate fleet management system by one of our group members in the future.

6. External Resources Used

References

[1] Saleh Albelwi. Deep architecture based on densenet-121 model for weather image recognition. *International Journal of Advanced Computer Science and Applications*, 13, 01 2022. 1

[2] A. H. Alkurdi A. B. Marqas R. Shamal Salih M. Asaad Zebari, N. . Enhancing brain tumor classification with data augmentation and densenet121. *Academic Journal of Nawroz University*, 12(4), 323–334, 2023. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 2

[4] Aneeshan Sain, Ayan Kumar Bhunia, Pinaki Nath Chowdhury, Subhadeep Koley, Tao Xiang, and Yi-Zhe Song. Clip for all things zero-shot sketch-based image retrieval, fine-grained or not, 2023. 2

[5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019. 1