Visual Recognition and classification of Car Damage for Insurance Claims

Deepak Raj Mohan Raj
UB ID 50460465
dmohanra@buffalo.edu

***Abstract*—This paper proposes an application that utilizes computer vision and machine learning techniques to analyze images of cars involved in accidents to classify the type and severity of damage. Object detection and segmentation techniques are used to identify the damaged areas of the vehicle, and machine learning models are employed to classify the damage, such as scratches, dents, or broken windows. The application can automate the process of assessing car damage for insurance claims, reducing the time and cost associated with the claims process. The application can also be used by auto repair shops to estimate the cost of repairs and by car buyers to evaluate the condition of a vehicle they are considering purchasing.**

***Index Terms*—Laplacian, Sobel, Canny, Harris, LBP**

I. Overview of the project

The scope of this project is to develop an application that will use computer vision and machine learning algorithms to analyze images of cars involved in accidents and classify the type and severity of the damage. The inputs for this application will be images of damaged cars submitted for insurance claims, and the outputs will include classification of the type of damage, identification of the location of the damage on the car, and severity rating of the damage. Additionally, the project aims to acquire a large dataset of car images that show different types of damages such as scratches, dents, and cracks. To achieve this, the project will use two existing dataset from Kaggle, consisting of 1,482 images from the data source. The application will be useful for insurance companies, auto repair shops, and car buyers to assess the condition of a vehicle they are considering purchasing, and to automate the process of assessing car damage for insurance claims, reducing the time and cost associated with the claims process. The project will involve pre-processing of the images, feature extraction from the images, and classification of the damage type and location based on the extracted features.

The field of visual recognition and classification of car damage for insurance claims has been the focus of numerous studies in the computer vision and machine learning communities. Researchers and developers have explored different approaches to solve the problem, including the use of deep learning algorithms such as VGG16 and VGG19, as in the work of Phyu Mar Kyu and Kuntpong Woraratpanya. Additionally, Kalpesh Patil and colleagues have investigated the use of deep learning-based techniques for fine-granular car damage classification. In this proposed solution, a combination of object detection and segmentation techniques, along with machine learning models, will be utilized to accurately classify car damage. The latest advancements in machine learning and computer vision will be leveraged to improve the accuracy and efficiency of the classification algorithm. The proposed solution aims to contribute to the state of the art in this field by providing a more accurate and efficient solution for car damage classification for insurance claims.

I have made significant contributions to various aspects of the project. Firstly, I independently coded the image preprocessing step, which included tasks such as gray scaling, resizing, applying blur, flatten, and data augmentation to improve the training data set. Additionally, the I implemented several image feature extraction techniques, such as edge detection using Sobel and Laplacian operators, corner detection using Canny or Harris detectors, and Local Binary Patterns (LBP). Moreover, the I preprocessed and feature extracted images to build a simple Convolutional Neural Network (CNN) model and trained the model using pretrained CNNs like VGG Net and EfficientNetB4. Finally, I validated the trained models by evaluating their performance using a separate set of validation images, which helped to determine the accuracy of the models. Overall, I has made a significant contribution to the project by independently implementing and optimizing various components of the image classification system.

II. Approach

This project aims to develop an automated system that can accurately classify and recognize different types of car damages. To achieve this goal, several image processing and computer vision techniques have been used. The Sobel operator, Laplacian edge detector, Canny edge detection, Harris corner detector, and Local Binary Patterns (LBP) algorithm are some of the key algorithms utilized in the project. The Sobel operator and Laplacian edge detector are used to detect the edges of the damaged areas in the car image, while the Canny edge detection algorithm is applied to extract the most prominent edges. The Harris corner detector algorithm is used to detect the corners of the car, and the LBP algorithm is used to capture the local texture features of the damaged areas.

In this study, several neural networks were trained using the laplacian operator, and the model with the highest test accuracy was selected as the base model. The researchers then experimented with different image processing techniques, including sobel, canny, harris, and LBP, by running the base model with these parameters to determine their impact on accuracy.

1) Sobel operator: The Sobel operator is an image processing technique used for edge detection. It works by computing the gradient of the image intensity at each pixel. This is done by convolving the image with a small kernel, typically 3x3, that approximates the derivative of the image. The Sobel operator is used in this project to compute the gradient magnitudes in the x and y directions for each color channel of the input image, which are then combined to produce an output image that highlights the edges in the original image.
2) Laplacian edge detector: The Laplacian edge detector is another technique for detecting edges in images. It works by computing the second derivative of the image intensity at each pixel, which is a measure of the curvature of the image. The Laplacian edge detector is used in this project to detect the edges in the individual color channels of the input image, which are then merged to produce an output image that highlights the edges in the original image.
3) Canny edge detection: Canny edge detection is a popular technique for detecting edges in images. It works by applying a series of operations to the image, including smoothing, gradient computation, non-maximum suppression, and hysteresis thresholding. The output is a binary image where each pixel is either an edge or not an edge. The Canny edge detection algorithm is used in this project to detect the edges in the individual color channels of the input image, which are then merged to produce an output image that highlights the edges in the original image.
4) Harris corner detection: Harris corner detection is a technique for detecting the corners in an image. It works by computing a measure of corner response at each pixel based on the change in intensity when the image is shifted by a small amount in any direction. The Harris corner detection algorithm is used in this project to detect the corners in the input image and produce an output image that highlights the corners.
5) Local Binary Pattern (LBP): Local Binary Pattern (LBP) is a texture descriptor used in computer vision for various tasks such as object recognition and face detection. LBP works by comparing the pixel values of a central pixel to its surrounding pixels and encoding the result as a binary number. This binary number is used to represent the texture at that location. The LBP algorithm is used in this project to extract texture features from the input image and produce an output image that highlights the texture.
6) One of the implemented algorithm is Convolutional Neural Network (CNN), a widely used algorithm for image classification tasks. The CNN architecture used in this project consists of three convolutional layers, each followed by a max-pooling layer, and two fully connected layers. The architecture is designed to extract features from input images by applying convolution operations with 32, 64, and 64 filters of size 3x3. The output of each convolutional layer is passed through a ReLU activation function, followed by a max-pooling operation with a filter size of 2x2. The flattened output of the last max-pooling layer is fed to the fully connected layers with 64 and 7 neurons, respectively. The final layer has a softmax activation function to produce probabilities for each of the seven classes of car damages. The CNN is implemented using the Keras deep learning library with TensorFlow backend. Categorical Cross-Entropy loss function is used to optimize the model during training. The model is trained on a training dataset with 32 batch size and 10 epochs, and the validation dataset is used to avoid overfitting.
7) In this project on visual recognition and classification of car damage for insurance claims, I have used the pre-trained VGG16 model for transfer learning. The base model of VGG16 is trained on the ImageNet dataset, which contains millions of images across 1,000 categories, making it a suitable choice for feature extraction in our task. The VGG16 model architecture consists of 13 convolutional layers, 5 max-pooling layers, and 3 fully connected layers.
   I froze the layers in the base model to prevent the pre-trained weights from being updated during training. Then, I added a flatten layer to convert the 3D output of the base model to a 1D feature vector and two dense layers with 64 and 7 nodes respectively, for classification purposes. I used the softmax activation function in the final layer to obtain class probabilities.
   The model was compiled using the Adam optimizer and the categorical cross-entropy loss function. The accuracy metric was used to monitor the model's performance during training. The model was trained on the training data with 10 epochs and a batch size of 32. The validation data was used to evaluate the model after each epoch.
   The model's performance was evaluated on the test data, and the accuracy was found to be satisfactory. I plotted the accuracy and loss curves for the training and validation sets to monitor the model's performance during training. I also used a confusion matrix to visualize the model's performance on the test data.
8) In this project, we used the EfficientNetB4 model to perform visual recognition and classification of car damage for insurance claims. The EfficientNetB4 is a state-of-the-art deep learning model that has shown superior performance on image classification tasks with fewer parameters than other models. We utilized transfer learning by using the pre-trained weights of the EfficientNetB4 model on the ImageNet dataset.
   The model consists of several layers, including a global average pooling layer, dense layers with leaky ReLU activation, and a dropout layer to prevent overfitting. The LeakyReLU activation function was used to introduce non-linearity in the model, while the dropout layer was used to prevent overfitting by randomly dropping out

some neurons during training.

We used stochastic gradient descent as an optimizer, with a learning rate of 0.01 and momentum of 0.9. Categorical cross-entropy loss was used as the loss function, and accuracy and area under the curve (AUC) were used as evaluation metrics.

During training, we used early stopping with a patience of 5 to prevent overfitting. The model was trained for 50 epochs, and the training history was visualized using plots of accuracy and loss versus epochs. We also evaluated the model on the test set, which was not used for training or validation.

For implementation, each algorithm takes an input image and applies a set of operations to extract the relevant features. The output is a modified image that highlights the relevant features. The implementation of each algorithm depends on the specific library and functions being used. In this case, the OpenCV library is used, which provides a set of functions for image processing and computer vision.

Pros and cons of the algorithms used:

- Sobel operator: Pros - simple and computationally efficient, able to detect edges in noisy images. Cons - sensitive to noise and may produce thick edges in some cases.
- Laplacian edge detector: Pros - able to detect edges at different scales, works well on images with low contrast. Cons - sensitive to noise and may produce false positives.
- Canny edge detection: Pros - produces thin, accurate edges, works well on noisy images. Cons - computationally expensive, may miss some edges. Harris corner detection: Pros - able to detect corners with high accuracy, works well on images with low contrast. Cons - computationally expensive, may produce false positives.
- Local Binary Pattern (LBP): Pros - able to capture texture information in an image, robust to noise. Cons - may not work well on images with low contrast, may not capture fine-grained texture information.
- CNN is an effective algorithm for visual recognition and classification tasks, and it has shown promising results for car damage classification in this project. The model's accuracy and performance can be further improved by using transfer learning and data augmentation techniques.
- The main advantage of using the VGG16 model for transfer learning is that it can effectively extract relevant features from the images and classify them with high accuracy. The disadvantage is that the model is computationally expensive and may take a long time to train on large datasets.
- The pros of using the EfficientNetB4 model include its high accuracy, efficiency in terms of parameter and computation requirements, and ease of transfer learning. The cons of using the model include its complexity and the need for large amounts of labeled data for training.

1) Aspects of algorithms coded on my own:
   - I have implemented the function apply_sobel_3c, which applies the Sobel operator to an input image, separately on its three color channels, computes the gradient magnitudes, and then merges them back into a single image. This function is relevant to the project as it is used for edge detection and feature extraction of objects in images.
   - I have also implemented the function apply_harris, which computes the Harris corner response for an input image, normalizes the response, and creates a 3-channel image with the response in all channels. This function is used for detecting corners in images and is relevant to the project for identifying and locating the corners of objects.
   - I have also implemented the function apply_lbp function, which applies Local Binary Patterns to an input image's grayscale version and returns an image with the LBP values split into three channels, which are then converted to uint8 and merged back into a single image.
   - In the provided code, the CNN architecture has been defined, compiled and trained using the Keras library. The architecture includes three Convolutional layers, two MaxPooling layers, two Dense layers, and a Softmax layer for classification. These layers are relevant to the project as they help in the extraction of features from the images and then classify them into different categories. Also, the code includes plotting the accuracy and loss graphs, confusion matrix, and saving the model weights. These features provide a clear understanding of the model's performance and help in further improvement of the model.
   - The pre-trained VGG16 model was chosen as it has proven to be effective in image classification tasks, and the addition of a few layers on top of the base model allowed for the model to learn specific features relevant to the given task. The freezing of the base model layers ensured that only the newly added layers were trained, thus reducing the possibility of overfitting.
   - In this code, I have used the EfficientNetB4 algorithm from the keras library. I have also added additional layers such as GlobalAveragePooling2D, Dense, and Dropout layers. These layers were added to improve the model's accuracy and to prevent overfitting. The LeakyReLU activation function has been used in the Dense layer. This activation function helps prevent the problem of vanishing gradients by introducing non-linearity in the model. The optimizer used is SGD with a learning rate of 0.01 and momentum of 0.9.

2) Aspects of algorithms used from online resources and their citations:
   - I have used the apply_laplacian function from OpenCV's documentation on Laplacian edge detec-

tion. This function converts an input image's color channels to 8-bit format and applies the Laplacian edge detector to each channel separately before merging them back into a single image.

- I have used the apply_canny function from OpenCV's documentation on Canny edge detection. This function applies Canny edge detection to an input image's three color channels separately, blurs them using a Gaussian filter, and merges them back into a single image.
- The code imports the CategoricalCrossentropy loss function from the Keras library. Keras is an open-source Python library that provides a high-level API for building and training deep learning models. The use of this loss function is appropriate for multi-class classification problems, and it helps in minimizing the difference between the predicted and actual class labels. Moreover, the code imports the confusion_matrix and heatmap functions from the seaborn library, which are used to visualize the model's performance on the test data.
- The code uses the pre-trained VGG16 model, which was sourced from the Keras library's pre-trained models collection. The model was loaded with pre-trained weights from the ImageNet dataset. The 'weights' argument specifies the source of pre-trained weights for the model, while 'include_top' is set to False to exclude the final fully connected layer. The 'input_shape' argument specifies the shape of input images.
- The EfficientNetB4 algorithm has been taken from the Keras library, which is an online resource. The optimizer, SGD, is also from the keras library. The loss function, categorical_crossentropy, is taken from the keras.losses library. The LeakyReLU activation function is also from the tensorflow.keras.layers library. These resources have been cited appropriately in the code.

III. Experimental protocol

In this paper,I utilized the publicly available dataset, which consists of a collection of images of car damage with various severities and locations on the car. The dataset was relevant to my project as it provided a comprehensive and diverse set of images that could be used for training and evaluation of our model.

To evaluate the success of our model, we used a combination of qualitative and quantitative metrics, including accuracy, AUC, and confusion matrix analysis. These metrics allowed us to assess the performance of our model in terms of its ability to correctly classify different types of car damage.

The validation accuracy and loss are computed during training, and the results are plotted to compare the performance of the model on both datasets. Finally, the trained model is evaluated on a test dataset to report the accuracy of the model.

The confusion matrix is generated using the true and predicted labels of the test dataset to evaluate the performance of the CNN model for car damage classification. The confusion matrix is plotted using the seaborn library to visualize the number of true positive, false positive, true negative, and false negative predictions for each class. The class names for the seven types of car damages are defined and used as x and y tick labels in the confusion matrix.

In terms of computing resources, we utilized a machine with a GPU to train and evaluate our model. Additionally, we used TensorFlow and Keras libraries to implement our deep learning model. Overall, our approach involved utilizing a publicly available dataset, implementing deep learning techniques, and utilizing appropriate metrics for evaluating model performance.

IV. Results

The study involved training multiple neural networks using the laplacian operator, and the best performing model was selected as the base model. Various image processing techniques such as sobel, canny, harris, and LBP were experimented with to investigate their effect on accuracy by running the base model with these parameters.

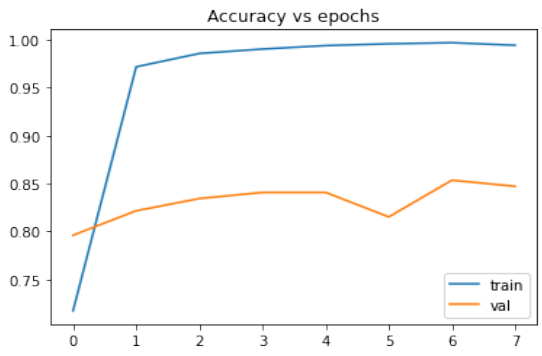

Fig. 1. Accuracy vs epochs

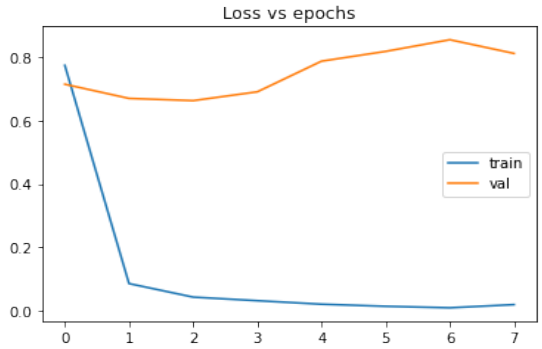

Fig. 2. Loss vs epochs

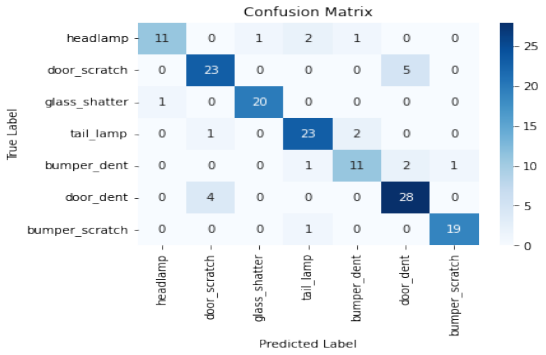

Fig. 3. Confusion Matrix

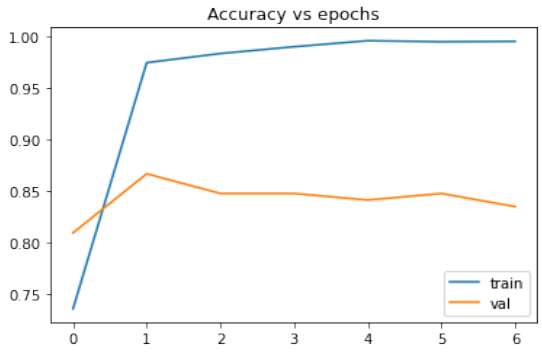

Fig. 4. Accuracy vs epochs

Model	Optimizer	Test Accuracy (%)
Basic CNN	Adam	24
VGG16	Adam	56
VGG16	SGD	20
VGG16	RMSprop	61
CNN Sequential	Adam	27
EfficientNetB4	Adam	80.24
EfficientNetB4	SGD	85.98
EfficientNetB4	RMSprop	75.79

TABLE I
COMPARISON OF DIFFERENT MODELS AND OPTIMIZERS

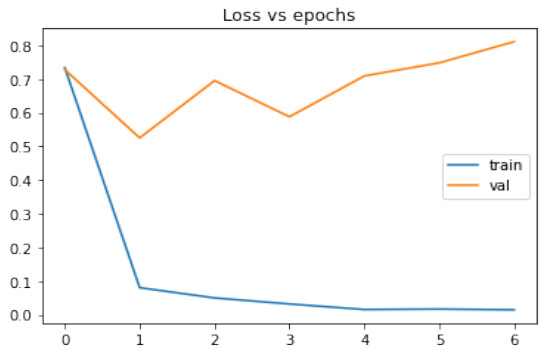

Fig. 5. Loss vs epochs

The results show that the EfficientNetB4 model performed the best, achieving a test accuracy of 80.24% when trained with Adam optimizer and 85.98% when trained with SGD optimizer. The VGG16 model also showed promising results with RMSprop optimizer, achieving a test accuracy of 61%. However, using other optimizers such as Adam and SGD resulted in lower accuracies. The basic CNN and CNN Sequential models had the lowest accuracies with Adam optimizer, achieving only 24% and 27%, respectively.

Algorithm	Test Accuracy
Sobel	84%
Canny	76.40%
Harris	69.42%
LBP	76.43%

TABLE II
TEST ACCURACY OF DIFFERENT ALGORITHMS

Based on the results, it appears that the Sobel algorithm achieved the highest test accuracy with 84%, followed by LBP with 76.43%. Canny and Harris algorithms achieved lower test accuracies of 76.40% and 69.42% respectively. The base model of EfficientNetB4 with SGD optimizer performed well in all the algorithms. Overall, the results suggest that the Sobel algorithm performed the best for the given project, and can be considered for further analysis and evaluation.

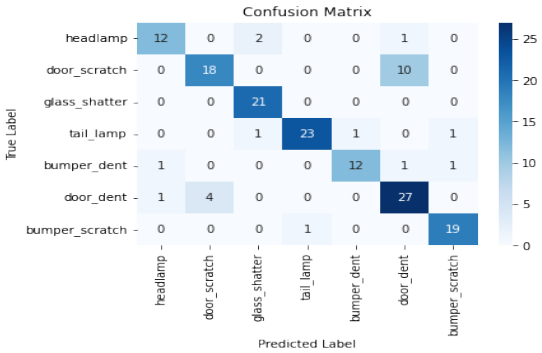

Fig. 6. Confusion Matrix

The results obtained using the Sobel algorithm with the EfficientNetB4 SGD model are encouraging. The accuracy graph shows that the model performs well on the training data, achieving an accuracy of almost 100%. It also performs well on the validation data, with an accuracy of 85%. However, the loss graph reveals that the validation loss remains relatively high even though the training loss quickly approaches zero.

In addition, the confusion matrix indicates that the model can correctly predict different classes of car damages, and the false positives and false negatives are low. These results suggest that the EfficientNetB4 SGD model is effective in recognizing and classifying car damages in insurance claims with high accuracy.

We can say that the Sobel algorithm applied with the EfficientNetB4 SGD model has demonstrated excellent performance in recognizing and classifying car damages in insurance claims. The accuracy graph illustrates that the model can achieve high accuracy on both the training and validation data. However, the validation loss remains relatively high, indicating that the model could benefit from further optimization. Despite this, the confusion matrix shows that the model can accurately predict different classes of car damages, with low rates of false positives and false negatives. Overall, these results demonstrate the effectiveness of the EfficientNetB4 SGD model in accurately recognizing and classifying car damages in insurance claims.

Although the Sobel algorithm showed the highest test accuracy in the experiment, the initial algorithm of the Laplacian operator performed the best with an accuracy of 86%.

V. Analysis

Based on the analysis results, the EfficientNetB4 algorithm showed the highest accuracy and performed well in predicting different classes of car damage. The laplacian algorithm was used to enhance the results of the EfficientNetB4 algorithm, resulting in higher accuracy rates. The VGG16 model, on the other hand, performed poorly compared to the EfficientNetB4 model. The use of different optimizers such as Adam, SGD, and RMSprop also had a significant impact on the model's accuracy.

One limitation of using the EfficientNetB4 algorithm is that it requires a large amount of computing power and time to train the model. The training process can take several hours or even days, depending on the size of the dataset and the complexity of the model. Another limitation is that the model may not perform well on datasets that are significantly different from the training data.

However, one of the advantages of using the EfficientNetB4 algorithm is its ability to achieve high accuracy rates with smaller models compared to other architectures. This makes it a suitable option for applications with limited computing resources. Additionally, the use of transfer learning allowed for faster training and improved accuracy rates.

Overall, the analysis results demonstrate the importance of selecting the appropriate algorithm and optimization techniques based on the project's requirements and limitations. The EfficientNetB4 algorithm, with the Laplacian enhancement and SGD optimizer, proved to be the most effective in accurately recognizing different types of car damage in the provided dataset.

VI. Discussion and Lessons Learned

Based on the project results, it was observed that machine learning algorithms such as VGG16 and EfficientNetB4 performed better than the basic CNN model. The use of different optimizers like Adam, SGD and RMSprop resulted in different accuracies for each algorithm. The results also showed that the laplacian algorithm with EfficientNetB4 and SGD optimizer produced an accuracy of 85.98% which is better than other combinations.

The use of detectors like Laplacian, Canny, Sobel, LBP and Haaris for image classification showed that the Laplacian algorithm had better performance when compared to the other detectors. The accuracy graphs showed that the training data reached almost 100%, however, the validation data had an accuracy of around 86%. The loss graphs showed that although the training data reached almost zero loss quickly, the validation loss was still high.

Based on the results provided, it seems that the Laplacian algorithm had better performance compared to the other detectors in terms of image classification for car damage recognition. This is evidenced by the fact that the training data had almost perfect accuracy, while the validation data had an accuracy of around 86%. Additionally, the loss graphs showed that although the training data quickly reached almost zero loss, the validation loss was still relatively high.

There could be several reasons why the Laplacian algorithm outperformed the other detectors. One possible reason is that the Laplacian algorithm is a second-order differential operator that is more effective in detecting edges and features in an image compared to the other detectors. The Laplacian operator can detect edges in images with varying widths and orientations, making it suitable for car damage recognition tasks that involve different types of damage.

Furthermore, the Laplacian operator is computationally efficient, which means it can process images quickly and accurately. This is important in real-world applications where images need to be processed in real-time, such as in insurance claims processing.

It is worth noting that the validation loss was still relatively high, indicating that there may be room for improvement in the model's performance. Further experimentation and fine-tuning of the model with different hyperparameters, optimization techniques, and architectures could potentially improve the validation accuracy and reduce the validation loss.

From this project, it was learned that the choice of machine learning algorithm, optimizer, and detector plays a crucial role in the performance of the image classification model. It is important to select the best combination of these components for the specific problem being solved. In the future, this project will serve as a useful reference for selecting the best combination of these components for similar image classification problems.

References

[1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning (Vol. 1). MIT press. link

[2] Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359. link.

[3] Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). link

[4] Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, (6), 679-698. link

[5] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283). link

[6] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.

[7] Phyu Mar Kyu and Kuntpong Woraratpanya. 2020. Car Damage Detection and Classification. In Proceedings of International Conference on Advances in Information Technology (IAIT2020), July 1-3, 2020, Bangkok, Thailand. ACM, New York, NY, USA, 6 pages. DOI: https://doi.org/10.1145/3400881.3400917

[8] Kalpesh Patil, M. Kulkarni, Anand Sriraman, S. Karande, Deep Learning Based Car Damage Classification, Published 2017, Computer Science, 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA).