ISSN 2278-3091
Volume 10, No.5, September - October 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse031052021.pdf
https://doi.org/10.30534/ijatcse/2021/031052021

Deep Learning Based Car Damage Detection, Classification and Severity

Ritik Gandhi[1]
[1]Shri Govindram Seksaria Institute of Technology and Science, Indore, India, ritikgandhi21@gmail.com

Received Date : August 06, 2021 Accepted Date : September 15, 2021 Published Date : October 06, 2021

ABSTRACT

In the accident insurance industry, settling the claim is a time-consuming process since it is a manual process and there is a gap between the optimal and the actual settlement. Using deep learning models, we are not only trying to speed up the process but also provide better customer service and increase the profitability of insurance companies. In this paper we are using various pretrained models such as VGG 16, VGG 19, Resnet50 and Densenet and based on these models, selecting the best performing models. We initially check whether the car is damaged or not using the Resnet50 model and if it's a damaged one we use the WPOD-net model to detect the license plate. To identify the damaged region, we use the YOLO model. At last, comes the damage severity which is implemented using the Densenet model. After implementing various models, we find out that transfer learning gives better results than fine-tuning. In addition to that we propose a framework that integrates all of this into one application and in turn helps in the automation of the insurance industry.

Key words: Deep Learning, Damage assessment (detection, classification and severity), Pre-trained CNN Models, YOLO

1. INTRODUCTION

The Global Auto Insurance market is projected to reach $1.06 trillion by 2027 and still a lot of money is being wasted when it comes to claims. The traditional method involves a tedious process wherein the customer submits the claim documents to the agent who in turn submits the claim documents to the company and then an external evaluator inspects the unit and correspondingly prepares his reports. The company reviews it and issues the LOA and then sends the car to the shop for repairs. This has forced the insurance firms to look out for solutions that include fair assessment and faster agreement of claims.

In this paper we try to employ different Convolutional Neural Network (CNN) models such as VGG16, VGG19, Resnet50, Densenet ,etc and based on the accuracy, select the models that work best for us. We couldn't find any publicly available dataset for the same and therefore created our own dataset by web scraping different sites. Manual filtering and annotating of the images were also done [15]. Since the dataset is small, we used Data Augmentation Techniques to synthetically enlarge the dataset. We consider the common car damages such as bumper dent, bumper scratch, door dent, door scratch, glass shattered, scratch, smash and some images of no damaged cars.

A number of techniques were tried such as directly training a CNN, using a pre-trained CNN model, using transfer learning from large CNNs and building an ensemble classifier. Out of all the techniques tried we observed that transfer learning works the best. The damaged part classification and its localization was done using the YOLOv3 model which classifies an image and draws the bounding boxes around them. To add more to it, YOLOv3 framework is significantly faster than R-CNN models because it has a bigger network and residual networks are added by adding various shortcut connections. To extract and read the license plate we used the WPOD-net model which detects the license plate no matter how different the distortion is and further rectifies the license plate area to a rectangular shape so that the detections can be further fed to the OCR network. The damage severity of the car was done on 3 parameters: Minor, Moderate, Severe.

Although a lot many minor factors were taken into account so as to make the model as much best performing as possible but along the way the focus was on the influence of certain hyper-parameters and searching theoretically defined ways to adapt them [2].

Since Deep Learning is one of the best techniques when it comes to image processing related tasks, a major challenge was to reduce the model training time since a traditional CNN model can be very time-consuming to perform image classification tasks and identify the correct weights for the network by multiple forward and backward iterations.

2. RELATED WORK

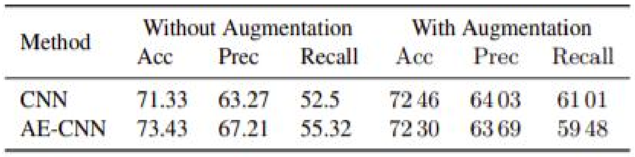
Whenever object detection comes into play, deep learning has always shown promising results. The most popular detection algorithms include the Convolutional Neural Networks (CNN), since they perform well for many computer vision tasks such as visual object recognition and detection [3][4]. With computing resources based on transfer learning solutions and extensive use of data, deep learning has been outstanding in image classification [5][6].

To perform different tasks of localization and detection different models are proposed, however according to [7], they tried to implement a complete system with transfer learning based on CNN models but could not calculate the damage severity.

Pre-trained CNN models are very complicated to understand because of their intense variance but they can still be used as a feature extractor. Their weights can be freely downloaded and applied via transfer learning.
Structural damages have also been identified and studied using the CNNs in [8] where the authors propose a deep learning-based method to characterize the cracks on a composite material.

In case of small number of labeled samples autoencoders have improved the performance of the classifier. Multi sensor-data fusion techniques have been used to solve vehicle body damage problems [9]. Unsupervised pre-training techniques have improvised the general performance of the classifier as compared to the supervised techniques. CNN models also have tremendous applications in ship-target detection as stated by Wang et al [10] which solves the problem of closely aligned targets and multi-scale targets. Based on RCNN, building target detection algorithms have been proposed which remote sense the images of different scenes [11]. To handle automatic vehicle damage detection via photographs 3D CAD Models were used [12]. The YOLO Object Detection model was applied in [13] although the results weren't quite satisfactory and up-to the mark. In [14] the team collected different images and sorted the dataset into many classes and since the dataset containing the images were less, they synthetically enlarged the dataset 5 times. Since thy couldn't achieve an appropriate accuracy, they went with predefined models and from output of pre-trained models, trained a linear SVM. Based on the experiments, using SoftMax Classifier is better than Linear SVM. For images, Convolutional Auto Encoders (CAE) have shown good results.

Table 1. Test accuracy with CNN training



Method	Without Augmentation			With Augmentation		
	Acc	Prec	Recall	Acc	Prec	Recall
CNN	71.33	63.27	52.5	72.46	64.03	61.01
AE-CNN	73.43	67.21	55.32	72.30	63.69	59.48

Most of the papers majorly concentrate on CNN models to detect the damaged part with techniques via transfer learning and some researchers use a better segmentation algorithm with the camera type image for analysis.
Related to the ALPR (Automatic License Plate Recognition) systems are Scene Text Spotting (STS) which find and read text/numbers in natural scenes. Many systems proposed typically use image binarization or gray-scale analysis to find candidate proposals.

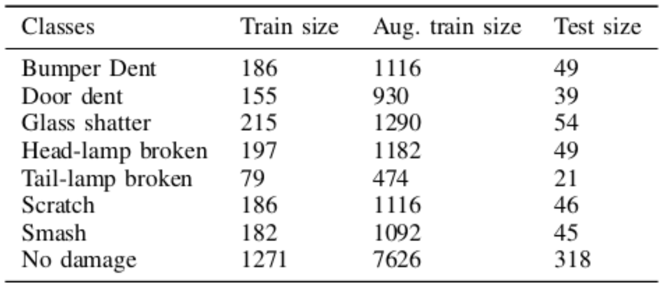
3. DATASET DESCRIPTION

There were a couple of datasets related to the car damage classification but none of them served the purpose we wanted to get a proper architecture; hence we created our own dataset that contained images. We considered images of cars that were damaged and undamaged. If the cars were damaged, then the damaged part was considered such as bumper dent, bumper scratch, door dent, door scratch, windshield damage, head lamp broken, tail lamp broken, smash. There was another dataset that was specifically used for predicting the damage severity of car that was classified as minor, moderate and severe. For the license plate dataset, a dataset of the Indian car license plate images was used.

- Dataset 1- Training and validation sets of damaged and undamaged cars.
- Dataset 2-Training and validation sets of Damage on Front, Rear and Side.
- Dataset 3-Training and validation sets of Damage Severity Minor, Moderate, Severe.
- Dataset 4- Training and validation sets of different damaged parts such as door dent, door scratch, bumper dent, bumper scratch etc.
- Dataset 5- Training and validation sets of number plate with different distortions.

Since we had a small dataset, we used **Data Augmentation** to enlarge the dataset. To improve the execution of models and expand small sized datasets data augmentation gives an ideal solution as explained in [15]. Although there are a couple of approaches for the same, we enlarged the dataset twice using horizontal flip transformations and random rotations between -30 and 30. The YOLOv3 models are trained on the COCO Dataset but the images had to be annotated using third party tools called labelImg. 3 YOLO models developed individually identify various damaged parts of the car. The YOLOv3 models are trained on the COCO Dataset but the images had to be annotated using third party tools called labelImg. 3 YOLO models developed individually identify various damaged parts of the car.

Table 2. Description of the dataset



Classes	Train size	Aug. train size	Test size
Bumper Dent	186	1116	49
Door dent	155	930	39
Glass shatter	215	1290	54
Head-lamp broken	197	1182	49
Tail-lamp broken	79	474	21
Scratch	186	1116	46
Smash	182	1092	45
No damage	1271	7626	318

After Data Augmentation for each dataset the number of files were as follows:

- Dataset 1- Original Data + Data Augmentation 2 (3980 Train files, 490 Test files)
- Dataset 2- Original Data + Data Augmentation 2 (1996 Train files, 193 Test files)
- Dataset 3- Original Data + Data Augmentation 2 (1968 Train files, 189 Test files)

The CNN was trained on both the original dataset and augmented dataset.

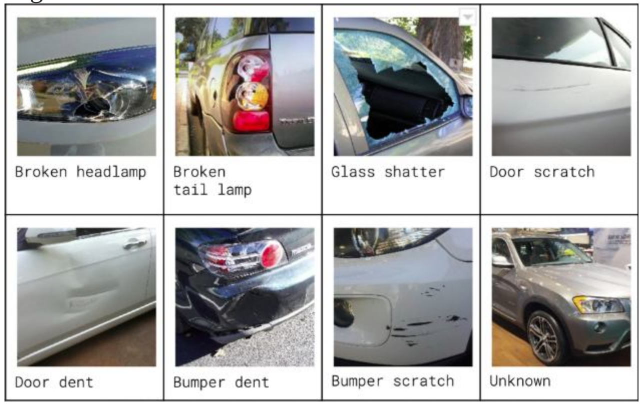


Fig 1. Sample images for car damage types

4. EXPERIMENTS AND IMPLEMENTATIONS

Initially, a CNN was trained with random initializations and for every convolutional layer a RELU non-linearity is used. Furthermore, the results also showed that data augmentation improves the performance and generalization as compared with training on the original dataset. The pretrained models were better than the models implemented from scratch and therefore VGG16, VGG19, Densenet and Resnet50 were imported without fully connected layers. To compare all the models Logistic Regression was chosen with features extracted from this model. Two models were trained by keeping Logistic Regression as the baseline model wherein the first model had layers as non-trainable and the second model had layers as trainable. Hyperparameter tuning of logistic regression was done and using best alpha the models were created. Stage 1 was compiled using Binary cross entropy loss whereas, Stage 2 and Stage 3 were compiled using Categorical Cross entropy. **Stochastic gradient descent optimizer (SGD)** and accuracy were used as the metric. Each model was trained for 50 epochs and the best model was saved using Model Checkpoint.The overall application is divided into 4 stages:

The first stage involves in detecting whether the car is damaged or not. The second stage involves extracting and reading the license number plate of the damaged car. Next stage involves the localization of the damaged part and figuring out which part of the car is damaged using the YOLO model. The last stage classifies the severity of the damaged car. Figure 2 depicts a flowchart of developing car damage assessment architecture

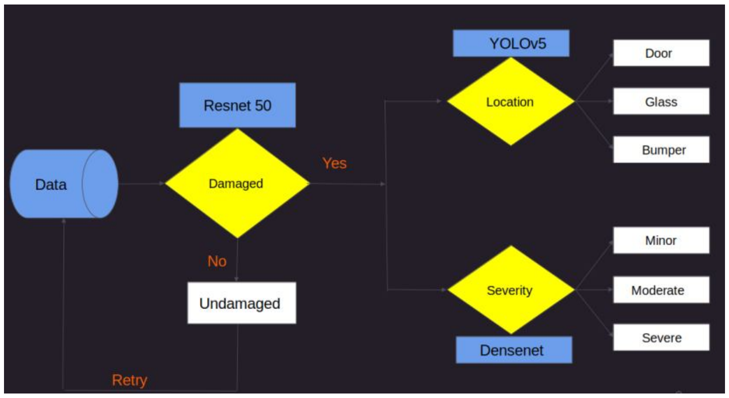


Fig 2. A flowchart depicting the overall process of the car damage assessment.

Stage 1: Detecting whether the car is damaged or not

There are different pretrained models like VGG16, VGG19, Densenet but **Resnet50** turned out to be the most accurate model to validate whether the car is damaged or not. The dataset contains train and validation sets such as Bumper Dent, Bumper Scratch, Door Dent, Door Scratch, Glass Shattered, Head Lamp, Tail Lamp, Undamaged, etc. If the car is undamaged then it simply detects it and if it's a damaged one, then there are further localizations made by the YOLO models. The model shows an accuracy close to **89%** on the validation set.

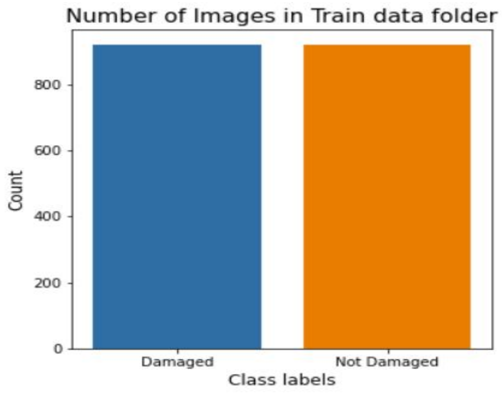


Fig 3. Images in train data folder

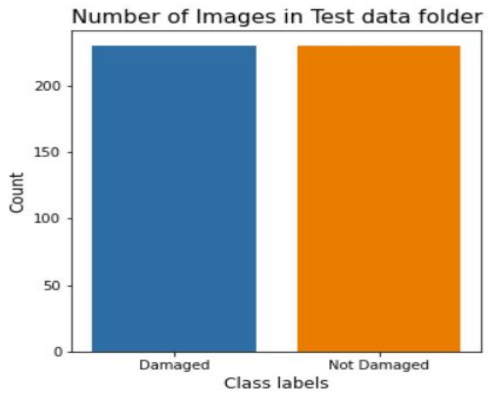


Fig 4. Images in test data folder

The images for the damaged and not damaged classes are equal leading to no class imbalance in both the training and testing data folder.

Stage 2: Extracting and Reading the license plate

The major aim in this stage is to extract the cropped image of the license plate and read the same using the **WPOD-net** model **(Warped Planar Object Detection Network).** To decrease the computational cost, it is better to convert the cropped image to the grey image and then to enhance the contrast and differentiate between license plate and other parts of image a grey level processing is applied. To highlight the difference between the background and the license plate frontier, the edges are detected using Roberts' operator.
An approach using the CNN model involves character segmentation done on the binary image of the preprocessed license plate and the extracted one. The CNN model is trained over a dataset of alphanumeric characters to recognize the segmented characters efficiently with an accuracy of about **93%.**

In different distortions the WPOD-net model detects the license plates and regresses the coefficients that actually unwraps the license plate into a proper rectangular shape.

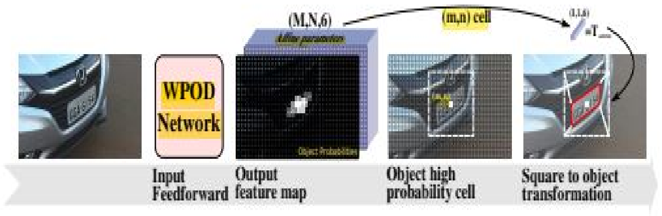


Fig 5. Mechanism of the WPOD-net model to extract the license plate.

The second approach uses Pytesseract which is an optical character recognition tool. The preprocessed image is passed to the Pytesseract OCR engine and we get the predicted license plate number from the image.

The license plate reader was tested over a dataset of the Indian car license plate images and the score was determined by comparing the predicted sequences of both Pytesseract and CNN with the actual value, using a Sequence Matcher.
The best of the two was taken as the score of that image. The accuracy was close to **80.34%.**

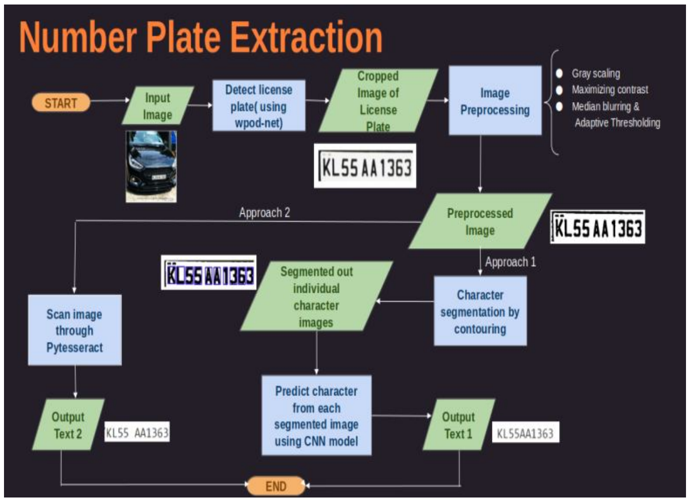


Fig 6. A flowchart depicting two different approaches for the number plate extraction

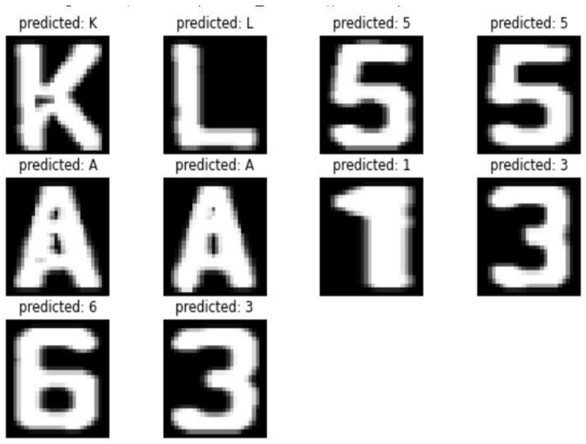
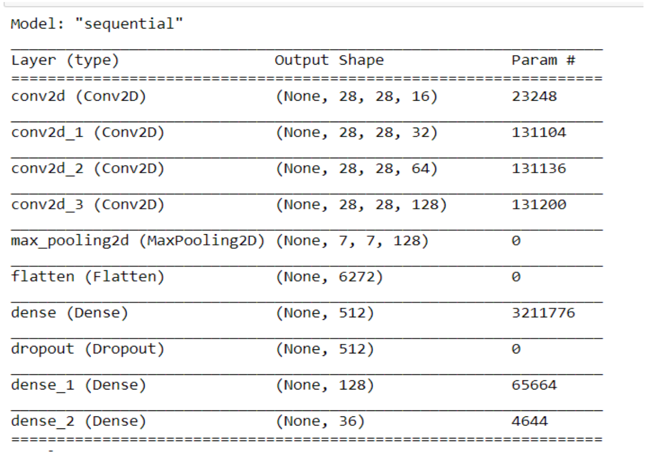


Fig 7. Predictions on each segmented image



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        23248
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 32)        131104
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 64)        131136
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 128)       131200
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 512)               3211776
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_2 (Dense)              (None, 36)                4644
=================================================================
```

Fig 8. CNN Architecture

Stage 3: Damaged Part Classification and Localization using YOLO

The damaged part classification is done using the YOLO model, if the car is damaged then then this model is used to localize the damaged part of the car. YOLO refers to **“YOU LOOK ONLY ONCE”** and is one of the most versatile models when it comes to object detection. It classifies and finds damaged part of a car in an image and draws the bounding boxes around them.

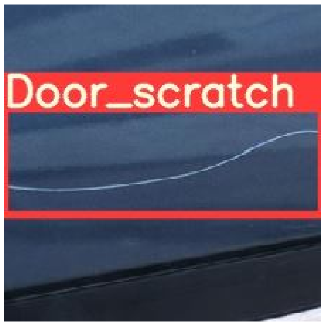


Fig 9. Door scratch

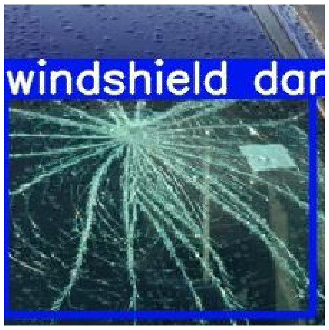


Fig 10. Windshield damage

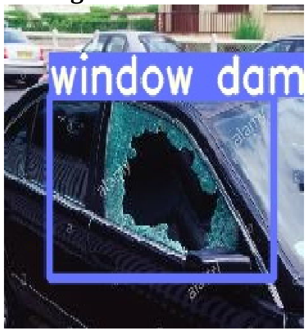


Fig 11. Window damage

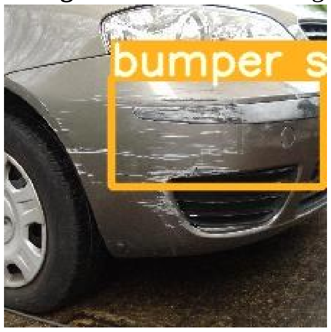


Fig 12. Bumper scratch

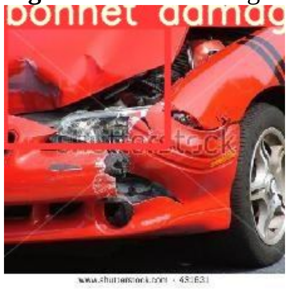


Fig 13. Bonnet damage

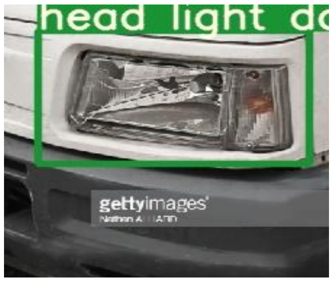


Fig 14. Head light damage

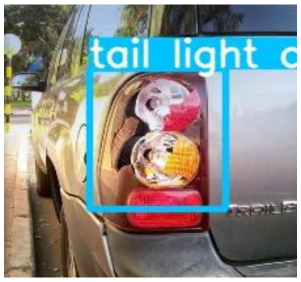


Fig 15. Tail light damage

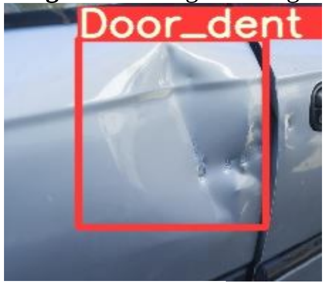
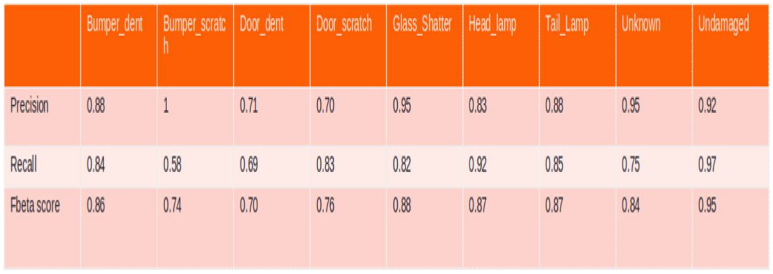


Fig 16. Door dent

Since there is no proper data, LabelImg tool was used for creating bounding boxes and giving classes. Certain specific files were created for training yolo. Since we had 3 classes the YOLO was trained for 7000 epochs and the weights were saved for multiples of 1000. The observation showed that YOLO custom 5000 weights gave better results than other models.

YOLO divides all the input images into the **SxS** grid system and each grid is responsible for object detection. The grid cells are actually responsible for prediction the boundary boxes for the detected objects. For every box there are 5 main attributes that are to be considered **x and y** for coordinates, **w and h** for width and height of the object, and a **confidence score** for the probability that the box containing the object. 9 classes are used for classification such as bumper dent, bumper scratch, door dent, door scratch, windshield damaged etc.



	Bumper_dent	Bumper_scratch	Door_dent	Door_scratch	Glass_Shatter	Head_lamp	Tail_Lamp	Unknown	Undamaged
Precision	0.88	1	0.71	0.70	0.95	0.83	0.88	0.95	0.92
Recall	0.84	0.58	0.69	0.83	0.82	0.92	0.85	0.75	0.97
Fbeta score	0.86	0.74	0.70	0.76	0.88	0.87	0.87	0.84	0.95

Table 3. Accuracy of Resnet50 damage classification model

The results mentioned in the above table were from a validation set of 500+ images with 9 classes and the accuracy of the overall model were close to **88.99%**.

Stage 4: Car Damage Severity

The classification of car damage severity is as follows:

- **Minor Damage** – It typically involves slight damage to the vehicle that does not impede the vehicle to cause severe injuries. It includes the headlight scratches, dents and digs in the hood or windshield, from gravel or debris, scratches in the paint.
- **Moderate Damage -** Any kind of damage that impairs the functionality of the vehicle in any way is moderate damage. It involves large dents in hood, fender or door of a car. Even if the airbags are deployed during collision, then it comes under moderate damage.
- **Severe Damage –** Structural damages such as bent or twisted frames, broken/bent axels, missing pieces of the vehicles and in some cases even the destruction of airbags. These types of damages are a big threat to the human life.

The densenet model was chosen to ensure the maximum flow between the layers in the network. In addition to that, the dense connectivity pattern requires fewer training parameters

than the traditional models such as VGG16, VGG 19. For training purposes, the Adam optimizer was chosen due to its fast convergence. The accuracy of the overall model was close to **70%**.
Categorical cross entropy was used as the loss function and the model was trained for 100 iterations.
Training - {Minor-278, Moderate-315, Severe-386, Total-979}
Testing - {Minor-48, Moderate-55, Severe-68, Total-171}

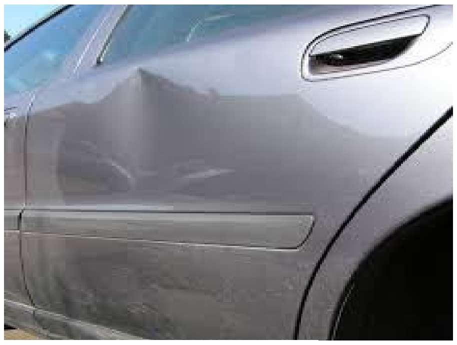


Fig 17. Minor damage

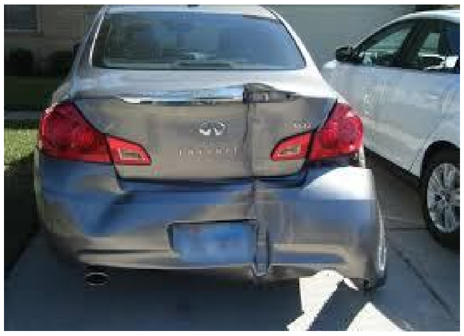


Fig 18. Moderate damage

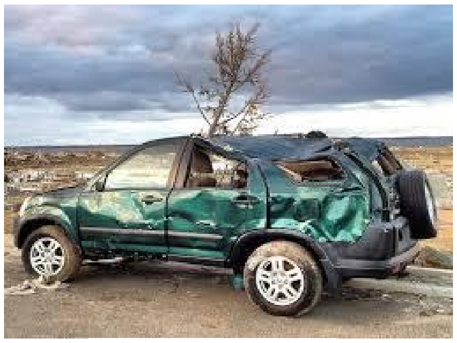
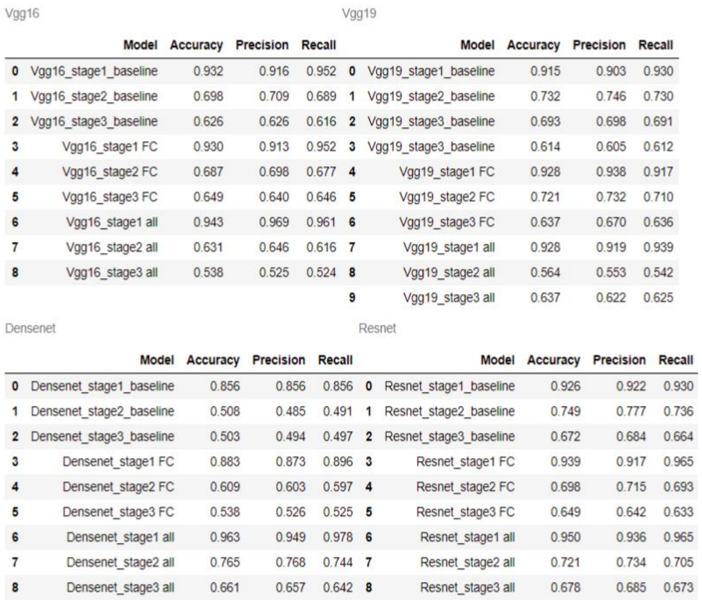


Fig 19. Severe damage

5. MODEL SELECTION

Accuracy, Precision and Recall are chosen as the three different metrics to estimate the performance of our different transfer learning models such as VGG16, VGG19, Resnet50 and Densenet. The higher the matrices the better our model performs.


Vgg16

	Model	Accuracy	Precision	Recall
0	Vgg16_stage1_baseline	0.932	0.916	0.952
1	Vgg16_stage2_baseline	0.698	0.709	0.689
2	Vgg16_stage3_baseline	0.626	0.626	0.616
3	Vgg16_stage1 FC	0.930	0.913	0.952
4	Vgg16_stage2 FC	0.687	0.698	0.677
5	Vgg16_stage3 FC	0.649	0.640	0.646
6	Vgg16_stage1 all	0.943	0.969	0.961
7	Vgg16_stage2 all	0.631	0.646	0.616
8	Vgg16_stage3 all	0.538	0.525	0.524

Vgg19

	Model	Accuracy	Precision	Recall
0	Vgg19_stage1_baseline	0.915	0.903	0.930
1	Vgg19_stage2_baseline	0.732	0.746	0.730
2	Vgg19_stage3_baseline	0.693	0.698	0.691
3	Vgg19_stage3_baseline	0.614	0.605	0.612
4	Vgg19_stage1 FC	0.928	0.938	0.917
5	Vgg19_stage2 FC	0.721	0.732	0.710
6	Vgg19_stage3 FC	0.637	0.670	0.636
7	Vgg19_stage1 all	0.928	0.919	0.939
8	Vgg19_stage2 all	0.564	0.553	0.542
9	Vgg19_stage3 all	0.637	0.622	0.625

Densenet

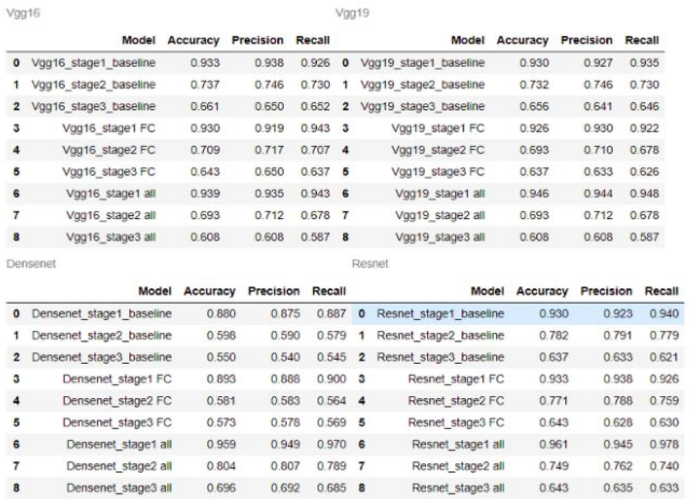
	Model	Accuracy	Precision	Recall
0	Densenet_stage1_baseline	0.856	0.856	0.856
1	Densenet_stage2_baseline	0.508	0.485	0.491
2	Densenet_stage3_baseline	0.503	0.494	0.497
3	Densenet_stage1 FC	0.883	0.873	0.896
4	Densenet_stage2 FC	0.609	0.603	0.597
5	Densenet_stage3 FC	0.538	0.526	0.525
6	Densenet_stage1 all	0.963	0.949	0.978
7	Densenet_stage2 all	0.765	0.768	0.744
8	Densenet_stage3 all	0.661	0.657	0.642

Resnet

	Model	Accuracy	Precision	Recall
0	Resnet_stage1_baseline	0.926	0.922	0.930
1	Resnet_stage2_baseline	0.749	0.777	0.736
2	Resnet_stage3_baseline	0.672	0.684	0.664
3	Resnet_stage1 FC	0.939	0.917	0.965
4	Resnet_stage2 FC	0.698	0.715	0.693
5	Resnet_stage3 FC	0.649	0.642	0.633
6	Resnet_stage1 all	0.950	0.936	0.965
7	Resnet_stage2 all	0.721	0.734	0.705
8	Resnet_stage3 all	0.678	0.685	0.673

Fig 20. Original Data

On the original data for stage 1 that is to check whether the car is damaged or not we can observe that Densenet trained on all layers is performing better than other models. The accuracy of this model is **96.3%,** precision of **94.9%** and recall of **97.8%**.
For stage 2 which is the Damage Localization again Densenet outperformed the rest of the models with an accuracy of **76.5%**, precision of **76.8%** and recall of **74.4%**. For stage 3 Resnet perfromed better than rest of the models with an accuracy of **67.8%,** precision of **68.5%** and recall of **67.3%.**


Vgg16

	Model	Accuracy	Precision	Recall
0	Vgg16_stage1_baseline	0.933	0.938	0.926
1	Vgg16_stage2_baseline	0.737	0.746	0.730
2	Vgg16_stage3_baseline	0.661	0.650	0.652
3	Vgg16_stage1 FC	0.930	0.919	0.943
4	Vgg16_stage2 FC	0.709	0.717	0.707
5	Vgg16_stage3 FC	0.643	0.650	0.637
6	Vgg16_stage1 all	0.939	0.935	0.943
7	Vgg16_stage2 all	0.693	0.712	0.678
8	Vgg16_stage3 all	0.608	0.608	0.587

Vgg19

	Model	Accuracy	Precision	Recall
0	Vgg19_stage1_baseline	0.930	0.927	0.935
1	Vgg19_stage2_baseline	0.732	0.746	0.730
2	Vgg19_stage3_baseline	0.656	0.641	0.646
3	Vgg19_stage1 FC	0.926	0.930	0.922
4	Vgg19_stage2 FC	0.693	0.710	0.678
5	Vgg19_stage3 FC	0.637	0.633	0.626
6	Vgg19_stage1 all	0.946	0.944	0.948
7	Vgg19_stage2 all	0.693	0.712	0.678
8	Vgg19_stage3 all	0.608	0.608	0.587

Densenet

	Model	Accuracy	Precision	Recall
0	Densenet_stage1_baseline	0.880	0.875	0.887
1	Densenet_stage2_baseline	0.598	0.590	0.579
2	Densenet_stage3_baseline	0.550	0.540	0.545
3	Densenet_stage1 FC	0.893	0.888	0.900
4	Densenet_stage2 FC	0.581	0.583	0.564
5	Densenet_stage3 FC	0.573	0.578	0.569
6	Densenet_stage1 all	0.959	0.949	0.970
7	Densenet_stage2 all	0.804	0.807	0.789
8	Densenet_stage3 all	0.696	0.692	0.685

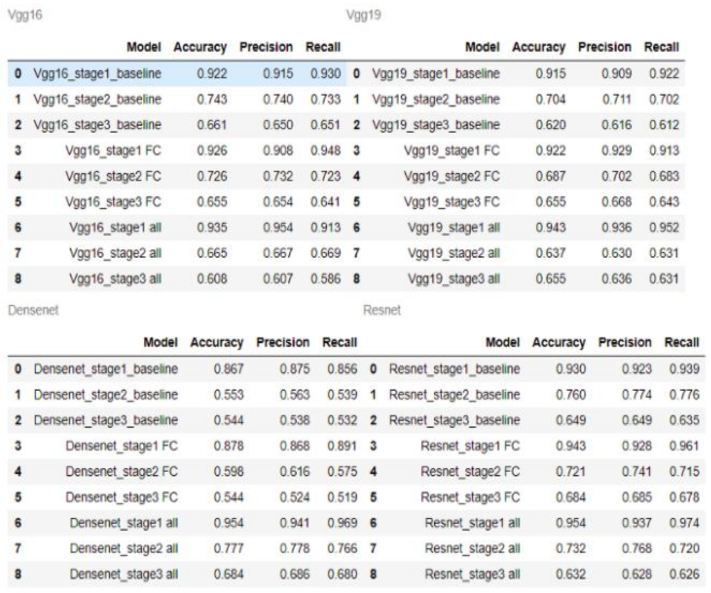
Resnet

	Model	Accuracy	Precision	Recall
0	Resnet_stage1_baseline	0.930	0.923	0.940
1	Resnet_stage2_baseline	0.782	0.791	0.779
2	Resnet_stage3_baseline	0.637	0.633	0.621
3	Resnet_stage1 FC	0.933	0.938	0.926
4	Resnet_stage2 FC	0.771	0.788	0.759
5	Resnet_stage3 FC	0.643	0.628	0.630
6	Resnet_stage1 all	0.961	0.945	0.978
7	Resnet_stage2 all	0.749	0.762	0.740
8	Resnet_stage3 all	0.643	0.635	0.633

Fig 21. Original Data + Data Augmentation 1

On the original data with the first data augmentation, we observe that resnet and densenet trained on all layers performed better than other models. On the resent model we get an accuracy of 96.1% but precision is lower than densenet model.For densenet model the accuracy is **95.9%**, precision is **94.9%** and recall of **97.8%**. For damage localization

Densenet performs better than other models with an accuracy of **80.4%,** precision of **80.7%** and recall of **78.9%.** Resnet perfromed better than other models for Damage Severity with an accuracy of **69.6%**, precision of **69.2%** and recall of **68.5%**.



Vgg16

	Model	Accuracy	Precision	Recall
0	Vgg16_stage1_baseline	0.922	0.915	0.930
1	Vgg16_stage2_baseline	0.743	0.740	0.733
2	Vgg16_stage3_baseline	0.661	0.650	0.651
3	Vgg16_stage1 FC	0.926	0.908	0.948
4	Vgg16_stage2 FC	0.726	0.732	0.723
5	Vgg16_stage3 FC	0.655	0.654	0.641
6	Vgg16_stage1 all	0.935	0.954	0.913
7	Vgg16_stage2 all	0.665	0.667	0.669
8	Vgg16_stage3 all	0.608	0.607	0.586

Vgg19

	Model	Accuracy	Precision	Recall
0	Vgg19_stage1_baseline	0.915	0.909	0.922
1	Vgg19_stage2_baseline	0.704	0.711	0.702
2	Vgg19_stage3_baseline	0.620	0.616	0.612
3	Vgg19_stage1 FC	0.922	0.929	0.913
4	Vgg19_stage2 FC	0.687	0.702	0.683
5	Vgg19_stage3 FC	0.655	0.668	0.643
6	Vgg19_stage1 all	0.943	0.936	0.952
7	Vgg19_stage2 all	0.637	0.630	0.631
8	Vgg19_stage3 all	0.655	0.636	0.631

Densenet

	Model	Accuracy	Precision	Recall
0	Densenet_stage1_baseline	0.867	0.875	0.856
1	Densenet_stage2_baseline	0.553	0.563	0.539
2	Densenet_stage3_baseline	0.544	0.538	0.532
3	Densenet_stage1 FC	0.878	0.868	0.891
4	Densenet_stage2 FC	0.598	0.616	0.575
5	Densenet_stage3 FC	0.544	0.524	0.519
6	Densenet_stage1 all	0.954	0.941	0.969
7	Densenet_stage2 all	0.777	0.778	0.766
8	Densenet_stage3 all	0.684	0.686	0.680

Resnet

	Model	Accuracy	Precision	Recall
0	Resnet_stage1_baseline	0.930	0.923	0.939
1	Resnet_stage2_baseline	0.760	0.774	0.776
2	Resnet_stage3_baseline	0.649	0.649	0.635
3	Resnet_stage1 FC	0.943	0.928	0.961
4	Resnet_stage2 FC	0.721	0.741	0.715
5	Resnet_stage3 FC	0.684	0.685	0.678
6	Resnet_stage1 all	0.954	0.937	0.974
7	Resnet_stage2 all	0.732	0.768	0.720
8	Resnet_stage3 all	0.632	0.628	0.626

Fig 22. Original Data + Data Augmentation 2

With the second augmentation we observe that the Resnet and Densenet trained on all layers performed better than the other models. The accuracy of the Resnet model is **95.4**% but the precision is lower than the densenet model. The Densenet model performs better for the damage localization with an accuracy of **77.7%,** precision of **77.8%** and recall of **76.6%.** Resnet performs better than other models with an accuracy of 68.4% on damage severity.

6. CONCLUSION

We started by exploring the applicable deep learning algorithms for the car damage detection and also created new datasets which provided us to explore the detection, classification and the severity of the damaged cars. The pre-trained models were experimented by fine-tuning and transfer learning with certain regularization techniques. From the above models we can safely conclude that Resnet model works best to detect whether a car is damaged or not, YOLO models to identify the car damage classification and the densenet model to check the severity of the car damage. Regarding the proposed models there are still overfitting issues but there is still room for improvements in terms of accuracy. In addition to that if we have a proper high-quality dataset with adequate features and labels we can also try to predict the cost of repairing for the damaged car part and that would help the auto-insurance industry to make better and cost-effective solutions.

REFERENCES

1. "https://www.irmi.com/articles/expert-commentary/controlling-claims-leakage-through-technology."
2. Jeffrey de Deijn. 2018. Automatic Car Damage Recognition using Convolutional Neural Networks. (2018).
3. B. Y. Lecun Y., Bottou L. and H. P., "Gradient-based learning applied to document recognition," Proceedings of IEEE, vol. 86, no. 11, 1998.
4. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.
5. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778.
6. Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional network for large-scale image recognition arXiv preprint arXiv:1409.1556(2014).
7. Ranjodh Singh, Meghna P Ayyar, Tata Sri Pavan, Sandeep Gosain, and Rajiv Ratn Shah. 2019. Automating Car Insurance Claims Using Deep Learning Techniques. In2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM).IEEE, 199–207.
8. M. G. M. R. G. Soumalya Sarkar, Kishore K. Reddy, "Deep learning for structural health monitoring: A damage characterization application," in Annual Conference of the Prognostics and Health Management Society, 2016.
9. S. Gontscharov, H Baumgartel, A.Kneifel, and K.-L. Krieger, Algorithm development for minor damage identification in vehicle bodies using adaptive sensor data processing," Procedia Technology, vol. 15, pp. 586 { 594,2014. 2nd International Conference on System-Integrated Intelligence: Challenges for Product and Production Engineering.
10. G. Wang and S. Liang, ''Ship object detection based on mask RCNN,'' in Proc. Radio Eng., 2018, pp. 947–952
11. J. Li and W. He, ''Building target detection algorithm based on mask RCNN,'' in Proc. Sci. Surv. Mapping, Apr. 2019, pp. 1–13.
12. S. Jayawardena, Image based automatic vehicle damage detection. PhD thesis, College of Engineering and Computer Science (CECS), 12 2013.
13. Mahavir Dwivedi, Malik Hashmat Shadab, SN Omkar, Edgar Bosco Monis, Bharat Khanna, and Satya Ranjan. Deep Learning Based Car Damage Classification and Detection.