Fast Detection and Classification of Vehicle Surface Damages

Project report of AMI Group 06

Zucheng Han	Huiyu Wang	Xin Zhang	Mingcong Li	Wantao Li
03739137	03746516	03739647	03753855	03740431
Fengrui Gan	Runze Li	Xueqing Nie	Yingyi Zhang	Bowen Yuan
03749735	03740923	03746517	03749873	03701634

August 30, 2022

Motivation

With the development of the car rental market, the utilization rate of vehicles in car rental companies stays at a high level. Evaluating the status of the car and detecting whether there are damages on vehicles during the period of the rental is an essential issue for car rental companies, because even minor damages can bring loss to the companies' properties. The task of inspecting the damage of cars is usually done manually and immediately when the automobiles are returned, which takes lots of time. When numerous autos are waiting to be sent back at the same time, customers have to wait for a long time due to the damage examination of each vehicle, which might cause complaints to the car rental companies. Thus, it is vital importance to develop a method to detect car damages faster. With machine learning methods, the surface damages can be recognized and classified quickly and precisely with well-trained model. When the rental cars are returned, quantities of images from different angles will generate automatically. The algorithm will then mark out the potential damage areas and show what the damage types belong to. With the assistance of this program, staffs from car rental companies can avoid unnecessary time for manual checking the exterior of car surface and focus on the marked areas. This method could significantly improve the efficiency during the process when the cars are returned.

---

1 Project Description

This project is developed based on the vehicle damage data of WENN, which focuses on detecting surface damages on cars. The task of this project is to detect whether there is damage on the car's surface from the car photos and classify the damage types. Damages are categorized by "dent", "scratch", "rim" and "other". Pictures from different angles of the car are provided. In around 1000 images, the damaged areas are marked out with a bounding box, and the rest are original photos without any marks. Although the damage categories are defined, the dataset with bounding boxes has no labels. The team members must label the dataset at the beginning and train models with the manually labeled data. The trained models can detect and classify damage in other images.

In this project, different training models, such as ResNet50, CNN, KNN, InceptionV3, etc., are tried in our group, and the models with the best performance are selected out. Besides, active learning is used in this project to help refine the model. In this way, the dataset is expanded and the performance of the model is also improved after some iterations. In the end, docker encapsulates the web equipped with classification algorithms. Then the Website in the docker image is deployed in Kubernetes, which could be accessed by specific IP address and ports.

Research Question

- What is the best model for detecting and predicting the surface damage of vehicles with the its images from different angels?
- How to preprocess the original data and train the model with the preprocessed data?
- How to present the results of our project in form of website and how to make an easy and convenient user interaction interface?

Goals

The goal of our project is to build a website where users can upload one or multiple images, in which the model can predict if the damage belongs to scratch, dent, rim, or others. Besides, the website which refines classification model by active learning should be considered. Moreover, when improving the accuracy and the generalization performance of the model, it is crucial to reduce the training time as much as possible to lower the waiting time of users on the website.

2 Data basis

Data is the most fundamental component of our analysis. Data quality is one of the most critical factors affecting the final performance of the analytical results. For the vehicle damage pictures, the measurements of the data quality mentioned here are completeness, clarity and uniqueness. In this chapter, we will interpret the entire data preprocessing workflow that allows us to obtain a dataset with complete, clear and unique damage images of vehicles.

Sources and Data Collection

Our industry partner WENN provides us with various images that are collected and divided into small datasets. The pictures are taken from different angles and not every image contains damage. Thus, the first step is to capture and crop the damage.

Preprocessing

The goal of preprocessing is to handle the input image data in our project for later training process. Preprocessing can remove irrelevant information from images and restore useful information, improving recognition reliability. After going through a grayscale transformation, geometric transformation, and noise removal, the images serve as the foundation for the subsequent classification's reliability.

Labelling

For a classical classification task, the labeling of the data is always necessary. However, the difficulty in this project is that all the images are unlabeled. Therefore, our idea is to train the model with a limited amount of labeled data and then use active learning to find the most special image from the unlabeled data and label them in order to improve the classification effect.

In the labeling process, many images contain two or more types of damage and sometimes distinguishing between dent, scratch, and other damage is hard. After obtaining some training results, we discovered that ambiguity of damage types is a significant issue for the training model.

3 Data Model

During the whole project, our group tried both general classification and deep learning methods such as logistic regression, support vector machine, VGG16, ResNet50 and so on to predict the labels of damage. Then the prediction performances of all methods are evaluated. In our project, InceptionV3 are decided as our final algorithm. For comparison, we also reserve the opportunities for users to use ResNet18 for prediction. Moreover, active learning approach is used in our project to smartly select out images that need to be manually modified and thus, improve the performances of our training model. Active learning is suitable to use if a large amount of data needs to be labelled when a small amount of labelled data is available and it consists of 3 steps:

1. obtain more uncertain sample data for classifying
2. manually review the data
3. manually labelled data need to be retrained by the model

In this section, we briefly introduce the approaches that we tried during our project and the training and evaluation results of some approaches are also shown in their corresponding subsections.

General classification approaches

In general, sophisticated machine learning algorithms powered by cutting-edge approaches like convolutional network have better performances in classification. Some machine learning algorithms are used in our project as benchmarks such as KNN, Logistic Regression, SVM, and Random Forest. For datasets, the ratio of the training dataset to the test dataset is 8 to 2. The best combination of parameters is found out by the GridSearchCV function and then the model is tested on test dataset and documented for future reference.

In addition, some real-world related issues happened when we test the conventional algorithm. For example, KNN method has a poor performance and the logistic regression and RandomForest algorithms have excessively long calculation time when there are many hyperparameters. Therefore, we develop an autoencoder model to increase the calculation speed of machine learning algorithms and enhance the performance of traditional machine learning methods. The autoencoder model decreases the dimensionality of the image to extract corresponding features and does not utilize a convolutional technique.

K- Nearest Neighbor, Logistic Regression, SVM and Random Forest

The K-Nearest Neighbor algorithm is a common benchmark algorithm because it serves as the standard non-parametric classification approach. The value of one data point is decided by its intermediate neighbors[4]. Prior to normalization in KNN, the input image data is flattened by using the smallest Euclidean distance as the criterion to compare the neighbors in various input image datasets.

Although categorizing car damage is more like a nonlinear problem, we still use linear logistic regression for a reference. Additionally, We flatten and normalize the image data. This process is similar to the data processing in KNN. By multiplying the input image pixels with the correlation

coefficient, a polynomial equation is created, which can be used to identify the damage types. For support vector machine, it operates effectively in nonlinear classification. The preprocessed inputs can also be implicitly mapped to initial images with high-dimensional feature spaces[7]. As a result, the conclusion could be drawn that SVM can perform better compared to logistic regression.

Random forests can achieve the highest maximum accuracy compared to other classification techniques. It is able to quickly handle enormous input data with many variables and automatically balance the dataset when it contained certain uncommon labels. The four traditional machine learning methods above have the same process of normalization on the input data, which makes it easier to compare the effectiveness of traditional machine learning algorithms.

Autoencoder

When large-scale images are used as input, reduction of dimensionality is important in reducing the computation time and increasing accuracy and it extracts features in the damage images. The features are utilized as the input of the classification models. After reducing the dimensionality, the images classified as "other" will be randomly distributed to other three damage types, which lowers the model's accuracy.

The popular techniques of nonlinear dimensionality reduction such as TSNE are inadequate to meet our goal. In order to reduce the input data into the data with 64-dimensional features, we create our own autoencoder model. Our autoencoder model is created without convolutional layers to represent the limitation of conventional machine learning algorithms.

Autoencoder and TSNE can correct wrong labels

Due of the different definitions of damage types, we confuse the image labels of dent and scratch when we label the images initially. Moreover, reduction of dimensionality helps us to review and correct labels of dataset in early stage. This tells us the reason why the model has difficulty distinguishing the difference of different damage types. The TSNE dimensionality reduction approach is a solid choice for nonlinear problems. Nevertheless, TSNE dimensionality reduction is unable to extract features, classify the damage and calibrate the images in the initial dataset. The TSNE approach is then utilized to reduce secondary dimensions with the help of our own autoencoder model and in this way, We are able to achieve good classification results.

As Fig.1 shows, there are clear differences between "rim" and other damage types and "dent", "scratch" and "other" are overlapped with each other. This is similar to the situation that we encountered in our project. "Rim" is the easiest damage type to obtain the correct prediction results and "dent", "scratch" and "other" are difficult to be distinguished with each other.

However, when the high dimensional image data is reduced to two dimensions, much information in the images are lost. For instance, irrelevant data points may overlap with "dent" and "scratch" data points. Although the accuracy of damage classification for 2D data is low, incorrect labels can be identified with this method. Approximately 30 wrong labels are corrected with this method.

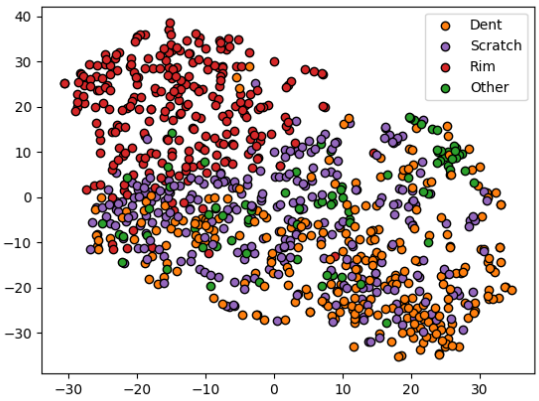


Figure 1: Autoencoder and tsne

Result and Restriction to the conventional algorithm

From our opinion, the random forest algorithm has the highest accuracy in four traditional algorithms without autoencoder. The accuracy of KNN is low and logistic regression requires a long computation time. In contrast, the application of autoencoder in dimensionality reduction improves the accuracy of KNN and reduces the computation time of logistic regression and random forest significantly. And the classification accuracy of SVM models are also improved as Fig.2 shows.

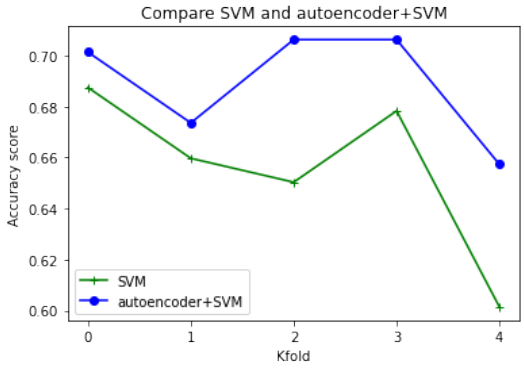


Figure 2: Autoencoder improves the performance of SVM

Although autoencoder can significantly improve the accuracy of traditional machine learning models when the initial classification accuracy is low, the accuracy can hardly be improved when it reaches around 70%. The reason is that traditional machine learning algorithms rarely perform convolution operations to extract local features of objects, and the efficiency of summarizing and extracting image feature information is low.

Deep learning classification approaches

Traditional feature extraction techniques are application-specific and rely heavily on manually created extractors and have limited generalization ability and robustness. In contrast, deep learning is a method for data-driven feature extraction, which learns from a lot of samples to obtain representations of deep and dataset-specific features. Therefore, in order to increase the accuracy after utilizing conventional methods, we try the application of deep learning methods such as CNN, VGG16, ResNet50, InceptionV3 and MobilenetV2.

CNN and Recurrent CNN

At the beginning of this project, we used a simple CNN model to obtain a basic classification among the dataset, but the classification results of CNN did not meet our expectations [6]. Recurrent CNN is based on the simple CNN method and the structure is simple. Fig.3 shows that recurrent CNN is constructed with one or multiple layers of CNN, a recurrent layer and a time-dependent dense layer. The image analysis provided by the Recurrent CNN is similar to the simple CNN. However, the recurrent layer provides a recursive effect, which improves the results based on the previous images recursively.

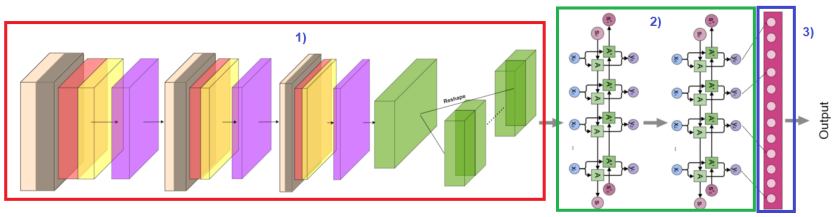


Figure 3: Recurrent CNN

VGG16

The model VGG16 outperforms other models with its low demand for computation resources. This model is expected to have a high fitting ability with many parameters. However, the training time of VGG16 is excessively long and tuning the parameters is difficult. Moreover, a large storage capacity is required for VGG16 and this makes it inconvenient to deploy [9]. From Fig.4, it can be seen that the final accuracy of our trained VGG16 is 0.77.

InceptionV3

GoogleNet, also known as InceptionV1, widens the neural network. In addition, inputs of InceptionV1 are concatenated after performing numerous convolutional calculations. In several perspectives, InceptionV2 has a better performances than InceptionV1 because InceptionV2 has a BN layer, but InceptionV1 does not. On the other hand, the model is initially normalized during input calculation in InceptionV2.

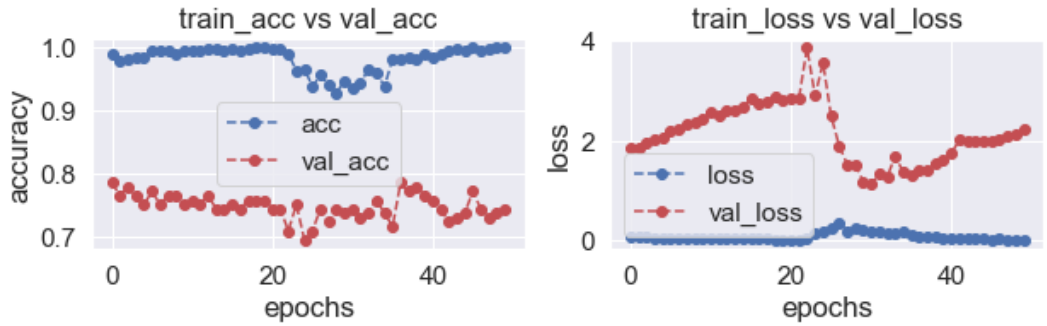


Figure 4: VGG16 accuracy and loss

Inceptionv3 is developed from InceptionV2 and divides a huge convolution into smaller convolutions. It is a 48-layer deep pre-trained convolutional neural network that has already been trained on over a million images from the ImageNet database. As a result, the network has learned detailed feature representations for a diverse set of images. InceptionV3 also requires less computation parameters and information while maintaining the perceptual field. It is designed to compute the convolution result and the pooling result from the inputs. The convolution result is then concentrated with the pooling result because the maximum pooling layer in downsampling leads to a significant loss of information [10].

ResNet50

As Fig.5 shows, input data is processed with 5 stages in a ResNet50 model. The stage 0 has a simple structure and is regarded as the preprocessing for input data and the next four stages have similar structures. There are two basic blocks named Conv Block. Conv Blocks have different dimensions of input and output, so these two blocks cannot be directly connected in series, and they are used to change the dimensions of the network. In contrast, the input and output dimensions of Identity Blocks are the same and they can be connected in series to deepen the neural network [2].

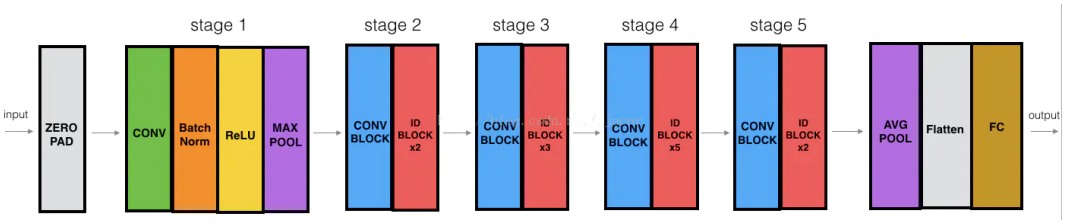


Figure 5: ResNet50

A simple demonstration of this method is shown below. If the shape of the image is (C, H, W) and the input is x, the three convolution blocks on the left side of the Identity Block are the function F(x). The two are summed as F(x)+x, and then it is processed with the ReLU activation function to obtain the outputs of the Identity Block. The output image still has the shape of (C, H, W) and this shows

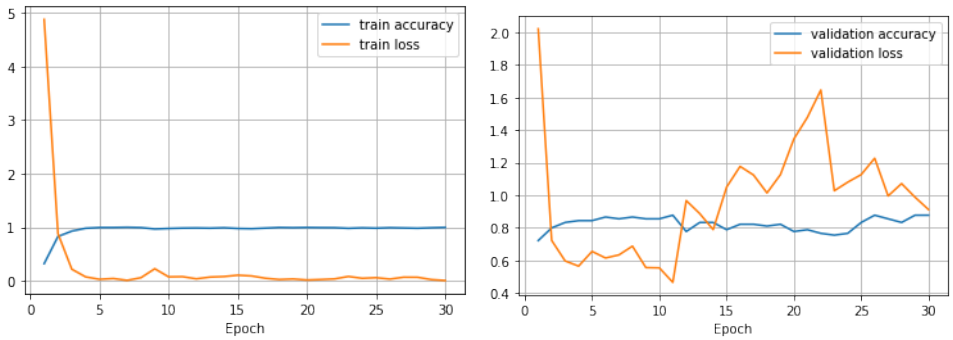
that the Identity Block corresponds to the case where the input x has the same number of channels as the output F(x).

The Conv Block has one more convolution layer on the right side compared to the Identity Block, which represents the function G(x). The Conv Block corresponds to the case where the number of channels of input x and output F(x) is different, and this convolution layer transforms x into G(x), which serves to match the difference between input and output dimensions. Thus, the summation G(x)+F(x) can be performed. Because CNN finally converts the input image into a tiny but deep feature map, the output channel increases when the model learns more complex data with a unified smaller kernel and deeper network. Therefore, it is necessary to use a Conv Block to transform dimensions before the data enters the Identity Block and the Identity Block can be successively connected later. A conv2D layer are added in the shortcut path and changes the dimension in the main path, which is in line with the dimension in the shortcut path.

The dataset is spilted as follows, 80% of data for training, 10% of data for validation and the rest data for test. The training accuracy and loss are shown in Fig.6(a) and the validation accuracy and loss are shown Fig.6(b). The test accuracy of this model on test dataset is about 0.78. Furthermore, the parameters for this model are listed in Table 1.

Table 1: Hyperparameters for ResNet50

Setting	Type/Value
Optimizer	Adam
Learning Rate	0.005
Epochs	10
Batch Size	64



(a) Accuracy and Loss of Training Set

(b) Accuracy and Loss of Validation Set

Figure 6: Accuracy and Loss of ResNet50

ResNet18

To extract critical features from training data and discover significant patterns, network depth is crucial for solving complicated image analysis problems using deep learning. However, adding

additional neural network layers can be computationally expensive and challenging because of the gradients. Using shortcuts or skip connections to skip over some layers, a residual network, or ResNet for short, is an artificial neural network that aids in constructing deeper neural networks. Besides, deeper network layers can be built without running into the issue of disappearing gradients.

ResNet comes in a variety of iterations, such as ResNet18, ResNet34, ResNet50, and so forth. Despite the fact that their architectures are the same, 18,34 and 50 indicate the number layers. To avoid overfitting or failing to generalize the model on our picture dataset, we use ResNet18 as our final model for training active learning because its complexity is halfway between ResNet50 and basic CNN. The Fig.7 shows how residual block look and what is inside these blocks. Identity Block and Convolution Block were explained in ResNet50. Compared to ResNet50, ResNet18 is missing the middle three layers in both blocks.

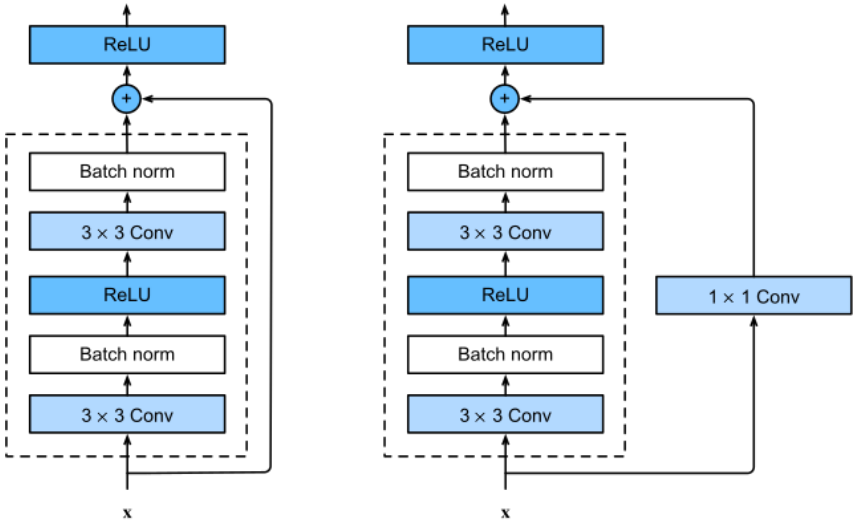
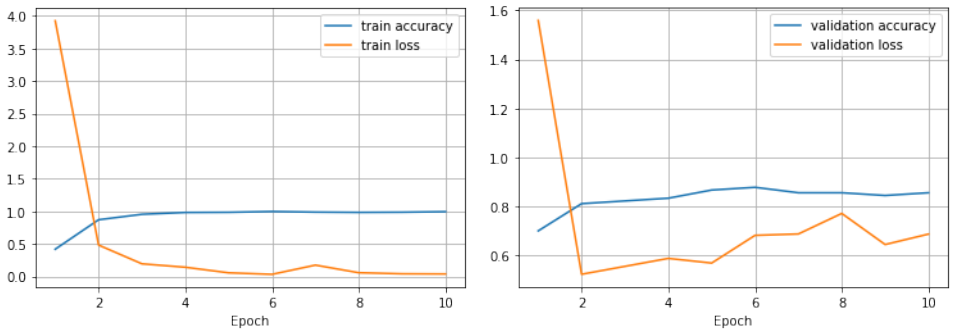


Figure 7: ResNet18 block with and without convolution

DenseNet201

In a traditional convolutional neural network, if there are L layers, then there will be L connections. In contrast, there are $L(L+1)/2$ connections in DenseNet. One advantage of DenseNet is that the network is narrower and has fewer parameters. The connection in DenseNet makes transferring features and gradients more efficient and it is easier to train the neural network. In addition, DenseNet works well in classification because each layer has direct access to the gradients from the loss function and the original input signal, leading to the implicit deep supervision. The dense connection is equivalent to directly connect input and loss in each layer and mitigates the gradient vanishing problem. Besides, a dense connection has the effect of regularization and it has effects on overfitting [3]. In our project, the batch size is set to 16, which is half as small as ResNet50 to reduce the use of GPU and the final test accuracy is 0.81.



(a) Accuracy and Loss of Training Set (b) Accuracy and Loss of Validation Set

Figure 8: Accuracy and Loss of DenseNet201

EfficientNetB7

When neural network models are too wide or too deep, adding features into the model initially can help build and enhance the performances of the training models. However, the models become saturated and contain more parameters in short time and it is inefficient for model training. In EfficientNet, the features are added gradually in a more efficient way. To increase the complexity and the performance of the model, we scale the model properly [11].

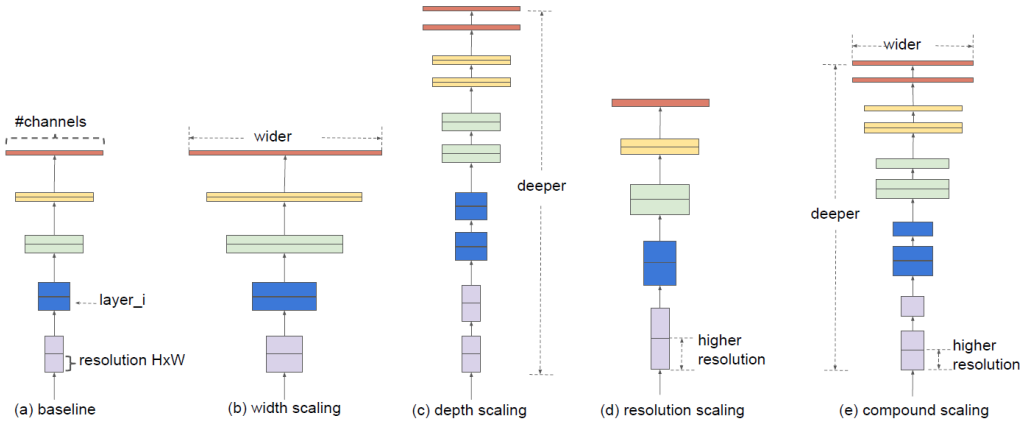
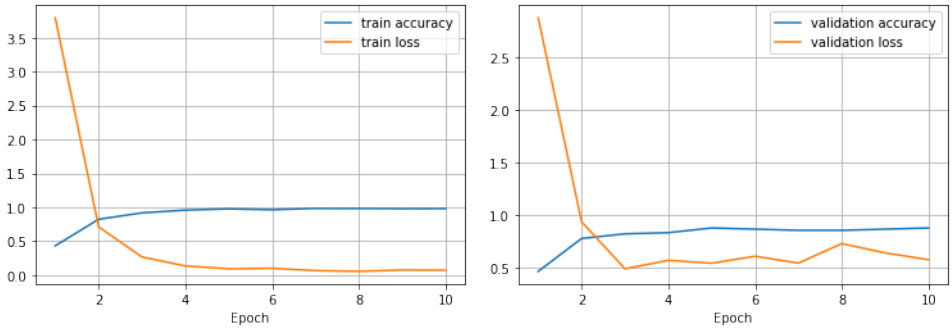


Figure 9: EfficientNet Model Scaling

The commonly used scaling methods are shown in Fig.9. (a) is a baseline method. (b)-(d) are traditional scaling methods that increase one dimension of the network's width, depth or resolution. (e) is a composite scaling method, which scales all three dimensions uniformly with a fixed scale. When the depth of the network increases, the width of the model and the resolution of the input image data need to increase to get the best classification results. When the depth increases, a higher input image resolution is needed to ensure the validity of the perceptual field and a wider

network to ensure that more features can be captured [1]. In our project, the batch size is set to 16, which is the same as DenseNet201 and the final test accuracy is 0.79.



(a) Accuracy and Loss of Training Set

(b) Accuracy and Loss of Validation Set

Figure 10: Accuracy and Loss of EfficientNetB7

MobileNetV2

MobileNetV2 is a lightweight convolutional neural network based on the module inverted residual with linear bottleneck. This module takes an input as a low-dimensional compressed representation, which expands to a high-dimensional representation. The input is then filtered with a lightweight depth-wise convolution. Features are subsequently projected back to a low-dimensional representation with a linear convolution. Fig.11 shows the architecture of MobileNetV2, and there are two types. One type is residual block with stride one, and the other type is residual block with stride two [8].

This model is popular in classifying images, because it has a lower number of parameters and requires less computation power. This proves that it is suitable for mobile devices and embedded systems. Besides, MobileNetV2 is developed from MobileNetV1 and MobileNetV1 is pretrained for image classifications. Due to the pretraining in a large dataset of images, we can expect high accuracy performances. Based on the analysis, MobileNetV2 is applied to achieve the classification of cropped damage patches in four classes in our project. The damage dataset that we use is cropped, which is beneficial for us to do the classification directly. Besides, the images in the dataset are labeled manually. The validation loss and validation accuracy are shown in Fig.12 and the final test accuracy of this model is 0.829545.

The top-5 accuracy of the model is used to compare the classification performances. The selected three models shown in Table.2 are trained on the same dataset and are compared in size, test accuracy, depth and time. In a nutshell, Inception V3 has the highest Top5-accuracy. The model MobilenetV2's is superior compared to the other two methods. The best time accuracy trade-off is provided by the ResNet 50.

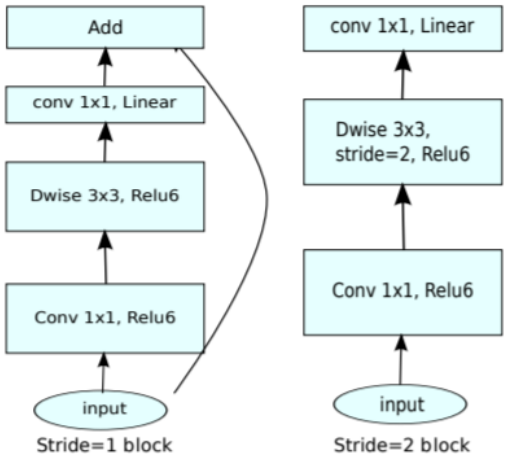
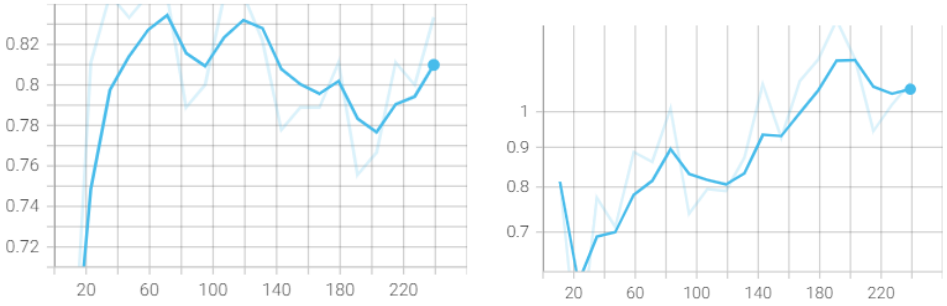


Figure 11: Architecture of MobileNetV2



(a) Validation Accuracy (b) Validation Loss

Figure 12: Accuracy and loss of MobilNetV2

Active Learning Approach

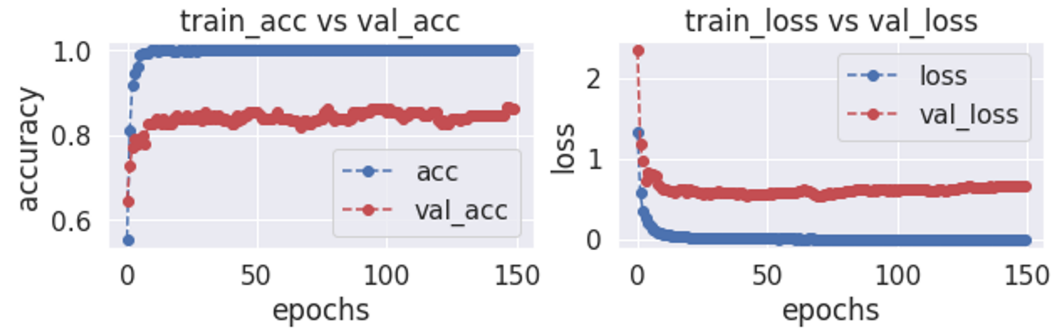
In active learning, the initial labels are applied to develop a training model that serves as a foundation and the developed model is utilized to offer predictions for the unlabeled datasets. The model then scans each of its predictions, identifies the one or a few which are the hardest to classify and asks a label for it from the oracle, that means a human. The selection of the most difficult forecasts depends on if that is done informative and representative. Informative measures of unlabeled instances help reduce uncertainty in statistical models, while representative measures help to improve how well they represent the structure of the input model. To enhance the model, the label is provided by the oracle, and the newly labeled data is then mixed with the previously labeled data. The method is repeated after recording the model's performance. In each iteration, the amount of labeled data increases, and the performance of the model changes [5]. With the help of active learning, not merely has the cost of annotation been reduced, but the effectiveness of the model has been improved.

Table 2: Comparison of the three models

Model	Size(MB)	Top5-accuracy	Depth	Time (ms) per inference step (GPU)
Inception V3	92	93.7	189	6.9
Mobilenet V2	14	90.1	105	3.8
ResNet 50	98	92.1	107	4.6

In active learning, the performance of the trained model could potentially be improved in each iteration of labeling, but there is a trade-off between the model performance and the cost of acquisition. The cost of acquisition may be estimated beforehand and the improvement becomes apparent after an iteration of active learning. Therefore, choosing when to stop this process can be either a static or a dynamic choice. Static criteria include things like a labeling budget and performance target thresholds. Dynamic choices, on the other hand, are based on the trade-off between the acquisition cost and model performance. In the initial iterations, it can be expected to observe a relatively significant increase in performance in each iteration of labeling. As the model become refined, the incremental improvement from new iterations decreases, while the cost in each iteration of labeling remains constant or only minor changes of cost occurs. The active learning process ends when acquiring new labels is no longer beneficial [5].

On our website, users are able to upload annotated damage pictures of vehicles and predict their corresponding damage types in our project. Because of the limited accuracy performances of the trained model, incorrect prediction results can occur. After manually correcting the wrongly labeled images, the performances of the trained model could be further improved with active learning.



InceptionV3 Accuracy InceptionV3 Loss

Figure 13: Accuracy and loss of InceptionV3

4 Results

The prediction results of damage types are presented on the website interface. After the users upload images of the car damage, the website shows users possible predicted damage types. In this section, the performances of model InceptionV3 and ResNet18 with active learning are presented.

Observations

Fig.13 shows the accuracy and loss of the model InceptionV3. After around 10 epochs of training, the accuracy is more than 80% and the loss is less than 1 in the validation.

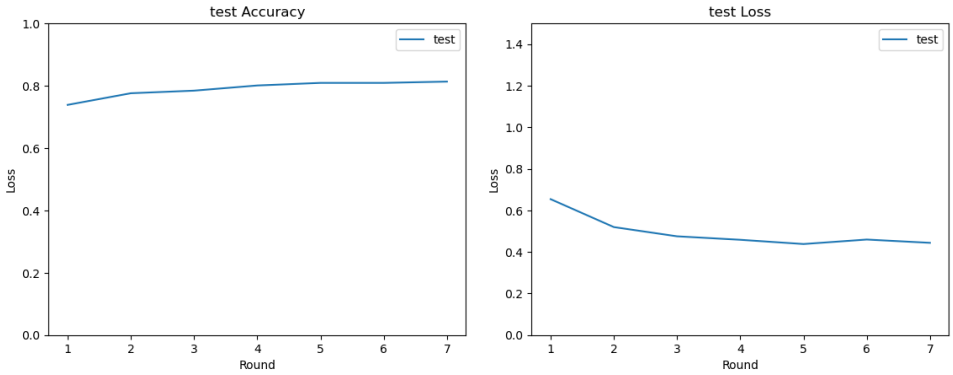
As seen on our website, most of the predicted labels are correct, which proves the accuracy of the InceptionV3. If there is some wrong predictions, users can revise the predicted results manually. When all the damage are correctly predicted and revised, users can download the prediction results as a json file.

Fig.14 shows the accuracy and loss of ResNet18 without active learning in classification. The validation accuracy is roughly around 70% and validation loss is around 0.7. Additionally, ResNet18 has a better performance in test dataset and the accuracy is around 90% and the loss is around 0.3 as Fig.15 depicts.

For active learning with ResNet18, the performances with the sampling methods MarginSamplingSelection, MinStdSelection and EntrophySelection are similar. Fig.16 shows that the initial accuracy can reach up to above 60%. Besides, for their validation sets, the accuracy keeps improving and reaches nearly 100% after 36 rounds of training. In contrast, if the active learning uses random sampling, the accuracy performances in both test datasetset and validation dataset are slightly worse than the previous two sampling strategies. After around 8 rounds of training, the accuracy achieves at around 60%. Unlike the previous two strategies, the validation accuracy does not improve with the increasing training rounds of active learning.

(a) ResNet18 accuracy (b) ResNet18 loss

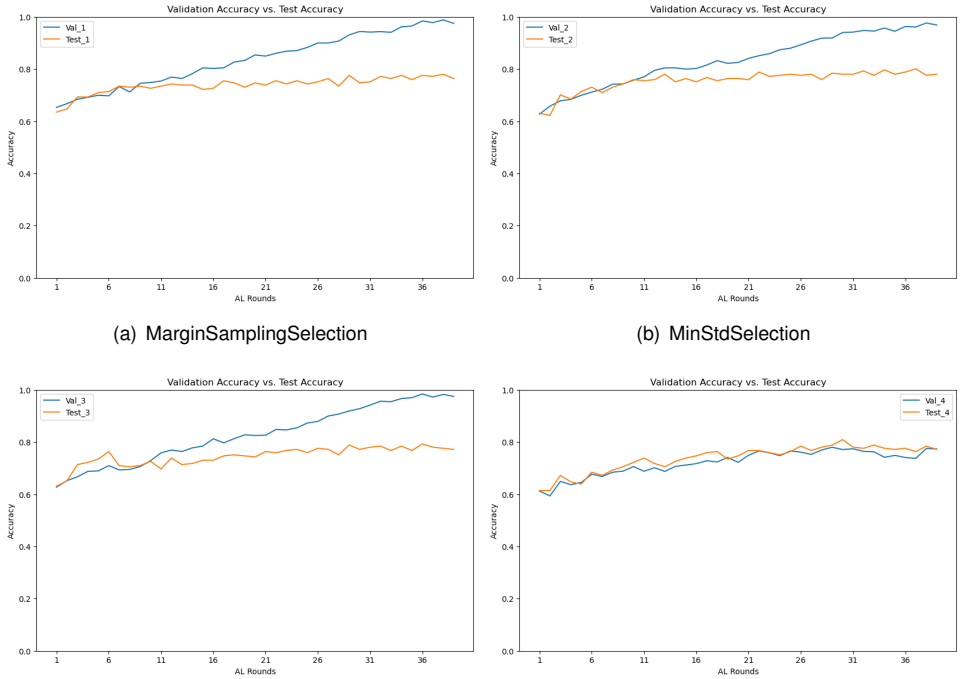
Figure 14: ResNet18 accuracy and loss



(a) ResNet18 accuracy on the test dataset (b) ResNet18 loss on the test dataset

Figure 15: ResNet18 accuracy and loss in test dataset

Fig.17 shows the loss of active learning with different sampling strategies. For MarginSsamplingSelection, MinStdSelection and EntrophySelection, the initial loss in test dataset is a little below 0.6 and continues to decrease to around 0.5 after 36 epochs of training. For their loss in the validation dataset, the initial loss is a little below 1. The loss in validation sets continues to decrease as training epochs increase and reach nearly 0 after around 36 training epochs. In contrast, for random sampling, the loss in testsets and validation sets is similar. The initial loss is above 1 and continues to increase in the first 2 or 3 training epochs. Then, the loss in the testsets decreases to around 0.8 and the loss in the validation dataset decreases to around 0.6 after 36 training epochs.



(a) MarginSamplingSelection

(b) MinStdSelection

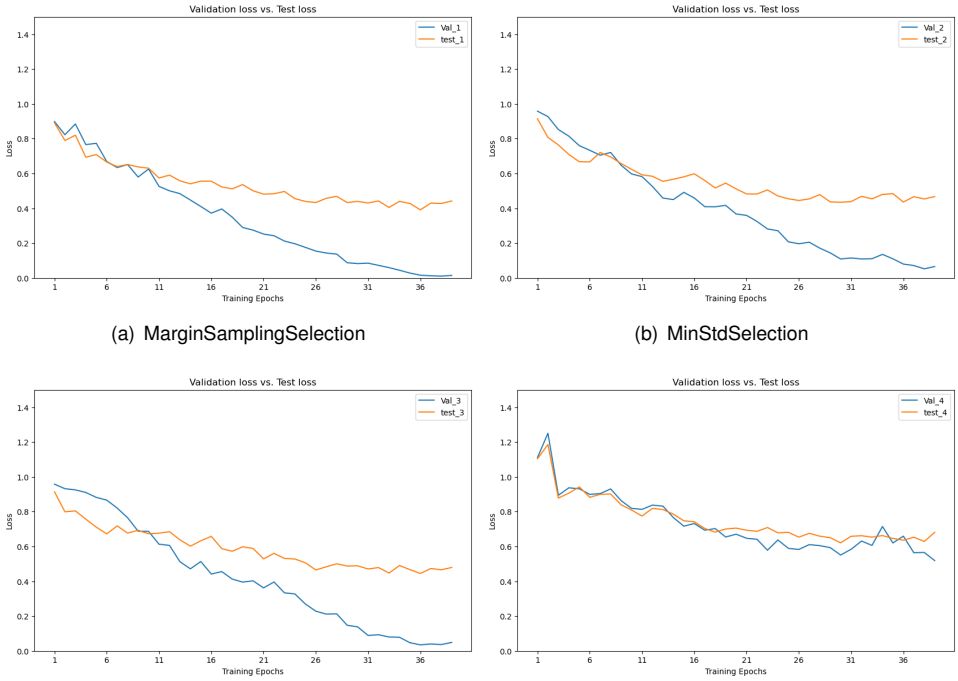
(c) EntropySelection

(d) RandomSelection

Figure 16: Accuracy of active learning with different sampling methods

Trends

InceptionV3 is used to predict the types of damage. Due to the limited size of dataset, active learning with ResNet18 is used to refine the model and expand the training dataset. The performance of InceptionV3 is as good as our expectation, despite some prediction errors, which can be corrected by users. The training time of this model is acceptable, which means that users do not need to wait long to obtain the prediction results and this is important for the users' experiences with the website in the real-life application. In contrast, ResNet18 has a better classification performances in the test dataset and thus, active learning is implemented with ResNet18. The performance of active learning varies with the sampling methods and so far satisfying in our project. With the MarginSamplingSelection, MinStdSelection and EntropySelection sampling strategies, the accuracy in the validation dataset is quite high. In addition, With small number of correct and existed labels, our model can be optimized with the corrected misprediction results through active learning. As discussed above, the accuracy of active learning increases with the increasing epochs of training. When this website operates in real-life scenarios, the model can continuously optimize itself with the correct and expanding datasets.



(a) MarginSamplingSelection

(b) MinStdSelection

(c) EntropySelection

(d) RandomSelection

Figure 17: Loss of active learning with different sampling methods

5 Discussion

In this section, we will discuss the possible reasons of the prediction results from InceptionV3, and the ResNet18 with active learning. Then, we will assess the algorithm from different perspectives. Lastly, we will go through our project's research topic and goal once again.

Interpretation of the results

For InceptionV3, it can be seen from Fig.13 that after around 10 training epochs, the accuracy of this model in the validation dataset reaches above 80% with slight oscillations. With the training dataset, the model shows a satisfying performance in accuracy. On the website, users can revise the inaccurate prediction results. From these findings, it can be concluded that this model can predict the damage types satisfactorily despite some minor errors, which can be corrected manually to provide accurate overall prediction results.

For ResNet18 in classification problems without active learninig, it can be seen from Fig.**??** that the initial accuracy of this model is about 65%. After around 10 training epochs, the validation accuracy reaches 70% with small changes when the training epochs increase. Nevertheless, due

to the better classification performances, we choose ResNet18 to implement the active learning algorithm.

Performance of active learning varies depending on the sampling techniques used. For MarginSamplingSelection, the initial accuracy in both validation dataset and test dataset are roughly 70%. However, as training epochs increase, accuracy in the validation dataset improves. For MinStdSelection and EntropySelection, the accuracy performances are comparable. The initial accuracy in both datasets with both sampling strategies is roughly above 60%, the accuracy in the validation dataset increases when the training epochs increase. After around 36 training epochs, the accuracy in the validation dataset with both sampling strategies continually improves. For random selection, the initial accuracy in the validation dataset and test dataset is around 60%. When the training epochs increase, the accuracy performance in both datasets improves and reaches around 70%. Unlike the other three sampling strategies, the accuracy curve in the validation dataset with this sampling strategy follows the accuracy curve in the test dataset. Although there are slightly differences in accuracy between sampling strategies, active learning's overall accuracy performance can be deemed to be satisfactory based on the results. Besides, when the training epochs increase, the performances of active learning continues to improve. Moreover, the accuracy of active learning model trained with around 255 images in the training dataset reached 81.25% in our project, which meets our expectation. Moreover, the performances of active learning are similar to the performances of ResNet18 discussed above and this proves the effectiveness of active learning in refining the model.

Critical assessment of the results and assumptions

As the accuracy results shown above, the assumptions are matched. In the above subsection, it is shown that the features of different damage types are mixed with each other, which makes it hard for the model to identify and predict. Thus, there are still possibilities that the model's accuracy performance can be improved. In addition, the scarcity of datasets is another factor. The amount of damage images with bounding boxes is only around 900 and is not enough. When new images are uploaded and there are possibilities that wrong predictions results will occur. The two presumptions that we think can affect how well the InceptionV3 model performs.

Since the size of initial dataset is limited for active learning, the effectiveness of this method can still be enhanced with larger dataset. The initial accuracy of all four sampling strategies is around 60% because pre-trained models are used and the models are able to classify the damage types with certain degree of accuracy. When the training epochs increase, the performance can improve because more data is used and the model's performance is enhanced. Nevertheless, the validation accuracy of the model with MarginSamplingSelection, MinStdSelection and EntropySelection will become rather high when the training epochs are large. In each iteration, the remaining validation dataset will be split into new validation dataset and training dataset. The new training dataset is fed to the model and will expand in each iteration. After numerous iterations, the size of the remaining validation dataset will converge to about one third of the original dataset and validation accuracy will be nearly 100%. Fig.18 shows the confusion matrix of active learning in one iteration in our project.

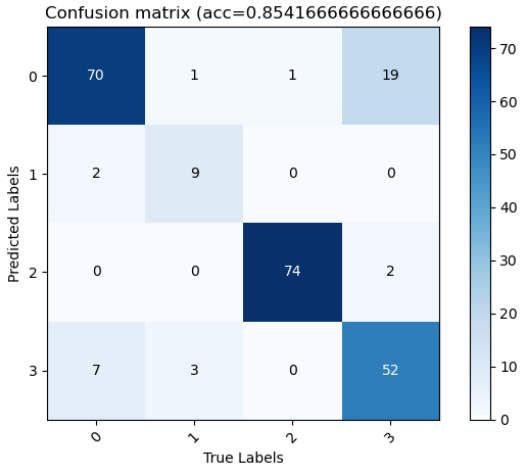


Figure 18: Confusion matrix of ResNet18 with active learning

Proposed Answer to the Research Question

After plenty of different model trails, we choose InceptionV3 and active learning with ResNet18 as our final models integrated on our website. InceptionV3 has a good generalization ability and after around 10 epochs of training, the validation accuracy remains roughly above 80%. In the real-life application on our website, the majority of InceptionV3's prediction results are accurate and a few results need manual correction. Additionally, we also encountered the problem of lack of data during this project. There are only approximately 900 images with bounding boxes of the damage areas that can be used to train our model. Active learning with ResNet18 is used to address this issue. When users upload new images, those images with correct labels can be used to retain the model and the performance of the training model will continuously increase and reach a high level when more people use our website for damage prediction.

For the visualization and interaction parts, our group had many ideas at the beginning of this project. We intended to create an interface, for instance, that displays the potential damage type and the precise location of the damage area. When staffs view the results on our website, it will be simpler for them to identify the damage and assess the condition of the car. However, due to limited time and for the reasons of simplicity and convenience, we choose our current format of website. The website is divided mainly into 4 parts. The "labelling" part introduces what are the four types of damage and how to distinguish them. "Website" part shows users how to use our website. "Classification" part is the main function part. Users can upload their car damage pictures and predict the damage types. Lastly, the "Video" part guide users to the video of our group. Users can easily interact with the algorithm on this website and download the prediction results in one compact json file. This website realizes the easy and convenient interaction between users and our prediction algorithm.

6 Conclusion

In this project, we presented the model of the ResNet18 with active learning and InceptionV3 to predict the surface damage types of cars because of theirs good generalization performances. Active learning in our project is used to expand the training datasets and optimize our model during iterations of model retraining with the mispredicted images.

Summary of the Results

Different models have been tried in this project and they show different prediction performances. Compared with the deep neural networks, the conventional algorithms have shortcomings in accuracy. The conventional algorithms reach the accuracy of around 70% in the validation sets and it is hard to further improve the accuracy. It is mainly because that they have limited efficiency in extracting and summarizing visual feature information and thus, they can rarely employ convolution operations to extract local features of objects. In comparison, the accuracy of InceptionV3 and ResNet18 can reach more than 80% and these two algorithms have satisfactory performances in predicting the types of damage correctly despite some wrong predictions. With the implemented InceptionV3 or ResNet18 algorithm, users can upload damage pictures and download the prediction results easily and conveniently on our website. Besides, with the active learning method, the training model can be optimized continuously and the accuracy of the same model could be improved to some extent, which can be found out when the website is in operation and more images are uploaded and predicted.

For the algorithm active learning with ResNet18, the accuracy performances vary with the sampling stategies. As shown above, the accuracy of active learning with MarginSamplingselection, MinStdslection and Entropyselectrion sampling methods in the validation dataset are around 70% after around 8 training epochs and can reach a rather high accuracy performance when the training epochs are around 30. In our opinion, the possible reason is that the size of the available validation dataset decreases as the training epochs increase. With a larger dataset, the accuracy performances of active learning might achieve a stable state after certain training epochs. In contrast, the accuracy of active learning with Randomselection reaches a steady state at around 70% after 17 training epochs.

For the visualization part in our project, we develop a website for demonstration. With four main function parts, the introduction of labelling and how the classification works are explained in details. Users can upload their own car images of damage and the prediction results can be shown after a short training time. Lastly, users can download the report of prediction results as .json file, which is easier for reviewing and later use.

Future Work

For algorithm part, there are mainly two things that could be improved in the future. Firstly, as mentioned before, the types of "scratch", "dent" and "others" are mixed with each other. This makes it difficult for models to always have correct predictions. Thus, we would like to try more

other models and see whether there are better models for predictions of car damage. Secondly, we would like to improve our data preprocessing. We believed if the data is preprocessed well, a better performance can be achieved with the same model.

For the website design, we would like to improve the user experiences. At present, the website is more function-orientated. More elements can be added to the website to make it more attractive. Besides, there are many links on our website and we can make it more compact by integrating the functions all together and this can be more user-friendly interactive. Furthermore, to make this project more applicable, we would like to add one function that the damage status of car and the location information of the damage area can be displayed in real time. In this way, it will be easier for working staffs from car rental companies to check the status of the rental cars when they are returned.

7 Comments to the Group Work Experience

Our group worked together on this project for around three months to achieve a common goal. Because all of our group members are not so familiar with this topic, we had a hard beginning, but everything went well as time went by. In this project, we learned plenty different classification methods, model training, website design and also the quintessence of teamwork.

To have the best working efficiency of teamwork, we split the work into three major parts in the first place, which is "algorithm and models", "website design and docker" and "report and video". Then, the work was assigned to each group member, and every one was responsible for one or two parts. Because everyone in our group is highly motivated, the project went well throughout this whole semester.

However, there are still some problems in our group work experiences, which could be improved in the future. For example, some group members were not familiar with the implementation of algorithm and the design of website. Thus, some of the group members had to do most of the work and this showed that the work load was not distributed fairly. Nevertheless, the team members were gradually familiar with this project and the work as time went by and every team member could make contributions to this project similarly.

From this experience, we learned mainly two things about group work. Firstly, it is not possible to determine the fixed work for everyone when the project is progress. The workload should be adjusted dynamically because the progress of team members was not always the same. Secondly, it is scientific to assign the work based on the team member's interests. From experiences in this project, every team member did the work that they were interested in and thus, we were highly motivated during the whole project.

References

[1] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[4] James M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, (4):580–585, 1985.

[5] Fast Forward Labs. Learning with limited labeled data. 2019.

[6] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[7] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[10] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[11] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.