Car Damage Detection and Evaluation

Capstone Project

Our Team

Supervisor

Msc. Do Thai Giang

Pham Viet Duong

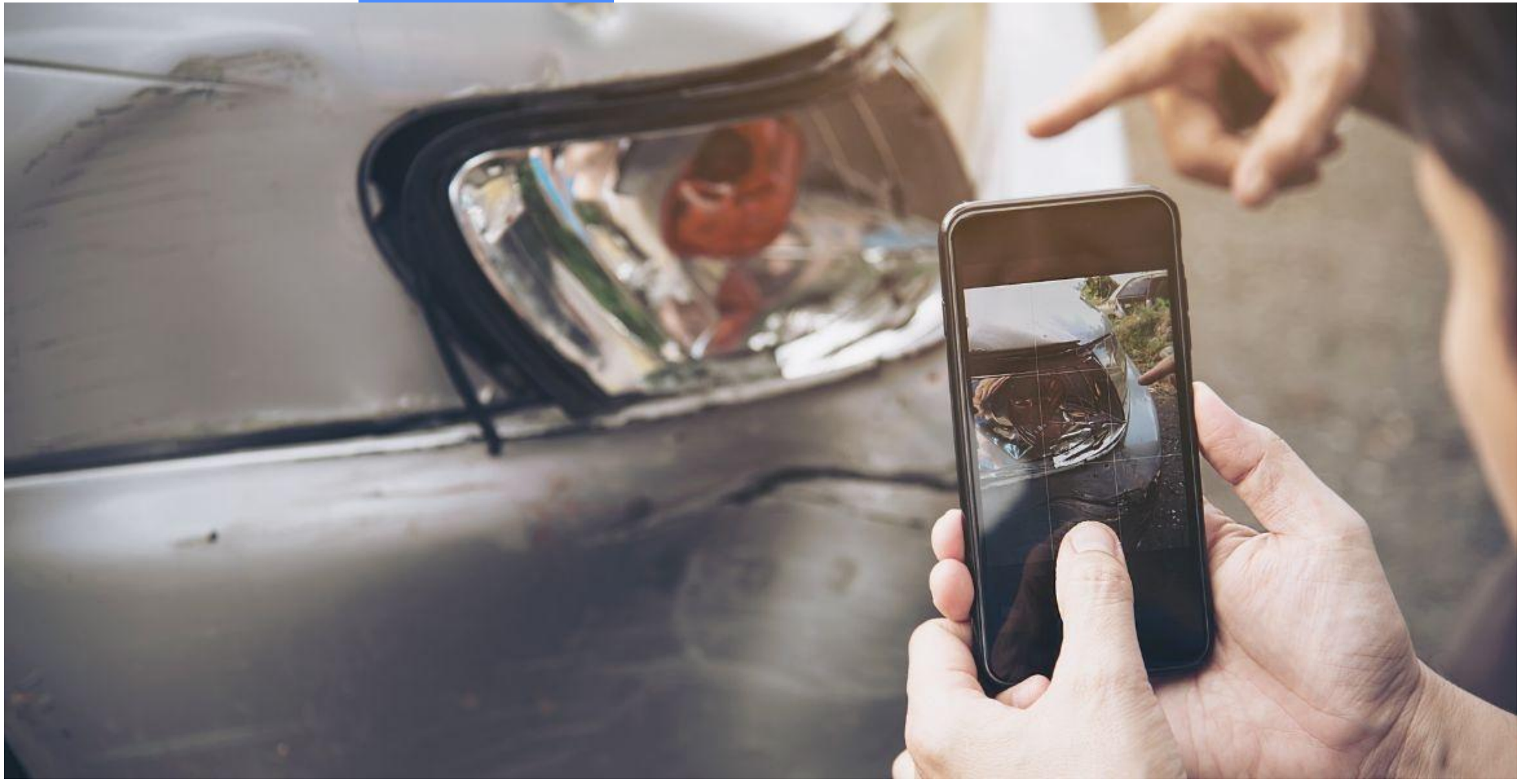
Do Huu Dai

Table of Contents

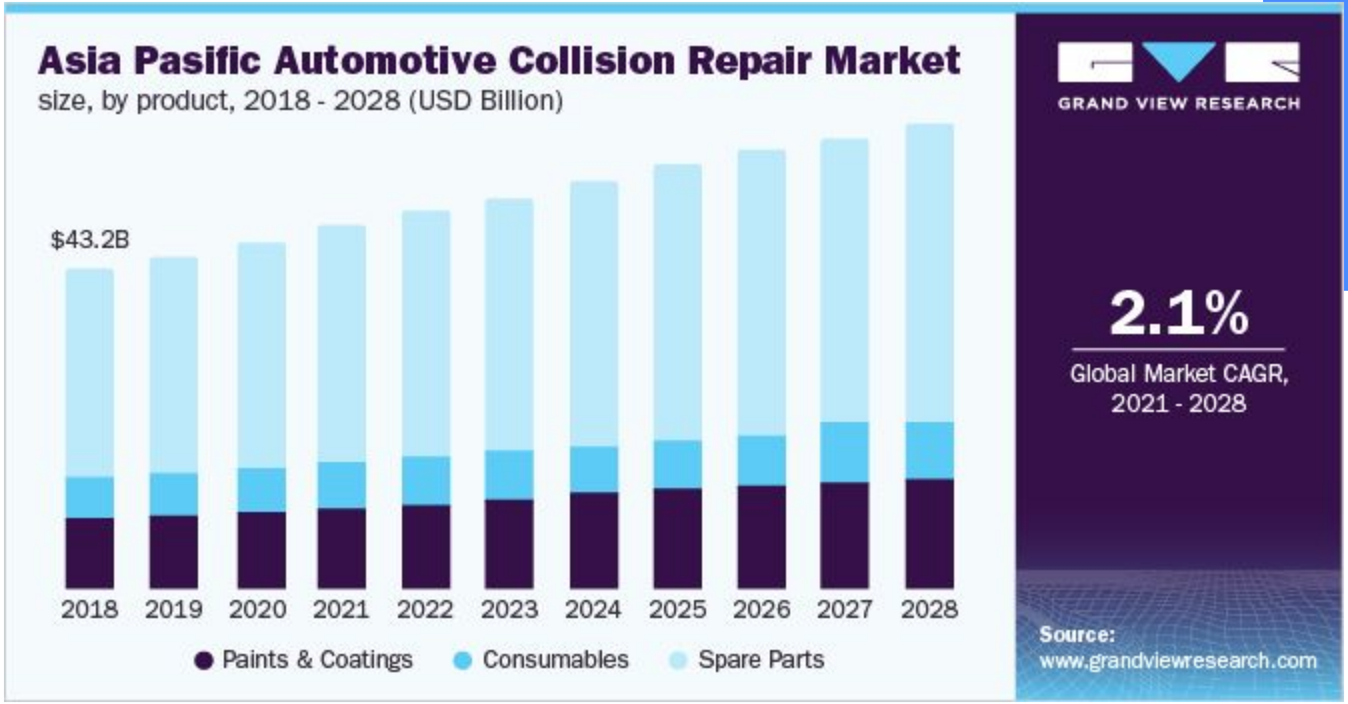
01

Introduction

Introduction



PROBLEM AND MOTIVATION



PROBLEM AND MOTIVATION

Insurance processing takes a lot of time and effort to handle. For example:

- Step 1: Notify relevant parties in the event of an accident.
- Step 2: Assessment of damages and losses.
- Step 3: Conducting car insurance compensation appraisal
- Step 4: Processing car insurance compensation

=> The work process takes a lot of time and effort from everyone involved.

PROBLEM AND MOTIVATION

Recognizing car parts and detecting damages will decrease the steps in the insurance claim processing.

Benefits:

- Identifying the location, evaluation the damaged and classifying them.
- Quickly identify vehicle information

=> This can quickly capture information, accelerate the processing of insurance claims, and provide highly accurate information.

02

Overview

How to solve this problem ?

Image Segmentation

Image Segmentation

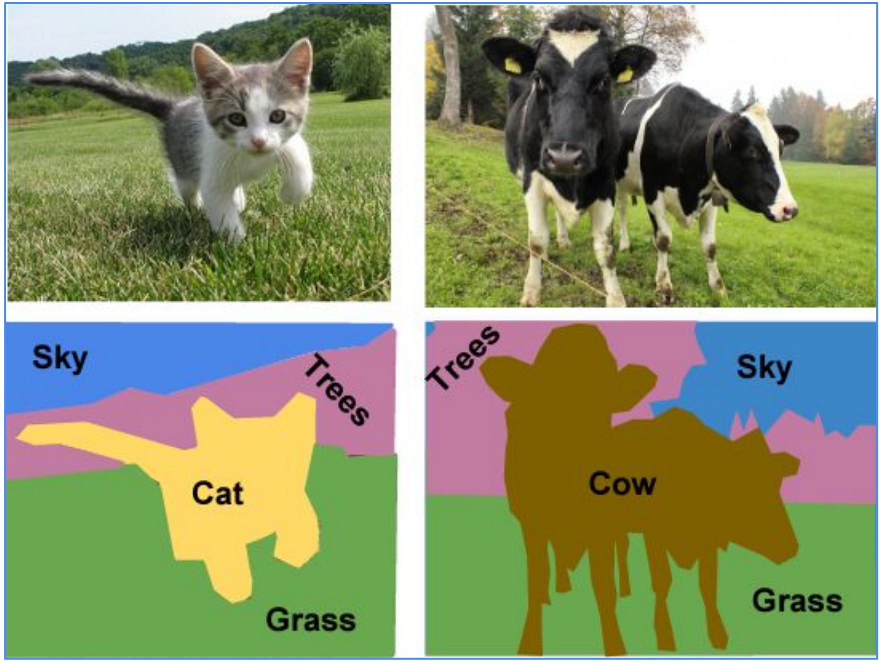
Image segmentation reduces the complexity of a digital image by dividing it into subgroups, or image segments, for further processing or analysis.

Two type of Image Segmentation:

- Semantic Segmentation
- Instance Segmentation

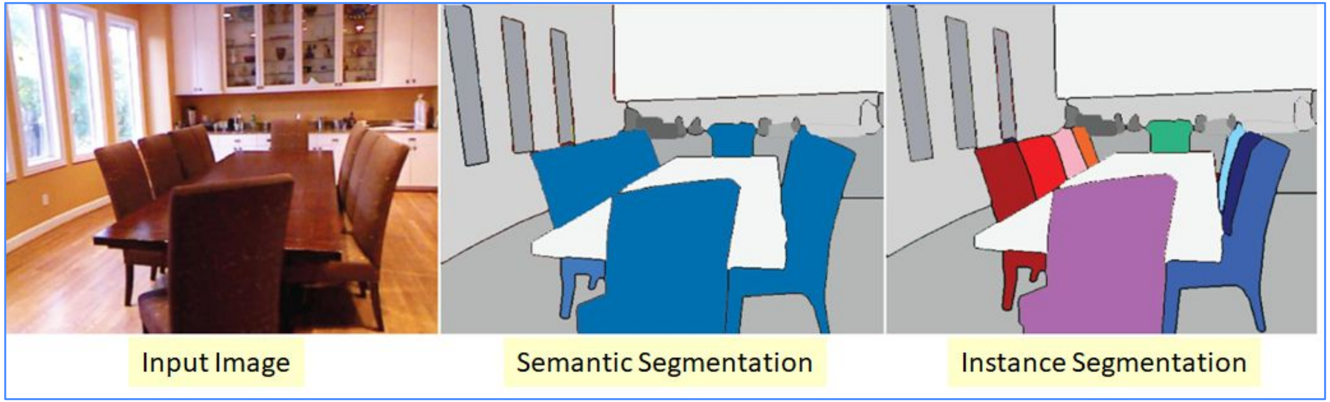
Semantic Segmentation

- **Label each pixel in the image with its respective category.**
- **Ignore distinctions between instances and only consider pixels.**



Instance Segmentation

- Instance Segmentation creates segment maps of each category and each instance of a class, making image inferences more meaningful.
- Instance Segmentation can be referred as a combination of semantic segmentation and object detection.



03

Dataset and Preprocessing

Dataset

Prepare

- Crawl data from multi-source
- Export data from company

Preprocess

- Clean data
- Labeling car-part

Export

- Export data with annotations

Dataset

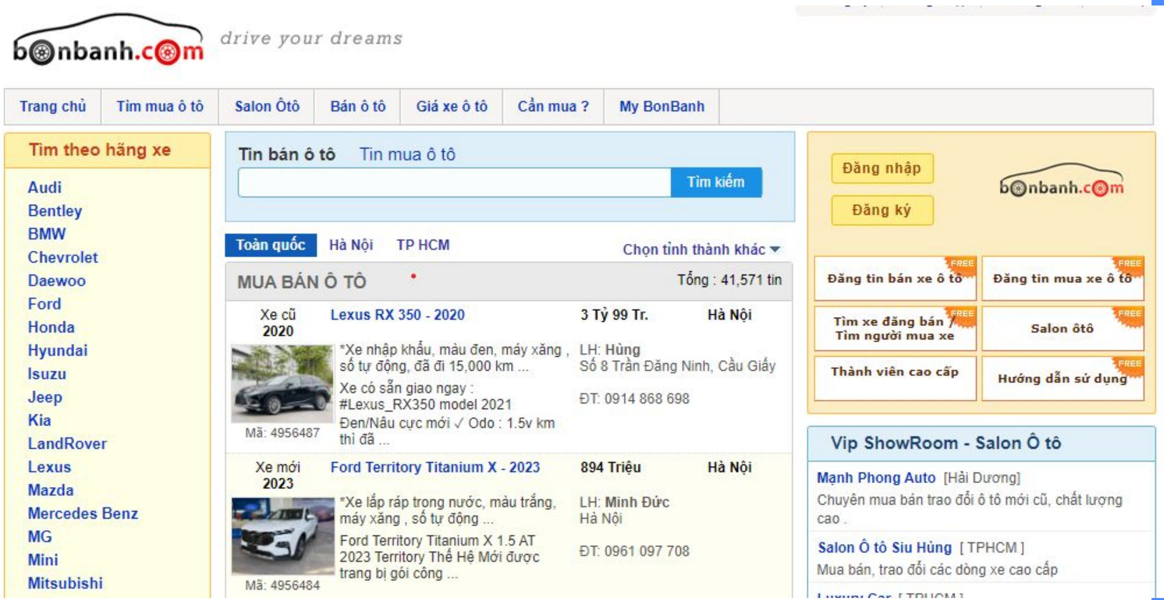
Dataset

- **Available**
  - Available data has been labeled including: images and annotations.
  - This data accounts for about 70% of our dataset.
- **Addition**
  - Crawl data by common method
  - Label car-part to remember and understand car.

Dataset

- **Crawl Data**

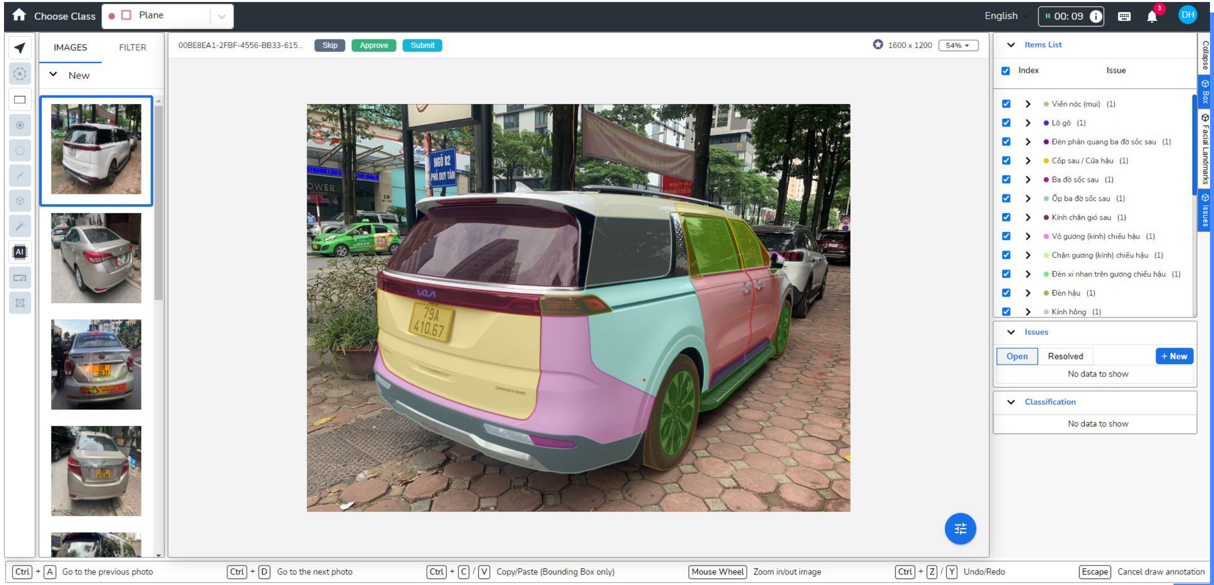
Using Selenium and BS4 to crawl data from Google and various websites, specifically car dealership websites: bonbanh, chotot,...



Dataset

- **Labeling Data**

To represent and serve the training process effectively, we use the Coco Annotations format and labels are annotated using polygons.

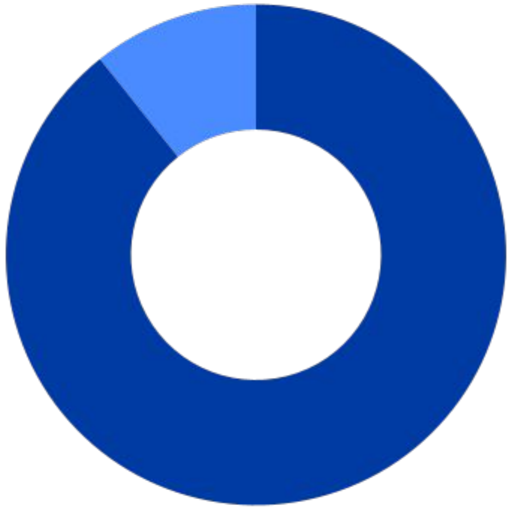


Overview Dataset

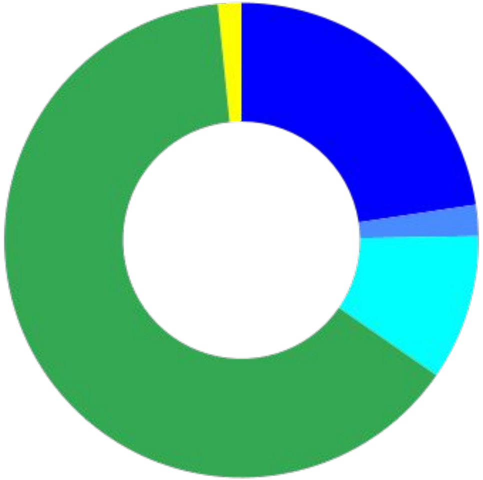
Train-set (90%)

Test-set (10%)

25,606 images
Total image of dataset



Overview Dataset



Scratched(63,8%)

Dent, flatten (thumb)(22,6%)

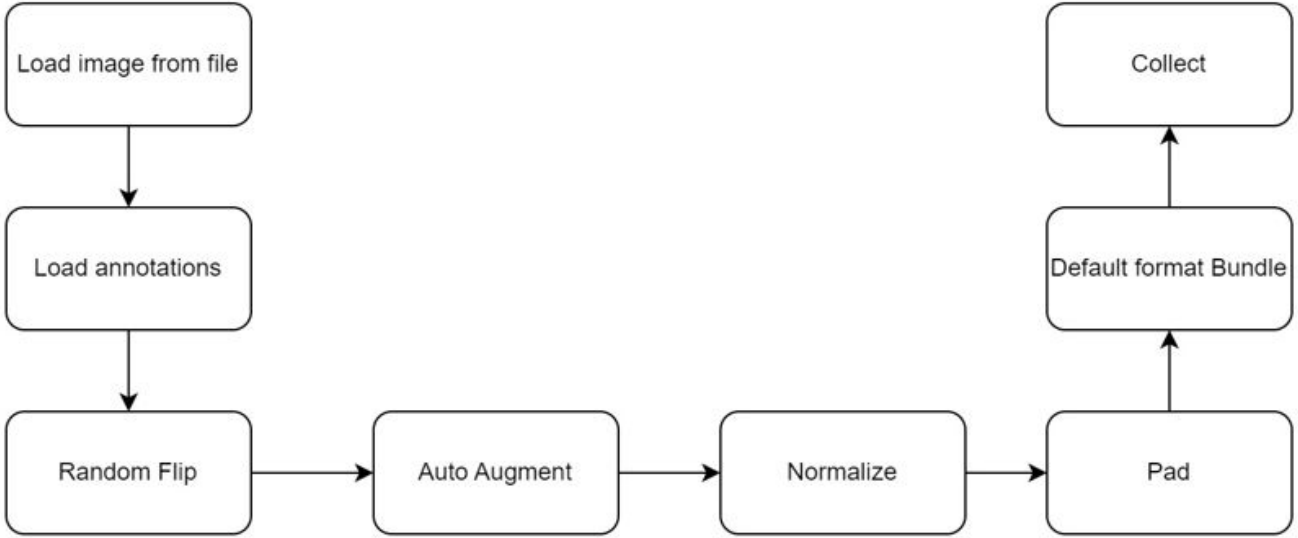
Broken, punctured, torn(9,9%)

Cracked(2,1%)

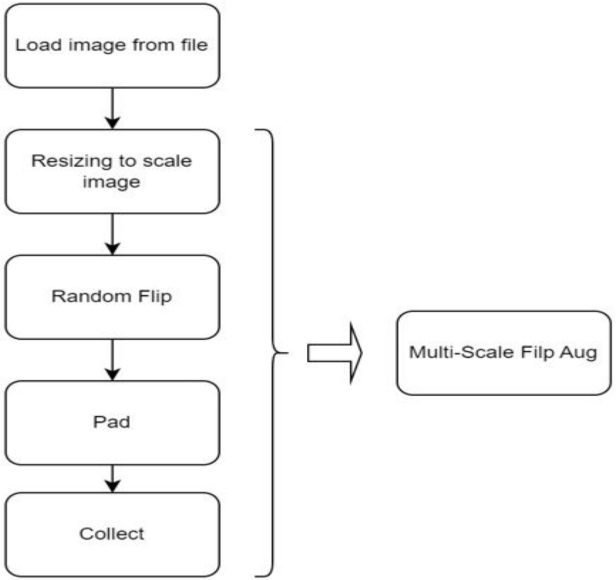
Shed(1,6%)

25,606 images
Total image of dataset

Training Pipeline



Test pipeline



04

Methodology

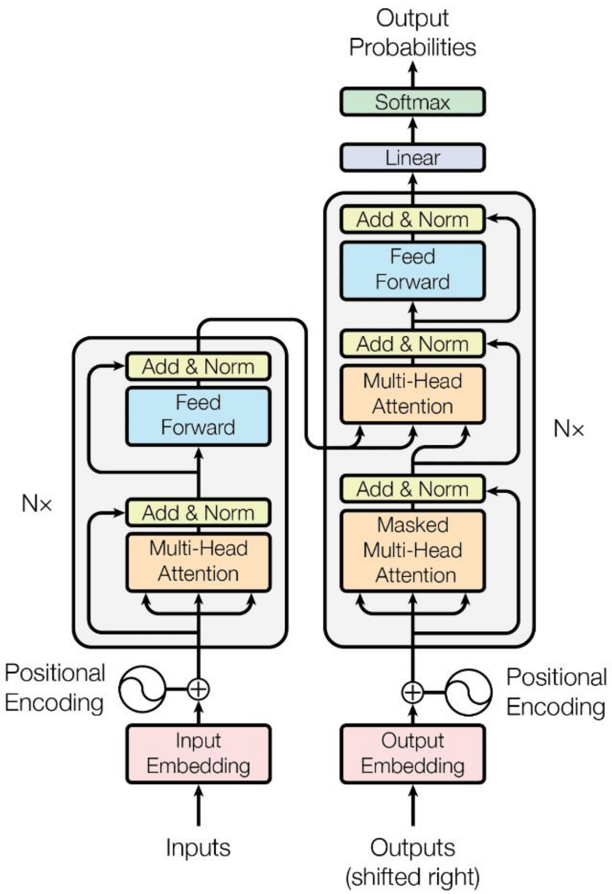
1. Swin Transformer
    a. Transformer Architecture
    b. Vision Transformer
    c. Swin Transformer
2. Cascade Mask R-CNN
    a. Mask R-CNN
    b. Cascade Mask R-CNN
3. Method

Transformer Architecture

The Transformer architecture is based on the self-attention mechanism, which allows the model to make predictions by selectively focusing on different parts of the input sequence.

A Transformer Block contains an encoder block and an decoder block:

- **An encoder consists of Multi-Head Attention (MSA), Feed Forward and Add & Norm.**
- **An decoder consists of Masked MSA, MSA, Feed Forward and Add & Norm.**



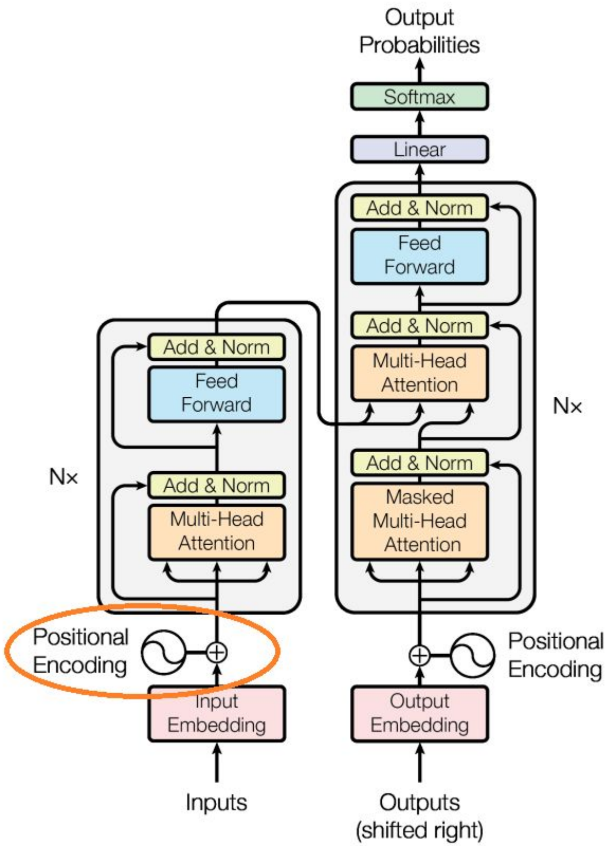
Positional Embedding

Positional encoding describes the location or position of an entity in a sequence by assigning a unique representation to each position.

Positional embedding formula:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

pos: is the position of the word in the sentence
i : is used for mapping column indices $0 \le i < d/2$
d_model : is the dimension of the output embedding space



Multi Head Attention

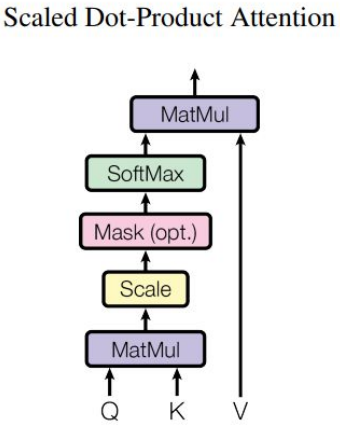
Scaled Dot-Product Attention:

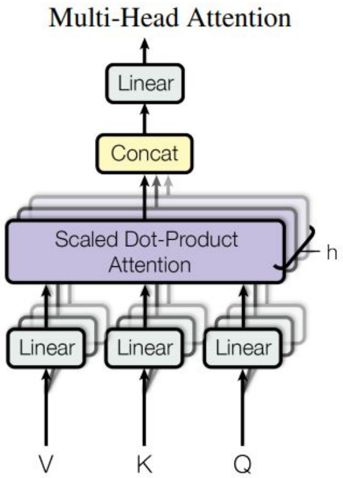
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Multi-Head Attention:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \dots, head_h)W^O$$

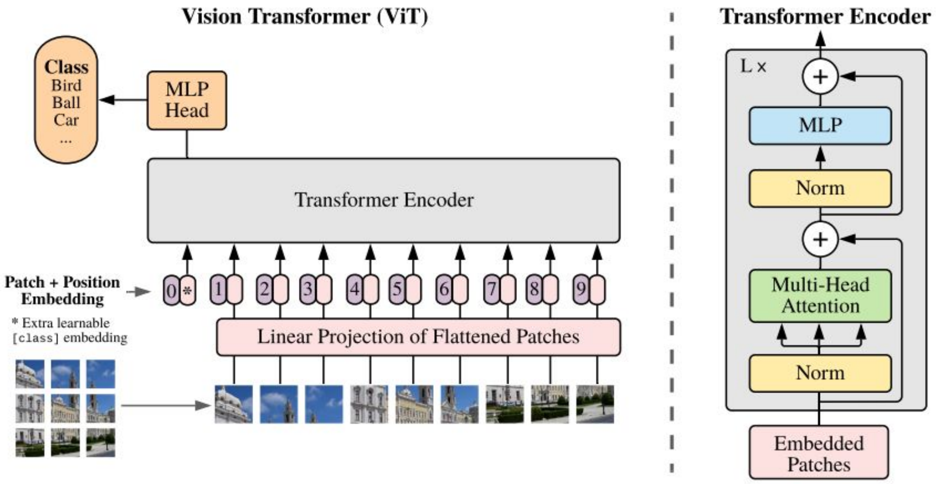
$$\text{where } head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$





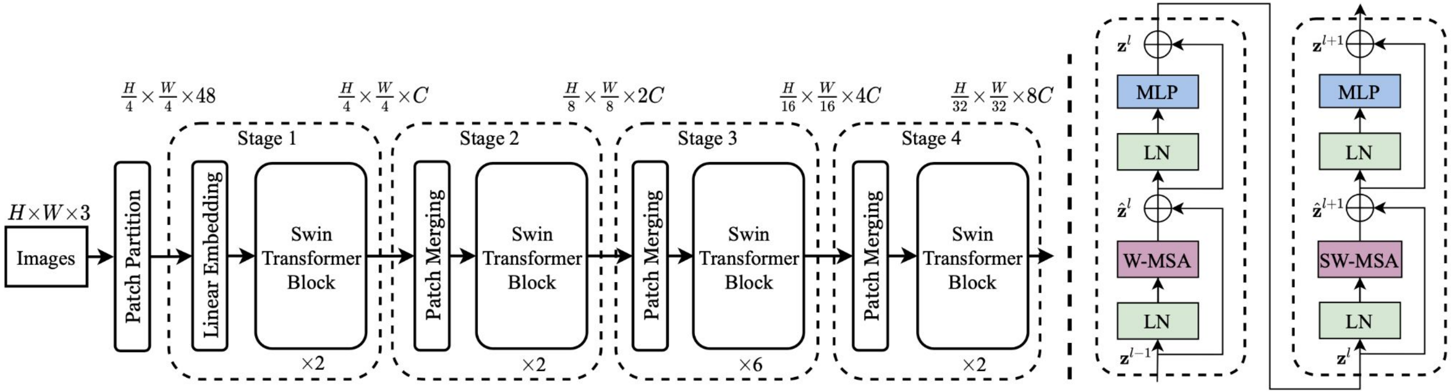
Vision Transformer

An image is divided into fixed-size patches, linearly embedded, and position embedded. The vector sequence is fed to a standard Transformer encoder.



The Norm layers are applied before the MSA and MLP blocks. Residual connection is still used after every sub-layer.

Swin Transformer



(a) Architecture

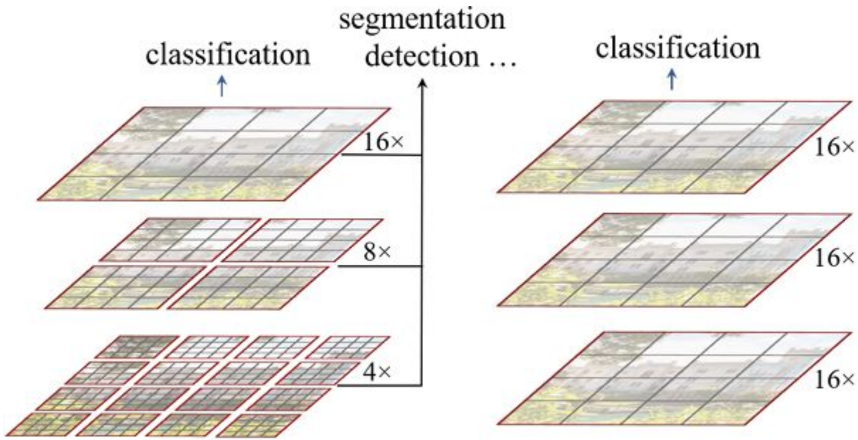
(b) Two Successive Swin Transformer Blocks

- **Swin-T: C = 96, layer numbers ={2, 2, 6, 2}.**
- **Swin-S: C = 96, layer numbers ={2, 2, 18, 2}.**
- **Swin-B: C = 128, layer numbers ={2, 2, 18, 2}.**
- **Swin-L: C = 192, layer numbers ={2, 2, 18, 2}.**

C is the channel number of the hidden layers in the first stage.

Patch Merging

The Patch-Merging layer merges four patches. So with every merge, both height and width of the image are further reduced by a factor of 2.

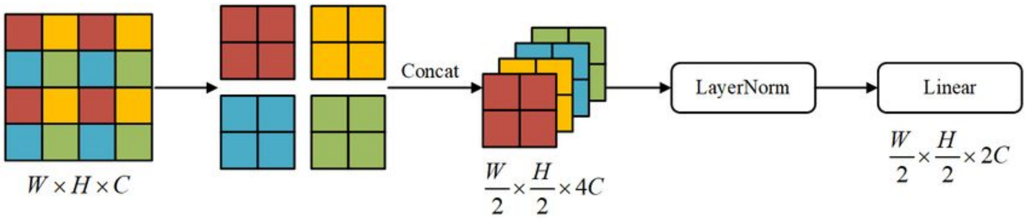


Stage-1, the input resolution is (H/4, W/4).

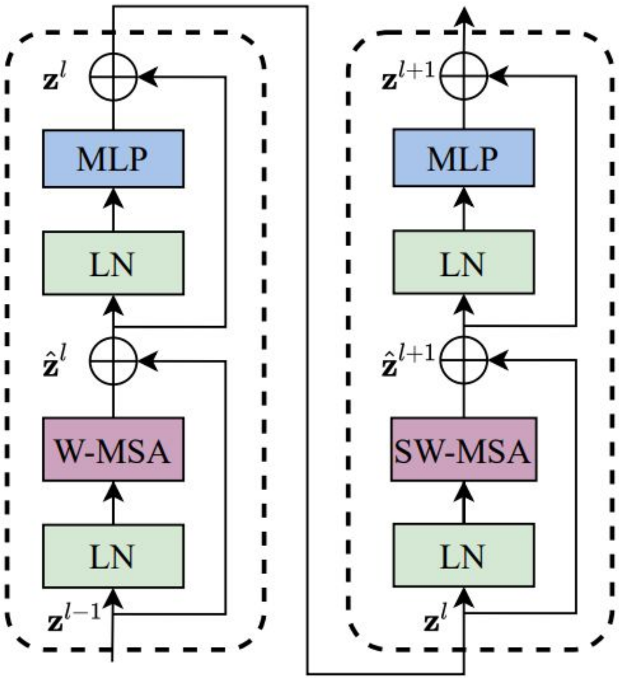
Stage-2, after patch merging, the resolution will change to (H/8, W/8).

Stage-3 the input resolution would be (H/16, W/16).

Stage-4, the input resolution would be (H/32, W/32).



Swin Block



Swin Transformer blocks are calculated as follows:

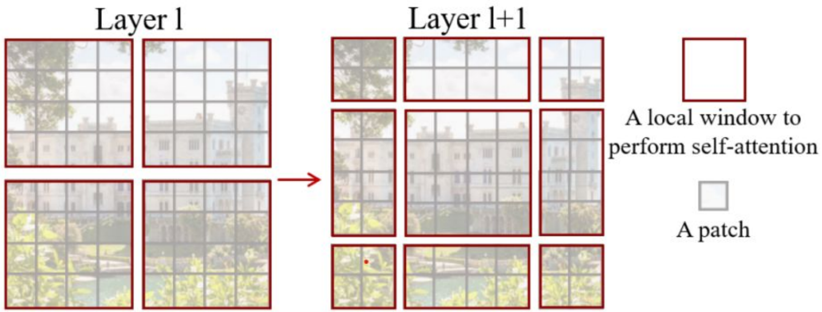
$$\hat{\mathbf{z}}^{l} = \text{W-MSA}\left(\text{LN}\left(\mathbf{z}^{l-1}\right)\right) + \mathbf{z}^{l-1},$$

$$\mathbf{z}^{l} = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l}\right)\right) + \hat{\mathbf{z}}^{l},$$

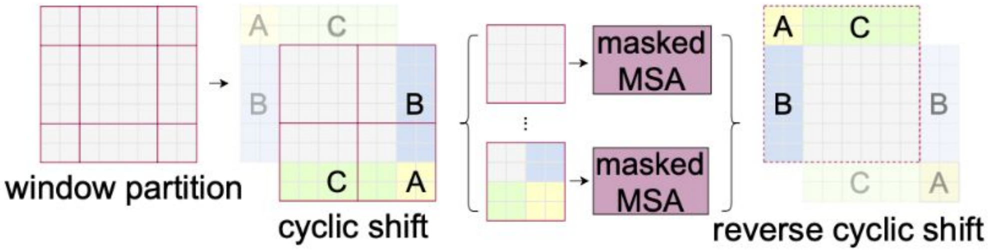
$$\hat{\mathbf{z}}^{l+1} = \text{SW-MSA}\left(\text{LN}\left(\mathbf{z}^{l}\right)\right) + \mathbf{z}^{l},$$

$$\mathbf{z}^{l+1} = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l+1}\right)\right) + \hat{\mathbf{z}}^{l+1},$$

Shifted Window



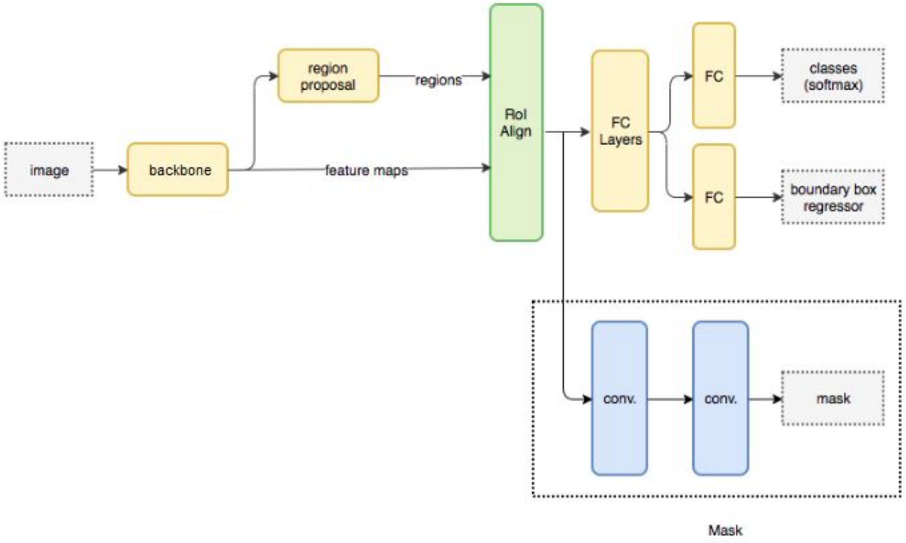
Shift the window by a factor M/2, where M is window size.



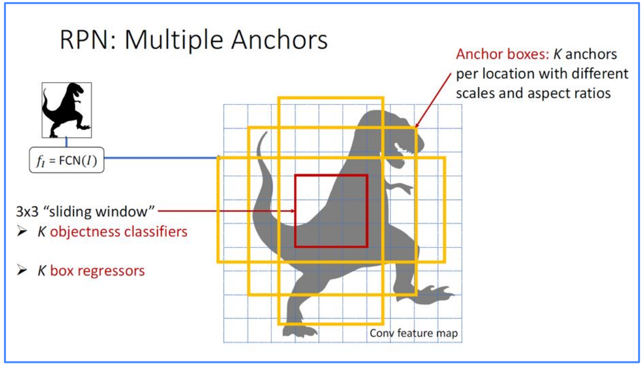
Swin Transformer uses cyclic shift to reduce computation heavy.

Mask R-CNN

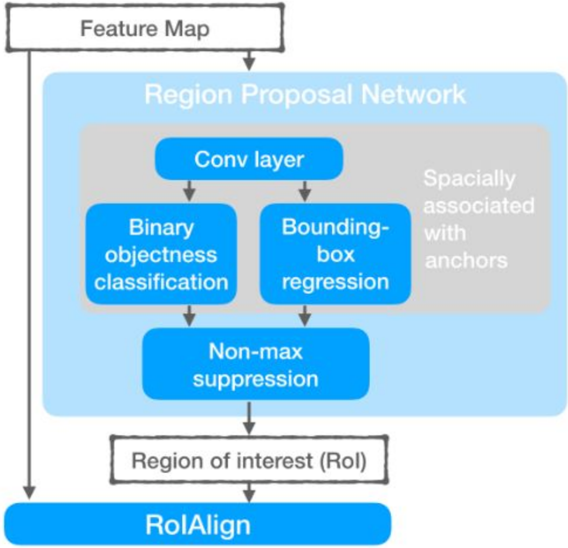
Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.



Region Proposal Network



At each position, the window slides into, multiple anchors will be generated with different scales and ratios



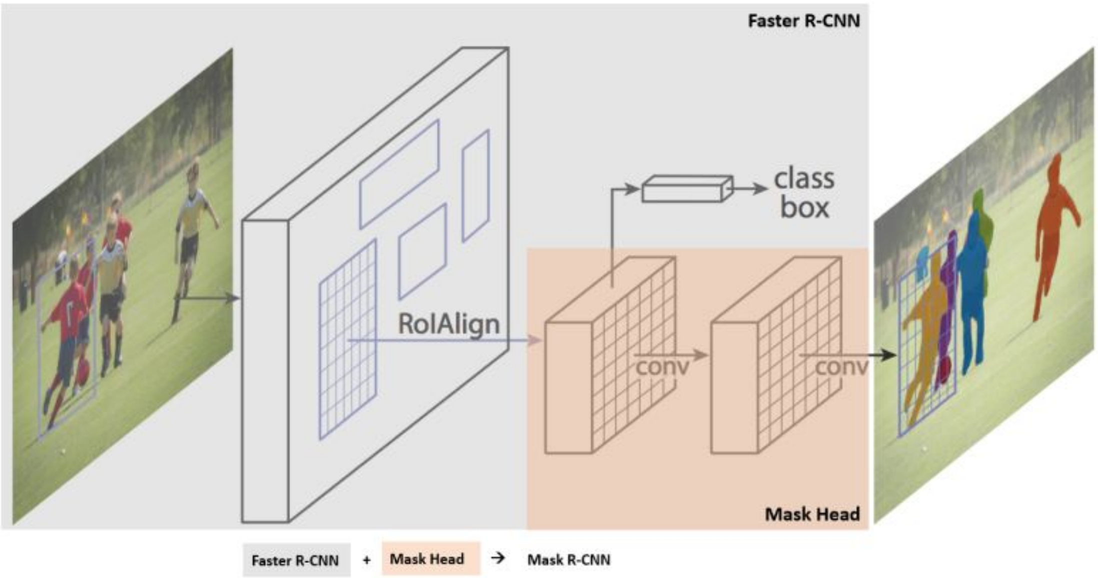
RPN uses a sliding window to run across the feature map to generate proposal anchors.

Anchors will be passed into two convolutional layers to be processed:

- **Bounding Box Classifier layer(cls): determine which anchor contains object.**
- **Bounding Box Regressor layer: finding properly coordinates of anchors.**

ROI output of RPN will be put into ROI align to resize into same size.

Mask Branch


Faster R-CNN
class
box
RoIAlign
conv
conv
Mask Head
Faster R-CNN
+
Mask Head
→
Mask R-CNN

Cascade Mask R-CNN

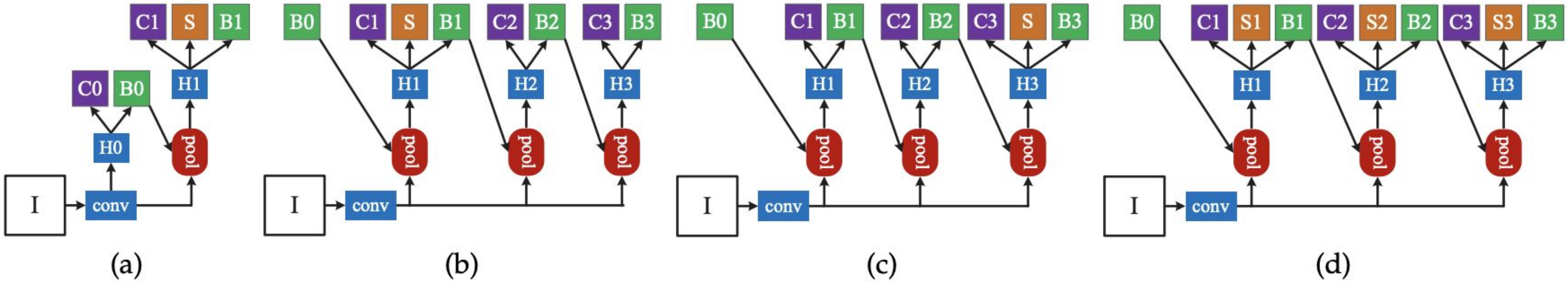
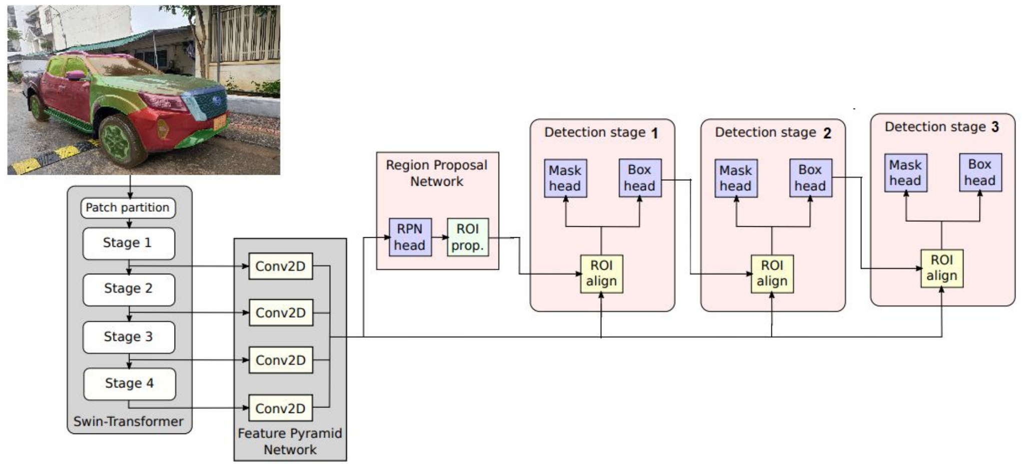


Fig. 6: Architectures of the Mask R-CNN (a) and three Cascade Mask R-CNN strategies for instance segmentation (b)-(d). Beyond the definitions of Fig. 3, "S" denotes a segmentation branch. Note that segmentations branches do not necessarily share heads with the detection branch.

Method



- **Cascade Mask R-CNN with Swin B Transformer as back bone**
- **Swin B Transformer integrates Feature Pyramid Network for different scales of objects**
- **Cascade architecture for improving higher accuracy of segmentation.**

Model Implementation

MMDetection is an object detection toolbox that contains a rich set of object detection, instance segmentation, and panoptic segmentation methods.

MMDetection is built on Pytorch.

LOSS FUNCTION

Classification Loss

$$\mathcal{L}_{\text{cls}}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*) \log(1 - p_i)$$

Bounding box Loss

$$\mathcal{L}_{\text{box}}(t^u, v) = \sum_{i \in \{x,y,w,h\}} L_1^{\text{smooth}}(t_i^u - v_i)$$

$$L_1^{\text{smooth}}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Mask Loss

$$\mathcal{L}_{\text{mask}} = -\frac{1}{m^2} \sum_{1 \leq i,j \leq m} \left[y_{ij} \log \hat{y}_{ij}^k + (1 - y_{ij}) \log(1 - \hat{y}_{ij}^k) \right]$$

Total Loss

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{mask}}$$

Evaluation metrics

Intersection of Union:

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$$

Precision and Recall:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Mean Average Precision:

$$mAP = \frac{1}{N} \sum_{i=2}^{N} (r_i - r_{i-1}) \frac{p_{i-1} - p_i}{2}$$

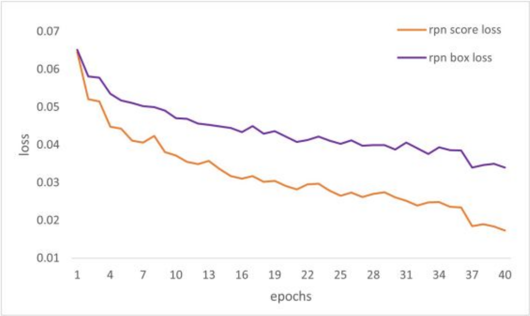
05

Result

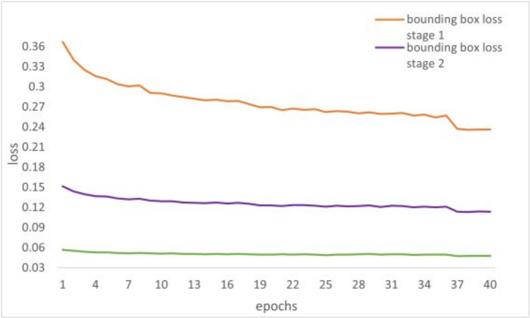
Training

- CPU: AMD Ryzen 7 5700G
- GPU: Radeon Graphics NVIDIA RTX A4000

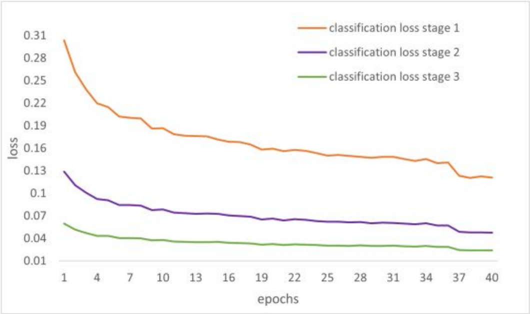
Loss during training



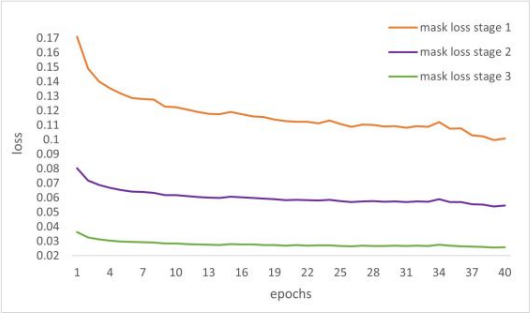
(a) RPN loss



(b) Bounding box loss



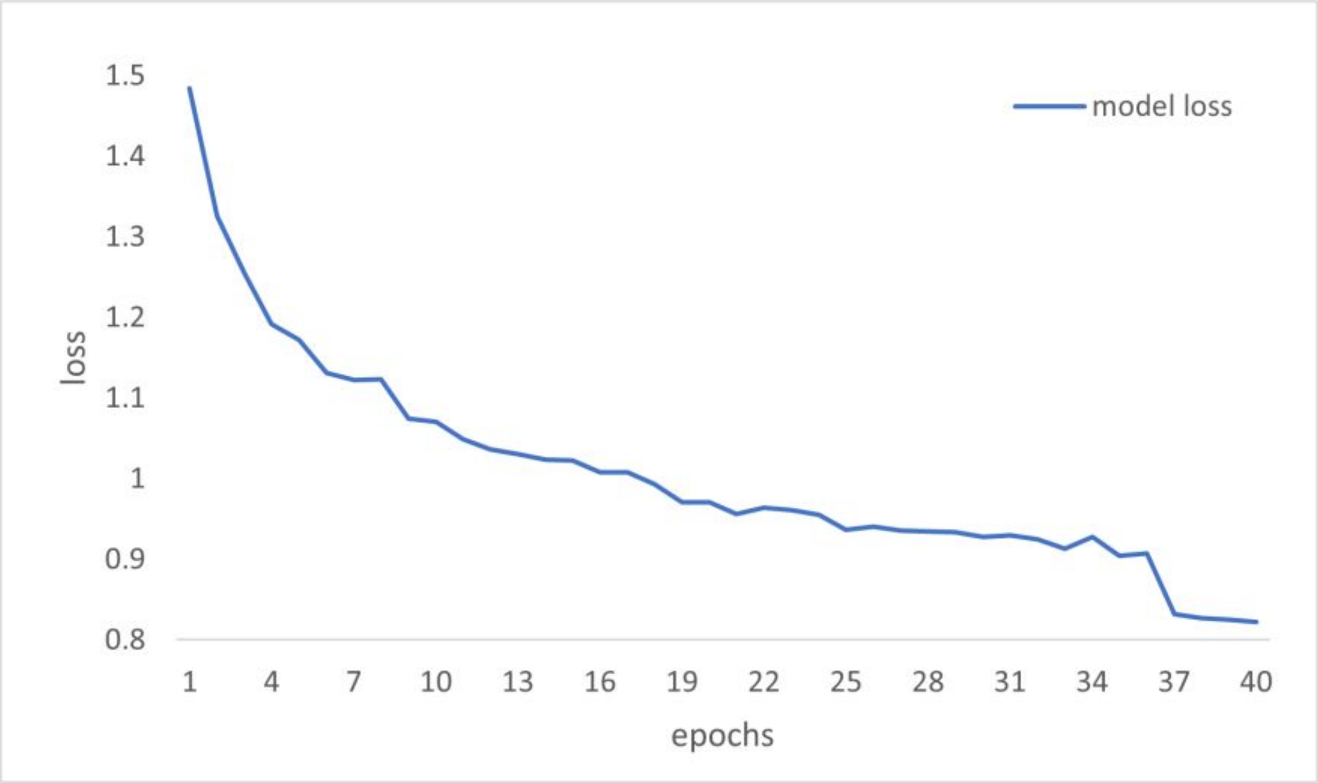
(c) Class loss



(d) Mask loss

Loss over epochs for SwinB-Cascade Mask RCNN

Total loss of model



Result of Damage detection

Result		
Damaged name	Recall	mAP
Dent, flatten (thumb)	0.586	0.421
Cracked	0.740	0.547
Broken, punctured, torn	0.415	0.117
Scratched	0.584	0.380
Shed	0.282	0.083

Models Comparison

Comparision between Models				
Model	bbox mAP 50	bbox mAP 75	seg mAP 50	seg mAP 75
Resnext101CascadeMaskRCNN	0.817	0.721	0.797	0.650
Resnext101 HTC	**0.838**	**0.724**	0.805	0.656
Swin-T Cascade Mask RCNN	0.817	0.699	0.801	0.661
Swin-B Cascade Mask RCNN	0.836	0.771	**0.817**	**0.705**

06

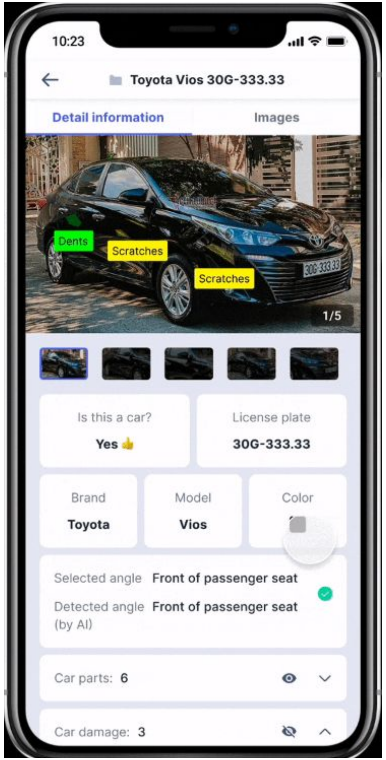
Conclusion and Q&A

Conclusion

- The effective use of segmentation of detecting parts into the insurance will be extremely beneficial for companies.
- By using Artificial Intelligence, the project achieves high accuracy in recognizing and distinguishing car parts such as doors, wheels, glass, etc...
- Reduce the time and effort required for claim processing by providing faster and more efficient service to customers.

FUTURE WORK

- Extend model training data.
- Improved damage accuracy.
- Application in real-life and some insurance company.
- Developing mobile applications to increase popularity among users.



THANK YOU

Q&A