An effective method for fashion parsing task

Final Year Project Report
4th Year Student Names
Khuc Giang Sinh

Under the supervision of
M.Sc. Do Thai Giang

Bachelor of Computer Science
Hoa Lac campus - FPT University
15 December 2021

DECLARATION

Project Title: An effective method for fashion parsing task
Authors: Khuc Giang Sinh
Student ID: HE141179
Supervisor: M.Sc Do Thai Giang

I declare that this thesis entitled An effective method for fashion parsing task is the result of my own work except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Khuc Giang Sinh
HE141179

Department of Computer Science
Hoa Lac Campus – FPT University

Date: December 15, 2021

ACKNOWLEDGEMENT

We would like to express our deep and sincere gratitude to our research instructor, Mr. Do Thai Giang, for giving us the opportunity to do research and providing invaluable guidance throughout this research. His dynamism, vision, sincerity, and motivation have deeply inspired us. He has taught me the methodology to carry out the research and to present the research works as clearly as possible. It was a great privilege and honour to work and study under his guidance. We would also like to thank our teacher for his empathy, and immense knowledge.

Besides our instructor, we are extending our thanks to our friends at FPT University for their support, patience, and friendship. Our sincere gratitude also goes to FPT University, which created a wonderful environment for our development during the four years we have studied here.

We also want to send huge love to our caring, loving, and supportive families. Their encouragement was a great comfort and relief for us during our hard times. Finally, we would like to thank all the people who have supported us to complete the project work directly or indirectly.

ABSTRACT

Fashion parsing is a fundamental task when deploying applications such as product images search, recommended, visual try on,..etc. It must first recognize the human body component of the input image in order to determine where the clothing area is located and then synthesise clothes in that location. However, this is quite complicated because the clothes are not uniform in style: wrinkling, fading overtime or the minor inter-class variance: The factors that make an image distinguishable from other classes are quite small (the long skirt image may be mistaken for a slightly shorter skirt or cross-domain issues: the user domain image is different from the store domain image.

This thesis presents an approach for a fashion parsing task: detect the type of clothes and segment on pixel level in the images. We found that not every feature map is important to pay attention to and conversely there are feature maps that bring a lot of important information. Previous works have not focused on using this mechanism for fashion parsing tasks. Therefore, we tested the attention mechanism on the Mask-RCNN with modified backbone feature extraction to know how this mechanism affects model result: Integrated channel attention in backbone to collect more important features about clothes and suppress less useful features. Experiments show that applying the channel attention module does not improve results than the original mask-rcnn and state of the art models.

Keywords: Fashion parsing, Object detection, Segmentation, Channel attention

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Motivation

The field of e-commerce is increasingly developing in the current technology era. Major e-commerce sites like Shopee, Lazada, Alibaba always try to come up with new features for their systems to help interact with customers better, typically the feature of searching products by image or language. To do this, it is necessary to solve the problem of product identification and zoning. Therefore, the laboratories at corporations are always trying to improve their models. However, for fashion clothing products, this is relatively complicated and difficult task. Firstly, There are many clothes/products in one photo, usually a photo of a person wearing a costume with 2 or more items. Secondly, the large intra-class variance: external conditions such as lighting, background noise, clothing shape, distortions and deviations between user domain and store domain images (shop domain) makes images from domains relatively/absolutely difficult to parsing. Thirdly, the minor inter-class variance: the factors that make an image distinguishable from other classes are quite small (the long skirt may be mistaken for a slightly shorter skirt,...)

In this thesis, we focus on detecting clothing types and localising them on input images from different conditions. There are many methods for this problem, but all are based on the basic instance and semantic segmentation models such as Mask-RCNN[1], U-Net[2], DeepLab[3],...

1.2 Related Work

Fashion parsing, also known as human parsing or clothes parsing, is a type of semantic segmentation in which labels are created based on clothing items such as dresses and pants. Figure 1 shows examples of fashion parsing tasks. Yamaguchi et al. [4] were the first to work on fashion parsing. By mutually improving two difficulties, they utilised the link between garment parsing and human posture estimation. In a Conditional Random Field model, apparel labels for each picture segment were predicted with regard to body components. The clothing predictions were then used as extra characteristics for pose estimation. Their research, on the other hand, was primarily focused on the limited parsing problem, in which test photographs were processed using user-supplied tags to indicate displayed apparel items. [5, 6] advocated garments parsing using a retrieval-based strategy to solve this problem. Similar photos from a parsed dataset were initially collected for a specific image, and then dense matching was used to transfer the nearest-neighbour parsings to the final result. Liu et al. [7] presented the fashion parsing task with weak supervision from colour-category tags instead of pixel-level tags since pixel-level labels were time-consuming for model training. To create category classifiers, they integrated the human posture estimation module with the (super)pixel-level category classifier learning module. They finished the parsing operation by applying the category tags. In 2019, human parsing problems were considered using hierarchical graphs. Wang et al. [8] characterised the human body as a hierarchy of multi-level semantic elements and used three techniques to capture human parsing information for improved parsing performance (direct, top-down, and bottom-up). Gong et al. [9] used hierarchical graph transfer learning based on the traditional parsing network to create Graphonomy, a generic human parsing model that consisted of two processes, to tackle human parsing in many domains with a single model without retraining on diverse datasets. It learned and propagated a compact high-level graph representation among labels within a single dataset before transferring semantic information across sev-

eral datasets. The Match R-CNN[10], which based on mask-rcnn method for both clothes detection and segmentation, landmark detection, fashion retrieve was published as demo in DeepFashion 2[10] dataset paper. In 2020, DeepFashion2 challenge announcement top winning method for fashion detection: Aggreation and Finetuning[11], DeepMark[12], DeepMark++[13].

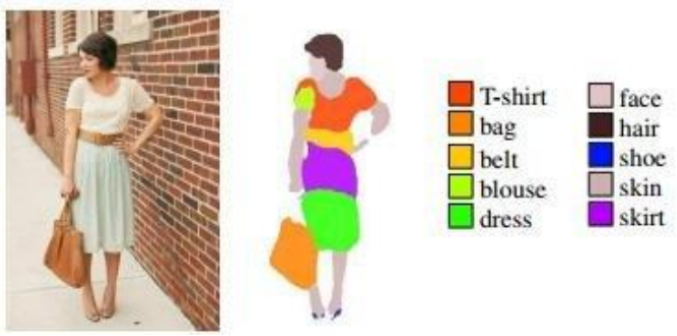

Figure 1.1: Example of Fashion parsing task[14]

1.3 Objective and Contribution

In this thesis, we purpose the approach for fashion parsing on the DeepFashion2 dataset. The task aim to classify accurate diversity clothes catergories, draw the bounding box which all-encompassing object and then labeling correct pixel. For detection and segmentation, we use the Mask-RCNN model. In the backbone feature extraction, we use Feature Pyramid Network[15] with ResNet101[16] at the bottom-up pathway. To enhance attention on channel to know where important information and how relationship between feature maps, the Squeeze and Excitation[17] network is integrated on all ResNet block before shortcut layer. DeepFashion2 is a dataset with variety of fashion models and data imbalance, so we designed a training strategy to handle this problem.

1.4 Outline

Chapter 1: A general introduction about Fashion parsing, and the scope of this thesis.

Chapter 2: The basic knowledgement to understand this thesis.

Chapter 3: An overview of all the methods used on this project for implementation.

Chapter 4: Details implementation of the method introduced in the previous chapter. We will try to pre-process and extract feature input images, and then detect bounding boxes and localise pixel

Chapter 5: Perform experimental results from the training model and evaluation metrics. We also include comparisons of our method to the baseline and SOTA method of the same task.

Chapter 6: Conclusion about the results of testing our models and possible future works that could be done for further improvisation.

Chapter 2

BACKGROUND

2.1 Neural Network

The Neural Network(NN)[18] is a computer programming paradigm that mimics the behaviour of human neural networks. Neural networks, when combined with deep learning(DL), are a flexible tool for solving a variety of complicated issues, such as image classification, natural language processing. Artificial intelligence is now a common topic in a variety of science and technology sectors.

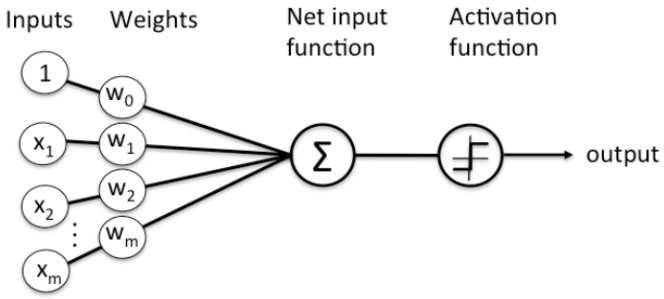

Figure 2.1: Simple Neural Network[19]

2.1.1 Basic Components

Unit

In terms of Neural Network, a unit is a node, also known as a neuron or a Perceptron, is a computing unit with one or more weighted input connections, a transfer function that somehow mixes the inputs, and an

output connection.

Layer

The Neural Network is made up of three types of layers: The neural network's initial data is stored in the input layer. Hidden layers are a layer that sits between the input and output layers and is where all of the calculation takes place. Produce a result for supplied inputs in the output layer.

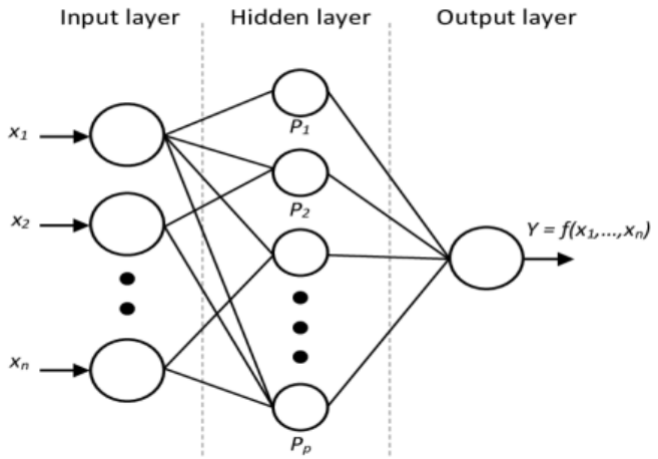

Figure 2.2: Layer of Neural Network[20]

Activation Function

In a neural network, an activation function specifies how the weighted sum of the input is turned into an output from a node or nodes in a layer. A weighted sum of inputs is passed through an activation function and this output serves as an input to the next layer.

ReLU

The rectified linear activation function (ReLU) is a piecewise linear function that, if the input is positive, outputs the input directly; else, it outputs zero(see Figure 2.3). Because a model that utilises it is quicker to train and generally produces higher performance, it has become the default activation function for many types of neural networks.

Sigmoid

The sigmoid function is a special form of logistic function and used as an activation function in a neural network to guarantee that the output of this

unit will always be between interval 0 and 1(see Figure 2.4). Because the sigmoid is a nonlinear function, the output of this unit will be a nonlinear function of the weighted sum of inputs.

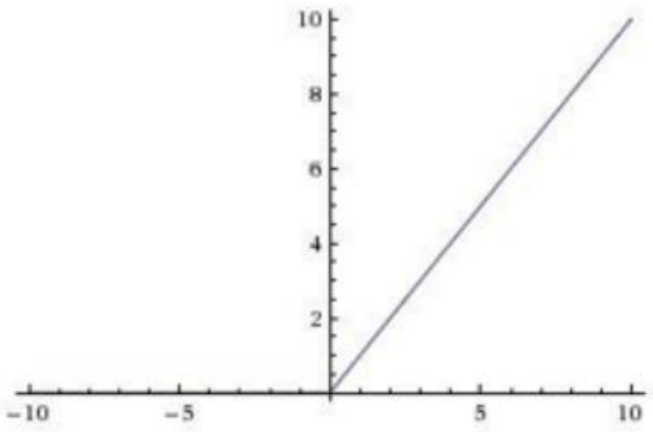

Figure 2.3: The ReLU function[21]

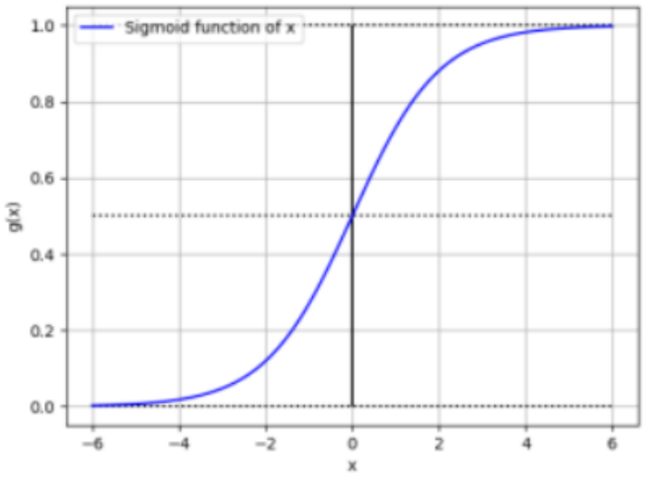

Figure 2.4: The sigmoid function[22]

Softmax

The softmax function calculates the probability of a class occurring out of the total of all possible classes. This probability is then used to determine the target class for the inputs. The softmax function turns the k-dimensional vector of any real values into a real-valued k-dimensional vector that sums to 1. The input values can be positive, negative, zero,

or greater than 1, but the softmax function will always turn them into a value in the range $(0:1]$.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

$\vec{Z}$: Input vector value for softmax function, (from z_0 to z_k)

Z_i : All z values are input vector values for softmax function. They can be any real number, positive, negative or zero. For example, an artificial neural network might have an output vector value of $(-0.62, 8.12, 2.53)$. This is not a correct probability distribution. That's why we need the softmax function.

e^{z_i} : The standard exponentiation function is applied to each input value. It will return a positive value greater than 0 . This value will be very small if the input is negative, and very large if the input is positive.

$\sum_{j=1}^{K} e^{z_j}$: The bottom line of the formula is a normalised cluster. It guarantees that the sum of the outputs will always be 1 and be in the range $(0:1]$. Thus, an exact probability distribution will appear.

$\sum_{j=1}^{K} e^{z_j}$: Number of classes in a multi-class classification.

Weight

In a neural network, the weight parameter changes input data within the network's hidden layers. A neural network is built up from nodes and a weight, a bias value are all contained within each node.

Loss function

Loss is a prediction error of the Neural Network. The Loss Function is a component to calculate the loss, or gradient. And gradients are used to update the weights of the Neural Network.

- Cross-Entropy loss:

$$\text{loss}(x, y) = -\sum x \log y$$

To understand the probability distribution of the data, cross-entropy is employed as a loss function. While other loss functions, such as squared loss, punish inaccurate predictions, cross entropy penalizes

incorrect predictions even more when they are anticipated with high confidence. Cross entropy differs from negative log loss in that it penalizes both incorrect but confident predictions and accurate but less confident predictions, whereas negative log loss does not penalize according to prediction confidence.

- Loss Smooth L1(Huber loss):

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

If the absolute error is less than 1, it uses a squared term; otherwise, it uses an absolute term. It is less susceptible to outliers than the mean square error loss, and it can avoid explosive gradients in some instances. We square the difference in mean square error loss, resulting in a value that is substantially bigger than the initial number. Gradients explode as a result of these high values. This is avoided in this case because integers bigger than one are not squared.

2.2 Convolution Neural Network

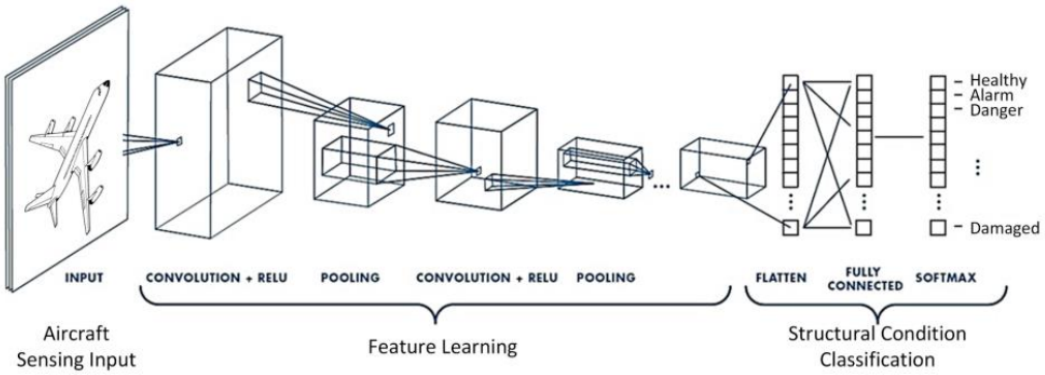

Figure 2.5: Example basic of CNN architecture[23]

A Convolutional Neural Network (ConvNet/CNN)[24] is a Deep Learning system that can take an input image, assign parameters (learnable

weights and biases) to various aspects/objects in the image, and distinguish between them. When compared to other classification methods, the amount of pre-processing required by a ConvNet is significantly less. While basic approaches need hand-engineering of filters, ConvNets can learn these filters/characteristics with enough training.

The design of a ConvNet is inspired by the organisation of the Visual Cortex and is akin to the connection pattern of Neurons in the Human Brain. Individual neurons can only respond to stimuli in a small area of the visual field called the Receptive Field. A number of similar fields can be stacked on top of each other to span the full visual field.

Pooling layer

The Pooling layer, like the Convolutional Layer, is responsible for shrinking the Convolved Features spatial size. Through dimensionality reduction, the computer power required to process the data is reduced. It's also beneficial for extracting rotational and positional invariant dominating features, which helps keep the model's training process running smoothly.

Pooling may be divided into two types: max pooling and average pooling. The largest value from the part of the picture covered by the Kernel is returned by Max Pooling. Average Pooling, on the other hand, returns the average of all the values from the Kernel's section of the picture.

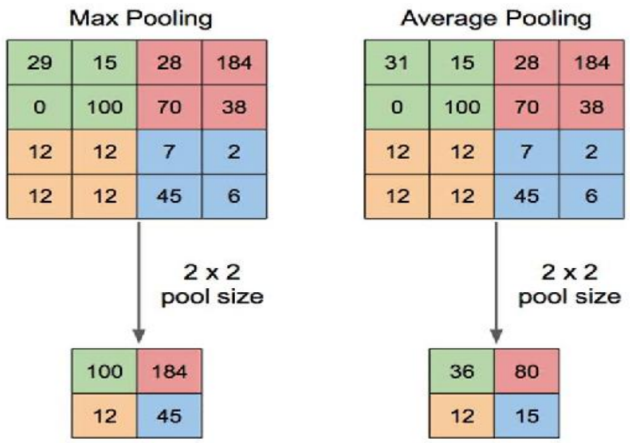

Figure 2.6: Max pooling: Each pooling operation selects the maximum value of the current view (Left); Average pooling: Each pooling operation averages the values of the current view (Right)[25]

Max Pooling works as a Noise Suppressant as well. It removes all noisy activations and conducts de-noising and dimensionality reduction at the same time. Average Pooling, on the other hand, just reduces dimensionality as a noise-suppressing strategy. As a result, we may conclude that Max Pooling is better than Average Pooling.

Fully Connected Layer(FCN)

Adding a Fully-Connected layer is a (typically) low-cost approach of learning non-linear combinations of high-level information represented by the convolution layer's output. In such an area, the Fully-Connected layer is learning a possibly non-linear function.

We'll flatten the image into a column vector now that we've turned it into a format suited for our Multi-Level Perceptron. Every round of training uses backpropagation to send the flattened output to a feed-forward neural network. The model can discriminate between dominant and certain low-level characteristics in pictures across a number of epochs and categorise them using the Softmax Classification technique.

2.3 Object Detection

Object detection is a critical computer vision problem that involves recognizing things of a certain class in digital photos and movies, such as birds, people, and cars. Face detection, facial identification, object tracking, and video surveillance are some of the other uses of object detection. Instead of limiting our focus to classify an image, we should broaden it to accurately estimate the positions of items in a digital image to acquire a thorough comprehension of it. The input for this task will be an image with one or more objects, and the output will be an image with bounding boxes around those different objects and a label specifying the object's class in the bounding box. Image Segmentation can help boost the object detection process even further.

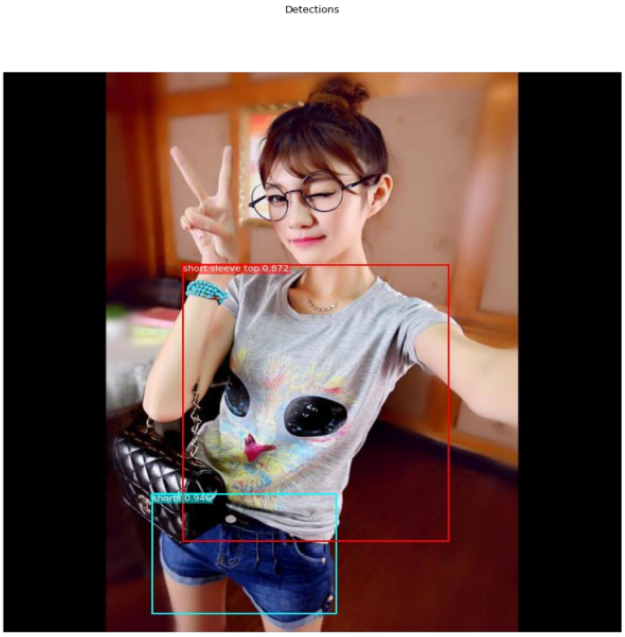

Figure 2.7: Example of object detection

2.3.1 Region Based Convolutional Neural Network(RCNN)

Convolution Neural Network (CNN) with a fully connected layer is not able to deal with the frequency of occurrence and multi objects. So, one way could be that we use a sliding window brute force search to select a region and apply the CNN model on that, but the problem of this approach is that the same object can be represented in an image with different sizes and different aspect ratio. While considering these factors we have a lot of region proposals and if we apply deep learning (CNN) on all those regions that would be computationally very expensive.

Ross Girshick et al.in 2013[26] proposed an architecture called R-CNN (Region-based CNN) to deal with this challenge of object detection. This R-CNN architecture uses the selective search algorithm that generates approximately 2000 region proposals. These 2000 region proposals are then provided to CNN architecture that computes CNN features. These features are then passed in an SVM model to classify the object present in the region proposal. An extra step is to perform a bounding box regressor to localize the objects present in the image more precisely.

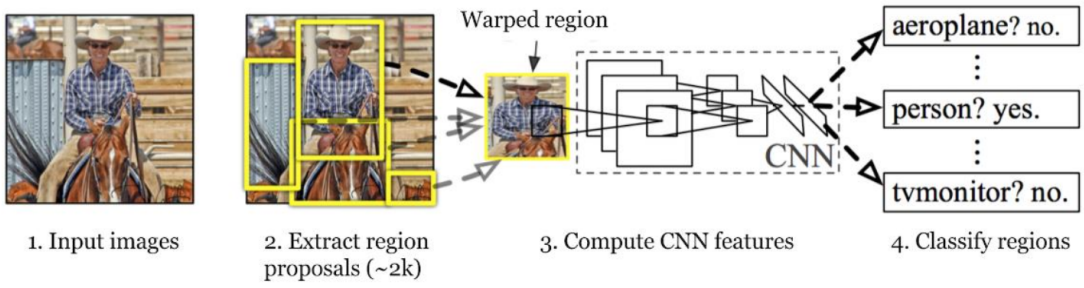

Figure 2.8: Example of RCNN workflow[26]

Region Proposals: Region proposals are simply the smaller regions of the image that possibly contain the objects we are searching for in the input image. To reduce the region proposals in the R-CNN uses a greedy algorithm called selective search.

Selective Search: Selective search is a greedy algorithm that combines smaller segmented regions to generate region proposals. This algorithm takes an image as input and output generates region proposals on it. This algorithm has the advantage over random proposal generation is that it limits the number of proposals to approximately 2000 and these region proposals have a high recall.

Selective search algorithm:

Step 1: Generate initial sub-segmentation of input image.

Step 2: Combine similar bounding boxes into larger ones recursively.

Step 3: Use these larger boxes to generate region proposals for object detection.

Fast RCNN

To speed up R-CNN, Girshick[27] improved the training process by merging 3 independent models into a common training framework and increasing computational sharing. This model is called Fast R-CNN. Instead of extracting CNN feature vectors for each region proposal, this model aggregates them into a CNN forward across the entire image and region proposals that share feature matrices. Then, the same feature matrices will be branched to be used for classification as bounding-box regression.

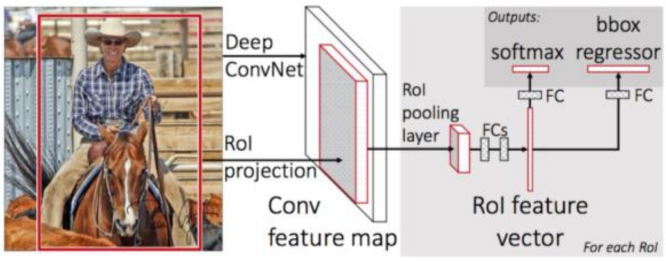

Figure 2.9: Example of Fast-RCNN workflow[27]

ROI Pooling: This is a type of max pooling used to convert features in the projected region of the image of any size (h x w) into a window with a fixed size (H x W). The input region will be divided into H x W grids, approximately for each subwindow of size h/H x w/W. Then use max-pooling for each grid.

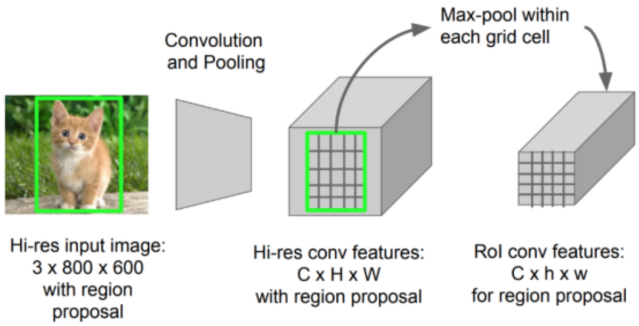

Figure 2.10: Example of ROI Pooling[28]

Faster R-CNN

One solution to speed up the Fast RCNN algorithm is to integrate the region recommendation algorithm into the CNN model. Faster RCNN [29] is doing exactly this: building a single model consisting of an RPN (region proposal network) and a Fast RCNN with a shared CNN.

Faster RCNN Workflows

Step 1: Using CNN pre-train on image classification

Step 2: Fine-tune RPN (region proposal network) for the region proposal task, initialised by pre-train image classifier. Positive examples have IoU ¿ 0.7, negative examples have IoU ¡ 0.3

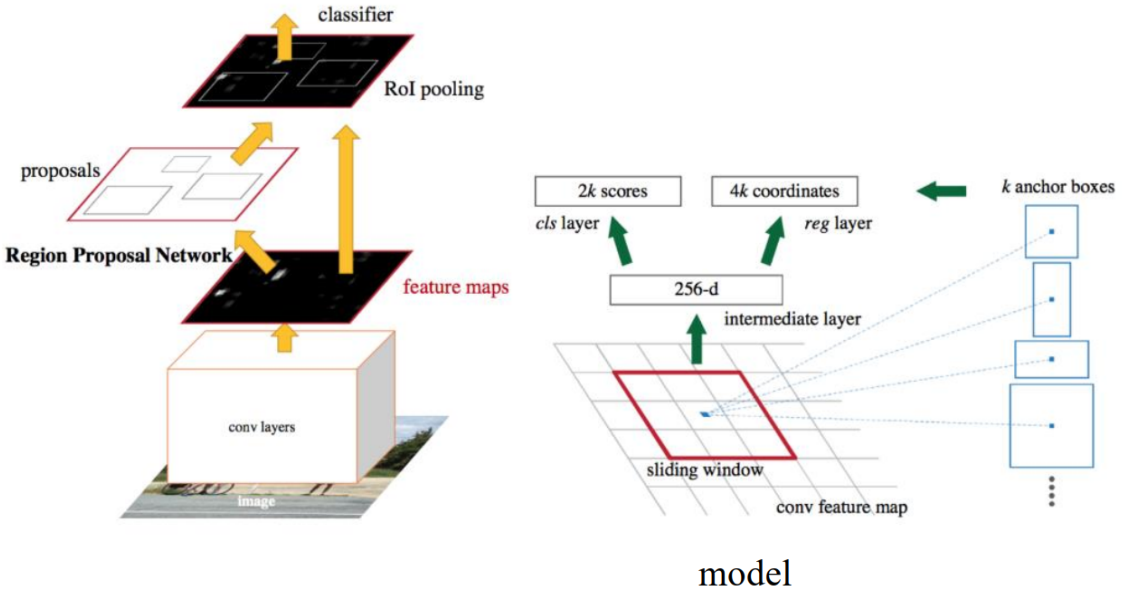

Figure 2.11: Faster R-CNN architecture[29]

+ Slide a small window of size n x n over the entire CNN feature of the image

+ At the centre of each window, we predict multiple regions with different scales and ratios at the same time. Anchor is a combination of sliding window centre, cal and ratio. For example, 3 scales + 3 ratios =¿ k = 9 anchors at each slip position.

Step 3: Train the Fast R-CNN model using proposals region generated from the current RPN -

Step 4: Then use Fast R-CNN network to initialise RPN training. While keeping the convolution layers shared, just fine-tune the specific RPN layers. At this stage, RPN and detection networks have shared convolution layers! -

Step 5: Finally, we will fine-tune the separate layers of Fast R-CNN.

Step 6: Steps 4-5 can be repeated to train the RPN and Fast R-CNN if needed.

Loss Function of Faster-RCNN: is a combination of classification loss and bounding-box regression loss:

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

$$\mathcal{L}\left(\{p_i\}, \{t_i\}\right) = \frac{1}{N_{\text{cls}}} \sum_i \mathcal{L}_{\text{cls}}\left(p_i, p_i^*\right) + \frac{\lambda}{N_{\text{box}}} \sum_i p_i^* \cdot L_1^{\text{smooth}}\left(t_i - t_i^*\right)$$

where loss classification defined as

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

2.4 Image Segmentation

Image Segmentation means that the problem will divide an image into many different image regions. Image Segmentation also has the same goal as object detection, which is to detect areas of an image containing objects and assign appropriate labels to them. However, the standard of accuracy of Image Segmentation is higher than that of Object Detection when it requires accurate prediction labels to every pixel. Although Image Segmentation requires a higher level of detail, in return the algorithm helps us to understand the content of an image at a deeper level when we know at the same time: The position of the object in the image, the shape of the object and which individual pixel belongs to the object.

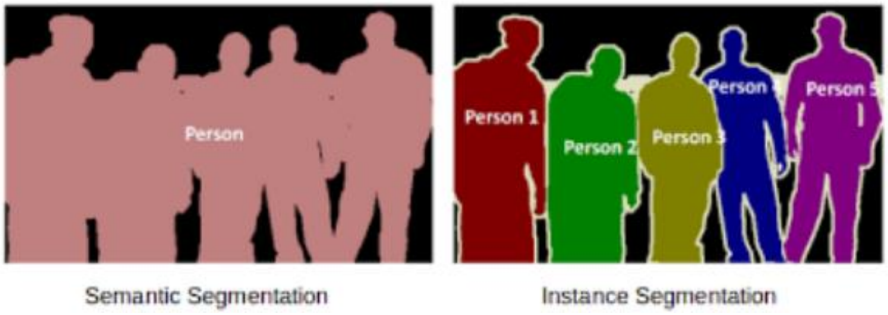

Figure 2.12: Different between Semantic and Instance segmentation[30]

2.5 Mask R-CNN

Mask R-CNN is an extended version of Faster R-CNN in pixel-level image segmentation. The key point is to separate the task of classification

and mask prediction at the pixel level. Based on Faster R-CNN, Mask RCNN adds a brand to predict the object's mask parallel with the bounding box branch. Mask detection is a fully-connected network applied to each RoI.

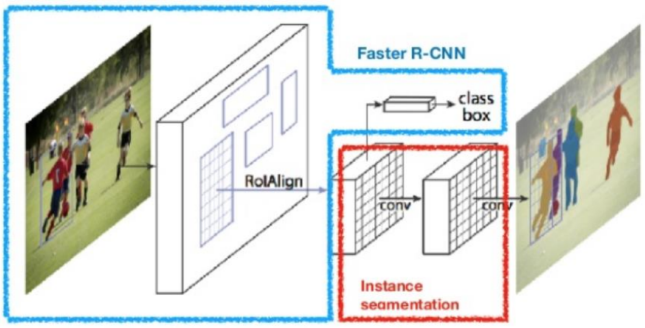

Figure 2.13: Mask-R CNN extend Faster-RCNN module[1]

Since pixel-level segmentation requires much smoother alignment than bounding-box, Mark R-CNN improves the RoI Pooling layer so that RoI can be mapped more accurately to regions of the original image.

ROI Align: RoIAlign is designed to find misplaced errors caused in RoI pooling. RoIAlign eliminates hash quantization, for example, by using x/16 instead of [x/16], so that the extracted features can be exactly aligned with the input pixels. Bilinear interpolation is used to calculate floating point position values in the input.

Loss Function of Mask-RCNN: is a combination of 3 losses: classification, localization and segmentation mask:

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

where: loss box and loss classification are the same as Faster-RCNN. Loss mask defined as:

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

where: y_{ij} is the label of cell (i, j) $\hat{y}_{ij}^k$ is predict value of pixel (i, j) for each class k

Mark RCNN generates a mask of size $m \times m$ for each RoI and each class (K classes). Therefore, the output is of size K $\cdot$ m2. Because the model is trying to learn one mask for each layer, there is no competition between layers to create the mask.

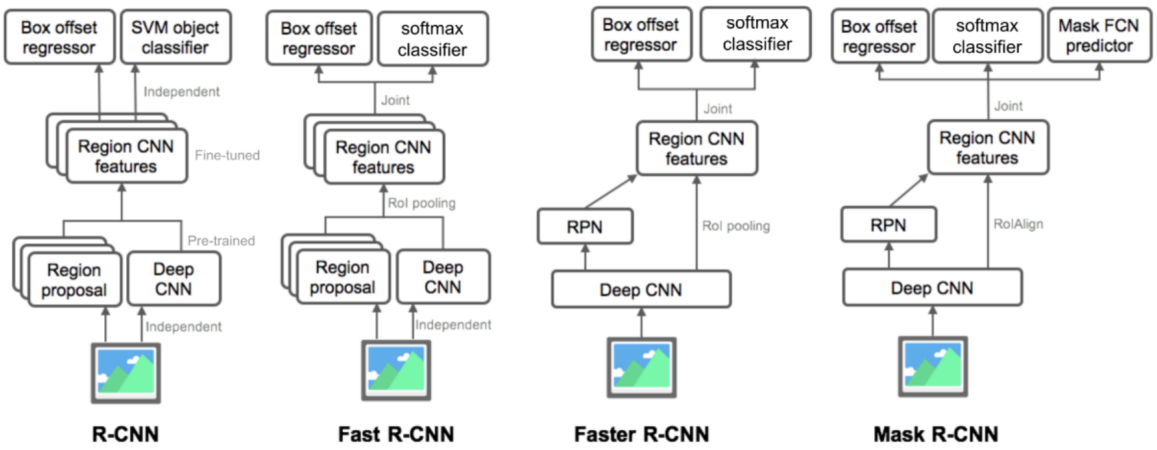

Figure 2.14: Difference of architecture between R-CNN, Fast R-CNN, Faster R-CNN and Mask-RCNN[31]

Chapter 3

METHODOLOGY

3.1 Network Architecture

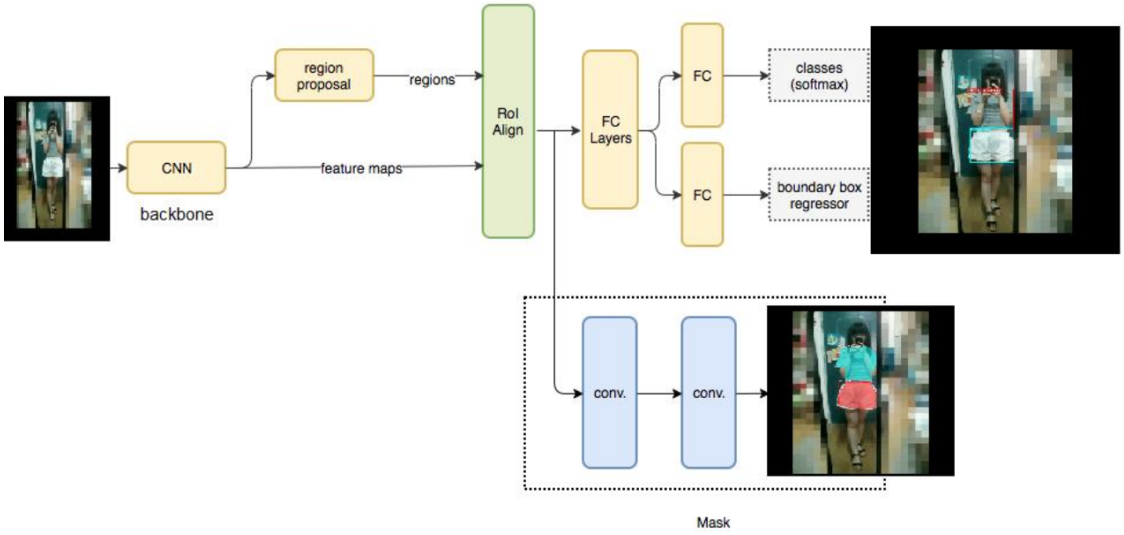

Figure 3.1: General our Mask-RCNN architecture[32]

Model workflow

1. The image input to backbone to feature extraction
2. Mask R-CNN has an additional Binary mask classifier block along with the RPN stage as in Faster R-CNN. This block generates masks for every class in the image, and the RPN will give the bounding box details.
3. Every image then goes through a CNN to generate multiple RoI using the mask classifier to obtain the feature maps.

4. The RPN block uses Non-Max suppression to get the nine anchors' highest object scores.

5. Mask R-CNN uses an RoI aligned network to get multiple bounding boxes around an object and warps it to a single dimension.

6. As in Faster R-CNN, these warped features are fed to an FC network to obtain the object class and the bounding box from Softmax and regressor blocks, respectively.

7. Along with the FC network, the warped features are fed to the Mask Classifier block, which uses multiple CNNs to generate the masks around every class's objects.

3.2 Feature extraction

The first step prediction model is feature extraction. In this step, we use backbone with Feature pyramid network and Resnet101 as bottom-up of this network.

3.2.1 Feature Pyramid Network

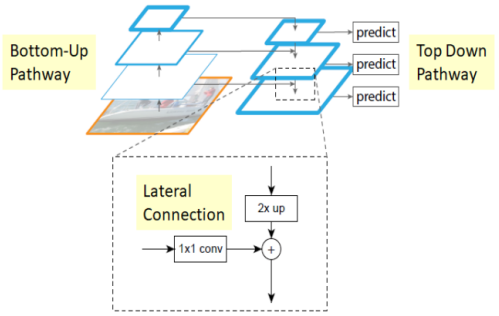

Figure 3.2: FPN architecture[33]

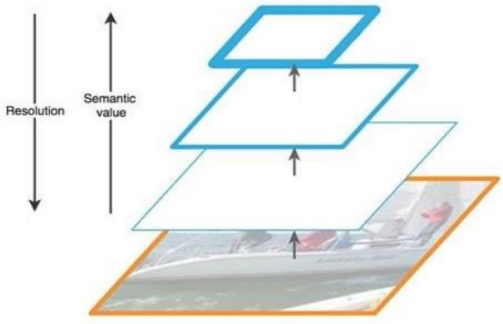

Figure 3.3: Top Down Pathway[33]

Object detection using multi-scale images for a small object is difficult. We can address this difficulty by detecting the item with a pyramid of the same picture, however it takes a long time. When speed is not a

concern, we can utilise this method. Feature Pyramid Network (FPN) is designed to address the accuracy and speed limitation issues. FPN can be replaced with the traditional feature extractors to obtain multiple feature maps, which provide better quality feature information. FPN has both bottom-up and top-down pathways. The dataflow looks as figure 3.2

The spatial resolution of the image grows as we progress from the top to the bottom levels, while the semantic value declines (figure 3.3]). The usual feature extractors used in CNNs are the bottom-up route.

Mask R-CNN follows a top-down approach. It combines semantically resilient low-resolution characteristics with semantically weak high-resolution features. FPN collects all semantic characteristics from a single scale, eliminating issues such as power consumption and memory use.

3.2.2 Channel Attention

To allow our model to emphasise important information features and suppress less useful features, we using Squeeze and Excitation Network(SE-Net) integrate bottom-up backbone ResNet101. The general architecture of SE-Net show in figure 3.4 and the SE-ResNet show in figure 3.5. he architectural unit is designed to improve the representational power of a network by enabling it to perform dynamic channel-wise feature recalibration.

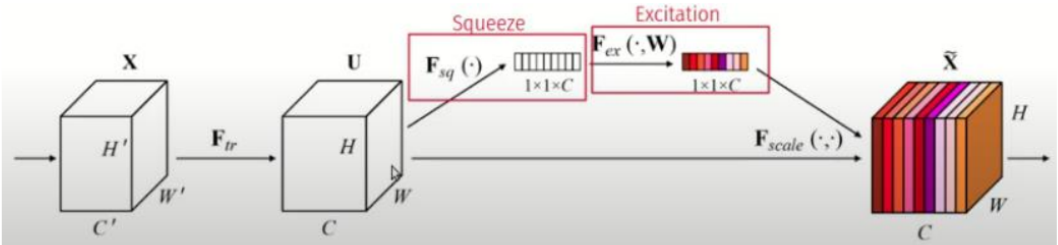

Figure 3.4: SE-Net architecture[17]

$$\mathbf{F}_{tr} : \mathbf{X} \rightarrow \mathbf{U}, \mathbf{X} \in \mathbb{R}^{H' \times W' \times C'}, \mathbf{U} \in \mathbb{R}^{H \times W \times C}$$

where: $\mathbf{F}_{tr}$ is the convolutional operator for transformation of X to U. This Ftr is the residual block. U be defined as:

$$\mathbf{u}_c = \mathbf{v}_c * \mathbf{X} = \sum_{s=1}^{C'} \mathbf{v}_c^s * \mathbf{X}^s$$

where V = [v1, v2, . . . , vc] is the learnt set of filter kernels.

Squeeze: Produce a channel descriptor by aggregating feature maps across their spatial dimension (HxW)

$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j)$$

The result of the transformation U can be thought of as a grouping of local descriptors whose statistics are representative of the entire image. It is suggested that global spatial information be crammed into a channel descriptor. This is accomplished by generating channel-specific data via global average pooling.

Excitation: Produce a collection of per-channel modulation weight

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma\left(\mathbf{W}_2 \delta\left(\mathbf{W}_1 \mathbf{z}\right)\right)$$

where δ is ReLU function

The workflow of this unit is:

Step 1: The block has a convolutional block as an input.

Step 2: Each channel is "squeezed" into a single numeric value using average pooling.

Step 3: A dense layer followed by a ReLU adds non-linearity and output channel complexity is reduced by a ratio.

Step 4: Another dense layer[34] followed by a sigmoid gives each channel a smooth gating function.

Step 5: Finally, we use the side network to weight each feature map of the convolutional block; this is called "excitation."

3.3 Region Proposal Network(RPN)

The Region Proposal Network(RPN)[1] applies a lightweight binary classifier to a large number of boxes (anchors) over the image and pro-

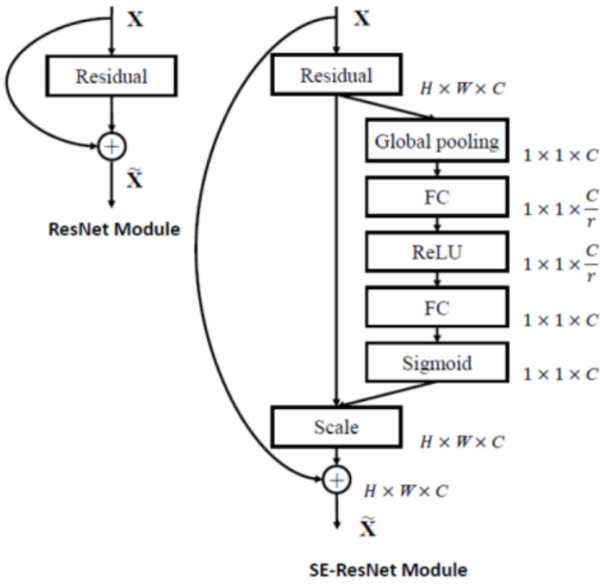

Figure 3.5: ResNet and SE-ResNet architecture[17]

duces object/no-object scores. Anchors with a high objectivity score (positive anchors) advance to the second stage of classification. Even positive anchors don't always completely cover objects. As a result, the RPN regresses a refinement (a delta in location and size) to be applied to the anchors in order to shift and resize it to the correct object boundaries.

The RPN targets are the RPN's training values. To create the targets, we start with a grid of anchors that span the entire image at various sizes, and then use a ground truth object to compute the anchors IoU. Positive anchors are those that have an IoU >= 0.7 with any ground truth object, and negative anchors are those that don't cover any object by more than 0.3 IoU. Anchors in between (i.e. cover an object by IoU >= 0.3 but < 0.7) are considered neutral and excluded from training. We compute the shift and resizing needed to have the anchor entirely cover the ground truth object when training the RPN regressor. The top ten highest anchor scores are shown in the figure below

Non-Maximum Suppression[35]: Object detection models typically provide a large number of bounding boxes as a result of their output. When such data is produced, there will be multiple bounding boxes for the same

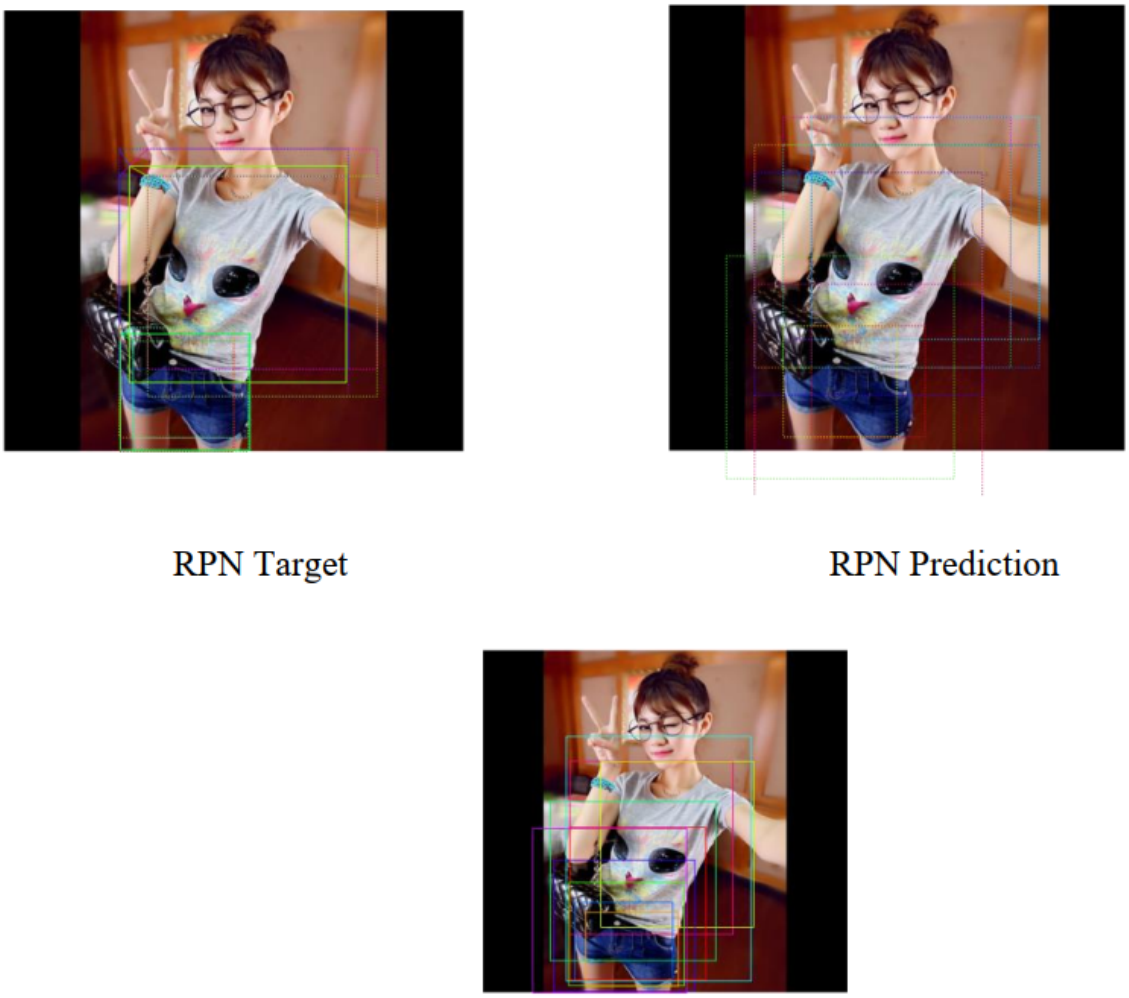

Figure 3.6: RPN output for each stage

object, resulting in information redundancy when our goal is to have only one bounding box per object. The Non-Maximum Suppression algorithm was created to solve this problem by removing redundant bounding boxes of the same object in the image.

- Input: An array of bounding boxes, each of which will have the form $(x1, y1, x2, y2, c)$, where: (x1, y1), (x2y2) are the top-left and bottom-right coordinates of the bounding box, respectively.

c is the confidence score corresponding to that box, returned from the object detection model.

- Output: An array of bounding boxes after the redundant bounding

boxes have been removed.

Following the preceding phase, all bounding boxes have been sent to the Proposal categorization stage. This stage takes the RPN's regional proposals and categorizes them. The output of this stage show in figure 3.7

3.4 Generating mask

This step takes the detections from the previous layer (fined bounding boxes and class IDs) and runs the mask head to build segmentation masks for each instance. We crop the mask pixel from image. The target mask is the truth localize pixel form dataset annotation and prediction mask is the predict localize pixel of model.

Figure 3.7: RPN classification for each stage

Figure 3.8: Target mask of dataset and prediction mask of model

Figure 3.9: Final prediction of model

Chapter 4

IMPLEMENTATION DETAIL

4.1 Dataset

To training and testing model, we use DeepFashion2 dataset. It is a large fashion database, contain 491K photos from both commercial retailers and consumers of 13 popular clothing categories. The dataset is split into a training set (391K images), a validation set (34k images), and a test set (67k images). Visualise and statistic of DeepFashion2 are shown in Figure 4.2

4.2 Implementation

4.2.1 Pre-processing

Data Argumentation

We use Data Argumentation[36] to rotate images by 25 degrees and add noise to generate more data in classes with small sample sizes.

Mini mask

We need quite a bit of memory to save the masks. Numpy uses 1 byte to store 1 bit value. Therefore, with an image size of 1024x1024, we need 1MB of ram memory to store it. If we have a dataset of about 1000 images, it requires 1GB of memory, which is quite large. In addition to consuming memory, they also slow down the training speed of the model. For improvement, instead of saving the entire mask of the whole image, Mini Mask[37] will only save the pixels of the mask in the bounding box.

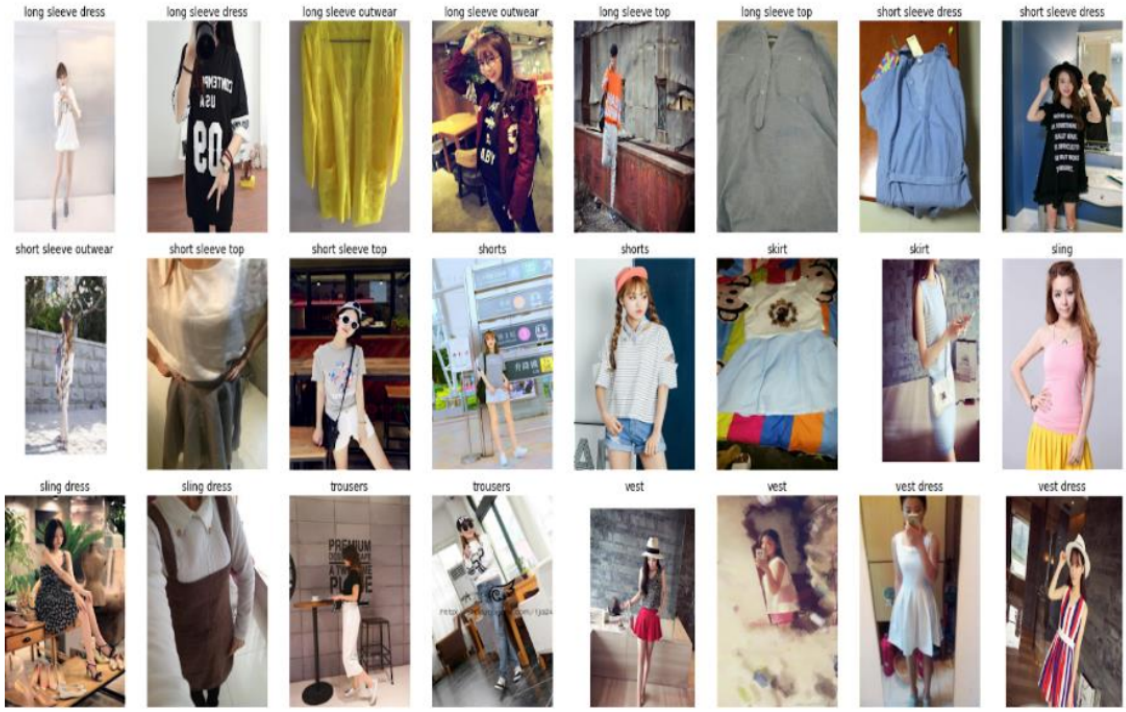

Figure 4.1: Example categories clothes image from our dataset

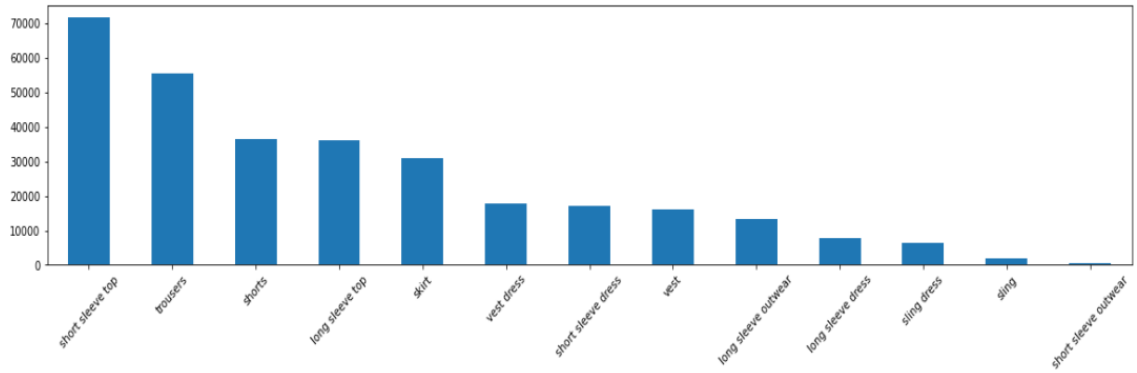

Figure 4.2: Statistic sample per categories of training set

Using this way, we will save the main memory.

4.2.2 Implementation detail

We implemented our Mask-RCNN model in Tensorflow and Keras . Because of changing architecture, we can not use the available pre-trained model weight to transfer learning. So firstly of the training step we use Common Object in Context(COCO)[38] dataset version minimal 2014 have

Figure 4.3: Example of our Data Argumentation

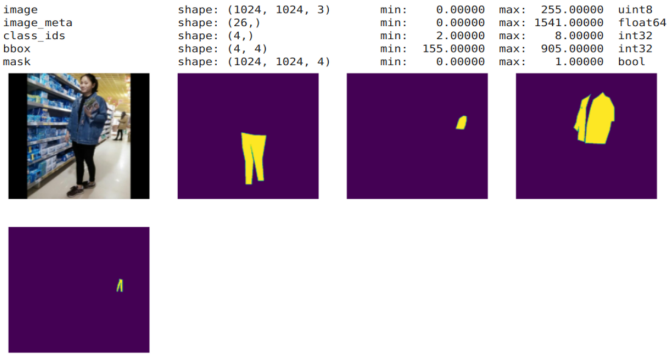

Figure 4.4: Mini mask

35k images to pre-training. Then we load weight and use DeepFashion2 to fine-tuning models, excluding the weight of fully connected layers because 2 dataset have different number classes. We limit 10000 samples for each class. All experiments were performed on an NVIDIA T4 GPU. We use SGD[39] optimizer algorithm with a weight decay of 0.0001 and momentum of 0.9, learning rate of 0.001 for 200 epochs of network head layer, 200 epoch of layer 4+ with same learning rate and 100 epoch to fine tuning all layer with learning rate 0.00025. Each epoch has 400 iterations. The training time is 20 days.

Chapter 5

EXPERIMENTAL RESULT

5.1 Evaluation Metrics

In this thesis, we evaluate Mask R-CNN's performance on various datasets by using Average Precision (AP). In general, precision is defined as

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} = \frac{\text{True Positive}}{\text{Total detections}}$$

Where,

True Positive = A correct prediction of a positive class

False Positive = An incorrect prediction of a positive class

False Negative = An incorrect prediction of negative class

True Negative = A correct prediction of a negative class

For COCO datasets or COCO type datasets, mean average precision (mAP) is often referred to as Average Precision. To better understand AP, we need Recall along with Precision. A recall is defined as

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = \frac{\text{True Positive}}{\text{Total Ground truths}}$$

AP is given by calculating the area under the Precision-Recall curve.

$$AP_T = \int_0^1 P(r)dr$$

Where, P(r) = Precision as a function of Recall T = IoU threshold (For AP50, T = 50% and for AP75, T = 70 AP for a class:

$$AP[\text{ class }] = \frac{1}{\text{Different Thresholds}} \sum AP(\text{ class, IoU })$$

AP for all classes:

$$AP = \frac{1}{\text{All Classes}} \sum AP(\text{ class })$$

Therefore, AP50 = AP at IoU @0.5 and AP75 = AP at IoU @0.75.

5.2 Qualitative Result

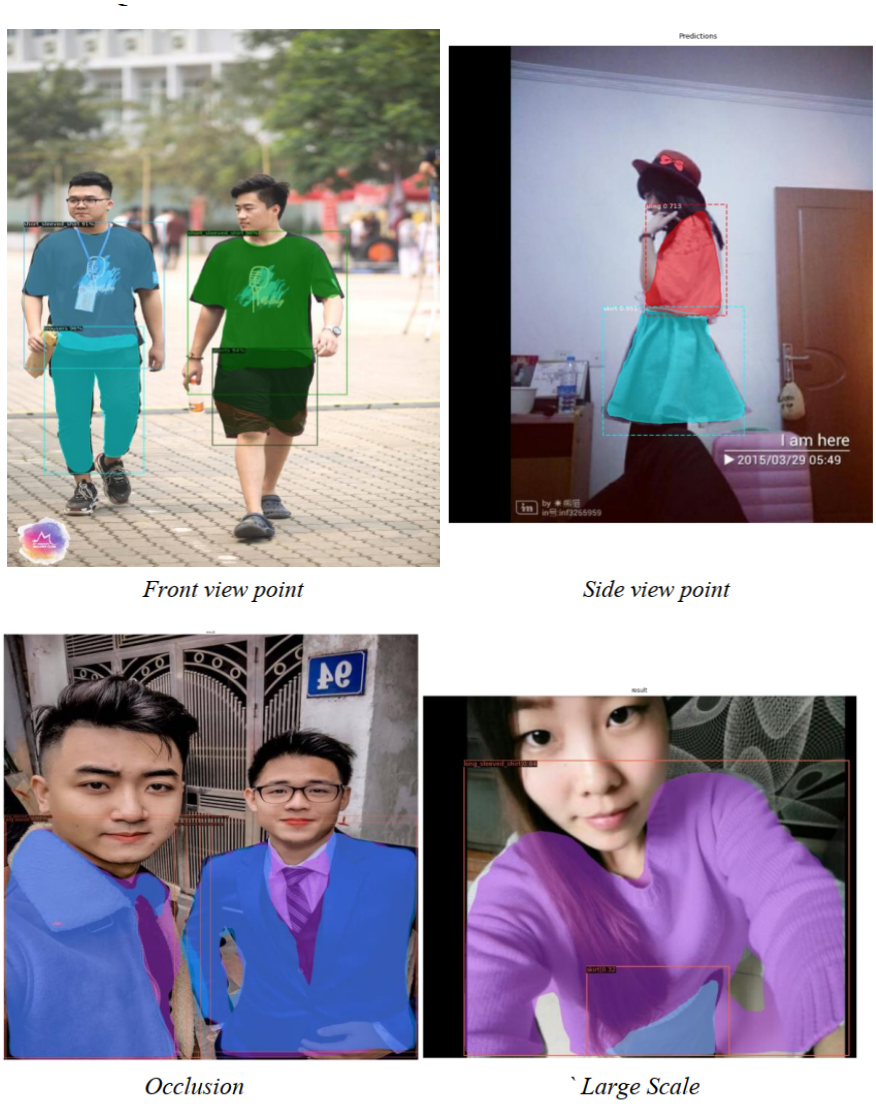

Figure 5.1: Model results on different image conditions

In Figure 5.1 are the model results under different conditions. We can see that the model is well recognized in normal conditions: the image has no noise and the viewing angle is not too narrow. In addition, for cases

where the object is influenced by multiple human pose(figure 5.2), the model still give good results. However, in the case of the person wearing the outwears, the model still cannot identify the shirt inside. With the segmentation results of both image conditions and human pose, we see that the model has not yet correctly localized the pixels at the edge of the garment. In fact, this is difficult task and unnecessary, because for applications of detection and segmentation problems, we only need to localize the pixels relatively.

Figure 5.2: Model results on multiple human pose

5.3 Quantitative Result

We have table 5.1 is the results of different IoU threshold: 0.5 and 0.75. Consider this table, the results range from 0.6 to 0.74. The area be calculated in height x width of object. The AP of small and medium object give low results, show that these object is difficult to detect.

	APbox	APmask
AP	63.395	64.207
AP(IoU=0.50)	74.340	74.304
AP(IoU=0.75)	71.164	72.028
APsmall	30.050	26.386
APmedium	44.432	37.941
APlarge	63.613	64.531

Table 5.1: AP for bounding box and mask of difference IoU and area(HxW)

5.4 Comparative with other method

We can see that the experimental results are inferior to other existing methods. Compared with aggregation and fine tuning, deepmark and deepmark++, these methods in turn use main models such as hybrid task cascade (this model is a new variant of mask-rcnn, has a large number of parameters and computational complexity. This is much larger than the other methods), centermark(this model has the smallest computational complexity when compared but gives quite good results). The match-rcnn and our model are both based on the mask-rcnn model with the same region proposal network and loss function, but different backbone(ResNet101+Channel attention of our model vs Resnet50 of original paper) and training strategy(learning rate and weight decay). Our method give lower result because we not have stronger computational power to fine tuning long schedule(original paper have 8 gpus for experiment but we only have 1 gpu) and the attention module require stronger weight to recognition what feature is important so it has omitted important information, making the extracted feature backbone becomes poor, so the model's results not as expected.

Method	APbox
Match R-CNN	0.667
Aggreation and Finetuning	0.764
DeepMark	0.723
DeepMark++	0.737
Our	0.633

Table 5.2: Compare APbox with other method

	Aggreation and Finetuning	DeepMark++	Our
Short sleeve shirt	0.867	0.804	0.703
Long sleeve shirt	0.814	0.724	0.673
Short sleeve outwear	0.54	0.347	0.417
Long sleeve outwear	0.823	0.724	0.721
Vest	0.761	0.679	0.656
Sling	0.656	0.422	0.449
Shorts	0.784	0.721	0.629
Trousers	0.81	0.739	0.658
Skirt	0.818	0.74	0.752
Short sleeve dress	0.807	0.721	0.659
Long sleeve dress	0.659	0.542	0.54
Vest dress	0.812	0.71	0.704
Sling dress	0.773	0.605	0.675

Table 5.3: Compare APbox of each class with other method

Chapter 6

Conclusion

6.1 Conclusion

This thesis proposes a lightweight method for fashion parsing tasks. Mean Average Precision is used as our evaluation metric to compare the predicted bounding box and mask with the ground truth in the image of the dataset. The methodology we choose can be integrated into any feature extraction backbone of detection and segmentation model for fashion parsing tasks by paying attention to important and less important features.

6.2 Future Work

All figures below show failure cases with clothes detection(Figure 6.1) and mask labelling(Figure 6.2). We can see that, with images that are too low quality or objects with too narrow an angle of view, are obscured or with clothing underneath is superimposed by a jacket, the model will not be recognisable. Our work is also limited by the resolution of the training data, because training high resolution images require a large amount of computing power that currently is not available. The mask-rcnn model through our testing also shows that it does not give good results with the studied problem and we believe that the above results can be improved by simpler methods and less computational resources.

Figure 6.1: Failure case with detection[10]

Figure 6.2: Failure case with mask labelling

Bibliography

[1] He, Kaiming & Gkioxari, Georgia & Dollar, Piotr & Girshick, Ross. (2017). Mask R-CNN. 2017 IEEE International Conference on Computer Vision (ICCV) 2980-2988. 10.1109/ICCV.2017.322.

[2] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. ArXiv preprint arXiv:1505.04597

[3] L. -C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille. (2018) DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. ArXiv preprint arXiv:1606.00915

[4] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg (2012) Parsing clothing in fashion photographs. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 3570–3577.

[5] K. Yamaguchi, M. H. Kiapour, and T. L. Berg. (2013) Paper doll Parsing: Retrieving similar styles to parse clothing items. In Proc. IEEE International Conference on Computer Vision (ICCV). 3519–3526.

[6] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg. (2014) Retrieving similar styles to parse clothing. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 37, 5, 1028–1040.

[7] S. Liu, J. Feng, C. Domokos, H. Xu, J. Huang, Z. Hu, and S. Yan. (2014) Fashion parsing with weak colour-category labels. IEEE Transactions on Multimedia 16, 1, 253–265.

[8] K. Gong, Y. Gao, X. Liang, X. Shen, M. Wang, and L. Lin. (2019) Graphonomy: Universal human parsing via graph transfer learning. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 7450–7459.

[9] W. Wang, Z. Zhang, S. Qi, J. Shen, Y. Pang, and L. Shao. (2020) Learning compositional neural information fusion for human parsing. In Proc. IEEE International Conference on Computer Vision (ICCV). 5703–5713.

[10] Ge, Yuying & Zhang, Ruimao & Lingyun, Wu & Wang, Xiaogang & Tang, Xiaoou & Luo, Ping. (2019). DeepFashion2: A Versatile Benchmark for Detection, Pose Estimation, Segmentation and Re-Identification of Clothing Images. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[11] Tzu-Heng Lin. Aggregation and Finetuning for Clothes Landmark Detection. ArXiv preprint arXiv: 2005.00419

[12] Sidnev, Alexey & Trushkov, Alexey & Kazakov, Maxim & Korolev, Ivan & Sorokin, Vladislav. (2019). DeepMark: One-Shot Clothing Detection. ArXiv preprint arXiv:1910.01225

[13] Sidnev, Alexey & Krapivin, Alexander & Trushkov, Alexey & Krasikova, Ekaterina & Kazakov, Maxim & Viryasov, Mikhail. (2020). DeepMark++: Real-time Clothing Detection at the Edge. ArXiv preprint arXiv: 2006.00710

[14] Tangseng, Pongsate & Wu, Zhipeng & Yamaguchi, Kota. (2017). Looking at Outfit to Parse Clothing. ArXiv preprint arXiv:1703.01386

[15] Lin, Tsung-Yi & Dollár, Piotr & Girshick, Ross & He, Kaiming & Hariharan, Bharath & Belongie, Serge. (2016). Feature Pyramid Networks for Object Detection.

[16] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. In Proc. IEEE

Conference on Computer Vision and Pattern Recognition, 770-778. 10.1109/CVPR.2016.90.

[17] J. Hu, L. Shen and G. Sun. (2018) Squeeze-and-Excitation Networks, IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132-7141, doi: 10.1109/CVPR.2018.00745.

[18] AGGARWAL, C. Neural Networks and Deep Learning: A Textbook, 1st ed. 2018 ed. Springer, 2018.

[19] J.Garbade, M. (2018) How to Create a Simple Neural Network in Python. KDnuggets. Available at https://www.kdnuggets.com/2018/10/simple-neural-network-python.html [October 2018]

[20] VERMA, Y. (2021) A Complete Understanding of Dense Layers in Neural Networks. DEVELOPERS CORNER. Available at https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/ [19 September 2021]

[21] Zhukovyts'kyy, Ihor & Pakhomova, Victoria & Domanskay, Halyna & Nechaiev, Andrew. (2019). Distribution of information flows in the advanced network of MPLS of railway transport by means of a neural model. MATEC Web of Conferences. 294. 04007. 10.1051/matecconf/201929404007.

[22] Wikipedia Contributors (2021). Sigmoid Function. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Sigmoid_function [Accessed 24 Dec. 2021].

[23] Elhamraoui, Z. (2020). Introduction to convolutional neural network. Available at https://medium.com/analytics-vidhya/introduction-to-convolutional-neural-network-6942c189a723. [28 May 2020]

[24] ALBAWI, S., MOHAMMED, T. A., AND AL-ZAWI, S. Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET) (2017), pp. 1–6.

[25] Yani, Muhamad & Irawan, S, & Setianingsih, Casi. (2019). Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. Journal of Physics: Conference Series. 1201. 012052. 10.1088/1742-6596/1201/1/012052.

[26] Girshick, Ross & Donahue, Jeff & Darrell, Trevor & Malik, Jitendra. (2013). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 10.1109/CVPR.2014.81.

[27] Girshick, Ross B.. Fast R-CNN. (2015). IEEE International Conference on Computer Vision (ICCV) , 1440-1448.

[28] Weng, L. (2017). Object Detection for Dummies Part 3: R-CNN Family. Available at https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html. [31 December 2017]

[29] Ren, Shaoqing & He, Kaiming & Girshick, Ross & Sun, Jian. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 39. 10.1109/TPAMI.2016.2577031.

[30] Varatharasan, Vinorth & Shin, Hyo-Sang & Tsourdos, Antonios & Colosimo, Nick. (2019). Improving Learning Effectiveness For Object Detection and Classification in Cluttered Backgrounds. 78-85. 10.1109/REDUAS47371.2019.8999695.

[31] Ayanzadeh, Aydin. (2019). A Study Review: Semantic segmentation with Deep Neural Networks. ArXiv preprint arXiv: 1704.06857

[32] Hui, J (2018) Image segmentation with Mask R-CNN. Available at https://jonathan-hui.medium.com/image-segmentation-with-mask-r-cnn-ebe6d793272 [19 April 2018].

[33] Abdulla, W (2018) Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow. Available at https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46. [20 March 2018]

[34] Rothe, Rasmus & Guillaumin, Matthieu & Van Gool, Luc. (2015). Non-Maximum Suppression for Object Detection by Passing Messages between Windows. In LNCS. 9003. 10.1007/978-3-319-16865-419.

[35] Huang, G., Liu, Z., and Weinberger, K.Q. (2017). Densely Connected Convolutional Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2261-2269.

[36] Jung, A., Wada, K., Crall, J., Tanaka, S., Graving, J., Reinders, C., Yadav, S., Banerjee, J., Vecsei, G., Kraft, A., Rui, Z., Borovec, J., Vallentin, C., Zhydenko, S., Pfeiffer, K., Cook, B., Fernández, I., De Rainville, F., Weng, C., Ayala-Acevedo, A., Meudec, R., Laporte, M. & Others(2020). imgaug. GitHub. Available at https://github.com/aleju/imgaug.

[37] Abdulla, W. (2017) Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. GitHub. Available at https://github.com/matterport/Mask-RCNN

[38] Lin, Tsung-Yi & Maire, Michael & Belongie, Serge & Hays, James & Perona, Pietro & Ramanan, Deva & Dollár, Piotr & Zitnick, C.(2014). Microsoft COCO: Common Objects in Context. ArXiv preprint arXiv: 1405.0312

[39] Desai, Chitra. (2020). Comparative Analysis of Optimizers in Deep Neural Networks. International Journal of Innovative Science and Research Technology. ISSN No:-2456-2165