DEGREE PROJECT IN ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

SEG-YOLO: Real-Time Instance Segmentation Using YOLOv3 and Fully Convolutional Network

ZHUOYUE WANG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

SEG-YOLO: Real-Time Instance Segmentation Using YOLOv3 and Fully Convolutional Network

ZHUOYUE WANG

Master in Information and Network Engineering
Date: December 11, 2019
Supervisor: Hanwei Wu
Examiner: Markus Flierl
School of Electrical Engineering and Computer Science
Swedish title: SEG-YOLO: Realtidsförgrundsegmentering med YOLOv3 och Fully Convolutional Network

Abstract

Computer vision technology has been widely applied to augment sports TV broadcast experience. Some of these broadcasts require accurate foreground segmentation on the outdoor sports scene, i.e., segmentation of potential sports players and highlight them. Segmentation on a full-HD video (1080p) resolution video stream with a high frame rate leads big challenge on the segmentation model. In this thesis, a deep learning-based framework is proposed for real-time instance segmentation.

Many traditional computer vision algorithms for segmentation based on background subtraction techniques, which are affected a lot by light-switch and targets' movements. On the other hand, most of the modern deep learning-based frameworks run at a deficient speed that cannot support real-time usage, although they have better robustness against scene changes. The proposed model, SEG-YOLO, is an extension of YOLO(You Only Look Once) version 3, which is one of the state of the art object detection model. The extension part is FCN(Fully Convolution Network), which is used for semantic segmentation. SEG-YOLO aims to overcome both the speed and accuracy problems on the specific outdoor sports scene, while its usage can also be generalized to some extent.

SEG-YOLO is an end to end model that consists of two neural networks: (a) YOLOv3, for object detection to generate instance bounding boxes and also for feature maps extraction as the input of phase b; (b) FCN, takes bounding boxes and feature maps as input and output segmentation masks of the objects.

For instance, segmentation in the specific outdoor sport like golf, the framework shows an excellent performance both in speed and accuracy according to the experiments, and it's superior to the state-of-the-art model. Moreover, it is proved that it can be used in real-time (30 FPS) broadcast TV with GPU acceleration. For non-specific scenes of the benchmark COCO dataset, its performance does not exceed the current state-of-the-art with respect to accuracy, but still has advantages regarding speed.

Sammanfattning

Ökad sändningsupplevelse kräver korrekt instanssegmentering på utomhussporter, dvs segmentering av potentiella spelspelare. Segmentering på full-HD video (1080p) upplösningsvideo med hög bildhastighet leder till stor utmaning på segmenteringsmodellen. I denna avhandling föreslås en djupinlärningsbaserad ram för realtidssegmentering.

Många traditionella datorvisningsmetoder för segmentering baserat på bakgrunds subtraktionsteknik, vilket drabbade mycket av ljusbrytare och målrörelser. Å andra sidan går det mesta av det moderna, djupt lärande baserade ramverket med en mycket låg hastighet som inte kan användas i realtid. Den föreslagna modellen, SEG-YOLO, är en förlängning av YOLO (You Only Look Once) version 3, som är en av de senaste teknikerna för detektering av objekt. Utvidgningsdelen är FCN (Full Convolution Network), som används för semantisk segmentering.

SEG-YOLO är en end-end-modell som består av två neurala nätverk: (a) YOLOv3, för objektdetektering för att generera instansbegränsande lådor och även för extraktion av funktionskartor som inmatning av fas b; (b) FCN, tar gränser och kartor som inmatnings- och utmatningsmaskar av objekten.

För de specifika utomhusscenerna visar ramverket en bra prestanda både på hastighet och noggrannhet jämfört med Mask R-CNN-ramverket. Och det är bevisat att det kan användas på realtidssändningstvätt med GPU-acceleration.

Acknowledgement

First of all, I would like to thank the thesis host company for offering me the opportunity to carry out my thesis in the company and also for providing computation resources and data. I want to thank my industrial supervisor Daniel and my KTH supervisor, Hanwei, for their advice during the project. Daniel always give me feedback on time, and he helped me finding someone for the data annotation works. Also, I would like to thank my examiner Markus for giving me much support during the whole project period. Many thanks to my friends and company employees, they helped me as much as they could, Ludvig, Christoffer, Dennis, Alex, Kaiyu. Last but not least, I would like to express my gratitude to my family and my girlfriend, Yini. This thesis could not be accomplished without them. Thank you.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Motivation and Aim . . . 2
- 1.3 Research Question . . . 3
  - 1.3.1 Challenges and Requirements . . . 3
  - 1.3.2 Problem Statement . . . 3
- 1.4 Demarcations . . . 3
- 1.5 Thesis Layout . . . 3

2 Related Works **5**
- 2.1 Traditional Methods . . . 5
- 2.2 Modern Methods . . . 6

3 Preliminaries **10**
- 3.1 Deep Neural Network . . . 10
  - 3.1.1 Basic Idea . . . 10
  - 3.1.2 Training a Deep Neural Network . . . 11
  - 3.1.3 Challenge of Training a Deep Neural Network . . . 12
- 3.2 Convolutional Neural Network . . . 15
  - 3.2.1 Basic Idea . . . 15
  - 3.2.2 Residual Block . . . 16
- 3.3 Object Detection . . . 17
  - 3.3.1 One-stage Object Detection . . . 17
  - 3.3.2 Two-stage Object Detection . . . 18
- 3.4 Fully Convolution Network . . . 18
- 3.5 Feature Pyramid Network . . . 18

4 Implementation **20**
- 4.1 Architecture . . . 20
  - 4.1.1 YOLOv3 . . . 21
  - 4.1.2 Intermediate . . . 24
  - 4.1.3 FCN Head . . . 26
- 4.2 Training . . . 27

4.2.1 Datasets 27
4.2.2 Data Augmentation 27
4.2.3 Implementation Details 28

5 Results **31**
5.1 Evaluation Metrics 31
5.1.1 Intersection over Union 31
5.1.2 Mean Average Precision 32
5.2 Model Performance 32
5.2.1 Speed Performance 33
5.2.2 mAP Performance 33
5.3 Result Analysis 35

6 Discussion **38**
6.1 Conclusion 38
6.2 Future Work 38

Bibliography **40**

Nomenclature

BN Batch Normalization

CNN Convolutional Neural Network

DNN Deep Neural Network

FCN Fully Convolutional Network

FN False Negative

FP False Positive

FPN Feature Pyramid Network

FPS Frames Per Second

GMM Gaussian Mixture Model

GPU Graphics Processing Unit

IoU Intersection-over-Union

KNN K-Nearest Neighbor

LTU Linear Threshold Unit

mAP mean Average Precision

MLP Multi-layer Perceptron

MOG Mixture of Gaussians

$R-CNN$ Region Based CNN

RGB Red-Green-Blue colour space

ROI Region of Interest

RPN Region Proposal Network

SSD Single Shot Detector

TN True Negative

TP True Positive

$YOLO$ You Only Look Once

Chapter 1

Introduction

1.1 Background

Many modern automation technology requires multi-sensor inputs for correct operations. Computer vision is one of the most import parts, and it has been applied in several areas, including autonomous driving, surveillance, and watching experience enhancement on broadcast TV. The thesis focuses on still camera vision for watching experience enhancement, precisely instance segmentation on sports TV broadcast streams.

In some specific outdoor sports scenes, like golf, the audience's watch experience could be enhanced by marking or highlighting the potential players and also their equipment. For instance, the golfer and their golf-club and golf-bag could be marked by precise masks covering them in a golf game (See Figure 1.1).

Figure 1.1: Highlight of the potential golfers and their equipment

This thesis aims to investigate and create a model only on golf game scene and answers the question, "What are the instances and where are their outlines?". The model could be extended for more general using except for golf games. For example, it can be used for football games, basketball games, tennis games.

Instance segmentation is a combination of three classic image detection tasks, classification, localization, and segmentation. In this thesis, the concept can be specified by two terms: (a) object detection, which contains classification and object localization; (b) semantic segmentation that generates a precision mask of the object. Each of the techniques is previously well studied.

However, the main challenges for this task are segmentation accuracy and speed. Current instance segmentation methods can be divided into two parts. The first category is, one-stage object detection model for example YOLO [1][2][3] and Single Shot Detector (SSD) [4] combined with a computer vision segmentation algorithm like MOG2 [5] and KNN [6] background subtraction. These kinds of algorithms run at high speed, but they generate bad quality masks and are sometimes unstable and will be affected by some scene changes such as illumination changes and similar environment color changes. The second kinds of models are fully deep learning-based models that consist of a two-stage object detection network and semantic segmentation network. The main drawback of these kinds of deep neural network (DNN) frameworks like Mask R-CNN [7] is that they are only capable of running at a meager frame rate, 5-10 FPS, which is not enough for broadcast usage.

1.2 Motivation and Aim

This research problem is part of the collaborative project with the host company under its broadcast technologies group. It aims to develop a real-time instance segmentator that generates masks on the human to highlight them. The segmentator will work with their current ball tracking system. For example, the segmentation masks on players can prevent them from blocking by the golf ball trace after they swing the golf club and start walking around.

The recent improvement of both speed and accuracy in object detection and semantic segmentation gives the inspiration that a better solution targeting this specific task can be developed by taking advantage of different models. Moreover, the thesis also investigates if the model can generalize

for non-specific scenes.

1.3 Research Question

1.3.1 Challenges and Requirements

This thesis encounters the main challenge that is to run the instance segmentation in real-time on a signal GPU and in the meanwhile, maintain high accuracy on the generated masks. The use case of the broadcast TV requires at least 25 frames per second processing 1080p uncompressed videos. Besides, there are many scene changes in the outdoor, including light-switch, leaf swing, shadow, etc. The segmentation accuracy is sometimes underwhelming in outdoor challenge scenes.

1.3.2 Problem Statement

The instance segmentation model takes video frames as input, and the output is masked images with detection bounding boxes surrounded by the objects. A real-time model in the thesis is defined as, on average, 25 FPS+ on a single Nvidia Titan X Pascal GPU.

The research problem can be described by the below question:

* *What is the feasible instance segmentation model for real-time used on outdoor sport scene?*

1.4 Demarcations

The task will focus on finding and developing an instance segmentation for a specific outdoor sports scene, golf, that has a difference with the relevant research field to a certain extent. The proposed model is tested on (1) public dataset COCO which is a standard dataset that measures the accuracy performance of the model. (2) The host company-owned dataset contains golf scenes only. The proposed model has some generalization to some extent, but its performance is only maximized on the specific dataset for product usage. Also, speed performance is also taken into a significant consideration, which will have a concession on accuracy performance.

1.5 Thesis Layout

The remaining chapters of the thesis are in the following structure: Chapter 2 states some previously related works target on the problem and give the brief introductions of their model. It also discusses the reason that the

related works are not suitable for specific problems. Chapter 3 presents the basic theory that our model is based on, from the most straightforward mathematical concepts to the advanced models. Chapter 4 presents our model architecture and its implementation details. The detection model and the ways to increase detection speed are also stated. Chapter 5 introduce the evaluation metrics and discusses the experiment results. In the end, Chapter 6 analyses the results and discusses future work.

Chapter 2

Related Works

In this chapter, some of the classical and state-of-the-art methods for foreground segmentation are presented.

2.1 Traditional Methods

In the traditional computer vision methods, the foreground segmentation tasks are generally implemented by the background subtraction technique.

Normally the foreground objects contain some general characteristics in the background, which are defined as a static scene that involves few changes and movements. A statistical model can describe these kinds of characteristics, for example, a Gaussian Mixture Model(GMM). GMM combines multiple weighted Gaussian models to mimic the background. Once the parameters of the GMM model is calculated, the model can present the background. The foreground object then can be detected by distinguishing pixels that do not belong to the background model. This process is referred to as background subtraction. Zivkovic et al. [8] developed a GMM based background subtraction algorithm, which can be seen as a representative algorithm for classical methods.(See Figure 2.1)

Figure 2.1: MOG algorithm for background subtraction [9]

However, the real world's scene is volatile. Many factors could change,

including light-switch, shadow, leaf-swing, raindrop, noise, etc. These changes could result in an unfavorable segmentation (See Figure 2.2).

Figure 2.2: MOG algorithm's performance under challenge scene

MOG2 [5] is a modified GMM based background subtraction algorithm, it mainly improved in (a) Add shadow detection function; (b) Use adaptive GMM against environment changes. However, it is still not good enough in our use case.

2.2 Modern Methods

In the generation of artificial intelligence, deep learning-based approaches are outstanding among almost all the algorithms. Nowadays, the success of deep learning relies on a massive quantity of data, a tremendous increase in computing power, and a better training algorithm. Convolutional Neural Network(CNN) [10] has kept driving the progress of the image recognition field, including image classification, object detection, and segmentation.

Instead of building a background model for a scene, CNN is capable of directly predicting the categories of each pixel with high accuracy, exclusive from the effect of light, shadow. DeepLab [11] is an end-to-end network that used for pixel-level prediction, which is also known as semantic segmentation (See Figure 2.3).

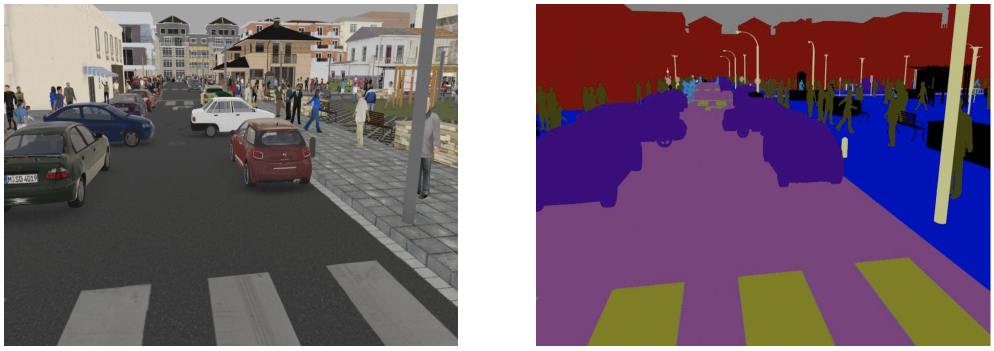

Figure 2.3: Semantic segmentation

The latest DeepLab V3+ [12] use encoder-decoder structure to improve the object's border and also use Xception as feature extraction network to enhance the classification performance. It becomes one of the typical modern methods for segmentation.

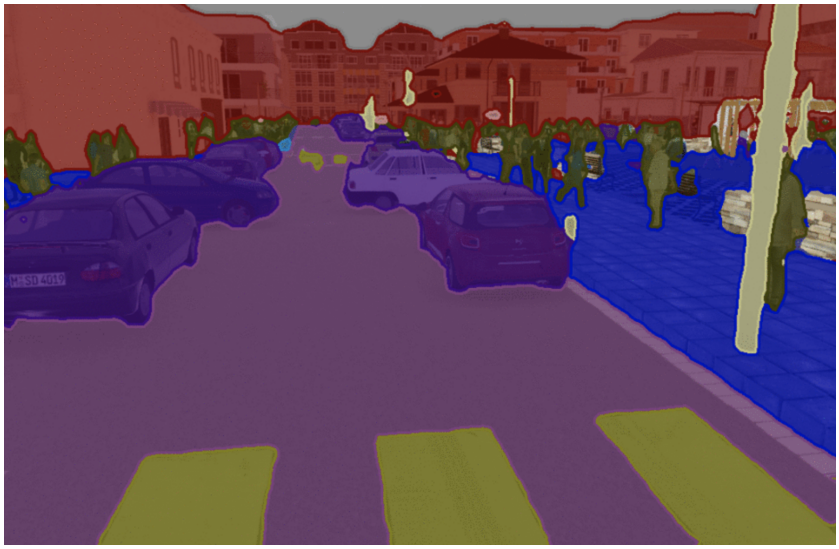

Figure 2.4: DeepLab v3+ semantic segmentation adds more details [12]

Instance segmentation not only requires the prediction of the specific class that each pixel belongs to but also it distinguishes different instances.(See Figures 2.5)

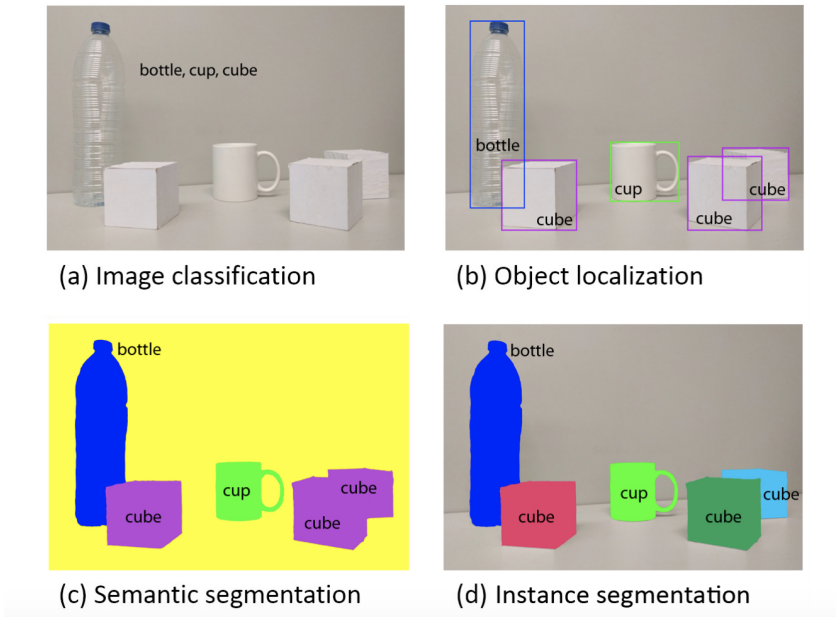

Figure 2.5: Difference between image classification, object detection, semantic segmentation [13], instance segmentation [14]

Mask R-CNN [7] is an instance segmentation model by combining object detection and semantic segmentation. It uses object detection to distinguish different objects, and then for each different object, it performs semantic segmentation(See Figure 2.6). Mask R-CNN has extraordinary performance, and it is always referred to as baseline in recent papers.

Figure 2.6: Mask R-CNN for instance segmentation [15]

However, the modern deep learning-based methods require high computation power, and they are always limited in speed. For instance, Mask

R-CNN runs only 5-9 frame per second(FPS) on a single GPU, which is too slow for real-time usage.

Chapter 3

Preliminaries

In this chapter, we will introduce some basic theory that our model is based on.

3.1 Deep Neural Network

Deep neural network has been widely used these years, in this section, we will introduce the basic idea behind a deep neural network.

3.1.1 Basic Idea

Linear Threshold Unit

A linear threshold unit (LTU) takes several inputs that associated with weights and then compute the weighted sum and followed with a step function. An LTU output can be seen as a line or hyperplane that separates the inputs.

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \mathbf{w}^\top \mathbf{x} \tag{3.1}$$

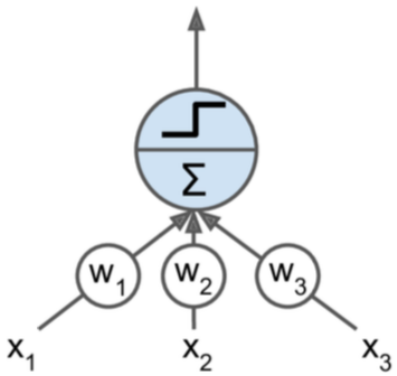

Figure 3.1: The linear threshold unit

Perceptron

The perceptron is a layer of LTU. The input fully connects to all the LTU neuron, and there is a bias neuron which inputs a constant all the time. Stacking more LTU makes perceptron simulate a complex region, for example, three LTU that output three lines which form a triangle area. With more than one LTU, a perceptron is also seen as an artificial neural network.

Multi-layer Perceptron and Deep Neural Network

The perceptron is not enough to simulate a complex non-linear function, which can be solved by stacking more layers of perceptron, which is called a multi-layer perceptron. Studies state that a two-layer perceptron followed by a sigmoid function can be used for universe approximation [16]. With more hidden layers (the middle layer except for the input and output layer), it is enough to simulate a complex function with any precision. A neural network with more than two hidden layers is regarded as a deep neural network.

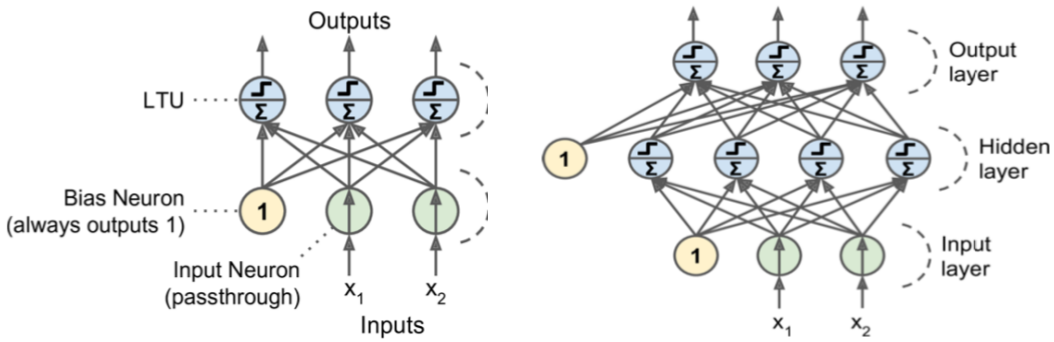

Figure 3.2: Perceptron and Multi-layer Perceptron

3.1.2 Training a Deep Neural Network

The cross-entropy between training data and the model's detection is usually the cost function of the DNN in classification problems. Training a DNN means adjusting the neuron's connection weights to minimize the cost function.

$$cost(\mathrm{y}, \hat{\mathrm{y}}) = -\sum_{\mathrm{j}} \mathrm{y_j} \log(\hat{\mathrm{y}}_\mathrm{j}) \tag{3.2}$$

y is the ground truth value which is either 1 or 0, where $\hat{y}$ is the predicted probability. We want to make the prediction values and the ground truth as close as possible.

A gradient-based backpropagation algorithm [17] to train a DNN is as follow steps:

- Forward the input through the neural network to generate prediction $\hat{y}$.
- Measure the cross-entropy error.
- Flow back the error's gradient to the lowest layer of the network.
- Change the weights and bias based on the flowed gradient.

$$\mathrm{J}\left(\mathbf{w}_{\mathrm{j}}\right) = cross\_entropy\left(\mathrm{y}_{\mathrm{j}}, \hat{\mathrm{y}}_{\mathrm{j}}\right) = -\sum_{\mathrm{j}} \mathrm{y}_{\mathrm{j}} \log\left(\hat{\mathrm{y}}_{\mathrm{j}}\right) \tag{3.3}$$

where $\mathbf{w}_j$ is the weight of the DNN and $\hat{y}$ can be seen as a non-linear function of the weight.

$$\mathrm{w}_{\mathrm{i,j}}^{(\mathrm{next})} = \mathrm{w}_{\mathrm{i,j}} - \eta \frac{\partial \mathrm{J}\left(\mathbf{w}_{\mathrm{j}}\right)}{\partial \mathrm{w}_{\mathrm{i}}} \tag{3.4}$$

3.1.3 Challenge of Training a Deep Neural Network

In reality, there are many challenges when training a deep neural network.

Overfitting

With a large number of parameters, the network has a high degree of freedom. The network is so complex that it will encounter the overfitting problem [18], which is that the network will have low accuracy on the test dataset, although it has high accuracy on the training dataset.

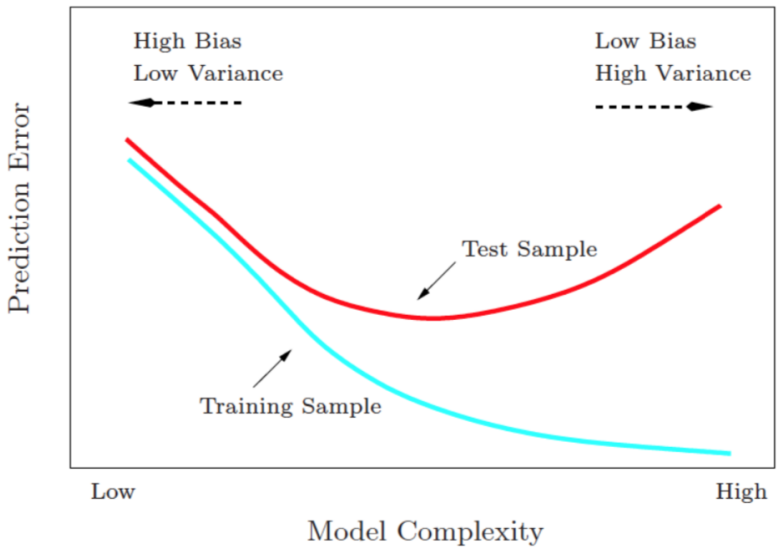

Figure 3.3: Model complexity and overfitting

Several ways in deep learning are designed to resolve the problems of overfitting.

- Regularization [19]: The penalty is added to the weight vectors in the cost function for restricting either the number or the value of the weight vectors.

- Early stopping [20]: When the training steps go by, the test error starts to increase due to the overfitting of the training dataset. The training should be stopped when the test error reaches the minimum.

- Dropout [21]: Randomly ignore some neurons during training to reduce the model complexity and increase the model sparsity.

- Data augmentation [22]: Create additional data by changing the angle, illumination, saturation, exposure, etc. In this way, the model is hard to overfit the data because of the massive data number.

Vanishing/Exploding Gradient

The back-propagation algorithm backflow the error gradient from the output to the input. Gradient will become smaller and smaller when the initial gradient smaller than one and become more substantial when it more significant than one in the beginning. As a result, the parameter will either remain unchanged or change a great value in the very deep neural network.
Overcoming the vanishing/exploding gradient [23] problem.

- Nonsaturating activation function: Using ReLU [24] instead of sigmoid activation function helps training. The gradient is minimal for the sigmoid function when it close to -1/1, which means it will saturate. Instead, ReLU = max(0, z) have bigger gradient. However, the dying ReLU problem occurs when some neurons stop output value except zero. The leaky ReLU can solve the dying ReLU problem and speed up training. LeakyReLU = max(az, z), where a is the slope when $z < 0$.

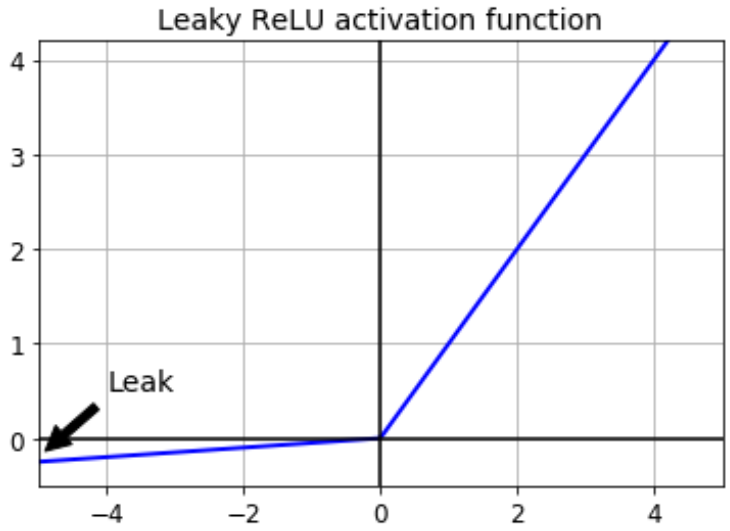

Figure 3.4: Leaky ReLU

- Gradient clipping [25]: Clip the gradients during backpropagation so that they never exceed some threshold, which solves the exploding gradient problem.

- Batch Normalization: Batch normalization [26] is a technique to address the problem that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. It is zero centering the input, then scaling and shift the result.

$$\mu_B = \frac{1}{m_B} \sum_{i=1}^{m_B} x^{(i)} \tag{3.5}$$

$$\sigma_B^2 = \frac{1}{m_B} \sum_{i=1}^{m_B} \left(x^{(i)} - \mu_B \right)^2 \tag{3.6}$$

Training Speed

With a better gradient descent algorithm, the training speed can be improved and, therefore, easy to converge.

- Momentum: Momentum is the concept that the object tends to continue moving. During training, the gradient with high momentum tends to keep moving and escape the local minimum pits. During each iteration, the local gradient multiplied with the momentum variable is added with the gradient, which is called momentum vectors. Updating the weights means subtracting the momentum vectors.

- RMSProp [27]: The learning rate will change over time, and for different parameters, the learning rate is different. The frequently changes parameters will have a lower learning rate, while the other parameters will have a larger learning rate.

- Adam [28]: The Adam algorithm combines the idea of momentum and RMSProp.

3.2 Convolutional Neural Network

In this section, we introduce both the basic idea and advanced structure of convolution neural network [29].

3.2.1 Basic Idea

The multi-layer perceptron(MLP) uses the fully connected layer as its basics, where each neuron connects with all the neurons in the previous layer. MLP usually could deal with many problems, and a deep MLP will have good results on them.

However, MLP is not a good network dealing with 2-D images. In MLP predictions, pixels of each image was flattened into a single vector as inputs, which is not an ideal solution because this process will lose the spatial information of the images. The vanilla MLP will encounter problems, including rotation, lighting, deformation, scale variation, etc.

Instead of fully connected for each neuron. Convolutional neural network takes convolutional layers as its basics. Convolution takes a filter and multiplying it over the entire area of an input image.

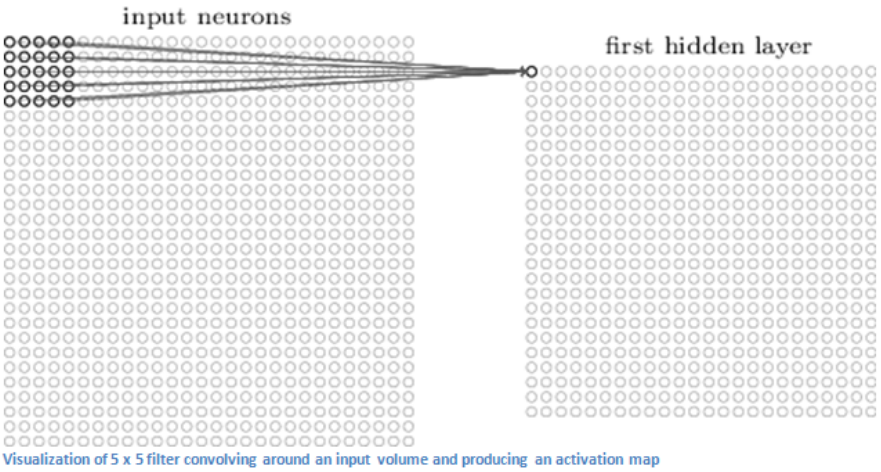

Figure 3.5: 2-D convolution

The filter is called 'kernel' in the convolutional layer, it slides over the entire image and calculates the output for each area it goes through. Formally, the convolutional layer is defined as:

3.2.2 Residual Block

Deep residual network(ResNet), was proposed by Kaiming He [30], took many championships in the classification competition including ILSVRC [31]. The network is going deeper in recent years for better results. However, the truth is that the prediction error will increase if the depth of the network is simply increased. This is mainly because stacking more layers makes the network hard to train due to the gradient vanishing and exploding problems that we mentioned above. There, the concept of residual block was then introduced.

Instead of the feed-forward structure of the original layers. Residual block introduced a short cut layer for easier learning (See Figure 3.6).

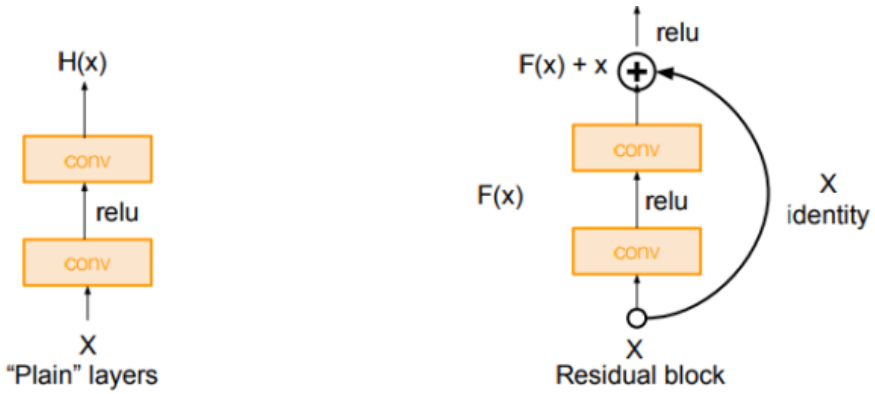

Figure 3.6: Residual Block

In the left figure, the network needs to fit the H(x) directly. On the contrary, the network can learn F(x) because x is already known. The residual shortcut layer successfully blocks the input, which is the major part when learning, and it stresses the small changing part, which is F(x).

With the concept of the residual block, modern, state-of-the-art CNN adapt the structure and make themselves deeper in the meanwhile still easy to train. With the high-level information gained from the deeper layer, these kinds of residual networks become imposing structures that widely be used.

3.3 Object Detection

Object detection [32] always conveys object classification and localization, and it is a typical problem in the computer vision. Currently, the deep learning-based object detection algorithm can be divided into two parts: one-stage object detection and two-stage object detection.

3.3.1 One-stage Object Detection

One-stage object detection algorithms, including YOLO [1] and SSD [4], are widely used on especially embedded systems because of their high speed. One-stage object detection algorithm is end to end that directly transforms the object bounding box localization problem to the bounding box coordinates regression problem. It computes the class confidence and the bounding box coordinates simultaneously.

3.3.2 Two-stage Object Detection

Two-stage object detection algorithms, including R-CNN series [7], is different in the detection process compared to one-stage algorithms. Firstly, it generates the bounding box proposal using a region proposal network (RPN). Then, the proposed region is classified using CNN. The two-stage object detection algorithms have higher accuracy than the one-stage algorithms, but they slow in speed.

3.4 Fully Convolution Network

Normally CNN will continue with multiple fully connected layers (dense layer), which project the feature map generated by CNN to a fixed-length feature vector. These kinds of classical CNN are suitable for image classification and regression problems.

Instead of image-level classification, fully convolution network [33] (FCN) classify the image at the pixel level. The convolutionalization, which is used in FCN, is the technique that is using the convolutional layer to substitute the dense layer. The output of FCN is an image labeled on each pixel.

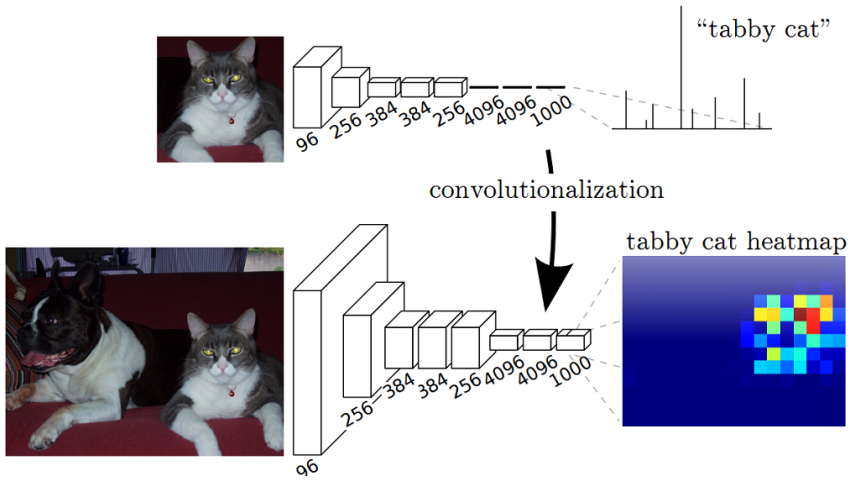

Figure 3.7: Convolutionalization and Fully Convolutional Network [33]

3.5 Feature Pyramid Network

Feature Pyramid Network (FPN) [34] is a unified structure that can be used for feature enhancement and multi-scale object detection. The author thinks that most object detection and segmentation tasks only used

the top-level feature map for prediction. However, the lower level has more accurate location information, which has been lost during the pooling. FPN not only uses different scale feature fusion but also predicts using each scales. FPN up-sampling the high-level feature map and merge with the lower level, the fusion feature map though a 3x3 convolutional layer for anti-aliasing.

Chapter 4

Implementation

In this chapter, we will introduce the implementation details of our model.

4.1 Architecture

SEG-YOLO model can be separated into 3 parts. The first part is YOLOv3, which proposes the region of interest (ROI) and regresses for the classes and confidence scores. In the thesis, a modified YOLOv3 is implemented, which also output the feature maps from every lats layers of the residual blocks. The second part is ROIAlign that takes different YOLO bounding box outputs and feature maps as inputs and generates the fixed-size ROI feature maps. The last part is Fully Convolution Network (FCN), which transforms the ROI inputs to the semantic mask outputs. The whole model architecture can be shown as below:

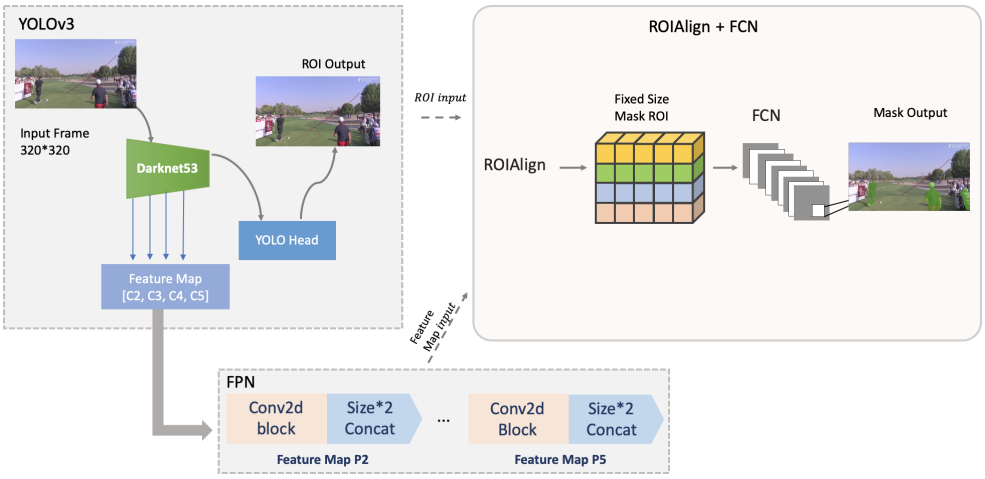

Figure 4.1: Model Pipeline

4.1.1 YOLOv3

You Only Look Once(YOLO) is an end-to-end network for object detection. YOLO splits the image to SxS block, and then each block is responsible for detecting those targets whose center points fall within the grid. After detection, Non-Maximum Suppression is used for eliminating the duplicated bounding boxes. The 3^{rd} generation of YOLO, YOLOv3 has integrated many cutting-edge technology, including residual block based backbone, feature pyramid network like network head for multi-scale prediction, batch normalization, anchor boxes prediction, etc.

Figure 4.2: YOLOv3 for object detection

Darknet53

Darknet53 is an efficient backbone for performing feature extraction. It is a very deep backbone that contains 53 convolutional layers and it also conveys many advanced structure including: (a) Residual blocks which add shortcut layers to make the network easier to train; (b) Inception structure that contains 3x3, 1x1 convolutional kernel which keep the respective field and decrease the computation cost; (c) Batch normalization layer makes the learning of layers in the network more independent of each other.

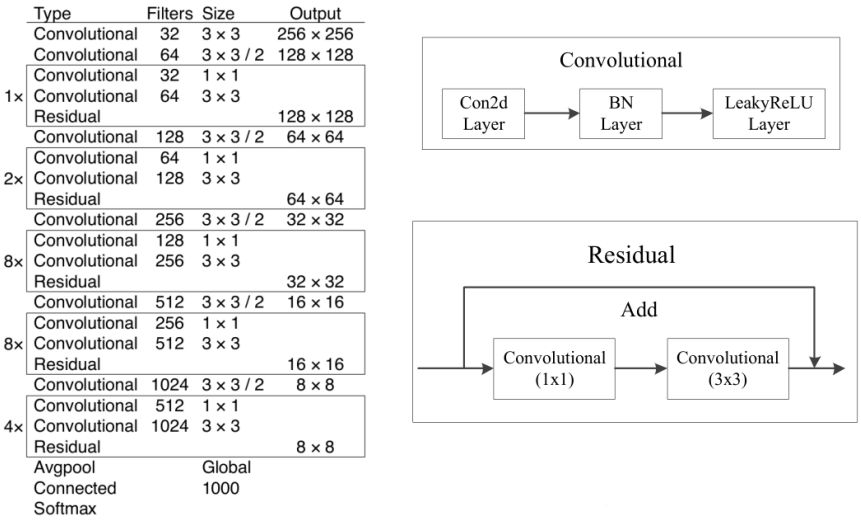

Figure 4.3: Darknet53 as feature extractor [1]

With the high efficiency of the Darknet53, it is selected as the feature extractor for both YOLOv3 head and later segmentation used. The comparison of Darknet53 and other popular backbone which are widely using in similar tasks including Mask R-CNN are shown as below:

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

Figure 4.4: Comparison of the backbone [1]

Darknet53 backbone is pre-trained on the ImageNet [35] dataset for the general feature learning step. Darknet53 has basically the same Top-1 and Top-5 accuracy on ImageNet classification task with ResNet101.However, it has fewer billions of operations and higher billion floating-point opera-

tions per second, which means it computes faster and has fewer calculations. These make Dearknet53 more efficient.

YOLO Head

YOLOv3 head adapts feature pyramid network (FPN) like structure to improve the multi-scale prediction. It makes three predictions regard to each different scales. The output tensor conveys bounding boxes coordinates, each classes' confidence scores and objects confidence (1 for object and 0 for non-object). The outputs of the YOLO head is post-processed by non-maximum suppression to eliminate the extra bounding boxes. The bounding boxes output is then prepared as ROI for the segmentation task.

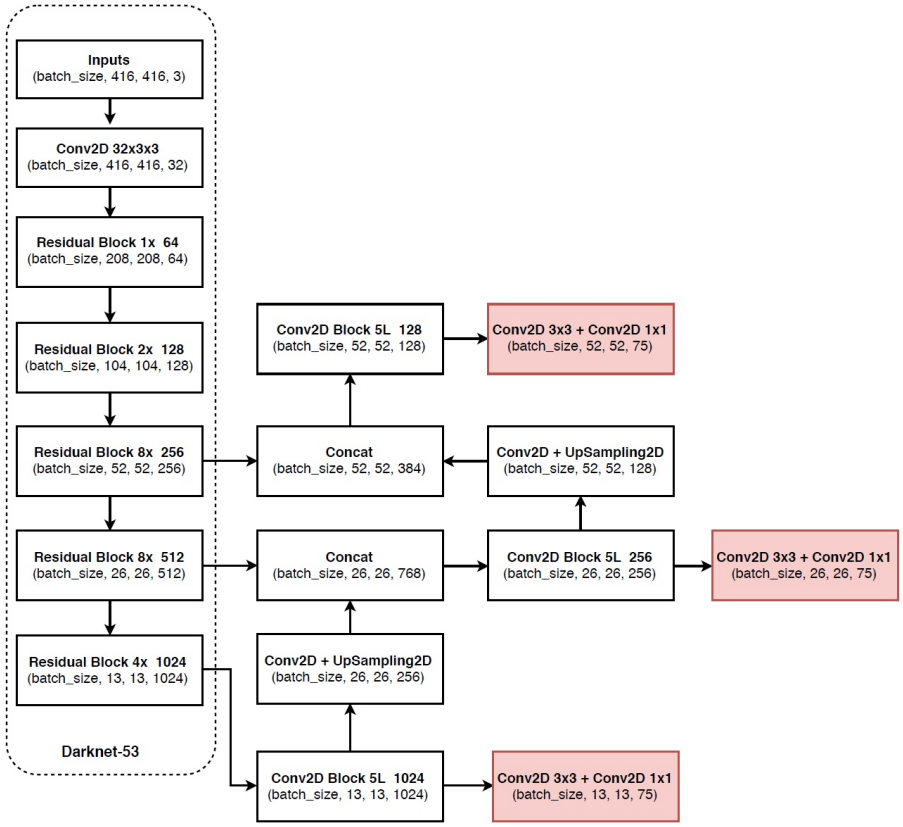

Figure 4.5: yolohead

4.1.2 Intermediate

ROIAlign

The mask generation network FCN requires fixed-size image input. Since YOLOv3 outputs different size bounding box for the object, the ROI's size needs to be unified. Faster R-CNN [36] proposed ROI pooling to rescale all the ROI regions on feature map to the same size.

ROIAlign is proposed by Mask R-CNN [7], and it is used for ROIs extraction on feature maps and then scale to the same size. The ROIAlign

is designed to fix the misalignment problem, which is that the scaled size output is not the same position as the original ROI's position. ROIAlign used bilinear interpolation for pixels value prediction instead of directly sampling the pixels. In this way, ROIAlign avoids two integer operation and keep the float point number and increase the precision.

FPN for Multi-Feature map

We extract the last layers of different feature map size from the backbone, which are layers 11, 36, 61, 74, and we name them C2, C3, C4, and C5 accordingly. When the layer goes more in-depth, the size of the feature map reduces to half of the previous layer. Then we up-sample C3, C4, C5 and do the feature fusion with their previous feature map. The fusion is named M2, M3, M4, M5, and they are intermediate feature maps. Finally, we input each intermediate feature map to a convolutional kernel and the output is P2, P3, P4, P5 (See Figure 4.6).
The feature maps set P2 to P5 is then used for ROIs extraction combining with the ROIAlign. The target feature map is selected according to the ROI's size.

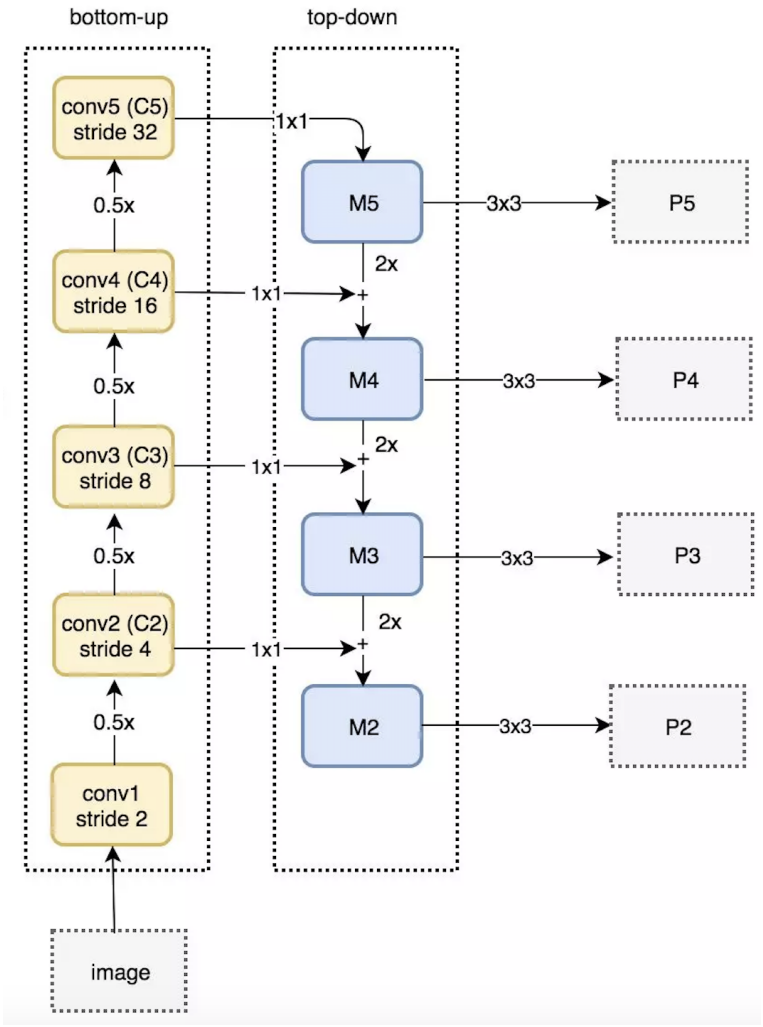

Figure 4.6: FPN for feature enhancement

4.1.3 FCN Head

Our FCN head takes fixed-size ROI and FPN's feature vector as input, and output the corresponding mask. The FCN structure can be shown as below:

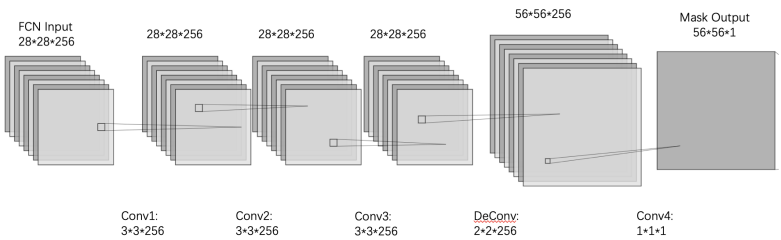

Figure 4.7: FCN head for mask generation

4.2 Training

4.2.1 Datasets

Microsoft COCO [37] is a large-scale dataset that contains annotations for object detection, segmentation, key-points detection, etc. There are 118 thousand images as a training dataset, which includes 270 thousand person instances and 886 thousand object instances.

Due to the data limitation, the model is pre-trained on the COCO dataset for the first round training. For the second round training, the model is fine-tuned by the golf games dataset, which contains 4000 images, including several outdoor scenes (See Figure 4.8).

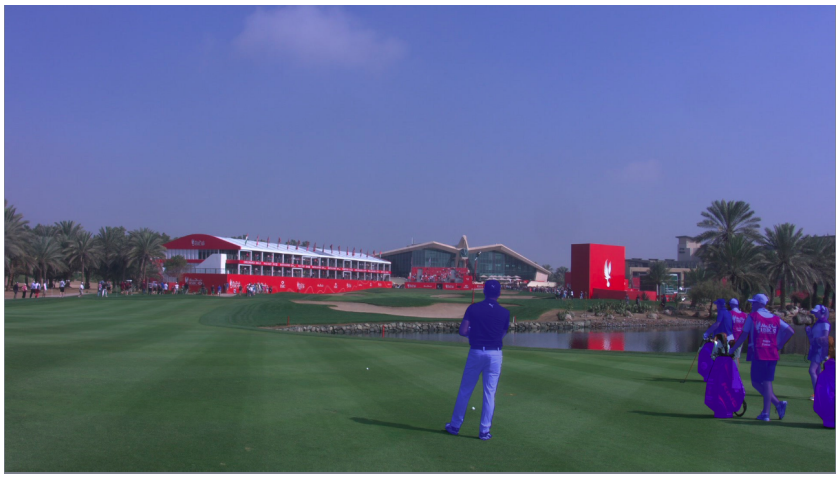

Figure 4.8: One example of golf game dataset offered by the company [1]

4.2.2 Data Augmentation

The model can be trained better by training on more data, and this will also reduce the chance of overfitting. Create fake data and add them to the dataset is a technique called data augmentation. For example, convolutional neural network has robustness against viewpoint, angle, size, illumination, and translation, but we need to tell the network that these belong to the same class, which addresses data augmentation. The same operation can be done on the training data. According to this model, the main challenge is the illumination changes so that the parameters, including saturation and exposure, are changed to generate more data.

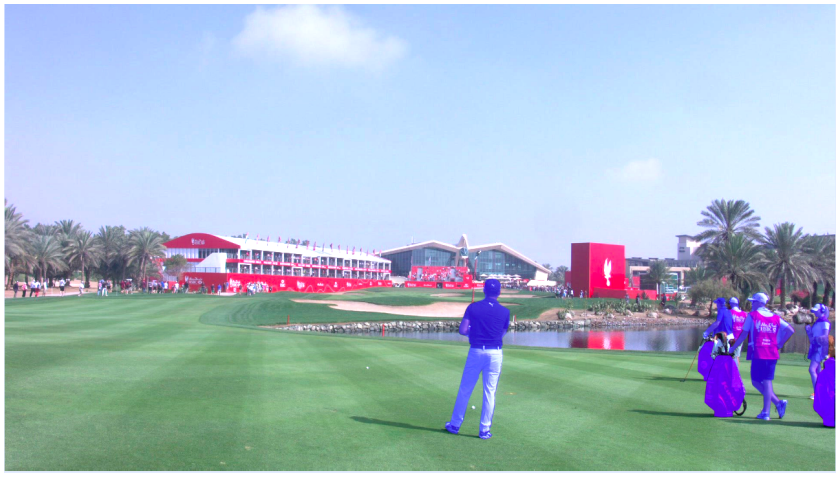

Figure 4.9: Data augmentation image that adjust exposure compare to the last image

4.2.3 Implementation Details

Our model is trained on a three-GPUs server and evaluated on a single GPU, which is Nvidia GTX1080Ti.

Chosen configuration

Both YOLOv3 and FCN head has some parameters that need to be settled because they are quite not sensitive, and they will not have a great impact on the result. The first table is YOLOv3's configuration.

Parameters	**Values**
Batch size	80
Momentum	0.9
Start learning rate	0.00033
Learning policy	Adam
Weight decay	0.0005

Table 4.1: YOLOv3 configuration

FCN head's parameters.

Parameters	Values
Batch size	18
Momentum	0.9
Start learning rate	0.001
Learning policy	Adam
Weight decay	0.001

Table 4.2: FCN head configuration

Loss Function

The model's lost function can be divided into two parts. The first part is YOLOv3's loss function.

For object detection, the classification loss at each block is the squared error for each class:

$$\sum_{i=0}^{S^2} 1_i^{\text{obj}} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c)\right)^2 \tag{4.1}$$

where $1_i^{obj} = 1$ if an object appears in the block, otherwise is zero. $\hat{p}_i(c)$ denotes the conditional class probability for class c in block i. $p_i(c) = 1$ if the ground truth is c, otherwise is zero. The total classification loss is the sum-up of loss in each grid S.

The localization error [38] measures how well the predict bounding box overlap with the ground truth bounding box.

$$\begin{aligned} &\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2\right] \\ &+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2\right] \end{aligned} \tag{4.2}$$

where λ_{coord} is the regularization term for the bounding box coordinates. There can be multiple bound box B in one grid S. The bounding box coordinate is predicted by knowing its box center $(\hat{x}, \hat{y})$ and its height and width $(\hat{w}, \hat{h})$, where x, y, h, w is the ground truth.

The confidence loss for detection or non-detection in a block is:

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 \tag{4.3}$$

$$\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \tag{4.4}$$

where 1_{ij}^{noobj} equals one if there is no object in the block and $\hat{C}_i$ is the box confidence in the block i.

The final object detection loss for YOLOv3 is the sum of the confidence, classification, localization losses.

The second part is loss for FCN which is defined as the binary cross-entropy loss [7], only including k^{th} mask if the region is associated with the ground truth class k:

$$-\frac{1}{m^2}\sum_{1\leq i,j\leq m}\left[y_{ij}\log\hat{y}_{ij}^{k}+(1-y_{ij})\log\left(1-\hat{y}_{ij}^{k}\right)\right] \tag{4.5}$$

where y_ij is the label of a cell (i,j) in the true mask for the region of size m^2; $\hat{y}_{ij}^{k}$ is the predicted value of the same cell in the mask learned for the ground-truth class k.

Chapter 5

Results

In this chapter, we present our experiment result given by our trained model.

5.1 Evaluation Metrics

5.1.1 Intersection over Union

Intersection-over-union (IoU) [39] is one of the most import evaluation metrics for object detection and segmentation. The basic idea is that it measures the overlap rate between the model's output bounding box and the true bounding box. More formally, to applied IoU we need: (a) The ground-truth bounding boxes; (b) The predict bounding boxes of our model. It can be calculated as below(See Figure 5.2):

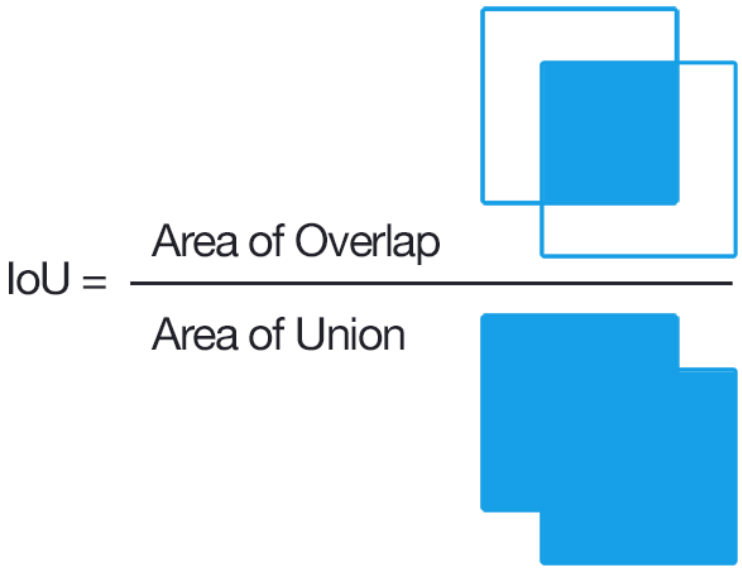

Figure 5.1: IoU Description

5.1.2 Mean Average Precision

Precision

Precision [40] measures how accurate the models are. Precision is the percentage of the correct predictions among all the predictions.

Recall

Recall [40] shows the ability to find the positive samples. It is the percentage of positive predictions among all the positive samples.

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

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

TP = True positive

TN = True negative

FP = False positive

FN = False negative

Figure 5.2: Precision and Recall

A true positive (TP) in the object detection means: Detection with IoU > threshold, on the other hand, false positive (FP) is the Detection with IoU < threshold.

mAP

Average precision (AP) [40] is a formal way that widely used in object detection and segmentation. It calculates the average precision value among all the recall values. For example, if there are M positive samples, the AP should be calculated by averaging the maximum precision value from recall=1/M to recall=M/M. The positive samples are counted when detection IoU > 0.5. Mean Average precision is the mean value for AP among all the categorises. In our thesis, the mAP should be calculated by averaging person's AP and golf bag's AP.

5.2 Model Performance

We compare our model with another famous instance segmentator Mask R-CNN. We trained the Mask R-CNN using the same dataset (Both public and own dataset) and the same GPU. The tables below show the difference between our model and Mask R-CNN.

5.2.1 Speed Performance

First, we perform the speed comparison for our model and the state-of-the-art method Mask R-CNN since the detection speed is the main consideration when we are deploying the model on a broadcast TV system. We compare our model and Mask R-CNN both on the COCO dataset and our own self-annotated dataset. We use the detection frame per second (FPS) as a speed evaluation metric. The table below presents speed performance for two models under different input sizes.

Method	**Backbone**	**Input Size**	**FPS**
Mask R-CNN	ResNet101-FPN	1024 x 1024	3.3
Mask R-CNN	ResNet101-FPN	512 x 512	9.2
Mask R-CNN	ResNet101-FPN	320 x 320	15.2
SEG-YOLO	Darknet53-FPN	608 x 608	17.4
SEG-YOLO	Darknet53-FPN	416 x 416	24.7
SEG-YOLO	Darknet53-FPN	320 x 320	**30.8**

Table 5.1: Speed Performance

The detection speed increase when we scale the input video frame to a smaller size. We achieve real-time 30 FPS detection with SEG-YOLO 320, and it is two times faster than Mask R-CNN with input size 320x320, ten times faster than the original Mask R-CNN.

5.2.2 mAP Performance

We also conduct an experiment regarding the mAP. We first calculate mAP for two models under the criteria that an object detection IOU > 0.5 is regarded as TP. We do the calculation for both the COCO dataset and our dataset.

Method	**Backbone**	**Input Size**	**mAP**
Mask R-CNN	ResNet101-FPN	1024 x 1024	58.0
Mask R-CNN	ResNet101-FPN	512 x 512	52.3
Mask R-CNN	ResNet101-FPN	320 x 320	47.9
SEG-YOLO	Darknet53-FPN	608 x 608	52.9
SEG-YOLO	Darknet53-FPN	416 x 416	49.1
SEG-YOLO	Darknet53-FPN	320 x 320	43.2

Table 5.2: mAP Performance for COCO dataset

Speed accuracy trade-offs could be easily achieved by SEG-YOLO on the COCO dataset. However, SEG-YOLO has a clear gap on mAP on the

COCO dataset compared to Mask R-CNN.

Method	**Backbone**	**Input Size**	**mAP**
Mask R-CNN	ResNet101-FPN	1024 x 1024	85.9
Mask R-CNN	ResNet101-FPN	512 x 512	83.2
Mask R-CNN	ResNet101-FPN	320 x 320	82.6
SEG-YOLO	Darknet53-FPN	608 x 608	84.1
SEG-YOLO	Darknet53-FPN	416 x 416	83.3
SEG-YOLO	Darknet53-FPN	320 x 320	82.4

Table 5.3: mAP Performance for own dataset

The performance gap between two models reduce on our own dataset since our dataset is much more easier than COCO dataset.

Then we further experiment with mAP across scales. We explicit calculate mAP for small (Area < 1024px), medium(1024px < Area < 9216px) and large(Area > 9216px) object in COCO dataset.

Method	**Backbone**	**Input Size**	S	M	L
Mask R-CNN	ResNet101-FPN	1024 x 1024	22.2	64.2	83.7
SEG-YOLO	Darknet53-FPN	320 x 320	14.9	53.3	79.9

Table 5.4: mAP Performance across scales for COCO dataset

We can know from the table that SEG-YOLO has difficulties dealing with small objects.

We also calculate small, medium and large objects' mAP in our own dataset.

Method	**Backbone**	**Input Size**	S	M	L
Mask R-CNN	ResNet101-FPN	1024 x 1024	66.4	87.8	91.2
SEG-YOLO	Darknet53-FPN	320 x 320	59.2	85.1	90.4

Table 5.5: mAP Performance across scales for own dataset

In our own dataset, SEG-YOLO has very similar performance as Mask R-CNN. They are all quite good at medium and large object detection. In our own dataset, small object are mainly golf bags which have simple shape and they are relatively easier to be predicted correctly.

5.3 Result Analysis

Compare to Mask R-CNN, SEG-YOLO runs much faster and it reaches real-time for the standard of broadcast TV. Due to its one stage structure and high-efficiency backbone (lightweight but good ability to extract image features). Since we are resizing the full-HD video to smaller size to speed up the detection, we lose many details of the small object, which is the reason that SEG-YOLO is performing unsatisfactorily on COCO dataset. The host company's dataset contains mainly people and some golf bags and they are mainly medium and large objects with smoother shape. SEG-YOLO has quite the same mAP result and outperforms on the speed. SEG-YOLO has quite good results on the host company's dataset. Below are some qualitative examples.

Figure 5.3: Qualitative examples that SEG-YOLO produces precise masks under normal light condition

SEG-YOLO also has capability dealing with weak lighting condition and shadow.

Figure 5.4: Mask prediction under dark lighting condition and shadow

However, there are also some human masks missing heads, which is one of the main problem of SEG-YOLO.

Figure 5.5: Mask prediction sometimes missing head

This frequently happens when golfers wear a white or green hat and it

appears that their hat's color is quite similar to the sky or tree nearby them, which confuses the detector.

Chapter 6

Discussion

In this chapter, we conclude the experiment result of the proposed model, including its significance and the potential improvement methods.

6.1 Conclusion

The report proposed a real-time instance segmentation model that can be used for specific outdoor sports broadcasts. The model has both state-of-the-art accuracy and speed. For the specific outdoor scenes, the model's performance is beyond Mask R-CNN. The model inference speed is three-time faster than Mask R-CNN. For the large and middle objects, the model's mAP is, on average 1% lower. Therefore, a segmentation model for real-time usage is developed and it can be used for many broadcast scenes and also the web live-streams.

6.2 Future Work

The proposed model has slightly lower performance on accuracy to ensure the real-time usage. The accuracy can be simply increased by increasing the network's input size and also the FCN's output mask size, which will result in slower detection speed. This can be probably solved by using multi-thread detection in the inference stage. For example, the detection can be divided into three threads, the first one for the preprocessing images, the second thread is for object detection and the last one for the instance segmentation. In this way, the model can run in parallel and increase the detection speed. On the other hand, the accuracy can be increased by using a stronger segmentation head instead of simple FCN. The original FCN head is relatively heavy and it's accuracy also can be improved. Single YOLOv3 for 320x320 resolution images can run at nearly 80 FPS and it reduces to 30 FPS when adding FPN and an FCN head. Recently, many researchers

worked on variants of FCN [41] [42] and made some improvements, which should be studied in more detail.

Bibliography

[1] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv* (2018).

[2] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[3] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[4] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[5] Zoran Zivkovic and Ferdinand Van Der Heijden. "Efficient adaptive density estimation per image pixel for the task of background subtraction". In: *Pattern recognition letters* 27.7 (2006), pp. 773–780.

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

[7] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[8] Zoran Zivkovic et al. "Improved adaptive Gaussian mixture model for background subtraction." In: *ICPR (2)*. Citeseer. 2004, pp. 28–31.

[9] Gustav Sandström. *Foreground detection in specific outdoor scenes: A review of recognized techniques and proposed improvements for a real-time GPU-based implementation in C++*. 2016.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[11] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2018), pp. 834–848.

[12] Liang-Chieh Chen et al. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 801–818.

[13] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[14] Alberto Garcia-Garcia et al. “A review on deep learning techniques applied to semantic segmentation”. In: *arXiv preprint arXiv:1704.06857* (2017).

[15] Waleed Abdulla. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. `https://github.com/matterport/Mask_RCNN`. 2017.

[16] Kurt Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural networks* 4.2 (1991), pp. 251–257.

[17] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. “Learning representations by back-propagating errors”. In: *Cognitive modeling* 5.3 (1988), p. 1.

[18] Douglas M Hawkins. “The problem of overfitting”. In: *Journal of chemical information and computer sciences* 44.1 (2004), pp. 1–12.

[19] Tomaso Poggio, Vincent Torre, and Christof Koch. “Computational vision and regularization theory”. In: *nature* 317.6035 (1985), pp. 314–319.

[20] Lutz Prechelt. “Early stopping-but when?” In: *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[21] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[22] Martin A Tanner and Wing Hung Wong. “The calculation of posterior distributions by data augmentation”. In: *Journal of the American statistical Association* 82.398 (1987), pp. 528–540.

[23] Sepp Hochreiter. “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pp. 107–116.

[24] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml*. Vol. 30. 1. 2013, p. 3.

[25] VH Quintana and EJ Davison. "Clipping-off gradient algorithms to compute optimal controls with constrained magnitude". In: *International Journal of Control* 20.2 (1974), pp. 243–255.

[26] Batch Normalization. "Accelerating deep network training by reducing internal covariate shift". In: *CoRR.–2015.–Vol. abs/1502.03167.–URL: http://arxiv. org/abs/1502.03167* (2015).

[27] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[28] Trishul Chilimbi et al. "Project adam: Building an efficient and scalable deep learning training system". In: *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*. 2014, pp. 571–582.

[29] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[30] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[31] Alex Berg, Jia Deng, and L Fei-Fei. *Large scale visual recognition challenge (ILSVRC)*. 2010.

[32] FF Li, J Johnson, and S Yeung. "Convolutional neural networks for visual recognition lecture notes". In: (2017).

[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[34] Tsung-Yi Lin et al. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2117–2125.

[35] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[36] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[37] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[38] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[39] Adrian Rosebrock. “Intersection over Union (IoU) for object detection”. In: *Online] http://www. pyimagesearch. com/2016/11/07/intersection-overunion-iou-for-objectdetection* (2016).

[40] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*. Vol. 463. ACM press New York, 1999.

[41] Huikai Wu et al. “FastFCN: Rethinking Dilated Convolution in the Backbone for Semantic Segmentation”. In: *arXiv preprint arXiv:1903.11816* (2019).

[42] Sharif Amit Kamran and Ali Shihab Sabbir. “Efficient yet deep convolutional neural networks for semantic segmentation”. In: *2018 International Symposium on Advanced Intelligent Informatics (SAIN)*. IEEE. 2018, pp. 123–130.

TRITA-EECS-EX-2019:822
www.kth.se