Pose Estimation of Swimmers from Digital Images Using Deep Learning

Xiaowen Cao

A thesis submitted to the Auckland University of Technology
in partial fulfillment of the requirements for the degree of
Master of Computer and Information Sciences (MCIS)

2021

School of Engineering, Computer & Mathematical Sciences

Abstract

Special sports environment of swimmers increases the difficulty of event monitoring and daily training. Computational vision makes it possible to solve these problems. The pose estimation of swimmers is a basic problem to be solved in tasks of relevant computer vision. The past methods not only require a complex implementation process, but also have a very unstable performance in the face of frequent morphological changes of swimmers, and most of them only fit the scenarios that include a single swimmer.

In this thesis, we implement a method for the pose estimation of swimmers based on deep learning, which fits scenarios containing multiple swimmers. We follow the top-down method and combine the HRNet with YOLOv5 to implement our model. Our method achieves ideal accuracy, the model is easy to be trained and deployed. In addition, we also propose a dataset with annotated key points for swimmers and a slew of datasets for swimmer detection. Our key point dataset is composed of the underwater view of swimmers. Compared with the side view, the torso of swimmers collected by the underwater view is much suitable for a broad spectrum of deep learning tasks.

Keywords: Multi-swimmer pose estimation, swimmer detection, HRNet, YOLOv5, DCNN.

Table of Contents

List of Figures

List of Tables

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgments), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Signature: Date: 20 December 2021

Acknowledgment

Firstly, I would like to express my gratitude for the financial and spiritual support of my parents during my studies with the Auckland University of Technology (AUT). During the online class, our lecturers gave me a good learning environment at home. The supports give opportunity for me to harvest the master's degree.

I also sincerely thank my supervisor Wei Qi Yan for his guidance in this study. His profound theoretical foundation, rich practical experience, rigorous and realistic academic attitude have benefited me a lot. His teaching has given me great help, enriched my research topic, and enlightened my knowledge and experience. In addition, I would like to thank the administrators of AUT, who have helped me solve a spate of problems.

s

Xiaowen Cao

Auckland, New Zealand

December 2021

Chapter 1
Introduction

In this chapter, we introduce background and motivations, the research questions, the contributions, objectives, and structure of this thesis, respectively.

1.1 Background and Motivation

Swimming is a popular event in sports competitions. However, special environment of swimmers increases the difficulty of event monitoring and daily coaching, it is coriaceous for people to intuitively understand the real-time dynamics and movement details of swimmers. After the competition, analysts often need to replay game footage and spend hours manually recording player gestures and analysing events to insights from the footage (Lienhart *et al.*, 2018). This has brought great resistance to swimming competitions and swimmers' daily training as well as limited the development of swimming.

In recent years, swimming analysis has been combined with machine vision, which has intelligent algorithms that automatedly and tediously analyses instead of using human eyes. Computer vision is able to make efficient, fast, and accurate analyses of movements, conditions, and environment, which help referees effectively supervise sports events, so as to obtain more accurate scores. In addition, these methods analyze athletes' performance from a more comprehensive perspective than human beings, effectively optimize athletes' performance and reduce athletes' risk as much as possible (Thomas *et al.*, 2017).

Pose estimation of swimmers is the basis of swimming-related machine vision tasks. The result of pose estimation is the most essential description of human motion that can be used as the input feature of machine vision-related tasks, such as motion recognition of swimmers, kinematic pose rectification of swimmers, drowning detection and rescue, so on (Zecha *et al.*, 2018; Liu *et al.*, 2015; Moeslund & Granum, 2001).

In the early days, underwater surveillance systems were not perfect that made it impossible for machine vision, which relies on sensor equipment (cameras, etc.), to develop rapidly in the swimming industry, most poses of swimmers are estimated from a side perspective or over water view with visual data taken from an ordinary camera installed outside of the transparent swimming lanes (Haner *et al.*, 2015; Woinoski *et al.*,

2020). In recent years, with the maturity of intelligent surveillance systems, important swimming events began to introduce underwater monitoring, for instance, in 2016, Rio Olympic Games, underwater photography robot carrying the new Canon 1DX MarkII made its debut, the underwater tracking camera also came out not long ago. Relevant equipment is also gradually popular in ordinary swimming pools (Giblin *et al.*, 2016; Sha *et al.*, 2013).

In the past, conventional methods based on graph structure or background modeling are thought as the most common solution to the problem of swimmer pose estimation (Tsumita *et al.*, 2019). However, these methods suffer from unsatisfactory detection accuracy, speed of detection, and the accuracy decreases further in the face of the unfamiliar pose or scene (Zhang *et al.*, 2021). With the adventure of big data, the emergence of abundant data and computing resources (e.g., GPU, etc) led to the burgeoning deep learning which is highly dependent on the amount of data (Cheng *et al*, 2017). Compared with the methods which rely on manually designed features, the methods based on deep learning using CNNs (i.e., convolutional neural networks) extract features from a large amount of data which obtains more abundant and effective information (Patel & Kalani, 2021).

Most of the methods proposed in recent years for the pose estimation of swimmers are only applicable to scenarios with a single swimmer, which is called the method of single-swimmer pose estimation. These methods only fit for post-match analysis or training based on the appropriate scene, which is unable to be applied in real-time interactive scenarios that include several swimmers, such as swimming contests. As a result, it is necessitous to devise a method that fits for the scenarios containing multiple swimmers, which is called the method of multiswimmer pose estimation in the following.

1.2 Research Questions

In order to fill the technical gap, in this thesis, we study the existing pose estimation and purpose to realize the method for the multiswimmer pose estimation based on deep

learning and optimize its performance. The main research questions of this thesis are as follows:

(1) Identifying the gaps in the pose estimation of swimmers through studying the development process and current situation of the related methods.

(2) What kinds of technology can be implemented to realize multi-swimmer pose estimation?

(3) For multiswimmer pose estimation problems, what are the superiorities of our approach compared with the antecedent approaches on accuracy and speed?

During the research process of this thesis, we need to choose an appropriate method based on deep learning to procure pose estimation of swimmers in scenarios with multiple swimmers. We need to make datasets and train our deep neural network to implement the function of the model. We adjust the training parameters several times and evaluate the performance of the trained models so as to obtain the best results.

1.3 Contributions

In this thesis, we implement the pose estimation method that is suitable for multi-person scenes based on deep learning. We follow the top-down method and construct the multi-swimmer pose estimator by combining the object detection model with the single-person pose estimation model. Moreover, underwater monitoring opens up a new view for machine vision related to swimmers. Therefore, unlike most previous studies, we decide to utilize the visual data obtained from the underwater perspective to implement our model. Specifically, we mainly make the following contributions:

(1) We propose an end-to-end multiswimmer pose estimation model. We use an interface to link the swimmer detection model with the single-swimmer pose estimation model to estimate the pose of swimmers in a scene containing multiple swimmers.
(2) We propose an annotated keypoint dataset with 2,500 images, which is the first dataset made of an underwater perspective and with scenes containing multiple

swimmers.

(3) We obtain the best performing models for pose estimation of a single swimmer by training two sizes of HRNet and performing multiple evaluations and optimization of the experimental results. Compared with previously proposed methods, our network has advantages in feature extraction as well as computational speed.

(4) We propose three annotated datasets for swimmer detection which include 421, 1755, and 3700 images respectively. At present, the dataset for swimmer detection is very rare, and our dataset is the first dataset for swimmer detection made using underwater perspective, which is suitable for deep learning.

(5) By training YOLOv5, we successfully implemented a swimmer detection model. Compared with the swimmer detection model implemented by traditional methods, our model has advantages in speed and accuracy.

In addition, in this thesis, we summarize and analyze the development of methods for pose estimation of swimmers, and systematically describe the technical status in this field. Our proposed method makes up for the technical gap in multi-swimmer pose estimation and swimmer detection in this field.

1.4 Objectives of This Thesis

Firstly, the existing human pose estimation methods are reviewed, the work of pose estimation for swimmers based on related pose estimation techniques is introduced, then the gaps in the field of pose estimation of swimmers are analyzed. Secondly, we propose a method for multiswimmer pose estimation. Finally, we implement and assess the method we propose.

In addition, we choose the top-down method to realize multiswimmer pose estimation. This kind of method is able to be interpreted as the assembly of an object detection model and a single-person pose estimation model. Therefore, we also introduce object detection methods and then select the appropriate method to realize the detection of swimmers.

1.5 Structure of This Thesis

Other chapters of the thesis are listed as follows:

- **Chapter 2: Literature Review.** In this chapter, we firstly retrace the deep learning method based on DNN. Then, we review the pose estimation techniques into two categories. The first part is for the single-person pose estimation. We introduce the traditional machine learning methods and deep learning-based methods respectively. The second part is for multi-person pose estimation. We introduce the top-down method and the bottom-up method respectively. In the top-down method, we depict the object detection technology and its application in multi-person pose estimation. In addition, while introducing each kind of method, we also review the related work of swimmer pose estimation based on this kind of method.
- **Chapter 3: Methodology.** Our proposed multiswimmer pose estimation model is interpreted in this chapter. This chapter is divided into four parts. The first part gives an overview of the multiswimmer pose estimation model. In the second part, we introduce the structure and implementation of HRNet for single-swimmer pose estimation. In the third part, we introduce the structure and implementation method of YOLOv5 for swimmer detection. In the fourth part, we introduce the YOLOv5 interface for the connecting of YOLOv5 and HRNet. What's more, the information for the dataset and experiment is also stated in this chapter.
- **Chapter 4: Result.** In this chapter, we implement the proposed method and show the evaluation and performance of the model. The evaluation results and performance of the YOLOv5 trained on three different sizes of datasets, the evaluation results and performance of the single-swimmer pose estimator based on HRNet, the evaluation results of multiswimmer pose estimation model based on the combination of YOLOv5 and HRNet, are demonstrated respectively. Moreover, we objectively analyzed the limitations of the above experiments.
- **Chapter 5: Analysis and Discussions.** We systematically summarize and analyze the experimental results, in this chapter.

- **Chapter 6: Conclusion and Future Work.** The conclusion of this thesis and the future work are unfolded in this chapter.

Chapter 2
Literature Review

In this chapter, we firstly overview the deep learning method based on DNN. Then, we group the pose estimation into two categories. The first part is the single-person pose estimation model. The second part is the multi-person pose estimation model. In the top-down method, we introduce object detection technology.

2.1 Introduction

The pose estimation of swimmers is regarded as a problem of human pose estimation, which means to calculate the pose parameters of various parts of the human body according to visual information. It mainly includes two-dimensional and three-dimensional methods. Two-dimensional estimation refers to the calculation of the two-dimensional coordinates of each key point of human body in the image plane. The focus of our research work is mainly on two-dimensional human pose estimation, which referred to as human pose estimation (Parekh & Patel, 2021).

According to the number of detected objects, human pose estimation is grouped into two categories: Single-person pose estimation and multi-person pose estimation (Zheng *et al.*, 2020). In this chapter, we firstly introduce the deep neural networks, then we interpret the two kinds of techniques mentioned above respectively. At present, most works on pose estimation of swimmers are based on single-human pose estimation methods, these works are also reviewed in the part of single-human pose estimation.

2.2 Deep Learning

Deep learning was firstly proposed in 2006 (Hinton & Osindero, 2006) which specifically refers to learning the regularity and representation of data through deep neural networks (DNNs). Neural network is composed of neurons with weights and biases. It is regarded as a model for information processing by simulating biological neurons. In the process of training, by adjusting the weight and bias of neurons, we finally obtain a model that processes the input information close to or in line with our expected output.

The mathematical representation of neurons was proposed in 1943 (McCulloch and Pitts, 1943) which proved that neurons can simulate different logical operation by connecting with each other and running synchronization. This is the earliest research on neural networks. The "deep" in DNN refers to a series of layers with more nonlinear operations than shallow learning. A shallow neural network generally consists of an input

layer, 1-2 hidden layers, and an output layer, while the hidden layer of the deep neural network is usually larger than 5 layers, which has the ability of deeper abstraction and dimensionality reduction (Bashar, 2019; Schmidhuber, 2015).

In the fully connected DNN, the neurons of each layer are connected separately to all neurons of the adjacent layers. The potential problem of fully connected DNN structures is an inflation of the number of parameters, a deficit that is particularly apparent when the inputs of the network are images. The image is made of pixels, and each pixel in turn is made of colour. For instance, each pixel in a 1000 × 1000 picture has three parameters to represent colour information. If we use a DNN containing 1M neurons in a single hidden layer to process this picture, this hidden layer is able to generate 3×10^{12} parameters. In the process of training, such a large amount of data easily leads to over fitting, the model is easy to learn unimportant features, and consumes a lot of computing resources. In addition, the spatial structure of the image is not taken into account in the process of digitization, so the network hardly retains the spatial features. Therefore, in the past, image-based machine vision tasks are a big challenge in deep learning (O'Shea & Nash, 2015).

While observing the outside world, human eyes usually observe the local information of the object first, and obtain the global information through the local information. During this process, the local features of the object (such as contour, boundary, human eyes, nose, mouth, etc.) are employed. The neocognitron (Fukushima & Miyake, 1982) was evolved from the human characteristics. Following the idea of neocognitron, the first Convolutional Neural Networks (CNN) was proposed (LeCun *et al.*, 1989).

For image recognition and classification tasks, the conventional classifier such as SVM (Cortes & Vapnik, 1995), takes use of artificial features that was designed based on the priori knowledge, like HOG feature (Dalal & Triggs, 2005) and SIFT feature (Lowe, 2004). The outstanding performance of CNN stems from its ability to extract efficient

representations such as high-level features from a vast amount of originally input data by using statistical learning methods (Alom *et al.*, 2019).

The convolutional layer is the core of CNN. Different from the traditional fully connected network, the neuron nodes in the convolution layer no longer need to connect all neurons in the previous layer but only feel the local area of the previous layer through the convolution core, the local area is also called the receptive field of convolution core. In addition to convolutional layers, input layers, excitation layers, pooling layers, fully connected layers are also typical structures of CNN. These layers are described in detail in the following:

- **Convolution layer.** Convolution operations refer to filtering small areas of an image by using a filter (convolution kernel) to obtain the eigenvalues of those small areas, which is also known as feature mapping. Once the local feature is extracted, its location relationship with other features is determined. In practice, there are often multiple convolution kernels. If an image block has a large value with this convolution, it is considered that the image block is very close to this feature. This also reflects on special structures, CNN is able to learn some information related to spatial structures. Specifically, if the number of input feature maps (i.e., the number of input channels) is n, the number of convolution layer filters (convolution cores) is m, each of M filters has N channels, which should be convoluted with the input n channels to obtain n feature maps, and then sum the n feature maps with the offset (one filter corresponds to one shared offset) to generate one feature map as the output of the convolution kernel(i.e., channel fusion).

  Similarly, the operations are also performed on other M-1 filters. Therefore, the final output of this layer is feature maps, which means the output channel is equal to the number of filters (Dumoulin & Visin, 2016). In this process, all elements on the same feature graph output share the same convolution kernel, which is called weight sharing. The convolution kernels corresponding to different feature maps are diverses. Therefore, all elements (neurons) on the same feature map (the same channel) are the same feature detection for different locations of the image. The weight-sharing

structure of the feature map reduces the complexity of the network model and the number of weights (Yi *et al.* 2016). This advantage is much evident when multidimensional images are input into the network. With regard to images, without convolution, the number of learning parameters is catastrophic (Albawi *et al.* 2017).

- **Excitation layer.** In CNN, the convolution operation is a linear operation of weighted summation. If the neural network only includes convolution layers, the output is a linear combination of inputs no matter how many layers there are, the expression ability of the network is limited, so it cannot learn the non-linear function. Therefore, CNN introduces an activation function that often acts on each neuron output from the convolution layer and the connective layer, which introduces a nonlinear factor to the neurons, makes the network more expressive, and nearly approaches any functions, so that neural network is applied to many nonlinear models. Activation functions are *sigmoid* and *tanh* function, ReLU function, and so on (LeCun *et al.* 2015; Krizhevsky *et al.* 2012).
- **Pooling layer.** In CNN, the pooling layer usually follows the convolution or excitation function, in order to simplify the output of the convolution layer. It divides the input into non-overlapping regions, reduces the resolution of the feature maps through pooling operation for each region. For example, max pooling is to filter the maximum value in the region and mean pooling is to calculate the average value for each region. After convolution operations, the image is still large, pooling is able to further reduce the data dimension, which not only greatly retrenches computation cost, but also helps avoid overfitting. In addition, pooling operation makes feature extraction not greatly affected by the change of target location (LeCun *et al.* 2015).
- **Full connection layer.** After multiple operations, the input of the network is output into multiple groups of feature maps, which are successively combined into a group of signals after full connection operation. Specifically, firstly it maps all the two-dimensional output of feature maps from the previous convolution layer or pool layer onto a one-dimensional feature vector, and then sum all the features with different weights to obtain the input expression. Finally, the model outputs a vector whose dimension is equal to the number of categories (the number of output neurons), which

represents the probability of each category. Then, the category with the highest score is judged as the input category. Full connection layer plays the role of “classifier” in CNN (O'Shea & Nash, 2015).

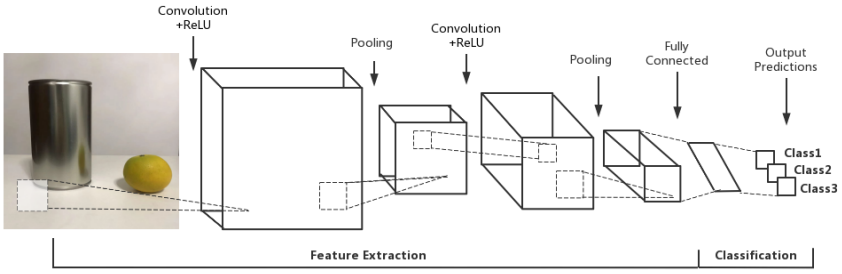


Figure 2.2: An example of the structure of CNN.

As shown in Figure 2.1, a complete convolution network is usually formed by repeated stacking of the structures. So far, many classical CNNs have been proposed which have some differences in organizational structure, but the basic network layers are similar. For instance, LeNet (LeCun *et al.*, 1998) is one of the earliest CNN models, which consists of two convolutional layers, two pooling layers, and two fully connected layers. The first CNN named AlexNet (Krizhevsky *et al.*, 2012) is regarded as a deeper and broader version of LeNet. It has five convolutional layers, three of them are connected to the max pooling layer, and three fully connected layers.

AlexNet takes advantage of ReLU as the activation function to solve the gradient dispersion problem of *sigmoid* function if the network is deep. It is worth mentioning that this study successfully applies CUDA to accelerate the training of DCNN and utilizes the powerful parallel computing ability of GPU to process a number of matrix operations during training (Sze *et al.*, 2017). The VGG network proposed by Simonyan and Zisserman (2014) takes small convolution kernels to replace the large convolution kernel to obtain a large receptive field. Their research outcome has proved that increasing the depth of the network is able to improve the final performance of the network to a certain extent and greatly reduce the error rate. In addition, model generalization is also enhanced. The full connection layer is replaced by the average pooling layer in GoogLeNet (Szegedy *et al.*, 2015), which greatly reduces the number of parameters. In

addition, 11 convolutions in the Inception module of GoogLeNet are utilized to reduce the dimension of the input signature graph, thereby reducing the amount of computation. Based on the above design, GoogLeNet has a deeper network than AlexNet and VGG with less computation and higher accuracy.

The well-known Batch Normalization (BN) method was proposed in GoogLeNet and Inception V2 (Ioffe & Szegedy, 2015). BN is a very effective regularization method, which is able to speed up the training of large convolution networks, the classification accuracy is greatly improved after convergence. Although the deeper the network is theoretically, the higher its performance is. In fact, if the network reaches a depth, the phenomenon of gradient disappearing becomes more and more obvious, the training effect of the network is not ideal, which limits the depth of the network. ResNet (He *et al.*, 2015) is related to residual network, which solves the problem of network disappearance due to the gradients by applying a shortcut connection between the output inputs instead of a plain stacked network. Compared with ResNet, DenseNet takes use of a dense connection mechanism. The input of each layer of the network includes the output of all previous layers. DenseNet improves the performance of the network through feature reuse (Huang *et al.*, 2017).

2.3 Single-Person Pose Estimation

In the past, most of the research methods are related to human pose estimation by using traditional methods based on graph structure. After 2014, the methods based on deep learning began to turn up. This kind of method broke the limitations of many traditional methods, improved the accuracy and speed of pose estimation to a new level. In this chapter, we classify and discuss the existing related technologies based on these two kinds of methods.

2.3.1 Traditional Methods

The basic idea of the conventional single-person pose estimation algorithm is to construct the human pose template library based on prior knowledge and utilize template matching to detect the human pose. Because the human body is non-rigid and there are various kinds of gestures, it is difficult for the human body template library to cover all kinds of human poses. In order to clinch this problem, pictorial structure algorithm is proposed (Fischler & Elschlager, 1973). This kind of methods adopt artificially designed features to represent the parts of the human body and the pair-wise bearing between the parts is constrained by using a spatial model, which makes the template matching have some flexibility while being reasonably constrained. Before 2012, the pictorial structure algorithm is widely harnessed in the tasks of pose estimation (Josyula & Ostadabbas, 2021).

A method (Zecha *et al.*, 2012) was proposed to estimate the pose of swimmers through part-based models that are trained discriminatively (Felzenszwalb *et al.*, 2009). In the work, the task of pose estimation is thought as a detection problem. Sub-models were trained for different poses of the same swimming style. The hypothesis is that every swimming style has periodicity, in the experiment, the methods split the swimming cycle into sections, the continuous images were treated in each period as the same pose, a sub-model was trained for it. Finally, these models were combined into a hybrid model, each hybrid model was specific to only one swimming style.

For example, breaststroke is split into four movements, the breaststroke hybrid model consists of four submodules and the butterfly swimming is only divided into two movements, the hybrid model of butterfly-stroke contains only two sub-models. The experimental results show that there was no obvious boundary between the actions, this method often made errors during the overrun, and more sophisticated algorithms are needed to reduce the occurrence of errors.

Before the emergence of the method based on deep learning, most studies were devoted to improving the graph structure, a slew of algorithms were committed to designing more effective features (Lowe, 2004), and others were committed to designing more flexible spatial models (Andriluka *et al.*, 2009; Yang & Ramanan, 2011;). However, the overall accuracy of pose estimation based on traditional methods is not high.

There are two main reasons for the low practical value of the traditional method. Firstly, the computational cost of low-level features such as specifically designed HOG and shift features is high, the features extracted by these methods are generally limited which cannot make full use of image information. Secondly, if the angle of pose changes a lot, the template model with a single shape cannot accurately match the changed pose, there may be multiple feasible solutions, the result of pose estimation is not unique. This makes the traditional method hard to be used in complex scenes (Josyula & Ostadabbas, 2021; Liu *et al.*, 2015).

2.3.2 Methods Based on Deep Learning

CNN has become a research hotspot in the field of machine vision since 2012, the earliest one for pattern classification problems, later for object detection and segmentation problems. DeepPose network (Toshev and Szegedy, 2014) was proposed firstly, which is the first time that CNN has been employed to solve the problem of human pose estimation. Since then, the related work on pose estimation of swimmers began to combine conventional methods with deep learning methods to improve the efficiency of the proposed models.

A method was proposed to estimate the pose of swimmers by using the DCNN representation of DPM (Zecha *et al.*, 2017). The conventional HOG feature was abundant, AlexNet (Krizhevsky *et al.*, 2012) was employed to extract depth features and completed the classification of components directly through the network, the relationships between the body parts are constrained by graphical model (Chen & Yuille, 2014). This work still retains the inertia of traditional methods, which achieved pose estimation based on the

graph structure method, as a result, this method still has the limitations brought by template matching.

In fact, DCNNs are able to obtain multi-scale and multiclass human node features under different receptive fields and contextual information of each feature, there is no need to use deformable part mode and constrain the relationship between body parts. Deep learning methods often integrate feature extraction, classification, and spatial location modeling into a network without independent disassembly.

According to the way of predicting key points, the pose estimation methods based on deep learning are split into 2-fold: The methods based on regression and the methods based on heatmaps. The regression methods output the key coordinates directly with an end-to-end structure (Carreira *et al*, 2016). The difficulty of this kind of methods is that the direct regression of coordinates from image sequence is a nonlinear problem, the full connection layer for regression coordinates would have low spatial generalization ability of models, which is able to easily lead to overfitting (Dang *et al.*, 2019). After that, a method of transition processing is proffered to generate a heatmap and estimate the position of key points (Wei *et al.*, 2016).

Heatmap is to represent each kind of coordinates with a probability graph and generate a probability for each pixel position in the image, which is utilized to represent the probability that the point belongs to the corresponding category of key points. The closer the distance from the key point position, the higher the probability of the pixel is tending to 1.00, and the distance from the key point position, the closer the probability of the pixel is 0, which is able to be simulated by Gaussian functions (Zhang *et al.*, 2020). Compared with regressing the coordinates of key points directly, the heatmap better preserves the spatial location information, which is helpful for network training. Through using the contextual information retained by heatmaps, CNN is able to establish the spatial structure between nodes implicitly, the deformable model based on prior knowledge is no longer needed (Chen *et al.*, 2020).

Heatmaps were firstly employed in convolutional pose machines (CPMs). The large receptive field is beneficial for the model to learn the spatial structure of various parts of the human body. The network of CPMs consists of four sequentially connected phases. In each stage, the classical VGG network was taken to extract belief maps. Deep supervision was added between each stage to ensure the quality of deep net training (Wei *et al.*, 2016).

CPMs were combined with temporal sequence models to solve the problem of pose estimation of swimmers (Einfalt *et al.*, 2018). In their work, the importance of spatial information and temporal information was emphasized on CPMs. Various styles of swimming pose information are integrated into the network together so as to boost the network better and obtain the contextual information between body parts. CNN is added to learn the temporal information and optimize the output of the CPMs network. The approach successfully improved the performance of the CPMs based on the task of pose estimation of swimmers.

In order to learn effective spatial and representational information, CPMs increase the receptive field through max pooling and multistep large convolution kernels. However, max pooling reduces the resolution of the image, resulting in the loss of details, and large convolution cores mean huge computational costs. Most of the high-performance networks proposed after CPMs follow the idea of increasing the receptive field, they generally obey the high-to-low and the low-to-high structures. High-to-low structures are utilized to increase the receptive field by reducing the resolution of the input which is known as downsampling. The low-to-high structure restores the low-resolution representation to the original resolution, which is known as upsampling. In order to improve network performance, the networks carry out multiple downsampling or upsampling. These processes are usually sequentially connected to form the entire network (Yang *et al.*, 2017).

For example, a symmetrical structure (Newell *et al.*, 2016) was proposed to concatenate high-to-low structures and low-to-high structures. The cascaded pyramid

networks and the simple baselines network take use of a structure that cascades heavyweight high-to-low structures with a lightweight low-to-high structure (Chen *et al.*, 2018; Sun *et al.*, 2019; Xiao *et al.*, 2018).

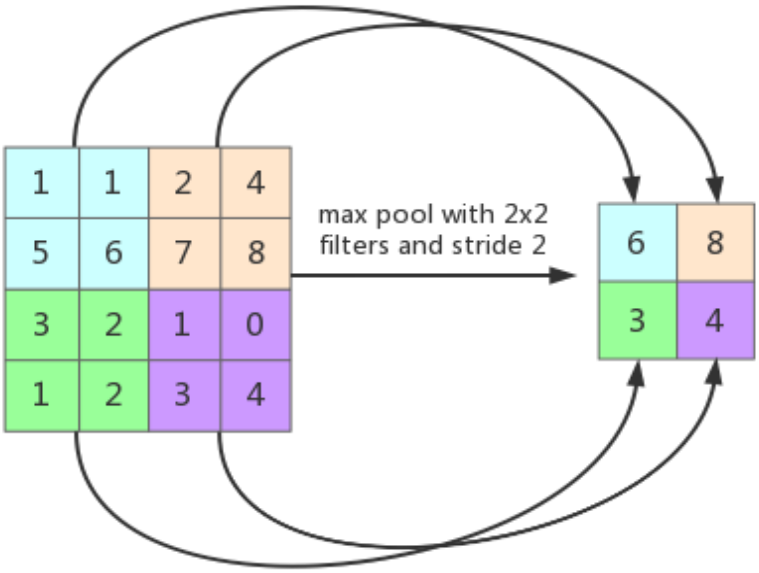


Figure 2.2: An example of max pooling

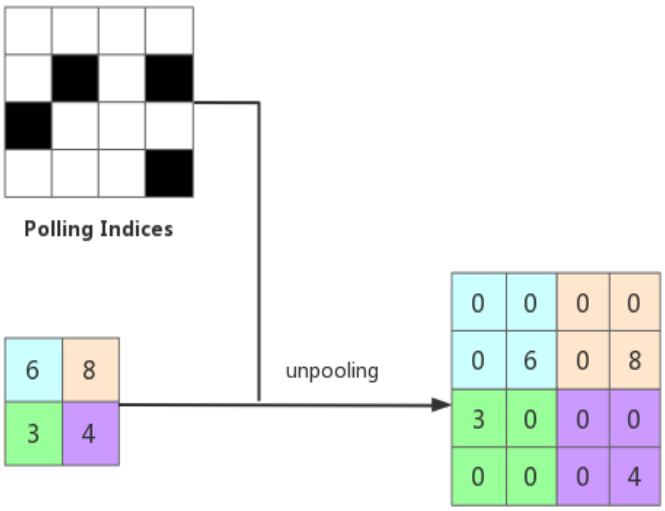


Figure 2.3: An example of max unpooling

The downsampling methods include max pooling and average pooling. Max pooling segments the input image into several rectangle areas and selects the maximum value of each region as output. The specific operation is shown in Figure 2.2, assuming that the

input is a 4 × 4 matrix, we utilize a 2 × 2 filter to pool it in the stride of 2, each element of the output matrix corresponds to the maximum value of 2 × 2 regions in the input matrix. Other values, other than the maximum value, are discarded in the process, which is information loss during the max pooling (Guo *et al.*, 2016). Upsampling is applied to restore the resolution of the feature map. There are three popularly harnessed upsampling methods:

(1) **Unpooling.** Unpooling is regarded as the inverse process of pooling. As shown in Figure 2.3, taking maximum unpooling as an example, it extends the location information of the maximum value retained after max pooling to feature map, and takes use of 0 to supplement the position other than the maximum value (Zeiler & Fergus, 2014).

(2) **Linear interpolation.** The essence of interpolation is to utilize known data to estimate unknown data. The nearest neighbor algorithm is the simplest and most broadly utilized algorithm, which has low computational complexity. In this method, among the four adjacent pixels, the value of the closest pixel is selected as the value of the pixel to be solved. This method may cause discontinuity in the grayscale of the image generated by interpolation, the obvious jagged shape may appear where the grayscale changes. However, because of its low computational complexity and fast generation speed, it is still adopted by many networks (Mazzini, 2018).

(3) **Transposed convolution.** Transpose convolution (deconvolution) is widely utilized in the fields of image segmentation. Deconvolution is able to be seen as a reverse convolution process (Long *et al.*, 2015). Similar to the convolution process, the deconvolution process requires parameter learning, which is to expand the size of the picture by training the transposed convolution check (Noh *et al.*, 2015). Compared with other simple filling methods, the data obtained by learning is more accurate. This is an ideal upsampling method at present. It has been utilized to recover the resolution of the feature map in some networks of pose estimation, for example, the DeeperCut network adds a deconvolution layer at the end for upsampling. However, transpose convolution significantly increases the computational complexity of the

model, so it is unrealistic to utilize transpose convolution for upsampling multiple times in the network (Dumoulin & Visin, 2016; Radford *et al.*, 2015).

In general, information loss is inevitable in the process of reducing the resolution. Most resolution enhancement methods expand the resolution by adding 0 or simply estimating a similar value. For example, in the stacked hourglass network, max pooling is accommodated to reduce the resolution of the feature map, and linear interpolation is offered to restore the resolution of the input (Newell *et al.*, 2016). A few methods are able to recover pixels that are relatively close to the original image by learning. However, the pixel loss caused by downsampling is not able to be completely reversed by upsampling. As a result, the performance of these networks is always constrained by the information loss caused by downsampling.

Location and detail information is contained in the low-level features that have low semantics and more noise. The receptive field of high-level network is relatively large, the high-level features have stronger semantic information, but the resolution is very low and the ability to perceive details is poor, the fusion of these two features improves the performance of the model. In the stacked hourglass network, the skip path is added to preserve the high-resolution representation before downsampling the feature image and fuse the feature with the corresponding feature image in the way of concatenation and add respectively before and after each upsampling. Similarly, the features are fused with the same resolution in the process of upsampling and downsampling by means of channel splicing in U-Net (Ronneberger *et al.*, 2015). However, these networks only have an additional unidirectional feature fused from high-resolution to low-resolution or from low-resolution to high-resolution.

In deep nets, intermediate supervision is added between stages to optimize the training process. For instance, in CPM, the loss is calculated at the end of each phase to optimize the training results (Wei *et al.*, 2016). Each hourglass network adds loss for supervision, the output heatmaps set is applied to calculate the error with the true value (Newell *et al.*, 2016). 3D U-Net adds intermediate supervision in the process of two

upsampling. Intermediate supervision has been proved to be beneficial to network training in many studies (Çiçek *et al.*, 2016).

2.3.3 Dataset

Nowadays, most researchers obtained the pose information of swimmers from side-mounted cameras (Greif & Lienhart, 2010). For instance, the data collected by the camera that fixed on the other side of the transparent wall of a special swimming pool. The special swimming pool needs to be equipped with a reverse flow device to keep the swimmer in a relatively fixed position during swimming (Zecha et al., 2012). This kind of data is usually accompanied by a large number of surficial bubbles and refracted noises, researchers need to use a variety of methods to recover the swimmer's occluded or distorted limbs, which usually means more complex calculations and longer running time. In addition, such a complex experimental scene makes the application of the model which is limited to swimming lanes with similar configurations and cannot be used for the events that occur in ordinary swimming lanes. On the other hand, the side view means that only one swimmer is included in each image, so data collected from the side view cannot be used in the study on multi-swimmer pose estimation.

2.4 Multi-Person Pose Estimation

The deep learning-based methods achieved ideal performance on the task of single-person pose estimation, as a result, the method based on deep learning has been taken into account to implement multi-person pose estimation. Compared with the task of single-person pose estimation, the task of multi-person pose estimation is much difficult, its input always contains multiple human objects, in addition to the correct detection of the location of keypoints, but also need to correctly determine which target the keypoints belongs to. At present, there are generally two kinds of approaches for multi-person pose estimation, which are top-down and bottom-up approaches. However, the existing research methods of multi-person pose estimation based on deep learning do not cover

the scenario of multi-swimmer. In the following, we review the two kinds of methods for multi-person pose estimation.

2.4.1 Bottom-Up Method

The bottom-up method was divided into two steps. The first step is to find all the key points from the input image, such as all heads, left hands, knees, etc., this step is usually directly realized by using a single person pose estimation model as a key point detector. The second step is to assign these key points to each target. In general, this kind of methods firstly detect the keypoints and then clusters the key points (Zheng *et al.*, 2020).

OpenPose is a classic multi-person pose estimation model which takes use of CPM (Wei *et al.*, 2016) as the component to find the position of each joint from the input, and then proposes Part Affinity Fields (PAF) to assemble the human body (Cao *et al.*, 2019). The basic principle of PAF is to establish a directional field between two adjacent key points which represents the degree of correlation between the two key points (Cao *et al.*, 2017).

Another important work is associated embedding (Newell *et al.*, 2017). Similar to the idea of OpenPose, the hourglass is firstly utilized to detect all key points in the input. The idea of associative embedding was proposed in the assembly of joint points. This method outputs an embedding for each key point. The embedding of the same person is as close as possible, and the embedding of different people is as different as possible.

In addition to OpenPose and Associative Embedding, DeepCut (Pishchulin *et al.*, 2016) and DeeperCut (Insafutdinov *et al.*, 2016) are also excellent. DeepCut takes advantage of Fast R-CNN as the body part detector to detect all the body parts, then marks each part as its corresponding part category, and utilizes integer linear programming to combine these parts to form a complete skeleton. DeeperCut is an optimization scheme, which adopts a more powerful residual net to extract body parts, and makes use of image

conditioned pairwise terms to compress the number of nodes in the candidate area, so as to improve the detection speed.

In general, the bottom-up method firstly predicts all keypoints of each person in the input image and then groups them through manikin fitting or other algorithms. The advantage of this type of approach is that the computations do not increase substantially as the number of people in the image increases. However, bottom-up approaches face challenges in correctly grouping corresponding body parts if the overlap occurs between objects (Li *et al.*, 2018).

2.4.2 Top-down Method

The top-down method is also divided into two steps, the first step is to find all human objects in the given image, which often needs to be achieved by using the object detection method. The second step is to estimate the pose of each human object separately, the single-person pose estimation is often directly adopted (Zheng *et al.*, 2020).

In terms of detection performance, compared with the bottom-up method, the top-down method is usually much robust for two reasons: First of all, the recall obtained by the top-down method is higher because this kind of method detects the human body, and the human body is often larger than the part, so it is easier to be detected accurately.

Secondly, the human body detection frame is helpful for spatial alignment, and the single-person estimation method learns more information about the relationship between the whole and the part, which is very helpful for point positioning (Kocabas *et al.*, 2018). In terms of detection speed, top-down methods separate the process of pose estimation into two parts, it is often assumed that the detection speed of top-down methods is lower than that of bottom-up methods (Sun *et al.*, 2020). However, with the continuous improvement of the detection speed of the human body detectors, the speed of the top-down methods is also greatly improved, Therefore, in recent years, more work has taken the top-down method into account (Papandreou *et al.*, 2017).

The advantage of the bottom-up method is that object detection speed is seldomly influenced by the increase of targets in the scene. This advantage is particularly obvious if facing the crowd. However, our task is to realize the pose estimation of swimmers. The number of swimmers in the lane is limited, the advantage of the bottom-up method is not obvious to us. On the other hand, the top-down method, which achieves a better balance in accuracy and speed, is more suitable for our task. Therefore, we implement multi-swimmer pose estimation based on the top-down method. The methods for human pose estimation mentioned in this thesis are shown in Table 2.1.

Table 2.1: An review of deep learning techniques used for pose estimation

Methods		References
Single-Person Pose Estimation	DeepPose	Toshev and Szegedy, 2014
	Convolutional pose machines	Wei et al., 2016
	Stacked hourglass	Newell et al., 2016
	Cascaded pyramid networks	Chen et al., 2018
	Simple baselines	Xiao et al., 2018
Multi-Person Pose Estimation	OpenPose	Cao et al., 2019
	Associated embedding	Newell et al., 2017
	DeepCut	Pishchulin et al., 2016
	DeeperCut	Insafutdinov et al., 2016

2.4.3 Object Detection

In the top-down method, the performance of the detector is critical, the following is an overview of the object detection methods based on deep learning which are able to realize the human detector. The task of object detection is to find out interesting objects in the given data to predict their position and size, in the meantime, which is one of the core problems in the domain of machine vision. After deep learning was introduced into object detection, the accuracy and speed of this task have been raised to a new level, object detection methods have been developed at an unprecedented speed. The object detection methods based on deep learning are grouped into two categories, one is two-stage method, the other is the one-stage way (Zhao *et al.*, 2019).

Table 2.2: The methods for object detection in R-CNN, Fast R-CNN, and Faster R-CNN.

Nets	Region proposal	Feature extraction	Classification	Rect refine
R-CNN	selective search	DeepNet	SVM	regression
Fast R-CNN	selective search	DeepNet		
Faster R-CNN	DeepNet			

The two-stage object detection methods split the detection process into two stages: Generating region proposals, classifying the position of the candidate regions. Region proposals mean filtering out regions with a high probability of being an object (Hosang *et al.*, 2015). There are several exemplify methods for the realization of proposals, such as Selective Search, EdgeBoxes, and so on (Zitnick & Dollár, 2014). CNN based on region proposals is canonical representative of the two-stage object detection method. For instance, Region CNN (R-CNN) was proposed in 2015 (Girshick *et al.*, 2014) which is a milestone in the use of deep learning for object detection and has a long-standing impact on this field. The selective search was employed to generate candidate regions before detection, so as to reduce the degree of information redundancy and improve the detection speed (Uijlings *et al.*, 2013). Then, CNN was applied to the candidate regions for the feature extraction, finally, take SVM for the classification, regression model for the refining of the bounding box.

Unlike R-CNN, SPP-Net (He *et al.*, 2015) convolutes and generates candidate regions, which not only reduces the storage capacity, but also speeds up the training speed. In addition, it makes use of spatial pyramid pooling to solve the problem of different size images. However, the detection speed of R-CNN and SPP-Net is still slow. Fast R-CNN improved R-CNN by introducing multitask learning and integrating multiple steps into a model. Classification and regression were probed in a CNN network, which greatly reduced the complexity of training and make training more convenient.

In addition, Fast R-CNN takes use of the smooth L1 function to prevent the gradient from becoming too large for some points that deviate greatly and has better robustness. The use of truncated SVD also greatly ameliorates the detection speed of Fast R-CNN (Girshick, 2015). However, Fast R-CNN still needs a dedicated candidate window

generation module. In order to address this problem, Faster R-CNN utilizes the Region Proposal Network (RPN) instead of the selective search module and merges the RPN into Fast R-CNN to generate an end-to-end detection network (Ren *et al.*, 2015). There are boosts in efficiency all the way from R-CNN, to Fast R-CNN, and to Faster R-CNN. As shown in Table 2.2, for Faster R-CNN, the significant distinction between R-CNN and Fast R-CNN is that the four steps, required for object detection, are all implemented based on deep neural networks that are calculated based on GPUs, which greatly improves the working efficiency of the model. However, Faster R-CNN only utilizes top-level features for prediction.

Feature Pyramid Network (FPN) selects features at all scales that have rich semantic information. The prediction based on feature layer of the scale makes the generated proposals better than the Faster R-CNN algorithm that only performs prediction on the top layer (Lin *et al.*, 2017).

In the subsequent development, the position-sensitive score map (Dai *et al.*, 2016) was proposed to address the positional sensitivity of detection. Mask R-CNN (He *et al.*, 2017) was proposed as a method of object detection and segmenting multitask collaborative learning, which introduces the ROI align to replace ROI pooling so as to get better positioning. However, the operation speed of the two-stage target detection algorithm is always unsatisfactory.

In order to make the object detection meet the real-time requirements, the one-stage objects detection method was proposed based on regression analysis. This kind of methods abandon the candidate box and treat the object detection as a regression analysis problem of object location and category information. The detection results are output directly through the primary feedforward network, therefore, the detection speed has been greatly improved (Zou *et al.*, 2019).

The framework of one-stage object detection was firstly proposed in YOLO (i.e., You Only Look Once) whose core idea is to take use of the whole image as the input of the network and directly return the location and category of objects in the output layer.

The basic network of YOLO is Darknet customized based on the architecture of GoogLeNet, where 1×1 convolution layer and 3×3 convolution layer are utilized instead of inception module. The network of YOLO consists of 24 convolution layers and 2 full connection layers. In the process of detection, the input image is separated into $N \times N$ grids, each grid cell is in charge of detecting the objects whose center is in this grid. The output of each grid is information of a slew of bounding boxes and a group of probabilities representing that the object belongs to a certain category (Redmon *et al.*, 2016).

The information of the bounding box includes the center coordinates of the bounding box relative to the grid boundary, the width and height related to the picture, and the confidence score of the bounding box. Confidence scores represent the likelihood of the network to contain an object in the predicted binding box, which is calculated by using IOU of the predicted box versus the true binding box. If there is no object in the binding box, the confidence score is 0. If the grid cell contains an object, no matter how many bounding boxes it contains, the grid cell only predicts one group of probability. By multiplying the probability of each class with the confidence score of each bounding box, we obtain confidence score of each bounding box for each class. The bounding box with the highest confidence is taken as the output of visual object detection, which also means that each grid predicts only one object at most (Redmon *et al.*, 2016; Zaidi *et al.*, 2021).

Because the input of YOLO is the whole image, it predicts the bounding box of all classes of visual objects in the image at the same time. Compared with the method based on sliding window or RPN that obtains more context information, which increases the accuracy of detection. However, the limitations of YOLO are also obvious. Compared with the two-stage method, the detection result of YOLO for small objects, especially dense small objects, is very poor, mainly because YOLO divides the image into grids for detection, the grid may include multiple different kinds of small targets, but each grid of YOLO only predicts a limited number of boxes belonging to the same class (Redmon *et al.*, 2016).

Single shot multibox detector (SSD) (Liu *et al.,* 2016) inherits the idea of rapid detection in YOLO to integrate all detection processes into a convolutional network with the addition of multi-scale feature mapping. The PRN in Faster R-CNN generates nine classes of anchors on the top-level feature map to predict objects of different shapes and sizes. On this basis, SSD improves the processing method of multi-size objects and utilizes feature pyramids instead of top-level feature maps for prediction. The feature pyramid of SSD refers to the utilize of six scales of convolutional layers for prediction, and different convolutional layers obtain features of different sizes, which are exploited for the detection of objects with different specifications. Specifically, SSD generates default boxes of different sizes based on feature maps of each scale for prediction. The closer the top-level feature map is, the larger the size of the default box is.

Taken SSD-300 as an example, the backbone of SSD-300 is adapted from VGG-16, which converts the two fully connected layers of VGG-16 into ordinary convolution layers and adds some new convolution layers. SSD-300 extracts six feature maps of different scales for prediction, generates a series of concentric default boxes, centers on the midpoint of each point on the feature map, and then generates several preliminary qualified default boxes, through detection and classification. Finally, overlapped or incorrect default boxes are eliminated through non-maximum suppression (NMS) to generate the final detection results. Different from the full connection layer adopted by using YOLO models, SSD directly applies two convolutions to extract the detection results from different feature maps, one for outputting category confidence and the other for outputting the location of the bounding box (Liu *et al.,* 2016).

SSD runs at a speed that is comparable to that of YOLO, the detection accuracy almost caught up with the Faster R-CNN. However, the SSD suffers from relatively obvious drawbacks. For example, the form of the prior box in a network is not able to obtain through learning and needs to be pre-determined. The parameter of the prior box applied by each layer of feature in the network is disparate, which makes the debug processes highly empirically dependent. Moreover, SSD adopts the idea of the hierarchy of pyramid functions, but the recall of small targets is still not ideal. This is due to SSD

using low-level features to detect small targets, while low-level features are extracted by a few convolutional layers and suffer from insufficient feature extraction (Liu *et al.*, 2016).

YOLO was improved and YOLOv2 was proposed (Redmon & Farhadi, 2017), the focus is on solving the deficiencies of YOLO in terms of the performance on recall and accuracy. The basic network of YOLOv2 is Darknet-19, which includes 19 convolutional layers and five pooling layers. The network removes all dropout layers and adds batch normalization to each convolution layer, which makes the convergence speed of the network faster. The network also deletes the full connection layer, takes use of the anchor box to predict the position of the candidate box, and makes use of the K-means clustering method to cluster and calculate a better anchor template in the training set, which prominently enhances the recall rate of the algorithm. Because the full connection layer is removed, the resolution of the input image is no longer limited.

In addition, YOLOv2 connects shallow features with deep features to improve the detection ability of the model for small objects. In terms of training, they utilize the resolution 448×448 of images as input which is higher than that of YOLO, to pretrain the classifier and use images with different resolutions to multi-scale training, so that the model is able to adapt to image input with different sizes and enhance the prediction robustness of the model to multi-scale images (Thuan, 2021).

YOLOv3 (Redmon & Farhadi, 2018) was put forward based on the improvement of YOLOv2. In the feature extraction part, the Darknet-53 network structure was employed instead of the original Darknet-19, which is a classification network with better comprehensive performance than ResNet-152. Similar to the ideas of the feature map pyramid by using SSD, YOLOv3 also takes advantage of feature maps of diverse scales for the detection of objects. The large receptive fields obtained from small-size feature maps are used to detect large-size objects, and the small receptive fields obtained from large-size feature maps are applied to detect small-size objects.

In terms of classification method, the logistic loss is used instead of softmax loss to make the model suitable for the multiple label classification task, which guaranteed the accuracy of target detection while accounting for the real-time nature. YOLOv3 takes use of a larger-scale model to further improve the detection accuracy, and the detection speed is slightly slower than YOLOv2, but compared with other detection algorithms, such as RetinaNet, SSD, DSSD, the comprehensive performance of YOLOv3 is still the best (Fu *et al.*, 2017). In addition, YOLOv3 also provides a lightweight tiny-darknet, with a faster detection speed at the expense of reduced accuracy (Redmon & Farhadi, 2018; Lin *et al.*, 2017).

Darknet-53 is mainly composed of 1×1 and 3×3 convolution layers. Each convolution layer is followed by a batch normalization layer and a Leaky ReLU, to prevent overfitting. Leaky ReLU is a variant of ReLU. ReLU sets all negative values to zero, while Leaky ReLU gives all negative values a non-zero slope, which solves the problem (Maas *et al.*, 2013). The group of convolution layer, batch normalization layer, and Leaky ReLU is called DBL. There are 53 DBLs in Darknet-53 (Redmon & Farhadi, 2018).

Unlike Darknet-19, Darknet-53 does not use max pooling layer for downsampling, which is able to reduce the information loss caused by downsampling to some extent. The input of the network is added with the original input after two DBLs, which is a kind of conventional residual units. The purpose of introducing residual units into the network is to extract features and avoid gradient vanishing or exploding. Darknet-53 concatenates the output of the middle layer of the network and the upper sampling results of a later layer to achieve the purpose of multi-scale feature fusion. YOLOv3 designed three network outputs of different scales to predict objects at different scales (Redmon & Farhadi, 2018).

YOLOv3 was refined by using a large number of tricks that contributed to improving detection accuracy and the improved model was named YOLOv4 (Bochkovskiy *et al.*, 2020). The method substantially increases the detection accuracy of YOLOv3 under

conditions that guarantee detection speed. A general framework was proposed for the object detection task consisting of backbone, neck, head. The following content introduces these three structures in YOLOv4：

- **Backbone.** YOLOv4 takes use of CSPDarknet-53 implemented by the combination of Darknet53 and cross stage partial network (CSPNet) as the backbone. Darknet-53 is the backbone of YOLOv3, which is composed of a series of residual structures. In Darknet-53, each convolution layer was followed by a batch normalization layer and a Leaky ReLU, while in CSPDarknet-53, mish function replaces the Leaky ReLU as the activation function, as a result, the basic convolution unit is called CBM in CSPMarknet-53. Mish allows positive values to reach any height, which avoids saturation caused by positive capping. It gives a negative gradient instead of directly using a zero boundary (Misra, 2019). Mish improves the accuracy of CSPMarknet-53 in the classification task.

  The CSPNet was proposed to solve the problem that network structures require extensive inferential calculation, which was caused by repeating gradient information. Therefore, the main purpose of CSPNet is to achieve a richer gradient combination and reduce the amount of calculation, which is achieved by dividing the feature map of the basic layer into two parts, and then merging them through a cross phase hierarchy (Bochkovskiy *et al.*, 2020; Wang *et al.*, 2020).

  The existence of pooling layers makes the convolution layer insensitive to the dropout operation, Dropblock operation was employed for regularization in YOLOv4. Dropblock refers to the random discarding of local areas, which has a similar effect with using cutout for data augmentation. The difference between the two operations is that Dropblock applies the cutout operation to every feature map in the network. In addition, the probability of Dropblock is able to be modified at different stages of training, Dropblock is a more delicate operation than the cutout operation used in image enhancement (Ghiasi *et al.*, 2018).

- **Neck.** At the end of the backbone, an SPP module designed according to the SPP-net (He *et al.*, 2015) is added to increase the receptive field of the network. In the feature fusion, YOLOv4 takes use of the modified path aggregation network (PANet) for

parametric aggregation from different backbone layers for different detector levels ((Bochkovskiy *et al.*, 2020; Liu *et al.*, 2018).

- **Head.** YOLOv4 inherits the head of YOLOv3 for multi-scale prediction, improves the loss function during training, and takes use of the Diou NMS method to filter the prediction box instead of using NMS. Diou NMS takes into account not only the overlapping regions of the prediction box and the ground truth box, but also the distance between the center points of the two boxes (Zheng *et al.*, 2020).

After YOLOv4 was proposed, the first version of YOLOv5 was released, which has four scales: YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x (Jocher *et al.*, 2020). YOLOv5s network has the smallest scale, fastest detection speed, and lowest detection accuracy. The other three networks deepen and widen the network on the basis of YOLOv5s, to improve the accuracy, but the time of detection consumption continued to increase (Liu *et al.*, 2021).

YOLOv5 possesses a much higher detection speed while the detection ability is close to that of YOLOv4 in most detection tasks (Li *et al.*, 2021; Yang *et al.*, 2020), the size of the model is much smaller than that of YOLOv4, which makes YOLOv5 well suited to be deployed on mobile devices for real-time detection. Our task is to implement the detection of underwater swimmers, which requires a high speed of model detection and convenience of deployment. As a result, we take use of YOLOv5 to procure the swimmer detector (Chen *et al.*, 2021; Fang *et al.*, 2021).

Chapter 3
Methodology

The main content of this chapter is to clearly articulate research methods, which satisfy the objectives of this thesis. The details of the research methodology for our proposed multi-swimmer pose estimation model are interpreted in this chapter. This chapter mainly has three parts. The first part introduces HRNet for single-swimmer pose estimation, the second part is related to YOLOv5 for swimmer detection, the third part includes the YOLOv5 interface.

3.1 Research Methods

Our proposed multi-swimmer pose estimation model is interpreted in this chapter, which consists of two parts, the first part is the swimmer detector which is implemented by YOLOv5. The second one is the single-swimmer pose estimation model which is implemented by using HRNet. The two parts are connected through the detection interface to form an end-to-end model.

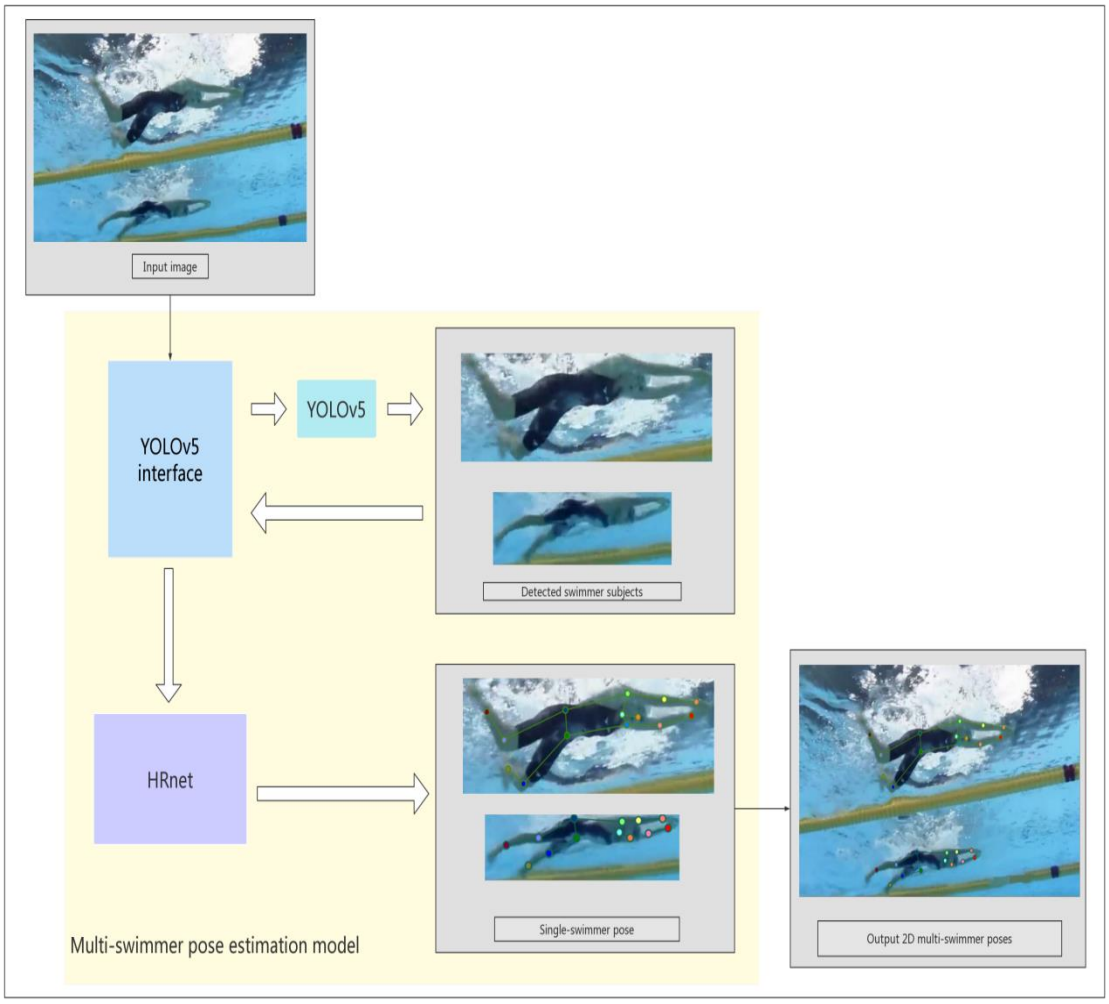


Figure 3.1: An overview of our multi-swimmer pose estimation model.

The structure of our model is described as Figure 3.1. Firstly, a set of bounding boxes are gotten from the input images by the swimmer detector, then these bounding boxes are feed into the single-swimmer pose estimation model to predict the pose of swimmer in

each person box. The output of our multi-swimmer pose estimation model is the collection of the pose information for all the swimmer in each image.

3.2 HRNet for Single-swimmer Pose Estimation

In this thesis, we implement a single-swimmer pose estimation method based on the heatmaps, High-Resolution Network (HRNet) is taken as the core network. For each kind of key point, we will generate a corresponding heatmap by the feature maps output by HRNet. The heatmap includes the possibility of each pixel in the graph to be this kind of key points. We will calculate the coordinates of the point according to the probability diagram.

In HRNet, the branches of the network with different resolutions are connected in parallel, which makes the network maintain a high-resolution representation throughout the whole process. The information is exchanged between branches through cross fusion to obtain a more reliable high-resolution representation (Sun *et al.*, 2019).

In the past, most networks only encapsulated unidirectional feature fusion from high-resolution to low-resolution or from low-resolution to high-resolution, while each fusion layer of HRNet includes a bi-directional fusion between high-resolution and low-resolution, which aids the network to continuously optimize the high-resolution representation. The network does not adopt supervised learning, which reduced the computational complexity to some extent, and the accuracy of the network was not affected. Compared with the network using a series structure, HRNet has advantages in retaining high-resolution features and making full use of all levels of features (Sun *et al.*, 2019).

3.2.1 The Structure of HRNet

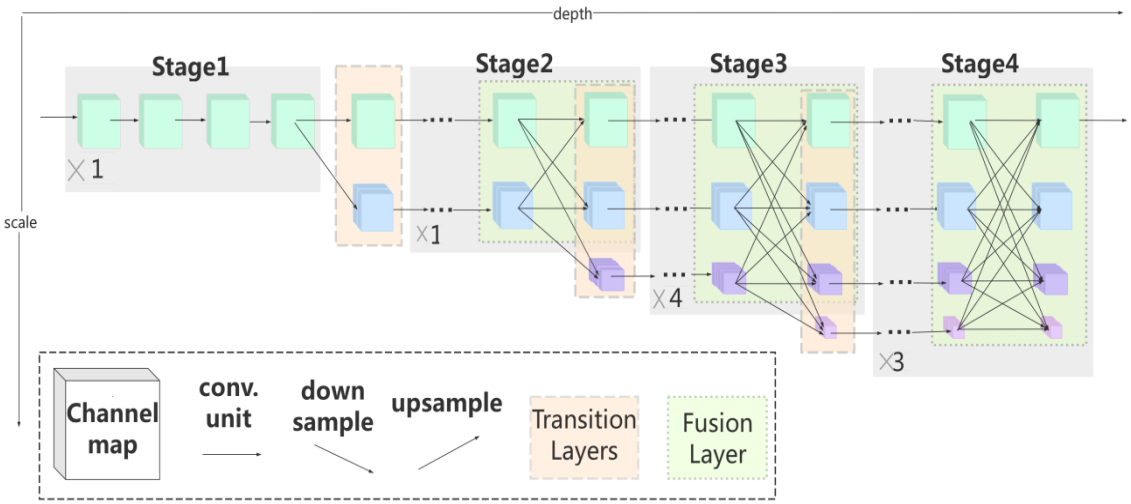


Figure 3.2: An overview of the main structure of HRNet. Some duplicated channel maps are represented as ellipsis.

The structure of the HRNet is shown in Figure 3.2. Vertically, the backbone of HRNet consists of four parallel branch networks. The resolution of the parallel network is halved one by one from top to bottom. The branches exchange information through feature fusion. The network is divided into four stages horizontally. The first stage includes a high-resolution sub-network, and a low-resolution network will be added in each subsequent stage. Except for the fourth phase, the transition layer is added to the end of each phase of the network to generate the new subnetwork(Wang *et al.*, 2020).

We take Figure 3.3 as an example of the network structure, W_n is applied to represent the subnet, n is the stage of the network.

$$
\begin{aligned}
W_1 \rightarrow \quad & W_2 \rightarrow W_3 \rightarrow W_4 \\
\searrow \quad & W_2 \rightarrow W_3 \rightarrow W_4 \\
\searrow & W_3 \rightarrow W_4 \\
\searrow & W_4
\end{aligned}
$$

Figure 3.3: The proposed network structure

Figure 3.4 shows the change in the resolution of the network, where RE_n is employed to represent the resolution of the network, and n represents the stage to which the network belongs:

$RE_1 \rightarrow RE_2 \rightarrow RE_3 \rightarrow RE_4$
$\searrow$ ½$RE_2 \rightarrow$ ½$RE_3 \rightarrow$ ½RE_4
$\searrow$¼$RE_3 \rightarrow$ ¼RE_4
$\searrow$ ⅛RE_4

Figure 3.4: The changes of resolution in HRNet

Fusion layer and transition layer. The input for the fusion layer is a collection of feature maps of stage *n* which has *s* subnetwork $\{F_1, F_2, \cdots, F_s\}$. Input maps are aggregated to produce each output map. The collection of output maps is represented as $\{O_1, O_2, ..., O_s\}$, Eq. (3.1) shows the specific calculation process, where *f* refers to the resolution of the output feature image *Y*. If the resolution of the input feature map *X* is less than *f*, function *A* represents the upsampling calculation. if the resolution of the input feature map *F* is higher than *f*, function A represents the downsampling calculation. If the resolution of the input feature map *X* is equal to *f*, function A represents direct replication. The resolutions of the input feature map of one subnet is as same as the resolutions of the output feature map (Wang *et al*., 2020).

$$O_f = \sum_{i=1}^{S} A(F_i, f) \tag{3.1}$$

The input of the transition layer connected after the fusion layer is the output O_s of the subnetwork with the lowest resolution at this stage. A new channel O_{s+1} is generated by downsampling, Eq. (3.2) shows the specific calculation process, where f_s refers to the resolution of O_s.

$$O_{S+1} = A(O_S, \frac{1}{2} f_s) \tag{3.2}$$

3.2.2 Instantiation

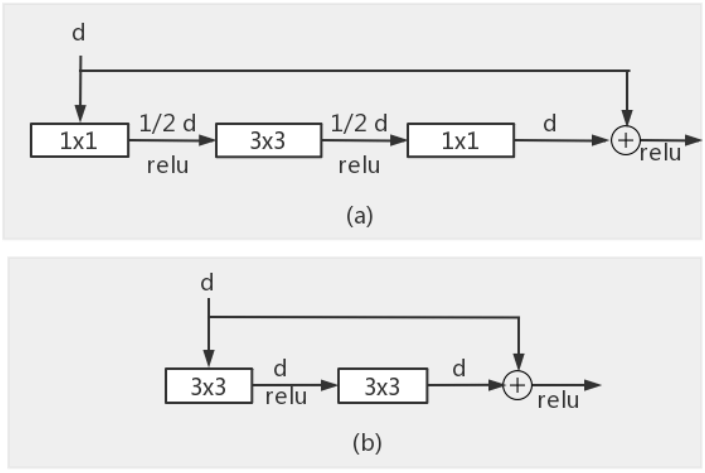


Figure 3.5: ResNet block, where *d* means the number of channels. (a) Bottleneck block (b) Basic block.

The first stage contains only one branch network, which mainly encapsulates four residual units, as shown in Figure 3.5(a). The structure of the residual unit is as same as that of the bottleneck in ResNet-50 (He *et al.*, 2016), its width is 64, including a 1×1 convolution for dimensionality reduction, a 3×3 convolution for feature extraction, a 1 × 1 convolution for dimensionality reduction. The first two convolutions are followed by batch normalization and nonlinear activation of ReLU, the last 1×1 convolution without using ReLU, in order to maintain the diversity of features. The structure of the jump connection includes two possibilities: If the number of input feature channels is as same as the output, it is added directly, otherwise, a 1×1 convolution is added to match the dimension of the output feature, which usually occurs after the resolution is reduced. The structure of this connection is to avoid the gradient dispersion and degradation problems caused by the increase of network layers. The first phase contains only one branch, so the fusion module is not included in the phase.

The second, third, and fourth stages are similar in structure, they are all composed of repetitive modules with similar structures. The second stage includes one module, which is composed of two parallel subnetworks. The third stage contains four repetitive modules, and each module consists of three parallel subnetworks. The fourth stage

contains three repetitive modules, and the module is composed of four parallel subnetworks. The subnetworks in the modules that make up each stage that utilities the same structure, that is, four residual network units.

The structure of each residual network unit is as same as that of the basic block in ResNet-50 (He *et al.*, 2016), as is shown in Figure 3.5(b), which includes two 3×3 convolutions, the first of which is followed by batch normalization and nonlinear activation ReLU. The second convolution is followed by only one batch normalization, and the second, third and fourth stages all include multiple network branches with different resolutions, the end of each module in the stage is added fusion layer, to fuse the features at different resolutions. In total, eight times of multi-scale fusions were carried out in the second, third, and fourth stages (Wang *et al.*, 2020).

Taking the fusion layer of the third stage as an example, as shown in Figure 3.6, it spans all parallel networks in this stage and merges the characteristic graphs of all sub-networks. The output of each sub-network after passing through the fusion layer is the aggregation of all sub-network inputs in that stage. Before aggregation, the characteristics of the same network branch are copied directly. The low-resolution feature map improves the resolution by using nearest neighbor upsampling and a 1×1 convolution to obtain the same number of channels as the high resolution. An appropriate number of stride 3×3 convolution with stride 2 is used to reduce the resolution and increase the number of channels of high-resolution feature maps to match that of the low-resolution subnetwork. We do not use max pooling for downsampling to avoid the loss of information during dimensionality reduction. Downsampling through strided 3×3 convolutions is able to reduce the loss of information (Springenberg *et al.* 2014; Wang *et al.*, 2020).

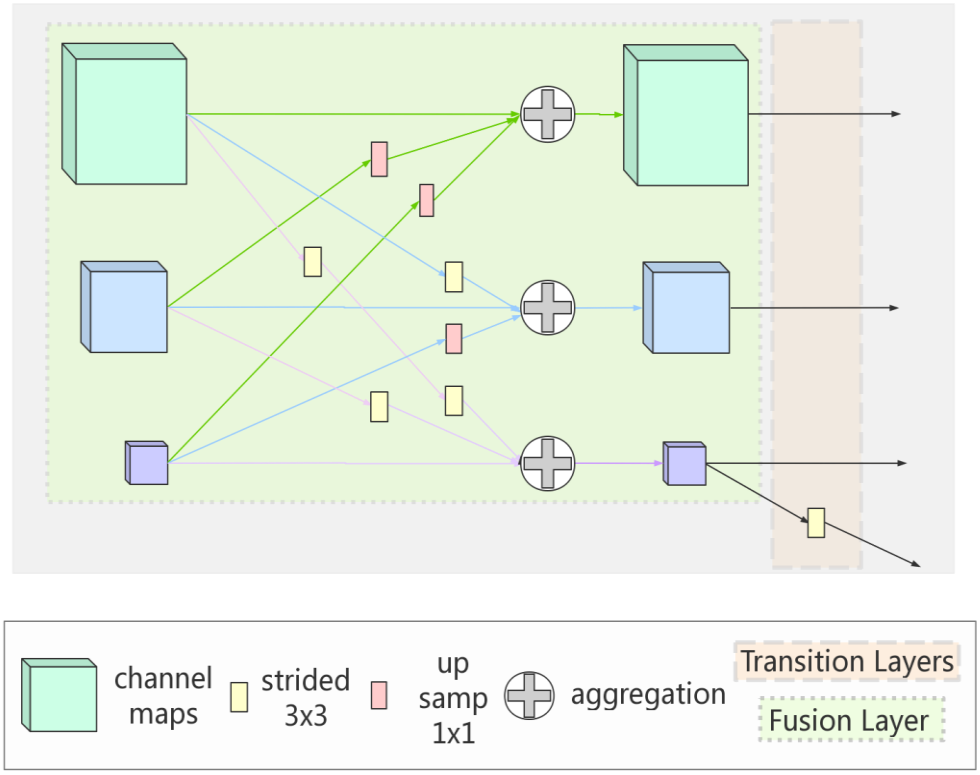


Figure 3.6: The structure of fusion layer and transition layer, where strided 3×3 means a strided 3×3 convolution, up samp. 1×1 include nearest neighbor up-sampling and a 1 × 1 convolution.

After the last fusion layer in each stage, we add a transition layer to generate a new subnetwork with more channels. As shown in Figure 3.6, in the transition layer, strided 3×3 convolutions are applied to adjust the channels and resolution of the network, the resolution of the new subnetwork is halved.

In our experiment, we implement a small network named HRNet-W32 and a large network named HRNet-W48. The main difference between this two networks is the width of the sub-network. The widths of the four parallel subnetworks of HRNet-W32 are 32, 64, 128, and 256 respectively. The widths of the four parallel subnetworks of HRNet-W48 are 48, 96, 192, and 384, respectively (Sun *et al.*, 2019).

3.2.3 Heatmap

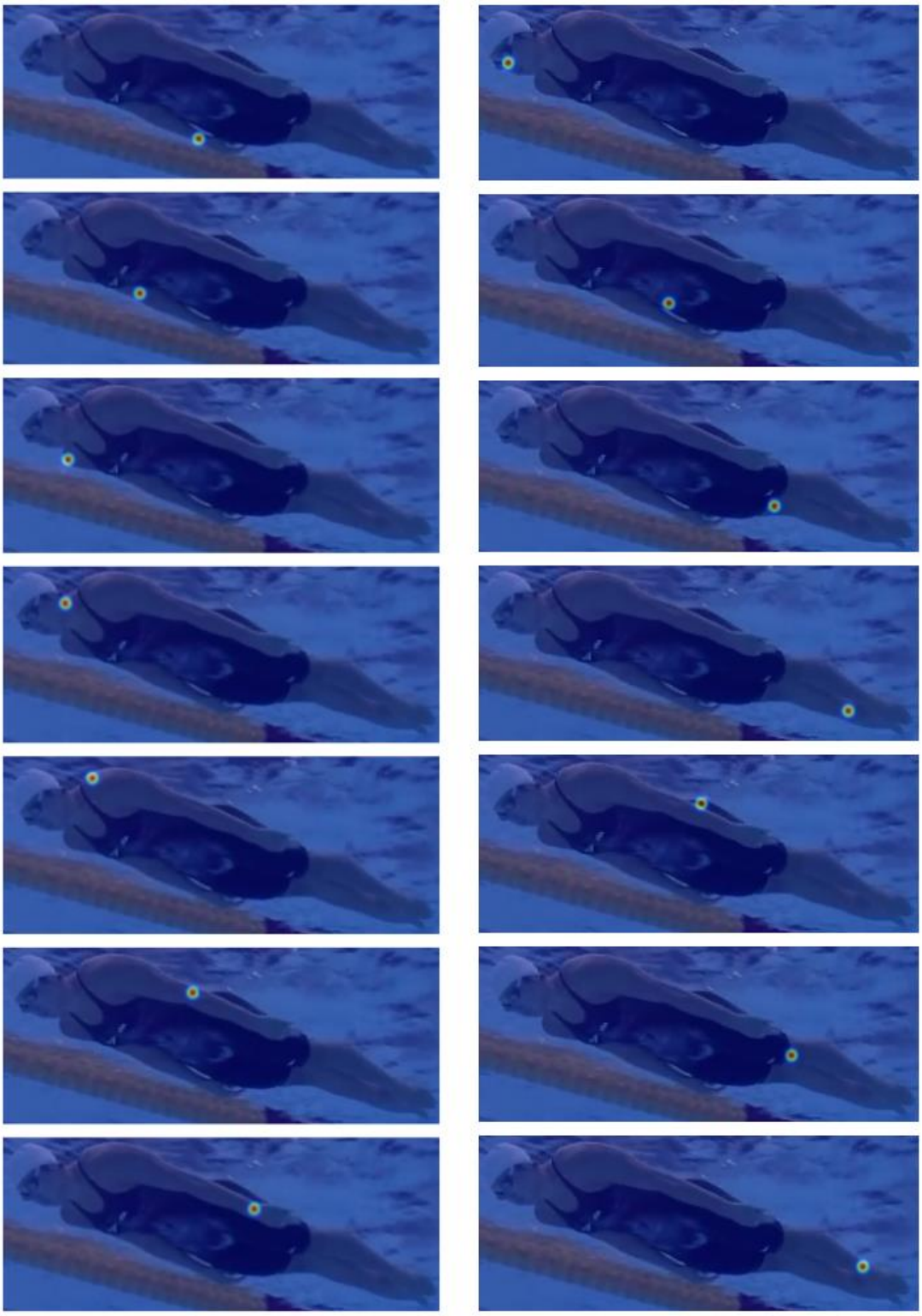


Figure 3.7: Visualization of the heatmap of the 14 key points

The fourth stage consists of four network branches and outputs four feature maps in different resolutions. We abandon the three lower resolution outputs and only take use of the representation of the highest resolution sub-network output to return to the heat map. According to the experience, this hardly affects the performance of the network but reduces the computational complexity. The groundtruth heatmap is generated by using 2D Gaussian distribution, centered on the real position of each key point, and the standard deviation is 2 pixels (Wang *et al.*, 2020). Figure 3.7 displays the visualization of the heatmaps.

3.2.4 Loss Function

We take the following three different losses for experiments:

- **Mean Square Error.** We define the loss function as Mean Square Error (MSE) which was used to compare the predicted heatmaps with the groundtruth heatmaps, Eq. (3.3) shows how the MSE is calculated, where m is the number of samples, T_i is the ground truth value, O_i is the predicted value.

$$MSE = \frac{1}{2m}\sum_{i=1}^{m}(T_i - O_i)^2 \quad (3.3)$$

- **MSE with Online Hard Keypoints Mining.** Online hard keywords mining (OHKM) was proposed by (Chen *et al.*, 2018), which sorted the results of MSE output and screened the top eight key points with the largest loss as difficult cases for key regression.
- **MSE and Bone loss.** Bone loss (Kulon *et al.*, 2020) was applied to compare the distance between predicted key points and the distance between groundtruth points, so as to ensure the model learns the spatial features between key points, such as the length of the bone. In the experiment, we take use of Bone loss and MSE for joint training, in the following, we call it MSEBone loss.

3.2.5 Dataset

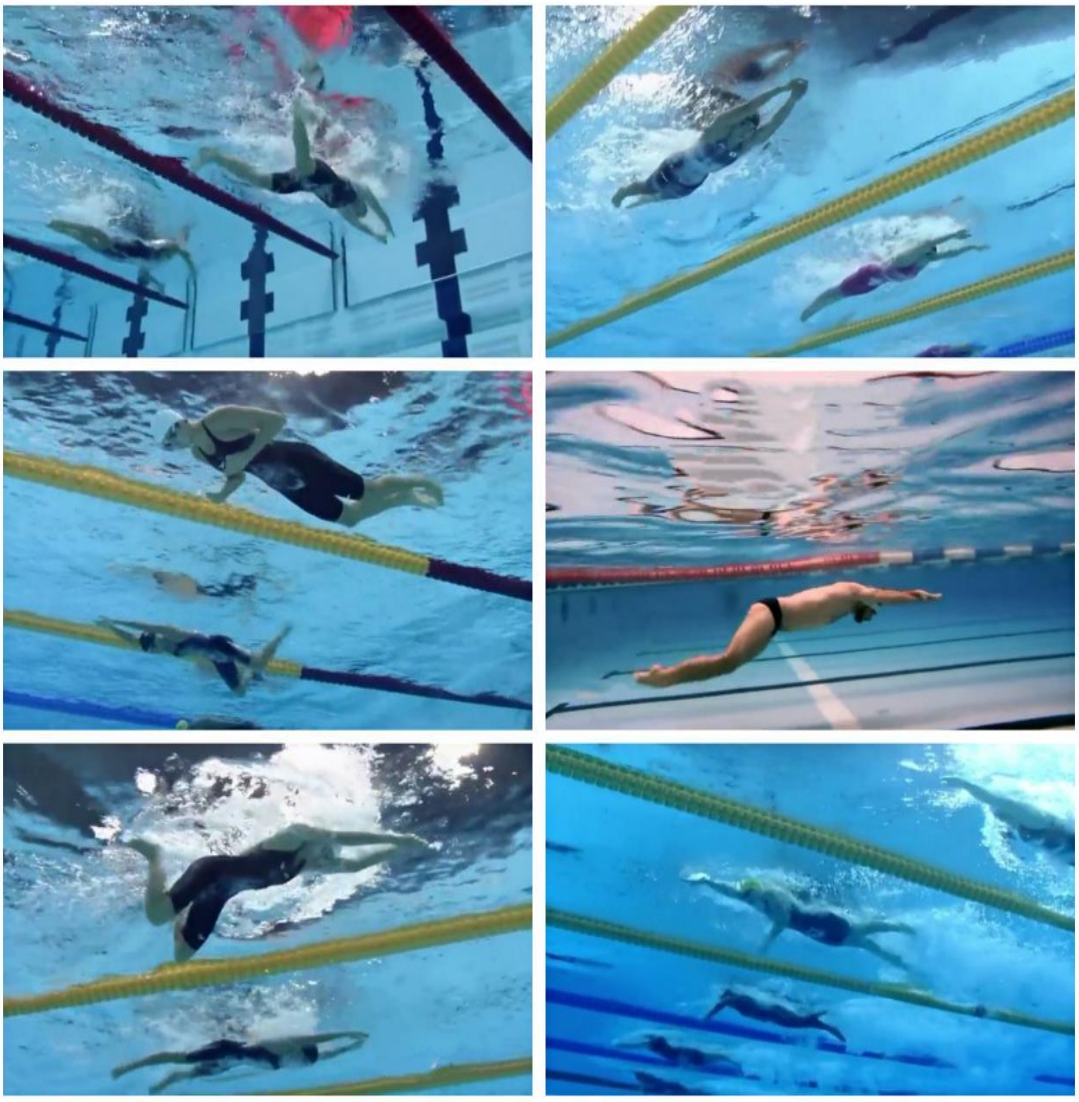


Figure 3.8: Sample images of swimmer datasets

As shown in Figure 3.8, our dataset was collected by using video footage of swimming events, which records the athletes from an underwater view. Compared with the angle of side viewing, we observe a more complete swimmer's body. The dataset includes 2,500 images with 3,615 annotated swimmers. Figure 3.9 shows the statistics on the number of key points contained in each sample in the swimmer key point dataset, which indicates the key points of the samples in our dataset are relatively complete. The dataset includes human bodies of both genders, including breaststroke, freestyle, butterfly, and backstroke. Almost all angles of underwater viewing are covered, in order to make the

model better adaptive to noises, we added 20% noisy images, including water reflection, bubbles, dim light, occlusion, and other factors. In addition, we also add empty swimming lanes as negatives. In the experiment, the dataset is divided into training_val set and test set in the proportion of 9:1, then the training_val set is divided into training set and validation set in the proportion of 9:1.

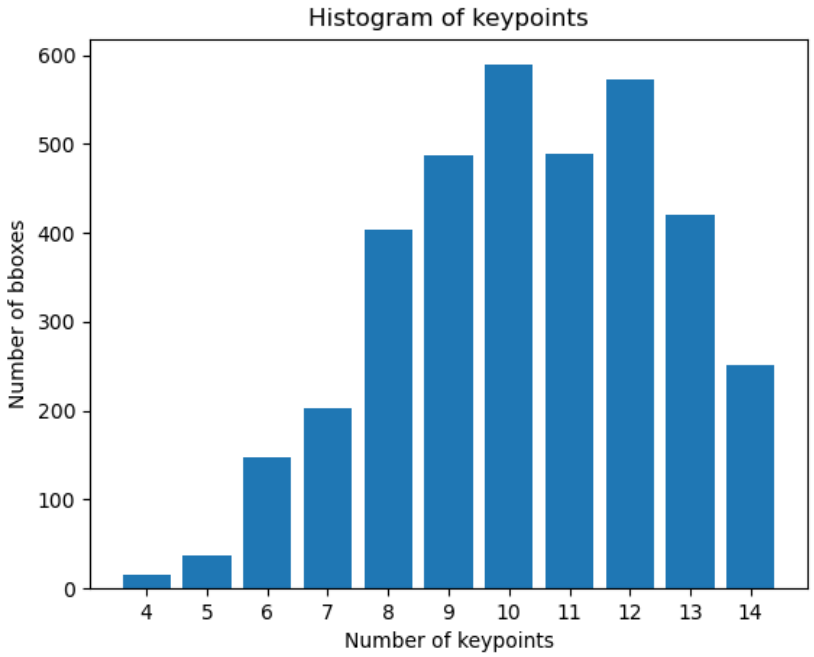


Figure 3.9: Statistics on the number of key points contained in each sample in the swimmer key point dataset. The vertical axis represents the number of bounding boxes, each bounding box contains a swimmer, the horizontal axis shows the number of key points contained in each detection box.

In the pose estimation for swimmers, the importance of facial key points are lower than that of the key points on the body, so we reduce the number of facial key points of the skeleton compared with that of the COCO dataset (Lin *et al.*, 2014). As shown in Figure 3.10, our human key point model includes 14 key points: Nose, neck, left elbow, right elbow, left shoulder, right shoulder, left hip, right hip, left knee, right knee, left ankle, right ankle, left wrist, right wrist.

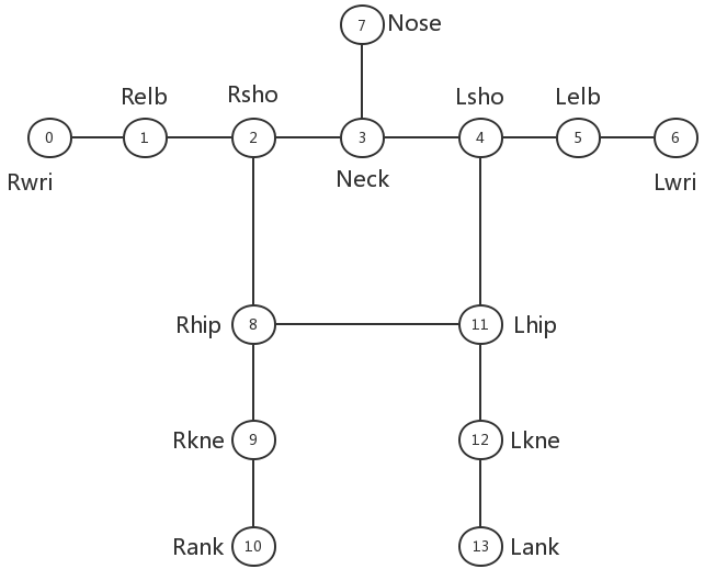


Figure 3.10: A complete example of the key points of the swimmer annotated in our dataset

3.2.6 Evaluation Metrics

We mark the location of each key point in the form of (x, y, v) as ground truth, where (x, y) is the coordinate of the key point, v is the visibility symbol. If the value of v is 0 which means that the node is not annotated. If v is 1.00 which means that the node has been annotated but cannot be seen because of occlusion and other factors. If v is 2 means that it has been annotated and is able to be seen.

The key point detector outputs the position of the key for each object. The format of predicted key points is as same as the ground truth. At present, the visibility index has not been employed for evaluation and calculation, the key point detector does not need to predict the visibility of key points. The following indicators are used to evaluate the performance of the model in the experiment:

(1) **Object Key Point Similarity (OKS).** We define OKS by using Eq. (3.4). Our main purpose is to calculate the similarity between the true value and the key points predicted by the model.

$$OKS = \sum_i [\exp(-d_i^2/2s^2k_i^2)\delta(v_i > 0)] \,/\, \sum_i [\delta(v_i > 0)] \tag{3.4}$$

In Eq. (3.3), the Euclidean distance between the true point and the detected key point is expressed. The scale of the target is represented by *s*, which shows the square root of the area occupied by the target in ground truth that is calculated by the coordinates of the bounding box: $\sqrt{(y_2-y_1)(x_2-x_1)}$, k_i denotes the normalization factor of the *i* skeletal point. The higher the value of *k*, the worse the annotation effect of this point in the whole dataset, which means that the annotation of this dimension point is more difficult. On the contrary, it means that this point is easier to be marked. v_i is the visibility symbol, the detector does not predict v_i, so only v_i marked in the dataset is used here. Combined with the function δ, we filter the marked key points by using the value of *v*.

In the calculation, *k* is regarded as a constant, the corresponding *k* of different key points is different. For example, the number *k* of human torso key points is often much larger than human facial key points. These constants are obtained by calculating the standard deviation σ of the information labeled by different people in the COCO dataset which contains 5,000 images. COCO dataset contains 17 categories of human skeleton key points, the corresponding σ of each key point is shown in Table 3.1 (Lin *et al.*, 2014). Our dataset only contains 14 key points, so we make adjustments to the sigma array. The adjusted sigma array and the corresponding key points are shown in Table 3.2.

Similar to IOU, we take use of OKS to calculate AP and AR. We give a threshold *K*. If the OKS of the target is greater than the threshold, it means that the skeletal point of the current target has been correctly detected. If the OKS of the target is less than the threshold, it means that there is a phenomenon of false detection or omission of the skeletal point of the current target. *AP* is equal to the proportion of OKS greater than *T* to all OKS.

In the experiment, we calculate the following indicators to evaluate the performance of the model: $AP^{0.5}$ and $AR^{0.5}$ if the threshold K is 0.5. $AP^{0.75}$and $AR^{0.75}$ when the threshold *K* is 0.75, the average value of *AP* and *AR* at ten positions if the threshold *K* is 0.50, 0.55, ⋯, 0.95. The average *AR* at ten positions when *k* is 0.50, ...,

0.95. AP^M and AR^M for medium objects, AP^L and AR^L for large objects (Lin *et al.*, 2014).

Table 3.1: The sigma of each key point in COCO dataset

Dataset annotation order	Joint name	Sigma
0	nose	0.026
1	Leye	0.025
2	Reye	0.025
3	Lear	0.035
4	Rear	0.035
5	Lshoulder	0.079
6	Rshoulder	0.079
7	Lelbow	0.072
8	Relbow	0.072
9	Lwrist	0.062
10	Rwrist	0.062
11	Lhip	0.107
12	Rhip	0.107
13	Lknee	0.087
14	Rknee	0.087
15	Lankle	0.089
16	Rankle	0.089

Table 3.2: The sigma of each key point in our dataset

Dataset annotation order	Joint name	Sigma
0	Rwri	0.062
1	Relb	0.072
2	Rsho	0.079
3	Neck	0.035
4	Lsho	0.079
5	Lelb	0.072
6	Lwri	0.062
7	Nose	0.026
8	Rhip	0.107
9	Rkne	0.087
10	Rank	0.089
11	Lhip	0.107
12	Lkne	0.087
13	Lank	0.089

(2) **Percentage of Correct Keypoints**. We take use of the percentage of correct key points (PCK) to evaluate the detection ability of the model for each type of joint. We define PCK by using Eq. (3.5), where *TP* means the number of correctly predicted points. If the distance between the detected key point and the true point is less than the threshold, the prediction point is considered to be correct, the distance is normalized by the scale of the person (Xiao *et al.*, 2018), *T* means the number of true key points. In the experiment, we set the threshold to 0.50.

$$PCK = \frac{TP}{T} \tag{3.5}$$

3.2.7 Experiment Setting

In the experiment, we set the aspect ratio of the detection box of the swimmer to 4:3 and then cut the detected swimmer out of the image, we adjust its size to a fixed resolution of 384 × 288 before importing it into the network.

We increase the amount of data in the dataset through data augmentation, including three ways: Randomly rotate the image between -35 and 35, randomly scale the image to 0.65 to 1.35 times, and flip the image horizontally randomly.

Before training, we convert the key points marked in the dataset into heatmaps for subsequent calculation. We take use of Gaussian to calculate the heat maps, we set σ to 2. Therefore, in order to speed up the calculation speed, we only carry out Gaussian distribution on the region of 3σ. The size of the Gaussian kernel is 6, the size of the resulting Gaussian distribution is 13.

In the training process, the learning rate is set to 1.00×10^{-3}, the decay gamma is set to 0.10, the decay steps are set to 170 to 200 epochs, the batch size is set to 12. We train the model on NVIDIA GeForce RTX2080 super GPU. The operating system is Ubuntu20.04. The virtual environment is Python 3.8. The training ended after the 200-th

iterations, with a total time of around 12 hours for the big network and around 8 hours for the small network.

In the testing part, cropped swimmer boxes are sent into the trained pose estimation model for keypoints detection. The average heatmaps of the original image and the flipped image were taken as the output heatmap. Following the previous works (Chen *et al.*, 2018; Law & Deng, 2018; Xiao *et al.*, 2018), we generate the location of each keypoint by moving the highest response for the quarter offset in the direction from the highest response to the second high response.

3.3 YOLOv5 for Swimmer Detection

YOLOv5 has four scales of networks, the network structures of these models are similar. YOLOv5s is the smallest and most basic model among the four networks, the other three models deepen and widen the network on the basis of YOLOv5s. In this part, we take YOLOv5s as an example to introduce the network structure of YOLOv5.

3.3.1 The Structure of the YOLOv5

The network structure of YOLOv5 is mainly composed of the backbone, neck, and head. Backbone refers to the convolutional network used for feature extraction on the original image. Neck refers to the structure used to achieve feature fusion. Head refers to the structure used to classify and position the objects.

The backbone of YOLOv5 mainly is composed of the focus structure, the CBS unit, the C3 module, and the spatial pyramid pooling (SPP) structure. The focus structure integrates the information of the feature map in the width and height dimension into the channel dimension through the slicing operation. The purpose of this structure is to reduce the amount of calculation and raise the inference speed (Jocher et al., 2020). CBS unit as the basic convolution unit is employed for feature extraction or downsampling.

The C3 module extracts rich information features from the input image. The structure of the C3 module is modified from the CSPNet, which is purpose to solve the problem of gradient information repetition in deep CNN and integrate the gradient changes into the feature maps (Wang et al., 2020). In the C3 module, the stacked residual blocks are divided from a single pipeline into two pipelines. In the trunk pipeline, the original residual blocks continue to be stacked. The branch pipeline is applied to skip the stacked residual blocks and merges the feature maps of the beginning with the end of the trunk after through a CBS unit. C3 module enhances the learning ability of CNN, maintains the accuracy while lightening the network, and reduces the computational bottleneck and memory cost. There are two kinds of C3 modules in YOLOv5, one for the backbone and another for the neck.

SPP module in YOLOv5 is inherited from that of YOLOv4. The main function of the SPP module is to increase the receptive field of the network, through several max pooling layers with different scales (Song *et al.*, 2021). The SPP module extracts important contextual features and hardly reduces the inference speed of the network (Bochkovskiy *et al.*, 2020).

In the backbone, in order to expand the receptive field of the network for the extraction of high-level features, the resolution of the feature map needs to be reduced multiple times by downsampling which leads to the loss of low-level features. However, both the higher-level and the low-level feature are helpful for the detection of different scales of objects (Liu *et al*., 2018). As a result, to minimize the loss of information in the process of network deepening, in the neck of YOLOv5, PANet is applied to implement feature fusion between the feature maps in different stages for the detection of different scales of objects.

PANet of YOLOv5 is implemented by combining the C3 module with the modified PANet proposed in YOLOv4. The modified PANet mainly consists of an FPN structure and a path augmentation structure. Compared with the original PANet proposed by Liu *et al.* (2018), the addition operation for the shortcut connection is replaced by the

concatenation operation (Bochkovskiy *et al.*, 2020). The feature fusion ability of YOLOv5 is further strengthened by changing some of the basic convolution units in the modified PANet to the C3 module.

YOLOv5 takes use of the same head structure with YOLOv3 which outputs three feature maps at three different scales. During the processing of prediction, anchor boxes are applied to multi-scale feature maps to detect objects of different sizes, the finer grid cells are applied to detect the finer objects. Finally, the output is vectors with class probabilities, objectness scores, and bounding boxes.

3.3.2 Instantiation

In YOLOv5 structure, CBS unit is the basic unit of the networks, which includes a convolution layer followed by a batch normalization layer and a sigmoid weighted linear unit (SiLU) (Elfwing *et al.*, 2018). According to the kinds of convolution layer that included in the CBS unit, we classified the CBS unit into three types:

(1) The CBS, which includes the 3×3 convolution, is called 3×3 CBS. The function of the 3×3 convolution is feature extraction.

(2) The CBS which encompasses the 1×1 convolution is called 1×1 CBS, where the 1×1 convolution is applied for reducing the dimensions of the channel for the output.

(3) The CBS, which encapsulates 3×3 convolution of 2 strides, is called strided 3×3 CBS, where the 2-strided 3×3 convolution is applied for downsampling.

CBS is also a component of other modules, such as focus structure, C3 module, and so on, the backbone, neck, and head of the network, are stacked by these above structures.

The backbone of the network includes a focus structure and three C3_1 modules each of which is followed by a strided 3×3 CBS unit, the end of the backbone is an SPP module. The focus structure mainly consists of 4 parallel slice layers, a concatenation layer, and a 3×3 CBS unit. The process of slicing operation and concatenation operation shows the

input representation with the scale of $W \times H \times C$ is sliced into four representations with a scale of $W/2 \times H/2 \times C$, then the four representations are connected through the concatenation layer which outputs a representation with the scale of $W/2 \times H/2 \times 4C$ (Zhou *et al.*, 2021).

SPP module includes a 1×1 CBS unit follows by using four parallel operations, three of them are max pooling layers with kernel sizes of 5×5, 9×9, 13×13, respectively, and the other is a skip connection which connects the input and output of the three max pooling layers, the four parallel followed by a concatenation layer, the output of the concatenation layer finally goes through a 1×1 CBS unit.

C3 module has two kinds of structures: the C3_1 module which is used in the backbone, and the C3_2 module which is offered in the neck. The C3_1 module is split into two branches. The first branch includes a 1×1 CBS unit, and the second branch encompasses a 1×1 CBS unit followed by N bottleneck structures. These two branches are finally connected into one branch by a concatenation layer followed by a 1×1 CBS unit. The bottleneck includes a 1×1 CBS unit followed by a 3×3 CBS unit. Finally, the output of the 3×3 unit is added with the input of the 1×1 CBS unit through a skip connection. The C3_2 module has a similar structure as the C3_1 module, where the N bottleneck structures are changed to N convolution blocks. The convolution block is composed of a 1×1 CBS unit followed by a 3×3 CBS unit.

In the neck of the network, the output of the SPP module firstly go through the FPN structure which is formed by the stacking of two structurally identical FPN modules, The FPN module is composed of a C3_2 module followed by a 1×1 CBS unit, an upsampling layer, and a concatenation layer. In the sampling layer, we upsample the output of the 1×1 CBS unit through the nearest neighbor algorithm. Similarly, the path augmentation structure is also formed by the stacking of two PAN modules. The PAN module includes a C3_2 module followed by a strided 3×3 CBS units whose output will be concatenated with the output of the CBS unit with the same resolution in the FPN structure. The second PAN module is followed by a C3_2 module as the end of the PANet.

The head of the neural network is composed of three 1×1 convolution layers, the input of each convolution layer is the output of the final three C3_2 modules in PANet.

3.3.3 Loss function

In YOLOv5, different loss functions are employed for the classification of positive samples, the classification of foreground and background, and the regression of the bounding box (Kasper-Eulaers *et al.*, 2021). Our task is a single classification problem, so the classification of positive samples is undesired, only the following two loss functions are used in our experiment:

(1) CIoU loss. We take use of CIoU loss for the regression of bounding boxes. CIoU loss is based on DIoU loss defined as Eq. (3.6). In Eq. (3.9), besides the coverage area and center point distance, CIoU loss adds the aspect ratio factor calculated by Eq. (3.8), where v is the consistency of aspect ratio calculated by Eq. (3.7), α is a positive trade-off parameter (Zheng *et al.*, 2020).

$$L_{DIoU} = 1 - IoU + \frac{\rho^2\left(b, b^{gt}\right)}{c^2} \tag{3.6}$$

$$\upsilon = \frac{4}{\pi^2}\left(\arctan\frac{\omega^{gt}}{h^{gt}} - \arctan\frac{\omega}{h}\right)^2 \tag{3.7}$$

$$\alpha\upsilon = \frac{\upsilon^2}{\left(1 - IoU\right) + \upsilon} \tag{3.8}$$

$$L_{CIoU} = 1 - IoU + \frac{\rho^2\left(b, b^{gt}\right)}{c^2} + \alpha\upsilon \tag{3.9}$$

In our experiment, Eq. (3.9) is calculated in the following ways with the same meaning:

$$CIoU = IoU - \frac{D_{center}^{\ 2}}{D_{diagonal}^{\ 2}} - \alpha\upsilon \tag{3.10}$$

$$L_{CIoU} = 1 - CIoU \tag{3.11}$$

In Eq. (3.10), D_{center} is the distance between the center of ground truth and the center of the prediction box. $D_{diagonal}$ is diagonal distance of the minimum closure region containing both the prediction box and ground truth box.

We take use of *BCEWithLogitsLoss* for the classification of background and upground. *BCEWithLogitsLoss* is composed of *BCELoss* defined as Eq. (3.12) and *sigmoid* defined as Eq. (3.13), *BCEWithLogitsLoss* is defined as Eq. (3.14), where l_n is the loss corresponding to the n-th sample.

$$\ell(x, y) = L = \{l_1, ..., l_N\}^T, l_n = -\omega_n \left[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n) \right] \tag{3.12}$$

$$Sigmoid(x) = \sigma(x) = \frac{1}{1 + \exp(-x)} \tag{3.13}$$

$$\ell(x, y) = L = \{l_1, ..., l_N\}^T, l_n = -\omega_n \left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right] \tag{3.14}$$

3.3.4 Dataset

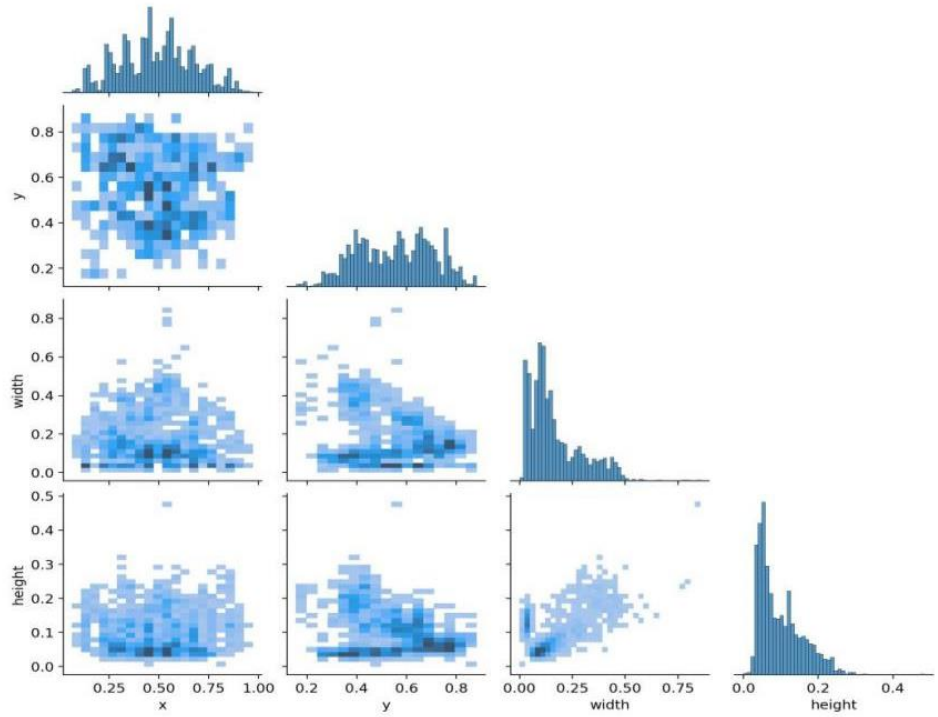
Figure 3.11: The example of images in our datasets.

Our dataset comes from 20 videos related to swimming competitions with a total duration of 2 hours. We randomly select video frames from these videos to make datasets, as shown in Figure 3.11, the video frames we selected include swimmers taken from various angles, such as surface view, underwater view, far distant view, and close view.

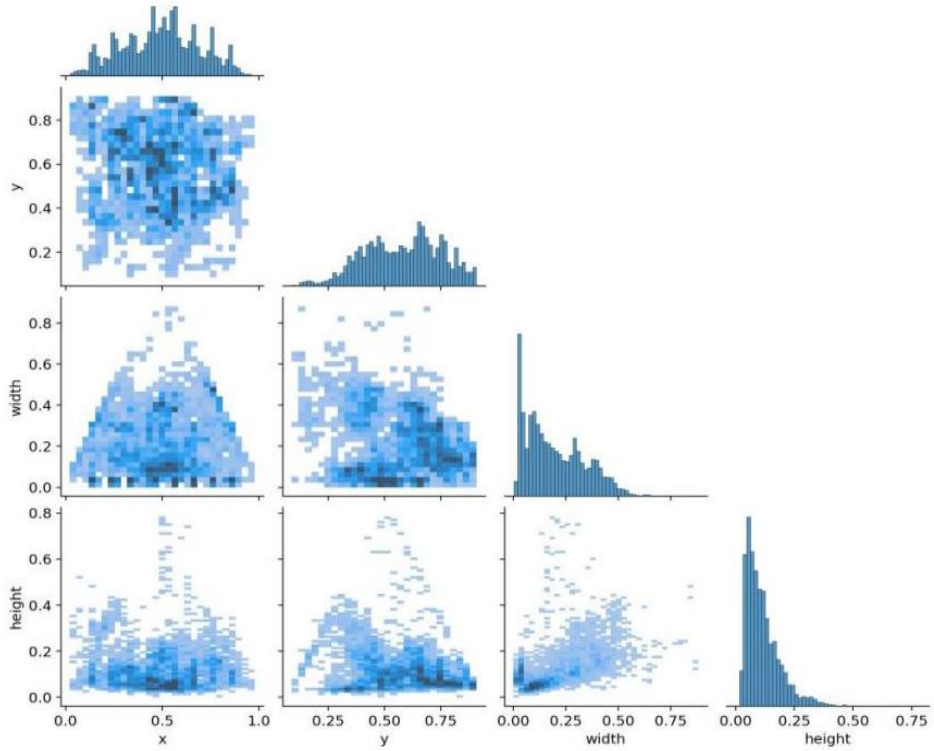
In order to find the most appropriate amount of data, we totally create three datasets with different scales: Swimmer-421, Swimmer-1755, Swimmer-3700. Swimmer-421 is the smallest dataset which contains 421 images. On the basis of Swimmer-421, we add 1,334 images to increase the total amount of images to 1,755 and named the new dataset as Swimmer-175. Similarly, Swimmer-3700 is made based on Swimmer-1755, by adding 1,945 images. As shown in Figure 3.12. the data sources of the three datasets are the same, so their data structures are similar.

We annotate all three datasets, and each dataset only labeled swimmers. Figure 3.13 shows the number, width, height, and center coordinates of all bounding boxes in the three datasets. In the experiment, each of the three datasets is divided into training_val

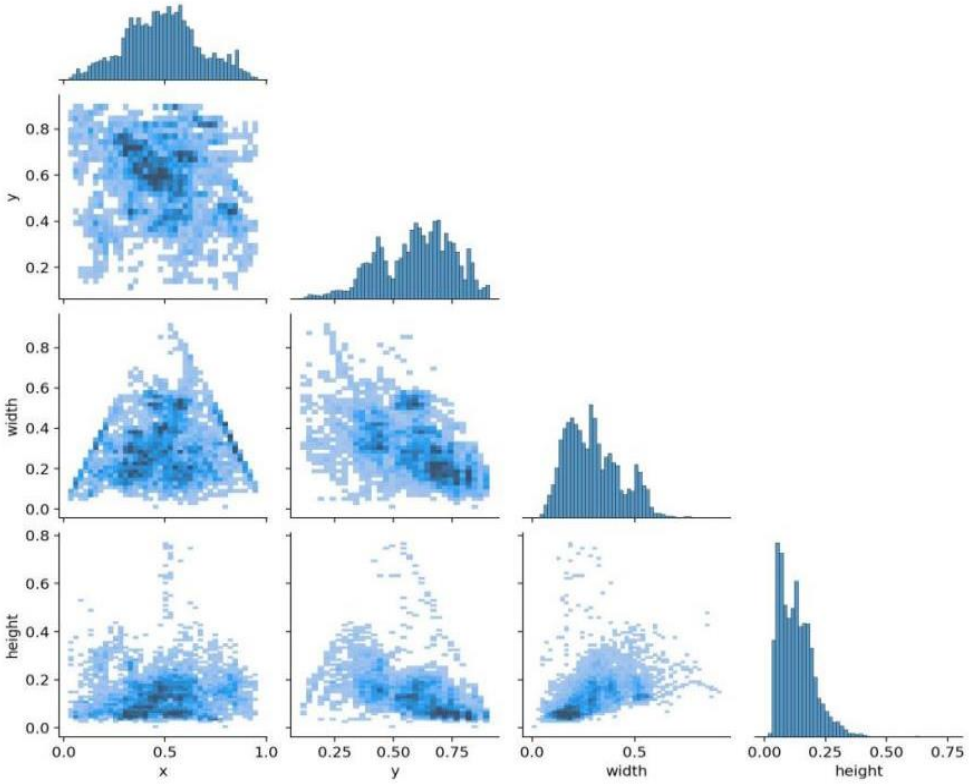
set, test set in the ratio of 9:1, then the training_val set is divided into training set and validation set in the ratio of 9:1.



(a)

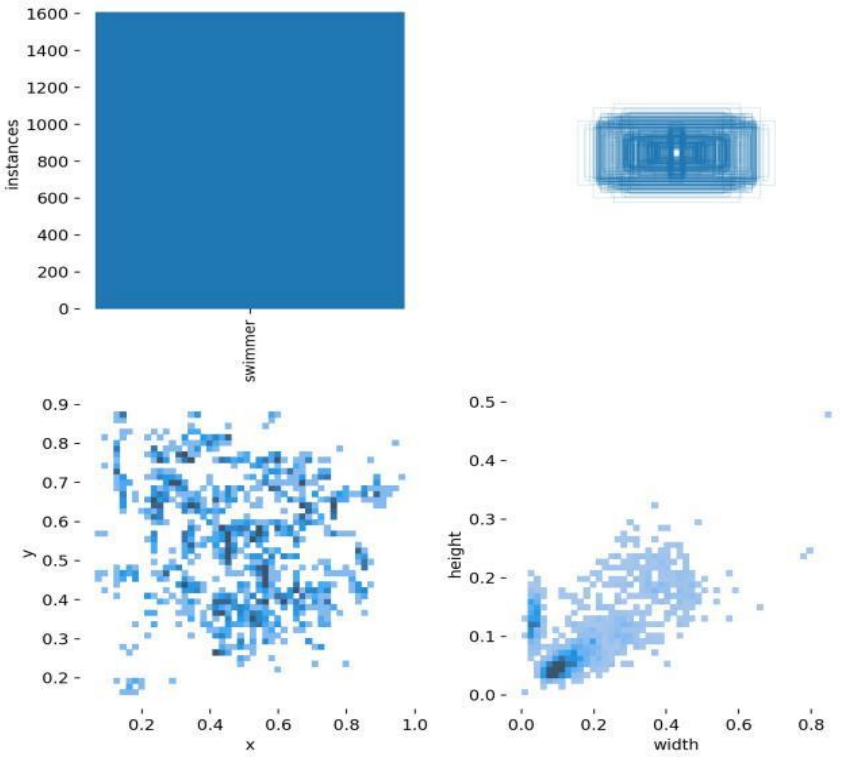


(b)

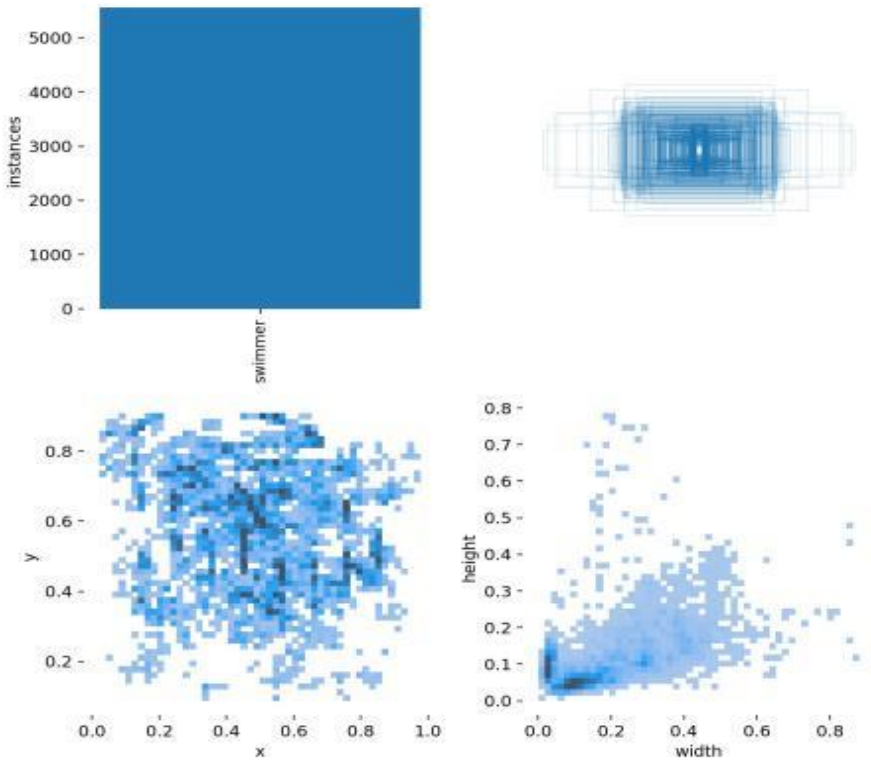


(c)

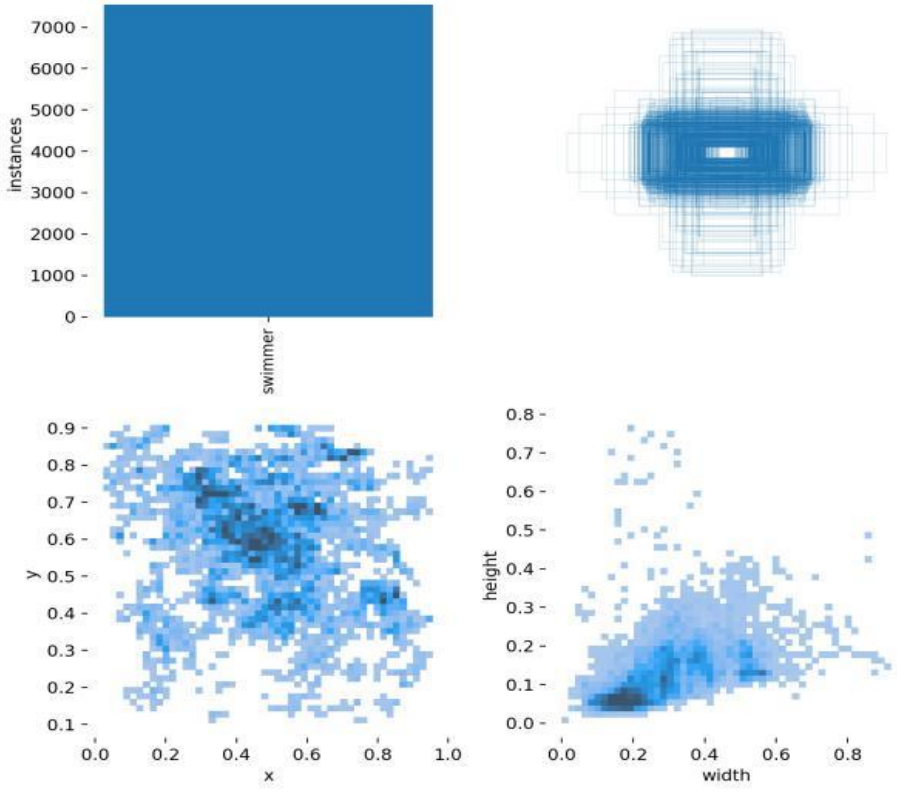
Figure 3.12: The structures of the three swimmer datasets (a) The data structures of the Swimmer-421 (b) The data structures of the Swimmer-1755 (c) The data structures of the Swimmer-3700 dataset



(a)


instances
5000
4000
3000
2000
1000
0
swimmer
y
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
x
height
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
width

(b)


instances
7000
6000
5000
4000
3000
2000
1000
0
swimmer
y
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.2
0.4
0.6
0.8
1.0
x
height
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
width

(c)

Figure 3.13: Annotated labels of the three swimmer datasets (a) The information of bounding box annotated in Swimmer-421 (b) The information of bounding box annotated in Swimmer-1755 (c) The information of bounding box annotated in Swimmer-3700

3.3.5 Evaluation Metrics

In the experiment, precision represents the probability of correct prediction among all the results predicted as positive samples. Recall indicates the proportion correctly predicted in all positive samples. We use recall as abscissa and precision as ordinate to generate the PR curve. We calculate the area under the PR curve as the average precision (AP), and mAP represents the average value of each category of AP. Intersection over Union (IoU) is employed to evaluate the correctness of the boundary box. It represents the ratio of the intersection and union of detection box and ground truth.

In our experiments, we set a threshold for the IOU. If the IOU of the detection box is greater than the threshold, it is considered as the right box. mAP@0.5 was applied to represent the mAP if the threshold of IOU is set to 0.5. mAP@0.5: 0.95 shows the average mAP over different IoU thresholds (from 0.5 to 0.95 in steps of 0.05). Furthermore, we take use of Frames Per Second (FPS) to evaluate the detection speed of the model, which refers to the number of images detected by the model per second.

3.3.6 Experiment Setting

We conducted two experiments to select the most suitable YOLOv5 model as the detection head of our multi-swimmer pose estimation model. There are totally three datasets with different sizes are used in the experiment. In the first experiment, we utilize the smallest data named Swimmer-421 to train and test YOLOv5s and YOLOv5x, respectively. We compared the experimental results of the two models to select the specifications of the model for the second experiment. In the second experiment, we take the dataset named Swimmer-1755 and the dataset named Swimmer-3700 dataset to train and test YOLOv5s respectively and compare the experimental results with the results of

YOLOv5s in the first experiment to explore the impact of the amount of data on the performance of the model. The input size of both YOLOv5s and YOLOv5x are 640×640.

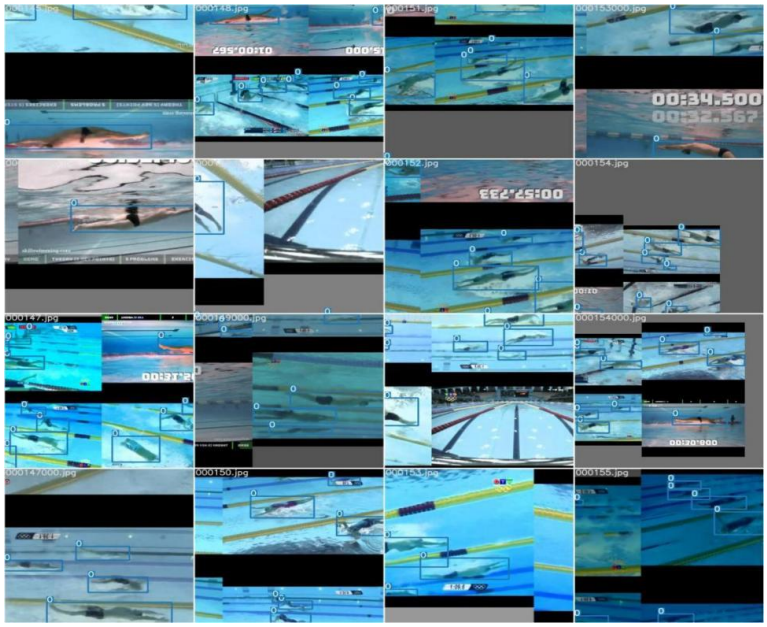


Figure 3.14: The result of the mosaic operation

We enhance the dataset before the training of each model. We set the parameters *hgain*, *sgain*, *vgain* to 1.50×10^{-2}, 0.70, 0.40, respectively to randomly enhance the gamut of the image. We randomly translate, scale, and flip the images with adjustment factors of 0.10, 0.50, 0.50, respectively. Furthermore, we also carried out the mosaic operation on the image. As shown in Figure 3.14, the mosaic operation refers to randomly selecting four training images, scaling, and putting them together into the same picture. Mosaic helps to improve the detection of small objects (Bochkovskiy *et al.*, 2020; Kong *et al.*, 2021).

In the experiment, we set the batch size to 16 to train YOLOv5s, and batch size to 6 to train YOLOv5x. Except for batch size, other training parameters of the two models are set as follows: We warm up the model in the first three epochs, which is a learning rate optimization method. The specific way is using a small learning rate at the beginning of the model training. During the process of warming up, the model tends to be stable gradually. The operation of warming up accelerates the convergence speed of the model and makes the model more effective (Goyal *et al.*, 2017). The pre-set learning rate for

training is applied after the first three epochs. We set the initial learning rate to 0.01 and the final learning rate to 2.00×10^{-3}. We apply stochastic gradient descent (SGD) in training, enable *nesterov*, set momentum to 0.94, and weight decay to $5.00\ x10^{-4}$.

All the models were trained based on NVIDIA GeForce RTX2080 super GPU. The operating system is Microsoft Windows 10. The virtual environment is Python 3.8. The training is ended after the 200-th iterations, with a total time of around 2.3 hours for the YOLOv5x on the Swimmer-421 dataset, 0.40 hours for the YOLOv5s on the Swimmer-421 dataset, 1.6 hours for YOLOv5s on the Swimmer-1755 dataset, 3.30 hours for the YOLOv5s on the Swimmer-3700 dataset.

3.4 YOLOv5 Interface

We set the parameters for YOLOv5 model in the process of parameters setting of HRNet, and then invoke the YOLOv5 model via the YOLOv5 interface with these parameters, to detect swimmers in the picture. The YOLOv5 interface mainly includes two parts: Image preprocessing and detection.

Before feeding the pictures into the YOLOv5 model, it is necessary to standardize the size of input pictures by adaptive scaling, YOLOv5 takes use of a new method modified from the original adaptive scaling strategy of YOLOv4, which takes advantage of the minimal information to fill the images. The new method greatly reduces information redundancy and substantially improves the detection speed of the model. In image preprocessing, we take use of Algorithm 1 to realize adaptive filling of the input images:

Algorithm 1 Adaptive Image Rescaling

Input: Original Image
Output: Rescaled Image

```
function autobox(imgo,input_shape = 640,color = (114, 114, 114),
 mode =′ auto′)
    imgshape ← [imgo_Width,imgo_Height]
    if input_shape is integer then
        auratio ← float(input_shape)/max(imgshape)
    else
        auratio ← max(input_shape)/max(imgshape)
    end if
    new_unpad ← (integer(round(imgshape[1] ∗ auratio)),
      integer(round(imgshape[0] ∗ auratio)))
    if mode is ′auto′ then
        dwidth ← MOD(input_shape − new_unpad[0], 32)/2
        dheight ← MOD(input_shape − new_unpad[1], 32)/2
    else if mode is ′square′ then
        dwidth ← (input_shape − new_unpad[0])/2
        dheight ← (input_shape − new_unpad[1])/2
    else
        raise error NotImplementedError
    end if
    topadd,bottomadd ← integer(round(dheight − 0.1)),
      integer(round(dheight + 0.1))
    leftadd,rightadd ← integer(round(dwidth − 0.1)),
      integer(round(dwidth + 0.1))
    img ← OpenCV.resize(imgo,new_unpad,
      interpolation = OpenCV.INTER_LINEAR)
    img =
OpenCV.copyMakeBorder(img,topadd,bottomadd,leftadd,
      rightadd,OpenCV.BORDER_CONSTANT,value = color)
    return img,auratio,dwidth,dheight
end function
```

In the detection section, we invoke YOLOv5s model by using the parameters in the HRNet, and send the image to a suitable format for detection, the detection process is implemented by using Algorithm 2.

Algorithm 2 Detection Process

Input: Image

Output: Detection Result

```
1:  function predict(images,color_mode =′ BGR′)
2:      img ← prepare_data(images,color_mode = color_mode)
3:      img ← load img to GPU
4:      if length(img) <= max_batch_size then
5:          detections ← model(img,augment = augment)[0]
6:      else
7:          detections ← torch.empty((img.shape[0], 10647, 85)) to GPU
8:          for i = 0 → length(img),stride = max_batch_size do
9:              detections[i : i + max_batch_size] = model
                (img[i : i + max_batch_size]).detach()
10:         end for
11:     end if
12:     detections ← Non − Maximum Suppression(detections, 0.5,
            0.45,agnostic = agnostic_nms)
13:     for i = 0 → length(images) do
14:         if detections[i] is not None then
15:             detections[i] ← scale_coords(detections[i],
                img[i].shape[1 :],images[i].shape[: 2]
16:         end if
17:     end for
18:     return detections
19: end function
```

where the function *scale_ coords* in line 15 is applied to map the predicted coordinates back to the original image. It is implemented in the following code:

Algorithm 3 Calculate Bounding Box Coordinates on Original Image

Input: NMS Result, Rescaled Image, Original Image
Output: Coordinates of Bounding Box

```
function scale_coords(coords,from_image_shape,to_image_shape)
    gain ← max(from_image_shape)/max(to_image_shape)
    coords[first column,third column] ← coords[first column,
      third column] − (from_image_shape[1] − to_image_shape[1]
      ∗ gain)/2
    coords[second column,fourth column] ← coords[second column,
      fourth column] − (from_image_shape[0] − to_image_shape[0]
      ∗ gain)/2
    coords[first four columns] ← coords[first four columns]/gain
    coords[first four columns] ← coords[first four columns].clamp
      (min = 0)
    return coords
end function
```

Chapter 4
Results

The main content of this chapter is the evaluation and performance of the swimmer detector based on YOLOv5, the evaluation of the single swimmer estimator is based on HRNet, the evaluation and performance of the multi-swimmer pose estimation model are based on the combination of YOLOv5 and HRNet. Moreover, we objectively analyze the limitations of our experiments.

4.1 Swimmer Detection

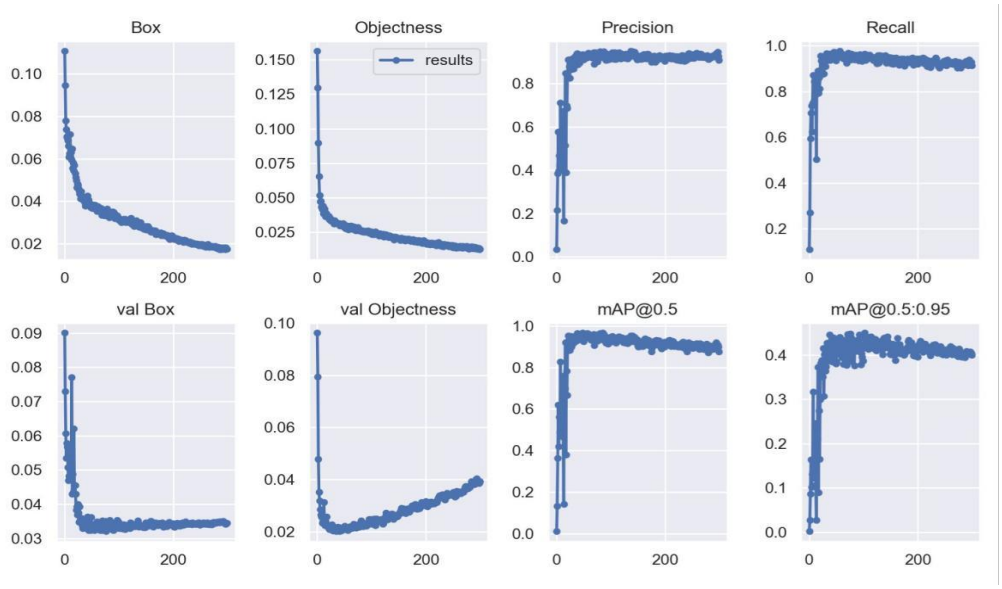
In this part, we unfold the training and testing results of YOLOv5x and YOLOv5s based on Swimmer-421 dataset, training and testing results of YOLOv5s based on the Swimmer-1755 dataset and Swimmer-3700 dataset. Table 4.1 shows the parameter quantity of the two models. YOLOv5x has more params and higher GFLOPs than that of YOLOv5s, which means higher spatial complexity and computational complexity.

Table 4.1: The parameters of YOLOv5s and YOLOv5x

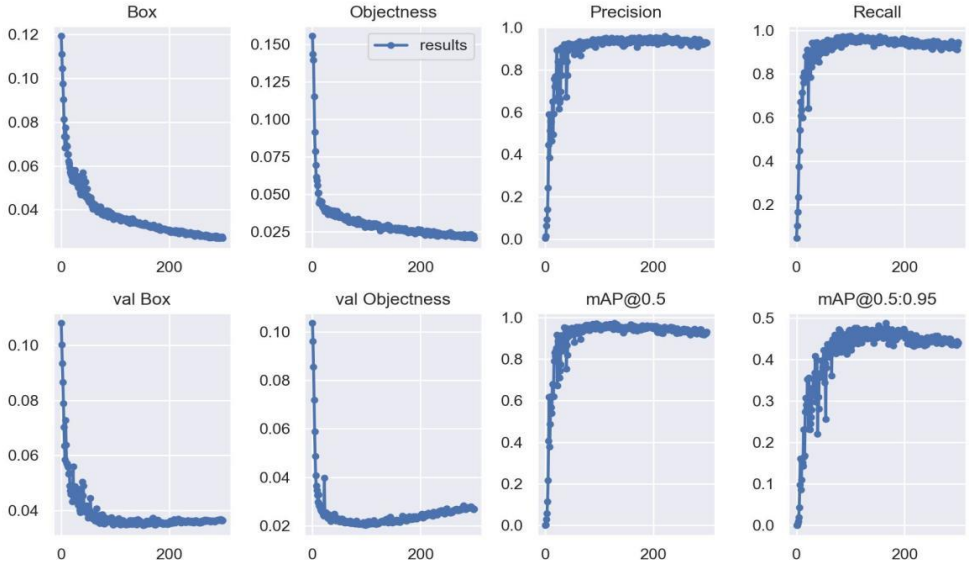
Backbone	YOLOv5s	YOLOv5x
#Params	7.10M	87.20M
GFLOPs	16.30	217.10

4.1.1 Results on Swimmer-421 Dataset

Figure 4.1 (a) shows that during the training of YOLOv5x, the objectness loss on the training set declined continuously, whereas the objectness loss on the validation set declined first and then rose, which is overfitting phenomenon. Figure 4.1 (b) shows overfitting also occurs during the training of YOLOv5s, but it is not as serious as YOLOv5x. Therefore, we believe that the reason for overfitting phenomenon is the scale of dataset is too small for the models with high complexity.



(a)



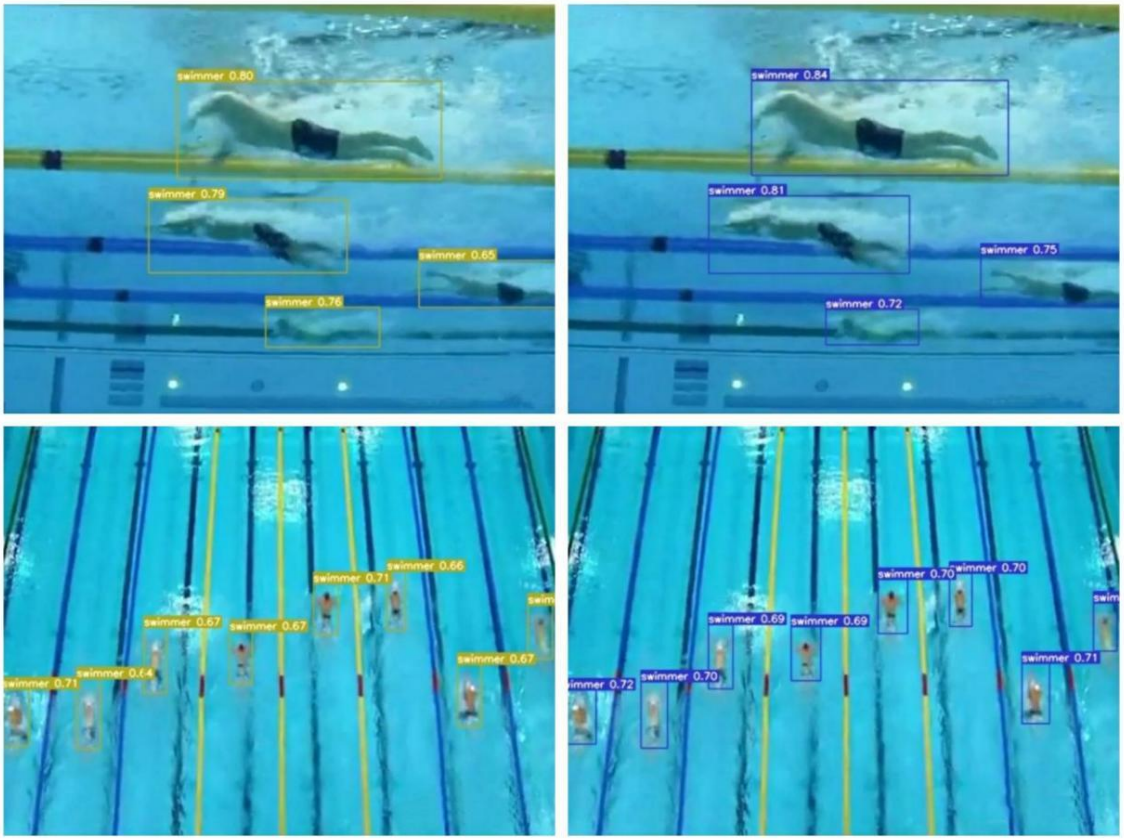
(b)

Figure 4.1: Training result of YOLOv5x and YOLOv5s on the Swimmer-421 dataset. (a) Training result of YOLOv5x on Swimmer421 dataset (b) Training result of YOLOv5s based on Swimmer421 dataset. The figures named *box* and *val_box* show the loss of the bounding box on the training set and the validation set during the training period, respectively. The figures named *objectness* and *val_objectness* show the loss of the classification of foreground and background on the training set and the validation set, respectively. The other four figures show the following indicators of the model on the validation set during the training period: Precision, Recall, mAP@0.5, and mAP@0.5: 0.95.

The results of YOLO v5x and YOLOv5s based on the test set are shown in Table 4.2, precision (95.00%) and mAP@.5:.95 (48.70%) of YOLOv5s are higher than those of YOLOv5x, which may result from the severe overfitting of YOLOv5x during the training. In terms of detection speed the average detection speed of YOLOv5s for each image is 196.0fps, which is much higher than 47.00 fps of YOLOv5x, the YOLOv5s model outperformed YOLOv5x in both detection ability and detection speed. Figure 4.2 shows the performance of the two models for objects of different sizes, the detection effect of both models is excellent.

Table 4.2: Test result of YOLOv5s and YOLOv5x that trained with Swimmer-421 set

Model	YOLOv5x	YOLOv5s
Precision	94.40%	95.00%
Recall	96.30%	94.80%
mAP@0.5	95.10%	94.90%
mAP@0.5:0.95	44.90%	48.70%
Time(Inference)	19.90 ms	4.00 ms
Time(NMS)	1.30 ms	1.10 ms
Speed(Total)	47.00 fps	196.00 fps



(a) (b)

Figure 4.2: (a) The examples of detection results for YOLOv5s (b) The example of detection results for YOLOv5x.

Swimmer detection is a simple classification problem, there are only two types of objects to be detected: Swimmer and background, and the amount of data for the experiment is limited, the network scale of YOLOv5s is already sufficient for our task, YOLOv5x with deeper and wider network structure is not only easy to overfit but also has slower detection speed, therefore, we take use of YOLOv5s for the latter experiments.

4.1.2 Results on Swimmer-1755 and Swimmer-3700

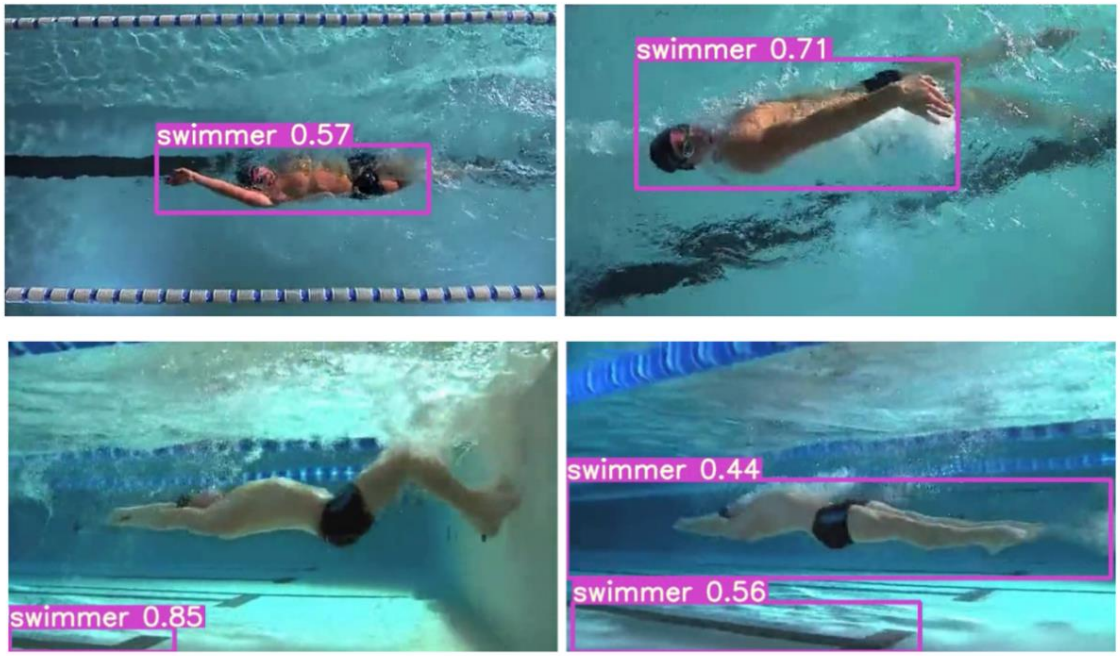
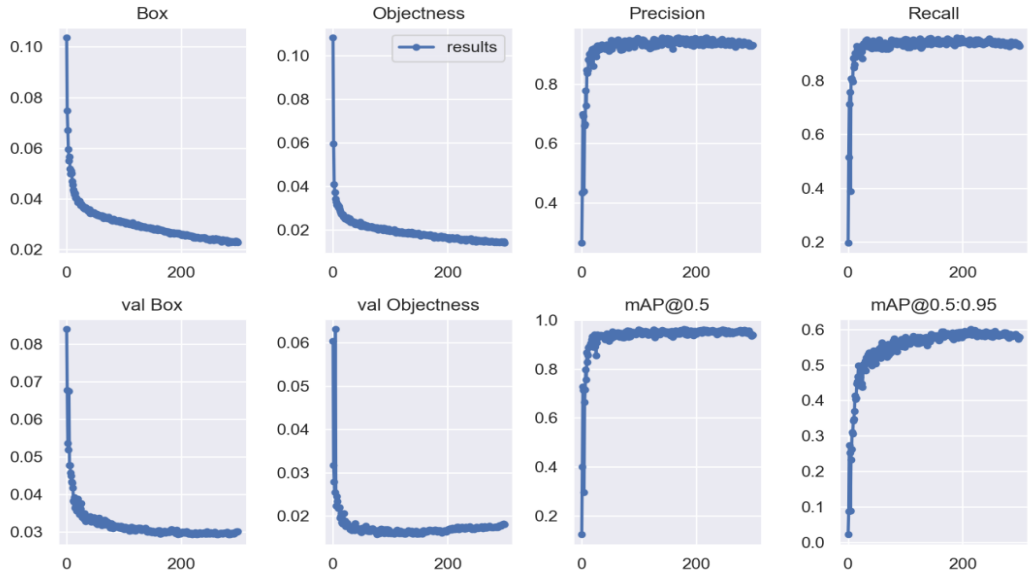


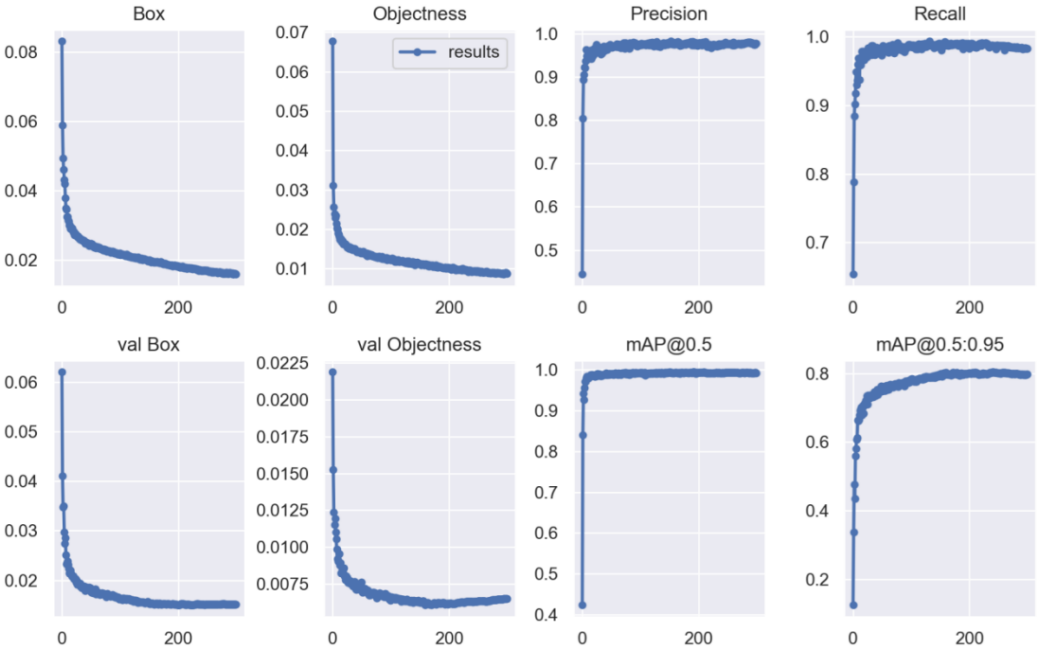
Figure 4.3: Detection results of YOLOv5s on the wild dataset.

We optimized the performance of YOLOv5s to generate a swimmer detector for multi-swimmer pose estimation. In order to obtain more objective results, we take use of the trained model to detect a set of data other than the three datasets we proposed to demonstrate the variation of their detection ability, which is hereafter referred to as the wild dataset. The example of the detection results on the wild dataset with YOLOv5s that trained on Swimmer-421 is shown in Figure 4.3. The model is prone to misunderstanding the background as swimmers in the underwater view compared to the other view, and the error of the bounding boxes detected is also greater. We believe that these phenomena

arise because YOLOv5s overfits the information in Swimmer-421, taking some background information as spuriously correct features. What's more, the dataset in the experiment is small. The angle of a swimmer in the dataset is limited, so the model is prone to errors when detecting some strange angles.



(a)



(b)

Figure 4.4: (a)Training result of YOLOv5s based on the Swimmer-1755 dataset (b) Training result of YOLOv5s based on Swimmer-3700 dataset

We trained and tested the YOLOv5s model by using training set and the test set of the Swimmer-1755 dataset and the Swimmer-3700 dataset respectively. As shown in Figure 4.4, the overfitting is significantly improved by training with the Swimmer-1755 dataset and almost disappeared by training with the Swimmer-3700 dataset, we see that the training set of Swimmer-3,700 has the most appropriate data volume for YOLOv5s.

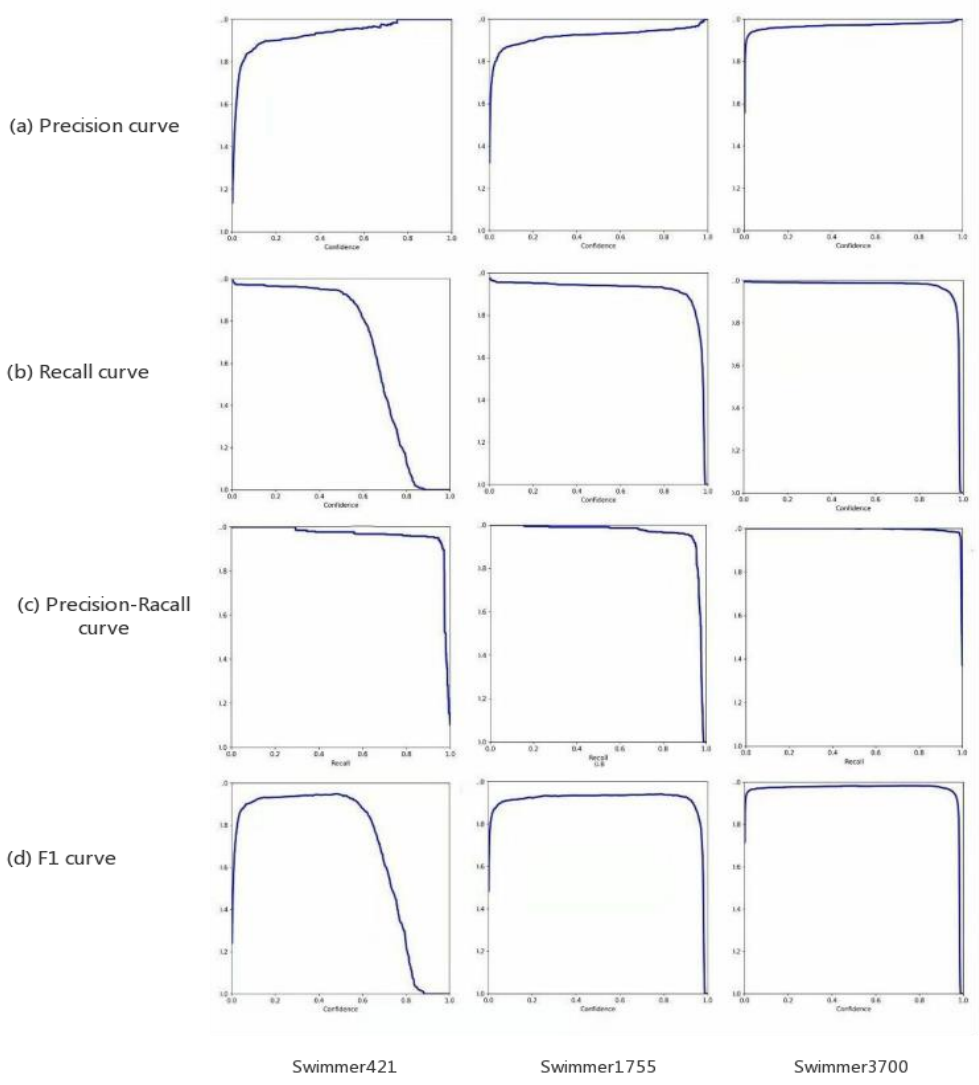


Figure 4.5: The precision curve, recall curve, precision-recall curve, F1 score curve of YOLOv5s trained with different datasets.

The test results of the new models are shown in Table 4.3. With the increase of training data, the precision, recall, and mAP of this model are all increased. For instance, the model trained on Swimmer-3700 improves the mAP@0.5 by 4.40% (versus model trained on Swimmer-421), 2.50% (versus model trained on Swimmer-1755). As shown

in Figure 4.5, YOLOv5 trained with Swimmer-3700 dataset obtained the best performance on all the curves.

Table 4.3: Test result of YOLOv5s that trained with different datasets

Models	YOLOv5s		
Datasets	Swimmer-421	Swimmer-1755	Swimmer-3700
Precisions	95.00%	94.70%	97.60%
Recalls	94.80%	97.90%	98.50%
mAP@0.5	94.90%	96.80%	99.30%
mAP@0.5:0.95	48.70%	59.30%	80.30%
Time(Inference)	4.00 ms	3.30 ms	2.90 ms
Time(NMS)	1.10 ms	1.00 ms	1.00 ms
Speed(Total)	196.00 fps	232.6 fps	256.40 fps

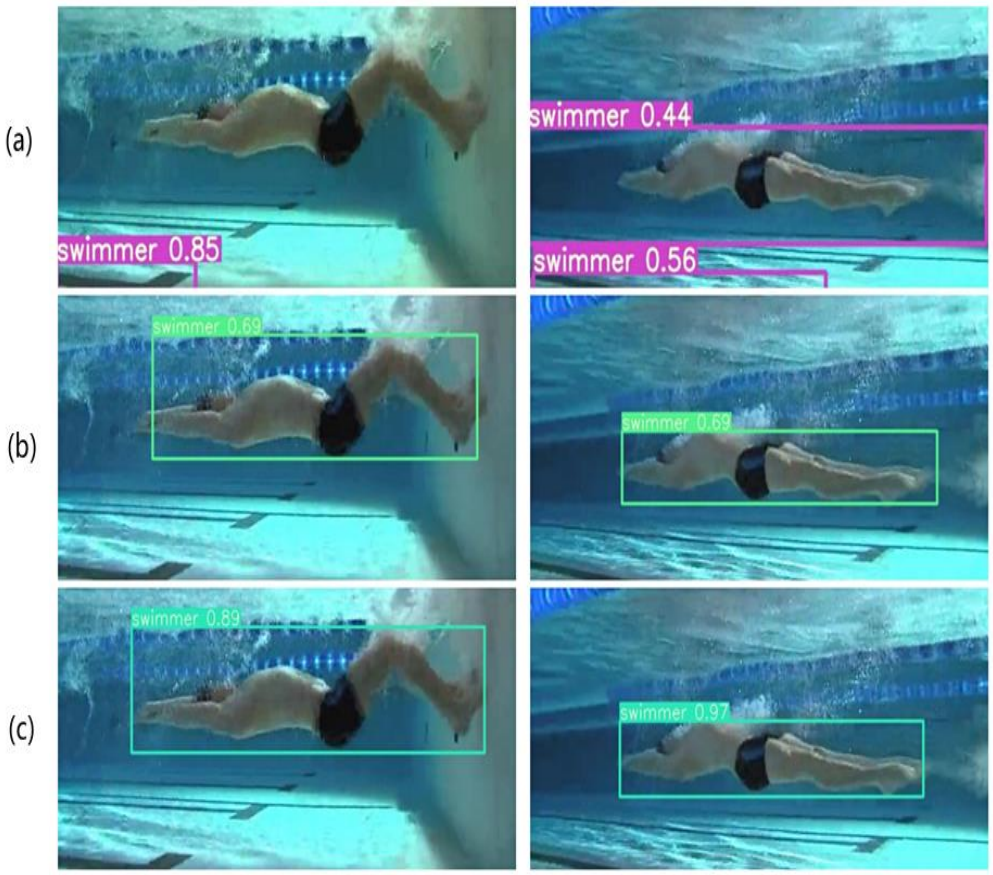


Figure 4.6: The detection results based on the wild dataset for different models (a) Model trained with Swimmer-421 (b) Model trained with Swimmer-1755 (c) Model trained with Swimmer-3700

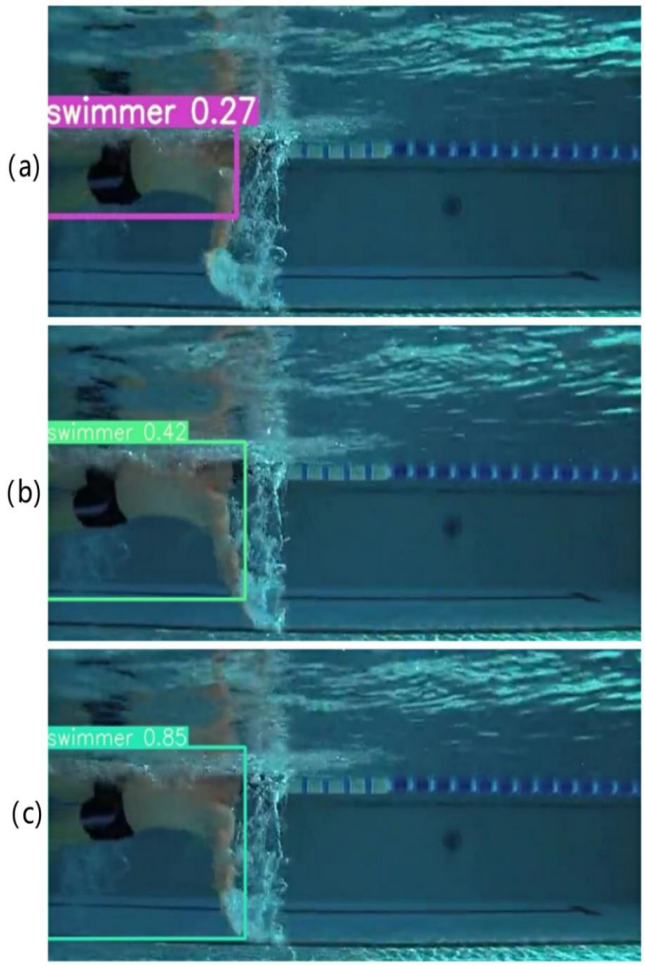


Figure 4.7: The detection results based on the wild dataset for different models (a) Model trained with Swimmer-421 (b) Model trained with Swimmer-1755 (c) Model trained with Swimmer-3700

Figure 4.6 shows the evolution of performance for the models based on the images with easily confusing backgrounds. With the increase in the amount of data, the phenomenon of false detection gradually disappeared. As shown in Figure 4.7, the boxes generated by the model become more and more accurate. According to the experimental results, YOLOv5s which was trained based on Swimmer-3700 performs the best on both the test set and the wild datasets, so we take it as the final swimmer detector.

4.2 Result of Single-Swimmer Pose Estimation Model

We trained the HRNet of two sizes three times by using three different loss functions, respectively. The input of the two networks is the group of bounding boxes including a single swimmer cropped from the input images. The position of the bounding boxes is annotated by humans.

We evaluate the performance of the model using indicators such as AP, AP^M, APL, AR, the results are shown in Table 4.5 and Table 4.6. Both of the two networks were trained with MSE loss that achieve the most ideal effect. Compared with that of using other loss functions, the large network has obtained the best value on all indexes except AP^M and AR^M. For example, we obtained an AP rate of 95.30% and an AR rate of 96.60%, but these scores are only slightly superior to the results of the small network. The small network trained with MSE loss obtained an AP rate of 95.20% as well as an AR rate of 96.50%. The small networks are ahead of large networks by 1.50% in AP^M and 1.60% in AR^M. The network using MSEBone loss for joint training obtained the worst results, which may be due to the difficulty of learning spatial information, the model needs a large amount of data. However, our dataset is small, which is not enough to obtain ideal results.

Table 4.4 shows the parameter quantity of this model, the parameter quantity of the small network is 28.50M while the large network is 63.60M, which means, the small model has less spatial complexity than the large network. In terms of computational complexity of the model, the GFLOPs were 16.00 for the small network and 32.90 for the large network. This illustrates the low computational complexity of small networks, which also means that small networks possess lower temporal complexity. So, the small network is faster in training and prediction.

Table 4.4: The parameters of HRNet with two sizes

Backbone	HRNet-W32	HRNet-W48
#Params	28.50M	63.60M
GFLOPs	16.00	32.90

Table 4.5: The performance of HRNet-W32 on the test set

Backbone	HRNet-W32		
Loss	MSE (%)	MSE+OHKM (%)	MSE+BONE (%)
AP	95.20	87.40	42.00
AP^{50}	99.00	99.00	95.40
AP^{75}	98.00	98.00	24.80
AP^{M}	92.40	84.20	36.20
AP^{L}	95.80	88.20	44.30
AR	96.50	91.20	48.60
AR^{50}	99.70	99.70	97.30
AR^{75}	98.60	98.90	40.90
AR^{M}	95.70	90.40	47.80
AR^{L}	96.60	91.30	48.70

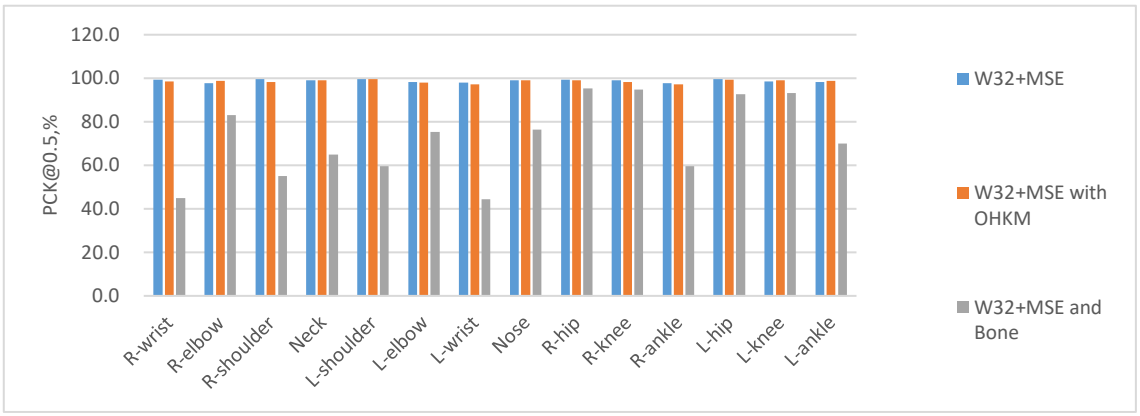
Table 4.6: The performance of HRNet-W48 on the test set

Backbones	HRNet-W48		
Loss	MSE (%)	MSE+OHKM (%)	MSE+BONE (%)
AP	95.30	95.00	76.40
AP^{50}	99.00	99.00	99.00
AP^{75}	99.00	97.00	85.00
AP^{M}	90.90	91.80	68.30
AP^{L}	96.10	95.50	78.20
AR	96.60	96.40	80.90
AR^{50}	99.70	99.70	99.20
AR^{75}	99.20	98.60	88.20
AR^{M}	94.10	95.30	76.70
AR^{L}	97.10	96.50	81.60

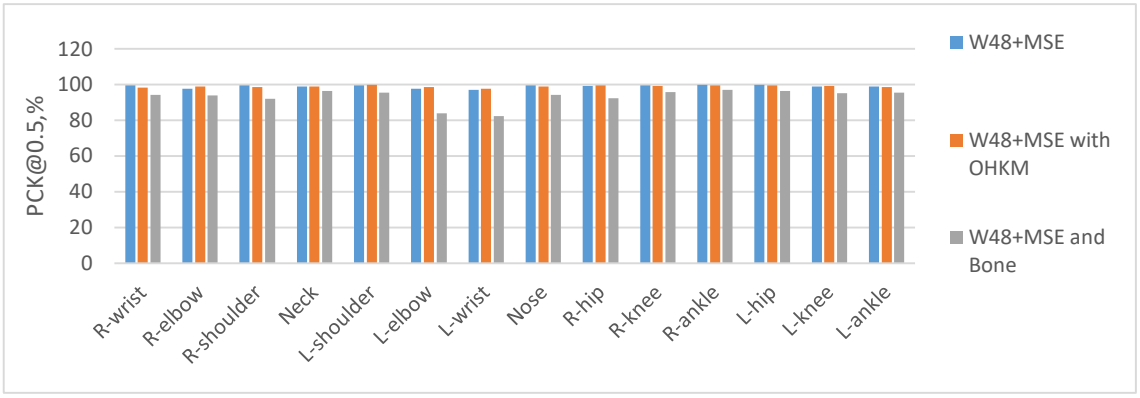
Figure 4.8 shows the performance of HRNet-W48 and HRNet-W32 networks for each type of key point using PCK@0.50 as the metric. Taken the large network trained

with MSE as an example, the network has a strong recognition ability for most joint points.

For example, the scores on joint points of shoulders, noses, and hips for large networks are more than 99.00%. The recognition ability of the network for elbow and wrist joints is weak, only a score of about 97.00% is obtained. The reasons for this phenomenon are: Firstly, the hand joints need to enter and exit the water frequently to complete the stroke, which makes the hand joints disappear intermittently in the underwater perspective. Consequently, the number of hand key points in the dataset is less than that of other joint points. Secondly, the bubbles caused by the arm in the stroke will also block the hand joint points, which blurs the characteristics of the hand joints. In general, our model still achieves ideal detection results after training with a small amount of data. This proves that our method is very practical in swimmer pose estimation.



(a)



(b)

Figure 4.8: The performance of networks for each type of keypoints by using PCK@0.50 metric (a) The scores of HRNet-W32 with MSE loss, MSEOHKM loss, MSEBone loss, respectively (b) The scores of HRNet-W48 that trained with MSE, MSEOHKM loss, MSEBone loss, respectively

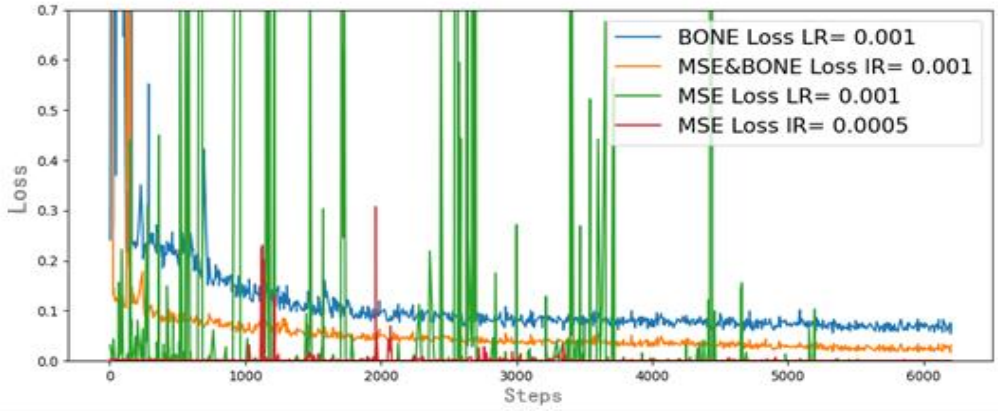


Figure 4.9: Value loss curve of HRNet-W48 during the training with different training parameters. LR means initial learning rate.

Value loss of this model is based on the validation set during the training time. As shown in Figure 4.9, though the network trained with MSE loss achieved the best performance, its value loss fluctuated violently in the training process. We believe that this is caused by using a small size batch and a big learning rate during the training. Since the batch size is limited by computing resources, we reduced the learning rate and retrained the HRNet-W48. We set the initial learning rate as 5.00×10^{-4}, and the learning rate is attenuated at 50, 80, 110, 140, 170, and 190 epochs with 0.70 as the decay rate. The red curve in Figure 4.9 shows the changes of value loss during training, compared to the results using 1.00×10^{-3} as the initial learning rate, it converges better.

Table 4.7: The performance of HRNet-W48 trained with different initial learning rates

Backbone	HRNet-W48	
Loss	MSE	
Learning rates	1.00×10^{-3}	5.00×10^{-4}
AP	95.30%	95.60%
AP^{50}	99.00%	99.00%
AP^{75}	99.00%	99.00%
AP^{M}	90.90%	92.50%

AP^L	96.10%	96.20%
AR	96.60%	97.10%
AR^{50}	99.70%	99.70%
AR^{75}	99.20%	99.20%
AR^M	94.10%	95.90%
AR^L	97.10%	97.30%

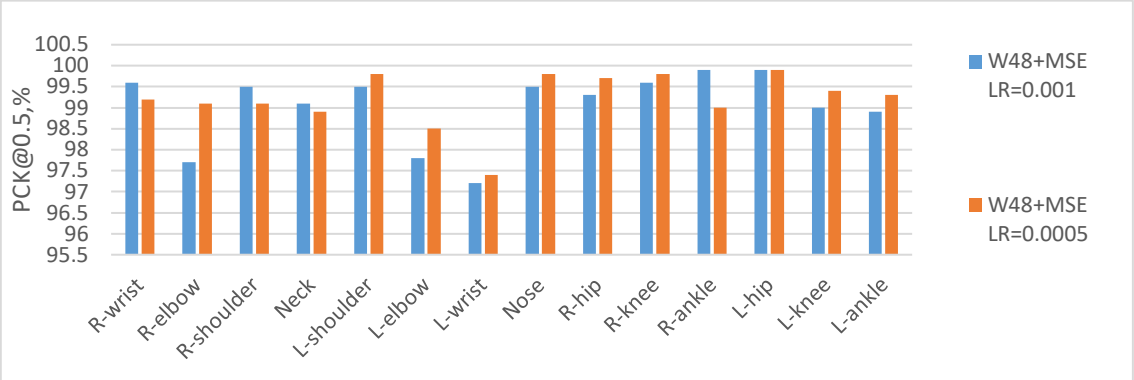


Figure 4.10: The performance of HRNet-W48 that trained with different learning rates for each type of keypoints. LR means initial learning rate.

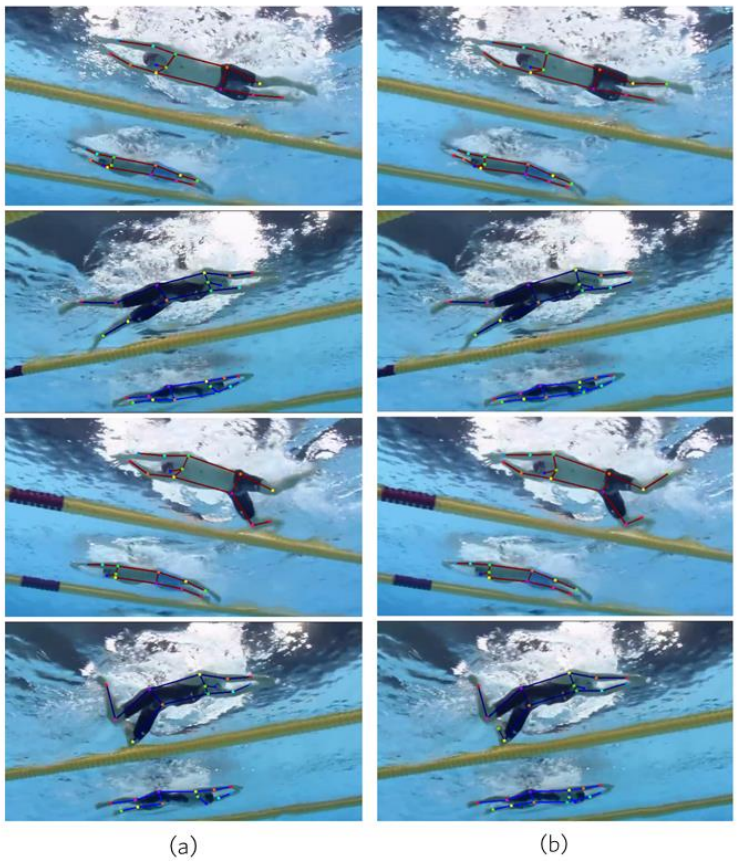


Figure 4.11: The result on the recording of the swimming event (a) The result of HRNet-W48 takes use of 1.00×10^{-3} as the initial learning rate (b) The result of HRNet-W48 utilizes 5.00×10^{-4} as the initial learning rate.

The testing results of the HRNet-W48 trained with different learning rate are shown in Table 4.7, which reveal that the new model that trained with a smaller learning rate performs better on most evaluating indicators. Figure 4.10 shows the performance of the two models for each type of key point, the new model obtains stronger capability on identifying hard key points, such as elbow, knee, and ankle. As shown in Figure 4.11, the new model accurately detects the missing joint points of the previous model. Therefore, in the following experiments, we combine the new model with YOLOv5s to form our multi-swimmer pose estimation model.

4.3 Result of Multi-swimmer Pose Estimation Model

In this section, we show the test results of the multi-swimmer pose estimation model, which is actually the result of changing the input of the single-swimmer pose estimation model from the manually annotated bounding boxes to the bounding boxes detected by the swimmer detector. The swimmer detector and the single-swimmer pose estimation model are obtained from the previous experiments of this thesis.

As shown in Table 4.8, we evaluate the model using metrics such as AP, AP^M, AP^L, AR, and the performance of the single-swimmer pose estimation model is shown again for comparison. The AP rate of the multi-swimmer pose estimation model is as same as that of the single-swimmer pose estimation model, the AR rate of 96.90% is slightly lower than that of the single swimmer pose estimation model (97.10%). In general, the swimmer detector has no significant impact on the performance of the model.

Table 4.8: The performance of our multi-swimmer pose estimation model on the test set

Backbones	HRNet-W48 (%)	HRNet-W48 &YOLOv5s (%)
AP	95.60	95.60
AP^{50}	99.00	99.00

AP^{75}	99.00	97.00
AP^{M}	92.50	92.50
AP^{L}	96.20	96.20
AR	97.10	96.90
AR^{50}	99.70	99.40
AR75	99.20	99.10
AR^{M}	95.90	95.60
AR^{L}	97.30	97.30

4.4 Limitations of This Research Project

(1) The scenes on the two kinds of datasets we used are not rich, so the recognition ability of our model will be impaired in cross-scene tasks.

(2) Most of our experimental results are based on small datasets with a single class, which makes our conclusions have a bit of limitation and are not applicable to the results obtained on large multiclassification datasets.

Chapter 5
Analysis and Discussions

We systematically summarize and analyze the experimental results of all the models in this chapter.

5.1 Analysis

The precision of YOLOv5s trained on the Swimmer-421 dataset is 95.00% which is better than that (94.40%) of YOLOv5x trained on the Swimmer-421 dataset and that (94.70%) of the YOLOv5s trained on the Swimmer-1755 dataset. YOLOv5s got the highest precision at 97.60% after the training based on the Swimmer-3700 dataset. The mAP@0.5 of YOLOv5s trained on the Swimmer-421 dataset is 94.90% which is lower than that (95.10%) of YOLOv5x trained on the Swimmer-421 dataset and that (96.80%) of YOLOv5s trained on the Swimmer-1755 dataset. YOLOv5s achieved the highest mAP@0.5 at 99.30% after training on the Swimmer-3700. In terms of detection speed, the average time for YOLOv5s trained on the Swimmer-421 to detect each image is 196.00 fps, outperforming the 47.00 fps of YOLOv5x.

Among the models that used 1.00×10^{-3} as the initial learning rate during the training, the HRNet-W48 achieved an AP rate of 95.30% and an AP^{L} rate of 96.10% after training by using MSE loss which outperforms the HRNet-W48 trained by using other loss functions, and all the HRNet-W32. However, the AP^{M} of HRNet-W32 trained with MSE loss is 92.40% which is 1.5 points higher than the AP^{M} of HRNet-W48 trained with MSE loss. The highest AR rate 96.60% and the highest AR^{L} rate 97.10% are achieved by HRNet-W48 trained with MSE loss. The highest AR^{M} rate 95.70% is achieved by the HRNet-W32 trained with MSE loss.

On the other hand, compared with the HRNet-W48 taking use of 1.00×10^{-3} as the initial learning rate during the training, the HRNet-W48 makings use of 5.00×10^{-4} as the initial learning rate get better results on most of the metrics, such as AP^{M} of 92.50%, an AR^{M} of 95.90%, which also get the best performance, among all the models.

The AP of our multi-swimmer pose estimation model is 95.60% which is as same as that of the trained HRNet-W48 that takes use of ground truth bounding box as input. The AR rate of our multi-swimmer pose estimation model is 96.90% which reduces by 0.20% compared with 97.10% of the trained HRNet-W48.

5.2 Discussion

After training the YOLOv5s and YOLOv5x on the Swimmer-421 dataset, we found that both of the two models achieved high degrees of performance, compared with the YOLOv5x, YOLOv5s performs better on most evaluation metrics. By contrasting the loss curves during the training, we found that overfitting happens on both of the two models and the overfitting is much severe for YOLOv5x which has higher model complexity, which is also the reason why it performed worse than the YOLOv5s. We believe that overfitting arises because our dataset contains only a single class of objects which has limited variation in the appearance and motion, the features that need to be learned are limited, and the size of the Swimmer-421 dataset is small which both make the models over learn bootless features based on the training set, treating the background features as false-positive features. This outcome proves that the network size of the YOLOv5s is adequate for our task and a deeper network does not mean better performance. Furthermore, in terms of detection speed, the advantage of the YOLOv5s is obvious, it acquired four times faster than the YOLOv5x.

According to the results of the first experiment, we decided to take YOLOv5s to implement the swimmer detector. In the second experiment, we took use of two larger datasets to train YOLOv5s, with the intent of obtaining a swimmer detector that has better performance. YOLOv5s which was trained based on the Swimmer-3700 dataset performs best. The result shows that larger datasets assist the model to fit more effective features and further converge. Finally, we take the best performing model as the swimmer detector of the multi-swimmer pose estimation.

We compared the result of HRNet-w32 and HRNet-w48 that was trained with MSE loss, MSEOHKM loss, and MSEBone loss respectively. Among the three obtained HRNet-w32 models, the network trained with MSE loss achieved the best performance in most evaluation metrics. Similarly, HRNet-w48 trained with MSE also achieved the best performance compared with the other two HRNet-w48. We think the reason is that more data is needed while fitting difficult samples and spatial relationships, so the size of

our dataset limits other two networks to obtain better results. In addition, HRNet-w48 trained with MSE loss performs better than the HRNet-w32 trained with MSE loss on objects other than medium objects.

In order to improve the performance of the HRNet-w48 trained with MSE loss, we retrained the HRNet-w48 and reduced the initial learning rate during the training to match with the small batch size. Compared to the results of the model using 1.00×10^{-3} as the initial learning rate, the loss of the retrained model converges better. We finally chose the retrained HRNet-w48 as part of our multi-swimmer pose estimation model.

Our multi-swimmer pose estimation model was implemented by combining the selected YOLOv5s and the selected HRNet-w48. Our model gets the same precision as that of the selected HRNet-w48, but with a slightly lower recall rate than that of selected HRNet-w48. In terms of computational time, the swimmer detector has an average detection time of only 3.90 ms for each image which hardly contributes to the inferential time for the model. The experimental result illustrates that the performance of our proposed top-down multi-swimmer pose estimation model is hardly limited by the swimmer detector.

Chapter 6
Conclusion and Future Work

In this chapter, we will demonstrate the conclusion of the research and envision our future work.

6.1 Conclusion

The purpose of this thesis is to devise and implement a pose estimation method that fits the scenarios containing multiple swimmers. In this thesis, we introduced the method for human pose estimation and object detection. We also reviewed the development process and current research trends related to the pose estimation of swimmers. By comparing the existing methods, we finally chose the top-down method to achieve an end-to-end multi-swimmer pose estimation model based on deep learning.

In our approach, HRNet was employed to enforce the single-swimmer pose estimation model which innovatively alters the link between high and low-resolution from series to parallel, thus retaining the high-resolution representation in the whole network structure. At the same time, the network also introduces a cross-fusion module in high and low-resolution to improve the performance of the model. Compared with the single-swimmer pose estimation model implemented by traditional methods in the past, our model supports parallel computing such as GPU processing, which greatly shortens the running time of the model. Our model saves a lot of complex training processes which is easier to be deployed and implemented. On the other hand, YOLOv5s was applied to procure the swimmer detector, which achieves ideal performance on both detection speed and detection ability.

In summary, through combing the excellent performance swimmer detector and single-swimmer pose estimation model, our multi-swimmer pose estimation model breaks the limitations of the top-down-based method and achieved almost the same well performance as the single-swimmer pose estimation model with minimal increase in detection time. Compared with the prevenient methods for the pose estimation of swimmers which are only applicable to scenarios with a single swimmer, our method fits for the scenarios containing multiple swimmers, which means a wider application range.

6.2 Future Work

At present, our model does not fully meet the standard of real-time detection in terms of speed. In the future, we plan to adjust the structure of the single-swimmer pose estimation model so as to obtain a higher detection speed. The scale of our dataset is small, the expansion of the dataset will be completed in the future.

Moreover, our model is limited to apply the information in a single image so as to predict the position and pose of the object. However, in the prediction of continuous frames, the information of front and rear frames is also very important, which not only reduces the prediction time but also plays a reference role in the face of occlusion and other problems. As a result, we plan to apply this kind of information for detection.

The pose estimation of swimmers is the basis of many machine vision algorithms, the swimmers' pose correction based on pose estimation is the research direction of our future work.

References

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *Proceedings of International Conference on Engineering and Technology (ICET)* (pp. 1-6).

Al-Sarayreh, M., Reis, M., Yan, W., Klette, R. (2018) A deep learning approach for detecting the adulteration in red-meat products by hyperspectral imaging. In *Proceedings of IEEE Annual Workshop on Smart Sensors, Measurements and Instrumentation.*

Al-Sarayreha, M., Reis, M., Yan, W., Klette, R. (2019) A sequential CNN approach for foreign object detection in hyperspectral images. In *Proceedings of CAIP*.

Al-Sarayreha, M., Reis, M., Yan, W., Klette, R. (2020) Potential of deep learning and snapshot hyperspectral imaging for classification of species in meat. *Food Control.*

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., . & Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and architectures. *Electronics*, *8*(3), 292.

Andriluka, M., Roth, S., & Schiele, B. (2009). Pictorial structures revisited: People detection and articulated pose estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1014-1021).

Bashar, A. (2019). Survey on evolving deep learning neural network architectures. *Journal of Artificial Intelligence*, *1*(02), 73-82.

Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Cao, Z., Hidalgo, G., Simon, T., Wei, S. E., & Sheikh, Y. (2019). OpenPose: Realtime multi-person 2D pose estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(1), 172-186.

Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime multi-person 2D pose estimation using part affinity fields. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7291-7299).

Carreira, J., Agrawal, P., Fragkiadaki, K., & Malik, J. (2016). Human pose estimation with iterative error feedback. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4733-4742).

Chen, X., & Yuille, A. (2014). Articulated pose estimation by a graphical model with image dependent pairwise relations. *arXiv preprint arXiv:1407.3399*.

Chen, Y., Tian, Y., & He, M. (2020). Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, *192*, 102897.

Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., & Sun, J. (2018). Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7103-7112).

Chen, Y., Zhang, C., Qiao, T., Xiong, J., & Liu, B. (2021). Ship detection in optical sensing images based on YOLOv5. In *Proceedings of International Conference on Graphics and Image Processing (ICGIP 2020)* (Vol. 11720, p. 117200E).

Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016). 3D U-Net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 424-432). Springer.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273-297.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object detection via region-based fully convolutional networks. In *Proceedings of Advances in Neural Information Processing Systems* (pp. 379-387).

Dake, S., Nguyen, M., Yan, W., Kazi, S. (2019) Human tumour detection using active contour and region growing segmentation. In Proceedings of International Conference and Workshops on Recent Advances and Innovations.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)* (Vol. 1, pp. 886-893).

Dang, Q., Yin, J., Wang, B., & Zheng, W. (2019). Deep learning based 2D human pose estimation: A survey. *Tsinghua Science and Technology*, *24*(6), 663-676

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Einfalt, M., Zecha, D., & Lienhart, R. (2018). Activity-conditioned continuous human pose estimation for performance analysis of athletes using the example of swimming. In *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 446-455).

Elfwing, S., Uchibe, E., & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks, 107,* 3-11.

Fang, Y., Guo, X., Chen, K., Zhou, Z., & Ye, Q. (2021). Accurate and automated detection of surface knots on sawn timbers using YOLOv5 model. *BioResources, 16*(3).

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2009). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*(9), 1627-1645.

Fischler, M. A., & Elschlager, R. A. (1973). The representation and matching of pictorial structures. *IEEE Transactions on computers, 100*(1), 67-92.

Fu, C. Y., Liu, W., Ranga, A., Tyagi, A., & Berg, A. C. (2017). DSSD: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*.

Fu, Y., Yan, W. (2022) Fruit freshness grading using deep learning. *Springer Nature Computer Science.*

Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and Cooperation in Neural Nets* (pp. 267-285). Springer.

Gao, X., Nguyen, M., Yan, W. (2021) Human face image inpainting based on generative adversarial network. In *IEEE IVCNZ.*

Ghiasi, G., Lin, T. Y., & Le, Q. V. (2018). Dropblock: A regularization method for convolutional networks. *arXiv preprint arXiv:1810.12890*.

Giblin, G., Tor, E., & Parrington, L. (2016). The impact of technology on elite sports performance. *Sensoria: A Journal of Mind, Brain & Culture*, *12*(2).

Girshick, R. (2015). Fast R-CNN. In *Proceedings of IEEE International Conference on Computer Vision* (pp. 1440-1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 580-587).

Gowdra, N., Sinha, R., MacDonell, S., Yan, W. (2021) Mitigating severe over-parameterization in deep convolutional neural networks through forced feature abstraction and compression with an entropy-based heuristic. *Pattern Recognition.*

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., ... & He, K. (2017). Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Greif, T., & Lienhart, R. (2010). An annotated dataset for pose estimation of swimmers. Technical Report.

Gu, Q., Yang, J., Yan, W., Li, Y., Klette, R. (2017) Local Fast R-CNN flow for object-centric event recognition in complex traffic scenes. In *Proceedings of Pacific-Rim Symposium on Image and Video Technology*s (pp.439-452)

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, *187*, 27-48.

Haner, S., Svärm, L., Ask, E., & Heyden, A. (2015). Joint under and over water calibration of a swimmer tracking system. In *Proceedings of ICPRAM (2)* (pp. 142-149).

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In *Proceedings of IEEE International Conference on Computer Vision* (pp. 2961-2969).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904-1916.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770-778).

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527-1554.

Hosang, J., Benenson, R., Dollár, P., & Schiele, B. (2015). What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(4), 814-830.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4700-4708).

Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M., & Schiele, B. (2016). Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *Proceedings of European Conference on Computer Vision* (pp. 34-50). Springer.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning* (pp. 448-456). PMLR.

Ji, H., Liu, Z., Yan, W., Klette, R. (2019) Early diagnosis of Alzheimer's disease using deep learning. In *Proceedings of ICCCV*.

Ji, H., Liu, Z., Yan, W., Klette, R. (2019) Early diagnosis of Alzheimer's disease based on selective kernel network with spatial attention. In *Proceedings of ACPR* (pp. 503-515)

Jocher, G., Nishimura, K., Mineeva, T., & Vilariño, R. (2020). YOLOv5. *Code Repository https://github. com/ultralytics/yolov5*.

Josyula, R., & Ostadabbas, S. (2021). A review on human pose estimation. *arXiv preprint arXiv:2110.06877*.

Kasper-Eulaers, M., Hahn, N., Berger, S., Sebulonsen, T., Myrland, Ø., & Kummervold, P. E. (2021). Detecting heavy goods vehicles in rest areas in winter conditions using YOLOv5. *Algorithms*, *14*(4), 114.

Kocabas, M., Karagoz, S., & Akbas, E. (2018). Multiposenet: Fast multi-person pose estimation using pose residual network. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 417-433).

Kong, W., Li, D., Li, J., Liu, D., Liu, Q., Lin, B., ... & Xu, C. (2021). Detection of golden crucian carp based on YOLOv5. In *Proceedings of International Conference on Artificial Intelligence and Education (ICAIE)* (pp. 283-286).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, *25*, 1097-1105.

Kulon, D., Guler, R. A., Kokkinos, I., Bronstein, M. M., & Zafeiriou, S. (2020). Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4990-5000).

Law, H., & Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 734-750).

Le, R., Nguyen, M., Yan, W. (2021) Training object detection neural network with synthetic dataset for transportation signs. In *Proceedings of IEEE IVCNZ 2021*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation, 1*(4), 541-551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Li, B., Zou, J., Wang, L., Li, X., Li, Y., Lei, R., & Sun, S. (2018). The Overview of Multi-person Pose Estimation Method. In *Proceedings of International Conference on Signal And Information Processing, Networking And Computers* (pp. 600-607). Springer.

Li, C. Yan, W. (2021) Braille recognition using deep learning. In *Proceedings of ACM ICCCV* (pp. 30 - 35)

Li, S., Gu, X., Xu, X., Xu, D., Zhang, T., Liu, Z., & Dong, Q. (2021). Detection of concealed cracks from ground penetrating radar images based on deep learning algorithm. *Construction and Building Materials*, *273*, 121949.

Liang, S., Yan, W. (2022) Multilingual speech recognition based on the end-to-end framework. *Springer Multimedia Tools and Applications.*

Lienhart, R., Einfalt, M., & Zecha, D. (2018). Mining automatically estimated poses from video recordings of top athletes. *arXiv preprint arXiv:1804.08944.*

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2117-2125).

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of IEEE International Conference on Computer Vision* (pp. 2980-2988).

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *Proceedings of European Conference on Computer Vision* (pp. 740-755). Springer.

Liu, C., Yan, W. (2021) Gait recognition using deep learning. *Handbook of Research on Multimedia Cyber Security* (pp. 214-226)

Liu, K., Tang, H., He, S., Yu, Q., Xiong, Y., & Wang, N. (2021). Performance validation of YOLO variants for object detection. In *Proceedings of International Conference on Bioinformatics and Intelligent Computing* (pp. 239-243).

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8759-8768).

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In *Proceedings of European Conference on Computer Vision* (pp. 21-37), Springer.

Liu, X., Nguyen, M., Yan, W. (2019) Vehicle-related scene understanding using deep learning. In *Proceedings of ACPR Workshop AAPS.*

Liu, X., Yan, W. (2021) Traffic-light sign recognition using Capsule network. *Springer Multimedia Tools and Applications.*

Liu, Z., Zhu, J., Bu, J., & Chen, C. (2015). A survey of human pose estimation: The body parts parsing based methods. *Journal of Visual Communication and Image Representation*, *32*, 10-19.

Liu, Z., Yan, W., Yang, B. (2018) Image denoising based on a CNN model. In *Proceedings of IEEE ICCAR.*

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3431-3440).

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91-110.

Luo, Z., Nguyen, M., Yan, W. (2021) Sailboat detection based on automated search attention mechanism and deep learning models. In *Proceedings of IEEE IVCNZ.*

Lu, J., Shen, J., Yan, W., Bacic, B. (2017) An empirical study for human behaviour analysis. *International Journal of Digital Crime and Forensics 9 (3), 11-17.*

Lu, J., Nguyen, M., Yan, W. (2018) Pedestrian detection using deep learning. In *Proceedings of IEEE AVSS.*

Lu, J., Nguyen, M., Yan, W. (2020) Comparative evaluations of human behaviour recognition using deep learning. *Handbook of Research on Multimedia Cyber Security* (pp. 176-1894)

Lu, J., Nguyen, M., Yan, W. (2021) Sign language recognition from digital videos using deep learning methods. In *International Symposium on Geometry and Vision.*

Ma, X., Yan, W. (2021) Banknote serial number recognition using deep learning. *Springer Multimedia Tools and Applications.*

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of ICML* (Vol. 30, No. 1, p. 3).

Mazzini, D. (2018). Guided upsampling network for real-time semantic segmentation. *arXiv preprint arXiv:1807.07466*.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of Mathematical Biophysics, 5*(4), 115-133.

Mehtab, S., Yan, W. (2022) Flexible neural network for fast and accurate road scene perception. *Springer Multimedia Tools and Applications.*

Mehtab, S., Yan, W., Narayanan, A. (2021) *3D vehicle detection using cheap LiDARs and RGB images.* In *Proceedings of IEEE IVCNZ 2021*.

Mehtab, S., Yan, W. (2021) FlexiNet: Fast and accurate vehicle detection for autonomous vehicles-2D vehicle detection using deep neural network. In *Proceedings of ACM ICCCV* pp. (43-49)

Le, R., Nguyen, M., Yan, W. (2021) Augmented reality and machine learning incorporation using YOLOv3 and ARKit. *Applied Sciences*.

Li, R., Nguyen, M., Yan, W. (2017) Morse codes enter using finger gesture recognition. In *Proceedings of International Conference on Digital Image Computing: Techniques.*

Liu, X., Yan, W., Kasabov, N. (2021) Vehicle-related scene segmentation using CapsNets. In *Proceedings of IEEE IVCZN.*

Lu, J., Nguyen, M., Yan, W. (2020) Deep learning methods for human behaviour recognition. In *Proceedings of IEEE IVCNZ.*

Misra, D. (2019). Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, *4*, 2.

Moeslund, T. B., & Granum, E. (2001). A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, *81*(3), 231-268.

Newell, A., Huang, Z., & Deng, J. (2017). Associative embedding: End-to-end learning for joint detection and grouping. In *Proceedings of Advances in Neural Information Processing Systems*, *30*.

Newell, A., Yang, K., & Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *European Conference on Computer vision* (pp. 483-499). Springer.

Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of IEEE International Conference on Computer Vision* (pp. 1520-1528).

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Pan, C., Yan, W. (2019) A learning-based positive feedback in salient object detection. In *Proceedings of IEEE IVCNZ.*

Pan, C., Yan, W. (2020) Salient object detection based on perception saturation. *Multimedia Tools and Applications* 79 (27-28), 19925-199445.

Pan, C., Liu, J., Yan, W. Zhou, Y. (2021) Salient object detection based on visual perceptual saturation and two-stream hybrid networks. *IEEE Transactions on Image Processing.*

Papandreou, G., Zhu, T., Kanazawa, N., Toshev, A., Tompson, J., Bregler, C., & Murphy, K. (2017). Towards accurate multi-person pose estimation in the wild. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4903-4911).

Parekh, P., & Patel, A. (2021). Deep learning-based 2D and 3D human pose estimation: A survey. In *Proceedings of Second International Conference on Computing, Communications, and Cyber-Security* (pp. 541-556), Springer.

Patel, M., & Kalani, N. (2021). A survey on pose estimation using deep convolutional neural networks. In *Proceedings of IOP Conference Series: Materials Science and Engineering* (Vol. 1042, No. 1, p. 012008).

Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P. V., & Schiele, B. (2016). DeepCut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4929-4937).

Qin, Z., Yan, W. (2021) Traffic-sign recognition using deep learning. In *Proceedings of International Symposium on Geometry and Vision.*

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434.*

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7263-7271).

Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779-788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of Advances in Neural Information Processing Systems*, *28*, 91-99.

Ren, Y., Nguyen, M., Yan, W. (2018) Real-time recognition of series seven New Zealand banknotes. *IJDCF* 10 (3), 50-66, IGI Global.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 234-241). Springer.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85-117.

Sha, L., Lucey, P., Morgan, S., Pease, D., & Sridharan, S. (2013). Swimmer localization from a moving camera. In *Proceedings of IEEE International Conference on Digital Image Computing: Techniques and Applications (DICTA)* (pp. 1-8).

Shen, Y., Yan, W. (2019) Blind spot monitoring using deep learning. In *Proceedings of IEEE IVCNZ.*

Shen, D., Xin, C., Nguyen, M., Yan, W. (2018) Flame detection using deep learning. In *Proceedings of IEEE ICCAR.*

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Song, Q., Li, S., Bai, Q., Yang, J., Zhang, X., Li, Z., & Duan, Z. (2021). Object Detection Method for Grasping Robot Based on Improved YOLOv5. *Micromachines*, *12*(11), 1273.

Song, C., He, L., Yan, W., Nand, P. (2019) An improved selective facial extraction model for age estimation. In *Proceedings of IEEE IVCNZ.*

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806.*

Sun, K., Geng, Z., Meng, D., Xiao, B., Liu, D., Zhang, Z., & Wang, J. (2020). Bottom-up human pose estimation by ranking heatmap-guided adaptive key point estimates. *arXiv:2006.15480*.

Sun, K., Xiao, B., Liu, D., & Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 5693-5703).

Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE, 105*(12), 2295-2329.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-9).

Thomas, G., Gade, R., Moeslund, T. B., Carr, P., & Hilton, A. (2017). Computer vision for sports: Current applications and research topics. *Computer Vision and Image Understanding, 159*, 3-18.

Thuan, D. (2021). *Evolution of YOLO Algorithm and YOLOv5: The State-of-the-art Object Detection Algorithm*. Bachelor's Thesis, Oulu University of Applied Sciences

Toshev, A., & Szegedy, C. (2014). DeepPose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1653-1660).

Tsumita, T., Shishido, H., Kitahara, I., & Kameda, Y. (2019, March). Swimmer position estimation by lane rectification. In *Proceedings of International Workshop on Advanced Image Technology (IWAIT) 2019* (Vol. 11049).

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International Journal of Computer Vision*, *104*(2), 154-171.

Wang, C. Y., Liao, H. Y. M., Wu, Y. H., Chen, P. Y., Hsieh, J. W., & Yeh, I. H. (2020). CSPNet: A new backbone that can enhance learning capability of CNN. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 390-391).

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., & Xiao, B. (2020). Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, J. Yan, W. (2016) BP-neural network for plate number recognition. *International Journal of Digital Crime and Forensics (IJDCF)* 8 (3), 34-45.

Wang, J., Bacic, B., Yan, W. (2018) An effective method for plate number recognition. *Multimedia Tools and Applications 77 (2), 1679-1692.*

Wang, J., Yan, W. (2020) BP-neural network for plate number recognition. *Deep Learning and Neural Networks: Concepts, Methodologies, Tools and Applications.*

Wang, L., Yan, W. (2021) Tree leaves detection based on deep learning. In *Proceedings of International Symposium on Geometry and Vision.*

Wang, X., Yan, W. (2019) Human gait recognition based on frame-by-frame gait energy images and convolutional long short-term memory. *International Journal of Neural Systems 29 (12), 1-1238*

Wang, X., Yan, W. (2020) Non-local gait feature extraction and human identification. *Springer Multimedia Tools and Applications.*

Wang, X., Feng, S., Yan, W. (2021) Human gait recognition based on self-adaptive hidden Markov model. *IEEE ACM Trans. Comput. Biol. Bioinform. 18(3): 963-972*

Wang, X., Yan, W. (2019) Gait recognition using multichannel convolutional neural networks. *Neural Computing and Applications.*

Wei, S. E., Ramakrishna, V., Kanade, T., & Sheikh, Y. (2016). Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4724-4732).

Woinoski, T., Harell, A., & Bajic, I. V. (2020). Towards automated swimming analytics using deep neural networks. *arXiv:2001.04433.*

Xiang, Y., Yan, W. (2021) Fast-moving coin recognition using deep learning. *Springer Multimedia Tools and Applications.*

Xiao, B. Nguyen, M., Yan, W. (2021) Apple ripeness identification using deep learning models. In *International Symposium on Geometry and Vision.*

Xiao, B., Wu, H., & Wei, Y. (2018). Simple baselines for human pose estimation and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 466-481).

Xin, C., Nguyen, M., Yan, W. (2020) Multiple flames recognition using deep learning. *Handbook of Research on Multimedia Cyber Security* (pp. 296-307)

Xing, J., Yan, W. (2022) The improved framework of traffic sign recognition by using guided image filtering. *Springer Nature Computer Science.*

Xing, J. Yan, W. (2021) Traffic sign recognition using guided image filtering. In *Proceedings of International Symposium on Geometry and Vision.*

Yan, W., Chambers, J., Garhwal., A. (2014) An empirical approach for currency identification. *Multimedia Tools and Applications 74 (7)18 2014*

Yan. W. (2019) *Introduction to Intelligent Surveillance: Surveillance Data Capture, Transmission, and Analytics. Springer*

Yan, W. (2021) *Computational Methods for Deep Learning: Theoretic, Practice and Applications. Springer.*

s

Yang, G., Feng, W., Jin, J., Lei, Q., Li, X., Gui, G., & Wang, W. (2020). Face mask recognition system with YOLOV5 based on image recognition. In *Proceedings of IEEE International Conference on Computer and Communications (ICCC)* (pp. 1398-1404).

Yang, W., Li, S., Ouyang, W., Li, H., & Wang, X. (2017). Learning feature pyramids for human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1281-1290).

Yang, Y., & Ramanan, D. (2011). Articulated pose estimation with flexible mixtures-of-parts. In *Proceedings of CVPR* (pp. 1385-1392). IEEE.

Huang, Y., Sun, S., Duan, X., Chen, Z. (2016). A study on deep neural networks framework. In *Proceedings of IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)* (pp. 1519-1522).

Yu, Z., Yan, W. (2020) Human action recognition using deep learning methods. In *Proceedings of IEEE IVCNZ.*

Zaidi, S. S. A., Ansari, M. S., Aslam, A., Kanwal, N., Asghar, M., & Lee, B. (2021). A survey of modern deep learning based object detection models. *arXiv preprint arXiv:2104.11892*.

Zecha, D., Eggert, C., & Lienhart, R. (2017). Pose estimation for deriving kinematic parameters of competitive swimmers. *Electronic Imaging*, *2017*(16), 21-29.

Zecha, D., Einfalt, M., Eggert, C., & Lienhart, R. (2018). Kinematic pose rectification for performance analysis and retrieval in sports. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 1791-1799).

Zecha, D., Greif, T., & Lienhart, R. (2012). Swimmer detection and pose estimation for continuous stroke-rate determination. In *Proceedings of Multimedia on Mobile Devices 2012; and Multimedia Content Access: Algorithms and Systems VI* (Vol. 8304, p. 830410).

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Proceedings of European Conference on Computer Vision* (pp. 818-833). Springer.

Zhang, F., Zhu, X., & Wang, C. (2021). Single Person Pose Estimation: A Survey. *arXiv preprint arXiv:2109.10056*.

Zhang, F., Zhu, X., Dai, H., Ye, M., & Zhu, C. (2020). Distribution-aware coordinate representation for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7093-7102).

Zhang, L. Yan, W. (2020) Deep learning methods for virus identification from digital images. In Proceedings of IEEE IVCNZ.

Zhang, Q., Yan, W. (2018) Currency recognition using deep learning. In *Proceedings of IEEE AVSS.*

Zhang, Q., Yan, W., Kankanhalli, M. (2019) Overview of currency recognition using deep learning. *Springer Journal of Banking and Financial Technology* 3 (1), 59–69.

Zhao, K., Yan, W. (2021) Fruit detection from digital images using CenterNet. In *Proceedings of International Symposium on Geometry and Vision.*

Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212-3232.

Zheng, C., Wu, W., Yang, T., Zhu, S., Chen, C., Liu, R., ... & Shah, M. (2020). Deep learning-based human pose estimation: A survey. *arXiv preprint arXiv:2012.13392*.

Zheng, K., Yan, W., Nand, P. (2017) Video dynamics detection using deep neural networks. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-IoU loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, 07 (pp. 12993-13000).

Zhou, F., Zhao, H., & Nie, Z. (2021). Safety helmet detection based on YOLOv5. In *Proceedings of IEEE International Conference on Power Electronics, Computer Applications (ICPECA)* (pp. 6-11). IEEE.

Zhu, Y., Yan, W. (2022) Traffic sign recognition based on deep learning. *Springer Multimedia Tools and Applications.*

Zitnick, C. L., & Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *Proceedings of European Conference on Computer Vision* (pp. 391-405), Springer.

Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*.