A human pose recognition method based on the self-supervised learning

10^{th} semester project

Group 1033

AALBORG UNIVERSITY

STUDENT REPORT

Aalborg University
Control and Automation

Department of Automation and Control
Aalborg University
http://www.aau.dk/

AALBORG UNIVERSITY
STUDENT REPORT

Title:
A human pose recognition method based on self-supervised learning

Theme:
Robot Control
Machine Learning

Project Period:
Spring 2022

Project Group:
Group 1033

Participant(s):

Yuheng Wang

Supervisor(s):
Zheng-Hua Tan

Copies:

Page Numbers: 34

Date of Completion:
June 1, 2022

Abstract:

Human pose estimation and Self-Supervised Learning (SSL) are both popular topics in the Computer Vision (CV) domain. In this project, a classical Self-Supervised Learning (SSL) pre-training method: The jigsaw puzzle is designed and implemented to improve the recognition accuracy of human pose. The backbone network Convolutional Pose Machine (CPM) will be first trained based on the pre-training task. Then some layers will be transferred to the downstream task: human pose estimation. After some experiments to transfer some layers from the pre-training task to the downstream task, the SSL method increases the training effect when the dataset's amount is small. However, when we enlarge the dataset, the Supervised learning's training effect is still better than the SSL.

Contents

1 Introduction

1.1 Background information

This project is inspired by the author's last semester project[1]. A control strategy for an exoskeleton robot is developed in [1]. This exoskeleton needs to mimic human behavior, especially for body movement. Therefore, it is necessary to get an accurate human pose. Accordingly, this task is planned to solve during this semester. In this semester's project, a human pose recognition method based on the Computer Vision (CV) is developed. Since the Self-Supervised Learning (SSL) is a popular topic recently and has shown improvement in many fields[2][3], this project will mainly be based on it to improve the estimation accuracy.

1.2 Human pose recognition

Human pose recognition can be applied in many fields, such as gaming, human-machine interaction, health care, and even security[4]. The method to recognize human pose can be divided into two categories: sensor-based and CV based. For the sensor-based way, it needs people wearing some sensors, like Inertial Measurement Unit (IMU)[5]. However, this method's implementation cost is high, and the recognition accuracy is easily influenced by the communication between the sensors and the computer. Compared with the sensor-based way, CV based doesn't need people wearing additional equipment. Besides, there isn't too much cost after training the neural network. Therefore, this project will mainly focus on the CV based method.

1.2.1 Human pose estimation based on Computer Vision

The CV based human pose estimation has developed over many years. Its goal is to recognize key points of humans on figures or video frames, like elbows, knees, head, neck, etc. This task is a popular Artificial Intelligence (AI) task and it has developed for many years. The benchmark of it is shown in Figure 1.1:

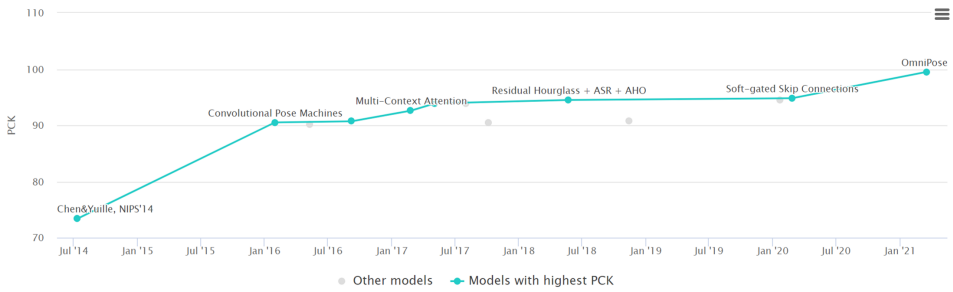

Figure 1.1: The development of human pose estimation method base on CV[6]

In Figure 1.1, the vertical axis is the Percentage of Correct Keypoints (PCK), which represents the percentage of detections that fall within a normalized distance of the ground truth. Here, this result is based on the Leeds Sports Pose (LSP) dataset, which contains 2000 figures of human pose in sports activities[7].

Since the year 2014, most state-of-the-art human pose estimation methods are based on CNN. Currently, the OmniPose method with best performance can achieve around 100 PCK[8].

However, considering the network complexity and the available implementation, this project mainly focse on the method called Convolutional Pose Machine (CPM)[9]. More detailed information about CPM can be found in Section 2.2.

1.3 Self-supervised learning

1.3.1 The categories of Machine Learning

Generally speaking, machine learning can be divided into three categories: Supervised learning, Unsupervised learning, and Self-Supervised Learning (SSL), which is shown in Figure 1.2.

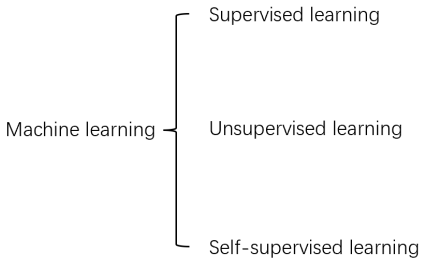

Figure 1.2: The categories of machine learning

1.3.2 Supervised learning and Unsupervised learning

The main difference between these three methods is whether they need labels and how they use them. For Unsupervised learning, it can extract the information from a dataset without labels[10]. For example, K-means, K Nearest Neighborhood (KNN), Gaussian Mixture Model (GMM) methods are belong to this category. However, Unsupervised learning can't achieve too complicated tasks, like human pose estimation. It is mainly used in data preprocessing.

Supervised learning requires data with labels to train a model[11]. In many cases, the more annotated data we have, the better performance the model will be. However, annotating labels for data is always a time-consuming task. For the human pose estimation task, the researchers need to recognize every joint by eyes and annotate it among thousands of figures. To save labor time and improve the recognition accuracy, the Self-Supervised Learning (SSL) is a new method to provide this shortcomings.

1.3.3 Self Supervised learning's concept

SSL is a learning method which is sort of "human-like"[12]. When people learn something, the most efficient way is not directly learning the final objective. If they spend some time on some relative tasks to get familiar with this topic, this will improve the learning effect. Learning a new language is a good example. If someone watches movies and listens to songs in this language before learning its grammar, this can help them learn this new language quickly. SSL's idea is based on this.

For SSL, it contains two tasks: a pre-training task and a downstream task. In the pre-training task, the researchers designed the training goal, which is set to let the network "get familiar" with the final training task. For pre-training task, it belongs to Supervised learning. However, the labels of its training dataset are not annotated by humans but by code or other methods. This makes it possible to generate plenty of data with labels. When pre-training is converged, some layers in this network will be transferred to the downstream task. The downstream task also belongs to Supervised learning. Its training object is same as the origin training object. With trained layers from the pre-training task, the downstream task can achieve the training goal only with a small amount of labeled data.

For some tasks, it is hard to get too much dataset with labels in the CV domain, like the human pose recognition task, SSL is an excellent way to improve the training effect. There are different pre-training tasks. For example, it can be recovering the masked data figures[13][14], recovering the color of a figure[15], shuffling the figure as a random combination of many patches[16], etc. For these recovering tasks, the original figures without mask or color information or shuffled are labels, which don't require human annotation. This project will use a pre-training task called Jigsaw puzzles. It will be shown in Chapter 3.

1.4 Problem formulation

In general, the goal of this project can be described as:

"Design a human pose estimation method based on Computer Vision (CV), then use the Self-Supervised Learning (SSL) to improve the recognize accuracy"

In the next chapter, it will introduce the entire architecture of the project and the backbone neural network, CPM, used in this project.

2 Architecture

This chapter introduces the overall architecture and the backbone neural network of this project.

2.1 Overall architecture

As mentioned in Section 1.3, the entire training process contains two tasks: the pre-training task and the downstream task. The architecture of this project is shown in Figure 2.1:

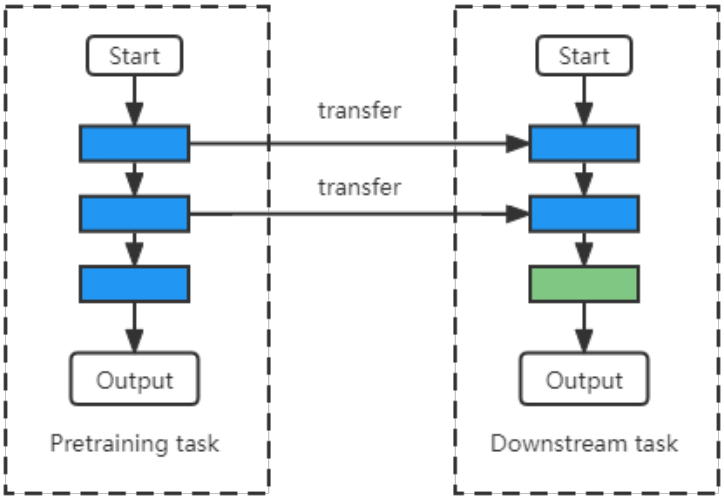

Figure 2.1: The architecture of this project

After training the pre-training task, some layers of it will be transferred to the downstream task (Blue blocks in the Figure 2.1). These layers' parameters will not change during the downstream task. Here, the pre-training task's network and downstream task's network are almost identical.

As mentioned before, the backbone network used in this project is called CPM, which will be introduced in the next section.

2.2 Convolutional pose machine

The Convolutional Pose Machine (CPM) method is firstly published in the The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) in 2016[9]. The network structure is shown below:

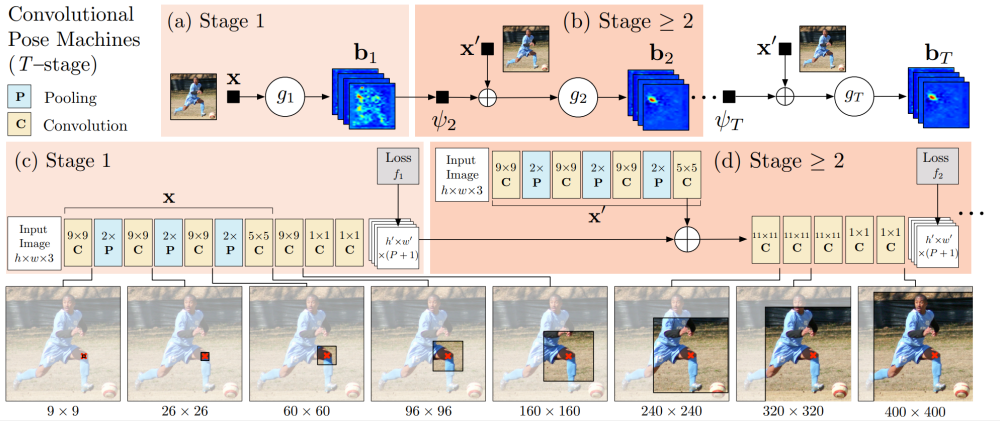

Figure 2.2: The structure of CPM[9]

In Figure 2.2, the entire architecture network can be divided into two types: "Stage 1" and "Stage ≥ 2". For "Stage 1", the input figures are cropped into 368×368. After some pooling and convolution layers, the output is a set of figures called belief maps. Each set of belief maps contains $k + 1$ 46×46 figures. Here, k represents the number of key human body points in the dataset. And "1" represent the background information. Each key point of the human body has a belief map. The network calculates the loss between these outputs and labels, then uses backpropagation to update the parameters of previous layers.

For "Stages ≥ 2", it combines the belief map output from the previous stage (ϕ_T in Figure 2.2) and the processed original input figure. After some convolution and pooling stages, the loss can be calculated between the label's heatmap and the belief maps generated in this stage. Then the belief maps will output to the next stage as input.

In CPM, the labels are transferred into k+1 46×46 heatmaps. These heatmaps are generated based on Gaussian distribution. For example, if the coordinate of left elbow is (x_e, y_e), then the heatmap for this joint is a Gaussian distribution whose center is located at (x_e, y_e). The Probability Density Function (PDF) of a 2-D Gaussian distribution is:

$$f(X) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp[-\frac{1}{2}(X-u)^T\Sigma^{-1}(X-u)], X = (x_1, x_2 ... x_n) \tag{2.1}$$

Where:

d : Dimension of Gaussian distribution. $d = 2$ in this case
Σ : Co-variance matrix. Since the two variables are independent, it is only related to one parameter σ
X : Variable of this distribution. Dimension is two in this case
U : Center of this Gaussian distribution

For this Gaussian distribution, its co-variance parameter σ is set by researchers. An example of heatmaps is shown in Figure 2.3:

Figure 2.3: An example of heatmaps

The loss function used in CPM is Mean Squared Error (MSE). Its equation is:

$$Loss_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.2}$$

Where:

N : Total amount of pixels.
i : Index of pixels.
y_i : Value of i^{th} pixel in the label heatmap.
$\hat{y}_i$: Value of i^{th} pixel for the output of the neural network.

For example, Equation 2.3 shows the MSE for two 3×3 figures shown in Figure 2.4.

$$\begin{aligned} Loss = \frac{1}{9}((0.1-0.5)^2 + (0.8-0.1)^2 + (0.4-0.5)^2 + (0.9-0.4)^2 + (0.1-0.5)^2 \\ + (0.5-0.9)^2 + (0.5-0.5)^2 + (0.3-0.9)^2 + (0.4-0.3)^2) \\ = 0.178 \end{aligned} \tag{2.3}$$

0.1	0.8	0.4
0.9	0.1	0.5
0.5	0.3	0.4

0.5	0.1	0.5
0.4	0.5	0.9
0.5	0.9	0.3

Figure 2.4: Two 3×3 figures

During the training process, all the figures and their labels are randomly rotated (≤ 40 *degrees*) and resized (368×368) to improve the training effect, see Figure 2.5.

Figure 2.5: Rotated and resized figure

2.3 Experiment for supervised learning

An experiment for supervised learning has been implemented to get a benchmark of the CPM's performance. The code is based on an open-source Github repository: "https://github.com/namedBen/Convolutional-Pose-Machines-Pytorch". Since the configuration in this Github repository's software configuration is different from the configuration introduced in Appendix A, some detail commands in Python are changed.

In this experiment, the CPM is set to have six stages. That's because from Figure 2.6, when the amounts of stages are larger than five, the performance of this network will not improve too much.

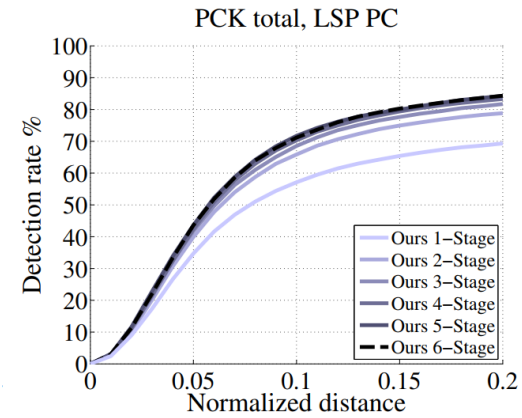

Figure 2.6: The performance for different amount of stages[9]

The co-variance co-efficient σ for the heatmaps of labels are set as three.

For this Supervised training, this project use LSP dataset to train. For LSP, it is formed by 2000 figures which describe human pose during sports activities. This task divides the LSP (contains 2000 figures) into three parts: Training dataset (contains 1600 figures), validation dataset(contains 200 figures) and test dataset (contains 200 figures). The LSP dataset contains the location information of 14 key points of the human body. These 14 key points are: right ankle, right knee, right hip, left hip, left knee, left ankle, right wrist, right elbow, right shoulder, left shoulder, left elbow, left wrist, neck, and head top. Therefore, the k variable is set as 14.

Training 75 epochs with the settings introduced above, an example of recognition is shown in Figure 2.7:

Figure 2.7: An example of human pose estimation

This project will use the best (lowest) loss as the criteria to evaluate the training effect. The reason is: During this experiment, the PCK and loss are found negatively correlated. That means the higher value of PCK is, the lower loss for the prediction. Besides, when the recognition accuracy is not high, the PCK criteria can not show the real training effect.

The best loss for Supervised learning is 320 (This value is amplified with a fixed gain for evaluation). This value will be used as the benchmark to evaluate the training effect of SSL. Besides, the general settings between SSL and Supervised learning are set as same.(Same dataset, same devices, same batch size, same learning rate, etc.) The comparison between the result of these two methods will be introduced in Chapter 4.

2.4 Sub-conclusion

In this chapter, the overall architecture of the entire project and the backbone network, CPM, are introduced. Besides, an experiment based on supervised learning for human pose estimation is also completed to get a benchmark for the following work.

In the next chapter, this report will introduce the pre-training task designed for this project.

3 Pre-training task

In Section 1.3, the idea of SSL has been introduced. This project will introduce a pre-training task inspired by Jigsaw Puzzles in the following sections.

3.1 Jigsaw puzzles

Jigsaw Puzzles were introduced in 1760 by John Spilsbury. In the beginning, it helps children learn geography[16]. During a Jigsaw puzzle game, the goal is to re-construct a figure with shuffled patches. An example of Jigsaw puzzle is shown below:

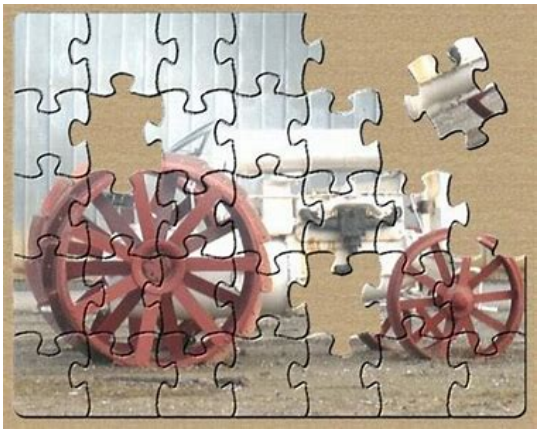

Figure 3.1: An example of jigsaw puzzles

During a Jigsaw puzzle game, the player needs to recognize the pattern of each patch, then combine them in the correct order. This progress inspires a method of SSL based on Jigsaw puzzles.

In this SSL method, the pre-training task is to let the neural network recover the shuffled figures with nine patches, see Figure 3.2. Even though this pre-training task can not achieve the final goal of recognizing the human pose, the neural network can learn some features of the patches like some edges or points. If some layers are transferred to downstream tasks to recognize the human pose, the features extracted from these layers will improve the performance. This progress is very similar to the

example of people learning new languages mentioned before. In this case, the pre-training task, recovering the Jigsaw, is like watching movies or listening to songs in the learning progress.

Figure 3.2: The origin and shuffled figures[17]

3.2 Labels generation

For this pre-training task, the figures are shuffled based on labels generated randomly. The labels are permutations combined with nine numbers from one to nine. For example, the label for the shuffled figures in Figure 3.2 is:

$$[8, 5, 1, 9, 3, 7, 6, 2, 4] \tag{3.1}$$

The spatial position this numbers mean is shown in Figure 3.3. This sequence in Equation 3.1 means: The patch in the position "1" of the shuffled figure is the patch in the position "8" of the original figure. The patch in position "2" of the shuffled figure is the patch in position "5" of the original figure. The patch in position "3" of the shuffled figure is the patch in position "1" of the original figure. The patch in position "4" of the shuffled figure is the patch in position "9" of the original figure. The patch in the position "5" of the shuffled figure is the patch in the position "3" of the original figure. The patch in the position "6" of the shuffled figure is the patch in the position "7" of the original figure. The patch in the position "7" of the shuffled figure is the patch in the position "6" of the original figure. The patch in position "8" of the shuffled figure is the patch in position "2" of the original figure. The patch in the position "9" of the shuffled figure is the patch in the position "4" of the original figure.

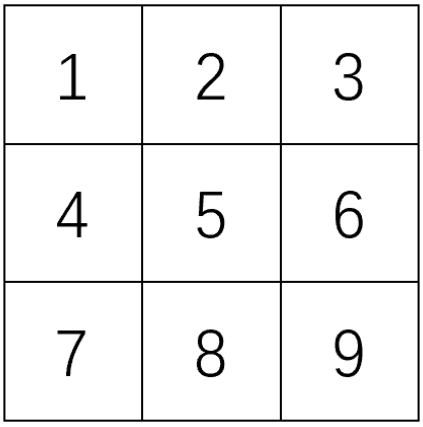

Figure 3.3: The origin and shuffled figures

Here, the label is guaranteed that every patch changes its position after being shuffled to improve the training effect. For example, "1" will never appear at index 1 of the label sequence.

3.3 Implementation

3.3.1 General settings

In the pre-training task, the backbone network is still Convolutional Pose Machine (CPM). The size of input images is still cropped into 368×368 to fulfill the requirement of CPM.

This pre-training task use mpii and LSP extend as the dataset. For the mpii, it contains more than 24000 figures (This project only uses 24000 of them) of people in different activities. For the LSP extend, it is similar to LSP, but the amount of figures is 10000.

When creating Jigsaw Puzzles, each figure is cropped into the square (only keep the middle part of the figure), and it will be divided into nine patches. Then these nine patches will be combined by the algorithm according to the label information. For each figure, this project shuffled it according to ten different label sequence. So there are 240000 figures generated by mpii and 100000 generated by LSP. During this combination, each patch will have a five pixels interval with other patches, see Figure 3.2.

During a training process, the choice of loss function will influence the training effect. This project has tried different loss functions, introduced in the following section.

3.3.2 Cross entropy based method

Cross entropy is a widely used loss function for the classification task. The equation of it is:

$$L_{crossentropy} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{K} y_{ic} log(p_{ic}) \tag{3.2}$$

Where:

N	: Total amount of samples.
i	: Index of samples.
K	: Total amount of class.
c	: Index of class.
y_{ic}	: Probability for i_{th} sample belongs to c_{th} class from the label.
p_{ic}	: Probability for i_{th} sample belongs to c_{th} class from the prediction.

For example, if we have three sets of samples and need to classify them into three classes. The predicted probability values of these samples are:

	Class 1	Class 2	Class 3
Sample 1	0.2	0.3	0.5
Sample 2	0.3	0.2	0.5
Sample 3	0.4	0.4	0.2

Table 3.1: An example of one hot format

The labels for these samples are:

	Class 1	Class 2	Class 3
Label 1	0	1	0
Label 2	1	0	0
Label 3	0	0	1

Table 3.2: An example of one hot format

Substitute the values in Table 3.1 and Table 3.2 in the Cross entropy equation, the Equation 3.2 becomes:

$$\begin{aligned} L = &-\frac{1}{3}((0 \times log(0.2) + 1 \times log(0.3) + 0 \times log(0.5)) \\ &+ (1 \times log(0.3) + 0 \times log(0.2) + 0 \times log(0.5)) \\ &+ (0 \times log(0.4) + 0 \times log(0.4) + 1 \times log(0.2))) \\ &= 1.3391 \end{aligned} \tag{3.3}$$

As shown in Table 3.2, the label needs to be transferred into "one-hot" format to use this loss function. In the Jigsaw puzzle problem, the label in the Equation 3.1 in one hot format is:

0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0

Table 3.3: An example of one hot format

In Table 3.3, the row of this table represents the index of the position for the shuffled figure. The column of this table represents which patch in the original figure will locate in this shuffled figure.

To implement this method in the pre-training task, some layers and settings of the CPM network need to be changed.

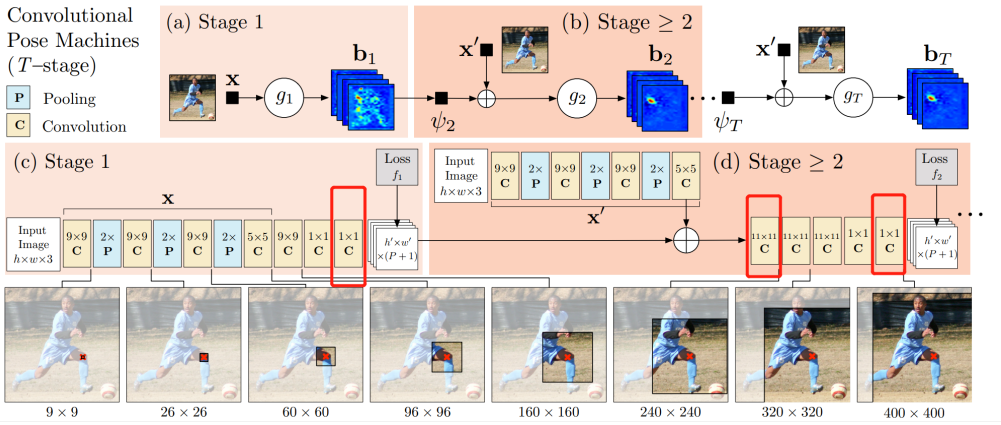

Figure 3.4: The changed blocks for Cross entropy based method

In Figure 3.4, the first and third red block's output channel's amount is changed to one instead of three. The second red block's input channel's amount is changed from 32+k to 33. Besides, a pooling layer is added after the first and third red block to output a 9×9 tensor.

However, after training around three epochs(only use LSP extend dataset, 100000 figures in total), the loss is not decreased, and the recognition accuracy is around 13%, which is almost the same as the accuracy of random sequence. Therefore, another method will be introduced in the next section.

3.3.3 Heatmap based method

As mentioned in Section 2.2, the origin CPM utilize heatmap and MSE loss function for the human pose estimation task[9]. Inspired by it, this Jigsaw puzzle task will also use heatmap and MSE loss function to improve the training effect.

The heatmap generates method needs to be re-design for the Jigsaw puzzles task. The center for the original heatmap is located at the co-ordinate of key points of humans. The center of the heatmap for Jigsaw puzzles only has nine potential locations. For a 46×46 figure, the relation between the coordinates and the labels corresponding to them is:

Label	Coordinate
1	$(8, 8)$
2	$(8, 24)$
3	$(8, 39)$
4	$(24, 8)$
5	$(24, 24)$
6	$(24, 39)$
7	$(39, 8)$
8	$(39, 24)$
9	$(39, 39)$

Table 3.4: The relation between the coordinate of heatmap's center and label

Besides, the co-variance co-efficient σ is also changed from three to five.

Here, [18] is found that its idea is very similar to ours. However, even though they published this paper earlier than this project, this project's idea is proposed independently. Moreover, [18] doesn't test CPM as backbone network.

Same as Subsection 3.3.2, some layers in the CPM network need to be changed since the input and output channels' amounts are not the same. In Figure 3.4, the first red block's output channel quantity needs to be changed from k to nine. The second red block's amount of input channel needs to be changed from $k + 1$ to ten. The third block's amount of output channel also needs to be changed from k to nine.

After training around five epochs (Mpii and LSP dataset. 340000 figures in total), The loss converges. Test the network on the LSP dataset (2000 figures), and the accuracy is around 63.5%. This accuracy means that this network and algorithm can predict the correct position of five to six patches for a shuffled figure with nine patches. For figures with more features, the recognition accuracy is higher than 63.5%. For some figures whose patches are very similar to each other, like the middle row of the Figure 3.2, the accuracy is around 40%.

When it achieves the highest recognition accuracy, the parameters for the model are saved. As mentioned in Section 2.1, some layers of this network will be transferred to the network of downstream task: Human pose estimation.

3.4 Sub-conclusion

In this chapter, a pre-training task: Jigsaw puzzles, is introduced. This task still uses CPM as the network to transfer more layers to the downstream task. A widely used loss function: Cross entropy, has been implemented to complete this task. However, the result is not satisfactory. Another method based on heatmap has a good effect. Using this network, the accuracy of recognizing the position of shuffled patches is more than 60%.

In the following chapter, the training strategy for the downstream task will be introduced.

4 Downstream task

As mentioned in Section 2.1, some layers of the pre-training task will transfer to the downstream task. However, a tricky question is which layers need to be moved and how to train the downstream task with these parameters. The following sections will answer these questions.

4.1 Overall settings

The dataset is set to be the same: 2000 figures (1600 for training, 200 for validation, 200 for testing) from LSP dataset to compare the effect between the Supervised Learning and SSL. Besides, the CPM network for the downstream task still contains six stages. Recall the benchmark mentioned in Section 2.3, the benchmark for Supervised learning on the same dataset is shown in Table 4.1

Method	Converge epochs	Minimum loss in the test dataset
Supervised learning	76	317

Table 4.1: The benchmark for CPM network in the Supervised learning

The following sections will introduce different ways to transfer layers. The "early stop" method was implemented to save the training time. If the loss in the validation dataset didn't decrease for 50 epochs, the algorithm would think the training has already converged and stop the training.

4.2 Transfer as much layers as possible

Recall the Figure 3.4. Because the amount of input and output channels are different between the pre-training and downstream tasks, the last layer in stage one and two layers in red blocks in the following stages must be changed. Since the CPM in this project contains six stages, so there are 11 layers that must be changed.

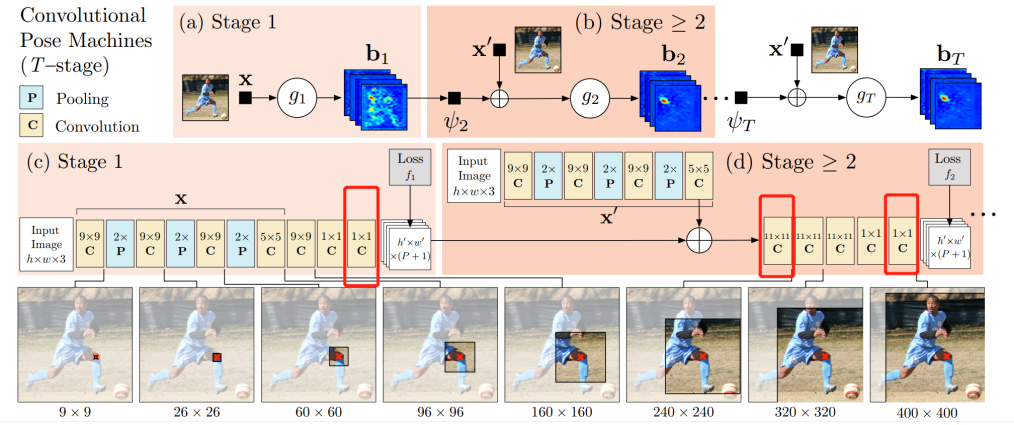

Figure 4.1: The changed blocks for Cross entropy based method

In this section, this downstream task will use all layers from the pre-training task except for the 11 blocks that must be changed. There are two ways to use these layers: Fix the parameters and fine-tuning the parameters. The fixed parameters are easy to understand: The parameters from the pre-training task will not training and updated during the downstream task. The fine-tuning means: The parameters from the pre-training task will be used as the initial value of the downstream task. Then these parameters will update together with other parameters in the downstream task. After training some epochs, the results for testing are listed below:

Method	Converge epochs	Minimum loss in the test dataset
Fixed parameter	150	480
Fine-tuning parameter	106	336

Table 4.2: The result for method one in downstream task

From Table 4.2, the best result is not as good as the Supervised learning. However, an interesting finding is: The minimum loss appears at the end of stage one instead of stage six. This phenomenon appears for both "fixed" and "fine-tuning" situations. Therefore, the next attempt only uses the first stage's layers from the pre-training stage.

4.3 Transfer the layers from the first stage

In this method, the layers from pre-training is shown in the red block of Figure 4.2.

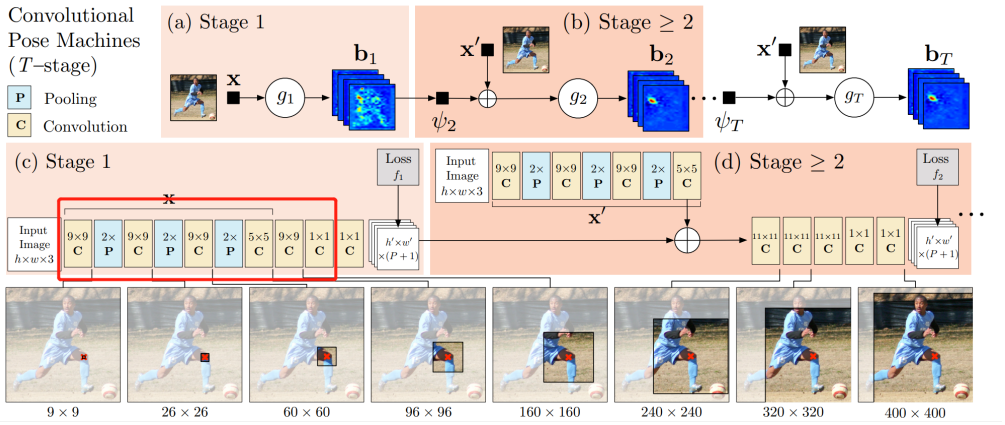

Figure 4.2: The changed blocks in the first stage

The result is shown in Table 4.3

Method	Converge epochs	Minimum loss in the test dataset
Fixed parameter	94	338
Fine-tuning parameter	128	320

Table 4.3: The result for method two in downstream task

In the Table 4.3, the loss of "Fine-tuning" is quite close to the minimum loss of the Supervised learning. However, since the parameters from the pre-training task in this case are updating together with others, these result can not show that SSL has improved the recognize accuracy. Therefore, another method for transferring layers is tested.

4.4 Transfer the layers from the first stage and middle stage

In this method, the transferred layers from pre-training is shown in the red blocks of Figure 4.3.

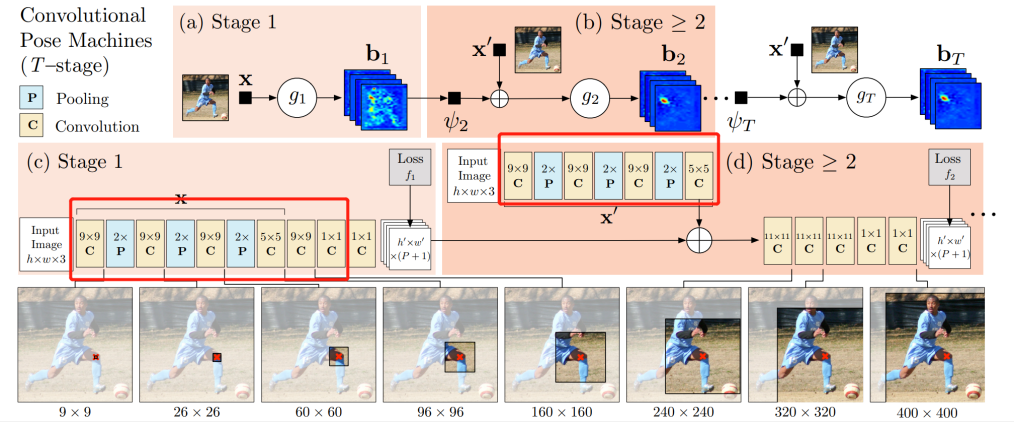

Figure 4.3: The changed blocks for the third method

The result of this method is shown in Table 4.4:

Method	Converge epochs	Minimum loss in the test dataset
Fixed parameter	53	460
Fine-tuning parameter	48	380

Table 4.4: The result for method two in downstream task

In this situation, the minimum loss doesn't always appear in the last stage but in the fourth or fifth stage in this method.

Compare the result in Table 4.2, Table 4.3, and the Table 4.4 with the result in Table 4.1, both of them are not better than the Supervised learning.

Since the SSL shows advantages when the amount of dataset is small, this project also tests the training effect when the dataset for SSL and Supervised learning is reduced. The result will be shown in the next section.

4.5 The downstream task on the reduced training set

In this experiment, the figures amount in the training set is reduced from 1600 to 200. After repeating the previous experiments, the method that substitutes the first and middle stages can improve the training effect. In other words, when the training set is reduced to 200, if we replace the first and middle stage from the pre-training task, the test result for this method is better than Supervised learning. However, in this situation, both Supervised learning and SSL 's training effect are very low. Both of them can't recognize the key point of the human body.

4.6 Sub-conclusion

In this chapter, several knowledge transfer methods for SSL have been implemented. However, both of them can't improve the training effect compared to Supervised learning. The reason behind it will be discussed in the Chapter 5.

5 Discussion

This chapter will discuss some interesting findings and explanations for some strange phenomena during the design and implementation process.

5.1 Convolutional pose machine

As mentioned in Chapter 1, there are plenty of network structures for human pose estimation. Except for the reason that there is available open-source code to implement, another reason is that CPM's multi-stage structure might bring some benefits for the knowledge transfer. If different stages extract different features from the pre-training task, then we can transfer the stage based on our demands.

However, this multi-stage structure also brings some problems. Generally speaking, it is easy for a CNN to over-fitting if it contains too many layers. In this project, the six stages CPM structure causes this problem. This might influence the training effect in the downstream task. Therefore, if the amount of stages used in the pre-training task is reduced from six to three or four, it might improve the training effect.

5.2 Pre-training task

At the beginning of this project, the Jigsaw puzzle is not the only SSL option. Many methods like MAe[13], Data2vec[14], and Moco[2] have been considered. However, some limitations caused these methods are not implemented. These limitations can be divided into the following categories:

- Hardware limitations
- Structure limitations

The method SimCLR[3] and Moco[2] are limited by the hardware. Both of them belong to contrastive learning. They need sizeable graphic memory to provide a large batch size. Especially for the SimCLR method, it needs hundreds GBs of graphic memory and plenty of graphic cards, even Tensor Processing Unit (TPU). The hardware in Appendix A can not satisfy this requirement.

For the structure limitation, it is mainly refer to the Transformer structure[19]. Many state-of-art SSL methods are based on the Transformer structure, like MAe[13], Data2vec[14]. The structure for MAe is shown in Figure 5.1

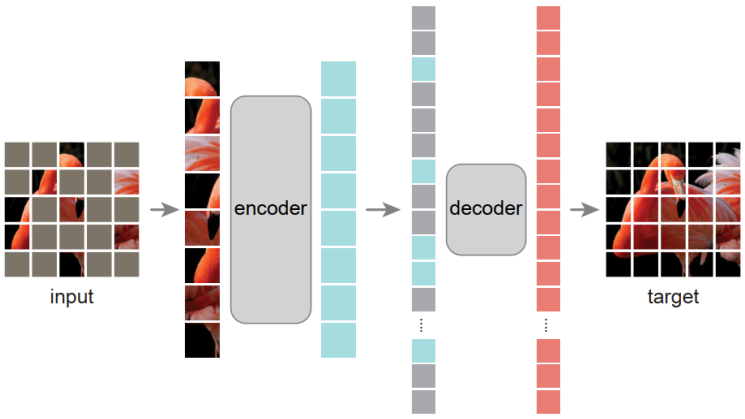

Figure 5.1: An example for the Transformer structure[13]

In this structure, the input of the network is not an entire image but a sequence of patches[20]. However, for the human pose estimation task, this structure might lose the spatial information of the human body. Therefore, all of the methods based on this structure are not implemented. Since one of the advantages of the Jigsaw puzzle is that it can learn the spatial information of a figure, that is also why this project picks it as the pre-training task.

In Chapter 3, the figures are cropped into the square. During the crop process, the human is placed at the center of the square to let the algorithm learn more human features. As mentioned before, the figures' amount in the Mpii dataset is more than 24000, but this project only utilize 24000 of them. These discarded figures include: human's location is placed at the margin of a figure (see Figure 5.2), the edge between human and background is not clear(see Figure 5.3), etc.

Figure 5.2: One discard figure from Mpii dataset[17]

Figure 5.3: One discard figure from Mpii dataset[17]

As mentioned in the Chapter 4, the result is not as good as the Supervised learning. One reason for the pre-training task might be that: Jigsaw puzzle is not the state-of-art SSL method (published in 2014). Therefore it can not learn too many features of a figure.

5.3 Downstream task

During the knowledge transfer part in the SSL, there is no standard question about how to transfer layers. Among the methods tried in this project, the Fine-tuning parameters' result is better than the Fixed parameters. Comparing these three methods shows that the fewer layers transferred, the better the training effect. This might be because the pre-training task is unsuitable for the human pose estimation task.

For the result of the downstream task, most of the best recognition results do not appear at the sixth stage. That might also be caused by the over-fitting problem in a deep neural network.

5.4 Future work

Since the result shown in this report is not satisfactory, to improve the recognition accuracy, some ways can be done in the future. For the backbone network, maybe some commonly used networks like ResNet[21] is a better choice.

Besides, maybe other SSL methods which based on the masked figure, the prediction of video frames, and contrastive learning might can improve the training effect. Except for this, enlarge the training dataset is also a option. In the future, maybe other human related dataset like FLIC can be used in this project.

6 Conclusion

As mentioned in the Chapter 1, the goal of this project is:

"Design a human pose estimation method based on Computer Vision (CV), then use the Self-Supervised Learning (SSL) to improve the recognize accuracy"

However, after one semester's work, the SSL's result is not as good as the Supervised learning when the dataset contains 1600 figures. If the amount of the training set is reduced to 200, the SSL's effect is better than the Supervised learning. However, both of them can not recognize an accurate human pose.

In general, this project designed and implemented a SSL pre-training method called Jigsaw puzzle. Besides, several experiments on knowledge transfer between the pre-training and downstream tasks have been done. Even if this project's goal is not achieved, the student has learned the entire deep learning procedure and lots of knowledge.

Bibliography

[1] Yuheng Wang. *The control strategy for a nonlinear joint of an exoskeleton.* AAU Student Report. 2021.

[2] Kaiming He et al. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* June 2020.

[3] Ting Chen et al. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning.* PMLR. 2020, pp. 1597–1607.

[4] Jamie Shotton et al. "Real-time human pose recognition in parts from single depth images". In: *CVPR 2011.* 2011, pp. 1297–1304. DOI: 10.1109/CVPR.2011.5995316.

[5] Yinghao Huang et al. "Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time". In: *ACM Trans. Graph.* 37.6 (Dec. 2018). ISSN: 0730-0301. DOI: 10.1145/3272127.3275108. URL: https://doi.org/10.1145/3272127.3275108.

[6] https://paperswithcode.com/sota/pose-estimation-on-leeds-sports-poses. Accessed April 20, 2022.

[7] Sam Johnson and Mark Everingham. "Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation". In: *Proceedings of the British Machine Vision Conference.* doi:10.5244/C.24.12. 2010.

[8] Bruno Artacho and Andreas E. Savakis. "OmniPose: A Multi-Scale Framework for Multi-Person Pose Estimation". In: *CoRR* abs/2103.10180 (2021). arXiv: 2103.10180. URL: https://arxiv.org/abs/2103.10180.

[9] Shih-En Wei et al. "Convolutional Pose Machines". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* June 2016.

[10] Zoubin Ghahramani. "Unsupervised Learning". In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures.* Ed. by Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 72–112. ISBN: 978-3-540-28650-9. DOI:

10.1007/978-3-540-28650-9_5. URL: https://doi.org/10.1007/978-3-540-28650-9_5.

[11] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. "Supervised Learning". In: *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*. Ed. by Matthieu Cord and Pádraig Cunningham. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–49. ISBN: 978-3-540-75171-7. DOI: 10.1007/978-3-540-75171-7_2. URL: https://doi.org/10.1007/978-3-540-75171-7_2.

[12] Emin Orhan, Vaibhav Gupta, and Brenden M Lake. "Self-supervised learning through the eyes of a child". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 9960–9971. URL: https://proceedings.neurips.cc/paper/2020/file/7183145a2a3e0ce2b68cd3735186b1d5-Paper.pdf.

[13] Kaiming He et al. "Masked Autoencoders Are Scalable Vision Learners". In: *CoRR* abs/2111.06377 (2021). arXiv: 2111.06377. URL: https://arxiv.org/abs/2111.06377.

[14] Alexei Baevski et al. "data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language". In: *CoRR* abs/2202.03555 (2022). arXiv: 2202.03555. URL: https://arxiv.org/abs/2202.03555.

[15] Carl Vondrick et al. "Tracking Emerges by Colorizing Videos". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[16] Mehdi Noroozi and Paolo Favaro. "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 69–84. ISBN: 978-3-319-46466-4.

[17] Mykhaylo Andriluka, Leonid Pishchulin an Peter Gehler, and Bernt Schiele. "2D Human Pose Estimation: New Benchmark and State of the Art Analysis". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[18] Kun Zhang et al. "Learning Positional Priors for Pretraining 2D Pose Estimators". In: *Proceedings of the 2nd International Workshop on Human-Centric Multimedia Analysis*. HUMA'21. Virtual Event, China: Association for Computing Machinery, 2021, pp. 3–11. ISBN: 9781450386715. DOI: 10.1145/3475723.3484252. URL: https://doi.org/10.1145/3475723.3484252.

[19] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[20] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929 (2020). arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

[21] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

Glossary

AI Artificial Intelligence. 2

CNN Convolution Neural Network. 3, 24
CPM Convolutional Pose Machine. ii, 3, 5, 6, 7, 8, 9, 11, 14, 16, 17, 18, 19, 24
CV Computer Vision. ii, 2, 3, 4, 5, 28
CVPR The IEEE Conference on Computer Vision and Pattern Recognition. 7

GMM Gaussian Mixture Model. 4

IMU Inertial Measurement Unit. 2

KNN K Nearest Neighborhood. 4

LSP Leeds Sports Pose. 3, 10, 14, 16, 17, 19

MSE Mean Squared Error. 8, 17

PCK Percentage of Correct Keypoints. 3, 11
PDF Probability Density Function. 7

SSL Self-Supervised Learning. ii, 2, 3, 4, 5, 11, 12, 19, 21, 22, 23, 24, 25, 26, 27, 28

TPU Tensor Processing Unit. 24

A Hardware and software configuration

A.1 Hardware description

All the experiments in this project are running on a HP OMEN Gaming laptop.

Figure A.1: HP OMEN 17 Laptop

This laptop's hardware is listed below:

CPU	Intel 12700H
RAM	16GB
GPU	Nvidia 3080Ti
Hard disk	Western Digital SN750 1T

Table A.1: Hardware configuration

The training process is running on the Nvidia RTX 3080 Ti graphic card. This graphic card is based on Ampere architecture. It contains 7424 CUDA units, and has 16 GB GDDR6 memory. This 16GB memory allows larger batch size for training.

A.2 Software environment

Since the graphic card is Nvidia 3080ti, the CUDA version needs to be the latest one. Therefore, the version of Ubuntu, pytorch is also determined by it. The configuration of software environment is listed below:

Environment/Software	Version
Ubuntu	20.04
Python	3.9.7
CUDA	11.6
Pytorch	1.11.0

Table A.2: Software environment configuration

A.3 Learning configuration

The configuration in the learning process is shown below, for the pre-training task, the configuration is:

Parameters	Value
Batch size	10
Workers	6
Learning rate	0.000004

Table A.3: Learning parameters configuration

The configuration for the downstream task and Supervised learning is:

Parameters	Value
Batch size	12
Workers	10
Learning rate	0.000004

Table A.4: Learning parameters configuration for downstream task and Supervised learning