Semi-supervised Training for Positioning of Welding Seams

by

Wenbin Zhang

Thesis submitted to the University of Ottawa
in Partial Fulfillment of the Requirements for the Degree of
Master of Computer Science
Concentration in Applied Artificial Intelligence

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Supervised deep neural networks have been successfully applied to many real-world measurement applications. However, their success relies on labeled data which is expensive and time-consuming to obtain, especially when domain expertise is required. For this reason, researchers have turned to semi-supervised learning for image classification tasks. Semi-supervised learning uses structural assumptions to automatically leverage unlabeled data, dramatically reducing manual labeling efforts.

We conduct our research based on images from Enclosures Direct Inc. (EDI) which is a manufacturer of enclosures used to house and protect electronic devices. Their industrial robotics utilizes a computer vision system to guide a robot in a welding application employing a laser and a camera. The laser is combined with an optical line generator to cast a line of structured light across a joint to be welded. An image of the structured light is captured by the camera which needs to be located in the image in order to find the desired coordinate for the weld seam. The existing system failed due to the fact that the traditional machine vision algorithm cannot analyze the image correctly in unexpected imaging conditions or during variations in the manufacturing process.

In this thesis, we purpose a novel algorithm for semi-supervised key-point detection for seam placement by a welding robot. Our deep learning based algorithm overcomes unfavorable imaging conditions providing faster and more precise predictions. Moreover, we demonstrate that our approach can work with as few as ten labeled images accepting a reduction of detection accuracy. In addition, we also purpose a method that can utilize full image resolution to enhance the accuracy of the key-point detection.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Professor Jochen Lang, for his consistent support, motivation, and guidance throughout my Master study. Dr. Lang continuously provided encouragement and was always willing and enthusiastic to assist in any way he could during the research project.

I wish to show my grateful appreciation to Mitacs Accelerate Internship and Enclosures Direct Inc., Ottawa, Canada who provided the financial support for the research.

I would also like to acknowledge my ex-roommate, Yang Zhou for giving a lot of ideas and encouragement to me.

Dedication

First and foremost, I have to thank my parents for their love and support throughout my life. A special feeling of gratitude to my parents whose words of encouragement to me that it is never too late to pursue your passion. They have been my source of inspiration and gave me strength. They continually provide their moral, spiritual, and financial support.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AdaGrad	Adaptive gradient algorithm
Adam	Adaptive moment estimation algorithm
ANFIS	Adaptive neuro-fuzzy inference system
ANN	artificial neuro network
CCD	Charge-coupled device
CSL	Cross structured light
CMM	Coordinate measuring machines
CNN	Convolutional neural network
CPM	Convolutional pose machine
DCNN	Deep convolutional neural network
EDSR	Enhanced deep super-resolution network
ELM	Extreme learning machine
ELU	Exponential linear unit
GAN	Generative adversarial network
GTAW	Gas tungsten arc welding
HPE	Human pose estimation
IID	Independent and identically distributed
I&M	Instrumentation & measurement
PRM	Pyramid residual module
ReLU	Rectified linear unit
ResNet	Residual network

RMSProp	Root mean square propagation algorithm
SGD	Stochastic gradient decent
SMAW	Shielded metal arc welding
SOMS	Stud online measuring System
SSC	Semi-supervised classification
SSL	Semi-supervised learning
SSR	Semi-supervised regression
SVM	Support vector machine
TSA	Training signal annealing
MLP	Multi-layer perceptron
MSE	Mean absolute error
NMRN	Normal map regression Network
SLFN	Single hidden-layer feed-forward neural network
SVHN	Street view house numbers

Chapter 1

Introduction

1.1 Problem statement

The general objective of this research is to design a machine vision system with increased accuracy and robustness compared to an existing legacy machine vision solution. The goal is pursued by utilizing modern machine learning methods in particular deep learning with novel semi-supervised training methods. Images are captured by a camera which containing the projected structured light. Our network takes images as input and output coordinate pairs. Then the outputs pass to an existing robot welding system to proceed the welding process.

Structured light sensing for welding seam tracking is one of the widely used techniques in robotic welding [1]. Images captured by the structured light sensors are affected by the intensity of the light in the welding process. In addition, different types of metal sheets produce varied light reflection, e.g., stainless steel produces strong specular reflections of the laser light. There is also a lot of noise in the images due to the irregular surface and shininess of the welding seam and due to the image background in the industrial environment. Existing machine vision systems are successfully applied in industry and

produce satisfactory results in many cases. However, these approaches lack robustness in face of variation in imaging conditions, changes in materials to be welded, and geometry changes of the welding setup. This a common scenario with legacy measurement systems in manufacturing. Some sample images captured from the camera of the system investigated in this thesis are shown in Figure 1.1.

The term structured light refers to the projection of light with a known shading pattern. The result is the projection of a known light pattern on the captured scene. The deformation of the expected pattern on the scene is detected and measured. To the application scenario of this thesis a laser projector and a camera are placed on top of the seam to be weld and a laser stripe projects an orthogonal pattern onto the desired seam. Since the position of the camera and the laser projector are fixed, the orientation of the projected laser stripe on 2D image is always the same. However, the location of the welding joint is different in each image. In the most cases, the projected laser stripe forms a 'W' shape in the image. Due to the fact that there are variation of the welding joint types and materials of the metal sheets, our dataset also contains irregular images. Various types of joints lead to different shapes in the 2D images. There are five major welding joint types commonly used in the industry such as butt joint, tee joint, corner joint, lap joint and edge joint, which are each made to stand up to the needs and forces of different applications such as butt joint, tee joint, corner joint, lap joint and edge joint. In addition, different types of metal sheets also affect shape variations in the 2D image. The reflectivity of the sheet metal can generate noise reflections in the image. The shape of the reflection can be chuncks, lines or spots. A simple setup illustration is shown in Figure 1.2.

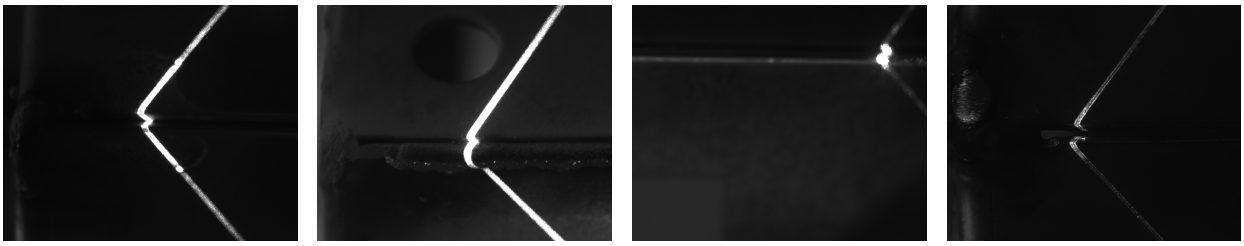
The use of supervised deep learning promises to localize the weld position much more robustly and more accurately in varied imaging conditions. However, supervised deep learning only works well with a large number of annotated samples. The annotation challenge is to precisely specify the desired seam location for a whole dataset. It is very labor intensive and may even require welding expertise, and in addition, multiple annotators

may differ in their choice of seam positioning. In this paper, we instead propose the use of semi-supervised regression (SSR) to reduce the cost of annotation and the negative impact on accuracy by poorly annotated examples. SSR has not been broadly investigated [2] and most of the common semi-supervised classification (SSC) techniques are hard to apply to regression problems.

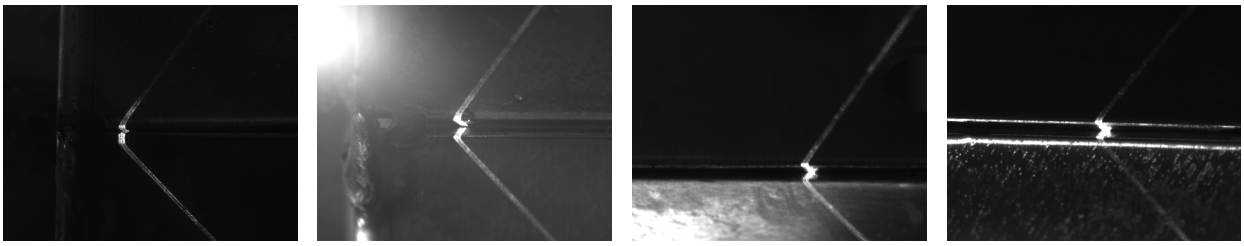
The specific vision-based measurement problem addressed in this paper is to determine a keypoint in an image that enables a calibrated welding robot to place a seam at the corner of an electrical enclosure. The electrical enclosure is manufactured from sheet metal. At present, there exists a machine vision system which finds the keypoint by analyzing the laser lines based on assumptions of their relative geometry in the images. The system fails if the image of the laser stripe is blurred, unexpected specular reflection are present in the images or the geometry is not as expected. We use deep learning in order to increase the robustness of the measurement process given geometric variations in the metal bending and when different types of metals are used. In order to apply supervised deep learning, large amounts of images with a label for the exact position of the expected weld joint would be needed in our application. Labeling keypoints in images is very time intensive and error-prone due to a highly repetitive process which needs to be performed with subpixel precision. Therefore, we develop a novel approach for automatic positioning of welding seams on sheet metal enclosures by semi-supervised deep learning with as few as fifteen manually labeled images.

1.2 Machine learning in metal sheet manufacturing

The use of industrial robots in welding is essential for automation or in hazardous and poor working environments. Robotic welding can improve weld quality, increase adaptability and reduce cost. In general, various types of optical measurements are used to control the trajectory of the robot path for seam tracking [3]. Vision-based measurement [4] can be



(a) Idea configuration (b) A bent edge (c) Blurred intersection (d) Intersection with a gap



(e) Idea configuration (f) A bent edge (g) Blurred intersection (h) Intersection with a gap

Figure 1.1: Variation of *valid* images captured by the welding robot. Our dataset also contains *invalid* images which do not show the location for the weld seam to be placed at all.

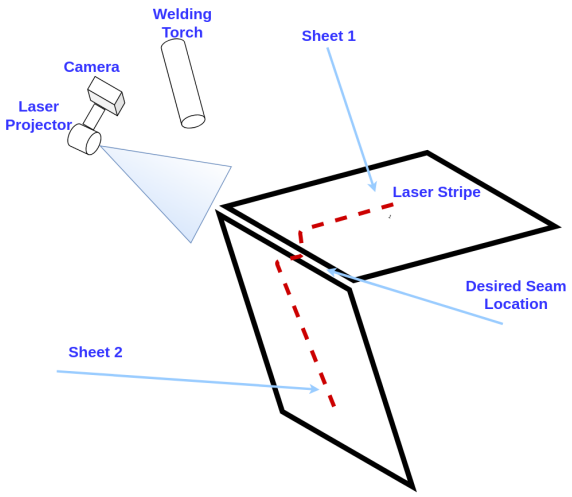


Figure 1.2: Welding robot configuration. The kinematic relationship between image and welding torch is calibrated.

utilized to recognize and find the position of welding creases and to define the weld paths. Xu et al. [5] describe how to mount a camera above a welding torch tip. Vision-based measurement is also used for defect detection of weld beads [6].

We address a vision-based measurement problem in metal sheet manufacturing. The measurement process needs to determine the exact position for a welding robot to place a seam at the corner of an electrical enclosure. We use deep learning in order to increase the robustness of the measurement process given geometric variations in the metal bending and when different types of metals with different optical reflection properties are used.

In order to apply supervised deep learning methods, large amounts of ground-truth results are needed, i.e., the exact position and classification of the expected weld joint are needed in our application. This information can be obtained from an expert by manually labeling images, but this process is very time intensive and error-prone due to a highly repetitive process which needs to be performed with sub-pixel precision. Currently, an analytic, feature-based machine vision algorithm is in use at the industrial site. This algorithm produces satisfactory results in many cases but lacks robustness which is a common scenario with legacy measurement processes in manufacturing. Therefore, we develop methods that are able to learn from as few as possible manually labeled results, augmented by unlabeled images captured by the legacy system. Our novel approach leads to automatic positioning of welding seams on sheet metal enclosures by a semi-supervised deep learning model with as few as ten manually labeled images.

1.3 Key-point localization

Machine learning is actively being used across many domains, such as commerce, entertainment, and increasingly in industrial settings. There are many deep learning approaches to image-based key-point detection on human faces and bodies. As far as we know, there are

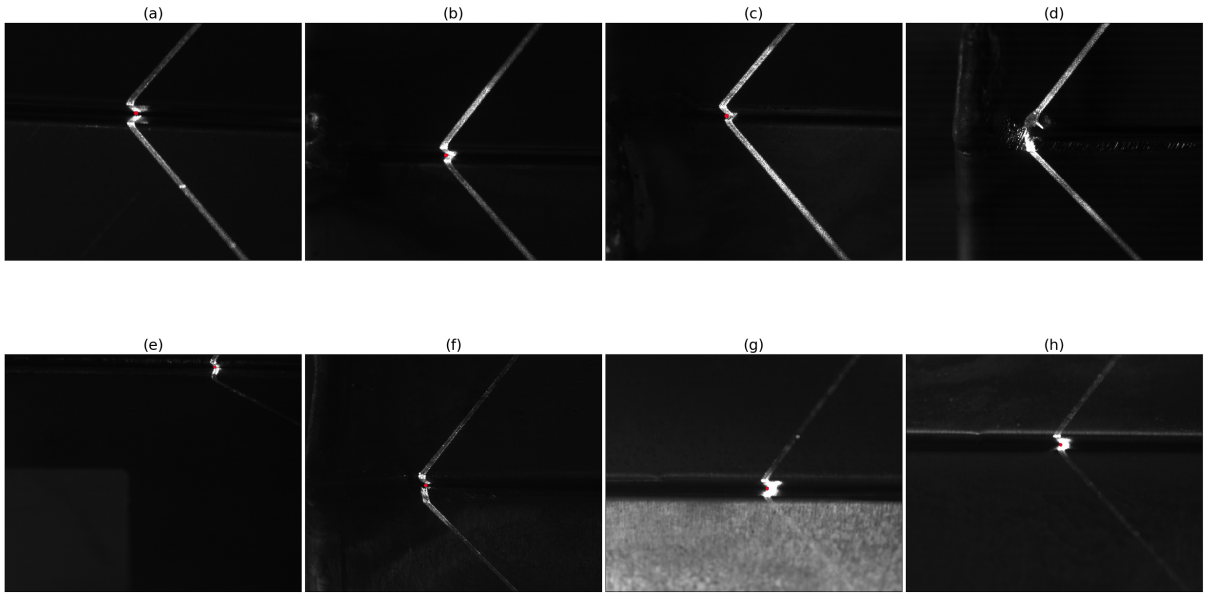


Figure 1.3: (a-c) Various types of valid images. The red dot indicates the key-point. (d) is an invalid image which is welded. (e-h) are all valid images where the traditional CV method fails to detect the intersection. (During the welding process, some of captured images are welded. These images are classified as invalid images. Invalid images are only in the unlabeled dataset. They can be easily excluded during the semi-supervised processes.)

only a few implementations that work on manufacturing images. Traditional algorithms for finding key-points by series of algorithmic steps rely on multiple computer vision (CV) operations such as dilation, Canny edge detection, Hough Transform, etc. A traditional CV method to locate the intersection of lines of projected laser stripes leads to low accuracy, and it is sensitive to imaging conditions such as luminance, orientation, and even the length of the lines. The coordinate prediction of the traditional CV method needs to find the intersection of lines. As a result, its performance highly depends on the edge detection method. If the two detected lines are too short, the current system fails to calculate the intersection. Some failure cases of the traditional algorithm are shown in Fig 1.3(e-h). Figure 1.3 shows some example images with laser line projections where needs to be found the intersection marked with the red dot coordinate.

1.4 Semi-supervised Learning

In many real-world applications, there is a large amount of unlabeled data available, while labeled data is limited. Labeling data is arduous and expensive. It normally requires domain experts, special devices, and experimental verification. As a result, Semi-Supervised Learning (SSL) methods have attracted much attention and have been applied in various fields by employing only a small number of labeled data along with a vast amount of unlabeled data. SSL aims to establish effective learning algorithms and improve modelling performance in order to achieve better results than those models obtained from labeled data alone by extensively analyzing the information of the unlabeled data. Semi-supervised learning by Chapelle et al. [7] is a core machine learning technique to effectively exploit unlabeled samples as much as possible.

Given a set of m labeled data $L_d = (x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$ and a set of n unlabeled data $U_d = x_{m+1}, x_{m+2}, ..., x_{m+n}$ with $x_i \in R^p, i = 1, 2, ..., m + n$ and $y_i \in R$, we want to predict the values of y for any new x. In supervised learning algorithms only the labeled data contribute to the learning process, but SSL algorithms use information from labeled and unlabeled data.

Depending on the nature of the output, semi-supervised classification and semi-supervised regression constitute the basic components of semi-supervised learning [2]. Various studies dealt with the implementation of semi-supervised classification techniques in many real world problems over the last few decades. Semi-supervised classification (SSC) refers to the case that the output variable is discrete. On the other hand, semi-supervised regression (SSR) refers to the case where the output variable is real-valued. Most of the SSC problems aim to predict a label from a finite set of class labels. The basic difference between image classification and image regression tasks is target variable in classification task is not continuous while in regression task it is continuous.

Some notable SSL methods that have been effectively used in numerous scientific fields

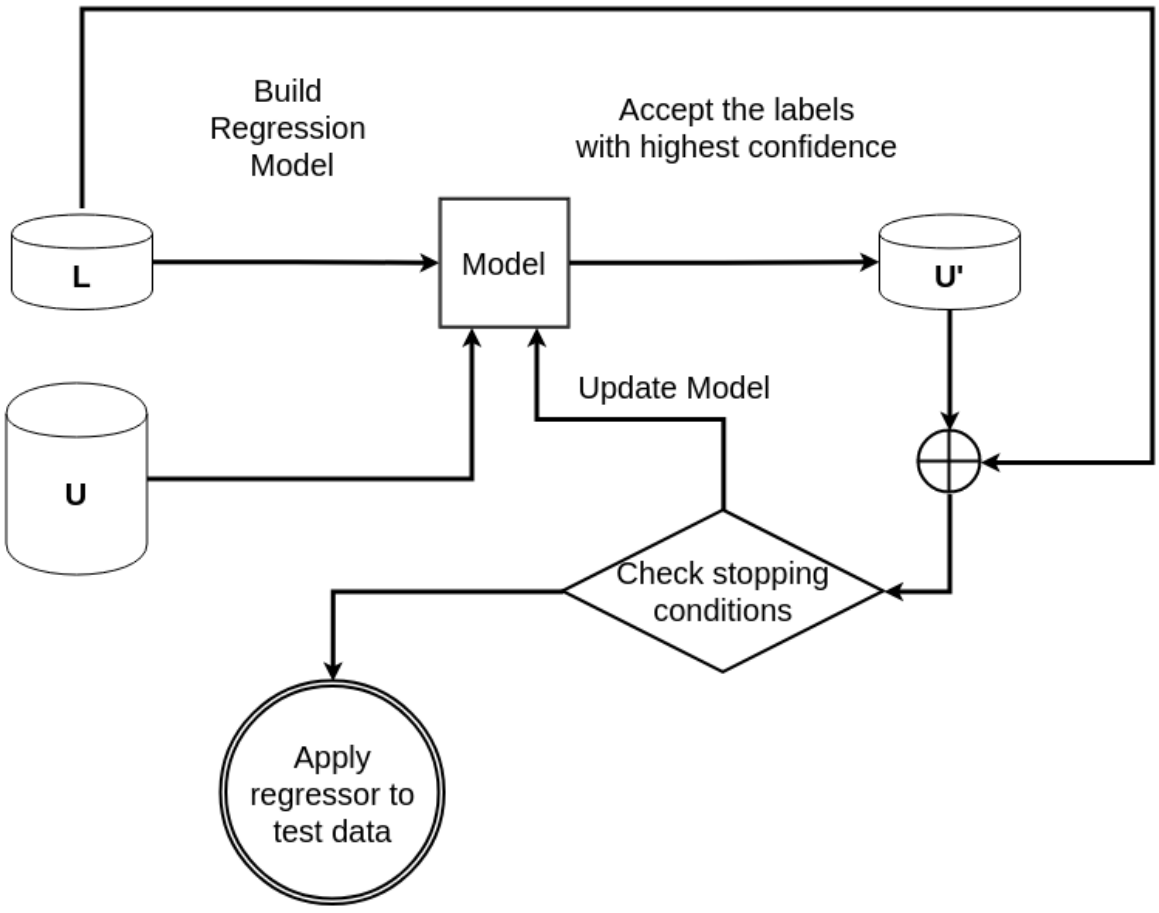


Figure 1.4: SSR framework

are Self-training [8], Co-training [9], and Transductive SVM (Support Vector Machines) [10]. However, most of previously mentioned studies deal with the implementation of classification techniques in contrast to regression. It is difficult to apply classification algorithms to regression problems due to the real valued type of the output variable in regression results. Our proposed algorithm addresses an SSR problem shown in Fig 1.4.

1.5 Thesis statement and contributions

This thesis proposes a semi-supervised deep learning solution for a vision-based measurement task in metal sheet manufacturing. We aim to show that our method leads to an accurate and robust key-point localization method based on minimal labeling effort. We make three major contributions.

The first contribution is introducing a semi-supervised training method for a vision-based measurement task that is successful with very few hand-labeled examples. To the best of our knowledge, this is the first time that a semi-supervised learning method is applied to a single key-point detection problem. Traditional deep learning-based algorithms apply semi-supervised strategies to solve classification problems, such as [11] [12] [13]. The typical solutions for key-point localization problems are using supervised learning methods, such as improved stacked hourglass network [14], stacked hourglass networks for human pose estimation [15], Stacked hourglass network for facial landmark localization [16]. We propose to use a semi-supervised learning method to solve a key-point localization problem by introducing a generator-discriminator architecture which enforces solution constraints sequentially. This main contribution results in a method to assess the reliability of the predictions from the regressor.

The second contribution is introducing a method to increase the precision of key-point localization in a two-stage network. Normally, the inputs have to be down-sampled to a

certain size before being fed into the network due to memory limitations or because of computational cost. To overcome this problem, in the proposed semi-supervised key-point localization network (SSLN), we introduce a two-stage network that uses only the attention area of the full resolution image without increasing the parameters significantly but which achieves higher precision.

The third contribution is we delivered our semi-supervised deep learning method to EDI Inc. replacing their existing computer vision system. Our algorithm can successfully detect welding joints which the legacy algorithm fails to calculate. Our proposed method outperforms the existing system in terms of both accuracy and speed.

1.6 Thesis structure

Our goal for this thesis is to focus on two main objectives: developing a deep learning based key-point localization algorithm and applying semi-supervised learning techniques to the algorithm. In Chapter 2, we introduce some fundamental knowledge such as basic neural networks, convolutional neural networks, and the techniques to train those models. In Chapter 3, we describe more related work to machine learning in instrumentation and measurement, key-point localization problems, and semi-supervised learning in computer vision. Chapter 4 present our novel semi-supervised localization method. Chapter 5 shows our experimental results and gives an analysis. Finally, Chapter 6 concludes the thesis.

Chapter 2

Related concepts

2.1 Convolutional neural network

Convolutional neural networks (CNNs) have become dominant in various computer vision tasks and attracting extensive interest across a variety of domains [17] [18] [19]. In this section, we briefly introduce CNN concepts and terminology.

2.1.1 Convolutional layer

Convolution is the first layer to extract features from an input image. The convolution operation preserves the relationship between pixels by learning image features using small squares of input data. As we described in Section **??**, the behavior of a neuron can be expressed as a function of vectors as Eq. (**??**). Similarly, a neural network layer can be formulated as

$$\mathbf{y} = f\big(\mathbf{W} \cdot \mathbf{X} + \mathbf{b}\big), \tag{2.1}$$

where $\mathbf{W}$ and $\mathbf{X}$ are matrices representing weights and inputs respectively, $\mathbf{b}$ and $\mathbf{y}$ are vectors for bias and output, the symbol $\cdot$ represents the mathematical operation of matrix

multiplication. The convolutional layer, which is the core component of CNNs, is quite similar to regular neural network layer. While the input, weight, bias and activation function remain unchanged, instead of matrix multiplication, the input and weight are combined with the convolution operation in the convolutional layers. One of the biggest advantage of using convolution operation instead of matrix multiplication is that it can save a lot of learnable parameters, which leads to much lower computational cost and faster speed. Its behavior can be expressed as

$$\mathbf{y} = f\big(\mathbf{W} * \mathbf{X} + \mathbf{b}\big), \tag{2.2}$$

where $*$ is convolution.

CNNs have several filters consisting of trainable parameters which can convolve a given image spatially to detect features like edges and shapes. Numerous of filters essentially learn to capture spatial information from the image based on the weights through back propagation. Moreover, these stacked layers of filters can be used to detect more complex spatial shapes from the spatial features learned at every subsequent level. As a result, the given input image can be boiled down into a highly abstracted representation in latent layers which helps prediction. Therefore, a CNN is actually a filter-based image processing system with self-learned filter coefficients. Usually, a standard CNN has a general structure of convolutional layers, pooling layer, etc. We will describe the details in the following paragraphs.

They are some additional parameters which are not learnable parameters present in CNNs. Typically, they can be defined while building the network and can be updated while tuning the network.

Kernel (filter). Kernel is a matrix containing weights to convolve the input with. The convolution filter provides the measurement for how close an input patch resembles a feature. The weights in the kernel matrix are derived during the back propagation process

when training. Smaller filters collect local information, while larger filters capture more global, high-level representative information from the input data. For computational efficiency, the size is commonly set to 3×3, 5×5 or 7×7. Moreover, recently, He et al [20] shows that bigger kernels such as 5×5 or 7×7 substituted by multiple smaller filters (3×3) on top of one another can be more efficient.

Padding. Padding is generally used to add columns and rows of zeros to keep the spatial sizes constant after convolution process. This operation can improve performance as it retains the information at the borders. For example, let the input be a $H \times W$ pixels image, the convolutional layer has a kernel of $aH \times aW$ pixels, where a is the size of the kernel. The size of the output image will be

$$(H - 2(aH \backslash 2)) \times (W - 2(aW \backslash 2)), \tag{2.3}$$

where $\backslash$ denotes integer division. Zero padding the input enables the kernel size and the output size to be controlled independently, so that the users do not need to balance between the kernel size and the network depth. For the inputs of the convolutional layer in Figure 2.1, 1 pixel zero-padding is used for all the four edges.

Stride. This is generally the number of pixels to skip while traversing the input horizontally and vertically during convolution after each element-wise multiplication of the input weights with those in the kernel.

Number of channels. The output number of channels is equal to number of kernels used for the convolution operation. The large the number of channels, the large the number of filters used, the more features can be learned.

Pooling-layer. The pooling operation involves a two-dimensional filter over each channel of the feature map and summarizing the features within the region covered by the filter.

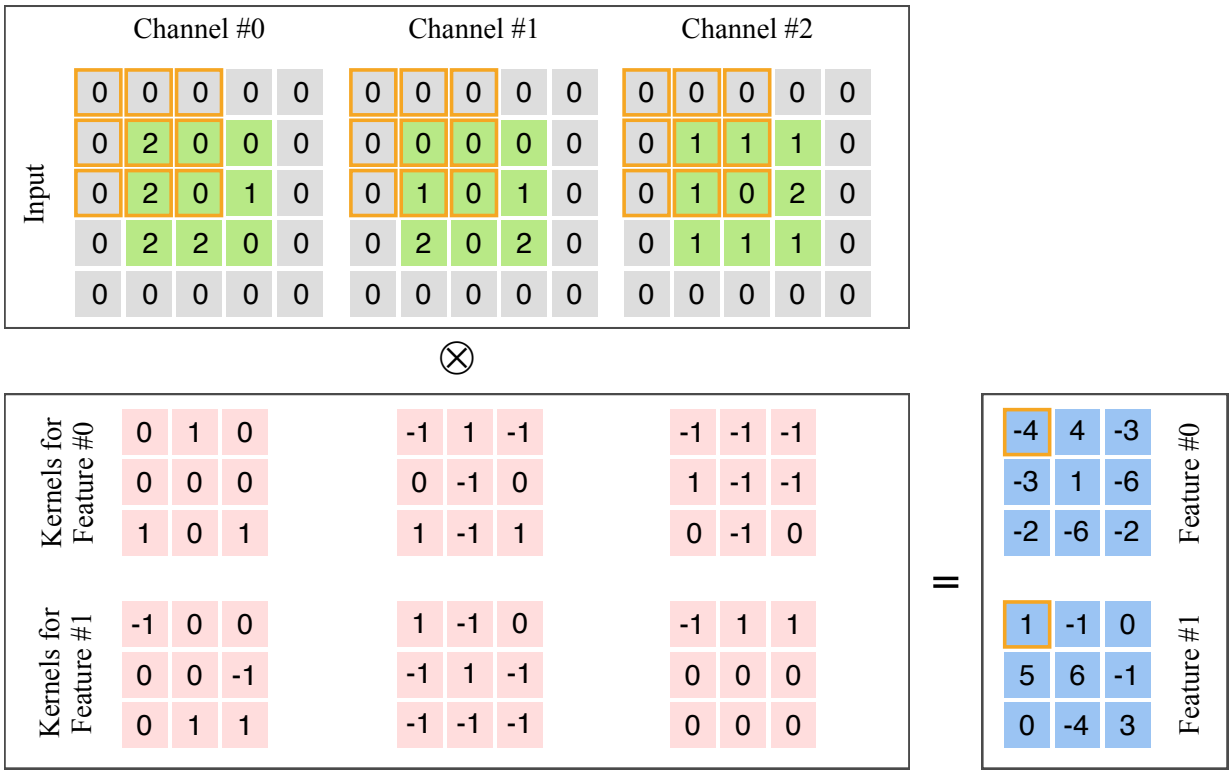


Figure 2.1: A convolutional layer.

The objective of using a pooling layer is to down-sample the input representation, reducing dimensionality by keeping the max value (for max-pooling) in the subregions binned.

Batch normalization. It is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. Generally, in deep neural network architectures the normalized input after passing through various intermediate layers becomes too big or too small which causes internal co-variate shift problem which impacts learning. By adding the batch normalization layer, we can standardize the input and pass it to the later layers. This has the effect of stabilizing the learning process and reducing the number of training epochs required to train deep neural networks.

Stride. Stride of a convolutional layer is the step of sliding. In other words, it controls how the filter moves around the input volume. Stride is normally set in a way so that the output volume is an integer and not a fraction. The formula for calculating the output size for any given convolutional layer is as follows:

$$O = \frac{W - K + 2P}{S} + 1. \tag{2.4}$$

Where O is the output (assume the same height and width), W is the input height and width, K is the filter size, P is the padding, and S is the stride. We will discuss this configuration in more details in Section 2.1.2.

2.1.2 Pooling layer

Pooling layer are frequently used to reduce the spatial dimensions, but not depth, on a convolutional neural network. It receives features from previous layer and replace patch of values by the sliding window according to its neighboring values. There are many advantages by using pooling layer: gain computational performance by having less spatial

information; lower chance to over-fit while having fewer parameters; get translation invariance. For example, the max pooling layer produces the maximum value within a small neighborhood and the average pooling layer reflects the average value of the neighborhood. The most common approach used in pooling is max pooling.

The concepts of kernel size, padding and stride in convolutional layers also apply to pooling layers. It has to be noted that the kernel size in pooling layers refers to the size of the rectangular support area. There is no learnable kernel existing in max and average pooling layers. Illustrations of a max pooling and an average pooling layer are given in Figure 2.2.

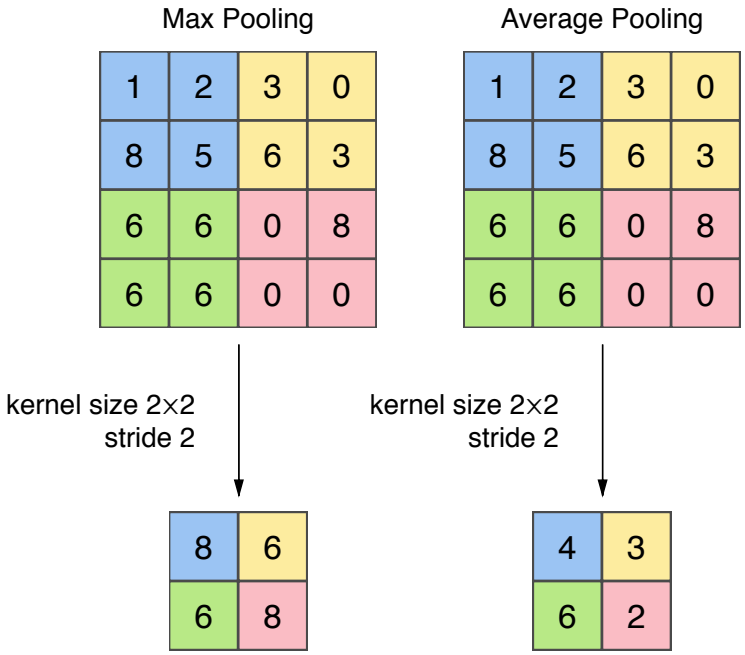


Figure 2.2: Pooling layers.

As shown in Figure 2.2, the output size of pooling layers will shrink when stride is greater than 1. Therefore, pooling layers are also used for down-sampling to reduce the computational cost.

In recent works, convolutional layers with increased stride are often used to replace pooling layers. It is argued that the replacement has the following advantages.

- The learnable kernels in convolutional layers provide more flexibility than regular pooling layers.
- The gradients of convolutional layers are less sparse than pooling layers (especially max pooling). This property might be helpful for the convergence of some networks, such as the generative adversarial networks (GANs).

2.1.3 Common CNN architectures

Most of neural network architectures in computer vision have one or more feature extraction components. Input images are converted to high-level latent space by these components for classification, regression or synthesis. In this section, we will describe some popular CNN architectures that are relevant for this thesis.

VGGnet. Simonyan [21] proposed a simple and effective design principle for CNN architectures. VGGnet is modular in layers pattern containing 19 layers to simulate the relation of depth with the representational capacity of the network. Its variants, VGG-19 and VGG-16, won the first and second place of ImageNet Large Scale Visual Recognition Competition (ILSVRC) localization task in 2014. The use of the small size filters provides an additional benefit of low computational complexity by reducing the number of parameters. As illustrated in Figure 2.3, VGG-16 has 16 layers with learnable weights. The network consists of 13 convolutional layers for feature extraction and 3 fully connected layers for feature analysis and classification. It also regulates the complexity of network by placing 1×1 convolutions in between the convolutional layers, which learn a linear combination of the resultant feature-maps. Therefore, VGGnet achieves fairly good performance while using a limited number of parameters.

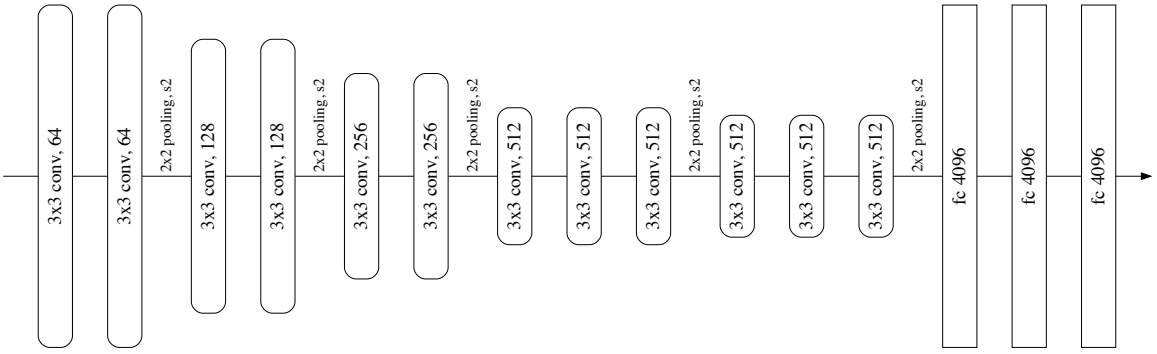


Figure 2.3: A VGG-16 network.

GoogleNet. It was the winner of the 2014-ILSVRC competition and is also known as Inception-V1 [22]. The model is composed of a basic unit referred to as an 'Inception cell' which performs different scales of convolutions and aggregates the results. Inception cells reduce computational cost incorporating multi-scale convolutional transformations using a split, transform and merge concept. Moreover, 5×5 convolutions were decomposed of two stacked 3×3 filters. The architecture of the inception block is shown in Figure 2.4.

ResidualNet [23]. Networks were becoming deeper, but the deeper structure makes it the vanishing gradients problem [24] [25]. The vanishing gradients problem has historically been one of the largest barriers to the success of deep neural networks. The problem introduced when extensive layers using certain activation functions are added to neural networks, the gradients of loss function approaches zero. The vanishing gradients problem causes that the weights and biases of the initial layers become extremely hard to update effectively during backpropagation. ResNet were a breakthrough idea to solve this problem which enabled the development of much deeper networks. The authors propose a remedy to this degradation problem by introducing *residual blocks* in which intermediate layers of a block learn a residual function with reference to the block input. A typical residual block is shown in Figure 2.5. The residual learning architecture makes extremely deep networks

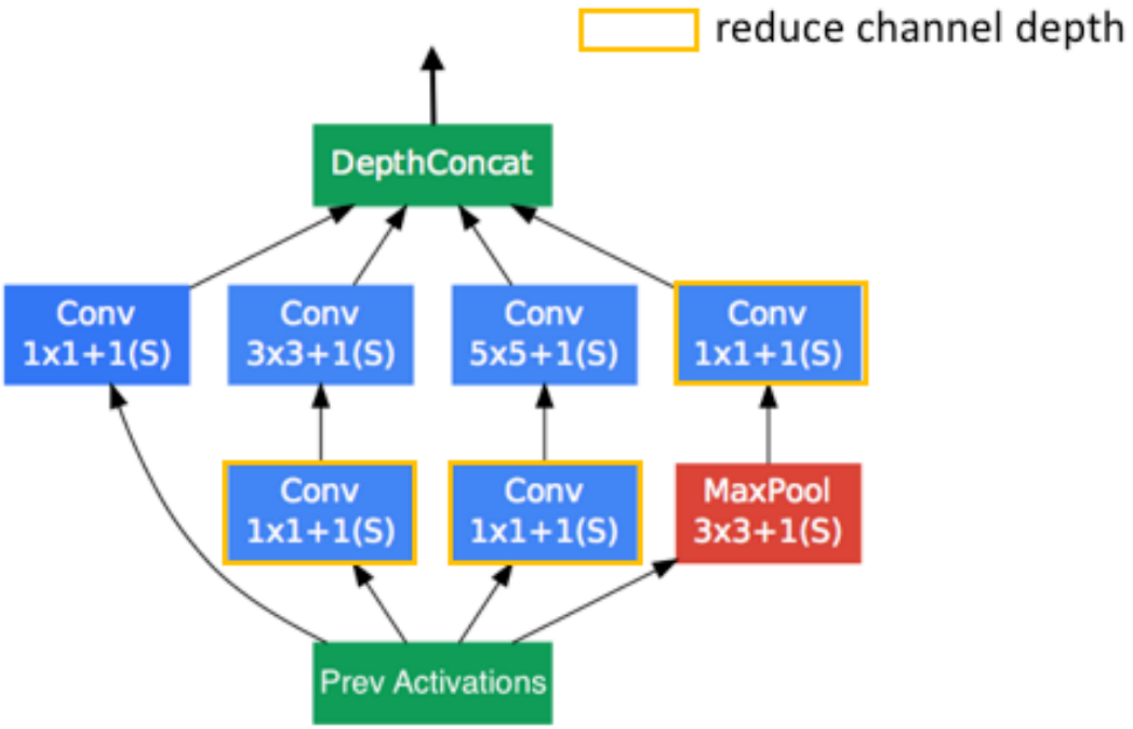


Figure 2.4: Basic architecture of the inception block showing the split, transform, and merge idea.

easier to optimize. Due to the significantly increased depth, ResNet achieves outstanding performance.

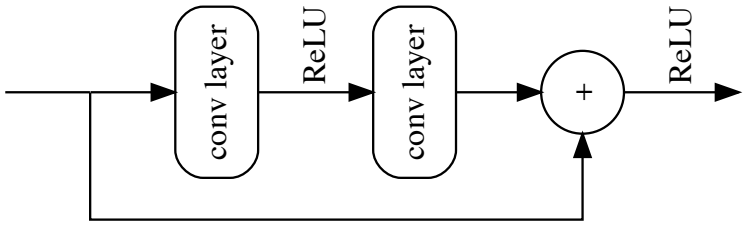


Figure 2.5: Residual block.

Autoencoder [26]. Autoencoders are a well-known type of unsupervised learning technique in which we leverage neural networks for the task of representation learning. The autoencoder models impose a bottleneck in the network which forces a compressed knowledge representation of the original input. If some sort of structure exists in the input data, the autoencoder structured models can capture abstract features and learn the weights from the data. Consequently, it leverages the information from the input data when forcing the input through the network's bottleneck. The architecture of a typical autoencoder is shown in Figure 2.6. Skip connections might be used in an autoencoder to facilitate information flows from the input to the output, as illustrated in Figure 2.7.

U-net. Originally U-net was first developed for biomedical segmentation. It is a classic encoder-decoder network structure. The encoder is the first half in the network diagram in Figure 2.8. It is usually a pre-trained classification network like VGG [27] or ResNet [23] which compresses the input image into feature representations by a maxpool downsampling after convolution blocks at multiple different levels. The second half of the architecture is

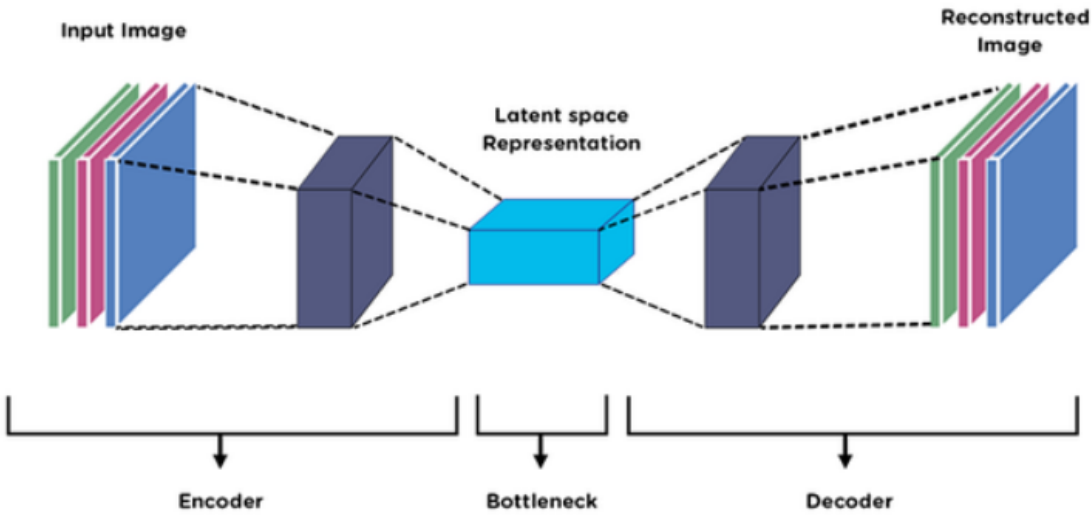


Figure 2.6: Autoencoder.

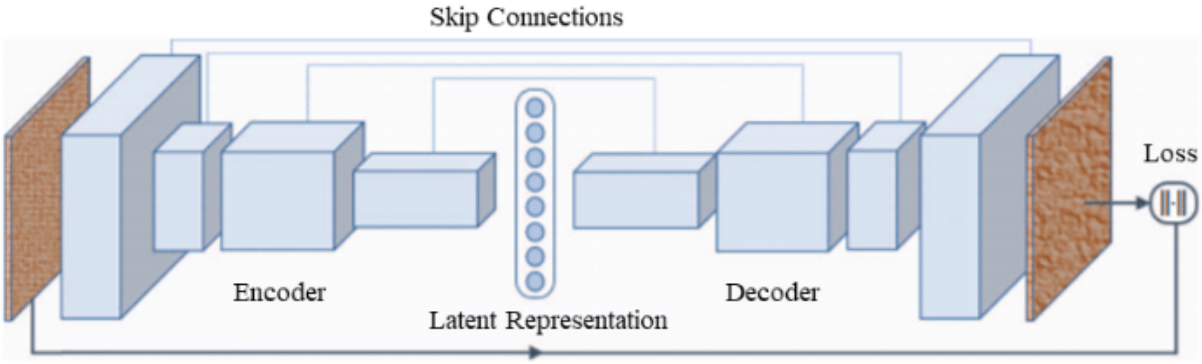


Figure 2.7: Autoencoder with skip connections.

the decoder. It consists of upsampling and concatenation followed by regular convolution operations.

Unlike the autoencoder net work that has exactly one latent space L with a nonlinear mapping from the input, in U-net there is a skip connection which maps the concatenation of input and latent space L to the output layer. This means in the sense of mapping the sample onto some well-defined latent space and then computing the output from it. The U-net cannot be split separately by encoder and decoder. The skip connection is a standard module in many convolutional architectures. By using the skip connection, it provides an alternative path for the gradient. It is experimentally validated that these additional paths are often beneficial for the model convergence. This is one of the reasons why we pick U-net as our generator structure.

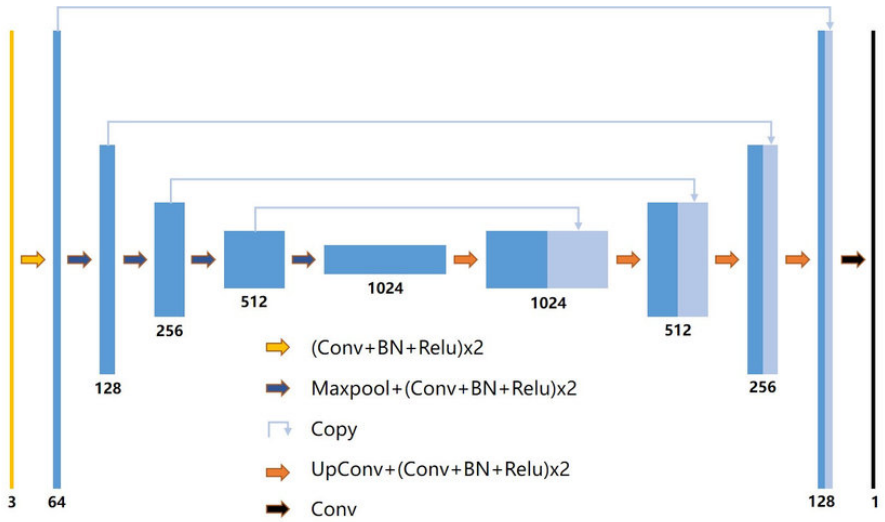


Figure 2.8: Unet architecture.

Generally speaking, CNNs are inspired by the structure of the brain which is constantly analyz the world around us. CNNs mimic the behavior of the brain which tries to make predictions about everything seen without conscious effort. They have a different architecture than regular neural networks. The layers are organized in 3 dimensions: width, height and depth. In addition, the neurons in one layer do not connect to all the neurons in the

next layer but only to a small region of it. The structures of our generator and discriminators models are inspired by these famous models. Furthermore, the final output will be reduced to a single vector of probability scores, organized along the depth dimension. In section 2.2, we demonstrate more details about how to train a neural network.

2.2 Training neural networks

Machine learning is defined by Mitchell [28] as

" A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. ".

Machine learning methods can typically be classified into two categories: supervised learning and unsupervised learning. In a supervised learning model, the algorithm learns on a labeled dataset, providing an answer key that the algorithm can leverage to evaluate its performance on training and validation data. The training set provides input examples that the model will be fit to. On the other hand, the validation set is used to estimate the performance of the model. In contrast, in unsupervised learning unlabeled data is provided to the algorithm, and it tries to make sense of the data by extracting features and patterns on its own. Our research is somewhere in the middle of the two models, which is called semi-supervised learning. It tends to use a small amount of labeled data bolstering a larger set of unlabeled data. Despite the different types of learning strategies, they all have some of the same properties and components. In this section, we will explain some fundamental properties for both learning strategies.

2.2.1 Dataset

The training dataset plays a very important role in machine learning as the ML model comprehends such data and memorizes the information for future prediction. Without having effective datasets, training of learning models for various crucial tasks cannot be accomplished. Even for developing semi-supervised learning models, initially, a minimal amount of labeled data is required to initialize the model and to update the model sequentially with unlabeled data. Typically, in supervised learning scheme, labeled data is separated into 3 groups, each group serves its specific training purpose as below:

- **Training dataset:** Samples of data that is used to fit the model.
- **Validation dataset:** Samples of data that is used to provide an unbiased evaluation of a model fit on the training dataset while tuning model parameters. The validation set is also known as the Dev set or the Development set.
- **Test dataset:** Sample of data used to provide an unbiased evaluation of a final model fit on the testing dataset. It provides the gold standard used to evaluate the model. Many times, the validation set is used as the test set, but this is not good practice because it introduces a dependency during tuning of model parameters.

There are some training strategies that can be used to increase the performance of a model such as k-fold cross validation. Cross-validation is a statistical method used to estimate the skill of machine learning models. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups.

3. For each unique group:

   (a) Take the group as a hold out or test dataset.

   (b) Take the remaining groups as a training dataset.

   (c) Fit a model on the training set and evaluate it on the test dataset.

   (d) Retain the evaluation score and discard the model.

4. Summarize the skill of the model using the sample of model evaluation scores.

There are various advantages in applying k-fold cross validation for training a model:

1. It is able to use all samples both for training and for testing while evaluating learning algorithm on examples it has never seen before. This is extremely helpful when we have very little data to split into training and test.

2. When we create five different models using a learning algorithm and test the model's on five different test datasets, we can be more confident in the algorithm performance since we are getting more metrics.

3. When building a pipeline of models, by cross validation, we can train two individual models by different datasets. Note, you want to split only the training and validation set into the k-fold and still have an independent test.

4. It is helpful for fine-tuning the model parameters.

2.2.2 Parameters and hyper-parameters

Being effective in developing deep neural networks requires to not only organize learnable parameters well but also to tune hyper-parameters. A model parameter is a variable of the model which can be estimated by fitting the given data to the model. In contrast, a model

hyper-parameter is set before the model training. Hyper-parameters cannot be learned by fitting the model to the data. However, hyper-parameters also play on important role in algorithm performance since they control model behaviors.

2.2.3 Loss function

The loss function is an objective mathematical function evaluating how well a specific algorithm models the given data by mapping a network to a real number. If model predictions deviate too much from actual results, the loss function will calculate a large number. Essentially, training a network is a process to successively optimize the value of the loss function to cause the model predictions to be close to the actual results. Gradually, by optimizing the loss function, the model learns to reduce the error in prediction. In this section, we will introduce several commonly used loss functions.

MSE and L2 loss. MSE is short for the mean squared error, which is defined as

$$\mathrm{MSE}(\hat{I}, I_{gt}) = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \Big[I_{gt}(i,j) - \hat{I}(i,j) \Big]^2, \tag{2.5}$$

where $\hat{I}$ is the predicted image, I_{gt} is the ground truth. Both of them are of size $M \times N$. MSE loss is often used when the outputs are images. In our case, our generator outputs predicted 2-dimensional heatmaps that is compared with heatmap representations of the coordinates groundtruth. L2 loss is the square of the L2 norm of the difference. In brief, it is just the sum of squared errors whose definition is

$$\mathrm{L2}(\hat{I}, I_{gt}) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \Big[I_{gt}(i,j) - \hat{I}(i,j) \Big]^2. \tag{2.6}$$

However, due to the fact that pixel values are often normalized within $[0, 1]$, the quadratic property of MSE/L2 loss may impede the convergence speed.

MAE and L1 loss. MAE, short for the mean absolute error, is formulated as

$$\mathrm{MAE}(\hat{I}, I_{gt}) = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \Big| I_{gt}(i,j) - \hat{I}(i,j) \Big|, \tag{2.7}$$

where $|\cdot|$ denotes the absolute value. Similarly, L1 loss is the sum of absolute errors, which is defined as

$$\mathrm{L1}(\hat{I}, I_{gt}) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \Big| I_{gt}(i,j) - \hat{I}(i,j) \Big|. \tag{2.8}$$

Generally, the L2 loss function is preferred in most cases. But when outliers are present in the dataset, due the linear property of the L1 loss function large errors have smaller influence in MAE than MSE and L2 loss. MAE loss is more robust in terms of training a model.

Cross entropy. The cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. If the predicted probability diverges from the actual label, the cross-entropy loss error increases (the upper bound is positive infinite). Similarly, the cross-entropy loss error decreases as the predicted probability converges from the label. The cross-entropy is a measurement of the divergence between two probability distributions in information theory. It can be used as the loss function for classification tasks. Let p_i be the true probability distribution and q_i is the predicted distribution. Considering a binary classification task, we assume $p \in \{y, 1-y\}$ and $q \in \{\hat{y}, 1-\hat{y}\}$. The cross-entropy loss is then defined as

$$\mathrm{CrossEntropy}(p, q) = -y\log\hat{y} - (1-y)\log(1-\hat{y}). \tag{2.9}$$

The cross-entropy is the most common loss function for classification tasks. We apply the cross-entropy loss function to our discriminators, from which decide whether the prediction from the generator is qualified.

2.2.4 Optimization

Optimization in the context of neural networks is usually non-convex optimization. Optimizers are algorithms or methods used to update weights of a neural network in order to reduce the objective function $L(x)$. Optimization algorithms or strategies are responsible for reducing the losses and to provide the most accurate possible predictions. The absolute lowest value of $L(x)$ is called global minimum. However, finding the true global minimum can be an extremely challenging problem and analytical methods are frequently not applicable. In contrast, a local minimum of a function is a point where the function value is smaller than at nearby points, but possibly greater than at a distant point. Most local minima incur a low cost, and thus a local minimum with reasonably low error is acceptable. Indeed, in the context of neural networks, we are usually satisfied with finding a local minimum with low cost rather than finding the global minimum.

Recent studies [29] indicate that in high dimensions, saddle points are more likely than local minima. Saddle points are also more problematic than local minima because close to a saddle point the gradient can be very small. Thus, gradient descent will result in negligible updates to the network.

A lot of work on solving optimization tasks or improving optimization methods in neural networks has been proposed successively. In this section, we will introduce the principles and progresses of commonly used optimization methods.

Stochastic Gradient Descent (SGD) [30] SGD is a variant of Gradient Descent. It tries to update the model's parameters more frequently. Almost all neural network optimizations are powered by the stochastic gradient decent (SGD) algorithm. In brief, the gradient decent algorithm treats each learnable parameter as one dimension of a high-dimensional space. For example, if the dataset contains 1000 items, SGD will update the model learnable parameters 1000 times in one cycle of the dataset instead of a single time

as in Gradient Descent. The iterative update procedure can be formulated as

$$w = w - \eta \nabla Q_i(w), \tag{2.10}$$

where w denotes the parameters, $Q_i(w)$ is the value of loss function at i-th iteration, η is the step size of update operation and is more frequently called the learning rate.

The advantages of SGD are:

- Converges in less time due to SGD updating model parameters frequently by a batch of samples data.
- The data can be shuffled to prevent periodic arrangements in each training session.
- Less computational cost since no need to store the values of the loss functions.

However, there are disadvantages of SGD:

- High variance in model parameters.
- Possibly pass over the global minima.
- Normally, a learning schedule for slowly reducing the value of learning rate is needed in order to get the same convergence as with gradient descent.

The SGD algorithm can be presented in pseudo-code as follows.

Input: Training data, learning rate η

Output: Model parameters w

Choose an initial vector of parameters w and learning rate η;

repeat

  Randomly shuffle examples in the training set;

  for $= 1, 2, ..., n$ **do**

    $w = w - \eta \nabla Q_i(w)$;

  end

until *an approximate minimum is obtained*;

In machine learning, the learning rate of SGD has been considered problematic. If the learning rate is low, then training is more reliable, but the optimization process will take a lot of time. However, if the learning rate is high, then training may not converge or even diverge since the optimizer overshoots the minimum. There are many variations of stochastic gradient descent for helping the optimizer decide how far to move the weights in the direction opposite of the gradient.

Nesterov accelerated gradient. Nesterov accelerated gradient (NAG) [31] refers to a general approach that can be used to modify a gradient descent-type method to improve its initial convergence. It is an optimization algorithm that helps to limit the overshoots in Momentum Gradient Descent Look Ahead Gradients. Computing $\theta - \gamma v_{t-1}$ gives us an approximation of the next position of the parameters which effectively looks ahead by calculating the gradient not with respect to current parameters θ but w.r.t the approximate future position of parameters. The behavior can be formulated as

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1}) \\ \theta &= \theta - v_t \end{aligned} \tag{2.11}$$

NAG first makes a big jump in the direction of the previous accumulated gradient, measures the gradient and then makes a correction. This anticipatory update prevents NAG from going too fast which has significantly increased the performance [32].

Adam Adam [33] is short for adaptive moment estimation which computes adaptive learning rates for each parameter. It keeps an exponentially decaying average of past gradients m_t which is similar to momentum. Adam behaves like a heavy ball with friction, which prefers flat minima in the error surface [34]. The formulas to compute the decaying average of past and past squared gradients m_t and v_t respectively are

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{2.12}$$

m_t and v_t estimates the first and second moment of the gradients respectively.

2.2.5 Back-propagation

Considering a dataset consisting of input-output pairs $(\vec{x_i}, \vec{y_i})$, where $\vec{x_i}$ is the input and $\vec{y_i}$ is the desired output of the network on input $\vec{x_i}$. The set of input-output pairs of size N is denoted X = $\{(\vec{x_1}, \vec{y_1}), ...(\vec{x_N}, \vec{y_N})\}$. The information flows forward through the input to the hidden layers and finally the output layer, which forms a feed-forward neural network. An error function $E(X, \theta)$, which defines the error between the desired output $\vec{y_i}$ and the calculated output $\hat{\vec{y_i}}$ of the neural network on input $\vec{x_i}$ for a set of input-output pairs $(\vec{x_i}, \vec{y_i})$ $\in X$ and a particular value of the parameters θ.

Definition of variables:

- w_{ij}^k: weight for node j in layer l_k for incoming node i

- b_i^k: bias for node i in layer l_k
- a_i^k: product sum plus bias (activation) for node i in layer l_k
- o_i^k: output for node i in layer l_k
- r_k: number of nodes in layer l_k
- g: activation function for the hidden layer nodes
- g_o: activation function for the output layer nodes

Training a neural network with gradient descent requires the calculation of the gradient of the error function $E(X, \theta)$ with respect to weights w_{ij}^k and biases b_i^k. Then, according to the learning rate α, each iteration of gradient descent updates the weights and biases collectively denoted θ according to

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial(X, \theta^t)}{\partial \theta}, \tag{2.13}$$

where θ^t denotes the parameters of the neural network at iteration t in gradient descent.

The back-propagation algorithm is dependent on the following five equations:

1. The partial derivatives:
$$\frac{\partial E_d}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1}. \tag{2.14}$$
2. The final layer's error term:
$$\delta_1^m = g'_0(a_1^m)(\hat{y_d} - y_d). \tag{2.15}$$
3. The hidden layers' error terms:
$$\delta_j^k = g'(a_j^k) \sum_{l=1}^{r^{k+1}} w_{jl}^{k+1} \delta_l^{k+1}. \tag{2.16}$$

4. A combination of the partial derivatives for each input-output pair:

$$\frac{\partial E(X,\theta)}{\partial w_{ij}^k} = \frac{1}{N}\sum_{d=1}^{N}\frac{\partial}{\partial w_{ij}^k}(\frac{1}{2}(\hat{y}_d - y_d)^2) = \frac{1}{N}\sum_{d=1}^{N}\frac{\partial E_d}{\partial w_{ij}^k}. \tag{2.17}$$

5. Finally, the update of the weights:

$$\Delta w_{ij}^k = \alpha\frac{\partial E(X,\theta)}{\partial w_{ij}^k}. \tag{2.18}$$

The back-propagation algorithm proceeds in the following steps (assuming a learning rate α and random initialization of the parameters w_{ij}^k):

1. Calculate the forward phase for each input-output pair $(\vec{x_d}, \vec{y_d})$ and store the results $\hat{y_d}$, a_j^k, and o_j^k for each node j in layer k by proceeding from layer 0, the input layer, to layer m, the output layer.

2. Calculate the backward phase for each input-output pair $(\vec{x_d}, \vec{y_d})$ and store the results $\frac{\partial E_d}{\partial w_{ij}^k}$ for each weight w_{ij}^k connecting node i in layer k_1 to node j in layer k by proceeding from layer m, the output layer, to layer 1, the input layer.

3. Evaluate the error term for the final layer δ_1^m by using Equation (2.15).

4. Back-propagate the error terms for the hidden layers δ_j^k, working backwards from the final hidden layer k=m-1, by repeatedly using Equation (2.16).

5. Evaluate the partial derivatives of the individual error E_d with respect to w_{ij}^k by using Equation (2.14).

6. Combine the individual gradients for each input-output pair $\frac{\partial E_d}{\partial w_{ij}^k}$ to get the total gradient $\frac{\partial E(X,\theta)}{\partial w_{ij}^k}$ for the entire set of input-output pairs X = $\{(\vec{x_1}, y_1), ..., (\vec{x_N}, y_N)\}$ by using Equation (2.17).

7. Update the weights according to the learning rate α and total gradient $\frac{\partial E(X,\theta)}{\partial w_{ij}^k}$ by using Equation (2.18).

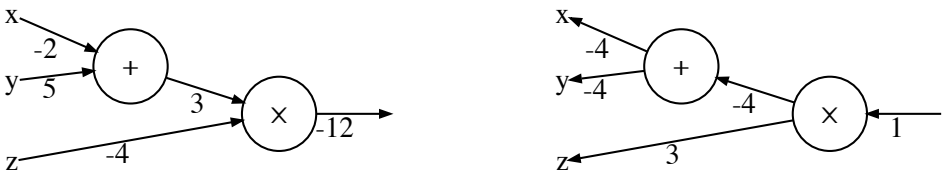
The back-propagation algorithm is based on the chain rule from differential calculus. The chain rule helps us find the derivative of a composite function. As an example, let f and g both be functions in real number domain and satisfy $y = g(x)$ and $z = f(g(x)) = f(y)$. The chain rule states that

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}. \tag{2.19}$$

Based on the chain rule, the back-propagation algorithm can be clearly demonstrated with the help of computational graphs. Assuming we are trying to calculate the derivatives for a function f, defined as

$$f(x, y, z) = (x + y) \cdot z.$$

The function can be expressed as a computational graph as illustrated in Figure 2.9. Assuming the training data we are using is $x = -2, y = 5, z = -4$, the forward propagation computes the results for each operation node as shown in the figure to the left. The back-propagation algorithm starts from the end and recursively computes the gradients conditioned on the property of each operation node all the way to the inputs, as shown in the figure to the right.



Forward propagation Back propagation

Figure 2.9: Forward and backward propagation.

Chapter 3

Literature review

In this chapter we give an overview about work related to the topics of this thesis. To survey the pertinent literature sources, we focus on machine leaning algorithms applied in welding, keypoint detection methods, and semi-supervised learning approaches. We start in Section 3.1 by discussing state-of-the-art machine learning approaches designed for welding systems. We describe in Section 3.3 several approaches related to keypoint detection approaches using deep learning. Finally, Section 3.4 gives a brief summary of semi-supervised learning methods with a focus on classification algorithms in machine vision. We focus on the above three research directions since the objective of our thesis is to address a keypoint localization problem for a welding seam system in a semi-supervised learning manner.

3.1 Machine Learning in Welding

Machine learning is increasingly applied to vision-based measurement in welding. Computer vision techniques have great success in fast inspection in many industry fields [35,36]. Only a few systems are developed for weld seam measurement. Zhang et al. [37] describe

a cross structured light (CSL) setup on a robot which uses a Hidden Markov model for weld line detection and tracking. A CSL device is designed to project cross of a red laser stripe on welded surfaces and capture the weld convexity in video sequences. The authors incorporate both the spatial characteristics of laser stripes and the continuity of the weld lines in an optimal framework.

Yang et al. [38] use a Convolutional Neural Network (CNN), a simplified Yolo v.3 network [39], for welding defect detection after localizing the weld bead. The authors employ data augmentation by image processing and a GAN network to enlarge the dataset size. Then the Yolo-V3 network is applied to automatically detect and identify weld beads. Last, an update mechanism is set up to enhance the adaptability of the neural networks. Their proposed method can quickly and efficiently detect and identify welded joints.

Normal maps of welding studs are fed to an hour-glass style CNN to create a heat-map for the 2D-localization of keypoints on the stud by Liu et al. [40]. Stud welding is a process by which a metal stud is joined to a metal workpiece by heating both parts with an arc. In this paper, a CNN named Normal Map Regression Network (NMRN) was proposed to estimate the 2D keypoint positions of the studs. The authors proposed a fusion block to make the NMRN work with normal map inputs. The proposed Stud Online Measuring System (SOMS) has the ability to measure multiple studs from one camera position. The resulting SOMS is efficient at inline measurement needed.

Wang and Shen [41] find the welding zone of water-cooled pipes in radiographic images by semantic segmentation with a specifically designed attention mechanism in their CNN. A CNN-based pixel-to-point module is designed by Zou and Lan [42] for finding weld points in the calibration of a robot laser vision system using reinforcement learning. While both traditional machine learning and deep learning techniques have been applied in vision-based measurement systems for welding, we use U-Net [43] as the baseline network for our heatmap generator which remains a common choice for pixel-to-pixel image tasks such as localization, and VGG-16 [44] as a classifier architecture for our discriminator. While these

baseline networks are faster than e.g., Coordinate measuring machines (CMM) [45], they are not efficient enough for online measurement. Also, large amount of labeled samples are needed to train these baseline networks. Therefore, these systems as is are incapable of achieving our objective. In the next section, we will describe keypoint detection approaches close to our research.

3.2 Weld Seam Tracking for Robotic Welding

Computer vision can be utilized to recognize and find the position of welding seams. There are various type of sensors used in robotic welding for performing seam tracking. We review below methods which can be categorized as vision sensor seam tracking [46] [47] [48] and laser vision sensor systems [49] [50]. The vision sensor systems use only CCD camera(s) for seam tracking. Complex algorithms are needed for extracting the weld seam position from the two-dimensional image plane and the algorithms tend to be less accurate than active vision systems. In contrast, our proposed method uses images captured with a structured light sensor. The laser vision sensor consists of two parts a laser projector and a camera. The laser projector produce a stripe which can be scanned by the camera. For traditional algorithm, image processing is a necessary step before extracting the feature point. The process involves median filtering for noise removal, thresholding by the uniform threshold algorithm to extract point of interest, de-noising for removing the spatter, thinning to extract the centre line from the binary image and curve fitting. In contrast, a deep learning based algorithm does not need this extra step.

Visual sensing technology is widely used in weld seam tracking, in particualr, binocular vision is commonly used for welding image processing. Qin et al. [51] propose a cascade algorithm composed of a medium value filter and homomorphic filter to de-noise welding images. It overcomes the problems caused by fuzzy edges and removes blur. An adaptive

threshold arithmetic for segmented images is put forward. The proposed method improves image details.

Tung et al. [46] propose an image-guided mobile robotic welding system for the SMAW process by installing two parallel CCD cameras, a five-axis SCORBOT-ER VII robot and SMAW equipment on a mobile vehicle. SMAW is a manual arc welding process that uses a consumable eletrode covered with a flux to lay the weld. Based on binocular images captured by the CCD cameras, the proposed algorithm compares the disparities between the two images and obtains the welding paths for cracks. The variation in the welding current with their proposed system is much smaller than operated by even senior technicians.

A seam measurement method based on a vision sensor for the space weld seam of a narrow butt joint is proposed by Shao et al [47]. The authors use three laser stripes with different wave length that are projected on the sheet metal, in which two red laser stripes are designed and used to measure the three-dimensional profile of the weld face. The third green laser stripe is used as light source to measure the edge and the centerline of the seam by the principles of passive vision sensing. All three laser stripes are captured and processed in a single image so that the three-dimensional position of the space weld seam can be obtained simultaneously. It provides an effective sensing technology to measure butt joint with no misalignment and width less than 0.1 mm by combining traditional methods. Their proposed method adopts passive vision technology instead of structured light to tackle the problem of sensing in narrow butt joint.

Liu et al [48] proposed a supervised learning algorithm for intelligent robotic welding. A fuzzy classifier is trained using the top ranked data pairs with labels provided by the welder. A supervised Adaptive Neuro-Fuzzy Inference System (ANFIS) model is proposed to correlate the 3-D weld pool characteristic parameters and the welder's adjustment on the welding speed. Their fuzzy inference system employing fuzzy if-then rules can model the qualitative aspects of human knowledge and reasoning processes without employing precise quantitative analyses. The fuzzy modeling can serve as a basis for rules with appropriate

membership functions to generate the stipulated input-output pairs.

Cheng et al [49] proposed an innovative method to detect the dynamic development of the weld pool. A CNN model is applied to train, verify and test the dataset to identify the penetration states such that the welding current can be reduced from the peak to the base level after the desired penetration state is achieved despite variations in manufacturing condition. The authors use dynamic development of the weld pool surface to predict the weld penetration that is unobservable. The authors construct composite images to represent this dynamic development with minimal yet adequate information dimensions. Finally, they use a convolutional neural network to mine the fundamental information from the dynamic development to predict the weld penetration. A single stripe laser pattern is generated to represent penetration information.

Yan et al. [50] proposed a seam tracking system based on laser structured light. Authors considered V joints for computing the feature point. They regard the center point of the weld groove as the characteristic point and the coordinates for this point are obtained by considering the coordinate values of the main line and the laser stripe. The hough transform algorithm is used to detect straight lines in the image and find the main line of the laser stripe in the system.

In this section, we discussed algorithms that utilize only vision sensor(s) and laser assisted vision sensors for welding seam tracking. There are also other types of sensors used in robotic welding for performing seam tracking. In this thesis, we do not propose a new sensing method by a novel method to find the keypoint for weld seam tacking in structured light images but we propose a new keypoint detection method in those images.

3.3 Keypoint detection

Finding the position of the seam point is a form of keypoint localization. Keypoint localization has been considered in numerous computer vision and vision-based measurement tasks. Keypoint localization is an essential part of automatic measurement applications in industrial manufacturing. Most use cases for keypoint localization aim to recognize and localize objects or parts in order to fulfill manipulation tasks. One of the most important application of keypoint detection that has attracted a lot of attentions from researchers is Human Pose Estimation (HPE). HPE has been developed for decades and aims to obtain the posture of a human body from given sensor inputs. Based on the different problem formulations, deep learning-based HPE methods can be categorized into regression-based or detection-based methods [52]. The regression-based methods directly map the input image to the coordinates of body joints which is very difficult since it is a highly nonlinear problem. Especially learning only one single point lacks robustness. Detection-based methods treat the body parts as image patches or heatmaps of joint locations by a 2-D Gaussian distribution centered at the joint location. The latter methods are able to extract dense pixel information from a small region representation which leads to stronger robustness [52]. This is one of the reasons we convert the coordinates of welding seam joints to heatmap representations as a preprocessing step, then mapping the input images to the heatmap representations by the generator.

Detection-based methods are developed from body part detection methods. However, body part detection methods are usually sensitive to the complexity of the background and to body occlusions. As a result, more recent works employed a heatmap to indicate the ground truth of the joint location [53, 54] and to provide more supervision information to facilitate the training of CNNs. Currently, there are promising approaches regarding generic solutions for the visual detection-based on Deep Neural Networks using monocular cameras with intermediate supervision [15]. This stacked hourglass introduced a novel and intuitive architecture and improved on all previous methods in HPE. It is called a

stacked hourglass network because the network consists of steps of pooling and upsampling layers, and it looks like an hourglass. The design is motivated by the need to capture information at every scale. While local evidence is essential for identifying features, a final pose estimation requires global context. The network performs repeated bottom-up, top-down processing with intermediate supervision. It also uses skip connections to preserve spatial information at each resolution and passes it along for upsampling.

Wei et al. [55] proposed the Convolutional Pose Machine (CPM) with intermediate input and supervision which learns spatial correlations among body parts. CPMs are made up of a sequence of convolution networks which produce a 2D belief map for the location of each part. At every stage in a CPM, the image features and belief maps produced in the preceding stage serve as input in the next stage. The network learns implicit spatial models through a sequential composition of convolutional architectures. It also introduces a systematic approach to designing and training such an architecture in order to learn image features and image-dependent spatial models for structured prediction tasks. Chu et al. [56] proposed multi-resolution attention maps from multi-scale features which utilizes micro hourglass residual units to increase the receptive field. The authors combine the holistic attention model which focuses on the global consistency of the full human body and the body part attention model, which focuses on the detailed description for different body parts. Yang et al. [57] proposed the Pyramid Residual Module (PRM) learns filters to input features with different resolutions. PRMs replaced the residual module of the Hourglass network to enhance the invariance across scales of Deep Convolutional Neural Networks (DCNNs) by learning features on various scales. Sun et al. [58] proposed the deep high-resolution network (HRNet) which set a new standard for keypoint detection and pose estimation tasks in the COCO dataset. HRNet follows a very simple idea by maintaining a high-resolution representation throughout the whole process. The architecture starts from a high-resolution sub-network as the first stage, and gradually adds high-to-low resolution sub-networks one by one to form more stages and connect the multi-resolution sub-networks

in parallel. Repeated multi-scale fusions are conducted by exchanging information across parallel multi-resolution sub-networks through the whole process. Heatmaps are regressed using an MSE loss, similar to our approach.

Chen et al. [59] proposed a GAN-based stacked conv-deconv architecture using two discriminators for distinguishing whether the pose is 'real' and the confidence is strong. Conv-deconv networks are composed of two parts: the first one is a classical convolutional network with subsampling operations which decrease the feature maps sizes, and the second part is a deconvolutional network with upsampling operations which increase the feature maps sizes back to the original input resolution. The model learns the distribution of the human body structures implicitly. The authors use generative adversarial networks to exploit the constrained human pose distribution. The model consists of a generative network and a discriminator which tell whether the predicted pose is geometrically reasonable. Peng et al. [60] proposed another GAN-based augmentation network to generate data augmentations without looking for more data. The authors design an augmentation network (generator) that competes against a target network (discriminator) by generating 'hard' augmentation operations online. The augmentation network explores the weaknesses of the target network, while the latter learns from 'hard' augmentations to achieve better performance.

In general, coordinate regression can be used to estimate keypoints, but recent methods often use heatmap regression [15, 16, 58, 61]. In human landmark estimation commonly many landmarks need to be found on RGB images. As a result, these models use numerous network layers in order to derive powerful feature maps from input images. Particularly, some models stack multiple sub-networks together to form a much deeper network [15, 16] which is computationally costly. However, in order to train these networks, the size of the input has to be down-sampled to a particular size (e.g. 64×64) to feed into the network [14] [15] [58] due to the limitation of GPU memory. This common preprocessing causes the networks to miss useful local features [62]. In contrast, our objective is to

localize a single landmark from each image precisely and our dataset contains only gray-scale images. We cannot obtain the desired localization precision if we apply the previously mentioned methods. However, instead our heatmap generator can output much more accurate results by using high resolution input images but requires only a relatively light-weight network.

3.4 Semi-supervised Learning

Machine learning has become very important in image processing tasks. A frequent problem when applying machine learning methods to these tasks is the lack of labeled data [63]. In contrast, in most of the cases, large sets of unlabeled data are more widely available than labeled data. Manual labeling of the images is an expensive and time-consuming process. Therefore, approaches that incorporate other data and labels are desirable. These approaches include semi-supervised learning, weakly supervised learning and transfer learning. In this thesis, we only focus on semi-supervised learning.

One way to deal with unlabeled data is referred to as Cluster-then-Label methods. It includes those methods that assume a joint probability model $P(x, y) = P(Y)P(x|y)$, where $P(x|y)$ is an identifiable mixture distribution, for example a Gaussian mixture model [64]. Hence it follows a determined parametric model [65] using both unlabeled and labeled data. Cluster-then-label methods are closely related to generative models. Instead of using a probabilistic model, they apply a previous clustering step to the whole dataset, and then label each cluster with the help of labeled data.

Semi-supervised learning [7] uses both labeled and unlabeled samples, typically assumed to be sampled from similar distributions. In SSL scenario, there are two sets of samples: labeled samples D_s and unlabeled samples U. The goal is to use the samples in U to improve the classifier f, where it is constructed only using samples in D_s. Typically, SSL

approaches make additional assumptions linking properties of the distribution of the input features to properties of the decision function [66], [67].

A popular approach to SSL in computer vision tasks is label propagation via self-training. The general idea is:

1. A classifier is first trained on the labeled samples.
2. The classifier is then used on the unlabeled samples in order to generate pseudo labeled samples.
3. Pseudo labeled samples or a subset of these samples are added to the training set to retrain the classifier.
4. This process can be repeated several times until the classifier reaches its optimal state.

Various semi-supervised methods have been pursued including pseudo-labels [68], generative models using auto-encoders [69] and generative adversarial networks [70, 71], and teacher-student networks [72].

Recent methods work by exploiting consistency regularization, entropy minimization and generic regularization [73]. In Mixmatch by Berthelot et al. [73], every batch, every labeled data point is augmented once and every unlabeled data point is augmented K times, where K is a hyper-parameter. The model is asked to predict the class for all the K augmented entries and their average is taken as the prediction for all the K entries. This average is sharpened to minimize the entropy and taken as the final prediction. Augmented labeled and unlabeled data are concatenated and shuffled. Mixmatch [74] combines different paradigms of SSL to achieve excellent performance on CIFAR-10 and STL-10 datasets. The CIFAR-10 dataset consists of 60k 32×32 color images in 10 classes. The STL-10

dataset is an image recognition dataset for developing deep learning algorithms, which contains $1,300$ labeled images and $100,000$ unlabeled images in 10 classes.

Unsupervised Data Augmentation [11] is a recent consistency regularization approach by Xie et al. It applies data augmentation on unlabeled data and enforces the consistency between the augmented unlabeled sample and the sample itself. It achieves state-of-art results on IMDb using only 20 labeled examples. IMDb is an online database of information related to films, television programs, home videos, video games, and streaming content online. It includes cast, production crew and personal biographies, plot summaries, trivia, ratings, fan and critical reviews. The authors introduce a detail to their method called training signal annealing (TSA). First, a standard supervised loss is computed when labeled data is available. Second, a consistency loss is computed between an example and its augmented version with the help of unlabeled data. The objective of data augmentation is to create a novel training data which resembles reality, by applying a transformation to an example without changing its label. They show an ablation study on Yelp-5, where it seems like Unsupervised Data Augmentation (UDA) [11] does not work at all without TSA.

Interpolation consistency training [75] instead uses interpolation between two unlabeled samples to enforce the prediction of mixed 'fake' labels. The authors follow the common practice in the semi-supervised learning literature [74, 76–79] and conduct experiments using the CIFAR-10 [80] and the Street view house numbers (SVHN) datasets [81], where only a fraction of the training data is labeled, and the remaining data is used as unlabeled data. A decision boundary which lies in a high-density region, will cut a cluster into two different classes.

In S4L [82], Zhai et al. train their models on an auxiliary task of predicting rotations simultaneously with classifying images. The authors present the semi-supervised learning classification approach. They assume a data generating joint distribution P(X, Y) over images and labels. The learning algorithm has access to a labeled training set D_l, which

is sampled independent and identically distributed (i.i.d.) from P(X, Y) and an unlabeled training set D_u, which is sampled i.i.d. from the marginal distribution P(X). They have bridged the gap between self-supervised learning and semi-supervised learning by the S4L framework which can be used to turn any self-supervised method into a semi-supervised learning model.

However, the above semi-supervised methods target only classification, and it is hard to see how to extend these methods to image regression problems. We solve an image regression problem effectively in a semi-supervised manner. There are some prior self-supervised methods for facial keypoint detection. Dong et al. [83] utilize a differentiable Lucas-Kanade [84] filter to compute a registration loss as supervision to improve the precision of landmark detectors on both images and videos. The method computes optical flow registration in the forward pass, and back-propagates gradients that encourage temporal coherency in the detector. Our data consists however only of single image input. Our approach is closest to Dong and Yang [72] who propose a self-paced learning algorithm for facial landmark detection from partially labeled samples. The method uses two student networks to generate pseudo-labeled keypoints which are than filtered by a teacher to only accept qualified pseudo-label for further training of the students. However, the labeled samples to train the two-students network initially have to be independent, which means more labeled samples are required for supervised training as for just a single student network. In contrast, our network shares the same number of labeled samples to train both a heatmap generator and two discriminators as teachers. By this architecture, our network requires no additional labeled data.

3.5 Summary

This chapter has presented the related work to this thesis. The chapter commenced with a general overview of machine learning in welding applications. Next, the state-of-the-art

methods of keypoint localization in HPE problems were described. Finally, some previous work related to semi-supervised learning for image classification was presented.

Unlike the different related works cited above, our study aims to develop a comprehensive method that combines all the features together which can satisfy our thesis objective described in Chapter 1. We have selected three primary directions of study for our development. In this thesis, we address these issues by proposing a semi-supervised learning algorithm in the form of a heatmap generator and two discriminators. Our approach can detect the keypoint for a welding system precisely and robustly by using a few labeled samples. We extend the work proposed by Yang et al. [16] to a semi-supervised learning algorithm that leads to a better adaptivity to our dataset requirements.

Chapter 4

Semi-supervised learning key-point localization

Our goal is to locate the intersection of pair of joints of metal plates from input images of size $W \times H \times 1$ (single channel images). There is an existing computer vision system for the task which proceeds by finding two edges and then calculating the intersection of them to locate the coordinate. Although the existing computer vision system provides a sufficiently accurate measurement for the welding task most of the time, there are challenges such as slow running speed, low accuracy, and lack of robustness due to unexpected imaging conditions.

In most of the ML-based key-point localization approaches, the model is trained by a large amount of labeled data. Which is time-consuming and expensive to obtain. Our generator and discriminator architecture do similar jobs as the generator and discriminator in a GAN [85], but we aim to find qualified pseudo-labels in semi-supervised learning by using a different training strategy. The overview workflow of the algorithm is shown in Fig 4.1. In this chapter we describe our semi-supervised key-point localization algorithm in details.

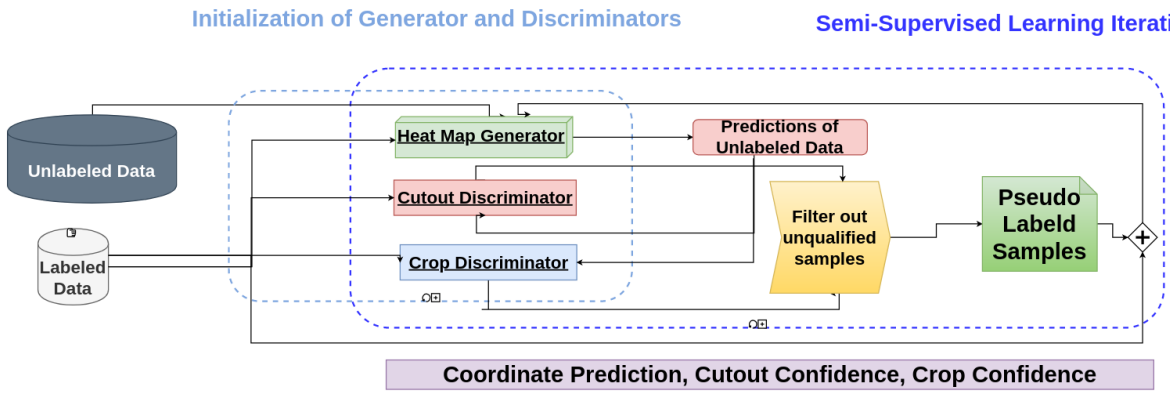


Figure 4.1: The overview of the semi-supervised network workflow.

We address the task by a semi-supervised learning approach including list of components which consisting of the following stages (shown in Fig 4.1):

1. A supervised learning heatmap generator model is trained from few labeled data. The model is capable to locate the approximate intersection.

2. A supervised learning discriminator is trained from the same labeled data. The model can discriminate if a portion of the image contains enough features of the intersection.

3. Another supervised learning discriminator is trained from the same labeled data. The model can discriminate if a modified image contains enough features of the intersection.

4. Pseudo label data generated from qualified predictions are selected according to their scores from the two discriminators.

5. The heatmap generator is improved by retraining from the mix of pseudo labeled and labeled images.

4.1 Heatmap Generator

There are two dominant ways to regress landmark coordinates. One is to regress the coordinates and another way is to regress a heatmap of the landmark, which is obtained by converting coordinates to heatmaps as a preprocessing step. The former method is simple and fast to train, but larger dataset and deeper model are needed to achieve higher accuracy; The latter one can achieve higher accuracy with small dataset and shallow model [86]. The ground truth coordinates are converted to a heatmap which is a visual representation where the hotter location means closer to the ground truth coordinates. There are many ways to create these heatmaps. In our approach, we pick an isotropic Gaussian kernel G (in Equation: 4.1) or kernel density estimation (KDE) as the method to generate heatmap labels from coordinates for each image. The σ determines the width of the Gaussian kernel.

$$Gaussian(\sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{\sum_{m=0}^{1}(p_{k,m}-c_m)^2}{2\sigma^2}}, \tag{4.1}$$

as the method to generate heatmap labels from coordinates for each training image. The width of the Gaussian kernel is determined by σ. Where c_0 and c_1 are the horizontal and vertical coordinates of the keypoint, respectively, and $p_{k,0}$ and $p_{k,1}$ are the horizontal and vertical pixel coordinates of the heatmap, respectively.

We follow the widely adopted pipeline [15, 59, 87] to predict the keypoint using a convolutional network. Since each image is supposed to contain only one intersection point, the output of the generator model is a single channel heatmap where the pixel that has the highest value indicates the location of the intersection. The design of **the heatmap generator** is inspired by an encoder-decoder architecture [88]. The use of this deep neural network (DNN) architecture is motivated by its outstanding results on both classification and localization problems. It is a neural network design pattern which is partitioned into three sections: the contraction, the bottleneck, and the expansion section [89]. There is

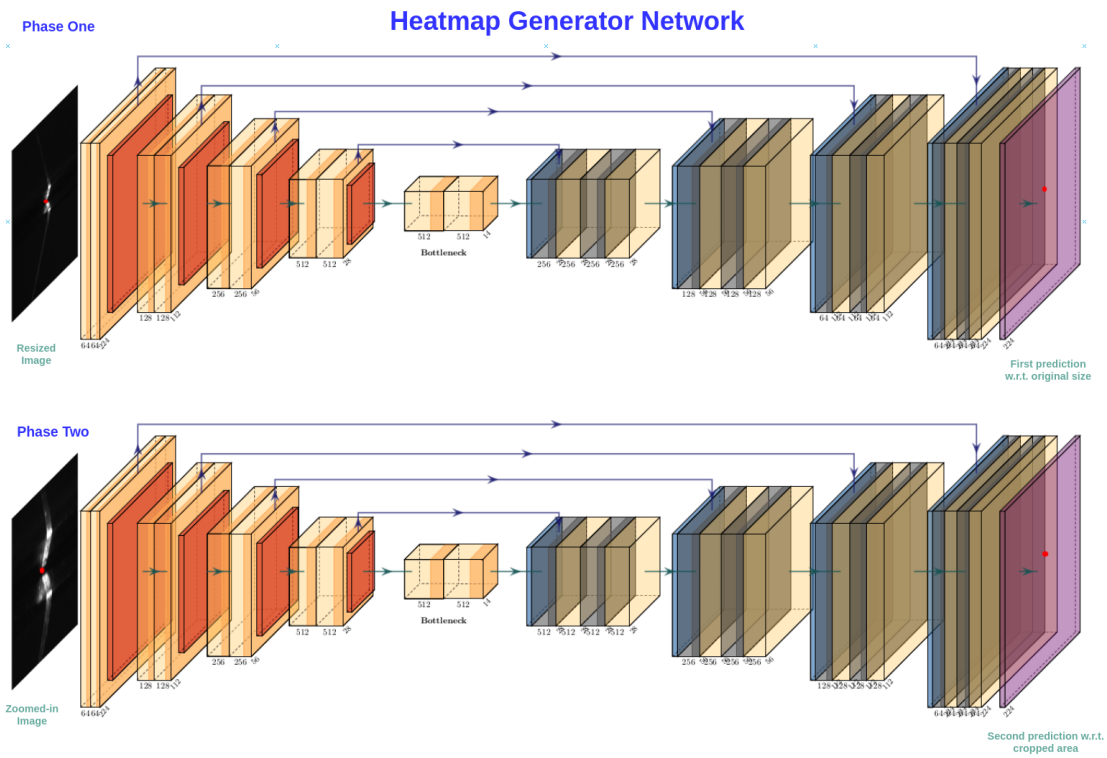


Figure 4.2: Heatmap generator network structure including stage 1 and stage 2.

a nonlinear mapping from the input space to a latent space, also a mapping from input space plus the latent space to the output space by a skip connection. The contraction section is made of many contraction blocks. Each block takes input and applies two 3×3 convolution layers followed by a 2×2 max pooling. Similar to the contraction layer, the expansion section also consists of several expansion blocks. Each block passes the input to two 3×3 CNN layers followed by a 2×2 up-sampling layer. The network reaches its lowest resolution at 14×14 pixels allowing smaller spatial filters to be applied to compare features across the entire space of the image. The general idea of our heatmap generator is as follows: Processing features down to a very low resolution by convolutional and max pooling layers until reaching the lowest resolution. The network begins the top-down sequence of upsampling features across scales by nearest neighbor upsampling of the lower resolution followed by an element-wise addition of the corresponding features while down-sampling. The architecture of the heatmap generator is shown in Fig 4.2.

The structure of our heatmap generator composed by two Unet [43] networks. Two Unet networks are applied identical architecture including the same input and output shapes. The first Unet is the first stage of the generator which takes the resized original image as input and outputs a rough location of the desired coordinate. The second stage of the generator can utilize the prediction from the first stage to crop an attention patch from the image and use it as an input to predict a more accurate coordinate. The structure of the two-stage generator realizes a coarse-to-fine estimation. The second stage of the network can output a more precise location which build on top of the first stage.

The advantage of this architecture is the network can effectively process and consolidate features across scales. This approach for going back and forth between different scales is extremely important for key-point detection problems because preserving the spatial location of features is essential to do the final localization step.

4.2 Improvements by zoom-in

Considerable progress has been made recently in deep learning for landmark detection tasks with the developments in supervised regression models. However, typically these models use very deep neural network structures to achieve the best performance. As a result, most input images have to be downsampled prior to uploading to the GPU. The network has limited capacity to look at details at coarse scale. This causes important information loss which affects the precision of the final prediction. If we increase the resolution of the input image directly, the number of parameters will be increased which increases the computational cost and sometimes even can exceed the CPU or GPU memory. To overcome this problem, we proposed a novel method that can make use of only partial full resolution from the original image which increases only a few parameters but also achieves better precision. The intuition behind our method is from attention-based convolutional neural networks. The model can extract features from a fine-grained attention area. This approach

consist of two identical networks, but each one has its own parameters. The first one estimates a rough location of the keypoint by taking the resized image as input. For the second network, instead of feeding the resized image, a cropped area from the original image centered by the first rough prediction which contains higher resolution used as the input. The second network can be considered as a refinement of the first prediction. The network is shown in Fig. 4.2.

We write the refinement network with stages $s \in \{1, 2\}$ with ψ_s be the generator function and trainable parameters θ_s for each stage. Thus, the output of the heatmap generator can be written as $y_2 = \psi_2(x_2; \psi_1(x_1; \theta_1); \theta_2)$ where x_2 is a full resolution zoom-in of the input image x_1 according to the prediction y_1. Outputs of networks y_1 and y_2 are in the format of heatmaps where the network predicts the probability of the desired keypoint at each and every pixel. Then the L_2 loss is

$$Loss = arg\, min_\theta \sum_{s=1}^{2} \left(\frac{1}{K} \sum_{k=0}^{K-1} ||y_{s,k} - \psi(x_{s,k}|\theta_s)||_2^2 \right) \tag{4.2}$$

where K is the number of pixels in the last layer of the network. We use the same architecture for both coarse and fine stage networks (see Figure 4.2). We train both stages of the heatmap generator by minimizing the L_2 distance between the prediction and ground truth heatmap. In order to train the second stage of the generator, we crop an area from the original image centered on an uniform random offset from the groundtruth coordinate. Fig. 4.3 show an example of the input image with groundtruth, zoom-in attention area, and heatmap representation.

The details about generating labels for training the second stage of the generator is as:

1. Randomly generating $\Delta d_1, \Delta d_2$ within a selected range from -r to r.
2. Adding $\Delta d_1, \Delta d_2$ to x_1^1, x_1^2 and assigning values to c'^1_1, c'^2_1 accordingly. Where (x_1^1, x_1^2) is the ground truth coordinate of the image.
3. Set $c1start$, c_{2start} to be the left and topmost coordinates of the cropped area.

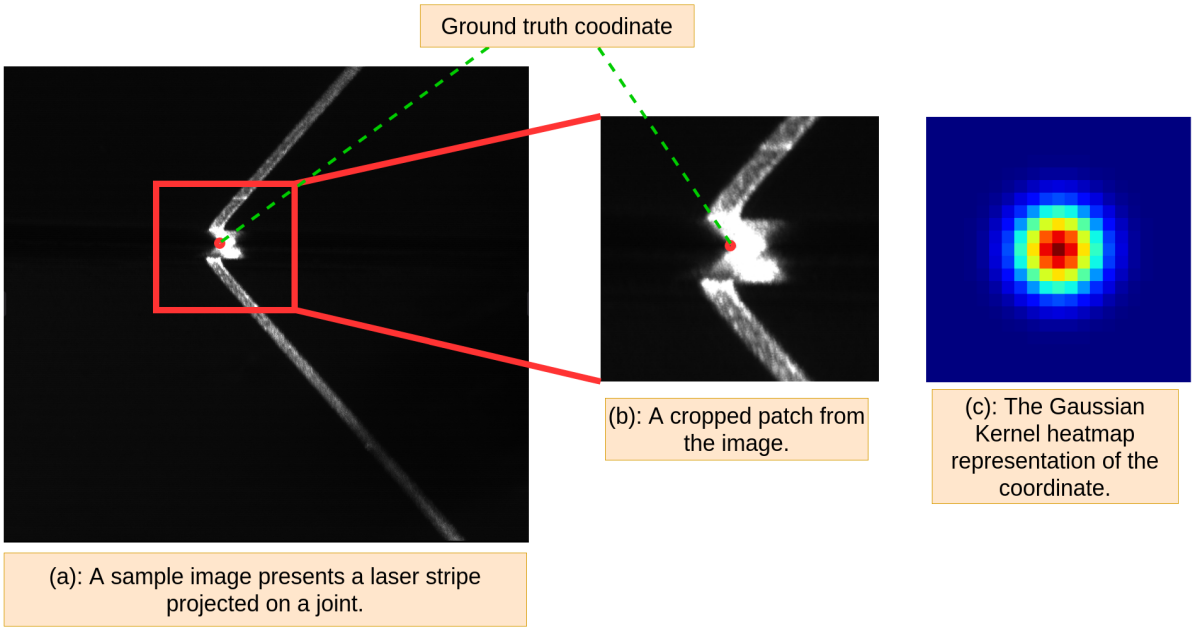


Figure 4.3: (a) A sample image of a joint with laser stripe projection. (b) A zoom-in area from the sample image. (c) The Gaussian heatmap representation of the ground truth coordinate.

4. Initializing c_{1peak} and c_{2peak} by w_{crop} / 2. Where (c_{1peak}, c_{2peak}) is the coordinates of the peak point w.r.t. cropping area.

5. Checking if c'^{1}_{1} - w_{crop} / 2 < 0 then $c_{1peak} = c_{1peak} + (c'^{1}_{1} - w_{crop}/2)$. Where w_{crop} is the width of the input size of the second part of the generator. Where c_{1peak} is the center of w_{crop}. Also, c_{1start} set to 0.

6. Checking if c' + w_{crop} / 2 > w_{image} then $c_{peak} = c_{peak} + (c' + w_{crop}/2 - w_{image})$. Where w_{image} is the width of origin image. Also set c_{start} to c'c_{start} - (c' + w_{crop} / 2 - w_{image})

7. Checking if c'^{2}_{1} - w_{crop} / 2 < 0 then $c_{2peak} = c_{2peak} + (c'^{2}_{1} - w_{crop}/2)$. Where w_{crop} is the width of the input size of the second part of the generator. Where c_{2peak} is the center of w_{crop}. Also, c_{2start} set to 0.

8. Checking if c' + w_{crop} / 2 > w_{image} then $c_{2peak} = c_{2peak} + (c'^{2}_{1} + w_{crop}/2 - w_{image})$. Where w_{image} is the width of origin image. Also set c_{2start} to c_{2start} - (c' + w_{crop} / 2

- w_{image})

9. Cropping the patch from the origin image as the input of the second stage of the generator.

10. Generating the heatmap according to c_{1peak}, c_{2peak} as the ground truth of the second stage of the generator.

In general, the first stage of the convolutional neural network takes down-sampled images and output heatmaps with approximate predictions. By our two-step learning approach, the fine-grained features learned by the convolutional neural network in the second stage improves the performance of predictions.

4.3 Discriminators

We build our two discriminators adopted from feature extraction layers in VGG-16 [44] and append 2 fully connected layers. The network structures for both discriminators also can be seen as the encoder from an Autoencoder [88] structure. Instead of outputting 1000 classes as in ImageNet, we simply output 1 scalar which has a value between 0 and 1. Each discriminator network takes an image as input and outputs the confidence score for the input image which is either a cropped area from the original image or the image with an area cut out and replaced by some random number. Both discriminators can examine the quality of the prediction from the heatmap generator, but from two different perspectives, which are detailed below. The two discriminators complement each other. By using the combination of two discriminators, it gives promising results on evaluating predictions for unlabeled samples. In the following two sections, we will demonstrate more details about the design of cutout and crop discriminators.

4.3.1 Discrimination by cutout

The cutout Discriminator removes a fixed size area from the image centered according to the predicted keypoint from the heatmap generator. It is inspired by Graham [90]. The idea of this discriminator is that we transfer a regression problem to a binary classification sub-problem. We want to have a validating network which can test if an image contains enough features in a certain area centered on the keypoint. As a result, there are two possible cases: first, if the coordinate is a good prediction, then after the informative area has been removed from the original image, the remaining part of the image should not contain enough features to locate the keypoint. As a result, these images should be marked as negative. On the other hand, if the remaining image still contains enough features to localize the keypoint, then it should be marked as positive.

We gain an extra benefit of being able to create additional label data to train the discriminator by using various differently placed cutouts with a fixed amount of labeled data from the regression problem. We add some uniform random variable $\Delta d_1, \Delta d_2$ to the ground truth coordinate. Then evaluate the Euclidean Distance (ED) between the original and modified coordinate and assign a label according to a maximum distance from the keypoint d_{max},

$$L_{cutout} = \begin{cases} 1 & \text{if } \sqrt{\sum_{m=0}^{1} (\Delta d_m)^2} \geq d_{max} \\ 0 & \text{if } \sqrt{\sum_{m=0}^{1} (\Delta d_m)^2} < d_{max} \end{cases}. \tag{4.3}$$

By applying the above method, for each labeled image we can generate N images with key features are cutout which labeled as 0. Where N is a hyper-parameter indicates the number of labeled image to geerate. We set N to 2 when we train the cutout discriminator. Also, we can generator $N-1$ images remain key features which labeled as 1. Since the original image is including key features, it can be labeled as 1. As a result, we have N

Figure 4.4: Modification of input images for the cutout discriminator networks based on the keypoint c_m and a random offset Δd_m. Red frames correspond to positive samples while yellow show negative samples.

images labeled as 1 and N images labeled as 0. The balanced positive and negative samples helps better training the cutout discriminator.

4.3.2 Discrimination by cropping

The crop discriminator has the same structure as the cutout discriminator. However, the input of the network uses only a small cropped area containing the most important features for the intersection. Base on the nominated cropping area from the image by the heatmap generator, the network is able to categorize if the cropped area contains enough features of target the intersection. From the labeled data, we can crop the area which is centered on the ground truth coordinate and label the sub-image as positive samples. On the other hand, cropping an area with randomly selected coordinates can be labeled as negative samples. The strategy for labelling this classification problem is shown in Eq. 4.4. The same advantage as with the cutout discriminator is that we can generate many invalid samples based on limited number of labeled data. However, we have to keep in mind that generating too many invalid samples may cause imbalance for the classification problem. Therefore, we keep the ratio of number of examples from positive and negative samples in order to better train the discriminator network. There are three types of images while

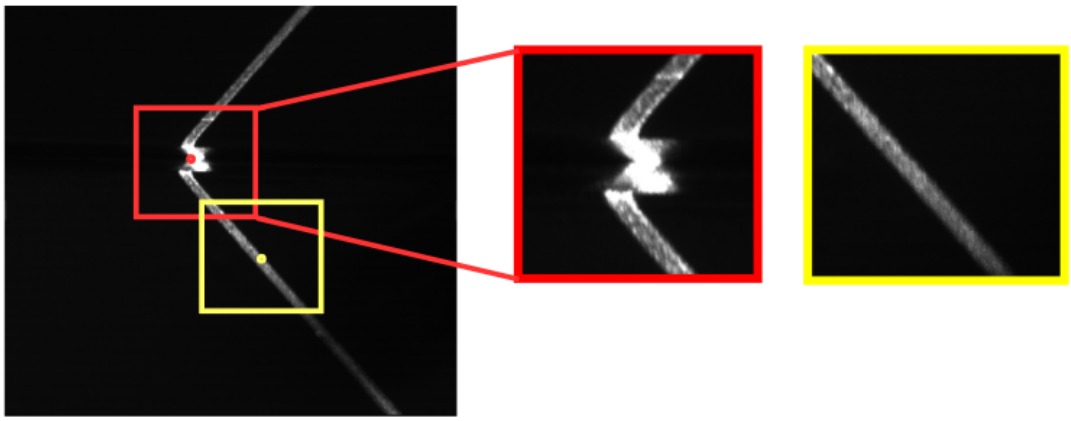


Figure 4.5: Modification of input images for the crop discriminator networks based on the keypoint c_m and a random offset Δd_m. Red frames correspond to positive samples while yellow show negative samples.

training this network:

- The Center crop from a valid image based on the ground truth coordinates labeled as valid since the area contains enough features of the intersection of two sheets.
- A random crop from a valid image with the distance between the center of the cropped area and the ground truth larger than a threshold T is labeled as invalid since it does not contain enough features of the intersection.
- A random crop from an invalid image is labeled as invalid since it does not contain any intersection features.

$$L_{crop} = \begin{cases} 0 & \text{if } \sqrt{\sum_{m=0}^{1} (\Delta d_m)^2} \geq d_{max} \\ 1 & \text{if } \sqrt{\sum_{m=0}^{1} (\Delta d_m)^2} < d_{max} \end{cases}. \tag{4.4}$$

Each discriminator has its own strategy to extract different types of features. So, they can capture outliers from their own perspective. The outliers are predictions are far away

from the groundtruth in terms of the Euclidean distance. As a result, the cutout and crop discriminators can supplement each other. By using two discriminators, we create a classification task as an intermediate step when we try to solve a regression problem. Formally, we denote the loss function of both discriminator networks as

$$Loss(D(x|\theta), L) = \frac{1}{N_D} \cdot \sum_{i}^{N_D} (logD(f(x^i)) + (1 - D(f(x^i)))). \tag{4.5}$$

Where f indicates a modification function on the sample image x using the random offset Δd_m that outputs positive and negative images labeled according Eqn. 4.3 and 4.4, respectively. Where $logD(f(x^i))$ refers to the probability that the positive sample is correctly classified. In contrast, $log(1 - D(f(x^i)))$ refers to the probability that the negative sample is correctly classified. The number of training images for the two discrimators is N_D.

4.4 Discussion

SSL has been the core of ML trying to exploit labeled and unlabeled data. Our proposed method uses unlabeled data in the learning process improving the predictive performance of the implemented SSL algorithms. Although SSL is not well studied compared with SSC, our semi-supervised learning algorithm aims to increase the accuracy of the predictions with a minimum number of labeled samples required to train a model.

In this chapter we provided the structure of our proposed semi-supervised learning network that includes a heatmap generator and two discriminators. We first convert our ground truth from (x, y) coordinates to heatmap representations in Gaussian Kernel format. Secondly, we train our heatmap generator and two discriminators by labeled images in supervised learning manner. Third, we produce predictions for all unlabeled images by the heatmap generator. Next, we generate pseudo labeled images from all the predictions

by two discriminators. We retrain the heatmap generator by pseudo labeled and labeled images. We run this semi-supervised learning iteration multiple times until no further improvement of the heatmap generator. Moreover, we presented a heatmap generator composed of a two-step learning approach. We will describe the experimental results and evaluation in Chapter 5.

4.5 Summary

In this chapter, we introduce the overall architecture of our proposed semi-supervised method. We describe the details of our heatmap generator and two discriminators. The heatmap generator is composed with two Unet [43] networks. The first stage of the generator predicts a rough location of the keypoint and second stage can produce more precise predictions by using the fine-grained images. The two discriminator can evaluate predictions from the generator, which filter out unqualified images from all predictions.

Chapter 5

Experimental results

In this chapter, we demonstrate steps to train the heatmap generator and two discriminators respectively. We conduct our experiments in five different groups. Each group of experiments aims to show the performance of a unique perspective. The first experiment compares our semi-supervised learning approach with the state-of-the-art methods w.r.t. the precision of predicted coordinates. In the second experiment, we show that the results of our algorithm effectively improve the performance of the heatmap generator by using various numbers of labeled samples. In the third experiment, we demonstrate that our semi-supervised method is robust and accurate by randomly selecting a fixed number of labeled samples. The fourth experiment demonstrates the efficiency of two discriminators in terms of detecting and removing outliers from predictions of unlabeled samples. The last experiment shows the improvements from our zoom-in technique in the performance of the generator by exploiting full pixel information of input images.

5.1 Experimental environment

5.1.1 Environment setup

The hardware used for the experiments is an Intel-based PC equipped with NVIDIA GPU. The system has an Intel Core i7-8700K CPU, 16 GB system memory and 512 GB Solid State Disk. The graphical adapter is an NVIDIA Geforce RTX 3080 which is a high-end GPU powered by the Ampere micro-architecture. There is 10 GB GPU-dedicated memory installed on this adapter to enable training of large neural network.

The software framework we used is PyTorch [91] version 1.8.0 which is a popular deep learning framework backed by Facebook, Inc. Another popular deep learning framework is Tensorflow [92] backed by Google LLC. We deployed our implementations on GNU Linux version 19.04. The GPU driver $V455.23.04$ and CUDA toolkit $V11.1$ are installed to support GPU acceleration.

5.1.2 Data Description

We have obtained our data from EDI Inc. which manufactures electrical enclosures. All the images are grayscale and of size of 1280×1024. Fig. 1.1 shows example images from the data set. The laser stripe projection is clearly visible. Different types of joint shapes generate different stripe patterns due to noisy reflections, varied exposure and different metals. All images are taken from the same angle and orientation which means the projection of the laser stripe is always towards a fixed direction for every image (see Fig. 1.2). We have 88,231 unlabeled images and we have hand-labeled 8,342 images for our experiments. Depending on the specific experiment, we utilize fewer of the labeled images to investigate how many labeled images are required by our method. All labeled images are annotated by the horizontal and vertical pixel coordinates of the starting point for the welding seam

which is the keypoint to be identified by our method. Unlabeled images include invalid images which we define as images where the location for the seam is not visible. These images occur at a ratio of approximately 1:9 and are easily identified by the discriminators during semi-supervised training.

The number of samples allocated for training and validation will vary with each experiment because we will investigate how many labels are needed. Moreover, these training and validation samples are randomly selected from all labeled images and the order is always shuffled. However, we keep the testing samples to the same 927 images in order to maintain the consistency of the testing results. These testing samples independent of training and validation samples which make sure the testing data never contribute to the training of model. Also, we reserve the same ratio for samples for training and validation as $9:1$, respectively. The advantage of sustaining the ratio is that we can observe the changes of performance while we change the total number of labeled samples used for semi-supervised learning algorithm. For all the experiments, we isolate 927 labeled images for testing. These images are absolutely not involved for training. Moreover, the testing dataset including variation scenarios such as bent edge, blurred intersection, intersection with gap, overexposed images, and reflected noise etc.

Training on all labeled samples

To best evaluate our semi-supervised method fairly, we divide our labeled samples into 3 groups: 7000 for training, 415 for validation, and 927 for testing. We find that using a batch size of 10, 15, and 20 leads to very similar performance. We determine the number of training steps and the learning rate schedule by the batch size for labeled images.

Training on small amount of labeled samples

In order to prove that out semi-supervised method is efficient on different labeled sample sizes consistently, we conduct our experiments on 200, 100, 50, 20, and 15 randomly selected labeled samples to train the heatmap generator.

5.1.3 Metric and evaluation.

There are several statistical metrics that have been used to evaluate the predictive efficiency of models. Amongst them, Mean Square Error (MSE) is predominantly used to measure the average error. We pick Mean Square Error (MSE) as our objective function for the heatmap generator as it can produce the closest heatmap to the target heatmap. Where MSE for our two stage generator is defined as

$$MSE(y|x, \theta_{1,2}) = \frac{1}{K} \sum_{k=1}^{K} (y_k - \psi_2(x_2; \psi_1(x_1; \theta_1); \theta_2)_k)^2. \tag{5.1}$$

Note that x is the full resolution input image where x_1 is the lower resolution input to the first stage and x_2 the zoom-in derived from the prediction of ψ_1. The peak value of the heatmap outmap of ψ_2 corresponds to the predicted keypoint but the other pixel values of the heatmap are of no importance in our application. Appropriately, to quantitatively evaluate the performance of the generated heatmap, we use the Euclidean Distance (ED) of the peak from the groundtruth coordinate to estimate the actual quality of the prediction. Hence, we report the Mean Euclidean Distance (MED) error.

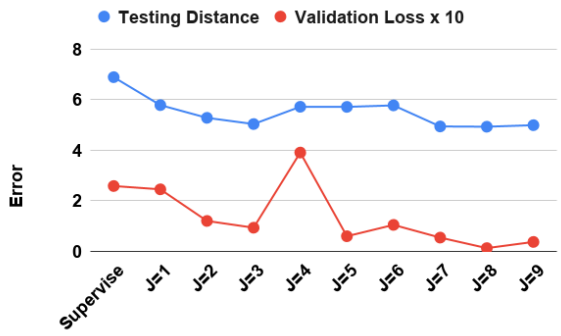
5.2 Training strategy

5.2.1 Generator training

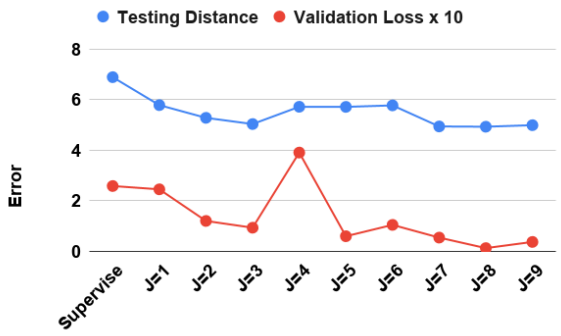
We separate our labeled data in three groups for all the experiments, namely training, validation and testing. Specifically, the testing dataset is not expected to be available when training the model on other data. Since we evaluate the performance of the heatmap generator on the testing dataset, it is really difficult to select the optimal model during the training stage. In order to overcome this problem, we find a heuristic technique which can provide an approximate solution. Based on extensive experiments, we find that there appears to be a correlation between the validation loss error and testing ED error. Figure 5.1 depicts a direct proportional relationship between the validation loss error and ED error in testing when we apply our semi-supervised approach on three different groups of independent training and validation datasets. We conduct a 3-fold cross-validation on 15 random selected labeled images. By observing the fluctuation generated during the model training stage, when the validation error goes up or down (red line in Fig. 5.1), the ED error for the testing samples (blue line in Fig 5.1) shows a similar trend in the same direction. As a result, we select the learnable parameters according to the model running on the validation dataset that gives the lowest ED error.

5.2.2 Cutout discriminator training

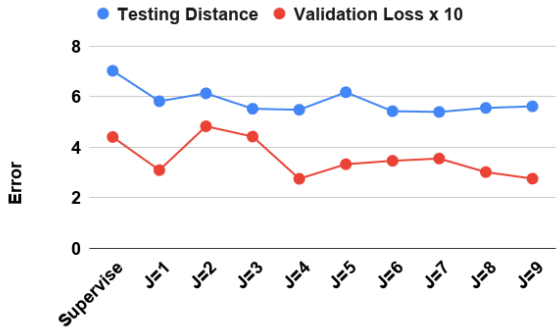
The success of training the cutout discriminator depends on properly replacing the key area in the image with other values for all the input samples before feeding them into the model. There are further important factors that should be considered in training the cutout discriminator which are the size, the shape and the value of the replaced area from the image. First, the size of the part of the desired intersection of two sheets is fixed. We are using a 160×40 patch to replace the selected area. If the size is too big, irrelevant pixel



(a) Fold 1



(b) Fold 2



(c) Fold 3

Figure 5.1: Example correlation between validation loss and testing ED error. Where J is the number of semi-supervised steps in our method.

information is removed which makes the prediction less sensitive. Similarly, if the size of the patch is too small, not enough useful local information is removed, which causes the prediction to be too sensitive. We make the width 4 times larger than the height because we want to make the vertical prediction more sensitive. This is one of the requirements from EDI.

There are at least two options for the replacement pixel values for the cutout area: random values and fixed values (e.g., zeros or ones). The ideal supplement values would be random, because otherwise, the deep learning model would possibly learn to cheat causing the model always to return a constant value. For example, using the same values to replace the selected area would always generate a squared shape within the cutout area independently of where the cutout area is in the image. One of the concerns is that this squared shape can be captured by the network as a feature. However, based on our experiments, replacing the selected pixels by random numbers is effective and gives slightly better performance than by replacing them with a fixed value in terms of the ED error. As a result, we recommend using random values to replace the cutout area.

Images that after the cutout process still include the keypoint are marked as negative samples, and images that no longer contain the keypoint as positive samples, respectively. The positive samples represent acceptable predictions since the intersection area features are successfully removed from the image. Conversely, the negative samples indicate unacceptable predictions due to the features remaining in the image.

5.2.3 Crop discriminator training

Unlike training the cutout discriminator, training the crop discriminator only considers the size and the shape of the cropped area from the image for the preprocessing step. We use a 224×224 pixels square as a cropping patch shape because the shape of the input crop

discriminator network is 224×224. Using the same dimensions simplifies the preprocessing pipeline and avoids the loss of information while downsampling.

All patches including the keypoint are marked as positive samples, and patches excluding the keypoint are marked as negative samples, respectively. The positive samples indicate acceptable predictions since the cropped area contains enough information. Conversely, the negative samples indicate unacceptable predictions.

5.2.4 Generator enhancement by semi-supervised process

The overall workflow for our semi-supervised network is shown in Figure 4.1. Our semi-supervised algorithm aims to progressively improve the performance of the heatmap generator. A key advantage of our algorithm is that three networks (generator and two discriminators) can share the same labeled data to train since they are independent networks, each network has a unique structure and goal. Moreover, during the process of training the two discriminators, we can generate additional training samples by random cutout and cropping from the original labeled images. This training strategy on discriminators reduces the required minimum number of labeled samples. The algorithm details are described in Fig. 5.2 where x^l indicates a labeled image and x^u indicates an unlabeled image. The training algorithms uses n_l labeled images and n_u unlabeled images.

The detailed steps about training semi-supervised networks are described in Algorithm 5.2. While maintaining the overall architecture of the network, there is still some flexibility in the specific implementation. For example, the shape of the Gaussian kernel in the heatmap can be varied; however, once the parameter decided, it should be fixed until finishing the semi-supervised process. the width and height of the mask for the cutout area from the original image used as input for the cutout discriminator can be varied; and, thresholds for both discriminators to disqualify invalid predictions can be adjusted.

Different choices have a moderate impact on the final performance and training of the network.

We conduct experiments to further demonstrate that our two discriminators are able to detect outliers effectively and consistently. In order to do so, we manually label 927 images as testing data which are all containing a desired key-point to be detected. First, we train a heatmap generator by a few training data with the approach described in Section 5.2.1. Second, we use the same training data to train both, cutout discriminator and crop discriminator with the approach described in Section 5.2.2 and Section 5.2.3. Finally, we measure the performance of the predictions by manipulating the threshold of the two discriminators. In Figure 5.3 (a), all the predictions are displayed in the histogram without any discriminator interference. Figure 5.3 (b, c) illustrate using only the cutout discriminator or the crop discriminator, respectively, to filter out disqualified predictions. Not only outliers are removed, but the mean error of predictions is also reduced. Figure 5.3 (d) shows the minimal error achieved by using both discriminators.

Another example demonstrates the importance of two discriminators while detecting outliers from the generator. Figure 5.4 to 5.9 show how the two discriminators influence the predictions from the heatmap generator. Figure 5.4 depicts the distribution of all predictions without any discriminator interference. There are 927 predictions and the mean error is 3.167. Most of the predictions are close to the ground truth except several outliers that are quite far away from the ground truth. Figure 5.5 and Figure 5.6 show the results after employing the crop discriminator with a threshold of 0.5 and of 0.9, respectively. The outliers are detected by the discriminator successfully. Hence, the mean error is reduced from 3.167 to 3.015 and 2.815, respectively. Similarly, Figure 5.7 and Figure 5.8 show the results after employing the cutout discriminator with the threshold of 0.5 and of 0.1, respectively. The mean error is reduced from 3.167 to 3.144 and 2.986, respectively. Lastly, Figure 5.9 depicts the distribution after employing both discriminators with the threshold 0.9 and 0.1, respectively. The mean Euclidean distance is reduced from 3.167 to 2.785,

Require: Labeled data $L = \{(x_i^l)|1 \leq i \leq n_l\}$

Require: Unlabeled data $U = \{(x_i^u)|n_l + 1 \leq i \leq n_u + n_l\}$

Require: Heatmap generator $\psi_{1,2}$ with $\theta_{1,2}$

Require: Discriminator D_{cutout} with θ_{cutout}

Require: Discriminator D_{crop} with θ_{crop}

Initialize $\theta_{1,2}$ by minimizing Eq. 4.2 on L

Initialize θ_{cutout} and θ_{crop} by minimizing Eq. 4.5 on L

Threshold $t_{cutout} \leftarrow 0 \ldots 1$

Threshold $t_{crop} \leftarrow 0 \ldots 1$

$J \leftarrow$ maximum # steps

for $j = 1$ to J **do**

Predict $y_{1,2}^j$ on U using $\psi_{1,2}$, and denote U with its pseudo labels as U'

Compute the confidence of each prediction $y_{1,2}^j$ for U' using D_{cutout} and D_{crop}, respectively

$L' \leftarrow$ qualified samples from U' determined by D_{cutout} and D_{crop} with t_{cutout} and t_{crop}, respectively.

Retrain $\theta_{1,2}$ on $L^{j+1} = \text{L} \cup \text{L'}$ by minimizing Eq. 4.2

end for

return Generator $\psi_{1,2}$ with optimized parameters $\theta_{1,2}$

Figure 5.2: Overall Algorithm.

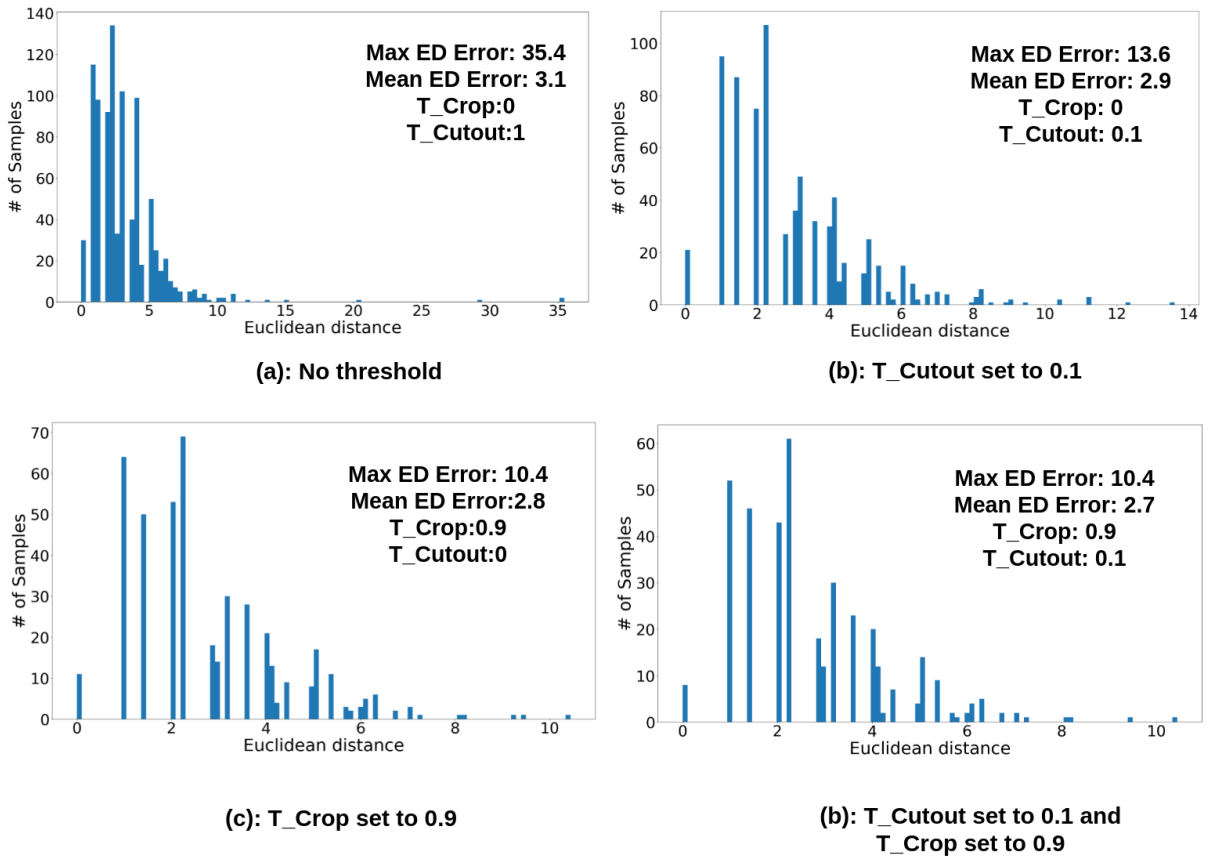


Figure 5.3: Error histograms of predictions from the heatmap generator while tightening thresholds of the cutout and crop discriminators. The x-axis indicates the Euclidean distance between the prediction and the ground truth. The y-axis indicates the number of predictions. (a) All predictions ($t_{cutout} = 0.0, t_{crop} = 1.0$). (b) $t_{cutout} = 0.1, t_{crop} = 0.0$. (c) $t_{cutout} = 0.0, t_{crop} = 0.9$. (d) $t_{cutout} = 0.1, t_{crop} = 0.9$.

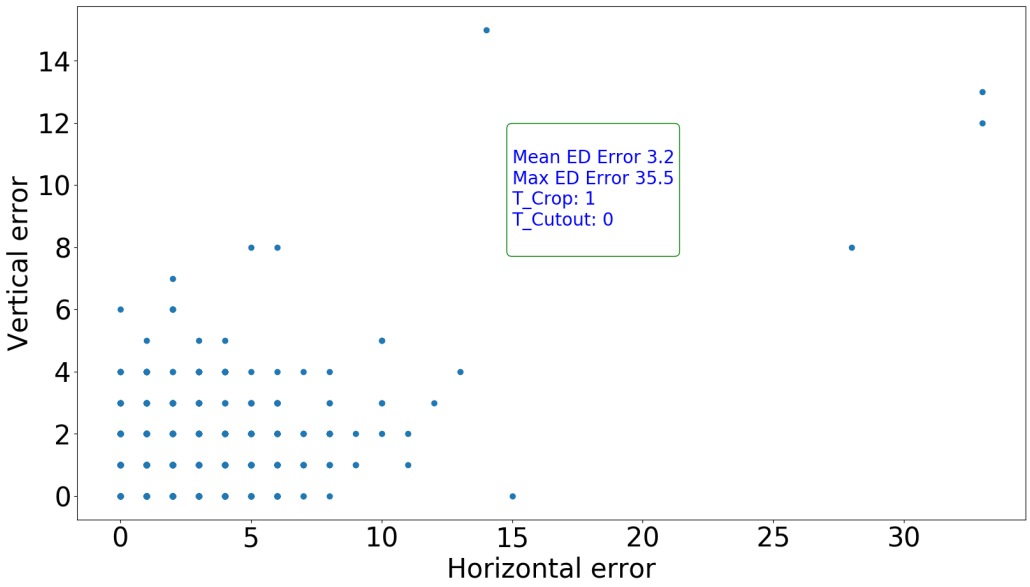


Figure 5.4: The distribution of all the predictions. (Note that images after cutout process labeled as 1 if key feature remains. So the threshold for cutout set to 1 accepts all the predictions. Images after crop process labeled as 1 if key feature remains. So the threshold for crop set to 0 accepts all the predictions.) The horizontal axis indicates how far away the prediction from the ground truth in terms of the distance in pixel. The vertical axis indicates how far away the prediction from the ground truth in terms of the vertical distance in pixel.

which gives the lowest error in all the cases.

5.3 Experimental results

5.3.1 Comparison with the state-of-the-art

Due to the fact that our task is not a common problem in the field of supervised learning and semi-suerpvised learning. The most related field to our task is Human Pose Estima-

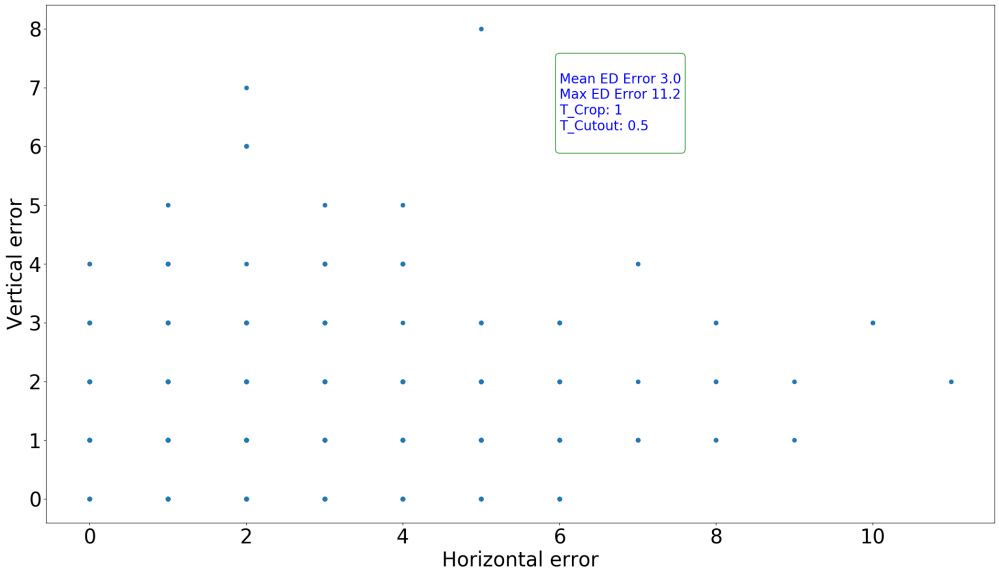


Figure 5.5: The distribution of the predictions with $T_{crop} \geq 0.5$.

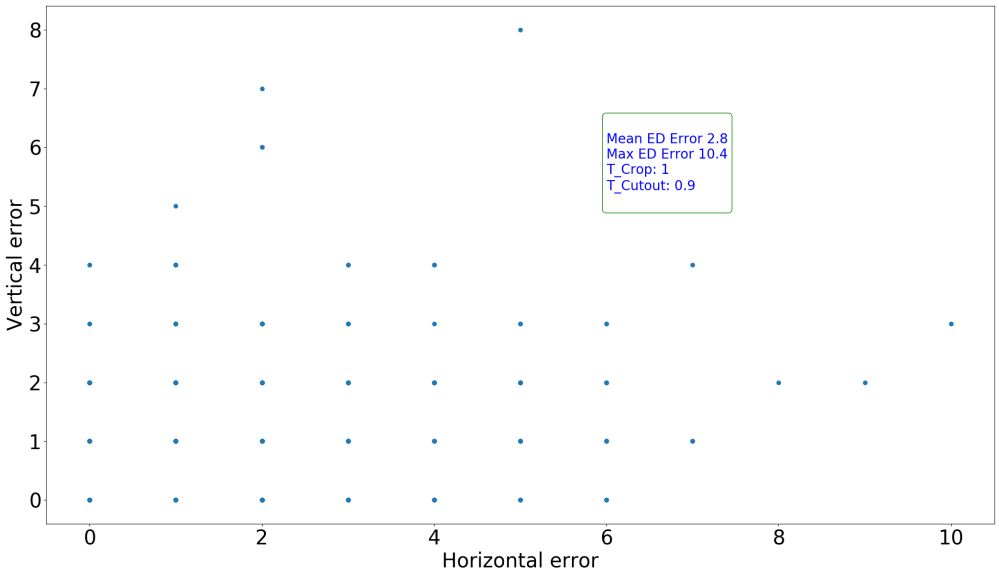


Figure 5.6: The distribution of the predictions with $T_{crop} \geq 0.9$.

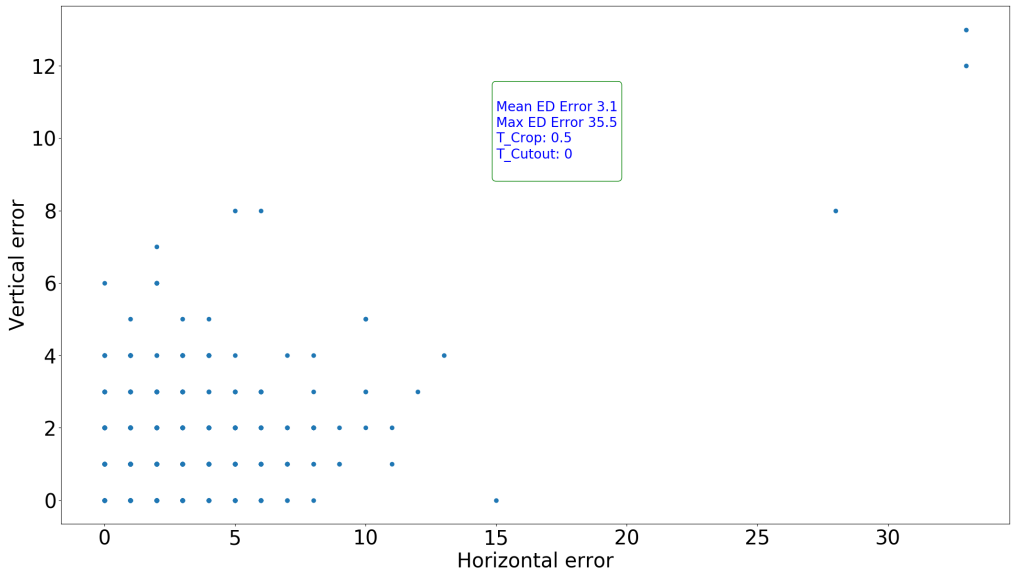


Figure 5.7: The distribution of the predictions with $T_{cutout} \leq 0.5$.

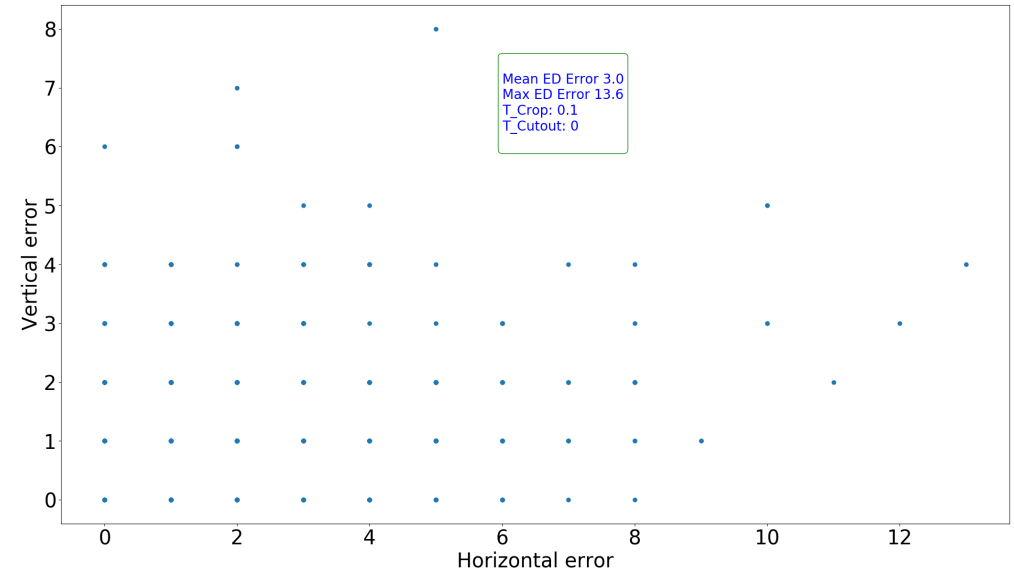


Figure 5.8: The distribution of the predictions with $T_{cutout} \leq 0.1$.

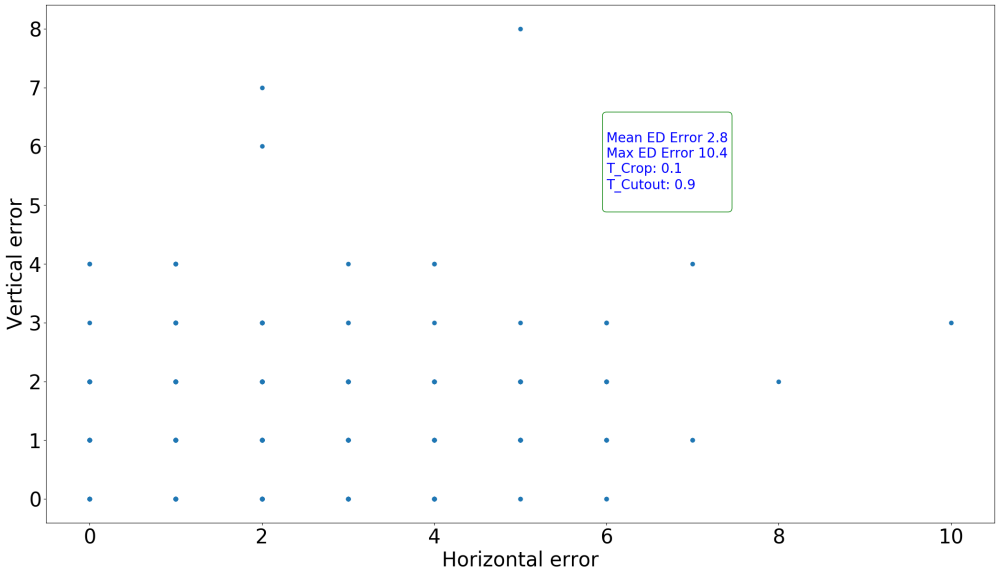


Figure 5.9: The distribution of the predictions with $T_{crop} \geq 0.9$ and $T_{cutout} \leq 0.1$.

tion (HPE). We pick three models to compare with in order to estimate the performance of our proposed method. The Simple Baseline provides [87]an effective baseline method in key-points localization tasks. It is helpful for evaluating our method in the field of HPE. The stacked Hourglass Network [15] and HRNet [58] are two benchmarks in HPE. Our first set of experimental results presented in Table 5.1, compares the landmark localization performance with a Stacked Hourglass Network [15], a Simple Baseline [87], and HRNet [58]. We run experiments by using 7000 labeled images for training, 415 labeled images for validation, and 927 labeled images for testing. We observe that our lightweight network is the best fit for the task at hand, yielding the lowest error of all methods considered. The average ED error is about 4 pixels less with our network than with the Hourglass [15] net with any of one, two and eight stacked hourglass models. We found HRNet [58] to perform similar to the Stacked Hourglass method, while the Simple Baseline performed worse. Using all labelled samples in the training set is sufficient for training our network. This can be seen as there is no significant improvement by applying our semi-supervised technique for three iterations. Next, we will report results with a reduced number of samples that

Network	Ours				Others		
	Supervised	$J = 1$	$J = 2$	$J = 3$	Stacked Hourglass [15]	Simple Baseline [87]	HRNet [58]
Labeled Images	7415	7415	7415	7415	7415	7415	7415
Unlabeled Images	0	88231	88231	88231	0	0	0
Input Size	224x224	224x224	224x224	224x224	256x256	256x256	256x256
Block Size	224x224	224x224	224x224	224x224	64x64	64x64	64x64
Euclidean Distance	**3.139**	**3.127**	**3.263**	**3.199**	7.471	8.244	7.401

Table 5.1: Comparison with state-of-the-art supervised methods. The number of semi-supervised steps in our method is J, (see Algorithm 5.2). Given a large number of labeled training images, semi-supervised steps do not improve model fit further.

demonstrate the ability of our semi-supervised technique to achieve better results than state-of-the-art supervised methods.

5.3.2 Comparison with different numbers of labeled data

We conduct experiments with different number of samples selected randomly from all labeled data and report test results on the 927 testing samples. The test samples are strictly used for testing and our methods uses only the training data for fitting. The details of the experimental results are shown in Table 5.2. In the first group of data, we randomly select 200 labeled samples and separate them by ratio of 9 : 1 for training and validation, respectively. The mean ED error is 5.278 pixel after supervised learning. By applying our semi-supervised process three times consecutively, the mean ED error can be reduced to 4.328, 3.968 and 3.885, respectively. We then reduce the total number of labelled data for training and validation to 100, 50 and 20, respectively, while keeping the same ratio for training and validation. As expected the error in the supervised step increases with the reduction in the number of labelled samples, however, the semi-supervised steps are able to reduce the error even with just 20 labeled images for training and validation. With 20

#Labeled (#training / #validation)	#Unlabeled	#Testing	Supervised	Semi-supervised Steps		
				$J = 1$	$J = 2$	$J = 3$
200 (180 / 20)	88231	927	5.278	4.328	3.968	3.885
100 (90 / 10)	88231	927	6.091	4.634	5.078	3.903
50 (45 / 5)	88231	927	7.718	6.213	5.256	4.245
20 (18 / 2)	88231	927	18.598	8.646	5.560	5.606

Table 5.2: Effect of number of labeled samples on testing error.

labeled after three semi-supervised training steps the error is reduced from a mean ED of 18.598 to 5.56 which is not quite as low as when using 200 labelled samples but it is still well below the Stacked Hourglass and HRNet methods (cf. with Table 5.1). Figure 5.10 summarizes the performance of our SSL approach demonstrating a consistent benefit in terms of error over supervised training independent of number of labelled samples. As to be expected, the benefit of semi-supervised training is largest when the number of labelled samples is small.

5.3.3 Comparison with random labeled data

We investigate the robustness of our approach with a small number of labeled images by training our semi-supervised method with different randomly selected label images. We randomly pick 3 groups of 15 labeled samples, and split each group randomly into 10 samples for training and 5 samples for validation. We evaluate the performance of each model again with the same testing dataset of 927 samples. The experimental results are shown in Table 5.3. The same number of labelled samples lead to similar performance after a sufficient number of semi-supervised training steps. We also run 3-fold cross validation on each 15 samples, and we end up with consistent results. Observing the error on the validation and on the testing dataset, we can conclude that there is a positive correlation

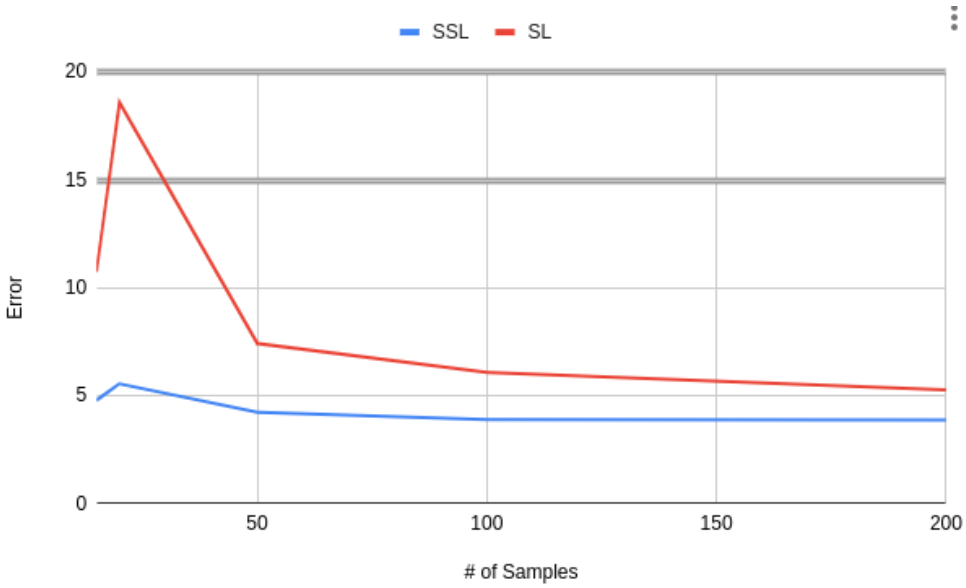


Figure 5.10: Performance of Semi-supervised learning vs. Supervised learning. The horizontal axis indicates the number of samples are used for both training and validation. The vertical axis indicates the mean Euclidean distance between the prediction and the ground truth.

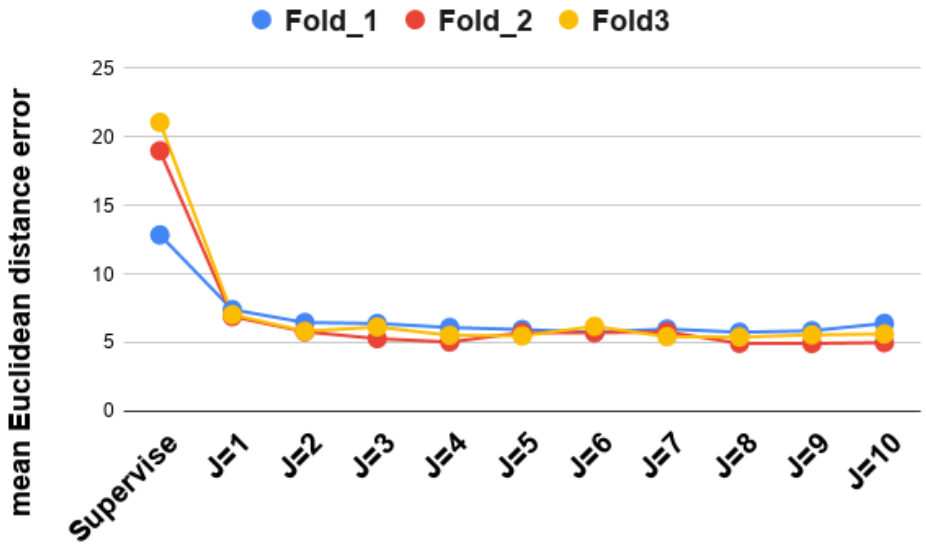


Figure 5.11: Performance of 3 groups of 15 labeled samples. We train our model by 15 labeled images (10 for training and 5 for validation) by 3-fold cross validation. Where J indicates the number of semi-supervised steps in our method. The figure depicts the performance of models running on 927 testing data.

between the validation loss and the testing error. Figure 5.11 demonstrates that training a model with the same number of random labeled samples can be expected to lead to similar performance after enough semi-supervised training steps.

The Fig 5.11 and Table 5.4 shows that the performance of the heatmap generator can be significantly improved by utilizing semi-supervised technique. After several semi-supervised iteration, the error drops from supervised learning stage dramatically and constantly. The accuracy of an enhanced heatmap generator trained by 15 random labeled samples is very close to a generator trained by 7415 labeled samples.

Set	#Labeled (#training / #validation)	#Unlabeled	#Testing	Supervised	Semi-supervised Steps							
					$J = 1$	$J = 2$	$J = 3$	$J = 4$	$J = 5$	$J = 6$	$J = 7$	$J = 8$
a)	15 (10/5)	88231	927	15.252	10.401	6.101	6.063	5.283	5.732	5.040	5.067	4.890
b)	15 (10/5)	88231	927	10.756	8.387	6.360	6.164	5.518	5.282	4.713	4.874	5.574
c)	15 (10/5)	88231	927	12.853	7.401	6.464	6.392	6.098	5.952	5.748	5.988	5.754

Table 5.3: Robustness of testing error with 15 labeled samples (10 for training and 5 for validation).

	Supervised Training without Zoom-in	Supervised Training with Zoom-in
Labeled Images	7415 (7000 / 415)	7415 (7000 / 415)
Unlabeled Images	88231	88231
Euclidean Distance	3.545	**3.139**

Table 5.4: Effect of zoom-in. Compare our method with Zoom-in feature removed.

5.3.4 Ablation study of Zoom-in

In order to quantify the effect of Zoom-in, i.e., stage two of our generator, we conduct two supervised training experiments by the same samples and the same hyper-parameters setups. The training samples including 7000 training images, 415 validation images and 927 testing images. We first train a generator with only phase one. Then we train another generator with two-stage structure. The result shows the importance of the second stage of the generator (see Tab. 5.4). Based on the observation of the results, we can conclude that our zoom-in technique, stage two of our generator, is efficient and consistent.

5.4 Summary

In summary in this chapter, we have described the experiments, the environment, and the results of our proposed method. We have explained the metrics used to evaluate the performance of our network and strategies to train the network. Based on extensive experiments, we have shown that the proposed approach outperforms state-of-art methods in the task at hand. We have further demonstrated that when using our semi-supervised training, only a few labeled images are needed. Especially, we have shown promising results that a semi-supervised learning technique is able to solve a regression task in an industrial machine vision applications.

Chapter 6

Conclusion and future work

6.1 Conclusion

Convolutional neural network architectures have made substantial advances in various computer vision applications. Finding enough and good quality labeled data is essential for supervised learning with these CNN models. Though building large labeled datasets for all scenarios is not practically feasible, we can leverage semi-supervised techniques to train models using only small amounts of labeled data and large amounts of unlabeled data. We showed that a semi-supervised learning technique can be applied in solving machine vision regression tasks in industrial applications such as our welding seam localization task.

Our vision-based measurement method is able to detect keypoints to guide a robot to properly reach a welding seam. Our heatmap generator uses a two-stage network which is able to zoom-in for higher accuracy compared to standard keypoint detection networks. However, the major novelty of our approach is a semi-supervised training strategy using only a few images to obtain reliable detection results. Our semi-supervised strategy trains the heatmap generator iteratively mixing few annotated samples with many training images without ground truth annotation. The generator produces pseudo labels for training images

without annotation in each step of the semi-supervised training. These pseudo labels are assessed by two discriminators in our novel design. We use two discriminators to solve two independent but complementary classification tasks in order to qualify pseudo labels. Based on this structure, we proposed our semi-supervised learning network. Our experiments demonstrate the success of our method on a real-world vision-based measurement problem.

A limitation of our approach is our proposed method can only detect a single point from an image. The cutout and crop operations can be only applied by a single location.

6.2 Limitations and Future work

There are some limitations of our thesis project. For example, the type and the orientation for the joints are very limited. There are only a few different types of valid joints required to be recognized in the project. The orientation of the laser projection is always from one direction since the camera is in an almost fixed position on the welding machine. Also, each image only contains a single keypoint to be predicted. As a result, these factors limit the variation of the task.

In addition, during the semi-supervised process, we have to generate predictions for all unlabeled images. If the amount of unlabeled images is too large, this step is extremely time-consuming. To overcome this issue, including a well-trained classifier may simplify this step by picking numbers of samples from each category rather than scanning the entire unlabeled dataset.

Many different adaptations, tests, and experiments have been left for future work. The following ideas could be developed and may improve our method:

1. More complex tasks: It could be interesting to extend our approach to more sophisticated regression problems in vision-based measurements such as human pose

estimation, human face estimation, etc. These problems are much more complex than the welding seam problem due to multiple keypoints, occlusions because of articulated limbs, and illumination changes.

2. Less constraint settings: The performance of our method with images taken from a variation of viewpoints or with changes in laser projection may be investigated. Unlike the current task, in the future, the shape of the joint may have various orientations. Also, involving more shapes of the laser projection introduces additional complexity for the problem.

3. Fewer labeled samples: Solving the task with only a single labeled sample would be a great challenge. One-shot learning or unsupervised training may be applied to the problem as well.

In conclusion, this thesis explored the use of the semi-supervised learning applied to keypoint detection. In future studies, we would like to see any extension of our work applied to more machine vision measurement problems in daily lives or in industrial manufacturing.

References

[1] A. Rout, B. Deepak, and B. Biswal, "Advances in weld seam tracking techniques for robotic welding: A review," *Robotics and computer-integrated manufacturing*, vol. 56, pp. 12–37, 2019.

[2] G. Kostopoulos, S. Karlos, S. Kotsiantis, and O. Ragos, "Semi-supervised regression: A recent review," *Journal of Intelligent & Fuzzy Systems*, vol. 35, no. 2, pp. 1483–1500, 2018.

[3] J. Fan, S. Deng, F. Jing, C. Zhou, L. Yang, T. Long, and M. Tan, "An initial point alignment and seam-tracking system for narrow weld," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 877–886, 2019.

[4] S. Shirmohammadi and A. Ferrero, "Camera as the instrument: the rising trend of vision based measurement," *IEEE Instrumentation & Measurement Magazine*, vol. 17, no. 3, pp. 41–47, 2014.

[5] D. Xu, Z. Fang, H. Chen, Z. Yan, and M. Tan, "Compact visual control system for aligning and tracking narrow butt seams with co 2 gas-shielded arc welding," *The International Journal of Advanced Manufacturing Technology*, vol. 62, no. 9-12, pp. 1157–1167, 2012.

[6] Y. Li, Y. F. Li, Q. L. Wang, D. Xu, and M. Tan, "Measurement and defect detection of the weld bead based on online vision inspection," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 7, pp. 1841–1849, 2009.

[7] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised learning*. MIT Press, 2006.

[8] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *33rd annual meeting of the association for computational linguistics*, pp. 189–196, 1995.

[9] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100, 1998.

[10] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 11, pp. 1529–1541, 2005.

[11] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation for consistency training," *arXiv preprint arXiv:1904.12848*, 2019.

[12] Q. Liu, L. Yu, L. Luo, Q. Dou, and P. A. Heng, "Semi-supervised medical image classification with relation-driven self-ensembling model," *IEEE Transactions on Medical Imaging*, vol. 39, no. 11, pp. 3429–3440, 2020.

[13] B. B. Gatto, L. S. Souza, E. M. dos Santos, K. Fukui, W. S. Júnior, and K. V. dos Santos, "A semi-supervised convolutional neural network based on subspace representation for image classification," *EURASIP Journal on Image and Video Processing*, vol. 2020, no. 1, pp. 1–21, 2020.

[14] H. Shi and Z. Wang, "Improved stacked hourglass network with offset learning for robust facial landmark detection," in *2019 9th International Conference on Information Science and Technology (ICIST)*, pp. 58–64, IEEE, 2019.

[15] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*, pp. 483–499, Springer, 2016.

[16] J. Yang, Q. Liu, and K. Zhang, "Stacked hourglass network for robust facial landmark localisation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 79–87, 2017.

[17] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[18] Y. LeCun and Y. Bengio, "Convolutional Networks for Images, Speech, and Time Series," *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10, p. 1995, 1995.

[19] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[21] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *The International Conference on Learning Representations (ICLR)*, 2015.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.

[24] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks*, vol. 5, pp. 157–166, Mar 1994.

[25] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.

[26] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[28] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[29] M. Benzi, G. H. Golub, J. Liesen, *et al.*, "Numerical solution of saddle point problems," *Acta numerica*, vol. 14, no. 1, pp. 1–137, 2005.

[30] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *Proceedings in Computational Statistics*, pp. 177–186, Springer, 2010.

[31] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2)," in *Doklady an ussr*, vol. 269, pp. 543–547, 1983.

[32] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8624–8628, IEEE, 2013.

[33] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *The International Conference on Learning Representations (ICLR)*, 2015.

[34] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *arXiv preprint arXiv:1706.08500*, 2017.

[35] Y. He, K. Song, Q. Meng, and Y. Yan, "An end-to-end steel surface defect detection approach via fusing multiple hierarchical features," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 4, pp. 1493–1504, 2019.

[36] Z. Liu, S. Wu, Q. Wu, C. Quan, and Y. Ren, "A novel stereo vision measurement system using both line scan camera and frame camera," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 10, pp. 3563–3575, 2018.

[37] L. Zhang, Q. Ye, W. Yang, and J. Jiao, "Weld line detection and tracking via spatial-temporal cascaded hidden markov models and cross structured light," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 4, pp. 742–753, 2013.

[38] L. Yang, J. Fan, Y. Liu, E. Li, J. Peng, and Z. Liang, "Automatic detection and location of weld beads with deep convolutional neural networks," *IEEE Transactions on Instrumentation and Measurement*, 2020.

[39] A. Farhadi and J. Redmon, "Yolov3: An incremental improvement," *Computer Vision and Pattern Recognition, cite as*, 2018.

[40] H. Liu, Y. Yan, K. Song, H. Chen, and H. Yu, "Efficient optical measurement of welding studs with normal maps and convolutional neural network," *IEEE Transactions on Instrumentation and Measurement*, 2020.

[41] L. Wang and Q. Shen, "Visual inspection of welding zone by boundary-aware semantic segmentation algorithm," *IEEE Transactions on Instrumentation and Measurement*, 2020.

[42] Y. Zou and R. Lan, "An end-to-end calibration method for welding robot laser vision systems with deep reinforcement learning," *IEEE Transactions on Instrumentation and Measurement*, 2019.

[43] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[45] R. J. Hocken and P. H. Pereira, *Coordinate measuring machines and systems*. CRC press, 2016.

[46] P.-C. Tung, M.-C. Wu, and Y.-R. Hwang, "An image-guided mobile robotic welding system for smaw repair processes," *International Journal of Machine Tools and Manufacture*, vol. 44, no. 11, pp. 1223–1233, 2004.

[47] W. J. Shao, Y. Huang, and Y. Zhang, "A novel weld seam detection method for space weld seam of narrow butt joint in laser welding," *Optics & Laser Technology*, vol. 99, pp. 39–51, 2018.

[48] Y.-K. Liu and Y.-M. Zhang, "Supervised learning of human welder behaviors for intelligent robotic welding," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 3, pp. 1532–1541, 2015.

[49] Y. Cheng, Q. Wang, W. Jiao, R. Yu, S. Chen, Y. Zhang, and J. Xiao, "Detecting dynamic development of weld pool using machine learning from innovative composite

images for adaptive welding," *Journal of Manufacturing Processes*, vol. 56, pp. 908–915, 2020.

[50] Z. Yan, D. Xu, Y. Li, and M. Tan, "A vision-based seam tracking system for submerged arc welding," in *Robotic Welding, Intelligence and Automation*, pp. 349–357, Springer, 2007.

[51] J. Qin, G. Ma, and P. Liu, "Image processing algorithm of weld seam based on crawling robot by binocular vision," in *2011 Second International Conference on Mechanic Automation and Control Engineering*, pp. 337–340, IEEE, 2011.

[52] Y. Chen, Y. Tian, and M. He, "Monocular human pose estimation: A survey of deep learning-based methods," *Computer Vision and Image Understanding*, vol. 192, p. 102897, 2020.

[53] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," *arXiv preprint arXiv:1406.2984*, 2014.

[54] A. Jain, J. Tompson, Y. LeCun, and C. Bregler, "Modeep: A deep learning framework using motion features for human pose estimation," in *Asian conference on computer vision*, pp. 302–315, Springer, 2014.

[55] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 4724–4732, 2016.

[56] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang, "Multi-context attention for human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1831–1840, 2017.

[57] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, "Learning feature pyramids for human pose estimation," in *proceedings of the IEEE international conference on computer vision*, pp. 1281–1290, 2017.

[58] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5693–5703, 2019.

[59] Y. Chen, C. Shen, X.-S. Wei, L. Liu, and J. Yang, "Adversarial posenet: A structure-aware convolutional network for human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1212–1221, 2017.

[60] X. Peng, Z. Tang, F. Yang, R. S. Feris, and D. Metaxas, "Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2226–2234, 2018.

[61] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 734–750, 2018.

[62] T. L. Munea, Y. Z. Jembre, H. T. Weldegebriel, L. Chen, C. Huang, and C. Yang, "The progress of human pose estimation: a survey and taxonomy of models applied in 2d human pose estimation," *IEEE Access*, vol. 8, pp. 133330–133348, 2020.

[63] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017.

[64] D. A. Reynolds, "Gaussian mixture models.," *Encyclopedia of biometrics*, vol. 741, pp. 659–663, 2009.

[65] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine learning*, vol. 39, no. 2, pp. 103–134, 2000.

[66] C. Olivier, S. Bernhard, and Z. Alexander, "Semi-supervised learning," in *IEEE Transactions on Neural Networks*, vol. 20, pp. 542–542, 2006.

[67] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[68] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *ICML Workshop on Challenges in Representation Learning*, 2013.

[69] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, "Semi-supervised learning with deep generative models," *Advances in neural information processing systems*, vol. 27, pp. 3581–3589, 2014.

[70] C. Zhang, Q. Zhang, and J. H. Hansen, "Semi-supervised learning with generative adversarial networks for arabic dialect identification," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5986–5990, IEEE, 2019.

[71] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," *Advances in neural information processing systems*, vol. 29, pp. 2234–2242, 2016.

[72] X. Dong and Y. Yang, "Teacher supervises students how to learn from partially labeled images for facial landmark detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 783–792, 2019.

[73] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Advances in Neural Information Processing Systems*, pp. 5049–5059, 2019.

[74] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Advances in neural information processing systems*, pp. 1195–1204, 2017.

[75] V. Verma, A. Lamb, J. Kannala, Y. Bengio, and D. Lopez-Paz, "Interpolation consistency training for semi-supervised learning," pp. 3635–3641, 2019.

[76] S. M. Laine and T. O. Aila, "Temporal ensembling for semi-supervised learning," Apr. 12 2018. US Patent App. 15/721,433.

[77] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[78] S. Park, J. Park, S.-J. Shin, and I.-C. Moon, "Adversarial dropout for supervised and semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[79] Y. Luo, J. Zhu, M. Li, Y. Ren, and B. Zhang, "Smooth neighbors on teacher graphs for semi-supervised learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8896–8905, 2018.

[80] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research),"

[81] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[82] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE international conference on computer vision*, pp. 1476–1485, 2019.

[83] X. Dong, S.-I. Yu, X. Weng, S.-E. Wei, Y. Yang, and Y. Sheikh, "Supervision-by-registration: An unsupervised approach to improve the precision of facial landmark detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 360–368, 2018.

[84] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[85] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[86] W. Huang, C. Yang, and T. Hou, "Spine landmark localization with combining of heatmap regression and direct coordinate regression," *arXiv preprint arXiv:2007.05355*, 2020.

[87] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 466–481, 2018.

[88] X.-J. Mao, C. Shen, and Y.-B. Yang, "Image restoration using convolutional auto-encoders with symmetric skip connections," 06 2016.

[89] A. Ramírez-Soriano, S. E. Ramos-Onsins, J. Rozas, F. Calafell, and A. Navarro, "Statistical power analysis of neutrality tests under demographic expansions, contractions and bottlenecks with recombination," *Genetics*, vol. 179, no. 1, pp. 555–567, 2008.

[90] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *CoRR*, vol. abs/1708.04552, 2017.

[91] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch:

An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alch-Buc, E. Fox, and R. Garnett, eds.), Curran Associates, Inc., 2019.

[92] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.