Weakly supervised learning with stochastic supervision and knowledge transfer

Xiaoou Lu

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Statistical Science

University College London

September 13, 2020

Abstract

In recent years, machine learning methods especially supervised learning methods have achieved great progress in both methodologies and applications. However, in supervised learning, each training sample requires a label to indicate its ground-truth. In many machine learning tasks, it is hard to get sufficient accurately labelled training samples. Weakly supervised learning is an extended setting of supervised learning to more general tasks. In this thesis, we focus on proposing novel methods for inaccurate supervision and incomplete supervision under the setting of weakly supervised learning.

In inaccurate supervision, problems with nondeterministic labels, such as stochastic supervision problems, are rarely discussed. In stochastic supervision, the supervision is a probabilistic assessment rather than a deterministic label. In Chapter 2, we provide four generalisations of stochastic supervision models, extending them to asymmetric assessments, multiple classes, feature-dependent assessments, and multi-modal classes, respectively. Corresponding to these generalisations, four new EM algorithms are derived. We show the effectiveness of our generalisations through illustrative examples of simulated datasets, as well as real-world examples of two famous datasets, the MNIST dataset, and the CIFAR-10 dataset.

For incomplete supervision problems, we focus on improving the semi-supervised learning in one domain/task by transferring knowledge from another domain/task or from many domains/tasks. In Chapter 3, a novel domain-adaptation-based method is proposed to improve a typical application of semi-supervised learning: the pose estimation, in which the implicit density estimation problem in the domain adaptation is solved by using a neural network to approximate it. The proposed method transfers the knowledge from the training samples in the synthetic data domain to improve the learner in the real data domain, and achieves state-of-the-art performance. In Chapter 4, we focus on transferring knowledge from many tasks to improve the semi-supervised few-shot learning. We use meta-learning to transfer knowledge from many meta-train tasks. A tailor-made ensemble method for few-shot learning is proposed to relieve the pseudo-label noise problem in the semi-supervised few-shot learning. The proposed method also achieves state-of-the-art performances in two widely used benchmark datasets (miniImageNet and tieredImageNet) in few-shot learning.

Acknowledgements

I would like to express my sincere thanks to my supervisor Dr. Jing-Hao Xue. During my Ph.D. study, he has been providing guidance and training to let us become a qualified lecturer and researcher. He also encourages and helps me apply for the Teaching Assistant Scholarship in the department. These experiences not only improve my teaching skill but also reduce my financial pressure during my Ph.D. study. He also provides opportunities and helps for me to build up international collaboration with other researchers. I would like to thank Dr. Xue for his patience and help during my Ph.D. study.

I would like to thank my second supervisor Prof. Christian Hennig especially for his comments and suggestions for academic writing and presentation.

I would like to thank Dr. Yang Wu for his discussion and suggestions on research and his warm help when I was visiting NAIST and Kyoto University.

I also want to express my gratitude to my collaborators, Dr. Xiaoxu Li, Mr. Ziqiang Zheng, and Dr. Fei Zhou. Especially Dr. Xiaoxu Li for the GPU device provided by her. Without these GPU devices, many experiments of my research even cannot finish.

I would like to thank all the staff of the Department of Statistical Science. Thanks for all your help in recent years.

I would like to thank Mr. Cinxin Liu and Mr. Arthur Clark. Your Sci-fi novels enrich the spare time of my Ph.D. study.

Lastly, I want to express my sincere gratitude to my family. Thanks to my parents who have been supporting me all the time. Thanks for my grandfather who taught me fearless and encouraged me to keep faithful to my belief even though he already passed away many years.

Impact

This thesis potentially can contribute to both methodology and application aspects of machine learning. The impact of the work presented in the thesis can benefit both inside and outside of academia.

For the impact on the inside of academia, there are several potential contributions. Firstly, the proposed four generalisations of the stochastic supervision model in Chapter 2 can be applied to ensemble learning and self-training to produce some novel methods and work for the machine learning community. Secondly, the proposed semi-supervised few-shot learning method in Chapter 4 can be extended to other types of data, such as sequential data and hyperspectral data. Thirdly, the proposed semi-supervised few-shot learning method improves the performance of few-shot image classification tasks, so it potentially can be applied to many machine learning applications, such as handwriting classification, Person ReID, and object detection.

For the impact on the outside of academia, the proposed method in Chapter 3 can be applied to many applications of hand pose estimation, such as computer games to improve the human-computer interaction. The proposed method in Chapter 4 can improve the performances of few-shot learning algorithms and extend the application scenarios of these algorithms, since these algorithms do not need many labelled samples. Our method also can reduce the workload of labelling data, since the pseudo label provided by our model can be seen as the proposal for the true label. The wide use of these machine learning algorithms potentially can create many new apps on computers and smartphones to provide fast and convenient service for our life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Weakly supervised learning

Supervised learning is a typical machine learning method, which learns a predictive model from many labelled training data to predict the class (classification task) or value of response (regression task) of unlabelled data. In recent years, supervised learning have achieved great success for many machine learning applications. In supervised learning, the training dataset contains training samples; each training sample contains a pair of a feature vector and a label of the feature vector; the feature vector includes the covariates that we used to predict; and the label is the variable that indicates the class (classification) or real value of response (regression). However, supervised learning often requires many labelled data to train a model with good quality of prediction, but, in many machine learning tasks, acquiring accurately labelled training samples is expensive or even impossible, hence the difficulty in obtaining sufficient high-quality training samples limits the usage of supervised learning. Thus, it is helpful to develop machine-learning models to be able to work with weak supervision[2], therefore, extending supervised learning algorithms to the weakly supervised settings is an important research problem in machine learning.

1.1.1 Taxonomy of weakly supervised learning

The weakly supervised learning tasks involve various settings of problems. According to a widely recognised taxonomy [2], weakly supervised learning tasks can be divided into the following three different types: 1) inaccurate supervision; 2) incomplete supervision, and 3) inexact supervision.

Inaccurate supervision takes the uncertainty of labels into account. In the inaccurate supervision, the supervision information is not necessarily the real label. The supervision information can be a label with noise, a score or a softmax vector. For example, in the classification task, a training example may be similar to several classes and difficult to categorise, so the annotator gives the sample a softmax vector to indicate the probability of class belonging. The annotator can be either a human or a pre-trained model. The supervision information provided by the annotator has the uncertainty of class belonging or target value, we use the terminology introduced by [3] and [4] to call the supervision "stochastic supervision".

In the setting of incomplete supervision, only a small subset of the training set is labelled. Incomplete supervision is a common scenario in many machine learning applications. For example, in hand pose estimation, we can easily get a large number of hand images, however, annotating the joint locations of a hand in an image is relatively expensive. The incomplete supervision tasks are often roughly categorised into two sub-types: semi-supervised learning and active learning. Active learning still needs human annotation; in this thesis, we will not involve active learning. In semi-supervised learning, a specially difficult case is semi-supervised few-shot learning. In the semi-supervised few-shot learning [5], each class only has few shots of examples that are labelled, which is insufficient to train a good learner. In order to solve this problem, it is natural to transfer knowledge from another domain (via domain adaptation) that is similar to the current domain and has sufficient labelled samples, or to transfer shared public knowledge from many similar learning tasks by meta-learning. The knowledge can be the training instances, the class centroids [6], the distance metric in the feature space [7] and the parameters of the neural network [8].

The last type of weakly supervised learning is inexact supervision. In the inexact supervision, the training samples only have coarse labels. For example, in a cancer diagnostic application, we only know which image has a tumor but we do not know the location of the tumor. The settings of inexact supervision methods are very different in different applications.

1.2 Inaccurate supervision

1.2.1 Summary of inaccurate supervision

Inaccurate supervision has received much research attention in recent years, however, most of the current studies are based on the label noise setting [9]. In the label noisy setting, the training set $(\{x_n\}_{n=1}^{N}, \{y_n\}_{n=1}^{N})$ contains paired training sample feature vector x_n and the corresponding label y_n, just like that in supervised learning. The main difference from supervised learning is the label y_n may contain label noise (i.e., mislabelling). The key part of the problem is how to find mislabelled data. Most of the existing methods try to first identify/detect and then rectify mislabelled data. After that, they either train the model on the cleaned data or relabel the mislabelled data [10]. In [11] and [12], graph-based methods are applied to remove suspicious mislabelled training samples. Since the label noise is related to the label, taking into consideration the relationship between the true label and the noisy label can help us identify the mislabelled data. Many recent studies try to model the joint probability of true label and noisy label. In [13], the true label is viewed as a latent variable, a transition matrix between the true label and the noise label is learned by an EM algorithm. Northcutt.et.al [14] proposed a process that contains three steps, pruning, counting, and ranking, to improve the efficiency of calculating the joint probabilities of the true label and the noise label.

1.2.2 Contribution to stochastic supervision

The aforementioned methods for inaccurate supervision only focus on the label noise setting that only allows for deterministic labels. However, in many real applications, labelled data often require expensive tests and measurements to provide an accurate label. However, we can easily receive weak supervision such as an inaccurate label provided by either a human or a model. The inaccurate labelled data are often labelled by certain experts/supervisors with subjective labelling to some extent. In many situations, an expert (or a model) even cannot provide deterministic labels. For example, in medical diagnostic, accurate diagnostic of whether a patient has a disease may require many very expensive tests and wait for a long time to do the test, however, we can obtain subjective diagnostic by a general practitioner or a prediction model (such as an app in a smartphone). A prediction model or an expert may not be perfectly sure whether a patient has a certain disease, and they can only provide a subjective assessment. The assessment is often expressed in a probabilistic manner. For example, in a classification task, the stochastic supervision can be a softmax vector or a score vector. An illustrative example is provided in the Figure 1.1.

Compared with a given deterministic label (or a one-hot vector), the stochastic supervision not only provides the annotator's assessment of class belonging but also the relation of different classes. However, there have been little research on stochastic supervision models. Aitchison and Begg [15] and Krishnan and Nandy [3] first introduced the stochastic supervision models for discriminant analysis. Titterington [4] introduced a new stochastic supervisor model that can accept assessment (a random variable) in the range of $(-\infty, \infty)$, and thus the assessment can be modeled by Gaussian distributions. However, Titterington [4]'s model has many strict assumptions. In order to generalise Titterington's model to more settings, we provide four generalisations of Titterington's model in Chapter 2, making it more flexible and generic to deal with more complicated real-world classification tasks.

1.3 Incomplete supervision

1.3.1 Summary of semi-supervised learning

Incomplete supervision also has received much attention in the machine learning community in recent years. Incomplete supervision has many different directions, in this thesis, we only focus on the semi-supervised setting. The illustrative example of semi-supervised learning is shown in the Figure 1.2.

The semi-supervised learning algorithms can be roughly categorised into five types: generative model based methods, discriminative model based methods, graph-based methods, ensemble-based methods and deep learning based methods.

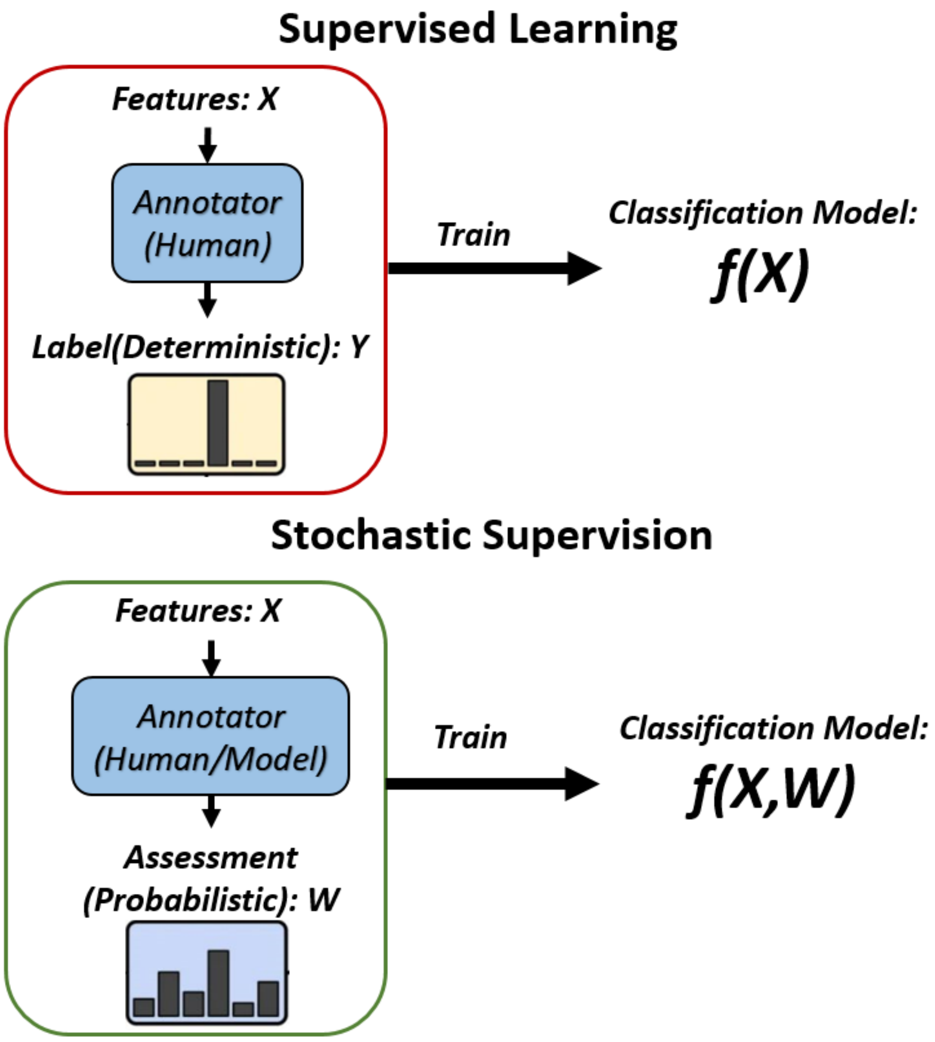


Figure 1.1: Illustration of stochastic supervision, by comparing supervised learning and stochastic supervision. There are mainly three aspects of the difference between supervised learning and stochastic supervision. Firstly, the stochastic supervision model receives inaccurate supervision (probabilistic assessment) rather than deterministic supervision. Secondly, in the stochastic supervision model, the annotator can be either human or model. Thirdly, the stochastic supervision model uses both features and assessments to make a prediction.

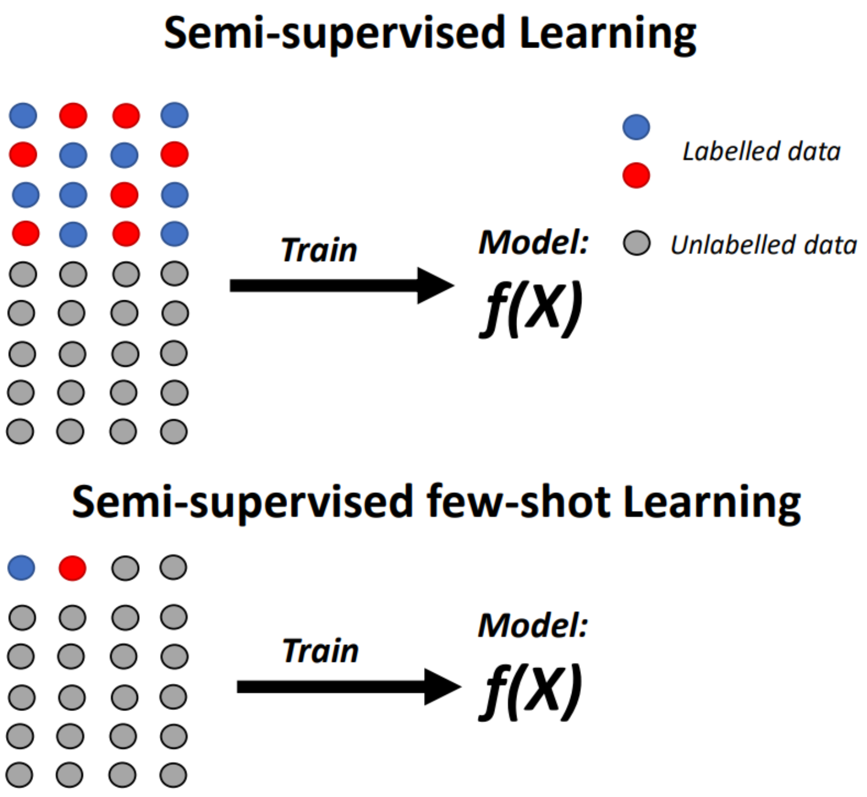


Figure 1.2: Upper panel: illustration of semi-supervised learning. In semi-supervised learning, the training data are partially labelled, and the model learns from both labelled and unlabelled data. Lower panel: illustration of semi-supervised few-shot learning, a special case of semi-supervised learning, where only few data are labelled for each class.

[16] and [17] assume that all labelled and unlabelled data are generated by a generative model and the corresponding label of the unlabelled data is a latent variable. They further proposed an EM algorithm to estimate labels of unlabelled data. [18] and [19] proposed two discriminative model based methods. They extend the support vector machine (SVM) to a semi-supervised setting. The semi-supervised SVMs not only consider the labelled data but also consider the unlabelled data. The hyper-plane is required to cross the low-density region of both labelled and unlabelled data. [20], [21] and [22] proposed several graph-based semi-supervised learning algorithms. In these methods, nodes of the graph represent the training sample and edges correspond to the relation or distance between training samples. However, these methods are transductive since the graph has to be reconstructed

when a new instance is added to the data set. The ensemble-based semi-supervised learning method tries to use several different classifiers to assign pseudo-labels for unlabelled data. [23] and [24] proposed multi-view based ensemble methods. Each base classifier is trained with different features (different views). The pseudo-label is given by the ensemble of different base classifiers. For deep learning based methods, the temporal ensembling[25] proposed a self-ensembling method for semi-supervised learning by taking an exponential moving average of the output of networks with different training epoch, different regularisation, and different augmentations. The mean-teacher [26] is an upgraded temporal ensemble method. The mean-teacher taking an exponential moving average on the model parameter rather than the predicted labels. Comparing to temporal ensembling, the mean-teacher can effectively and fast update ensembles when learning a large dataset [26] since temporal ensembling can only epoch-wise update ensemble while mean-teacher can step-wise update. Recently, there are many holistic methods try to unify state-of-the-art methods in a single framework. The MixMatch [27] uses MixUp to mix labelled and unlabelled data. The entropy of predicted labelled was reduced by the sharping operation. The ReMixMatch[28] improves the MixMatch by the distribution alignment and augmentation anchor methods. The distribution alignment aligns the marginal distribution of real label and the marginal distribution of predicted labels. The augmentation anchor force the prediction of strongly augmented data concise with the prediction of the corresponding (same input) weakly augmented data. The FixMatch [29] further chooses the most confident pseudo labelled data to train the model. These methods achieved state-of-the-art performances in semi-supervised learning, however, these methods require many labelled samples and it is hard to extend to few-shot learning tasks.

1.3.2 Contribution to semi-supervised learning with domain adaptation

However, the above methods only use labelled training samples from a single domain (i.e., the task domain). In semi-supervised learning, only using the labelled data of the specific task is often insufficient to train a high-quality model. In many machine learning applications, there are sufficient labelled training samples from other related domains (tasks). Transferring useful knowledge from a correlated domain can improve the model's performance in the test task. In order to exploit the knowledge from another domain, in Chapter 3, we proposed a domain-adaptation based method to use the training samples from another domain. Since hand pose estimation is a typical machine learning application that suits both semi-supervised learning and domain adaptation, we develop a novel semi-supervised learning algorithm for hand pose estimation.

1.3.3 Contribution to semi-supervised few-shot learning with meta-learning

Furthermore, semi-supervised few-shot learning is a very difficult case for semi-supervised learning, because in the few-shot learning tasks each classification task only contains very few labelled samples such that the labelled training samples are too few to train a good model. It is hence natural to transfer useful knowledge from many similar tasks and share the knowledge to each task (meta-learning) in order to improve the model performance. In Chapter 4, we provide an ensemble-based method for the extremely difficult semi-supervised few-shot learning task.

1.4 Aim of the thesis

As we discussed in section 1.2, since currently the stochastic supervision model still does not receive much research attention while the existing stochastic supervision model has relatively strong assumptions, providing generalisations of the stochastic supervision model is worth of research. Meanwhile, as we discussed in section 1.3, the existing semi-supervised learning algorithms in hand pose estimation only utilise labelled samples in the current domain. It is often insufficient to train a good model so that using labelled samples from another domain potentially can improve the model performance. For ensemble-based semi-supervised learning, existing methods are hard to apply to few-shot learning cases. Designing a tailor-made ensemble strategy for few-shot learning and combining with meta-learning (by transferring knowledge from many similar tasks) potentially can improve the performance of semi-supervised few-shot learning algorithms.

The aim of this thesis is to develop novel methods for weakly supervised learning tasks. More specifically, the aim of the thesis is to develop novel methods for stochastic supervision under inaccurate supervision and for the semi-supervised learning model under incomplete supervision, both in the weakly supervised learning framework.

1.5 Structure of the thesis

In this thesis, we focus on two problems in weakly supervised learning: inaccurate supervision and incomplete supervision. We propose several novel methods to address these two problems.

For inaccurate supervision, in Chapter 2, we propose four generalisations of the stochastic supervision model. The first generalisation extends the stochastic supervision models to asymmetric assessments cases. The second generalisation extends stochastic supervision models from two-class classification to multiple class classification. The third generalisation takes the dependence between feature and assessment into account. The fourth generalisation generalises the stochastic supervisor model to the multi-modal class case, where each class can contain several subclasses.

For incomplete supervision, we focus on improving the semi-supervised learning algorithm by transferring knowledge from another domain or from many tasks. In Chapter 3, we propose a novel method based on domain adaptation and implicit density estimation for a typical application of semi-supervised learning: the pose estimation. We use a generative adversarial net, a type of deep neural network, to solve the implicit density estimation problem in the domain adaptation. We transfer the knowledge of the training samples from the synthetic data domain to the real data domain to improve the learner in the real data domain. The proposed method achieves state-of-the-art performance.

Moreover, in Chapter 4, we focus on transferring knowledge from many tasks to improve a very difficult semi-supervised learning problem: semi-supervised few-shot learning. We utilise meta-learning to transfer knowledge from many meta-training tasks. We also propose a tailor-made ensemble method for few-shot learning to solve the pseudo-label noise problem in semi-supervised learning. Our method also achieves the state-of-the-art performance in two widely used few-shot learning benchmark datasets.

In Chapter 5, we provide a conclusive discussion and propose two future work: a multiple stochastic supervision model, which extends the stochastic supervision model to ensemble learning, and a multiple stochastic supervision ensemble for semi-supervised few-shot classification, which combines both inaccurate supervision with incomplete supervision problems. The EM algorithms of the two future work are also derived to fit the models.

The structure of the thesis is summarised in Figure 1.3.

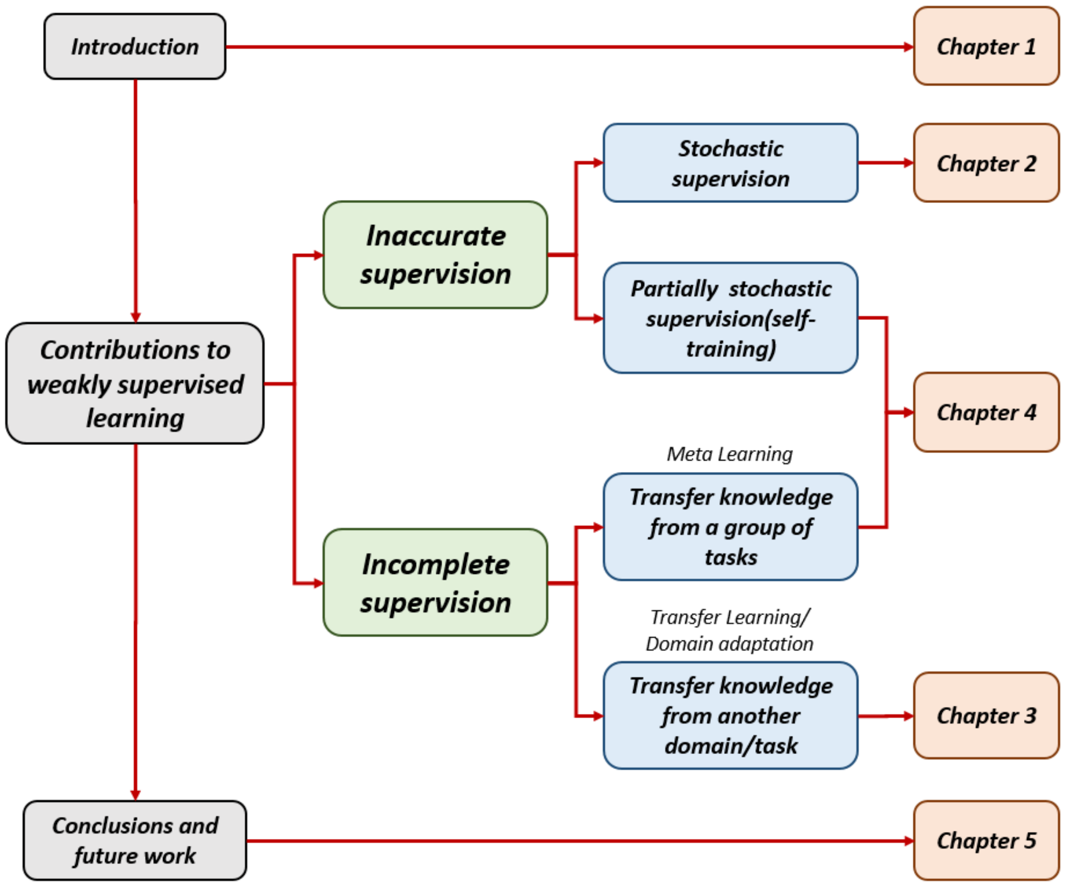

Introduction
Chapter 1
Stochastic supervision
Chapter 2
Inaccurate supervision
Partially stochastic supervision(self-training)
Contributions to weakly supervised learning
Chapter 4
Meta Learning
Transfer knowledge from a group of tasks
Incomplete supervision
Transfer Learning/ Domain adaptation
Transfer knowledge from another domain/task
Chapter 3
Conclusions and future work
Chapter 5

Figure 1.3: The structure of this thesis. Our contributions to weakly supervised learning are in Chapter 2, Chapter 3, and Chapter 4. Chapter 1 is the introduction. The work in Chapter 2 provides four new generalisations of the stochastic supervision model of inaccurate supervision. The work in Chapter 3 provides a novel domain-adaptation based method for semi-supervised learning of incomplete supervision. The work in Chapter 4 provides a novel method for semi-supervised few-shot learning, which is not only based on meta learning but also based on the self-training methods (that use stochastic supervision to provide assessments for unlabelled data). In Chapter 5 we provide conclusions and future work.

Part I

Contribution to Inaccurate Supervision

Chapter 2

Generalisations of stochastic supervision models

The stochastic supervision model is an important research direction to address inaccurate supervision problems in weakly supervised learning. When the labelling information is not deterministic, traditional supervised learning algorithms cannot be directly applied. In this case, stochastic supervision models provide a valuable alternative to classification. However, these models are restricted in several aspects, which critically limits their applicability.

In this chapter, we aim to provide four generalisations of stochastic supervision models, extending them to asymmetric assessments, multiple classes, feature-dependent assessments, and multi-modal classes, respectively. Corresponding to these generalisations, we derive four new EM algorithms. We show the effectiveness of our generalisations through illustrative examples of simulated datasets, as well as real-world examples of three widely used classification datasets, the MNIST dataset, the CIFAR-10 dataset, and the EMNIST dataset.

2.1 Introduction

2.1.1 Inaccurate supervision of weakly-supervised learning

The aim of various statistical learning methods is to infer the class belonging y of an input instance x. Classification and clustering are two extreme ends in the sense of the amount of labelling information provided for the inference of y. In classification, the deterministic labels $\{y_n\}_{n=1}^N$ of N training instances $\{x_n\}_{n=1}^N$, represented by a binary or multilevel categorical random variable y, are usually provided in advance to train a classifier $f(y|x)$ on the information from both the input and output spaces via $(\{x_n\}_{n=1}^N, \{y_n\}_{n=1}^N)$. The trained (supervised) classifier is then used to infer the real label y of a test instance x. In contrast, in clustering, no labelling information is provided at all, hence a clustering method $f(y|x)$ is built on the information from only the input space via $\{x_n\}_{n=1}^N$.

In between classification and clustering, there exists many weakly-supervised classification [30, 31, 32, 33, 34] problems with various types of information provided to help inference. One example is called semi-supervised classification [35, 36], where only part of the deterministic labels $\{y_n\}_{n=1}^N$ are provided for classifier training. Another example is called imperfect supervision [37, 38, 39, 9, 40], where there are some wrong deterministic labels provided in $\{y_n\}_{n=1}^N$. Multiple instance learning [41] also deals with weakly-supervised setting, where deterministic labels are provided for bags of multiple instances rather than for each specific instance. In this paper, we discuss another weakly-supervised classification scheme called stochastic supervision, which, in contrast to all the cases aforementioned, provides no deterministic labels $\{y_n\}_{n=1}^N$ but only probabilistic assessments $\{z_n\}_{n=1}^N$ for inference of y. In other words, only some side information about the output is provided.

2.1.2 Stochastic supervision model

A motivation for stochastic supervision is that, in many applications, labelled data often require expensive tests and measurements to provide an accurate label. However, we can easily receive weak supervision such as an inaccurate label provided by either a human or a model. These inaccurate labelled data are often labelled by certain experts or say supervisors with subjective labelling to some extent, and in many situations, an expert (or model) cannot provide deterministic labels. For example, in medical diagnostic, accurate diagnostic of whether a patient has a disease may require many expensive tests and wait for a very long time to do the test, however, we can obtain subjective diagnostic by an expert (for example a general practitioner) or a prediction model (for example an app in a smartphone). A prediction model or an expert may not be perfectly sure whether a patient has a certain disease, and they can only provide a subjective assessment, which is often expressed in a probabilistic manner. Although the weak supervision (probabilistic supervision) provided by an expert/model is not perfect, however, it is greatly cheaper than an accurate real label and often very fast to see the assessment. These probabilistic assessments can be represented by continuous random variables, from a space different from the discrete space of output label y. On the basis of these assessments (or say probabilistic labels), the statistical classification problem, of fitting a model to the training data and inferring the real labels of the test data, was studied under the nomenclature of stochastic supervision [15, 3, 4, 42, 43, 44]. The stochastic supervision model aims to improve the classification accuracy (of a given model/expert) by inferring the real label from the probabilistic assessments and features.

The research of stochastic supervision models for discriminant analysis was pioneered by Aitchison and Begg [15] and Krishnan and Nandy [3]. As with [3] we assume two classes, namely class 0 and class 1, with proportions π_1 and $\pi_2 = 1 - \pi_1$, respectively. In each class, the data available, including both the d-dimensional feature vector x of an instance and its supervisor's assessment z that the instance belongs to class j, follow a class-dependent distribution $f_j(x,z)$, for $j = 0,1$. The task is to infer the real label y of the instance (x,z).

In [3], the class-dependent joint data-generating distribution $f_j(x,z)$ was further factorised as $f_j(x,z) = f_j(x)q_j(z)$, by assuming that the features x and the assessment z are independent of each other in each class. By supposing the features x are continuous random variables in the range of $(-\infty,\infty)$, it was assumed that $x|y=0 \sim N(\mu_1,\Sigma)$ and $x|y=1 \sim N(\mu_2,\Sigma)$, two class-dependent d-variate Gaussian distributions. We denote the pdfs of $x|y=0$ and $x|y=1$ as $f_1(x)$ and $f_2(x)$, respectively. In the meantime, as the probabilistic assessment z is a continuous random variable in the range of $[0,1]$, it was assumed that $z|y=0 \sim \text{Beta}(a,b)$ and $z|y=1 \sim \text{Beta}(b,a)$, two Beta distributions symmetric between the two classes. We denote the pdfs of $z|y=0$ and $z|y=1$ as $q_1(z)$ and $q_2(z)$, respectively. That is to say, the model in [3] assumes that the data-generating process in class j follows a Gaussian distribution $f_j(x)$ for features x and a Beta distribution $q_j(z)$ for assessment z. Although the assessment z is given for each training instance x, the real label (denoted by y) is unknown, which leads to the likelihood of the training instance, or say the joint distribution of x and z, as $p(x,z) = \pi_1 f_1(x,z) + \pi_2 f_2(x,z)$. Hence this is a latent variable (finite mixture) problem, and the model was fitted by an EM algorithm in [3].

However, there are two problems with Krishnan and Nandy's stochastic supervision model. Firstly, it cannot accept any assessment that $z > 1$ or $z < 0$, while in some real problems the assessment can be a random variable in the range of $(-\infty, \infty)$. Secondly, the EM algorithm for this model is complicated, because there is no exact solution in the M-step for the estimation of certain parameters due to the adoption of the Beta distributions for assessment z.

In order to overcome the two issues above, Titterington [4] introduced a new supervisor's assessment $w = \log \frac{z}{1-z}$ to replace the original z. This transformation is called additive logistic transformation [45], which extends the range of the assessment from $[0,1]$ to the real line and thus the assessment can be modelled by Gaussian distributions. In Titterington's model, supervisor assessments $q_1(w)$ and $q_2(w)$ are assumed to follow two univariate Gaussian distributions $N(-\Delta,\Omega)$ and $N(\Delta,\Omega)$, respectively, where $\Delta > 0$ and $\Omega > 0$. In this model, the constraints of equal variances and symmetry in the assessment distributions between the two classes are preserved. Then Titterington [4] provided an EM algorithm to estimate parameters $\{\pi_1, \mu_1, \mu_2, \Sigma, \Omega, \Delta\}$.

2.1.3 Generalisations of the stochastic supervision model

In this chapter, we aim to generalise Titterington's model in four aspects, to make it more flexible and generic to deal with more complicated real-world classification tasks. We note that the first three aspects have been suggested and discussed by Titterington in section 5.2 of [4], though no detailed derivation was provided as we shall present in this paper. Our four generalisations are briefly described as follows.

1. *Asymmetric assessments.* In both Krishnan and Nandy's and Titterington's models, the two class-dependent distributions of assessments $q_j(z)$ (or $q_j(w)$)

were symmetric and with equal variances. Our first generalisation aims to relax this restriction on the parameter setting of supervisor's assessments.

2. *Multiple classes.* The past models were for two-class discrimination. Our second generalisation is designed for classification of multiple classes.

3. *Feature-dependent assessments.* In Krishhan and Nandy's [3] and Titterington's [4] work, the assessment and the features were modelled independent of each other. Our third generalisation aims to model their dependence.

4. *Multi-modal classes.* In the past research on stochastic supervision, each class was modelled by a Gaussian distribution, implying that there was only a single population for each class, which we call it a uni-modal class. In our fourth generalisation, we model the cases that each class contains multiple subclasses, making the class a multi-modal class.

We shall detail the four generalisations in four subsections of section 2.2 along with four EM algorithms and some numerical illustrations. In section 2.3, we present real-data examples to demonstrate the effectiveness of the generalisations.

2.2 Generalised models and their EM algorithms

2.2.1 Generalisation-1: asymmetric stochastic supervision

The first generalisation aims to relax the assumption of the parameter setting of supervisor's assessments of a stochastic supervision model. In Titterington's model [4], the distributions of assessments in two classes are $w|y=0 \sim N(-\Delta,\Omega)$ and $w|y=1 \sim N(\Delta,\Omega), w \in \mathbb{R}$. They are symmetric in the sense that their variances are the same and their means are the additive inverses of each other. The symmetric assumption makes the model easy to fit, however, this assumption is not realistic in practice. Titterington [4] also discussed this assumption and suggest to generalise this assumption as his future work. In this work, we generalise them to $w|y=0 \sim N(\Delta_1,\Omega_1)$ and $w|y=1 \sim N(\Delta_2,\Omega_2)$. We denote the pdfs of $w|y=0$ and $w|y=1$ as $q_1(w)$ and $q_2(w)$, respectively.

2.2.1.1 Formulation of generalisation-1

Our notation is established as follows. The observable dataset is denoted by $\mathscr{X} = \{X, W\}$, the latent variable set by $\mathscr{Y} = \{Y\}$, and the parameter set by $\theta = \{\pi_1, \pi_2, \mu_1, \mu_2, \Sigma, \Omega_1, \Delta_1, \Omega_2, \Delta_2\}$. $X = \{x_n\}, x_n \in \mathbb{R}^p$, $W = \{w_n\}, w_n \in \mathbb{R}$ and Y_n for $n = 1, \ldots, N$ are instances, assessments and real labels of the instances respectively. For each instance, $y_n = (y_{n1}, y_{n2})$ is a two-dimensional latent variable vector (a one-hot vector representing its real label) such that for class j we have $y_{nj} \in \{0, 1\}$ and for two classes together we have $\sum_{j=1}^{2} y_{nj} = 1$. That is, y_n is a latent indicator vector with only one element being true. x_n is the feature vector of a training sample, and we assume that, for a given class, x_n is Gaussian distributed: $x_n | y_{nj} = 1 \sim N(\mu_j, \Sigma)$. Hence, for complete data $(\mathscr{Y}, \mathscr{X}) = \{(y_n, x_n, w_n), n = 1, \ldots, N\}$, the complete-data likelihood is

$$p(\mathscr{Y}, \mathscr{X}) = \prod_{n=1}^{N} \{y_{n1}[\pi_1 f_1(x_n) q_1(w_n)] + y_{n2}[\pi_2 f_2(x_n) q_2(w_n)]\} \ .$$

Since this model contains latent variables y_n, we can estimate the model parameters by deriving an EM algorithm. In general, an EM algorithm [46] is an iterative algorithm providing a maximum likelihood solution for incomplete data. We can also use the EM algorithm for models with latent variables. In each of its iterations, the EM algorithm has two alternating steps, the expectation (E-)step and the maximisation (M-)step.

In the E-step, we fix current parameters and compute the expectation of the complete-data log-likelihood function with respect to the conditional distributions of latent variables given observed data $\mathscr{X}$: $Q(\theta, \theta^{old}) = \mathbb{E}_{\mathscr{Y}|\mathscr{X}, \theta^{old}}(\log p(\mathscr{Y}, \mathscr{X} | \theta))$.

In the M-step, we find new parameters by maximising the expectation obtained in the E-step: $\theta^{new} = \arg\max_{\theta} Q(\theta, \theta^{old})$.

2.2.1.2 EM algorithm of generalisation-1

E-step For the generalisation-1, in the E-step, we compute the posterior probabilities of latent variables $\gamma(y_{nj}) = p(y_{nj} = 1|\mathscr{X}, \theta)$. By the Bayes rule, we have

$$\gamma(y_{nj}) = \frac{p(x_n, w_n, y_{nj}|\theta)}{p(x_n, w_n|\theta)} = \frac{\pi_j N(x_n|\mu_j, \Sigma_j) N(w_n|\Delta_j, \Omega_j)}{\sum_{j=1}^{2} \pi_j N(x_n|\mu_j, \Sigma_j) N(w_n|\Delta_j, \Omega_j)} ,$$

which are called responsibilities that class j takes for explaining x_n [47].

M-step In the M-step, we take partial differential of $l(\theta) = Q(\theta, \theta^{old})$ with respect to $\theta = \{\pi_1, \pi_2, \mu_1, \mu_2, \Sigma, \Omega_1, \Delta_1, \Omega_2, \Delta_2\}$ and set it equal to zero to obtain updated parameters θ^{new}. It follows that

$$\mu_1^{new} = \frac{\sum\limits_{n=1}^{N} \gamma(y_{n1}) x_n}{\sum\limits_{n=1}^{N} \gamma(y_{n1})} , \ \mu_2^{new} = \frac{\sum\limits_{n=1}^{N} \gamma(y_{n2}) x_n}{\sum\limits_{n=1}^{N} \gamma(y_{n2})} ,$$

indicating that the updated mean μ_j^{new} of the features in class j becomes a weighted average of all data points from the two classes, weighted by the responsibilities; and similarly

$$\Delta_1^{new} = \frac{\sum_{n=1}^{N} \gamma(y_{n1}) w_n}{\sum_{n=1}^{N} \gamma(y_{n1})} , \ \Delta_2^{new} = \frac{\sum_{n=1}^{N} \gamma(y_{n2}) w_n}{\sum_{n=1}^{N} \gamma(y_{n2})} ,$$

i.e., the updated mean Δ_j^{new} of assessments in class j becomes a weighted average of all assessments over the two classes.

Also, the updated covariance matrix of the features is

$$\Sigma^{new} = \frac{\sum\limits_{n=1}^{N} \sum\limits_{j=1}^{2} \gamma(y_{nj})(x_n - \mu_j)(x_n - \mu_j)^T}{\sum\limits_{n=1}^{N} \sum\limits_{j=1}^{2} \gamma(y_{nj})} ,$$

a weighted pooled covariance matrix; and similarly the updated variances of class-specific assessments are

$$\Omega_1^{new} = \frac{\sum_{n=1}^{N} \gamma(y_{n1})(w_n - \Delta_1)^2}{\sum_{n=1}^{N} \gamma(y_{n1})} , \ \Omega_2^{new} = \frac{\sum_{n=1}^{N} \gamma(y_{n2})(w_n - \Delta_2)^2}{\sum_{n=1}^{N} \gamma(y_{n2})} .$$

Since the two mixing weights have to satisfy $\pi_0 + \pi_1 = 1$, we can set $\partial l(\theta)/\partial \pi_j + \lambda = 0$, where λ is a Lagrange multiplier. It then follows that $\pi_1^{new} = \frac{1}{N}\sum_{n=1}^{N} \gamma(y_{n1})$, $\pi_2^{new} = 1 - \pi_1^{new}$, indicating that each of the updated mixing weights is an average of the responsibilities.

2.2.1.3 Illustrative examples for generalisation-1

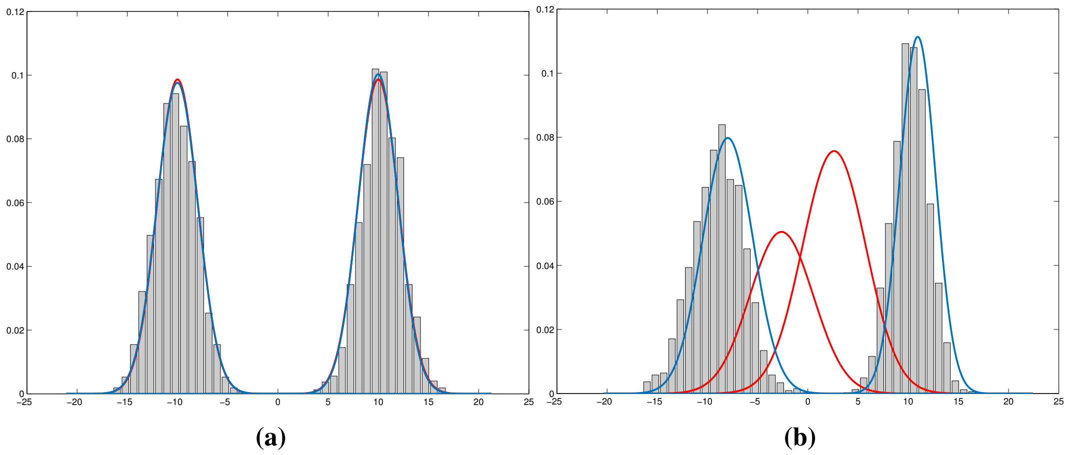
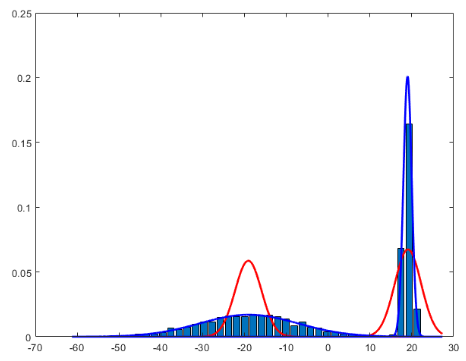


Figure 2.1: (a) Supervisor assessments with *equal* variances and *symmetrical* means between the two classes. Red curve: assessments density estimated by Titterington's model. Blue curve: assessments density estimated by the generalisation-1. (b) Supervisor assessments with *unequal* variances and *asymmetrical* means between the two classes. The rest caption is as for Figure 2.1(a).

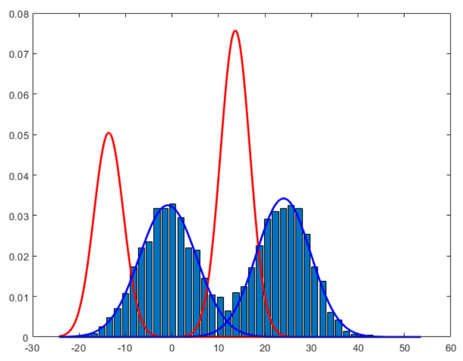
As shown in Figure 2.1(a) and Figure 2.1(b), compared with Titterington's original model, the generalisation-1 is more flexible in accommodating the distributions of supervisor's assessments of various shapes. Let us appreciate it from two aspects.

Firstly, we simulate the supervisor's assessments from two Gaussian distributions with *equal* variances and *symmetrical* means; this setting satisfies the assumption underlying Titterington's model. In this case, as shown in Figure 2.1(a), the generalisation-1 performs similarly to Titterington's model.

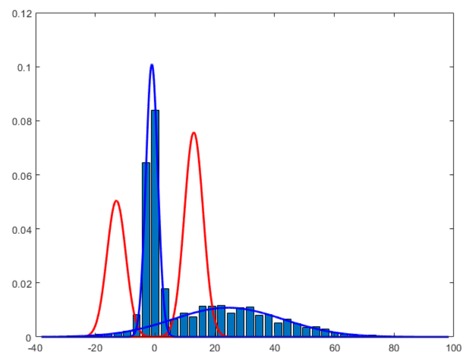
Secondly, we simulate the supervisor's assessments from two Gaussian distributions with *unequal* variances and *asymmetrical* means; this setting does not satisfy the assumption underlying Titterington's model. In this case, as shown in Figure 2.1(b), the generalisation-1 has much better fitting performance than Titterington's model.



(a)



(b)



(c)

Figure 2.2: Three extreme cases of supervisor assessments. (a) Supervisor assessments with large *unequal* variances and *symmetrical* means between the two classes. Red curve: assessments density estimated by Titterington's model. Blue curve: assessments density estimated by the generalisation-1. (b) Supervisor assessments with large *equal* variances and *asymmetrical* means between the two classes. The rest caption is as for Figure 2.2(a). (c) Supervisor assessments with large *unequal* variances and *asymmetrical* means between the two classes. The rest caption is as for Figure 2.2(a).

Besides the moderate unequal variances and asymmetrical case shown in Figure 2.1(b), we also present the superior fitting performances of the generalisation-1 in three extreme cases in Figure 2.2: supervisor's assessments simulated from two Gaussian distributions with large *unequal* variances and *symmetrical* means in Figure 2.2(a), large *equal* variances and *asymmetrical* means in Figure 2.2(b) and large *unequal* variances and *asymmetrical* means in Figure 2.2(c). Obviously, the generalisation-1 can provide better fittings than Titterington's model under these extreme unequal variances and asymmetrical cases.

2.2.2 Generalisation-2: multi-class stochastic supervision

Original stochastic supervision models were only for two-class discrimination. In practice, multi-class classification problems are also prevailing. Hence here we extend Titterington's model to multi-class cases, as suggested by Titterington [4].

2.2.2.1 Formulation of generalisation-2

Suppose there are J classes. As with [4], the supervisor's assessment of an instance x is now a J-variate vector of 'probabilities', $z = (z_1, \dots, z_J)$, and we can define a new assessment vector $w_j = \log \frac{z_j}{z_J}$ for $j = 1, \dots, J-1$, which extends the supervisor's assessments from $(0,1)$ to $(-\infty, \infty)$. Then we can assume that, for each class j, the assessments $w = (w_1, \dots, w_{J-1})$ follow $(J-1)$-variate Gaussian distributions: $q_j(w) = N(\Delta_j, \Omega_j)$, where $q_j(w)$ is the pdf of $w|y = j$.

Then, given the real label $y_n = (y_{n1}, \dots, y_{nJ})$ is unknown, the joint distribution of the observed features x_n and assessment w_n of the nth instance becomes $p(x_n, w_n) = \sum_{j=1}^{J} \pi_j f_j(x_n, w_n)$, where $f_j(x_n, w_n) = f_j(x_n) q_j(w_n)$ and $\pi_j = p(y_{nj} = 1)$ is the mixing weight of class j.

Before going further, we recall some notation to be used for the generalisation-2:

- set of the latent labels $Y = \{y_n\}$, for $n = 1, \dots, N$, where y_n is a J-variate latent vector of real labels, and we have $y_{nj} \in \{0,1\}$ and $\sum_{j=1}^{J} y_{nj} = 1$;
- set of the class mixing weights $\Pi = \{\pi_j\}$, for $j = 1, \dots, J$, where π_j is a scalar;
- set of the class means $U = \{\mu_j\}$, for $j = 1, \dots, J$, where μ_j is a d-variate vector;
- set of the class covariances $\Sigma = \{\Sigma_j\}$, for $j = 1, \dots, J$, where Σ_j is a $d \times d$ matrix;
- set of the assessment means $\Delta = \{\Delta_j\}$, for $j = 1, \dots, J$, where Δ_j is a $(J-1)$-variate vector; and
- set of the assessment covariances $\Omega = \{\Omega_j\}$, for $j = 1, \dots, J$, where Ω_j is a $(J-1) \times (J-1)$ matrix.

In order to make it clear, I will use a 1000 class(J=1000) classification task as an example. In this classification task, $z = (z_1, \ldots, z_{1000})$ is a 1000 dimensional softmax vector. The real label $y_n = (y_{n1}, \ldots, y_{n1000})$ is a 1000 dimensional one-hot vector. In order to derive the EM algorithm in a non-trivial case, we use the logistic transformation $(w_j = \log \frac{z_j}{z_{1000}})$ to transform the z. The transformed assessment $w = (w_1, \ldots, w_{999})$ is a 999 dimensional real value vector. We assume w are mixture of Gaussian distributed with 1000 components (since we have 1000 classes). For each component, $q_j(w) = N(\Delta_j, \Omega_j)$ is a 999 dimensional Gaussian distribution.

In this notation, the parameter set for the generalisation-2 is $\theta = \{\Pi, U, \Sigma, \Delta, \Omega\}$; the complete-data likelihood of observed data $\mathscr{X}$ and latent data $\mathscr{Y}$ is $p(\mathscr{Y}, \mathscr{X}|\theta) = \prod_{n=1}^{N} \sum_{j=1}^{J} y_{nj}[\pi_j N(x_n|\mu_j, \Sigma_j) N(w_n|\Delta_j, \Omega_j)]$, and the marginal likelihood of observed data $\mathscr{X}$ is $p(\mathscr{X}|\theta) = \prod_{n=1}^{N} \sum_{j=1}^{J} \pi_j N(x_n|\mu_j, \Sigma_j) N(w_n|\Delta_j, \Omega_j)$.

2.2.2.2 EM algorithm of generalisation-2

E-step In the E-step we can update posterior distribution of latent variables by setting $q^{new}(\mathscr{Y}) = p(\mathscr{Y}|\mathscr{X}, \theta^{old})$. Since

$$p(\mathscr{Y}|\mathscr{X}, \theta^{old}) = \prod_{n=1}^{N} \frac{\sum_{j=1}^{J} y_{nj}[\pi_j N(x_n|\mu_j, \Sigma_j) N(w_n|\Delta_j, \Omega_j)]}{\sum_{j=1}^{J} \pi_j N(x_n|\mu_j, \Sigma_j) N(w_n|\Delta_j, \Omega_j)},$$

we have the class responsibilities as

$$\gamma(y_{nj}) = \frac{\pi_j N(x_n|\mu_j, \Sigma_j) N(w_n|\Delta_j, \Omega_j)}{\sum_{j=1}^{J} \pi_j N(x_n|\mu_j, \Sigma_j) N(w_n|\Delta_j, \Omega_j)}.$$

M-step In the M-step, we update θ by $\theta^{new} = \arg\max_\theta \sum_{\mathscr{Y}} q^{new}(\mathscr{Y}) \log p(\mathscr{Y}, \mathscr{X}|\theta)$. Since the mixing weights π_j satisfy the sum-to-one constraint, as in section 2.2.1 we introduce a Lagrange multiplier λ and set $\partial l(\theta)/\partial \pi_j + \lambda(\sum_{j=1}^{J} \pi_j - 1) = 0$, which results in the updated mixing weights as $\pi_j^{new} = \frac{1}{N} \sum\limits_{n=1}^{N} \gamma(y_{nj})$, which is again an average of the responsibilities over all the data points. Similarly to the M-step in section 2.2.1, we can obtain the updated means and covariance matrices of features and assessments as

$$\mu_j^{new} = \frac{\sum_{n=1}^{N} \gamma(y_{nj}) x_n}{\sum_{n=1}^{N} \gamma(y_{nj})}, \ \Sigma_j^{new} = \frac{\sum_{n=1}^{N} \gamma(y_{nj})(x_n - \mu_{jk})(x_n - \mu_{jk})^T}{\sum_{n=1}^{N} \gamma(y_{nj})},$$

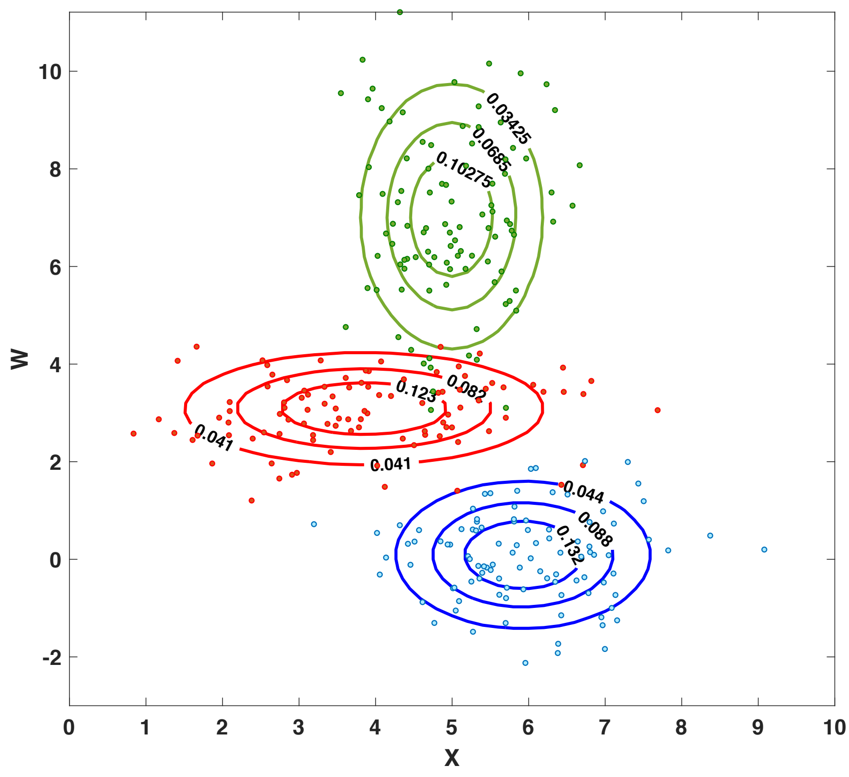
$$\Delta_j^{new} = \frac{\sum_{n=1}^{N} \gamma(y_{nj}) w_n}{\sum_{n=1}^{N} \gamma(y_{nj})}, \ \Omega_j^{new} = \frac{\sum_{n=1}^{N} \gamma(y_{nj})(w_n - \Delta_j)(w_n - \Delta_j)^T}{\sum_{n=1}^{N} \gamma(y_{nj})}.$$

2.2.2.3 An illustrative example for generalisation-2

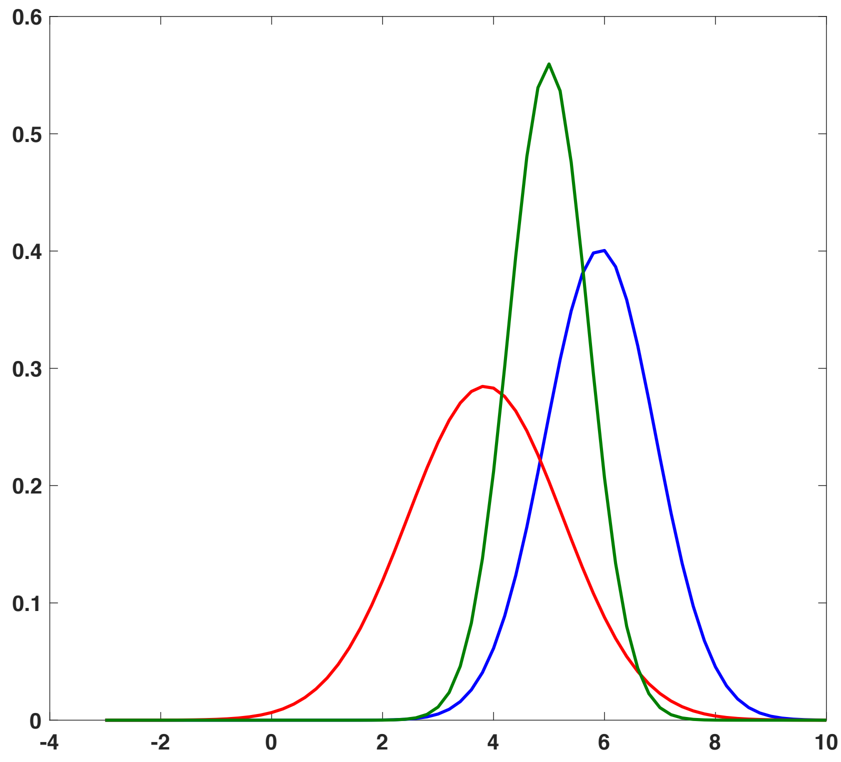
In Figure 2.3(a), we depict a simple example of three classes with a one-dimensional feature x (in the horizontal axis) and one dimension of the assessment w (in the vertical axis). The joint distribution of the feature and the assessment is thus a three-component mixture of Gaussian distributions. Figure 2.3(a) shows that the generalisation-2 works in this case. From Figure 2.3(b), we can observe that the feature's distributions of the three classes seriously overlap. However, with the assessments information added, we can see that the three classes are much more separable, as shown in Figure 2.3(a).

2.2.3 Generalisation-3: feature-dependent stochastic supervision

The original stochastic supervision model assumes features and assessments are independent. This is a strong assumption since in reality features and assessments are dependent (From the table 2.1, we can see in reality feature and assessment are dependent). Titterington [4] was also aware of this problem and he suggested to generalise the stochastic supervision model to the scenarios that the supervisor's assessment w is dependent on the features x. In the generalisation-3, we assume that there is a linear relationship between the assessment and the features. To check the validity of this assumption, we can calculate the Pearson correlation coefficient between x and w or the adjusted R^2 [48] when regressing w against x. In the exper-



(a)



(b)

Figure 2.3: (a) Joint distribution of feature and (one dimension of) assessment for three classes in red, blue, and green, respectively. The contour plots were estimated by the generalisation-2. Each contour is labelled by its corresponding density. (b) Distributions of the feature for three classes in red, blue, and green, respectively.

iment part, we also checked this assumption.

2.2.3.1 Formulation of generalisation-3

The formulation of this generalisation is quite similar to that of the original stochastic supervision model, except that the distribution of assessment is now conditional on the features by replacing $q_j(w)$ with $q_j(w|x)$. This makes the joint distribution of (x_n, w_n) as $p(x_n, w_n) = \sum_{j=1}^{J} \pi_j f_j(x_n) q_j(w_n|x_n)$.

As suggested in [4], a simple way to model $q_j(w_n|x_n)$ is to use the Gaussian distribution $N(\alpha_j + \beta_j^T x_n, \Omega_j)$, and in this case the joint distribution $f_j(x_n, w_n)$ is simply another Gaussian distribution $N(\nu_j, \Psi_j)$, where

$$\nu_j = \begin{pmatrix} \mu_j \\ \alpha_j + \beta_j^T \mu_j \end{pmatrix}, \Psi_j = \begin{pmatrix} \Sigma_j & \Sigma_j \beta_j \\ \beta_j^T \Sigma_j & \Omega_j + \beta_j^T \Sigma_j \beta_j \end{pmatrix},$$

α_j is a $(J-1)$-variate vector, and β_j is a $d \times (J-1)$ matrix.

2.2.3.2 EM algorithm of generalisation-3

E-step In the E-step, we can compute the responsibilities as

$$\gamma(y_{nj}) = \frac{\pi_j f_j(x_n, w_n)}{\sum_{j=1}^{J} \pi_j f_j(x_n, w_n)}.$$

M-step In the M-step, we can update ν_j by setting

$$\nu_j = \frac{\sum_{n=1}^{N} \gamma(y_{nj}) a_n}{\sum_{n=1}^{N} \gamma(y_{nj})},$$

where a_n is a concatenated vector of x_n and w_n. Similarly, the updated covariance matrix is

$$\Psi_j = \frac{\sum_{n=1}^{N} \gamma(y_{nj})(a_n - \nu_j)(a_n - \nu_j)^T}{\sum_{n=1}^{N} \gamma(y_{nj})}.$$

2.2.3.3 An illustrative example for generalisation-3

A simple example of dependent assessment and feature is illustrated in Figure 2.4. The joint distribution of assessment and feature follows a bivariate Gaussian distribution with positive non-diagonal elements in the covariance matrix. The y-axis

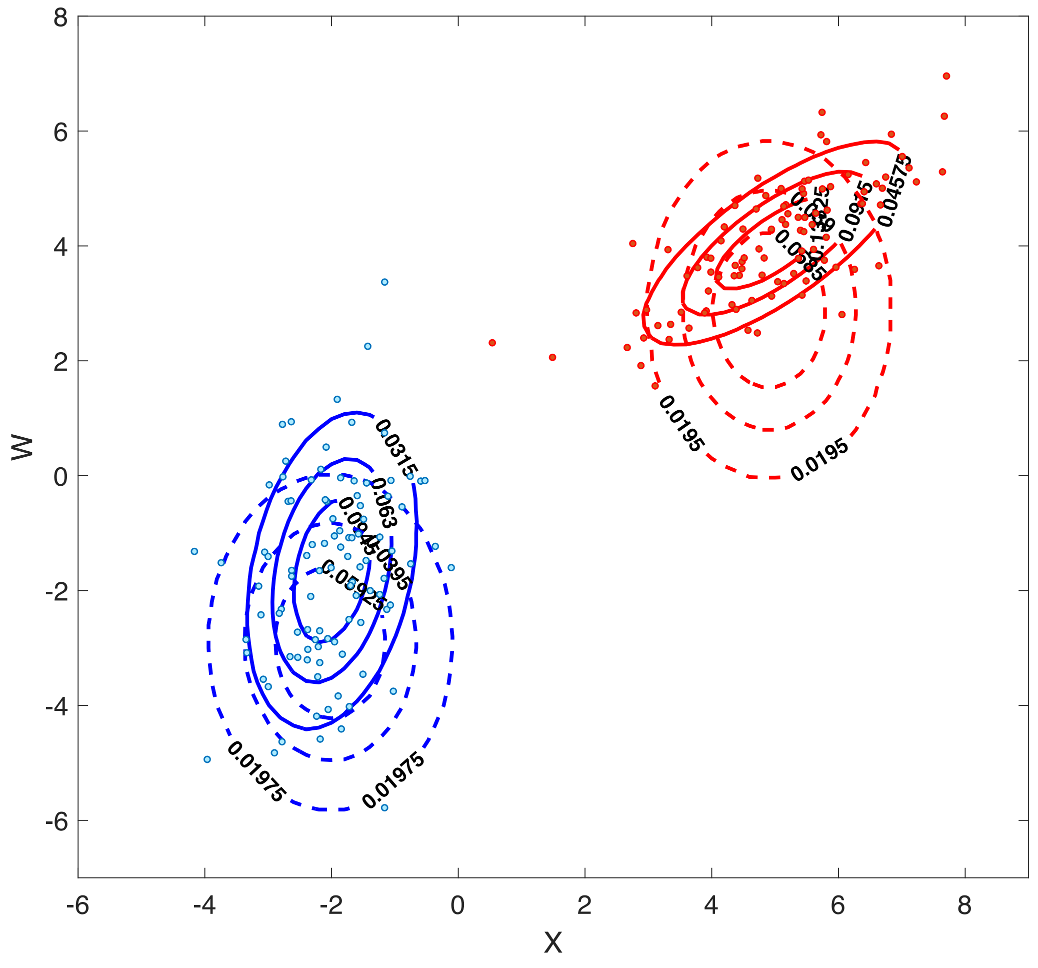


Figure 2.4: Joint distributions of feature and assessment. Dashed contour plots were estimated by Titterington's original stochastic supervision models. Solid contour plots were estimated by the generalisation-3. Each contour is labelled by its corresponding density.

in Figure 2.4 shows the assessment while the x-axis shows the feature. The Pearson correlation coefficient between the feature and assessment of the blue class is 0.8378 while that of the red class is 0.2994. It is clear that, compared with Titterington's original model, which assumes the independence between features and assessments, the generalisation-3 fits the joint distribution of the feature and the assessment much better, when they are indeed dependent.

2.2.4 Generalisation-4: multi-modal classes

In the original work of Krishnan and Nandy's model [3] and Titterington's model [4] and the three generalisations we have presented, each class is modelled by a Gaussian distribution, implying that there was only a single population for each class, which we call a uni-modal class. In practice, however, the distribution of each class can be much complicated, often having multiple modes, which cannot be described by a standard probabilistic distribution. In this context, we propose our generalisation-4 to model the cases that each class contains multiple subclasses, which makes the class a multi-modal class.

In fact, almost all continuous densities can be approximated with arbitrary accuracy by a mixture of Gaussian distributions [47]. For supervised discriminant analysis, the mixture of Gaussians have been studied well in [49, 50, 51, 52]. In the scenario of the stochastic supervision model, which is not deterministically supervised and is itself a mixture of Gaussians, we extend the model to a *mixture of mixtures of Gaussian distributions* [53, 54].

2.2.4.1 Formulation of generalisation-4

Suppose there are J classes and, for each class j, there are K_j subclasses. The total number of subclasses is $K = \sum_{j=1}^{J} K_j$.

We assume for each subclass the features x follow a Gaussian distribution $N(\mu_{jk}, \Sigma_{jk})$, such that each class can be modelled by a mixture of Gaussian distributions $f_j(x)$: $f_j(x_n) = \sum_{k=1}^{K_j} \phi_{jk} N(\mu_{jk}, \Sigma_{jk})$, where $\phi_{jk} = p(t_{njk} = 1 | y_{nj} = 1)$ is the mixing weight of subclass k within class j, and $t_{nj} = (t_{nj1}, \dots, t_{njK_j})$ is a latent vector, such that $t_{njk} \in \{0, 1\}$ indicating the membership of a subclass belonging to a class, and $\sum_{k=1}^{K_j} t_{njk} = 1$.

Given that the real label is also unknown and the instances were generated from J different classes, we have the distribution of features x as a mixture of J different mixtures $f_j(x)$ of Gaussian distributions: $p(x_n) = \sum_{j=1}^{J} \pi_j f_j(x_n)$, where $\pi_j = p(y_{nj} = 1)$ is the mixing weight of class j in the whole dataset, and $y_n = (y_{n1}, \dots, y_{nJ})$ is a latent variable vector of real class label such that $y_{nj} \in \{0, 1\}$ and $\sum_{j=1}^{J} y_{nj} = 1$.

Moreover, as before, for each class j, the supervisor's assessment w follows a univariate Gaussian distribution $N(\Delta_j, \Omega_j)$.

The notation for the generalisation-4 can be summarised as

- set of features $X = \{x_n\}$, for $n = 1, \dots, N$;
- set of the supervisor's assessments $W = \{w_n\}$, for $n = 1, \dots, N$;
- set of the latent class labels $Y = \{y_n\}$, for $n = 1, \dots, N$;
- set of the latent subclass labels $T = \{t_{njk}\}$, for $n = 1, \dots, N$, $j = 1, \dots, J$, $k = 1, \dots, K_j\}$;
- set of the class mixing weights $\Pi = \{\pi_j\}$, for $j = 1, \dots, J$;
- set of the subclass mixing weights $\Phi = \{\phi_{jk}\}$, for $j = 1, \dots, J$, $k = 1, \dots, K_j$;
- set of the subclass means $U = \{\mu_{jk}\}$, for $j = 1, \dots, J$, $k = 1, \dots, K_j$;
- set of the subclass covariances $\Sigma = \{\Sigma_{jk}\}$, for $j = 1, \dots, J$, $k = 1, \dots, K_j$;
- set of the assessment means $\Delta = \{\Delta_j\}$, for $j = 1, \dots, J$; and
- set of the assessment covariances $\Omega = \{\Omega_j\}$, for $j = 1, \dots, J$.

We also define $\mathscr{X} = \{X, W\}$, $\mathscr{T} = \{Y, T\}$, and $\theta = \{\Pi, \Phi, U, \Sigma, \Delta, \Omega\}$. The complete-data likelihood becomes

$$p(\mathscr{X}, \mathscr{T}|\theta) = \prod_{n=1}^{N} \sum_{j=1}^{J} \sum_{k=1}^{K_j} y_{nj} t_{njk} [\pi_j \phi_{jk} N(x_n|\mu_{jk}, \Sigma_{jk}) N(w_n|\Delta_j, \Omega_j)],$$

and the marginal likelihood of the features becomes

$$p(\mathscr{X}) = \prod_{n=1}^{N} \sum_{j=1}^{J} \left\{ \pi_j N(w_n|\Delta_j, \Omega_j) \sum_{k=1}^{K_j} \phi_{jk} N(x_n|\mu_{jk}, \Sigma_{jk}) \right\} .$$

2.2.4.2 EM algorithm of generalisation-4

The EM algorithm to fit the model can be derived as follows.

E-step In the E-step we can update distribution of latent variables by setting $q^{new}(\mathscr{T}) = p(\mathscr{T}|\mathscr{X}, \theta^{old})$. We can update the class responsibilities by setting $\gamma(y_{nj}) = p(y_{nj} = 1|\mathscr{X}, \theta^{old})$, and the subclass responsibilities by setting $r(t_{njk}) = p(t_{njk} = 1|\mathscr{X}, \theta^{old})$, which lead to

$$\gamma(y_{nj}) = \frac{\sum_{k=1}^{K_j} \pi_j \phi_{jk} N(x_n|\mu_{jk}, \Sigma_{jk}) N(w_n|\Delta_j, \Omega_j)}{\sum_{j=1}^{J} \sum_{k=1}^{K_j} \pi_j \phi_{jk} N(x_n|\mu_{jk}, \Sigma_{jk}) N(w_n|\Delta_j, \Omega_j)}$$

and

$$r(t_{njk}) = \frac{\pi_j \phi_{jk} N(x_n|\mu_{jk}, \Sigma_{jk}) N(w_n|\Delta_j, \Omega_j)}{\sum_{j=1}^{J} \sum_{k=1}^{K_j} \pi_j \phi_{jk} N(x_n|\mu_{jk}, \Sigma_{jk}) N(w_n|\Delta_j, \Omega_j)}.$$

M-step In the M-step, we can update θ by $\theta^{new} = \arg\max_\theta \sum_{\mathscr{T}} q^{new}(\mathscr{T}) \log p(\mathscr{T}, \mathscr{X}|\theta)$. It follows that

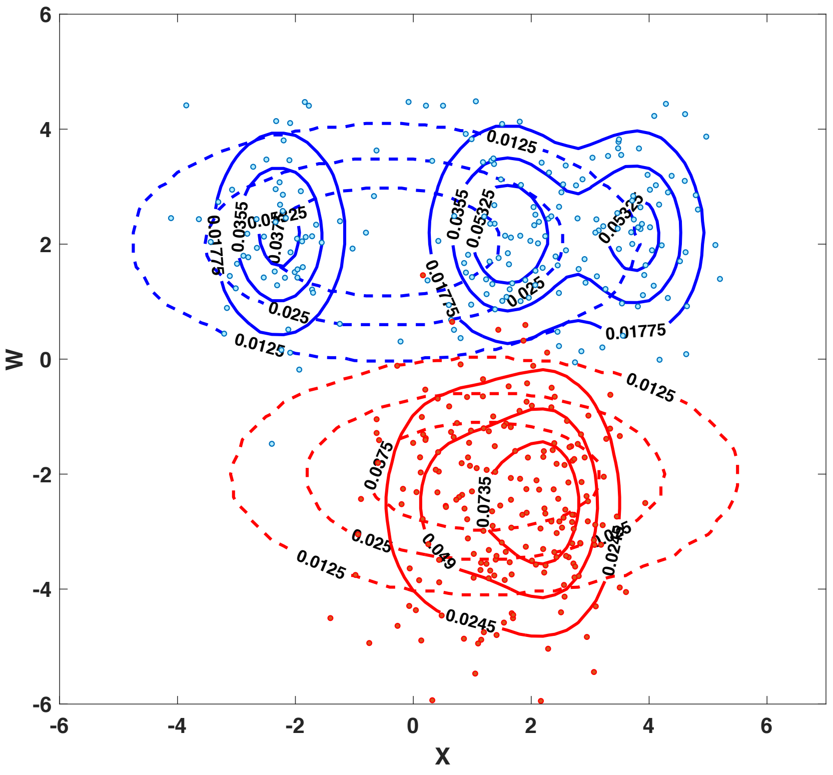
$$\pi_j^{new} = \frac{\sum\limits_{n=1}^{N} \gamma(y_{nj})}{N}, \quad \phi_{jk}^{new} = \frac{\sum\limits_{n=1}^{N} r(t_{njk})}{\sum\limits_{n=1}^{N} \gamma(y_{nj})}, \quad \mu_{jk}^{new} = \frac{\sum\limits_{n=1}^{N} r(t_{njk}) x_n}{\sum\limits_{n=1}^{N} r(t_{njk})},$$

$$\Delta_j^{new} = \frac{\sum\limits_{n=1}^{N} \gamma(y_{nj}) w_n}{\sum\limits_{n=1}^{N} \gamma(y_{nj})}, \quad \Sigma_{jk}^{new} = \frac{\sum\limits_{n=1}^{N} r(t_{njk})(x_n - \mu_{jk})(x_n - \mu_{jk})^T}{\sum\limits_{n=1}^{N} r(t_{njk})},$$

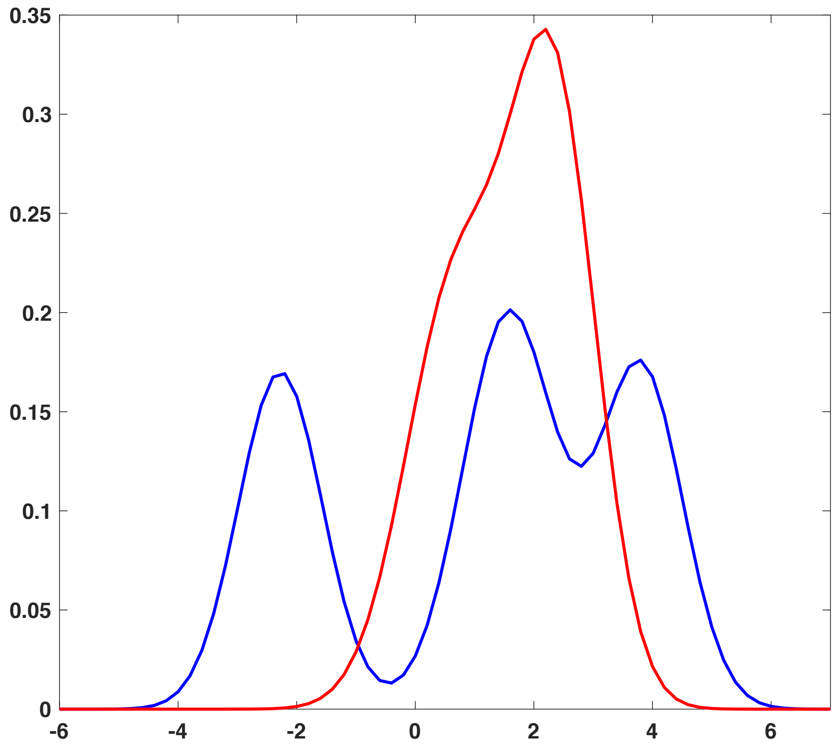
$$\Omega_j^{new} = \frac{\sum\limits_{n=1}^{N} \gamma(y_{nj})(w_n - \Delta_j)(w_n - \Delta_j)^T}{\sum\limits_{n=1}^{N} \gamma(y_{nj})}.$$

2.2.4.3 An illustrative example for generalisation-4

Figure 2.5(a) and Figure 2.5(b) illustrate an example of generalisation-4 for two classes, Class-A with a mixture of two Gaussian subclasses while Class-B with a mixture of three Gaussian subclasses. In this case, Class-A and Class-B are difficult to be modelled well by a single Gaussian distribution, if the original Titterington's model is adopted. Our generalisation-4, however, can handle such a complicated dataset, as shown in Figure 2.5(a). Moreover, comparing Figure 2.5(a) and Fig-



(a)



(b)

Figure 2.5: (a) Joint distributions of feature and assessment for two classes with subclasses: Class-A with two subclasses (red); Class-B with three subclasses (blue). Dashed contour plots were estimated by Titterington's original stochastic supervision models. Solid contour plots were estimated by the generalisation-4. Each contour is labelled by its corresponding density. (b) Distributions of feature for two classes with subclasses: Class-A with two subclasses (red); Class-B with three subclasses (blue).

ure 2.5(b), we can also observe that the data became more separable when the assessment information is added to the model: in Figure 2.5(b) there is a large overlap between the two classes when only the feature is used while in Figure 2.5(a) the two groups of points became separable when the feature and assessment are jointly modelled.

2.3 Real-data experiments

In stochastic supervision, as no deterministic labels were available to training, we cannot compare its classification performance to supervised learning methods such as linear discriminant analysis and support vector machines; on the other hand, it would also be unfairly to favour stochastic supervision if we evaluate it with unsupervised clustering methods such as k-means, given the latter does not even provide any assessment information. Hence we only compare our generalisations with other stochastic supervisors like Titterington's model, the comparison with which has been demonstrated in the previous sections with simulated data, and in the following experiments with real-world data.

In our experiments, the asymmetric and multi-class settings of generalisation-1 and generalisation-2 are also contained by the generalisation-3 and the generalisation-4. Actually, generalisation-3 and generalisation-4 are two different settings of generalisation-1 so that when we compare generalisation-3 and generalisation-4 with the original one, the generalisation-1 already been evaluated. It is same for the generalisation-2. When we compare generalisation-3 and generalisation-4 with the original one, the generalisation-2 already been evaluated.

2.3.1 Real-world datasets

We use three famous real-world datasets in our experiments: the MNIST dataset [55] is used to evaluate the effectiveness of the generalisation-3, the CIFAR-10 dataset [56] is used to evaluate that of the generalisation-4 and the EMNIST dataset [57] is used to evaluate both generalisations.

In MNIST, we aim to classify handwritten digits 3 and 5, which are hard to distinguish. The assessment and features show a strong linear relationship in these two classes, as shown in Table 2.1. In CIFAR-10, we divide the whole dataset into two large classes: the animal class (which includes bird, cat, deer, dog, frog, and horse) and the transportation class (which includes airplane, automobile, ship, and truck). This setting is reasonable for the generalisation-4, because the two large classes contain several subclasses. In EMNIST, we aim to classify three large classes: the digits class, the capital letters class, and the lower cases class. These three classes have 47 subclasses, including 10 digits subclasses, 26 capital letters subclasses, and 11 lowercase subclasses. In order to check the linear relationship between features and assessment, the adjusted-R-square between the assessment and features are shown in Table 2.1. Thus the EMNIST data is a mixture of feature-dependent assessments and multi-modal classes and is suitable to test both generalisations 3 and 4.

Dataset	MNIST		EMNIST		
	Digit 5	Digit 3	Capital Letters	Digits	Lowercases
Adjusted R^2	0.9801	0.9585	0.5585	0.6021	0.6050

Table 2.1: Adjusted R^2 when regressing the assessment against the features for the MNIST and EMNIST datasets.

2.3.2 Experiment settings

2.3.2.1 Assessments generation

Considering that stochastic supervision has assessments only and thus is not a supervised learning model, during the model training we need to ignore the labelling information and before the training we need to 'generate' the supervisor's assessments.

For the MNIST data, since we only implement the two class classification task, to generate such assessments we use logistic regression to generate the probabilities that an instance belongs to two classes as appropriate assessments. Note that the dependency between features and assessments in the generalisation-3 is satisfied when such an approach is adopted to generate assessments, because the posterior probabilities generated are dependent on the features. For the EMNIST data with more than two classes, we use a Naive Bayes classifier to generate the posterior probabilities as assessments.

Based on the assessments only, a simple intuitive approach to inferring y is to directly compare different elements of assessments. For example, for a two-class problem, let $y = 1$ if $w > 0$ and $y = 0$ otherwise; and for a J-class problem, set $y = \arg\max_{j\in\{1,\ldots,J\}} z_j$ (or $y = \arg\max_{j\in\{1,\ldots,J-1\}} w_j$ if at least one $w_j > 0$, and $y = J$ otherwise).

2.3.2.2 Parameters initialisation

Note that in the following initialisation settings, the samples that belong to class j are determined by assessments rather than true labels, because we cannot use true-label information for stochastic supervision methods.

In Titterington's model, the EM algorithm needs initial values of parameters π_j, μ_j, Σ, Δ and Ω. Here we use the sample estimates to initialise these parameters: π_j is the fraction of the estimated number of samples in class j over the total number of samples N, μ_j is the sample mean of the samples, Δ is the sample mean of the assessments of class 1 and $-\Delta$ for class 2, and Σ and Ω are the pooled covariance matrices of the features and the assessments over all J classes, respectively.

In the generalisation-3, α_j and β_j are obtained from the linear regression of the samples in the jth class against their associated w. The EM algorithm of this model needs initial values of π_j, μ_j, Σ_j, and Ω_j. We use the same initialisation settings of π_j and μ_j as those for Titterington's model. Similarly, Σ_j and Ω_j are initialised as the sample covariances of the features and the assessments of class j, respectively.

In the generalisation-4, for CIFAR-10 there are 6 subclasses for animal and 4 for transportation and for EMNIST there are 10 subclasses for digits, 26 for capital letters and 11 for lowercases. The EM algorithm of this model needs initial values of the following parameters: π_j, ϕ_{jk} μ_{jk}, Σ_{jk}, Δ_j and Ω_j. The initialisation of π_j and Ω_j is the same as that for the generalisation-3; Δ_j is initialised as the sample mean of the assessments of samples in class j. To initialise the subclass mean μ_{jk}, covariance matrix Σ_{jk} and mixing weight ϕ_{jk}, we apply k-means to class j: μ_{jk} and Σ_{jk} are set to the subclass means and covariance matrices estimated by k-means on class j, respectively, and ϕ_{jk} is set to the fraction of the number of samples in subclass k of class j over the total number of samples in class j.

2.3.2.3 Validation settings

Since the stochastic supervision models require a supervisor to provide assessments, we use a validation set to train the supervisor to provide assessments for the training set and the test set. For the training set, we discard the real label and use the assessment from the supervisor model and the training data jointly train the model. For the test set, we perform many tests to evaluate the proposed methods. We random sample test data from the whole test set. During each testing, we use both test data and assessment to predict the real label. We repeat the test many times and record classification accuracy to draw boxplots to compare the proposed extension with the original stochastic supervision model.

In the MNIST dataset, we randomly select 2500 samples for each class in the validation set. The training set contains 2500 training samples for each class. The training set and the validation set and the test sets have no overlapping. We perform 20 tests; for each test, 1000 samples are randomly selected to test. We record the classification accuracies on the test sets for all 20 evaluations.

In the CIFAR-10 dataset, we use the training/test split provided by Krizhevsky and Hinton [56], where the training set contains 50000 images with 30000 for the animal class and 20000 for the transportation class. In order to construct the validation set, we further divide the 50000 images in the training set into two datasets: a validation set contains 25000 samples and a training set contains 25000 samples. The test set contains 10000 images with 6000 for the animal class and 4000 for the transportation class. For each experiment, we use all the training samples to train the model and randomly select 1000 images from the test set to evaluation. We repeat the procedure 20 times and record the 20 classification accuracies on the test sets. All images are transformed to greyscale in the experiments.

In the EMNIST dataset, we divide the 3000 images from each subclass to a training set with 1200 images, a validation set with 1200 images, and a test set with 600 images. The assessments of the training images of each subclass are generated by the Naive Bayes model trained from the validation images. For each experiment, we use all 1200×47 training images to train the model and randomly select 1000 images from the whole test set with 600×47 images to test. We repeat the procedure 20 times and record the 20 classification accuracies on the test sets. The pixel values of the margin part of images in EMNIST are zeros, which leads to singular covariance matrices. Thus we add small white noises to these images to make the covariance matrices invertible. Since Titterington's model is used for binary classification and we have three classes here, the one-versus-all strategy [58] is applied here for Titterington's model.

2.3.3 Results

In our experiments, since generalisation-3 and generalisation-4 already contain the assumption of generalisation-1, they are two different settings of generalisation-1. When we compare generalisation-3 and generalisation-4 with the original one, we already evaluate generalisation-1. It is same for generalisation-2 since generalisation-3 and the generalisation-4 already contain the assumption of generalisation-2. In order to illustrate the improvement of generalisation-1 and generalisation-2, I provide the evaluation result in the table 2.2 and table 2.3.

Table 2.2: Comparing the generalisation-1 with the original model on two-class classification tasks.

Setting	Symmetric assessment	Asymmetric assessment	
Model	Original	Generalisation-1	
Model's setting		Generalisation-3	Generalisation-4
Data set	MNIST/CIFAR-10	MNIST	CIFAR-10
Average accuracy	0.836/0.709	0.874	0.72

Classification accuracies on the 20 test sets of MNIST, CIFAR-10 and EMNIST are boxplotted in Figure 2.6, Figure 2.7 and Figure 2.8, respectively. It is clear that the generalisation-3 and the generalisation-4 have higher boxes than Titterington's model in Figure 2.6 and Figure 2.7. This indicates the effectiveness of our generalisations when the data satisfy the associated conditions: in our experiments, the MNIST dataset satisfies the feature-assessment dependency condition in

Table 2.3: Comparing the generalisation-2 with the original model on multi-class classification tasks.

Setting	Two-class(one-versus-all)	Multi-class	
Model	Original	Generalisation-2	
Model's setting		Generalisation-3	Generalisation-4
Data set	EMNIST	EMNIST	EMNIST
Average accuracy	0.401	0.403	0.502

the generalisation-3 and the CIFAR-10 dataset satisfies the multi-modality condition in the generalisation-4.

For the EMNIST data, the generalisation-3 and generalisation-4 produce higher boxes than Titterington's model and the generalisation-4 has the best classification performance. This also shows the effectiveness of our models. Note that here the generalisation-4 has much better classification performance than the generalisation-3. One possible reason is that the multi-modal classes have more effect on the final results than the feature-dependent assessment, since the subclasses in each large class are clearly defined while the linear relationship between the assessment and features is not strong, as shown in Table 2.1. We also note that there is a large space for improvement in classification accuracy of EMNIST. By developing a new method that can deal with feature-dependent assessments and multi-modal classes together, we may further improve the classification performance on complex data such as EMNIST. We list this as our future work in the conclusions section.

2.4 Conclusions

In this chapter, we extended stochastic supervision models in four aspects, generalising them to asymmetric assessments, multiple classes, feature-dependent assessments, and multi-modal classes, respectively, to enhance their applicability. The experiments on both simulated data and real-world data demonstrate the effectiveness of our generalisations.

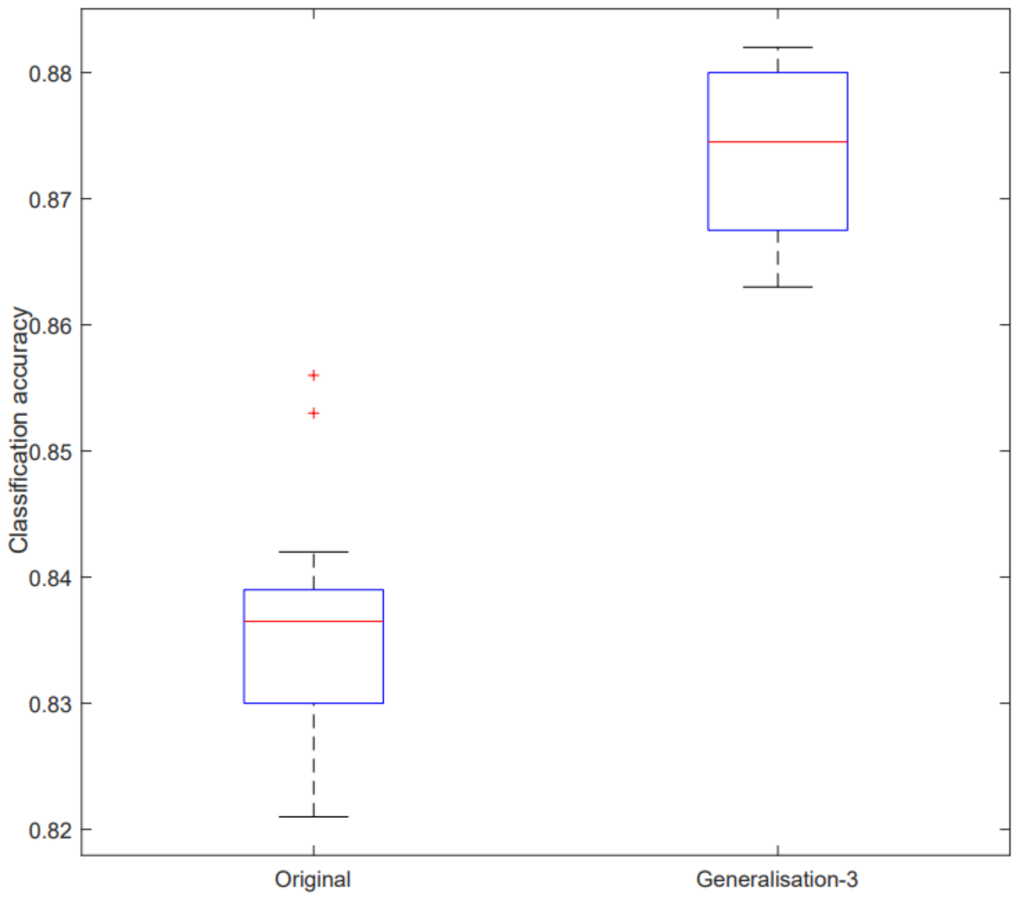


Figure 2.6: Classification accuracies of Titterington's model and the generalisation-3 on 20 test sets of MNIST. Comparing to the original stochastic supervision model, our generalisation-3 has higher classification accuracy on the MNIST dataset.

2.5 Future work

In the future, to enhance further our models' flexibility and generality, we shall explore nonlinear modelling for the relationship between assessments and features, as well as more sophisticated techniques for multi-modality modelling.

Except for nonlinear relations, the multiple supervisors is an important direction since multiple supervisors can not only increase the flexibility of the stochastic supervision models but also increase the model performance by assembling the assessments of different supervisors. Furthermore, since the multiple stochastic supervision model is closely related to ensemble learning, the ensemble of multiple stochastic supervision is an important problem to discuss and explore.

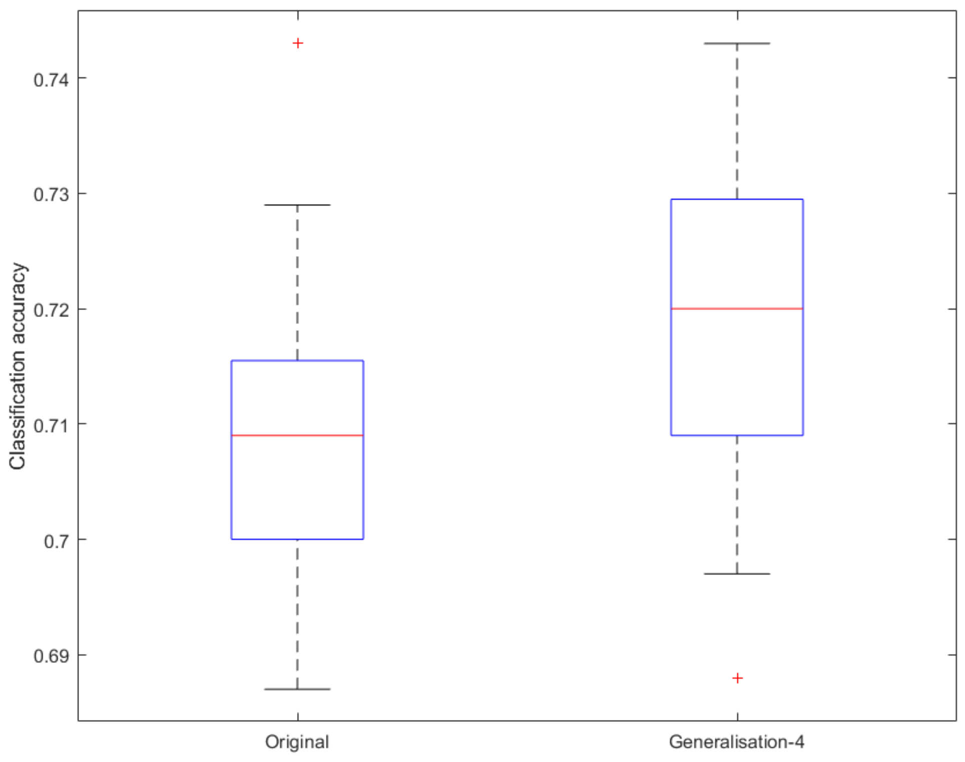


Figure 2.7: Classification accuracies of Titterington's model and the generalisation-4 on 20 test sets of CIFAR-10. Comparing to the original stochastic supervision model, our generalisation-4 has higher classification accuracy on the CIFAR-10 dataset.

Utilising the stochastic supervision model for semi-supervised learning is another interesting direction since in semi-supervised learning we can use the labelled sample to train an initial model and use the initial model to provide assessments for unlabelled samples. This method is also closely related to self-training. Since most existing self-training methods do not take the distribution of assessments into account, using the stochastic supervision model for semi-supervised learning potentially can improve the model performance.

Moreover, instead of using a fixed threshold of w to infer y, we propose to learn this threshold from data. Since we use the transformation $w_i = \log z_i/z_J$ to transform a softmax vector to a $(J-1)$ dimensional normal distributed random variable, learning the threshold of w is equivalent to giving different weights to different classes.

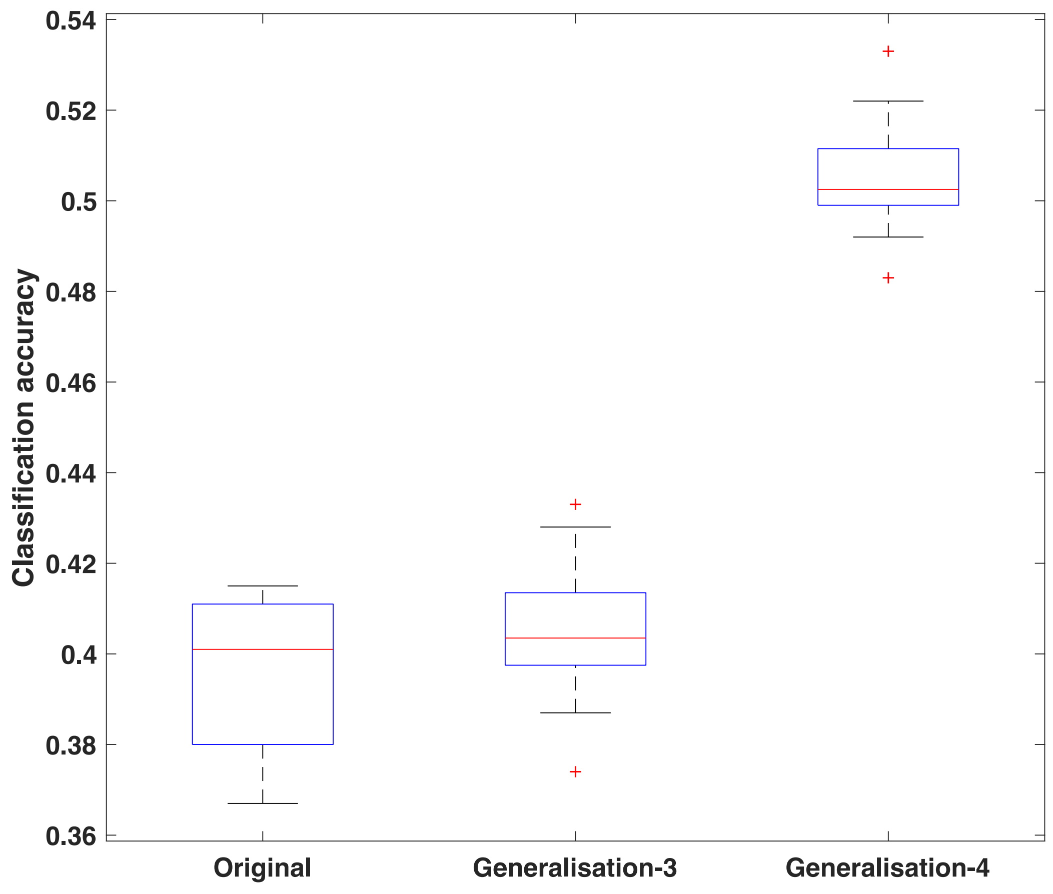


Figure 2.8: Classification accuracies of Titterington's model, generalisation-3 and generalisation-4 on 20 test sets of EMNIST. Comparing to the original stochastic supervision model, our generalisation-3 and generalisation-4 has higher classification accuracy on the EMNIST dataset.

By utilising the learned threshold, our model can adapt to more real-world scenarios where different classes have different importance. In addition, we propose to develop new algorithms that can provide superior classification performances under more complex situations, e.g. with both feature-dependent assessment and multi-modal classes.

Part II

Contribution to Incomplete Supervision

Chapter 3

Semi-supervised domain adaptation with implicit importance weight estimation for hand pose

Semi-supervised learning is a typical setting of incomplete supervision in weakly supervised learning. Compared with supervised learning, semi-supervised learning is a more general setting in machine learning applications. However, it is often insufficient to train a good model if we only utilise the limited number of labelled data of the task. In many machine learning applications, such as hand pose estimation, there are only a limited number of labelled real data since labelling by humans is expensive, but fortunately, there are plenty of labelled training samples in the synthetic data, for example, of hand pose images. Therefore, we expect that transferring knowledge of training instances and feature representation from the synthetic data can improve the model performance of the semi-supervised learning algorithm.

Since hand pose estimation is a typical task suitable for semi-supervised learning and domain adaptation, in this chapter we develop a semi-supervised learning algorithm with domain adaptation for hand pose estimation. We propose a domain adaptation method to address the insufficient labelled real data in semi-supervised learning under the covariate shift assumption. Since there is no explicit density function for complicated features of hand pose data, the classic importance weight estimation methods (e.g. density ratio estimation based on explicit distributions) fail in this case. Therefore, we use a deep neural network to estimate the importance weights for implicit data distributions. Our method achieves the state-of-the-art performance on the most widely used benchmark dataset, and we show that our weighted loss outperforms its unweighted counterparts.

3.1 Introduction

Semi-supervised learning is an important topic in the applications of machine learning, since in many applications it is hard to get enough labelled real training sample while unlabelled samples are easy to collect. Hand pose estimation is a machine learning application which is suit for semi-supervised learning. Hand pose estimation aims to infer the 3D locations of hand joints from one or a few images. It is a crucial task in many applications, such as virtual/augmented reality, human-machine interaction, and vision-based human action/activity understanding. In hand pose estimation accurate labelling of 3D hand poses by human annotators is expensive and time-consuming however, unlabelled samples are very cheap to collect. Recently, deep learning based methods have received success in many machine vision applications [59]. For the pose estimation problem, deep learning models can solve the feature learning and pose prediction tasks in an end-to-end manner and have achieved the state-of-the-art performance [60]. However, these methods require a great quantity of accurately labelled real images as training samples.

In order to reduce the required amount of labelled real data of hand poses, several attempts have been made to apply pure semi-supervised learning without knowledge transfer from synthetic data to hand pose estimation [61, 62], since semi-supervised learning methods only require a partially labelled training set. However, semi-supervised learning based approaches build on a small set of labelled real images, while such a small training set often comes from only a subset of the multimodal population. For example, in hand pose estimation applications, labelled training samples often from one or few people, however, the test set usually contains samples from many different people. That is, the training set may only contain little information of the test data. Since labelled training samples only cover few mode of the multi-mode distribution of the test set, the distributions of training samples and test samples can also be quite different.

In order to complete the insufficient labelled training samples, it is necessary to transfer knowledge of training instances and feature representation from labelled synthetic data. Many recent studies have tried to use synthetic images created by computer graphic software [63, 64, 65], since acquiring labelled synthetic images is much cheaper and easier. However, since real images and synthetic images belong to different domains (features from real images and features from synthetic images have different distributions), exploiting labelled synthetic images introduces a domain gap to model fitting, resulting in a dataset shift problem in machine learning, and thus the model fitted by synthetic images usually has a suboptimal performance on the test set of real images. However, underlying most machine learning algorithms, it is assumed that the test data and the training data have a similar distribution. As discussed above, this assumption is too strong to the synthetic images based or semi-supervised learning based hand pose estimation applications: when the training data and the test data have different distributions, the performance of the fitted model on the test data is prone to a remarkable decline.

In this chapter, we aim to propose an improved semi-supervised method based on transfer knowledge of training instances and feature representation from synthetic data and address the dataset shift problem that appears in both synthetic images based methods and semi-supervised learning based methods for hand pose estimation. The motivation of this work is illustrated in Figure 3.1. As shown in [66], the dataset shift problem can be solved through reweighting training samples by the density ratio between the test set and the training set. After the reweighting, the expected loss on the test data can reach the minimum. That is, we minimise the expected test loss by re-weighting instances in the source domain (training set): instances that are more similar to the ones in the target domain (test set) are weighted more. However, both the feature distribution of synthetic images and the feature distribution of real images are, unfortunately, unknown implicit distributions, so that the classical importance weight estimation methods cannot apply here since we cannot explicitly compute the density ratio. Therefore, in this paper, we propose a domain adaptation method to estimate the weight of each training instance by a weighting net. The relationship between the expected test loss minimisation and the optimisation of the weighting net is also discussed in this paper. Our method produces the state-of-the-art performance among all 2D input based hand pose estimation methods on the most widely used NYU benchmark dataset.

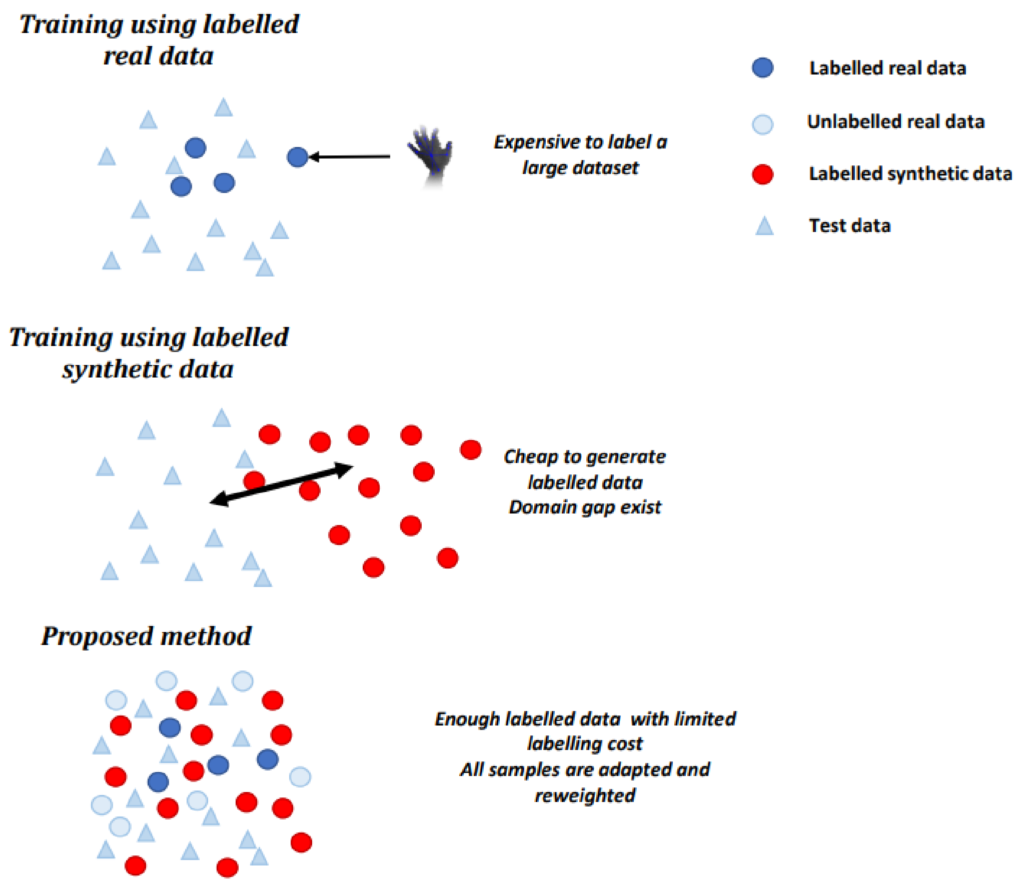

Training using labelled real data
Expensive to label a large dataset
Labelled real data
Unlabelled real data
Labelled synthetic data
Test data
Training using labelled synthetic data
Cheap to generate labelled data
Domain gap exist
Proposed method
Enough labelled data with limited labelling cost
All samples are adapted and reweighted

Figure 3.1: Motivation. Since labelling real samples by human annotators are expensive, real labelled data based methods often do not have enough training samples. Synthetic data based methods have enough labelled samples, but there exists a domain gap between synthetic training data and real test data. This work aims to provide a method that only requires a limited number of real labelled data and reduces the effect of the domain gap by adapting and reweighting all training samples.

The contributions of this work can be summarised as follows:

- We propose an improved semi-supervised method based on domain adaptation method with implicit importance weight estimation, which is realised by a deep neural network.

- We illustrate the data bias issue in the scenario of using synthetic data or partially labelled real data for hand pose estimation and offer a solution to this issue.
- We build a new hand pose estimation model with our proposed domain adaptation network and demonstrate that the model achieves state-of-the-art performance.

3.2 Related work

Pose estimation is an important task for many machine vision applications, such as motion reconstruction [67], hand tracking [68] and dance analysis [69, 70]. Many early work of hand pose estimation are model based [71, 72]. Model-based methods fit an explicit hand model by optimise a pre-specified cost function. However, these methods are sensitive and some models require user-based parameters [73]. Recently data-driven based hand pose estimation methods achieve great success. These methods learn a mapping from images to hand pose [74, 75]. However, most of these methods use hand-crafted features, which require human's prior knowledge to design or choose, and the performances of these methods are lower than current deep learning based methods.

In recent years, many studies [59, 76, 60] have shown that deep learning models can achieve the state-of-the-art performances in many computer vision applications. Since deep learning not only solve the feature learning and pose prediction tasks in an end-to-end manner but also learn a sophisticated function to predict 3D keypoints' location, it is natural to apply deep learning for hand pose estimation. Tompson et al. [1] made an early attempt to apply deep learning to predict locations of 3D keypoints. After that, Panteleris et al. [77] proposed a method which can estimate 3D keypoints' locations from a single camera and achieve the state-of-the-art performances.

However, the number of labelled training samples play a crucial role in deep learning [78]. Since annotating real images of hand poses requires a great amount of manual work, it is difficult to obtain a sufficient number of labelled training samples. In order to solve this problem, many methods have been reported in recent years. These hand pose estimation methods can be mainly categorised into three classes:

1. Semi-supervised learning based hand pose estimation methods (using both labelled and unlabelled real data).

2. Directly use synthetic data for hand pose estimation.

3. Combining semi-supervised learning with domain adaptation for hand pose estimation (using labelled synthetic data, and labelled and unlabelled real data).

Unlike supervised learning methods which only utilise labelled real data, many semi-supervised learning methods also exploit unlabelled real data. Poier et al. [79] developed a method that learns representations from unlabelled data. They assume that good features of hand pose can get appearance information from any viewpoint so that given an image from one viewpoint, the representation can minimise the reconstruction error of appearance from other viewpoints. This loss term can effectively use information in the unlabelled data, however, its exploitation of the information provided by only a few labelled real samples limits the performance of this method.

Since it is easy to obtain labelled training samples rendered from a 3D model, using synthetic data to train a neural network is a reasonable solution [80]. However, synthetic data is still quite different from real data, making the training data and the test data from two different domains, hence directly using synthetic data may lead to the problem of overfitting to the synthetic data. Although some studies [81, 80] have tried to relieve the overfitting problem by using pre-trained features, complicated augmentation, and fine-tuning techniques, synthetic data based methods are still affected by the domain difference, and the performance is still lower than the methods using both synthetic data and real data [82].

Many recent studies not only exploit knowledge from synthetic data but also extract information from unlabelled real data and few labelled real data [83, 61, 84, 85, 82]. Since these methods contain training samples from different domains, they adopt domain adaptation methods to minimise the domain gap. The method proposed in [82] learns a mapping between the real data feature space and the synthetic data feature space. The mapping is learned by using paired training samples (pairs of synthetic image and real image of the same pose), however, to use a large number of paired training samples still requires the labelling of many real images.

Generative adversarial networks [86] (GANs) is a generative model that can generate new synthetic samples that resemble to real samples. GANs have also been extended to transfer learning [87] and domain adaptations [88]. For pose estimation, Shrivastava et al. [89] and Mueller et al. [84] use GANs to transfer synthetic images to real-like images. However, there are two drawbacks of these GANs based methods. Firstly, GANs often suffer from the so-called mode collapse problem, and synthetic images are often mapped to one or a few modes of real image distributions, leading to the dataset bias. Secondly, these methods often assume that the few labelled real data have the same distribution as the test set. This assumption is unrealistic in our applications, since the test set of hand pose estimation often from many different people while the training samples are often only from one or few people.

Yamada et al. [90] proposed a method to solve the domain adaptation problem between the real training set and the test set under the covariate shift assumption. This work is the one most relevant to our work. However, there are still many differences between their method and ours. Firstly, we address a domain adaptation problem with two heterogeneous source domains. Secondly, we work on the feature space rather than the input space. Thirdly, we do not make the Gaussian kernel assumption for the importance weight, and the feature distributions can be any implicit distribution in our case.

3.3 Proposed method

As illustrated in Figure 3.2, the objective of the proposed method is to provide an improved semi-supervised hand pose estimation model, that can exploit information in both partially labelled real data and fully labelled synthetic data. In this section, we first give the formulation of our problem, then describe the approximation method of the objective function of our model and the network architecture, and finally provide the learning algorithms. The aim of this work is to transfer knowledge of instances and feature representation from synthetic data to improve the performance of the semi-supervised hand pose estimation methods.

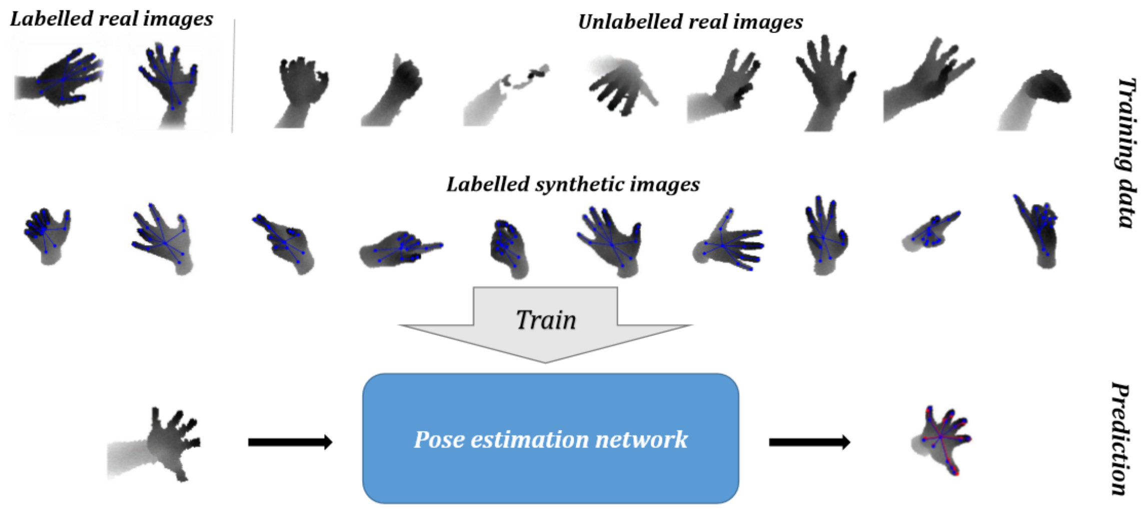


Figure 3.2: Overview of the proposed method. The proposed method aims to use a few labelled real depth images and plenty of labelled synthetic depth images and unlabelled real depth images to train a hand pose estimation network to predict locations of hand joints. The blue lines indicate the labelled hand pose, while the red lines are for the predicted hand pose. All illustrative example in this figure, including both real depth images and synthetic depth images, are extracted from the NYU dataset.

3.3.1 Problem formulation

A hand pose estimation model can be represented by a function $y = f(x)$ that fits on the source domain (training set) to predict the corresponding hand poses. In our work, the input is a depth image and the output is the 3D location of 14 keypoints of a hand. The covariate x is a vector with size 16384×1 (a flattened depth image with size 128×128) and the prediction y is a 42×1 vector denoting the 3D locations (3×1) of 14 joints of a hand. In the proposed method, the training set has two source domains: the fully labelled synthetic data $D_s = \{(x_i^s, y_i^s)\}_{i=1}^{n_1}$ containing n_1 labelled synthetic samples and the partially labelled original real data (or say old collected real data) $D_{r-old} = \{(x_i^r, y_i^r)\}_{i=1}^{n_2} \cup \{(x_j^r)\}_{j=n_2+1}^{n_3}$ containing totally n_3 real samples, of which n_2 samples are labelled and $n_3 - n_2$ samples are unlabelled. The test set $D_{t-new} = \{(x_i^t)\}_{i=1}^{n_4}$ contains n_4 new unlabelled test samples. Unlike the settings in the classic supervised learning or semi-supervised learning that assumes the covariate x has the same distribution on the old training set and the new test set, in real hand pose estimation applications, $p_s(x^s)$, $p_{r-old}(x^r)$, and $p_{t-new}(x^t)$ usually have different distributions. Since $p_s(x^s)$ are synthetic data and $p_{t-new}(x^t)$ are real data, the distribution of $p_s(x^s)$ and $p_{t-new}(x^t)$ are different. For the old real data $p_{r-old}(x^r)$ and the new test data $p_{t-new}(x^t)$, there are mainly two reasons that lead to the distribution of $p_{r-old}(x^r)$ and $p_{t-new}(x^t)$ being also different. Firstly, the training data often have a sample selection bias. (For example, many widely used hand pose estimation datasets only contain hand images collect from one or a few adults. For real applications, test samples from many different people and may contain children's hand pose images. Using new unlabelled real data can relieve the sample selection bias of the original real data.) Secondly, in real applications the environment is not necessarily stationary, the changing of the environment also leads to the data shift. Due to these reasons, we can view the new arriving unlabelled real data as the target domain. In applications, we often use a subset of new unlabelled real samples (with $n_5, n_5 < n_4$ samples) to construct a validation set. The validation set is utilised to estimate the difference between the old real data and new real data. Our task is to fit a model that minimises the expected loss $\mathbb{E}_{p_{t-new}(x^t)}[\mathscr{L}(f(x^t), y^t)]$ under certain loss function $\mathscr{L}$ in the target domain.

The rationale of the proposed method is that, given that the joint probability density functions of the complicated features are implicitly unknown in both the source domains and the target domain, in order to accurately re-weight the training samples from different source domains to well adapt to the target domain, a deep neural network should be developed to accurately estimate the importance weights for the implicit density functions. Therefore, our method mainly contains two parts: a hand pose estimation net and a weighting net. Figure 3.3 illustrates the problem setting and system architecture of our method.

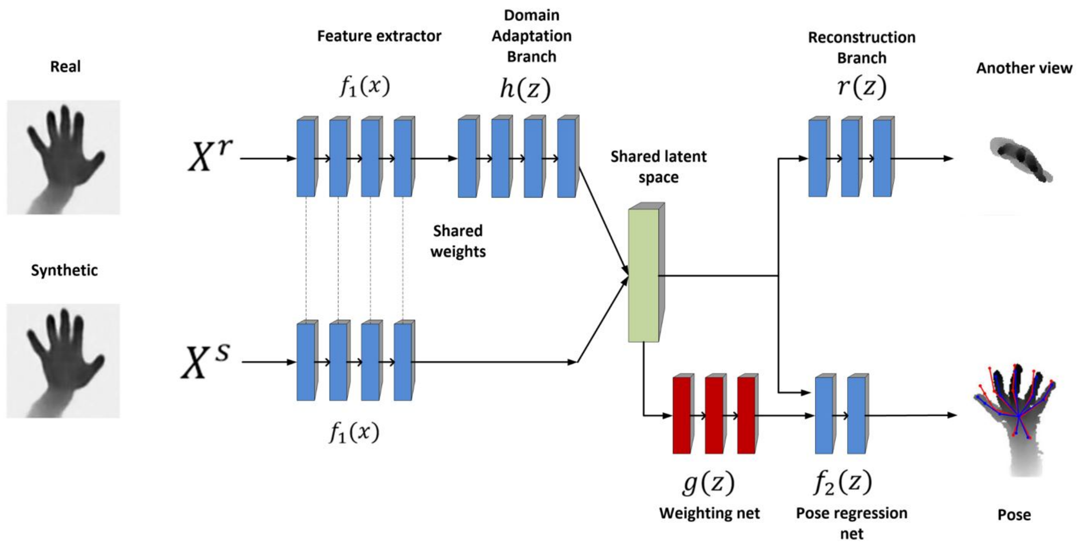


Figure 3.3: The problem setting and system architecture of the proposed method: Stacks of rectangles represent some neural networks; x denotes an input image and z stands for the features in the latent space; the red part is the weighting net $g(z)$, the blue part belongs to the hand pose estimation net which contains four modules $\{f_1(x), h(z), f_2(z), r(z)\}$, and the green part is the shared latent space. The feature extractor $f_1(x)$ learns a shared feature space to transfer knowledge of feature representation from labelled synthetic data. The weighting net $g(z)$ learns to weighting training samples to transfer knowledge of instances from labelled synthetic data. The domain adaptation branch $h(z)$ learns to minimize domain gap between real data (both labelled and unlabelled) and synthetic data. The reconstruction branch learns to recover an image of another view to ensure features in the shared latent space are pose invariant for different view. The pose regression net $f_2(z)$ learn a mapping to utilise feature representation in the shared latent space to estimate the pose.

3.3.2 Weighted empirical risk minimisation

The hand pose estimation net $f(x) = f_2(f_1(x))$ contains two cascading modules: the feature extractor $f_1(x) = z$ extracts features from the covariate x and learns a shared feature space for both synthetic data and real data; The aim of learning a shared feature space is to find a feature representation that can reduce difference of different data domain and improves the pose estimation accuracy. The regression net $f_2(z) = y$ predicts the hand pose by using the extracted features in the shared latent space. Both $f_1(x)$ and $f_2(z)$ are neural network. $f_1(x)$ is a ResNet [91] and $f_2(z)$ is a fully connected neural network. Since the data probability distributions $p_s(x^s)$, $p_{r-old}(x^r)$ and $p_{t-new}(x^t)$ in the three domains (two source domains and one target domain) are different, the distributions of extracted features $z^s = f_1(x^s)$, $z^r = h(f_1(x^r))$ and $z^t = h(f_1(x^t))$ also differ from each other.

The empirical risk minimisation (ERM) [92] to estimate model parameters θ, can be formulated as a standard way to learn the parameters;

$$\theta_{ERM} = \arg\min_{\theta} \left[\frac{1}{n_1} \sum_{i=1}^{n_1} \mathscr{L}(f_2^{\theta}(z_i^s), y_i^s) + \frac{1}{n_2} \sum_{j=1}^{n_2} \mathscr{L}(f_2^{\theta}(z_j^r), y_j^r) \right],$$

However, when $p_s(z^s) \neq p_{t-new}(z^t)$ and $p_{r-old}(z^r) \neq p_{t-new}(z^t)$, the parameters learned by this ERM will fail to converge to the optimal parameters [66],

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{p_{t-new}(z^t)}[\mathscr{L}(f_2^{\theta}(z^t), y^t)],$$

in the target domain.

Since the following identity holds:

$$\begin{aligned} \mathbb{E}_{p_t(z^t)}[\mathscr{L}(f_2^{\theta}(z^t), y^t)] &= \int p_{t-new}(z^t)[\mathscr{L}(f_2^{\theta}(z^t), y^t)]dz^t \\ &= \int p_s(z^t) \frac{p_{t-new}(z^t)}{p_s(z^t)} [\mathscr{L}(f_2^{\theta}(z^t), y^t)]dz^t \\ &= \int p_{r-old}(z^t) \frac{p_{t-new}(z^t)}{p_{r-old}(z^t)} [\mathscr{L}(f_2^{\theta}(z^t), y^t)]dz^t, \end{aligned} \tag{3.1}$$

in order to guarantee that the estimated parameters converge to the optimal parameters θ^* in the target domain, we can use the weighted empirical risk minimisation [66] (WERM) function to fit the model:

$$\theta_{WERM} = \arg\min_{\theta} \left[\frac{1}{n_1} \sum_{i=1}^{n_1} W_1(z_i^s) \mathscr{L}(f_2^{\theta}(z_i^s), y_i^s) + \frac{1}{n_2} \sum_{j=1}^{n_2} W_2(z_j^r) \mathscr{L}(f_2^{\theta}(z_j^r), y_j^r) \right], \tag{3.2}$$

where the weights

$$W_1(z_i^s) = \frac{p_{t-new}(z_i^s)}{p_s(z_i^s)} \text{ } and \text{ } W_2(z_j^{r-old}) = \frac{p_{t-new}(z_j^r)}{p_{r-old}(z_j^r)}$$

are density ratios of a given instance z between the target domain and the corresponding source domains. The loss term $\frac{1}{n_1}\sum_{i=1}^{n_1} W_1(z_i^s)\mathscr{L}(f_2^\theta(z_i^s), y_i^s)$ and $\frac{1}{n_2}\sum_{j=1}^{n_2} W_2(z_j^r)\mathscr{L}(f_2^\theta(z_j^r), y_j^r)$ transfers the knowledge of training instances of both real data and synthetic data to the target task.

Unfortunately, the density functions of $p_s(x^s)$, $p_{r-old}(x^r)$ and $p_{t-new}(x^t)$ are unknown, neither its form nor its parameters, hence we cannot directly compute weights $W_1(z_i^s)$ and $W_2(z_j^r)$, both of which are actually density ratios. Therefore, we propose to develop a deep neural network to estimate the density ratios $\frac{p_{t-new}(z_i^s)}{p_s(z_i^s)}$ and $\frac{p_{t-new}(z_j^r)}{p_{r-old}(z_j^r)}$.

3.3.3 Weighting net

The weighting net $u = g(z)$ estimates density ratios to assign each training sample an importance weight to improve the model's performance in the target domain. The input of the weighting net is a vector of extracted features z, either from two source domains or from the target domain, and the output is a softmax vector $u = (u_1, u_2, u_3)$ and u are related to class probabilities of three domains:

$$\left(\frac{p_s(z)}{p_s(z)+p_{r-old}(z)+p_{t-new}(z)}, \frac{p_{r-old}(z)}{p_s(z)+p_{r-old}(z)+p_{t-new}(z)}, \frac{p_{t-new}(z)}{p_s(z)+p_{r-old}(z)+p_{t-new}(z)}\right).$$

There are mainly two reasons to use the samples z in the shared latent space (the space generated by the extracted features) rather than the image x in the input space (the space generated by the input images). Firstly, the dimension of input space is greatly higher than that of the latent space, and the curse of the dimensionality leads to the data distribution being too sparse to accurately estimate the density ratio. Secondly, in the lower-dimensional latent space the overlap of the supports of data distributions is usually larger than that in the input space. This will facilitate the estimation of density ratios, because, if we estimate the density ratio for the samples with no support overlap between distributions, the estimated density ratio tends to be zero or extremely large. In addition, similar to the domain classifier (discriminator) in many adversarial model based domain adaptation papers, such as [88, 93], the softmax outputs $u = (u_1, u_2, u_3)$ are related to class probabilities $\left(\frac{p_s(z)}{p_s(z)+p_{r-old}(z)+p_{t-new}(z)}, \frac{p_{r-old}(z)}{p_s(z)+p_{r-old}(z)+p_{t-new}(z)}, \frac{p_{t-new}(z)}{p_s(z)+p_{r-old}(z)+p_{t-new}(z)}\right)$. In the following part we show that, when the weighting net becomes optimal, we can directly estimate the density ratios from $u = (u_1, u_2, u_3)$.

The loss function of the proposed weighting net is

$$\begin{aligned}\mathscr{L}_g = &\frac{1}{n_1}\sum_{i=1}^{n_1}\left[1 - \frac{u_1(z_i^s)}{u_2(z_i^s)}\right] + \frac{1}{n_3}\sum_{j=1}^{n_3}\left[\frac{1}{2}\left(\frac{u_1(z_j^r)}{u_2(z_j^r)}\right)^2 - \frac{1}{2}\right] \\ &+\frac{1}{n_1}\sum_{i=1}^{n_1}\left[1 - \frac{u_1(z_i^s)}{u_3(z_i^s)}\right] + \frac{1}{n_5}\sum_{j=1}^{n_5}\left[\frac{1}{2}\left(\frac{u_1(z_j^t)}{u_3(z_j^t)}\right)^2 - \frac{1}{2}\right],\end{aligned} \tag{3.3}$$

where $z_j^t = f_1(x_j^t)$ denotes the extracted features of $x_j^t \in D_v = \{(x_i^t)\}_{i=1}^{n_5}$. This loss function helps the weighting net to learn the density ratios. As this loss decrease, the output of the weighting net $u = (u_1, u_2, u_3)$ will converge to the class probability of three domains so that we can use the class probability to compute the density ratio. The derivation of this loss function is attached in the appendixA. The validation set D_v is a subset of the whole target domain D_t. We only use features of D_v to estimate the density $p_t(z)$ of the target domain, the rest part of the D_t is kept untouched.

The following proposition shows when the weighting net converges to the optimum, $u = (u_1, u_2, u_3)$ converges to:

$$\left(\frac{p_s(z)}{p_s(z)+p_{r-old}(z)+p_{t-new}(z)}, \frac{p_{r-old}(z)}{p_s(z)+p_{r-old}(z)+p_{t-new}(z)}, \frac{p_{t-new}(z)}{p_s(z)+p_{r-old}(z)+p_{t-new}(z)}\right)$$

.

Proposition 3.3.1. *When the weighting net arrives at the optimum, the following properties hold:*

$$\frac{u_1}{u_2} = \frac{p_s(z)}{p_{r-old}(z)}, \frac{u_1}{u_3} = \frac{p_s(z)}{p_{t-new}(z)}, \frac{u_2}{u_3} = \frac{p_{r-old}(z)}{p_{t-new}(z)}$$

Proof. The expected loss of the weighting net is

$$\begin{aligned}\mathbb{E}[\mathscr{L}_g] = &\mathbb{E}_{p_s(z^s)}\left[1-\frac{u_1(z^s)}{u_2(z^s)}\right]+\mathbb{E}_{p_{r-old}(z^r)}\left[\frac{1}{2}\left(\frac{u_1(z^r)}{u_2(z^r)}\right)^2-\frac{1}{2}\right]\\ &+\mathbb{E}_{p_s(z^s)}\left[1-\frac{u_1(z^s)}{u_3(z^s)}\right]+\mathbb{E}_{p_{t-new}(z^t)}\left[\frac{1}{2}\left(\frac{u_1(z^t)}{u_3(z^t)}\right)^2-\frac{1}{2}\right]\end{aligned} \tag{3.4}$$

$$\begin{aligned}= &\int p_s(z)\left[1-\frac{u_1(z)}{u_2(z)}\right]+p_{r-old}(z)\left[\frac{1}{2}\left(\frac{u_1(z)}{u_2(z)}\right)^2-\frac{1}{2}\right]\\ &+p_s(z)\left[1-\frac{u_1(z)}{u_3(z)}\right]+p_{t-new}(z)\left[\frac{1}{2}\left(\frac{u_1(z)}{u_3(z)}\right)^2-\frac{1}{2}\right]dz.\end{aligned} \tag{3.5}$$

Since

$$\begin{aligned}\mathbb{E}[\mathscr{L}_g] \geq \int \min_{u_1,u_2,u_3} &\left\{p_s(z)\left[1-\frac{u_1(z)}{u_2(z)}\right]+p_{r-old}(z)\left[\frac{1}{2}\left(\frac{u_1(z)}{u_2(z)}\right)^2-\frac{1}{2}\right]\right.\\ &\left.+p_s(z)\left[1-\frac{u_1(z)}{u_3(z)}\right]+p_{t-new}(z)\left[\frac{1}{2}\left(\frac{u_1(z)}{u_3(z)}\right)^2-\frac{1}{2}\right]\right\}dz,\end{aligned}$$

when the optimal u of the function $f_u(u)=\mathbb{E}[\mathscr{L}_g]$ must satisfy

$$\begin{cases} -p_s(z)u_1(-1)\frac{1}{u_3^2}+p_{t-new}(z)(\frac{u_1}{u_3})u_1(-1)\frac{1}{u_3^2}=0,\\ -p_s(z)u_1(-1)\frac{1}{u_2^2}+p_{r-old}(z)(\frac{u_1}{u_2})u_1(-1)\frac{1}{u_2^2}=0.\end{cases} \tag{3.6}$$

Hence we have

$$\begin{cases} \frac{u_1}{u_2}=\frac{p_s(z)}{p_{r-old}(z)},\\ \frac{u_1}{u_3}=\frac{p_s(z)}{p_{t-new}(z)},\\ \frac{u_2}{u_3}=\frac{p_r(z)}{p_{t-new}(z)}.\end{cases} \tag{3.7}$$

□

Based on the above proposition, weights $W_1(z)$ and $W_2(z)$ can be approximated by $\hat{W}_1(z)=\frac{u_3}{u_1}$ and $\hat{W}_2(z)=\frac{u_3}{u_2}$.

3.3.4 Hand pose estimation net

The hand pose estimation net contains three branches: the backbone $f(x) = f_2(f_1(x))$, the domain adaptation branch $h(f_1(x^r))$ and the reconstruction branch $r(z)$.

The backbone part is shared by both synthetic data and real data. It contains two modules: the feature extraction network $f_1(x)$ and the pose regression network $f_2(z)$. The loss function of the backbone part is the weighted L_2 loss:

$$\mathscr{L}_{pose} = \frac{1}{n_1}\sum_{i=1}^{n_1}\hat{W}_1(z_i^s)\|\hat{y}(z_i^s) - y_i^s\|_2 + \frac{1}{n_2}\sum_{i=1}^{n_2}\hat{W}_2(z_i^r)\|\hat{y}(z_i^r) - y_i^r\|_2, \tag{3.8}$$

where $\hat{y}(z_i^s) = f_2(z_i^s)$ is the estimated pose of the ith synthetic sample and y_i^s is its corresponding ground truth. Similarly, $\hat{y}(z_i^r) = f_2(z_i^r)$ is the estimated pose of the ith real sample and y_i^r is the corresponding label.

Similarly to prior studies [94, 82], we use pairwise samples (x_i^s, x_i^r) that have the same pose between the real training set and the synthetic set to learn a mapping from the extracted feature of real data to the shared feature space. The domain adaptation branch $h(f_1(x^r))$ is a network whose loss function contains two loss terms. The first loss term $\mathscr{L}_{ada}$ aligns all paired training samples in the feature space:

$$\mathscr{L}_{ada} = \sum_{i=1}^{k}\|h(f_1(x_i^r)) - f_1(x_i^s)\|_2,$$

where k is the number of paired training samples. Since the paired synthetic sample and real sample have the same pose and the pose estimation net is a deterministic function, the paired synthetic sample and real sample must have the same feature in the feature space. This loss forces the data with the same pose but from different domains to align to each other in the feature space. The second loss term is the adversarial loss $\mathscr{L}_{adv}$ which minimises the Pearson divergence between distributions of two source domains and the target domain:

$$\mathscr{L}_{adv} = \frac{1}{n_3}\sum_{j=1}^{n_3}\left[\frac{1}{2}\left(1 - \frac{u_1(z_j^r)}{u_2(z_j^r)}\right)^2\right] + \frac{1}{n_5}\sum_{j=1}^{n_5}\left[\frac{1}{2}\left(1 - \frac{u_1(z_j^t)}{u_3(z_j^t)}\right)^2\right],$$

where n_5 is the number of unlabelled sample in the target domain to estimate the density ratio.

The reconstruction branch $r(z)$ guarantees the features extracted from unlabelled data contain pose information and such pose information is not related to a specific camera view. Therefore, the extracted feature from a camera's view a must be able to reconstruct the input image in another camera's view (or same camera in another angle) b [79]. The loss function of the reconstruction branch is

$$\mathscr{L}_{re} = \frac{1}{n_1+n_3} \sum_{i=1}^{n_1+n_3} \|r(x_i^a) - x_i^b\|, \tag{3.9}$$

where x_i^a and x_i^b are the same training sample in two different camera's views: camera a's view and camera b's view.

Hence the total loss of the hand pose estimation net is

$$\mathscr{L}_{total} = \lambda_0 \mathscr{L}_{pose} + \lambda_1 \mathscr{L}_{ada} + \lambda_2 \mathscr{L}_{adv} + \lambda_3 \mathscr{L}_{re},$$

where λ_0, λ_1, λ_2 and λ_3 are weights of the loss terms.

3.3.5 Implementation details

We followed the pre-processing procedure introduced in Deepprior++ [95]. Each sample was cropped (resized) into a 128×128 patch (keep the central 128×128 pixels of the image and discard margin part) around the hand location.

We used a network architecture similar to Deepprior++, since Deepprior++ is a widely used network architecture and achieves good performance in hand pose estimation tasks. The feature extractor $f_1(x)$ is a ResNet [91] with four residual blocks. Each residual block has the same architecture as that in Deepprior++. The regression net $f_2(z)$ is a simple fully connected neural network with two hidden layers and each layer has 1024 hidden units. The output of $f_2(z)$ is a matrix with size 3×14. The domain adaptation branch $h(z)$ consists of four fully connected layers. The reconstruction branch $r(z)$ has the same architecture of the generator of DCGAN [96]. The weighting net $g(z)$ is a simple 3-layer fully connected network and each hidden layer has 256 hidden units. Details of the pre-train and the training algorithm can be found in Pre-train Algorithm 1 and Algorithm 2.

Algorithm 1 Pre-train algorithm

```
  Pre-training step:
Require: $D_s = \{(x_i^s, y_i^s)\}_{i=1}^{n_1}$
Require: Batch size $b$, learning rate $\gamma$.
  for number of epochs do
    for number of batches do
      Sample $\{(x_i^s, y_i^s)\}_{i=1}^{b}$
      Update $\theta_{f_1}$ and $\theta_{f_2}$:
          $$(\theta_{f_1}, \theta_{f_2})_{t+1} = (\theta_{f_1}, \theta_{f_2})_t - \gamma \bigtriangledown \frac{1}{b} \sum_{i=1}^{b} \|f_2(f_1(x_i^s)) - y_i^s\|_2$$
    end for
  end for
  return $\theta_{f_1}, \theta_{f_2}$.
```

As the importance weight $W \in (0, \infty)$ can be very large, multiplying an unusually large weight may cause the optimisation algorithm to converge to a trivial destination or even fail to converge. Since weights only measure the relative importance of training samples, in order to stabilise the training we clipped and normalised the weight (by having $W_i = b\frac{W_i}{\sum_{i=1}^{b} W_i}$, where b is the batch size).

As for the computational cost, we use a Nvidia TITAN Xpascal graphics card with 12G memory to train our model. The overall training time is 46 hours.

3.4 Experiments

This section contains four experiments: two main experiments and two ablation studies. The aim of the first main experiment is to compare the performance of our method with other state-of-the-art (SOTA) methods, under the setting that is prevalent in real applications (a training set with partially labelled real data). The experiment is implemented under our assumption of limited number of labelled real data (only 1,000 labelled real samples are provided). Since most of state-of-the-art methods only provide their experimental results on a fully labelled dataset, we implement the second main experiment using the whole dataset of fully labelled

Algorithm 2 Training algorithm

```
Require: $\lambda_0, \lambda_1, \lambda_2, \lambda_3$ and learning rate $\gamma$.
Require: $D_s = \{(x_i^s, y_i^s)\}_{i=1}^{n_1}$
Require: $D_{r-old} = \{(x_i^r, y_i^r)\}_{i=1}^{n_2} \cup \{(x_j^r)\}_{j=n_2+1}^{n_3}$
Require: $D_{t-new} = \{(x_i^t)\}_{i=1}^{n_4}$
Require: Pre-trained $\theta_{f_1}, \theta_{f_2}$.
  for number of epochs do
    for number of batches do
      Sample $\{(x_i^s, y_i^s)\}_{i=1}^{b}$
      Sample $\{(x_i^r, y_i^r)\}_{i=1}^{b}$
      Sample $\{(x_j^r\}_{j=1}^{b}$
      Sample $\{(x_j^t\}_{j=1}^{b}$
      Update weighting net $\theta_g$:
```

$$
\begin{aligned}
\theta_{g,t+1} = \theta_{g,t} - \gamma \bigtriangledown \frac{1}{b}\sum_{i=1}^{b}\left[1 - \frac{u_1(z_i^s)}{u_2(z_i^s)}\right] + \frac{1}{b}\sum_{j=1}^{b}\left[\frac{1}{2}\left(\frac{u_1(z_j^r)}{u_2(z_j^r)}\right)^2 - \frac{1}{2}\right] \\
+\frac{1}{b}\sum_{i=1}^{b}\left[1 - \frac{u_1(z_i^s)}{u_3(z_i^s)}\right] + \frac{1}{b}\sum_{j=1}^{b}\left[\frac{1}{2}\left(\frac{u_1(z_j^t)}{u_3(z_j^t)}\right)^2 - \frac{1}{2}\right],
\end{aligned}
\tag{3.10}
$$

```
      Update hand pose estimation net $\theta_{f_1}, \theta_{f_2}, \theta_h$ and $\theta_r$:
```

$$
\begin{aligned}
(\theta_{f_1}, \theta_{f_2}, \theta_h, \theta_r)_{t+1} = (\theta_{f_1}, \theta_{f_2}, \theta_h, \theta_r)_t - \\
\gamma \bigtriangledown \lambda_0 (\frac{1}{b}\sum_{i=1}^{b}\hat{W}_1(z_i^s)\|\hat{y}(z_i^s) - y_i^s\|_2 + \frac{1}{b}\sum_{i=1}^{b}\hat{W}_2(z_i^r)\|\hat{y}(z_i^r) - y_i^r\|_2) - \\
\gamma \bigtriangledown \lambda_1 \sum_{i=1}^{b}\|h(f_1(x_i^r)) - f_1(x_i^s)\|_2 - \\
\gamma \bigtriangledown \lambda_2 \frac{1}{b}\sum_{j=1}^{b}\left[\frac{1}{2}\left(1 - \frac{u_1(z_j^r)}{u_2(z_j^r)}\right)^2\right] + \frac{1}{b}\sum_{j=1}^{b}\left[\frac{1}{2}\left(1 - \frac{u_1(z_j^t)}{u_3(z_j^t)}\right)^2\right] - \\
\gamma \bigtriangledown \lambda_3 \frac{1}{b}\sum_{i=1}^{b}\|r(x_i^a) - x_i^b\|
\end{aligned}
\tag{3.11}
$$

```
    end for
  end for
  return $\theta_{f_1}, \theta_{f_2}$.
```

real samples to make a fair comparison with other SOTA methods. In the first ablation studies, we evaluate the proposed method under different settings (using only synthetic data, using only real data, using both synthetic and real data without weighting, and using both types of data with weighting). The aims of this ablation study are two-fold: firstly, to investigate whether we should use synthetic data; secondly, to evaluate the effectiveness of our weighting strategy. In the second ablation study, we investigate the effect of the number of labelled real samples used for training.

3.4.1 Datasets and evaluation metrics

The setting of our problem requires not only real depth images but also depth synthetic images and a few pairs of real and synthetic images. Most of the benchmark datasets only provide real depth images. The NYU data, which contains paired real and synthetic images, is the only public benchmark dataset that can implement our experiments so that we have carried out the evaluation on the NYU hand pose dataset. It is a very challenging public dataset containing hands from multiple subjects with a large range of 3D poses. The training set contains 72,757 frames of real samples and corresponding synthetic samples. The test set contains 8,252 samples. For each training sample, 3 real depth images captured from 3 different viewpoints and 3 rendered synthetic depth images are provided. During training, we mix real depth images and synthetic images in the same batch.

In the first main experiment, we use only 1,000 labelled real images (500 real images are paired with corresponding synthetic data and the rest of 500 real samples are unpaired) and 72,757 labelled synthetic samples with labels for training. We use 71,757 frames of unlabelled real data and 1,000 frames of labelled real data to estimate density ratios ($n_1 = 72,757$, $n_2 = 1,000$ and $n_3 = 72,757$).

Most of SOTA methods were implemented by using a whole dataset of fully labelled real samples and, obviously, the fully labelled real dataset contains much more information than the partially labelled dataset. In order to make a fair comparison, we also implement the proposed method with the fully labelled real dataset: in the second main experiment, we use all 72,757 labelled real samples and 72,757 labelled synthetic samples ($n_1 = 72,757$, $n_2 = 72,757$ and $n_3 = 72,757$). Although this setting is not the purpose of the proposed method, we still achieve competitive results compared to other SOTA methods.

We follow the standard convention [1] of hand pose estimation on this dataset: locations of 14 joints are evaluated for each hand. We use the two most widely used evaluation metrics:

- The average Euclidean distance between the ground truth of 3D locations and the prediction.
- The fraction of frames that all joints in the frame are within a threshold of the Euclidean distance to the ground truth.

3.4.2 Hyperparameter settings

Since our method contains several different losses and each loss has a quite different magnitude and thus a weight as hyperparameter, we use the validation set as usual to tune these weights. For the loss weights of hand pose estimation net, we set $\lambda_0 = 1.6$, $\lambda_1 = 0.2$, $\lambda_2 = 0.0001$ and $\lambda_3 = 0.0001$. The training process of the proposed method contains two steps. First, we use the synthetic data to pre-train the $f_1(x)$ and $f_2(x)$ for 100 epochs. Then we let the whole hand pose estimation net $\{f_1(z), f_2(z), h(z), r(z)\}$ adversarially train with the weighting net $g(z)$ for 75 epochs. The learning rate $\gamma = 0.000016$.

Since the weighting net in our method needs to estimate the density of the target domain, we split the test set into an unlabelled validation set and a test set to keep the real test set untouched during training. We randomly select samples into the validation set and the test set. We use from 20% to 50% samples for the validation set and from 80% to 50% samples for the test set. The unlabelled validation set is sent to the weighting net to estimate density ratios during training. The remaining test set is kept untouched and is only used in the test stage. Since the random selection of samples and the split of the dataset affect the quantitative results, we report the interval of our experiment results.

3.4.3 Comparison to state-of-the-art methods with only a partially labelled real training set

In this first main experiment, we implement our experiment on a common situation in applications: only a few labelled real samples are available. In order to compare with the SOTA methods that report their performance under the same setting, we use only 1,000 labelled real samples.

We compare the mean joint error with other SOTA methods in Table 3.1. However, only a few SOTA methods report their performances on a training set of only a few labelled real samples, therefore, we not only compare the performances reported in their papers but also compare with those from the 3rd party implementations. The **upper block** of Table 3.1 contains three methods (DeepPrior [97], DeepPrior++ [95] and Crossing Nets [62]) implemented by the LSPRPreView [79]. The **lower block** shows results of LPSRPreView, Crossing Nets and LSPS reported in their papers. From Table 3.1, we can see that the proposed method achieves state-of-the-art performance on a small set of labelled real samples.

Table 3.1: Comparison of different hand pose estimation methods on the NYU hand dataset [1] with only 1,000 labelled real samples. We compare the mean joint error with other state-of-the-art methods.

Method	Mean Error (mm)	# labelled real samples
DeepPrior (PreView) [97]	36.99	1,000
Crossing Nets (PreView) [62]	36.35	1,000
DeepPrior++ (PreView) [95]	31.01	1,000
LPSRPreView [79]	22.84	1,000
Crossing Nets [62]	17.70	1,455
LSPS [85]	16.60	1,000
Ours	**10.85**	1,000

3.4.4 Comparison to state-of-the-art methods with a fully labelled real training set

In this second main experiment, we compare our method with other state-of-the-art methods on the NYU hand data in terms of the mean joint error, and we not only compare with 2D input based methods, but also compare with the methods that use

3D input:

- 3D input based methods: 3DCNN [98], SHPR-Net [99], Hand PointNet [100], Point to Point [101] and V2V [102].
- 2D input based methods: Crossing Nets [62], LSPS [85] Lie-X [103], DeepPrior++ [95], Pose-REN [104], SDHI-GAN[105], ASST[106], DenseReg [107] and CrossInfoNet [108], and Feature Mapping [82] (Feature Mapping uses 5,000,000 extra synthetic training samples).

Table 3.2: Comparison of different hand pose estimation methods on the NYU hand dataset [1]. We compare the mean joint error with other state-of-the-art methods. The **upper block** is for 2D input based methods. The **lower block** is for 3D input based methods. GAN: generative adversarial nets; DA: domain adaptation.

Method	Mean Error (mm)	Input	Note
Crossing Nets [62]	15.5	2D	GAN
LSPS [85]	15.4	2D	
Lie-X [103]	14.5	2D	
ASST [106]	14.1	2D	GAN & DA
DeepPrior++ [95]	12.3	2D	
Pose-REN [104]	11.8	2D	
SDHI-GAN [105]	11.4	2D	GAN & DA
DenseReg [107]	10.2	2D	
CrossInfoNet [108]	10.08	2D	
Ours	$\mathbf{9.36 - 10.11}$	2D	
Feature mapping [82]	7.4	2D (5M extra samples)	DA
3DCNN [98]	14.1	3D	
SHPR-Net [99]	10.8	3D	
Hand PointNet [100]	10.5	3D	
Point to Point [101]	9.1	3D	
V2V [102]	8.42	3D	

The comparison results are summarised in Table 3.2, from which we can make two observations. Firstly, the proposed method achieved the state-of-the-art performance among the 2D input based methods. We note that, although the performance of the method "Feature Mapping" is higher than us, comparing Feature Mapping with other 2D input based methods is not very fair because Feature Mapping uses extra $5,000,000$ synthetic samples compared with other 2D based methods. Secondly, it is known that in general 3D input based methods have better performance then 2D input based methods, however, the performance of our method is even better than many 3D input based methods. Moreover, our work can be extended to the 3D input case, which may further improve the performance.

In Figure 3.4, Figure 3.5 and Figure 3.6, we also provide some visualised comparisons with the other 2D input based methods that do not use extra training samples, as well as some qualitative results from our method.

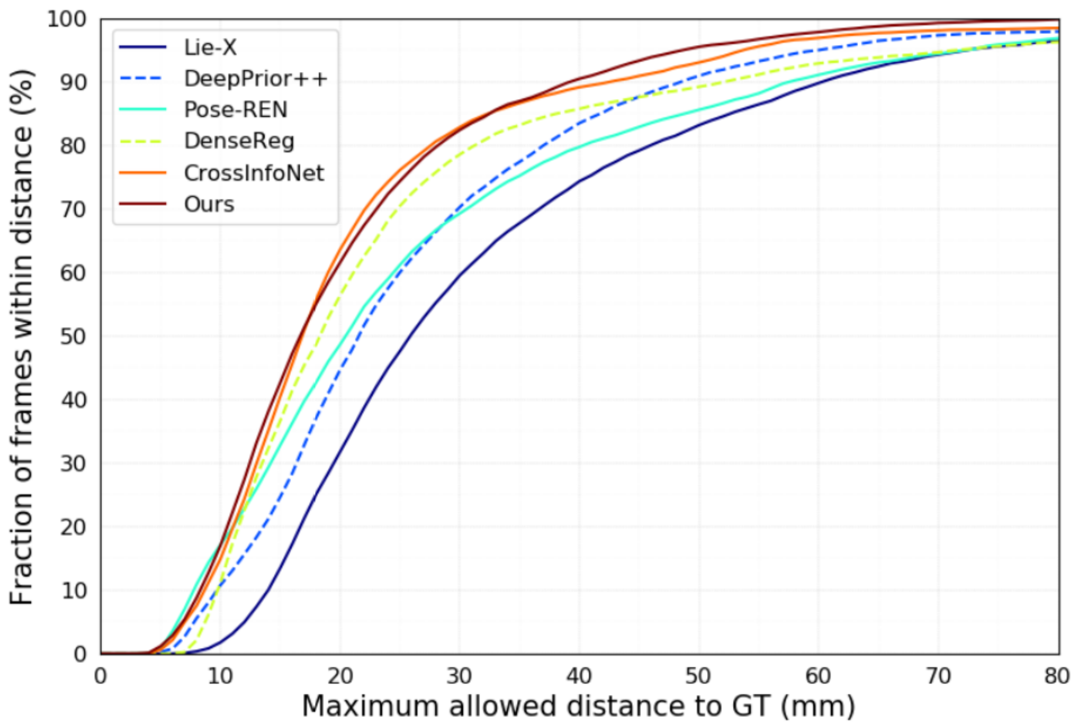


Figure 3.4: Comparison with other state-of-the-art 2D input based methods on the NYU dataset. Our method has the maximum area under the curve, in terms of the fraction of frames where all joints of a frame are within a maximum 3D distance from the ground truth.

3.4.5 Ablation studies

3.4.5.1 Effects of using two domains and the weighting strategy

In the first ablation study, we investigate two aspects of the proposed method: the effect of only using the data from either the real domain or the synthetic domain, and the effect of the weighting strategy. We compare the performances of four variant models of our method in the following four settings of training:

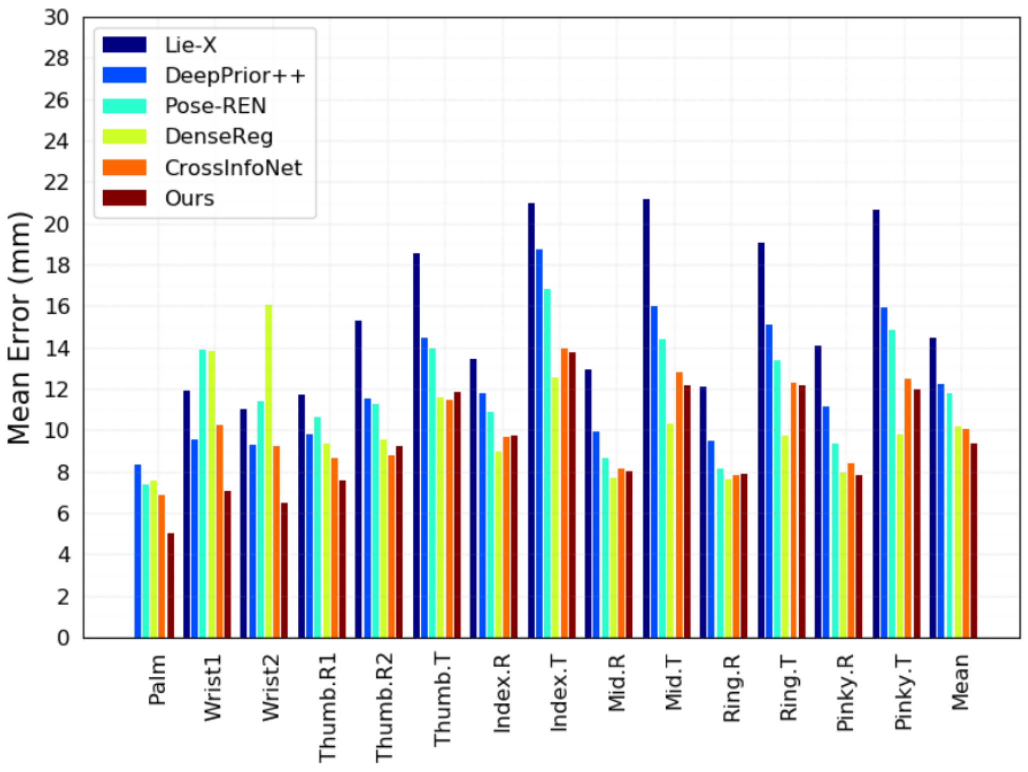


Figure 3.5: Comparison with other state-of-the-art 2D input based methods on the NYU dataset, in terms of average joint error. The results are consistent with those in Figure 3.4: our method has the minimum mean error here, and we have the maximum area under the curve in Figure 3.4. Compared with the fraction of frames within a maximum 3D distance, the mean error is more tolerant to a single large error of a joint.

1. training using only real data;
2. training using only synthetic data;
3. training using both synthetic data and real data, without the adversarial training with the weighting net;
4. training using both synthetic data and real data, with the adversarial training with the weighting net.

Since our method focuses on the situation that training set has a limit number of labelled real data, in the experimental settings 1, 3 and 4, we use only 1,000 labelled real samples. More specific, in the first setting, we only use 1,000 labelled real samples to train our model; in the second setting, we only use 72,757 of labelled

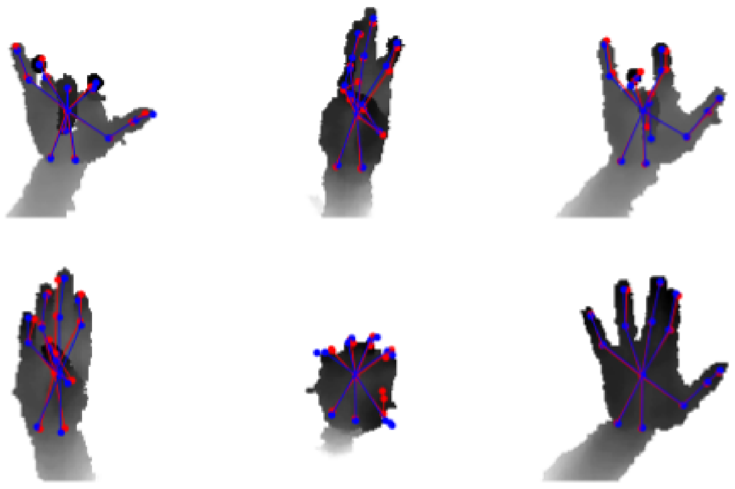


Figure 3.6: Qualitative results from our method. The joint locations are shown on the depth images. The prediction of joints is shown in red and the corresponding ground truth is shown in blue.

synthetic samples to train; in the third and fourth settings, we use both 1,000 labelled real samples and 72,757 labelled synthetic samples to train. The only difference between setting 3 and setting 4 is whether to use the weighting strategy.

Except for the differences above-mentioned, we use similar settings for all the compared models: the weights of different losses, batch size, parameters of the optimiser and the number of training epochs are fixed the same, and the only difference is the learning rate. Since the weight enlarges the variance of gradients, the optimal learning rate of adversarial training with the weighting net is smaller than that for the non-weighting one: the learning rates in the weighting and non-weighting cases are $\gamma_w = 0.000016$ and $\gamma_{non-w} = 0.0005$, respectively. The results reported in Table 3.3 show two patterns. Firstly, our method greatly improves the estimation accuracy, compared with the one only using the labelled real data and the one only using the labelled synthetic data. From the experimental results, we can see that, under the framework of our proposed method, using both real samples and synthetic samples is better than using only real or synthetic samples. Secondly, the weighting strategy reduces the mean error from 12.37mm to 10.85mm; that is, our adversarial weighting strategy improves 1.52mm (14%) in performance.

Table 3.3: Ablation study on the effect of the adversarial training with the weighting net on the performance.

Method	Mean Error (mm)	# labelled real samples
1. Real only	39.22	1,000
2. Synthetic only	18.84	0
3. Non-weighting	12.37	1,000
4. Weighting	**10.85**	1,000

3.4.5.2 Effect of the number of samples

In the second ablation study, we investigate the effect of the number of labelled real samples used for training. The experimental setting is the same as before, with the only difference in the number of labelled real samples used. We use 0, 1,000, 10,000, and all (72,757) labelled real samples, respectively, to train the model. The results are shown in Table 3.4. The average mean errors in the testing set are 18.84mm, 10.85mm, 10.38mm, and 9.36mm, respectively. That is, the more real labelled samples used, the lower the mean error.

Table 3.4: The effect of the number of labelled real training samples.

# labelled real samples	0	1,000	10,000	all (72,757)
Mean Error (mm)	18.84	10.85	10.38	9.36

3.5 Conclusion and future work

In this paper, we proposed an improved semi-supervised hand pose estimation algorithm based on implicit importance weight estimation method to correct the covariate shift problem in hand pose domain adaptation. We showed that, with partially labelled real data and fully labelled synthetic data, using the data from both domains leads to better results than either only using the labelled real data or only using the labelled synthetic data. The proposed weighting method can also improve the performance of the pose estimation model, and our method produces the state-of-the-art performance (both on partially labelled real dataset and fully labelled real dataset) among 2D input based methods on the most widely used NYU benchmark dataset.

As there exist some 3D based methods that have better performance than ours, one of our future work is to extend the method proposed in this paper to 3D input for higher performance.

Chapter 4

Pseudo-label robust self-training for semi-supervised few-shot classification

This chapter aims to address a special semi-supervised learning problem: the few-shot semi-supervised classification problem, where unlabelled data are available to complement a few shots of labelled data per class. Since in the few-shot learning labelled samples are too few to train a good learner for the current task, transferring knowledge from many related domains or tasks is a reasonable way to improve the model performance. This process of learning transferable knowledge is called meta-training, which extracts shared knowledge, such as features and a metric, from many tasks and transfers to each few-shot learning task. Besides the meta-learning, another potential way to improve a semi-supervised few-shot learning algorithm is to utilise the softmax stochastic supervision of a classifier to provide pseudo labels for unlabelled training samples to retrain the model. This learning method is called self-training in machine learning. Self-training is an effective way for semi-supervised learning since self-training expands the labelled training samples (although the supervision is not perfect). However, in few-shot classification, a single model based self-training is difficult to overcome the confirmation bias, due to the noise accumulation in pseudo labelling. Therefore, in this chapter, we propose an ensemble self-training method specifically designed for semi-supervised few-shot classification tasks, in order to relief the noise accumulation in pseudo-labelling and the confirmation bias of self-training. The experimental results show that the proposed method enjoys state-of-the-art performances on widely used benchmark datasets of few-shot classification.

4.1 Introduction

Deep learning methods have achieved great successes recently, however, these methods are data-hungry, since deep learning generally requires a great amount of labelled data as training samples to ensure a good performance. In many applications, it is too expensive or even impossible to label a dataset of a sufficient size. The scarcity of labelled data limits a potentially much wider use of deep learning in practice. Developing learning algorithms that can learn from a limited amount of data is hence a big yet important challenge to the machine learning community. Due to these reasons, few-shot learning has attracted crucial attention in recent years.

Few-shot learning aims to learn a classification or regression model from one or a few labelled samples per class. In this paper, we focus on the few-shot classification problem. Few-shot learning is a relatively new area of machine learning and most of the current few-shot classification methods follow the form of meta-learning [109, 110], which mainly contains meta-training and meta-testing two steps to fit a classification model. The meta-learning aims to learn some prior knowledge (such as a divergence metric [7], initialization of parameters [111], and parameters of the feature extractor) from many few-shot learning tasks. These prior knowledge can help the learner fast learn in some new tasks. In the meta-training step, the model updates these prior knowledge from many few-shot learning tasks. Each task also contains a training step and a test step. In the training step, the model optimises the parameters of the current task. In the test step, the model updates the parameters that relevant to these prior knowledge (meta knowledge). In the meta-testing step, the classification model learns to classify unseen classes based on one or a few labelled samples in each class; the model is retrained and tested on some new task which has no overlapping with the meta-training tasks. Many meta-learning based fully supervised few-shot learning methods have made significant progress in this area, however, these methods have a crucial drawback: they only use one or few *labelled* samples in the meta-testing step. Consequently, the classifier may overfit the labelled samples in the meta-testing. Since acquiring unlabelled data is much cheaper and easier than labelling data, exploring the information contained in unlabelled data to improve few-shot learning algorithms has attracted great attention. Recently, [112] proposed a self-training based semi-supervised few-shot classification algorithm and achieved state-of-the-art performance. However, this method has a big issue: it only uses a single neural network based classifier for self-training. A single classifier cannot address the confirmation bias (also called the label noise accumulation problem in pseudo labelling) by itself; and more concerned in few-shot learning, since deep networks are too complicated for the limited amount of data, using a single deep network based classifier may result in a very big variance in few-shot classification tasks. An effective way to reduce model variance is to assemble different predictions, however, many traditional ensemble learning algorithms cannot apply to one-shot or few-shot learning due to the following two problems. Firstly, it is hard to split the sample set since we only have one or a few samples per class. Secondly, it is difficult to implement many different network architectures to construct diverse prediction models for ensemble learning, since comparing to the number of training samples, most of the architectures contain too many parameters for few-shot classification tasks.

In order to address the above two problems and develop a label-noise (noise in pseudo labels) robust self-training algorithm, we propose a semi-supervised ensemble few-shot learning algorithm that is specially tailored for few-shot classification tasks. The proposed method learns ensembles to relieve the confirmation bias of a single classifier in the self-training process since different classifiers from different views can provide different information to redress the confirmation bias. Moreover, unlike many traditional ensemble learning algorithms, which cannot be applied to one-shot or few-shot learning, our ensemble strategy is designed for few-shot learning, aiming to reduce the variance of deep networks in few-shot classification problems. Meanwhile, our method is designed to deploy shared feature embedding in different layers of a ResNet; this design enables a high-performance model with a negligible increase in the number of parameters.

The contributions of this work can be summarised as follows. Firstly, we propose a meta-learning based semi-supervised ensemble method for few-shot classification. The proposed ensemble method leverages information from different hierarchical levels and reduces the variance of the obtained neural network based classification model, with only a slightly increased number of parameters. Secondly, we provide a model that can relieve the confirmation bias of a single classifier in the self-training for few-shot classification. Thirdly, our method achieves state-of-the-art performances in two widely used benchmark datasets of few-shot image classification.

4.2 Related work

We summarise the most relevant researches as below, rather than present a comprehensive survey on all related work as they are immense.

4.2.1 Supervised few-shot learning

Supervised few-shot learning for classification can be roughly grouped into two categories: meta-learning based methods and metric based methods.

Meta-learning based methods. This line of methods either learn a good initialization of model parameters or learn an optimizer. Their goal is a few-shot learner that can adapt novel samples within a small number of optimization steps. For example, [8] proposed an algorithm called model-agnostic meta-learning (MAML), which can find a single set of model parameters that can be adapted to individual tasks within a few steps of gradient descent. Building on MAML, [113] proposed a probabilistic and generalized version of MAML. Unlike these two methods which use the stochastic gradient descent, an LSTM-based meta-learner optimizer was proposed in [111] to train another learner in the few-shot regime. In addition, there is one kind of implicit meta-learning methods for few-shot learning, in particular, they integrate the meta learner and base learner into one RNN based model [114, 115]. For example, [114] proposed a memory-augmented model called memory-augmented recurrent network, which can obtain useful representations of base data via gradient descent and can predict unseen samples on novel data.

Metric based methods. This category mainly learns to compare the similarity or distance of two samples. Some work aims to learn feature embedding with a fixed metric [116, 117]. For example, [116] used a Siamese convolutional neural network for one-shot leaning, in which convolution layers are adopted in a VGG-styled structure and L1 component-wise distance is used to compute the distance between image features. Some work focus on learning a metric, such as the relation network [7, 118], by using a learnable module, e.g. a neural network, to model the similarity or distance metric between two samples. In addition to learning feature embedding and learning a metric, some work focuses on learning appropriate class prototypes, such as the prototype network [6] and its variants [119]. The prototype network introduced the prototype concept into the few-shot classification and minimizing the Euclidean distance between every sample and its class prototypes. Built on the prototype network, [119] proposed infinite mixture prototypes method to accommodate multiple class prototypes and outperformed the prototype network on the alphabet recognition task.

4.2.2 Semi-supervised few-shot learning

The supervised few-shot learning methods discussed above did not consider exploiting information from *unlabelled* samples. Therefore, built on the prototypical network, [5] proposed a first semi-supervised few-shot classification method, in which unlabelled samples are used to learn a more accurate class prototype. [112] adopted self-training method to use unlabelled data, and obtained state-of-the-art performance on the *mini*-ImageNet and *tiered*-ImageNet datasets. More specifically, it first trained a classifier on support data to predict unlabelled samples and added those samples with high-confidence prediction into original support data to retrain the few-shot classifier. The process is iterated and thus the design of few-shot classifier is crucial to the final performance. In contrast to [112], our method adopts an ensemble strategy to perform self-training, improve the accuracy and robustness of pseudo labelling for unlabelled samples, and reduce the variance of few-shot classifier.

4.2.3 Ensemble few-shot learning

One big challenge of few-shot learning is overfitting. Ensemble learning [120] is an effective approach to mitigate overfitting and reducing the variance of the model, the key of which is to obtain both accurate and diverse ensemble members [121]. In ensemble learning, there are some classical methods, such as bagging [122] and AdaBoost [123]; however, they are not suitable for few-shot learning. Bagging [124] can ensure diversity by bootstrap sampling but cannot ensure the accuracies of base classifiers in few-shot learning, since base neural networks have a big variance on few samples. AdaBoost obtains the diversity by re-weighting training data based on the training errors generated from the previous classifier; however, in few-shot learning, the first base classier, implemented by a complicated neural network, may have no training error on few training samples, resulting in the inapplicability of AdaBoost on few labelled samples. There are some new ensemble methods of neural network, such as snapshot ensembling [125] and temporal ensembling [25]; however, they are specially tailored for large scale of labelled samples, not for few labelled samples. Recently, [126] introduced a new ensemble mechanism for few-shot classification, which encourages large probabilities of ground-truth classes in two base classifiers and pushes dimension-wise orthogonality between probabilities of non-ground-truth classes in two base classifiers; however, their model needs to deposit a mass of model parameters and demands large amounts of memory, and moreover, it has not considered using unlabelled data. In contrast to [126], our method is semi-supervised and more suitable for few-shot classification.

4.3 Methodology

In this section, we provide details of the proposed method. We first give the definition and formulation of meta-learning based few-shot learning, then describe in detail the proposed ensemble strategy and self-training process of our model, and finally provide our learning algorithms.

4.3.1 C-way K-shot learning

The objective of few-shot (C-way K-shot) learning is: given C unseen new classes, let the classification model $\hat{y} = f(x)$ fast learn to classify unseen new classes with K shots of labelled samples of each class. A conventional supervised learning method, that directly trains the classifier by only using these $K \times C$ labelled samples, is highly likely to produce a very poor result. Hence, most of the few-shot learning algorithms adopt meta-training, which learns prior knowledge from many related tasks and then fast adapts to new tasks. The proposed method also follows the form of meta-learning, which mainly contains two steps: the meta-training step and the meta-testing step.

In the meta-training step, the proposed method learns a feature extractor and a relation network from many classification tasks. In the meta-testing step, the proposed method retrains a new classifier to fast adapt to the new task: learn to classify C new classes. We adopt the episode training proposed by [117]. In meta-learning, each episode is a classification task. Both the meta-training and meta-testing steps contain many episodes. Each episode contains a labelled support set S, an unlabelled support set U, and a query set Q. The whole dataset contains three exclusive parts: meta-training set, meta-test set and meta-validation set. Suppose the meta-training set contains T_1 classes, the meta-test set contains T_2 classes and the meta-validation set has T_3 classes; these three sets have no overlapping, and thus the whole dataset contains $T_1 + T_2 + T_3$ classes.

4.3.2 Construction of base classifier and the ensemble strategy

Since in few-shot learning it is hard to divide the small training set to fit different base classifiers, how to construct base classifiers becomes a crucial problem. Meanwhile, ensemble-based methods often lead to a large amount of parameters and extra computational cost. Therefore, a key challenge addressed, and the main novelty presented, in this work is how to design an ensemble method especially for semi-supervised few-shot learning with a limited amount of parameters and computational cost.

In the proposed method, each base classifier contains two modules: a feature

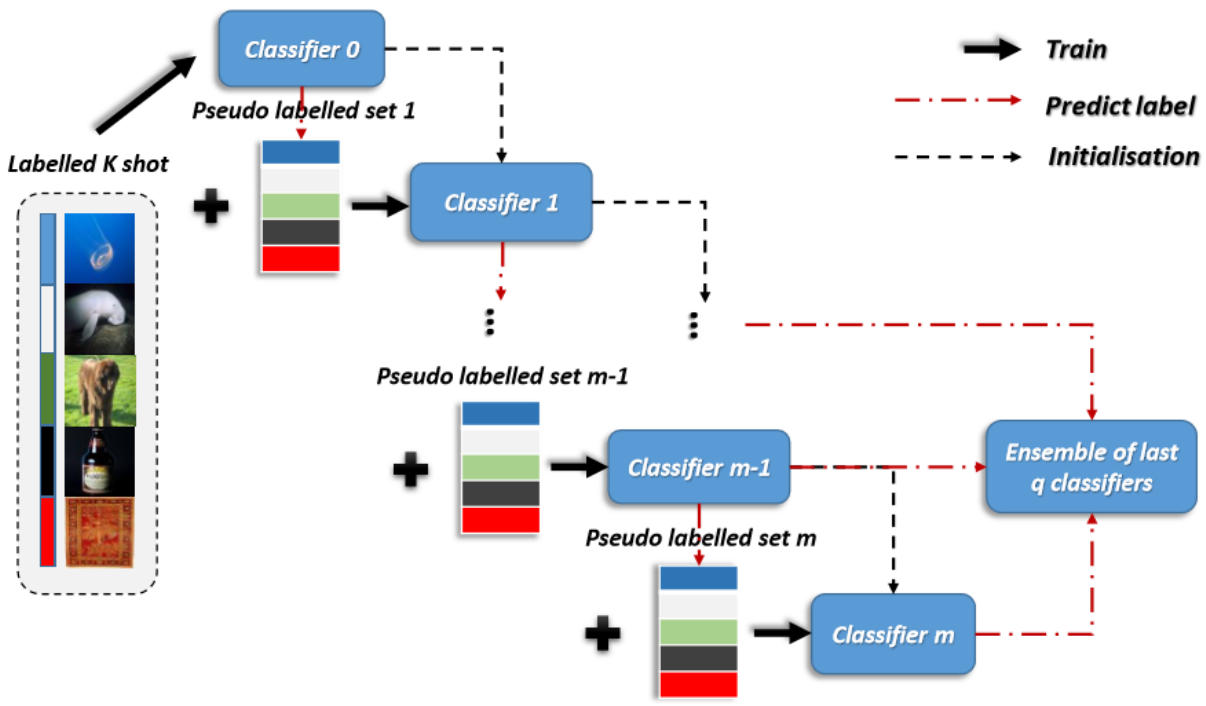


Figure 4.1: The self-training procedure and the construction of base classifiers of the proposed method: The classifier 0 is fitted by C-way K-shot labelled samples and provides pseudo labels (denoted by pseudo-labelled set 1) of the unlabelled subset. The classifier 1 is then fitted by the labelled support set and the pseudo labelled support set 1. Repeating the self-training process, we finally obtain m base classifiers. We use the last q classifier to construct the ensemble and make the prediction.

extractor and a classifier. The feature extractor learns to extract discriminative features from data and the classifier learns to predict labels based on these features. In order to reduce the number of parameters, in our method the feature extraction network $z_i = g_\phi(x_i)$ is shared by p selected base classifiers $\hat{y}_j = f_{\theta j}(z)$, for $j = 1, 2, \ldots, p$.

To build different base classifiers, classical ensemble learning often use different labelled training sets, which is unfortunately infeasible in few-shot learning due to the lack of labelled training data. Therefore, in the proposed method, we use different pseudo labelled training sets to get different base classifiers, under the following ensemble strategy, as illustrated in Figure 4.1. We first use the labelled training set $S_0 = \{(x_i, y_i)\}_{i=1}^{n_1}$, $n_1 = C \times K$, to fit an initial classifier f_{θ_0} by minimising the loss function $\mathscr{L}_0$:

$$\theta_0 = \arg\min_{\theta_0} \sum_{i=1}^{n_1} \mathscr{L}_0(f_{\theta_0}(g_\phi(x_i)), y_i),$$

where $\mathscr{L}_0(f_{\theta_0}(g_\phi(x_i), y_i) = -\sum_{c=1}^{C} y_{ic} \log f_{\theta_0}(g_\phi(x_i))_c$ is the cross entropy, in which $f_{\theta_0}(g_\phi(x_i))_c$ and y_{ic} are the cth elements of $f_{\theta_0}(g_\phi(x_i))$ and y_i, respectively. Then we use f_{θ_0} to predict pseudo labels for the unlabelled training samples: $\hat{y}_j = f_{\theta_0}(g_\phi(x_j))$ with $x_j \in U$. Since pseudo labelled data have different level of confidence, it is reasonable to select and weighting pseudo labelled training samples based on their relation score. The relation score is a similarity measure between the feature vector of the input and the average corresponding class representation in a feature space. Inspired by prior work [7, 112], we use a relation score to select and compute confidence weights for the pseudo labelled data. We select n_4 samples from the unlabelled support set U that have the highest relation scores s_{jc}, $0 < s_{jc} < 1$ to construct a new pseudo labelled support set $P_1 = \{(x_1, \hat{y}_1), \ldots, (x_{n_4}, \hat{y}_{n_4})\}$. Similarly to the relation network [7, 112] that learn a deep distance metric, the relation score is learned by a relation network $s_{jc} = f_{s\psi}(\mathbb{C}(g_\phi(x_i), g_\phi(x_j)_c)_c)$, where $g_\phi(x_j)_c$ is the class central of class c (mean of all real labelled samples $g_\phi(x_i)$ of class c) and $\mathbb{C}(.)$ is the concatenation operation. Then we construct the second training set $S_1 = \{S_0 \cup P_1\}$ and use S_1 to train the second base classifier f_{θ_1} by minimising the semi-supervised loss function proposed by[112] as follows:

$$\theta_1 = \arg\min_{\theta_1} [\sum_{i=1}^{n_1} L_{\theta,\phi,\psi}(x_i, y_i) + \sum_{j=1}^{n_4} L_{\theta,\phi,\psi}(x_j, y_j)].$$

$$L_{\theta,\phi,\psi}(x,y) = \begin{cases} -\sum_{c=1}^{C} y_{ic} \log f_{\theta_1}(g_\phi(x_i))_c, & \text{if} x \in S \\ -\sum_{c=1}^{C} \hat{y}_{jc} \log f_{s\psi}(\mathbb{C}(g_\phi(x_i), g_\phi(x_j)_c)_c f_{\theta_1}(g_\phi(x_j))_c, & \text{if} x \in P \end{cases} \tag{4.1}$$

The loss term of the labelled samples is still the cross entropy while the loss term of the unlabelled samples is a weighted cross entropy that logarithm weighted by the relation score.

We repeat these steps to obtain m base classifier $\{f_{\theta_1}, f_{\theta_2}, \ldots, f_{\theta_m}\}$, and finally we select the last q $(q \leq m)$ base classifiers to predict the label in the test set. For simplicity, the output of ensemble is taken as the average of outputs of q classifiers:

$$\hat{y}_i = \frac{1}{q}\sum_{k=0}^{q-1} f_{\theta_{m-k}}(g_\phi(x_i)).$$

4.3.3 Episodic training and testing

Suppose there are M episodes of training samples in the meta-training step. In each episode, we sample a classification task (by sampling C classes from T_1 classes) from the meta-training set and sample K labelled samples from each of the C classes. These $K \times C$ samples are the support set $S = \{(x_i, y_i)\}_{i=1}^{n_1}$, $n_1 = C \times K$. The query set $Q = \{(x_i, y_i)\}_{i=1}^{n_2}$ includes the rest labelled samples of the C classes. The unlabelled samples of the C classes are the unlabelled support set $U = \{x_j\}_{j=1}^{n_3}$, with $n_3 \gg n_1$ and $n_3 > n_4$ as usual in practice.

Each training episode consists of a task update step and a meta update step. In the task update step, parameters of the classifier is initialised by f_{θ_*}(the meta initialisation of classifiers), then we use the support set S and unlabelled support set U to update parameters of classifiers $f_{\theta_i}(z)$, $i = 1, 2, \ldots, p$. The parameters of classifier f_{θ_k} is learned by using the following loss function:

$$\theta_k = \arg\min_{\theta_k}[\sum_{i=1}^{n_1} L_{\theta,\phi,\psi}(x_i, y_i) + \sum_{j=1}^{n_4} L_{\theta,\phi,\psi}(x_j, y_j)].$$

In the meta update steps, we use the query set to compute the classification accuracy and update parameters of the feature extractor $g_\phi(x)$, the initialisation of classifier $f_{\theta_*}(g_\phi(x))$ and the relation network $f_{s\psi}(\mathbb{C}(g_\phi(x_i), g_\phi(x_j)_c)$ by the following formulae: (Different with the approximation in MAML[8], we do not cut down the higher order derivative and use the first order approximation.)

$$g_\phi \leftarrow g_\phi - \alpha_1 \nabla_\phi \sum_{i=1}^{n_2} L_{\theta,\phi,\psi}(x_i, y_i),$$

$$f_{\theta_*} \leftarrow f_{\theta_*} - \alpha_1 \nabla_\phi \sum_{i=1}^{n_2} L_{\theta,\phi,\psi}(x_i, y_i),$$

$$f_{s\psi} \leftarrow f_{s\psi} - \alpha_2 \nabla_\psi \sum_{j=1}^{n_4} L_{\theta,\phi,\psi}(x_j, y_j).$$

where α_1 and α_2 are learning rate.

The episodes in the meta-testing step are different from the episodes in meta-training, since in each episode now it only has the task update step. In every episode of meta-testing, in the task update step we just use the support set S and unlabelled support set U to retrain all base classifiers $\{f_{\theta_1}(z), f_{\theta_2}(z), f_{\theta_3}(z), \ldots, f_{\theta_p}(z)\}$, with other parts of our model fixed. The query (test) set Q is only used to evaluate the performances of the model.

The architecture of the proposed method is illustrated in Figure 4.1, the meta-training procedure is summarised in Algorithm 3 and the meta-testing procedure in Algorithm 4.

4.4 Experiments

In this section, we provide several experiments to evaluate our proposed method. We compare the proposed method with other state-of-the-art few-shot classification methods on two widely used benchmark datasets (*mini*-ImageNet and *tiered*-ImageNet). We also conduct ablation studies to evaluate the effectiveness of our self-training and ensemble strategies.

4.4.1 Datasets

***mini*-ImageNet** *mini*-ImageNet is an image classification dataset first provided by [117]. The *mini*-ImageNet dataset is one of the most widely used benchmark datasets for few-shot learning. It contains 100 classes of images and each class has 600 samples. We follow the conventional setting of the few-shot classification: the dataset is split to 64 classes, 16 classes, and 20 classes for the meta-training set, the meta-validation set, and meta-test set. The meta-training set, meta-validation set, and the meta-test set have no overlapping of classes.

Algorithm 3 Meta-training procedure of the proposed method

```
Require: Number of episodes $M$
  for $i = 1 : M$ do
    Task update step:
    Initialise the classifier $f_{\theta_0} = f_{\theta_*}$
    Sample the labelled support set $S = \{(x_i, y_i)\}_{i=1}^{n_1}$.
    Sample the unlabelled support set $U = \{x_i^t\}_{i=1}^{n_4}$ .
    Update classifiers $f_{\theta_k}$:
        $$f_{\theta_k} \leftarrow \arg\min_{\theta_k}[\sum_{i=1}^{n_1} L_{\theta,\phi,\psi}(x_i, y_i) + \sum_{j=1}^{n_4} L_{\theta,\phi,\psi}(x_j, y_j)]$$
    Meta update step:
    Sample the query set: $Q = \{(x_i, y_i)\}_{i=1}^{n_2}$.
    Test on the query set and update the feature extractor $g_\phi$, the initialisation of
    classifier $f_{\theta_*}(g_\phi(x))$, and the relation net $f_{s\psi}$:
        $$g_\phi \leftarrow g_\phi - \alpha_1 \nabla_\phi \sum_{i=1}^{n_2} L_{\theta,\phi,\psi}(x_i, y_i),$$
        $$f_{\theta_*} \leftarrow f_{\theta_*} - \alpha_1 \nabla_\phi \sum_{i=1}^{n_2} L_{\theta,\phi,\psi}(x_i, y_i),$$
        $$f_{s\psi} \leftarrow f_{s\psi} - \alpha_2 \nabla_\psi \sum_{j=1}^{n_4} L_{\theta,\phi,\psi}(x_j, y_j).$$
  end for
  return $g_\phi, f_{\theta_*}, f_{s\psi}$.
```

***tiered*-ImageNet** *tiered*-ImageNet is a relatively large dataset for few-shot image classification. It was proposed by [5] and contains 779,165 images with 608 classes. All 608 classes belong to 34 super classes. We also follow the conventional setting of previous work [5, 127]. The meta-training set contains 20 super classes (351 classes), the meta-validation set contains 6 super classes (97 classes), and the meta-test set has 8 super classed (160 classes). Classes in the meta-training, meta-validation, and meta-test sets are disjoint.

Algorithm 4 Meta-testing procedure of the proposed method

```
Require: Number of test tasks $T$
  for $t = 1 : T$ do
    Sample one new classification task:
    Sample the labelled support set $S_0 = \{(x_i, y_i)\}_{i=1}^{n_1}$ of the task.
    Sample the unlabelled support set $U = \{x_i^t\}_{i=1}^{n_3}$.
    Sample the query set $Q = \{(x_i^r, y_i^r)\}_{i=1}^{n_2}$ of the task.
    Load the feature extractor, initialisation of classifier and the relation net $g_\phi$, $f_{\theta_*}$, $f_{s\psi}$.
    Initialising the classifier:
    $f_{\theta_0} = f_{\theta_*}$
    Train the initial supervised classifier:
    $f_{\theta_0} \leftarrow \arg\min_{\theta_0} \sum_{i=1}^{n_1} \mathscr{L}_0(f_{\theta_0}(g_\phi(x_i)), y_i)$.
    for $k = 1 : m$ do
      Using $f_{s\psi}$ select a subset $U_k$ with $n_4$ unlabelled samples from $U$ and with $f_{\theta_{k-1}}$ to construct the pseudo labelled set $P_k = \{(x_1, \hat{y}_1), \dots, (x_{n_4}, \hat{y}_{n_4})\}, \hat{y}_j = f_{\theta_{k-1}}(g_\phi(x_j)), x_j \in U_k$.
      $$f_{\theta_k} \leftarrow \arg\min_{\theta_k} [\sum_{i=1}^{n_1} L_{\theta,\phi,\psi}(x_i, y_i) + \sum_{j=1}^{n_4} L_{\theta,\phi,\psi}(x_j, y_j)]$$
    end for
    Predict labels of samples in the query set:
    for $j = 1 : n_2$ do
      $\hat{y}_j = \frac{1}{q} \sum_{k=0}^{q-1} f_{\theta_{m-k}}(g_\phi(x_j)), x_j \in Q$
    end for
    Use $\hat{y}_j, j = 1, ..., n_2$ to compute classification accuracy $A_t$ of task $t$.
  end for
  return $1/T \sum_{t=1}^{T} A_t$
```

4.4.2 Network architecture and implement details

We use the network architecture proposed by [128] as the backbone of our system. The network contains two ResNet-12: One is the feature extractor and the other outputs the scaling and shifting parameters of the feature extractor. Each ResNet contains four residual blocks and each residual block contains three convolutional layers, and the convolutional layers have 64, 64, 64, 128, 128, 128, 256, 256, 256, 512, 512, and 512 (channels) kernels, respectively, with kernel size 3×3. The relation score net f_s consists of two convolutional layers (64 channels with kernel size 3×3) and two fully connected layers. All base classifiers $f_{\theta_1}(z), f_{\theta_2}(z), \dots, f_{\theta_p}(z)$ are a fully connected layer with 512 hidden units. The image size of the input is

84×84 in all experiments.

In the experiments, we use all the labelled samples in the meta-training set to pre-train (initialise) the model. Then we perform episode training to train the model, with $15,000$ iterations in the meta-training. The initial learning rate is 0.001 and it decays to half for each $1,000$ iterations until the learning rate decreases to 0.0001. In both mini-Imagenet and tiered-Imagenet dataset, for each five-way one-shot learning task, we select $20*5$ pseudo labelled samples from $30*5$ unlabelled samples based on the relation score to train a base classifier (Each base classifier of the current task has $21*5$ labelled and pseudo labelled training sample). For each five-way five-shot task, we select $30*5$ pseudo labelled samples from $50*5$ unlabelled samples to train a base classifier (Each base classifier of the current task has $35*5$ labelled and pseudo labelled training sample). For all 1-shot classification tasks, we assemble four base classifiers $f_{\theta_1}(z), f_{\theta_2}(z), f_{\theta_3}(z), f_{\theta_4}(z)$ to predict the label. For all 5-shot classification tasks, considering the number of unlabelled samples in the support set (since the number of splits of unlabelled training set depends on the number of training samples in the class with the minimum number of unlabelled samples), we only assemble three base classifiers.

4.4.3 Comparisons with the state-of-the-arts

We evaluate our method on two conventional few-shot classification tasks: 5-way 1-shot and 5-way 5-shot. Following the conventional setting of [111, 8, 132, 133, 134], in each episode, the query set Q contains 15 samples for each class (totally 75 samples) for evaluation 1-shot and 5-shot tasks. The overall accuracy is reported based on an average of 600 randomly selected learning tasks. The 95% confidence interval of accuracy is also reported.

We compare our method with other state-of-the-art methods on both *mini*-ImageNet and *tiered*-ImageNet datasets. The results of are reported in Table 4.1 and Table 4.2 for the *mini*-ImageNet and *tiered*-ImageNet datasets, respectively. From Table 4.1 and Table 4.2, we can see the proposed method achieves the best performance in the 1-shot classification tasks. In the 5-shot classification task, the proposed method still performs better than most of the state-of-the-art methods ex-

Table 4.1: Performance comparison of different few-shot learning methods for 5-way 1-shot and 5-way 5-shot classifications on *mini*-ImageNet: mean accuracy and its 95% confidence interval. The upper block shows the performance of supervised few-shot learning methods. The lower block shows the performance of semi-supervised few-shot learning methods. The MetaGAN is the only method that provides results both on supervised and semi-supervised few-shot learning tasks in *mini*-ImageNet. Since MetaGAN has many different settings, I only show the results with the highest performance.

	FSL Methods	5-way 1-shot	5-way 5-shot	Backbone	Remark
Supervised	PrototypicalNet [6]	44.42 ± 0.84	64.24 ± 0.72	CONV 4	
	Relation Net [7]	49.31 ± 0.85	66.60 ± 0.69	CONV 4	
	Adv.ResNet [129]	55.2	69.6	SRPN	
	MAML [8, 127]	46.47 ± 0.82	62.71 ± 0.71	CONV 4	
	Meta-LSTM [111]	43.44 ± 0.77	60.60 ± 0.71	CONV 4	
	MetaGAN [130]	52.71 ± 0.64	68.63 ± 0.67	ResNet 12	
	Matching Net [117]	48.14 ± 0.78	63.48 ± 0.66	CONV 4	
	EMFSC [126]	63.95 ± 0.61	81.59 ± 0.42	ResNet 12	220×220 input size
Semi-Sup	MetaGAN(Semi) [130]	50.35 ± 0.23	64.43 ± 0.27	ResNet 12	
	Few-shot SSL [5]	50.41 ± 0.31	64.59 ± 0.28	CON 4	
	LST [112]	70.01 ± 1.9	78.7 ± 0.8	ResNet 12	
	Ours	70.56 ± 1.80	77.61 ± 0.87	ResNet 12	

Table 4.2: Performance comparison of different few-shot learning methods for 5-way 1-shot and 5-way 5-shot classifications on *tiered*-ImageNet: mean accuracy and its 95% confidence interval. The upper block shows the performance of supervised few-shot learning methods. The lower block shows the performance of semi-supervised few-shot learning methods.

	FSL Methods	5-way 1-shot	5-way 5-shot	Backbone	Remark
Supervised	MAML [8, 127]	51.67 ± 1.81	70.30 ± 0.08	CONV 4	
	PrototypicalNet [6]	53.31 ± 0.89	72.69 ± 0.74	CONV 4	
	Relation Net [7]	54.48 ± 0.93	71.32 ± 0.78	CONV 4	
	MetaOptNet-SVM [131]	65.99 ± 0.72	81.75 ± 0.53	ResNet 12	
	EMFSC [126]	70.44 ± 0.32	85.43 ± 0.21	ResNet 12	220×220 input size
Semi-Sup	Few-shot SSL [5]	52.39 ± 0.44	70.25 ± 0.31	CON 4	
	LST [112]	77.7 ± 1.6	85.2 ± 0.8	ResNet 12	
	Ours	80.3 ± 1.69	84.5	ResNet 12	

cept for LST [112] and EMFSC [126]. We note that EMFSC used 220×220 high-resolution image input and ensembled 20 base classifiers), while our method only used the conventional setting for *mini*-ImageNet and *tiered*-ImageNet with input image size of 84×84. We compared results with both supervised few-shot learning methods and semi-supervised few-shot learning methods. The MetaGAN[130] is the only method that provides results of *mini*-ImageNet both on supervised and semi-supervised few-shot learning tasks. The MetaGAN has many different settings, I only show the results with the highest performance. I also provide the backbone architecture of each method in the tables.

4.4.4 Ablation studies

In ablation studies, we investigate two important problems: 1) whether self-training strategy can improve the performances; and 2) the effect of the number of classifiers in the ensemble.

Supervised or self-training based semi-supervised We compare the performances of supervised (only utilising labelled training samples) and the proposed self-training (utilising both labelled and unlabelled samples) based semi-supervised methods on 5-way 1-shot classification. The results are shown in the Table 4.3, from which we can see that the proposed semi-supervised self-training method can greatly improve the performance.

Table 4.3: Performance comparison under different settings for 5-way 1-shot classification. Mean accuracy from (Supervised that only utilising labelled training samples) using labelled support set only or from (Self-training that utilising both labelled and unlabelled samples) also using unlabelled support set with pseudo label providedself-training.

	Supervision type	
	Supervised	Self-training(Semi-supervised)
mini-ImageNet	57.26	70.56
tiered-ImageNet	67.99	80.3
# parameters	3M+2052	3M+2052

Number of classifiers in the ensemble The results in the Table 4.4 show two patterns: First, the more classifiers the better performance, with the relative perfor-

Table 4.4: Performance comparison under different settings for 5-way 1-shot classification. Mean accuracy from using different numbers of base classifiers. Bottom row: number of parameters in each model(M means a million).

	Size of ensemble		
	1 classifier	2 classifiers	4 classifiers
mini-ImageNet	67.86	69.40	70.56
tiered-ImageNet	78.6	80.2	80.3
# parameters	3M+513	3M+1026	3M+2052

mance improvement decreasing. This indicates that the proposed method is more pseudo-label robust than a single classifier, relieving the latter's confirmation bias. We also note that, in semi-supervised tasks, the ensemble size is also limited by the number of unlabelled samples, since the more we split unlabelled training set the less pseudo labelled data can be used for a single classifier. The number of splits of unlabelled training set depends on the number of training samples in the class with the minimum number of unlabelled data. Secondly, the proposed method only introduces a negligible increase of parameters.

4.5 Conclusions

We proposed an ensemble method specially tailored for semi-supervised few-shot image classification. The proposed method uses unlabelled samples to facilitate the construction of different base classifiers. The self-training method also improves the performance of the proposed method. The meta-learning which transfers knowledge of features and a metric from many meta-training tasks also improves the performance of each base classifier. Since base classifiers share most of the parameters (with shared feature extractor), the proposed method only has negligibly more parameters than a single model, while producing much better performance than the latter. On two of the widely used benchmark datasets for few-shot image classification, the proposed method achieved state-of-the-art performances.

Chapter 5

General Conclusions

5.1 Conclusions

In this thesis, we provide three works for two problems in weakly supervised learning: stochastic supervision under inaccurate supervision and semi-supervised learning under incomplete supervision.

For the stochastic supervision under inaccurate supervision, we provide four generalisations of stochastic supervision models in Chapter 2. Since the original stochastic supervision model has many strong assumptions and limitations, the motivation of the proposed work is to generalise the assumptions and settings of the stochastic supervision model. The proposed four generalisations extend the stochastic supervision model to asymmetric assessments, multiple classes, feature-dependent assessments, and multi-modal classes, respectively. Corresponding to these generalisations, we also derive four EM algorithms to fit the four new models. We show that the four generalisations can effectively improve the model fitting through illustrative examples of simulated datasets. We implement extensive experiments for these new generalisation. The experiment results have shown that, compared with the original stochastic supervision model, these new generalisations of stochastic supervision models not only extend the original model to a more general setting of classification tasks but also improve the classification accuracy on three famous and widely-used real-world classification datasets, the MNIST dataset, the CIFAR10 dataset, and the EMNIST dataset.

For semi-supervised learning of incomplete supervision, we provide two new methods in Chapter 3 and Chapter 4. In Chapter 3, we provide a semi-supervised learning algorithm for hand pose estimation based on domain adaptation. Unlike many existing semi-supervised learning methods which only make use of labelled and unlabelled samples on the current task, the proposed method transfers knowledge of features and takes advantage of labelled samples from a related domain (the synthetic data) to improve the model performance. The proposed method solves the covariate shift problem in domain adaptation by using the implicit importance weight estimation. Our extensive experiment results lead to two conclusions. Firstly, the proposed weight estimation method can reduce the mean error of estimation. Secondly, transferring knowledge of features and training instances from the synthetic data can greatly improve the model performance. The proposed method achieves the state-of-the-art performance on the NYU dataset.

In Chapter 4, we propose a new method for a challenging semi-supervised learning problem: semi-supervised few-shot learning. The proposed method is under the framework of meta-learning which transfers knowledge of features and a metric from many meta-training tasks. The proposed method provides a tailor-made ensemble method for few-shot learning. Many experiments are implemented on the two widely used few-shot learning benchmark datasets, and the experiment results show that the ensemble can improve the model performance. The proposed method also relieves the pseudo-label noise of the stochastic supervision for the self-training in semi-supervised learning. This work also achieves state-of-the-art performance on two widely used few-shot learning benchmark datasets.

5.2 Future work

To address the limitations of the settings of our proposed methods, there are several extensions of our current work that can be done in the future. We list two pieces of potential future work in this section.

5.2.1 Multiple stochastic supervision model

In Chapter 2 of stochastic supervision, we only consider the situation with one supervisor to provide the assessment. In many machine learning tasks, such as ensemble learning, there are many models that can provide the prediction(assessments). Generalising our current stochastic supervision models to the multiple supervisor case can not only increase the flexibility of the stochastic supervision models but also increase the model performance by assembling the assessments of different supervisors.

Here we also present the training algorithm of this future work. Following the notation of Chapter 2, in this setting a J class classification task contains K supervisors with assessments $z_k, k = 1, \ldots, K$. Each transformed assessment $w_k = (w_{k,1}, \ldots, w_{k,J-1})$, where $w_{k,j} = \log \frac{z_{k,j}}{z_{k,J}}, j = 1, \ldots, J-1$, follows a $(J-1)$-variate Gaussian distribution: $q_{k,j}(w) \sim N(\Delta_{k,j}, \Omega_{k,j})$, so the complete-data likelihood is

$$p(Y, X, W) = \prod_{n=1}^{N} \sum_{j=1}^{J} y_{n,j} [\pi_j f_j(x_n) q_{1,j}(w_{n,1}) q_{2,j}(w_{n,2}) \cdots q_{K,j}(w_{n,K})].$$

An EM algorithm can be further derived. In the E-step, the posterior distribution of latent variables is updated by

$$p(Y|X, W, \theta^{old}) = \prod_{n=1}^{N} \frac{\sum_{j=1}^{J} y_{nj} [\pi_j N(x_n|\mu_j, \Sigma_j) \prod_{k=1}^{K} N(w_{n,k}|\Delta_{k,j}, \Omega_{k,j})]}{\sum_{j=1}^{J} \pi_j N(x_n|\mu_j, \Sigma_j) \prod_{k=1}^{K} N(w_{n,k}|\Delta_{k,j}, \Omega_{k,j})} ,$$

and the corresponding class responsibilities is

$$\gamma(y_{nj}) = \frac{\pi_j N(x_n|\mu_j, \Sigma_j) \prod_{k=1}^{K} N(w_{n,k}|\Delta_{k,j}, \Omega_{k,j})}{\sum_{j=1}^{J} \pi_j N(x_n|\mu_j, \Sigma_j) \prod_{k=1}^{K} N(w_{n,k}|\Delta_{k,j}, \Omega_{k,j})} .$$

In the M step, the model parameter is updated by the following formula:

$$\mu_j^{new} = \frac{\sum\limits_{n=1}^{N} \gamma(y_{nj}) x_n}{\sum\limits_{n=1}^{N} \gamma(y_{nj})} , \Sigma_j^{new} = \frac{\sum\limits_{n=1}^{N} \gamma(y_{nj})(x_n - \mu_{jk})(x_n - \mu_{jk})^T}{\sum\limits_{n=1}^{N} \gamma(y_{nj})} ,$$

$$\Delta_{k,j}^{new} = \frac{\sum\limits_{n=1}^{N} \gamma(y_{nj}) w_{n,k}}{\sum\limits_{n=1}^{N} \gamma(y_{nj})}, \quad \Omega_{k,j}^{new} = \frac{\sum\limits_{n=1}^{N} \gamma(y_{nj})(w_{n,k}-\Delta_j)(w_{n,k}-\Delta_j)^T}{\sum\limits_{n=1}^{N} \gamma(y_{nj})}.$$

5.2.2 Multiple stochastic supervision ensemble for semi-supervised few-shot classification

In Chapter 4 of semi-supervised few-shot learning, we proposed a tailor-made method for few-shot learning to construct base classifiers. The ensemble of this method is simple and direct. Just like many discriminative model based ensemble methods, we simply take an average of different classifiers to make predictions. In reality, the observable assessments $w_k, k = 1,\dots,K$ and latent real label y have more complicated relations. It is natural to use the observable assessments w_k to inference the real label y, so that we can use a multiple stochastic supervision model to ensemble base classifier and provide pseudo labels by jointly inference the real label during the self-training.

The algorithm to fit the model of this future work is also provided here. The notation is the same as the previous future work. In this future work, the observable variable is the transformed assessments $w_{n,k}$, $k = 1,\dots,K$ of K classifiers. The latent variable is the real label y_n. We derive an EM algorithm to fit this model. We assume that $w_{n,k}$ follows a $(J-1)$-variate Gaussian distribution: $q_{k,j}(w) \sim N(\Delta_{k,j}, \Omega_{k,j})$. So that the complete-data likelihood is

$$p(Y,W) = \prod_{n=1}^{N} \sum_{j=1}^{J} y_{n,j}[\pi_j q_{1,j}(w_{n,1}) q_{2,j}(w_{n,2}) \dots q_{K,j}(w_{n,K})].$$

In the E-step, the posterior distribution of latent variables is updated by

$$p(Y|W,\theta^{old}) = \prod_{n=1}^{N} \frac{\sum_{j=1}^{J} y_{nj}[\pi_j \prod_{k=1}^{K} N(w_{n,k}|\Delta_{k,j},\Omega_{k,j})]}{\sum_{j=1}^{J} \pi_j \prod_{k=1}^{K} N(w_{n,k}|\Delta_{k,j},\Omega_{k,j})},$$

and the corresponding class responsibilities are

$$\gamma(y_{nj}) = \frac{\pi_j \prod_{k=1}^{K} N(w_{n,k}|\Delta_{k,j},\Omega_{k,j})}{\sum_{j=1}^{J} \pi_j \prod_{k=1}^{K} N(w_{n,k}|\Delta_{k,j},\Omega_{k,j})} .$$

In the M step, the model parameter is updated by the following formula:

$$\Delta_{k,j}^{new} = \frac{\sum\limits_{n=1}^{N} \gamma(y_{nj}) w_{n,k}}{\sum\limits_{n=1}^{N} \gamma(y_{nj})} , \; \Omega_{k,j}^{new} = \frac{\sum\limits_{n=1}^{N} \gamma(y_{nj})(w_{n,k} - \Delta_j)(w_{n,k} - \Delta_j)^T}{\sum\limits_{n=1}^{N} \gamma(y_{nj})} .$$

Appendix A

An appendix about the loss function of the weighting net

Let the $w(x) = \frac{p(x)}{q(x)}$ is the true density between $p(x)$ and $q(x)$. $w_\theta(x)$ is an estimator of the density ratio $w(x)$. The expected Bregman divergence[135] $BD(w\|w_\theta)$ between true density $w(x)$ and the estimator $w_\theta(x)$ with respect to the density $q(x)$ is [136]:

$$\begin{aligned}\mathbb{E}(BD(w\|w_\theta)) &= \int BD(w\|w_\theta)q(x)dx \qquad \text{(A.1)}\\ &= \int [f(w(x)) - f(w_\theta(x)) - f'(w_\theta(x))(w(x) - w_\theta(x))]q(x)dx \qquad \text{(A.2)}\end{aligned}$$

where $f(w(x))$ is the convex function of a f-divergence.

We minimize A.1 with respect to w_θ is equivalent to minimize the following term:

$$\int [f'(w_\theta(x))w_\theta(x) - f(w_\theta(x))]q(x) - f'(w_\theta(x))\frac{p(x)}{q(x)}q(x)dx \qquad \text{(A.3)}$$

$$= \int [f'(w_\theta(x))w_\theta(x) - f(w_\theta(x))]q(x)dx + \int (-1)f'(w_\theta(x))p(x)dx. \qquad \text{(A.4)}$$

For the Pearson divergence, since:

$$f(w(x)) = \frac{1}{2}(w(x)-1)^2 \tag{A.5}$$

$$f'(w(x)) = w(x) - 1 \tag{A.6}$$

minimizing A.3 is equivalent to minimizing the following formula:

$$\int [\frac{1}{2}w_\theta(x)^2 - \frac{1}{2}]q(x)dx + \int [1 - w_\theta(x)]p(x)dx.$$

Hence, the loss function of the weighting net is:

$$\begin{aligned}\mathscr{L}_g = &\frac{1}{n_1}\sum_{i=1}^{n_1}\left[1 - \frac{u_1(z_i^s)}{u_2(z_i^s)}\right] + \frac{1}{n_3}\sum_{j=1}^{n_3}\left[\frac{1}{2}\left(\frac{u_1(z_j^r)}{u_2(z_j^r)}\right)^2 - \frac{1}{2}\right] \\ &+ \frac{1}{n_1}\sum_{i=1}^{n_1}\left[1 - \frac{u_1(z_i^s)}{u_3(z_i^s)}\right] + \frac{1}{n_5}\sum_{j=1}^{n_5}\left[\frac{1}{2}\left(\frac{u_1(z_j^t)}{u_3(z_j^t)}\right)^2 - \frac{1}{2}\right].\end{aligned} \tag{A.7}$$

Bibliography

[1] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):1–10, 2014.

[2] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2018.

[3] Thriyambakam Krishnan and Subhas C Nandy. Discriminant analysis with a stochastic supervisor. *Pattern Recognition*, 20(4):379–384, 1987.

[4] D. M. Titterington. An alternative stochastic supervisor in discriminant analysis. *Pattern Recognition*, 22(1):91–95, 1989.

[5] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018.

[6] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[7] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

[9] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

[10] Görkem Algan and Ilkay Ulusoy. Image classification with deep learning in the presence of noisy labels: A survey. *arXiv preprint arXiv:1912.05170*, 2019.

[11] Fabrice Muhlenbach, Stéphane Lallich, and Djamel A Zighed. Identifying and handling mislabelled instances. *Journal of Intelligent Information Systems*, 22(1):89–109, 2004.

[12] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, pages 5596–5605, 2017.

[13] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. 2016.

[14] Curtis G Northcutt, Lu Jiang, and Isaac L Chuang. Confident learning: Estimating uncertainty in dataset labels. *arXiv preprint arXiv:1911.00068*, 2019.

[15] John Aitchison and Colin B Begg. Statistical diagnosis when basic cases are not classified with certainty. *Biometrika*, 63(1):1–12, 1976.

[16] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000.

[17] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[18] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In *AISTATS*, volume 2005, pages 57–64. Citeseer, 2005.

[19] Yu-Feng Li, Ivor W Tsang, James T Kwok, and Zhi-Hua Zhou. Convex and scalable weakly labeled svms. *The Journal of Machine Learning Research*, 14(1):2151–2188, 2013.

[20] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[21] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. 2001.

[22] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5070–5079, 2019.

[23] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[24] Zhi-Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541, 2005.

[25] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*, 2017.

[26] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[27] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5049–5059, 2019.

[28] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019.

[29] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[30] Geoffrey J McLachlan. Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association*, 70(350):365–369, 1975.

[31] Terence J O'neill. Normal discrimination with unclassified observations. *Journal of the American Statistical Association*, 73(364):821–826, 1978.

[32] Friedhelm Schwenker and Edmondo Trentin. Pattern classification and clustering: A review of partially supervised learning approaches. *Pattern Recognition Letters*, 37:4–14, 2014.

[33] Friedhelm Schwenker and Edmondo Trentin. Partially supervised learning for pattern recognition. *Pattern Recognition Letters*, 37:1–3, 2014.

[34] Daniel Ahfock and Geoffrey J McLachlan. On missing label patterns in semi-supervised learning. *arXiv preprint arXiv:1904.02883*, 2019.

[35] Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers, 2009.

[36] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 2010.

[37] CB Chittineni. Learning with imperfectly labeled patterns. *Pattern Recognition*, 12(5):281–291, 1980.

[38] T Krishnan. Efficiency of learning with imperfect supervision. *Pattern Recognition*, 21(2):183–188, 1988.

[39] UA Katre and T Krishnan. Pattern recognition with an imperfect supervisor. *Pattern recognition*, 22(4):423–431, 1989.

[40] Charles Bouveyron and Stéphane Girard. Robust supervised classification with mixture models: Learning from data with uncertain labels. *Pattern Recognition*, 42(11):2649–2658, 2009.

[41] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77:329–353, 2018.

[42] Thriyambakam Krishnan and Subhas C Nandy. Efficiency of discriminant analysis when initial samples are classified stochastically. *Pattern Recognition*, 23(5):529–537, 1990.

[43] Thriyambakam Krishnan and Subhas C Nandy. Efficiency of logistic-normal stochastic supervision. *Pattern Recognition*, 23(11):1275–1279, 1990.

[44] D. M. Titterington. Some recent research in the analysis of mixture distributions. *Statistics*, 21(4):619–641, 1990.

[45] John Aitchison. *The Statistical Analysis of Compositional Data*. Chapman & Hall, 1986.

[46] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and Extensions*. John Wiley & Sons, 2007.

[47] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[48] Henri Theil. *Economic forecasts and policy*. North-Holland Pub. Co., 1961.

[49] Trevor Hastie and Robert Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 155–176, 1996.

[50] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002.

[51] Geoffrey McLachlan and David Peel. *Finite Mixture Models*. John Wiley & Sons, 2004.

[52] Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. Finite mixture models. *Annual Review of Statistics and Its Application*, 6:355–378, 2019.

[53] Ryan P Browne, Paul D McNicholas, and Matthew D Sparling. Model-based learning using a mixture of mixtures of gaussian and uniform distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):814–817, 2011.

[54] Cinzia Viroli and Geoffrey J McLachlan. Deep gaussian mixture models. *Statistics and Computing*, 29(1):43–51, 2019.

[55] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[56] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[57] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[58] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[59] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018:7068349, 2018.

[60] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

[61] Natalia Neverova, Christian Wolf, Florian Nebout, and Graham W Taylor. Hand pose estimation through semi-supervised and weakly-supervised learning. *Computer Vision and Image Understanding*, 164:56–67, 2017.

[62] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Crossing Nets: Combining GANs and VAEs with a shared latent space for hand pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 680–689, 2017.

[63] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets deep learning for car instance segmentation in urban scenes. In *British Machine Vision Conference*, pages 81.1–81.12, September 2017.

[64] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2315–2324, 2016.

[65] Chi Li, M Zeeshan Zia, Quoc-Huy Tran, Xiang Yu, Gregory D Hager, and Manmohan Chandraker. Deep supervision with shape concepts for occlusion-aware 3D object parsing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5465–5474, 2017.

[66] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert MÃžller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005, 2007.

[67] Anastasios Yiannakides, Andreas Aristidou, and Yiorgos Chrysanthou. Real-time 3D human pose and motion reconstruction from monocular rgb videos. *Computer Animation and Virtual Worlds*, 30(3-4):e1887, 2019.

[68] Andreas Aristidou. Hand tracking with physiological constraints. *The Visual Computer*, 34(2):213–228, 2018.

[69] Ioannis Rallis, Ioannis Georgoulas, Nikolaos Doulamis, Athanasios Voulodimos, and Panagiotis Terzopoulos. Extraction of key postures from 3D human motion data for choreography summarization. In *International Conference on Virtual Worlds and Games for Serious Applications*, pages 94–101. IEEE, 2017.

[70] Andreas Aristidou, Qiong Zeng, Efstathios Stavrakis, KangKang Yin, Daniel Cohen-Or, Yiorgos Chrysanthou, and Baoquan Chen. Emotion control of unstructured dance movements. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–10, 2017.

[71] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. In *British Machine Vision Conference*, pages 101.1–101.11, 2011.

[72] Srinath Sridhar, Franziska Mueller, Michael Zollhöfer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. Real-time joint tracking of a hand manipulating an object from RGB-D input. In *European Conference on Computer Vision*, pages 294–310, 2016.

[73] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. Real-time 3D hand pose estimation with 3D convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(4):956–970, 2018.

[74] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *European Conference on Computer Vision*, pages 852–863, 2012.

[75] Georg Poier, Konstantinos Roditakis, Samuel Schulter, Damien Michel, Horst Bischof, and Antonis A Argyros. Hybrid one-shot 3D hand pose estimation by exploiting uncertainties. *arXiv preprint arXiv:1510.08039*, 2015.

[76] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[77] Paschalis Panteleris, Iason Oikonomidis, and Antonis Argyros. Using a single RGB frame for real time 3D hand pose estimation in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, pages 436–445, 2018.

[78] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *IEEE international Conference on Computer Vision*, pages 843–852, 2017.

[79] Georg Poier, David Schinagl, and Horst Bischof. Learning pose specific representations by predicting different views. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 60–69, 2018.

[80] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In *European Conference on Computer Vision*, pages 682–697, 2018.

[81] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In *IEEE International Conference on Computer Vision*, pages 1521–1529, 2017.

[82] Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Feature mapping for learning fast and accurate 3D pose inference from synthetic images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4663–4672, 2018.

[83] Danhang Tang, Tsz-Ho Yu, and Tae-Kyun Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *IEEE International Conference on Computer Vision*, pages 3224–3231, 2013.

[84] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. GANerated hands for real-time 3D hand tracking from monocular RGB. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–59, 2018.

[85] Masoud Abdi, Ehsan Abbasnejad, Chee Peng Lim, and Saeid Nahavandi. 3D hand pose estimation using simulation and partial-supervision with a shared latent space. *arXiv preprint arXiv:1807.05380*, 2018.

[86] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[87] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[88] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[89] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images

through adversarial training. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2107–2116, 2017.

[90] Makoto Yamada, Leonid Sigal, and Michalis Raptis. Covariate shift adaptation for discriminative 3d pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2):235–247, 2013.

[91] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[92] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[93] Han Zou, Yuxun Zhou, Jianfei Yang, Huihan Liu, Hari Prasanna Das, and Costas J Spanos. Consensus adversarial domain adaptation. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 5997–6004, 2019.

[94] Francisco Massa, Bryan C Russell, and Mathieu Aubry. Deep exemplar 2D-3D detection by adapting from real to rendered views. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6024–6033, 2016.

[95] Markus Oberweger and Vincent Lepetit. Deepprior++: Improving fast and accurate 3D hand pose estimation. In *IEEE International Conference on Computer Vision*, pages 585–594, 2017.

[96] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[97] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015.

[98] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. 3D convolutional neural networks for efficient and robust hand pose estimation from

single depth images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1991–2000, 2017.

[99] Xinghao Chen, Guijin Wang, Cairong Zhang, Tae-Kyun Kim, and Xiangyang Ji. SHPR-Net: Deep semantic hand pose regression from point clouds. *IEEE Access*, 6:43425–43439, 2018.

[100] Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan. Hand PointNet: 3D hand pose estimation using point sets. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8417–8426, 2018.

[101] Liuhao Ge, Zhou Ren, and Junsong Yuan. Point-to-point regression pointnet for 3D hand pose estimation. In *European Conference on Computer Vision*, pages 475–491, 2018.

[102] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2V-PoseNet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5079–5088, 2018.

[103] Chi Xu, Lakshmi Narasimhan Govindarajan, Yu Zhang, and Li Cheng. Lie-X: Depth image based articulated object pose estimation, tracking, and action recognition on lie groups. *International Journal of Computer Vision*, 123(3):454–478, 2017.

[104] Xinghao Chen, Guijin Wang, Hengkai Guo, and Cairong Zhang. Pose guided structured region ensemble network for cascaded hand pose estimation. *Neurocomputing*, 2019.

[105] Wangyong He, Zhongzhao Xie, Yongbo Li, Xinmei Wang, and Wendi Cai. Synthesizing depth hand images with GANs and style transfer for hand pose estimation. *Sensors*, 19(13):2919, 2019.

[106] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. Augmented skeleton space transfer for depth-based hand pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8330–8339, 2018.

[107] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Dense 3D regression for hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2018.

[108] Kuo Du, Xiangbo Lin, Yi Sun, and Xiaohong Ma. CrossInfoNet: Multi-task information sharing based hand pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9896–9905, 2019.

[109] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[110] Xiaoxu Li, Zhuo Sun, Jing-Hao Xue, and Zhanyu Ma. A concise review of recent few-shot meta-learning methods. *arXiv preprint arXiv:2005.10953*, 2020.

[111] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[112] Xinzhe Li, Qianru Sun, Yaoyao Liu, Qin Zhou, Shibao Zheng, Tat-Seng Chua, and Bernt Schiele. Learning to self-train for semi-supervised few-shot classification. In *Advances in Neural Information Processing Systems*, pages 10276–10286, 2019.

[113] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018.

[114] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pages 1842–1850, 2016.

[115] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563, 2017.

[116] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, 2015.

[117] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[118] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2019.

[119] Kelsey Allen, Evan Shelhamer, Hanul Shin, and Joshua Tenenbaum. Infinite mixture prototypes for few-shot learning. In *International Conference on Machine Learning*, pages 232–241, 2019.

[120] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial Intelligence*, 137(1-2):239–263, 2002.

[121] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*, pages 231–238, 1995.

[122] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[123] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[124] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[125] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations*, 2017.

[126] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. In *IEEE International Conference on Computer Vision*, pages 3723–3731, 2019.

[127] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.

[128] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2019.

[129] Akshay Mehrotra and Ambedkar Dukkipati. Generative adversarial residual pairwise networks for one shot learning. *arXiv preprint arXiv:1703.08033*, 2017.

[130] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. MetaGAN: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems*, pages 2365–2374, 2018.

[131] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[132] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. TADAM: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pages 721–731, 2018.

[133] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[134] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238, 2018.

[135] Takafumi Kanamori, Taiji Suzuki, and Masashi Sugiyama. f-divergence estimation and two-sample homogeneity test under semiparametric density-ratio models. *IEEE Transactions on Information Theory*, 58(2):708–720, 2011.

[136] Masatoshi Uehara, Issei Sato, Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. Generative adversarial nets from a density ratio estimation perspective. *arXiv preprint arXiv:1610.02920*, 2016.