Gaussian Processes based Transfer Learning for Online Multiple-Person Tracking and Building Blocks for Deep Learning

A thesis submitted

for the degree of Doctor of Philosophy

By

Baobing Zhang

Department Electronic and Computer Engineering

College of Engineering, Design and Physical Sciences

Brunel University London

August 7, 2020

Declaration of Authorship

I hereby declare that the content mentioned in this paper has not been submitted for the acquisition of a degree in other schools or places. The content mentioned in the article is under the guidance of my supervisor and the results of exchange ideas with experts in this area.

"Thanks to my solid academic training, after submitting my dissertation, I will walk through the scenes of the laboratory one by one silently. Walking slowly, hoping to print these scenes in my mind. Although four years is not a long time, it is not a fleeting show. Everything has a spirit, for anything that has accompanied me, even a host or a device, I will also be grateful. Four years of Ph.D. experience will turn into memories forever."

Baobing Zhang

Abstract

This thesis mainly study the application of the Gaussian Processes (GPs), especially the application of GPs regression for multiple-person tracking. Furthermore, we explore the scalability of the GPs model. Most existing multiple-person tracking algorithms are still limited by abrupt human pose change, scale change and lighting condition tend to drifts. We introduce a GPs regression based observation model to deal with these challenges. During the tracking process, background information is taken into account to cope with the dynamic background, the GPs regression based observation model fuses prior information to make a tracking decision, which can cope with short term occlusion. Another benefit is that the information of the target in the current frame can be extracted to re-weight the target information in the previous frame, after that a tracking decision is made, this can be seen as a transfer learning strategy. Recently, neural networks have made breakthroughs in various fields. Especially convolutional neural networks (CNNs) have made significant progress in image processing area. The key to the success of CNNs in image processing is that it has a high model complexity and can process high-dimensional features layer-by-layer. After that, extracting different levels of abstract information for further use. We explore the scalability of GPs model, by stacking multiple GPs together, we can construct a deep architecture with building blocks of GPs. This deep hierarchy Gaussian process model is capable of processing high-dimensional input and extracts different levels of abstract information, which is suitable for image processing. We further explore the GPs model equipped with the convolutional kernel, making it benefit from the non-local generalization of convolutional structure and achieving better performance for image data.

Acknowledgements

First of all, I would like to thank my supervisor, Professor. Maozhen Li , who brought me into the hall of academic research and opened up new heights and starting points in my life. And gave me a lot of support and help during my research, not only a supervisor but also a kind-hearted elder.

I would like to thank my family, my parents, for giving me a lot of financial and spiritual support when my research is going to be the most difficult. Every time I call my dad, I can feel the power behind the support. The strength of the family supports me to stick to it until nowadays.

I am very grateful to Professor Qicong Wang from Xiamen University. Dr. Wang has worked in the field of image processing for many years. Every time I discuss with Dr. Wang, I can benefit a lot. Thank you, Dr. Zhengwen Huang, for your help and support. Thanks to the school's technicians and the technical support given by the staff.

Contents

List of Figures

List of Tables

List of Abbreviations

BM	Boltzmann Machine
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DHGP	deep hierarchy Gaussian process
ELBO	Evidence Lower Bound
GPs	Gaussian processes
GP	Gaussian process
GAN	Generative Adversarial Networks
HOG	Histogram of Oriented Gradients
KL divergence	Kullback-Leibler Divergence
LP	Label Propagation
NNs	Neural Networks
RNN	Recurrent Neural Network
RBM	Restricted Boltzmann Machine
SVM	Support Vector Machine

List of Symbols

$\mathbb{E}$	Expected value
i.i.d.	independent and identical distributions
$\langle \cdot \rangle$	Denote expectations

Dedicated to my family…

Chapter 1

Introduction

This chapter briefly describes the background, challenges, motivations, major contributions to the problem investigated and the structure of this thesis.

1.1 Background

Automatic video surveillance of dynamic and complex scenes is one of the most active research topics in Computer Vision. It aims to automatically detect, recognize and track people and objects from image sequences in order to understand and describe dynamics and interactions among them. Computer vision and video-based surveillance have the potential to assist in maintaining public safety and security. Virtually all public spaces and critical infrastructures in the European Community have multiple sensor surveillance systems installed, many of which claim to have automatic surveillance features. Typical application domains for video surveillance include public areas (city, streets, school campuses, museums), transport (airports, train stations, underground, motorways), retail (theft prevention, understanding shopper behavior), and financial institutions (banks and casinos).

Nowadays, most video surveillance systems can only play a passive surveillance role. They are mainly used for post-hoc verification and cannot play the role of prevention and alarm. As a result, the safety and practicability of

the entire system are reduced, which cannot meet modern society. For high-security requirements. The intelligent video surveillance system can identify different objects, find abnormal events in the surveillance picture, and can issue alarms and provide useful information in the fastest way so that it can more effectively assist security personnel in handling crises and maximize the Reduce false positives and false negatives. The emergence of intelligent video surveillance technology has enabled the transformation of surveillance methods from passive to active. It can realize the uninterrupted detection of the video 24 hours 7 days a week, automatically find the abnormal events in the monitoring picture, which greatly improves the security and practicality of the intelligent monitoring system.

Object tracking is a widely studied issue in the machine vision area. It is divided into single target tracking and multiple-target tracking. The former tracks a single target in the video frame, while the latter tracks multiple targets in a series video frame and obtain the motion trajectory of these targets at the same time. Vision-based automatic target tracking has important applications in intelligent monitoring, motion and behavior analysis, and autonomous driving. For example, in an automated driving system, the target tracking algorithm must track the movement of moving cars, pedestrians, and other animals, and make predictions about their future position and speed. In the field of virtual reality, the purpose of human-computer interaction needs to be achieved according to the movements and trajectories of the characters captured by the camera. Compared with single-target tracking, multiple-target tracking is more complicated, except for the object deformation and background interference exists in single-target tracking. Usually, it is also necessary to solve the problems of interaction and occlusion between

targets, automatic initialization, and termination of targets, accurately distinguishing different targets, and how to re-identify the object when it reappears. There are also other issues that need to be considered for multiple-target tracking, such as the position and size of multiple independent targets in the video sequence, changes in the appearance of multiple targets, different movement modes, the impact of dynamic lighting, and mutual occlusion, merging, and separation of multiple targets. Until now, the problems mentioned above are still not well solved.

Since the concept of deep learning proposed by Geoffrey Hinton and others around 2006, with the availability of large amount of data and the continuous growth of the underlying hardware computing power represented by GPUs, deep neural networks have released its potential and made huge breakthroughs in areas such as speech recognition, machine translation, image processing, etc. Among them, the image classification technology based on deep convolutional networks has exceeded the accuracy rate of the human eye, the speech recognition technology based on deep neural networks has reached an accuracy rate of 95%, and the machine translation technology based on deep neural networks has approached the average translation level of humans. The dramatic improvement in accuracy has brought computer vision and natural language processing to the industrialization stage, bringing the rise of new industries. A large number of research resources have been invested in this field all over the world, which has caused explosive development in this field. Deep learning is a powerful tool in the era of big data. Compared with traditional machine learning algorithms, deep learning technology has two main advantages. First, deep learning technology can continuously improve its performance with the increase of data size, while traditional machine learning algorithms can hardly use massive data to continuously improve its performance. The second is that deep learning technology can directly extract features from the data, reducing the work

of designing a feature extractor for each problem, while traditional machine learning algorithms need to manually extract features. In view of the success of deep learning, the development of models in the field of machine learning seems to have to go deeper. A deep model represents higher model complexity, which can fit more complex functions. The deep model can extract different levels of abstract features from the data, which greatly saves tedious data pre-processing processes and labor costs, and can utilize the data more efficiently. Current deep models are mainly based on the building block of neural networks, and only a few other types of deep models have been proposed. Despite huge progress has been made, deep neural networks still have their shortcomings, the model complexity is very high with a huge amount of parameters, and different structures are required for each different task. The amount of data required is huge, but it is difficult to obtain a large amount of labeled data in some specific fields, and the labeled cost is high. Current neural networks lack causal inferencing and are poorly interpret. Combining the interpretability of traditional models, the causal reasoning ability, and the high model complexity of deep neural networks together will be the development direction of the machine learning field in the future.

1.2 The Issues Solved by Gaussian Processes

In probability theory and statistics, Gaussian process is a kind of stochastic process, whose its property is inherited from the normal distribution, and every finite set of random variable obey multivariate normal distribution. It extends multi-dimensional Gaussian distribution to infinite dimensions. In the machine learning community, GPs are a class of non-parametric methods developed based on statistical learning theory and Bayesian theory, the GPs model is fully specified by its mean and covariance function. The GPs

is a powerful model that can be used to represent a distribution over functions. The key idea behind GPs is that a function can be modelled using an infinite-dimensional multivariate Gaussian distribution. In other words, every point in the input space is associated with a random variable and the joint distribution of these is modeled as a multivariate normal distribution. If two input points are close to each other then we expect the value of the function at those points to be similar, in terms of the GPs model: random variables corresponding to nearby points should have similar values when sampled under their joint distribution. Most modern techniques in machine learning tend to avoid this by parameterizing functions and then modelling these parameters (e.g. the weights in linear regression). However, GPs are non-parametric models that model the function directly. This comes with a important benefit: not only can we model any black-box function, but we can also model uncertainty in a probability way. Quantifying uncertainty can be extremely valuable in certain cases. Compared with neural networks (NNs), support vector machine (SVM), the GPs model has much fewer parameters and is suitable for dealing with high-dimensional, non-linear complex regression problems, and capable of strong generalization ability.

1.3 Main Challenges and Motivations

Currently, multiple-person tracking technology in surveillance videos is receiving an increasing attention from both the academic and business communities. As an important basic technology, multiple-person tracking has an important value in many application scenarios such as security monitoring and passenger flow statistics. Multiple-person tracking technology aims to obtain the complete motion trajectory of each target from the appearance to the disappearance from the field of view in the surveillance video scene through computer vision technology. Due to the large differences in

occlusion and complex movement modes in different monitoring scenarios, multiple-person tracking has always been a difficult technique in image recognition.

Data association is an important technique in multiple-person tracking research [1]. Data association is divided into online-based [2] [3] and offline-based methods. The offline-based method obtains the detection information for a period of time and then obtains the motion trajectory of each target during this period according to some optimized strategies. However, this method cannot output each target trajectory in the current frame, which is not suitable for security in applications that require strong real-time performance. The online-based data association method outputs the motion trajectories (positions) of all tracked targets in the current video frame after the current video frame is obtained, and does not modify the motion trajectories obtained before. Although there is no time delay, this method does not make good use of the tracking target information in the current frame, and often the matching error will occur, resulting in discontinuous tracking trajectories and exchange of tracking trajectories. To deal with these issues, we present a novel multiple-person tracking framework by introducing a new Gaussian Process Regression based observation model, which learns in a semi-supervised manner. The background information is taken into consideration to build the discriminative tracker, training samples are re-weighted appropriately to ease the impact of the potential sample misalignment and noisy during model updating. Unlabeled samples from the current frame provide rich information, which is used for enhancing the tracking inference.

In recent years, breakthroughs have been made in the areas of image recognition and speech recognition. This is mainly due to the huge development of deep learning [4], which has also become one of the most popular technology in the field of artificial intelligence [4]. Deep learning is derived from the study of artificial neural networks. Its motivation is to build neural

networks in a similar way to the human brain for analysis and learning. In essence, it is a very powerful model for fitting complex functions. Although great progress has been made, deep learning technology is still imperfect and needs further development. First, the complexity of the deep neural network model is very high, and a large number of parameters lead to a large model size. The second is the large amount of data required for model training, and the training data sample acquisition and labeling costs are expensive, and some scene samples are difficult to obtain. Third, the application threshold is high, the algorithm modeling and the parameter adjustment process are complicated, the algorithm design cycle is long, and the system implementation and maintenance are difficult. The fourth is the lack of causal reasoning. Fifth, there is the problem of interpretability [5]. Due to internal parameter sharing and complex feature extraction and combination, it is difficult to explain what the model has learned. However, due to security considerations and ethical and legal needs, the interpretability of the algorithm is necessary. Based on the successes and problems of deep learning, other types of deep models have been proposed, such as the deep forest model, which is a kind deep model based on the decision tree. Similar to deep neural networks, the deep forest model is a kind of deep structure and capable of extracting features at different levels of abstraction, and has a high model complexity. It has achieved good results on specific data sets [6] [7] [8].

Here we propose a probabilistic deep learning model based on the building block of GPs, which is a statistical probability model with high model complexity thus can fit complex functions. It has much fewer parameters comparing with deep neural networks and is capable of causal inferencing, and variational inference technique is used for training. Because the traditional GPs model is not good at solving image processing problems, further we introduce the convolutional kernel into the GPs model so that the GPs model has the ability to capture different scale patterns in the image, which

make it more suitable for solving image processing problems.

1.4 Major Contributions

The main contribution of this thesis is to propose a transfer learning mechanism based on Gaussian process regression for multiple-person tracking. And then, we verified the feasibility and performance of deep learning based on the GPs building block. Furthermore, we verified the performance of the convolutional kernel in the GPs model and its applicability to different image processing tasks.

For the multiple-person tracking application, we explore how future information can be used to facilitate a tracking decision. To achieve this goal, we employ a novel observation model, which learns in a semi-supervised fashion. The current state of the tracking targets has a significant influence on the final tracking decision. Because a dynamic model utilizes only the previous one state to estimate the current state of the tracking targets, which tends to drift, we fuse all the previous tracking targets information that is what we called prior information to enhance tracking inference, thereby to alleviate drift. The new observation is to update adaptively to avoid the loss of sample diversity. All of these features of the new observation model help alleviate the problem of drifting. There are several major advantages:

- It takes into account the background information in the tracking inference, which is more stable for dynamic background tracking.
- To deal with object occlusions, it fuses all the prior information for tracking decisions, rather than utilizes only the previous one state of the tracking targets.

- It features online transfer learning by utilizing the extracted knowledge from the current state of the tracking targets for generating tracking decisions.

- Experimental results show the proposed observation model outperforms a number of state-of-the-art methods on a number of benchmark datasets.

The deep hierarchy Gaussian process model, being a deep model different from a deep neural network, has a number of advantages such as causal inferencing and fitting uncertainty. Since the deep hierarchy Gaussian process model is a statistical probability model based on the building block of the Gaussian process, it has a complete theoretical foundation of statistical probability. The computation complexity of GPs model has always been the main problem hindering its development. We have experimentally verified that the inducing points input can speed up the inferencing of the Gaussian process model and map the original data to a richer feature space, which is more conducive to classification and regression tasks. The deep hierarchy Gaussian process model is a kind of deep model constructed by stacking multiple GPs together. The model complexity is much higher than deep neural networks. It can fit well complex functions and can extract abstract features at different levels of the data. The introduction of the convolutional kernel has made neural networks a huge breakthrough in image processing. Based on the same idea, the convolutional kernel is introduced into GPs models to make them better handle image processing tasks. We have also verified that the inducing patches also have the ability to accelerate model inference and improve the model performance for image processing tasks.

1.5 Thesis Structure

The rest of this thesis is organised as follows:

- Chapter 2 first introduces the basic knowledge and nature of the Gaussian process, further explains the importance of the Gaussian process model and common covariance functions, and then introduces multi-dimensional Gaussian and parameter estimation, at last discussed the related work.

- Chapter 3 first introduces transfer learning, and proposes a new kind of Gaussian process regression based multi-pedestrian tracking framework. Next, the construction process and advantages of the new appearance model are introduced, and finally the experimental results.

- Chapter 4 first compares the Gaussian process with restricted Boltzmann machine which is the cornerstone model of the deep neural network reflects the advantages of the Gaussian process model, further introduces the deep hierarchy Gaussian process model. Next, the sparse approximation inferencing of the deep hierarchy Gaussian process model and the variational Bayesian optimization method are introduced, finally the experimental results.

- Chapter 5 introduces the convolutional kernel into the Gaussian process model, and further introduces several variants of the convolutional kernel. At the end, the experimental comparison results of Gaussian process models equipped with different convolutional kernels are given.

- Chapter 6 concludes the whole thesis, analyzes the current status of the GPs community, and points out the directions of future work.

Chapter 2

Literature Review

2.1 Basic Gaussian Processes

Gaussian process is a kind of stochastic process, which extends multivariate Gaussian distributions to infinite dimensionality. The property of Gaussian process is inherited from the normal distribution. So that the linear combination of any random variable in the GPs obeys the normal distribution. Each finite-dimensional distribution is a joint normal distribution, GPs is regarded as an infinite-dimensional generalized extension of the joint normal distribution [9].

In supervised learning, two types of methods are usually used to determine the mapping function. The first type is parameterized method, which assumes that training data is generated by a function $f(x; w)$ defined by a parameter w. In this case, the map function $f(x; \cdot)$ and the specific parameter set w together define the parameterized model, and the parametric regression is to find a set of parameters that make the data the "best" interpretation. The parametric methods focus on reducing errors on the training dataset, thus easily leading to overfitting problems. In contrast, the non-parametric method does not make any assumption about the form of the mapping function in advance, they are free to learn any form of a mapping function from the training data. GPs can be employed as non-parametric prior distributions over the latent function f [10]. We briefly state the definition of GPs for

regression here, for more application about GPs please refer [10]. Assume we have a set of training data with n samples $\mathcal{D} = \left\{x_j, y_j\right\}_{j=1}^{n}$, where each input x_j is associated to a real-value output y_j. Define a GPs to model a possible latent function f which maps from input x_j to y_j, its property is fully determined by the mean and covariance function, the common practice is to set a zero-mean and the behavior of the GPs is fully controlled by the covariance function $k\left(x_i, x_j\right)$. In this case, $f(\boldsymbol{x}) \sim GP\left(0, k\left(x_i, x_j\right)\right)$. The most commonly used covariance function is the Squared Exponential Kernel.

$$k_{\mathrm{SE}}\left(x_i, x_j\right) = \sigma^2 \exp\left(-\frac{\left(x_i - x_j\right)^2}{2\ell^2}\right) \tag{2.1}$$

with hyperparameters σ (the latent function power) and ℓ (length-scale, which defines how rapidly the covariance decays along each dimension). The Gaussian processes regression model can be expressed as some noise-less latent function f plus independent noise ε as follows.

$$\boldsymbol{y} = f(\boldsymbol{x}) + \varepsilon \tag{2.2}$$

Where $\boldsymbol{x}$ is the input vector, f is the latent mapping function, and y is the observed value polluted by additive noise. Further, we assume $\varepsilon \sim N\left(0, \sigma_n^2\right)$. The prior distribution of the observed values $\boldsymbol{y}$ is

$$\boldsymbol{y} \sim N\left(0, K(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I_n\right) \tag{2.3}$$

Where the bold $\boldsymbol{x}$ is a vector of the collection of input x. In this case, the model can be used to predict the observed value of y_* of the new test input x_* .

$$p_{GP}\left(y_* | x_*, \mathcal{D}\right) = \begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I_n & k\left(\boldsymbol{x}, x_*\right) \\ k\left(x_*, \boldsymbol{x}\right) & k\left(x_*, x_*\right) \end{bmatrix}\right) \tag{2.4}$$

Where $K(\boldsymbol{x}, \boldsymbol{x}) = K_n = (k_{ij})$ is $n \times n$ order symmetric positive-definite covariance matrix, and matrix element $k_{ij} = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ used to measure the correlative relationship between x_i and x_j. $K\left(\boldsymbol{x}, x_*\right) = K\left(x_*, \boldsymbol{x}\right)^{\mathrm{T}}$ is the $n \times 1$ order covariance matrix between the test input x_* and the input $\boldsymbol{x}$ of the training set; $k\left(x_*, x_*\right)$ is the test input x_* itself covariance; I_n is n-dimensional identity matrix. The posterior distribution of y_* associated with the test input x_* is

$$\begin{aligned} p_{\mathrm{GP}}\left(y_* | x_*, \mathcal{D}\right) =& \mathcal{N}(y_* | k\left(x_*, \boldsymbol{x}\right)\left[k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I_n\right]^{-1} \boldsymbol{y}, \\ & \sigma_{noise}^2 + k\left(x_*, x_*\right) - k\left(x_*, \boldsymbol{x}\right)\left[k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I_n\right]^{-1} k\left(\boldsymbol{x}, x_*\right)) \end{aligned} \tag{2.5}$$

with

$$\mu = k\left(x_*, \boldsymbol{x}\right)\left[k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I_n\right]^{-1} \boldsymbol{y}$$

$$\sigma^2 = \sigma_{noise}^2 + k\left(x_*, x_*\right) - k\left(x_*, \boldsymbol{x}\right)\left[k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 I_n\right]^{-1} k\left(\boldsymbol{x}, x_*\right)$$

corresponding to the posterior mean and variance of the observed value y_*. Gaussian processes regression has a strict theoretical foundation for statistical learning, it is good at handling complex problems such as high dimensions, small samples, and non-linearities. Compared with support vector machine (SVM), neural networks (NNs), GPs has much fewer parameters due to its non-parametric nature which prevents overfitting, and can model uncertainty in a probabilistic way. Much progress has been made in this area in recent years, some of them applied GPs to big data [11] [12] [13], others explored sparse inference methods for GPs [14] [15] [16] [17] [18] [19]. There are also many works that explored the deep Gaussian processes structure [20] [21] [22] [23] [24] [25], some other works explored the relationship between Gaussian processes and neural networks [26] [27] [28] [29] [30] [31] [32]. In the next section, we will look at some illustrative examples of GPs.

2.1.1 Illustrative Examples

Here, we explain Gaussian processes for regression with a few examples. First, we use a generated dataset, the input x is located in a certain range, the corresponding output y is generated with a sin function plus a randomly generated decimal. The fitted Gaussian processes regression model can be seen in FIGURE 2.1. It can be seen that the fitted model has an obvious periodic structure similar to the sin function which generates the output y. In this case, we do not add any noise term in the generating function. In another case, the generating function with noise term is as shown in FIGURE 2.2. Where the red shaded area is the possible noise response corresponding to the related input.

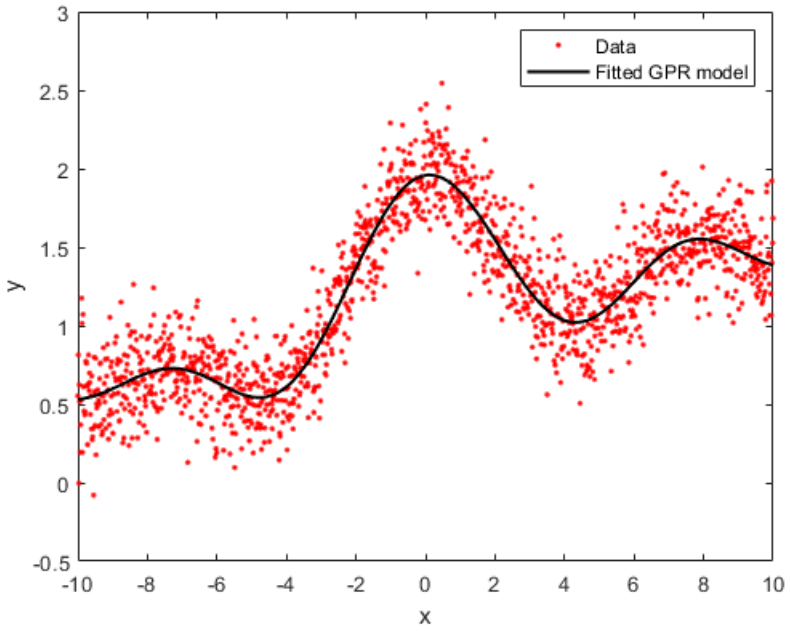

FIGURE 2.1: Gaussian processes regression model fitted from generated data

The following example is demonstrated on a real dataset, we use Gaussian processes regression model to predict the age of abalone. The dataset [33] [34] is collected from the UCI machine learning repository [35]. The

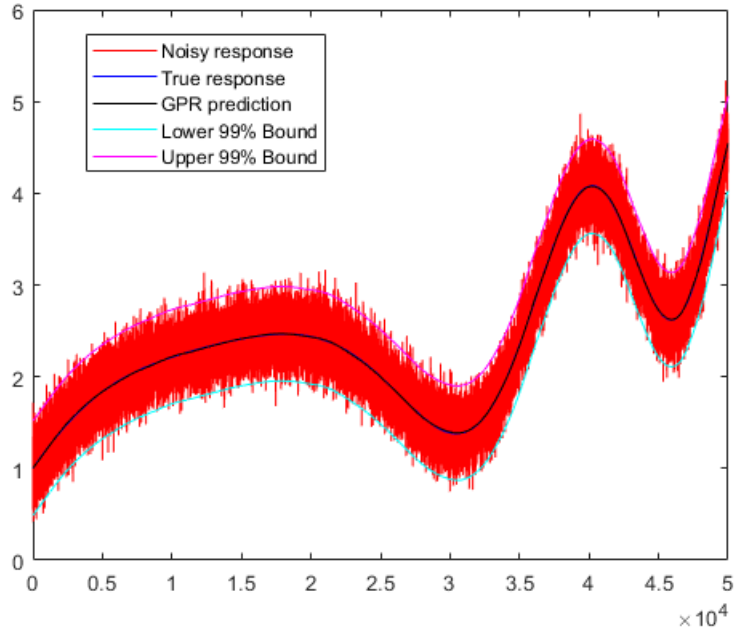

FIGURE 2.2: The visualization of Gaussian processes regression model with noise disturbance

age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope traditionally. Besides, we can predict the age of abalone through some characteristics of abalone, such as gender, diameter, weight, length, etc. This dataset has 8 characteristics of abalone as attribute inputs and the number of the ring (age) of abalone as the predicted output. We fit the Gaussian processes regression model using a subset of the dataset and then use it to predict the age of the abalone, the prediction results are shown as FIGURE 2.3. It can be seen that the prediction results cover most of the normal age, and a few older or early years are not well predicted.

The last example explains how GPs models uncertainty in a probabilistic way. As can be seen in FIGURE 2.4, there are 5 samples, the three prediction models in this figure have to pass through these 5 samples. The blank between the lower prediction (blue one) and the upper prediction (black one) represents a lack of prior knowledge in this area. The GPs model can well fit the lack of prior knowledge and the existence of uncertainty in this area by

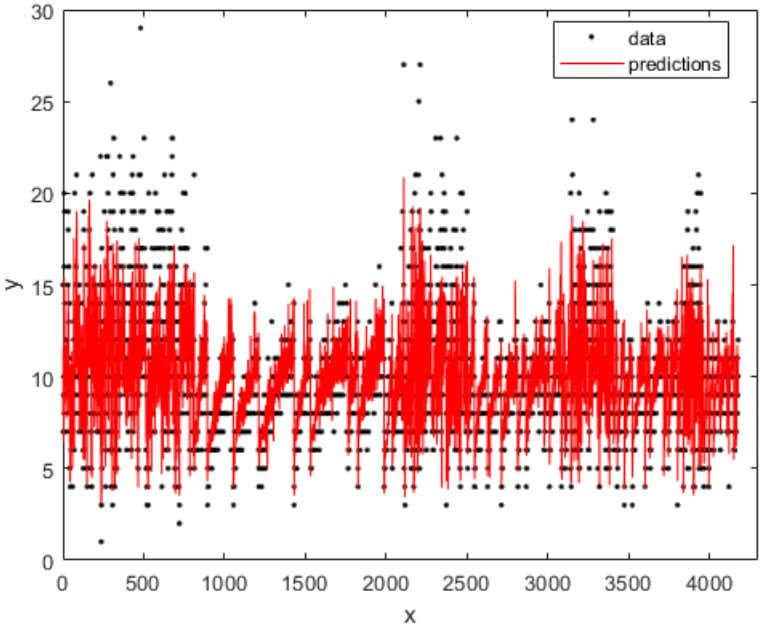

FIGURE 2.3: Abalone age prediction results

means of probability.

2.2 Why Gaussian Processes is Important

The normal distribution also known as the Gaussian distribution is the most important probability distribution in statistics. Many natural phenomena conform to normal distribution. For example, female height, blood pressure, measurement error. This is mainly due to the following two laws.

2.2.1 Law of Large Numbers

In probability theory, the law of large numbers describes that when a large number of repeated experiments are performed, the experimental results should be close to the expected value [36]. The law of large numbers has two main manifestations: the weak law of large numbers and the strong law of large numbers. For a set of independent and identically distributed samples $(x_1, \cdots, x_n)$, the sample mean is as follows

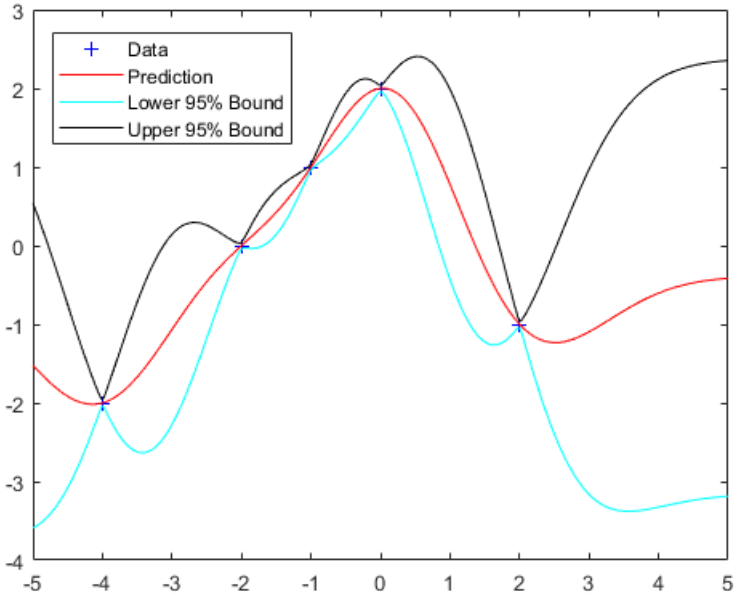

FIGURE 2.4: How GPs handle the lacking of prior knowledge and uncertainty

$$\bar{x}_n = \frac{1}{n}(x_1 + \cdots + x_n) \tag{2.6}$$

Where expected value $\mathrm{E}(x_1) = \cdots = \mathrm{E}(x_n) = \mu$ and they are infinite sequences of Lebesgue integrable random variables. Covariance $\mathrm{Var}(x_1) = \cdots = \mathrm{Var}(x_2) = \sigma^2 < \infty$, the limited assumption here is unnecessary. The difference between strong and weak law of large numbers is the difference in the way of convergence.

The weak law of large numbers (also called Khinchin's law) states that the sample mean converges to the expected value in probability.

$$\bar{x}_n \xrightarrow{P} \mu \quad \text{as } n \to \infty \tag{2.7}$$

That is, for any positive number ε,

$$\lim_{n\to\infty} \mathrm{P}(|\bar{x}_n - \mu| > \varepsilon) = 0 \tag{2.8}$$

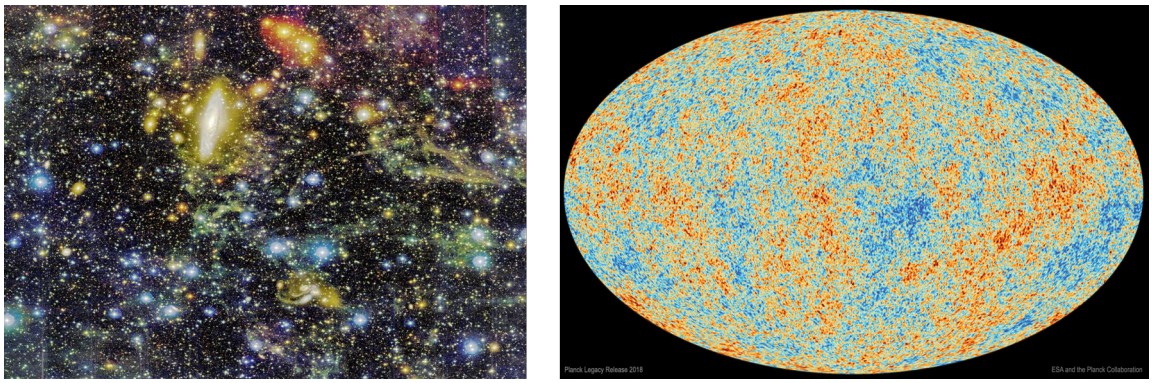

FIGURE 2.5: Gaussian distribution at an extreme large scale

The strong law of numbers states that the sample mean converges to the expected value with probability 1.

$$\bar{x}_n \xrightarrow{\text{a.s.}} \mu \quad \text{as } n \to \infty \tag{2.9}$$

That is,

$$\mathrm{P}\left(\lim_{n\to\infty} \bar{x}_n = \mu\right) = 1 \tag{2.10}$$

In addition, there are Chebyshev's theorem and Bernoulli's law of large numbers etc. For more details see [37].

2.2.2 Central Limit Theorem

The second important law is the Central limit theorem, it is one of the most fundamental and profound concepts in statistics and maybe in all of the mathematics. The Central limit theorem refers to a class of theorems in probability theory that discuss the partial sum distribution of random variable sequences asymptotic to normal distribution. Here we briefly introduce the central limit theorem in the case of independent and identical distributions (*i.i.d.*).

Assume we have a set of *i.i.d.* random variables $(X_1, X_2, \cdots, X_i)$, and the mean is $\mathrm{E}(X_i) = \mu$ and the covariance is $\mathrm{D}(X_i) = \sigma^2 \neq 0$. Then for any real

number X, we have

$$\lim_{n\to\infty} P\left(\frac{\sum_{i=1}^{n}(X_i - n\mu)}{\sqrt{n\sigma^2}} \leq x\right) = \phi(x) \tag{2.11}$$

Among them $\phi(x)$ is standard normal distribution when $n \to \infty$. This sequence is said to have a standard normal limit distribution. Here the idea of the limit is introduced into probability theory, when $n \to \infty$, there is convergence in probability. Here is the normal distribution after normalization, if it is more general, when $n \to \infty$, $\sum_{i=1}^{n} X_i \cong N\left(\mu, \sigma^2\right)$. This is an approximation to a normal distribution, the μ and σ^2 depend on circumstances. Here is a brief introduction to the case of the central limit theorem. For more complicated situations and rigorous mathematical proofs, please refer to professional mathematical papers [37] [38]. From the above theorem, we can know that the effects of many independent random factors are very small, and the effects of their superposition can be regarded as approximately obeying the normal distribution. Such as, the error measurement is affected by factors like ambient humidity, ambient temperature, measuring tool accuracy, and the mood of the surveyor. These influencing factors are independent, and the influences are all very small. In the end, the total measurement error caused by their sum of approximately follows a normal distribution. Another example is the Galton board, at an extreme large scale, we believe that the cosmic background microwave radiation and the interstellar matter distribution approximate the normal distribution as shown in FIGURE 2.5. The picture on the left is interstellar matter captured by the NASA Hubble Telescope [1] and the picture on the right is the visualization of cosmic background microwave radiation made by the European Space Agency Planck prober [2].

[1]https://www.nasa.gov/mission_pages/hubble/main/index.html
[2]https://apod.nasa.gov/apod/ap180722.html

2.3 Covariance Function

The GPs model is determined entirely by its mean and covariance when using a GPs model. The common practice is that setting a zero-mean, so the correlative relationship and uncertainty included in the data are completely encoded in the covariance. It is important to choose a proper kernel function to construct the covariance matrix. Here, we introduce several commonly used kernel functions.

- **Linear Kernel:** This is the simplest kernel function, it is given by the inner product of two elements $< x, y >$ plus a constant c

$$k(x, y) = x^T y + c \tag{2.12}$$

- **Polynomial Kernel:** The Polynomial kernel is a non-stationary kernel. Polynomial kernels are well suited for problems where all the training data is normalized. The slope α, constant term c and polynomial degree d is adjustable.

$$k(x, y) = \left(\alpha x^T y + c\right)^d \tag{2.13}$$

- **Squared Exponential kernel:** The Squared Exponential kernel function is a classic robust radial basis kernel. The robust radial basis kernel has good anti-interference ability to the noise in the data, and its parameters determine the scope of the function.

$$k_{\mathrm{SE}}(x, y) = \sigma^2 \exp\left(-\frac{\|x - y\|^2}{2\ell^2}\right) \tag{2.14}$$

- **Exponential Kernel:** The exponential kernel function is a variant of the Squared Exponential kernel function. It adjusts the L2 distance between the vectors to the L1 distance. This change will reduce the dependence of the parameters, but the scope of application is relatively

narrow.

$$k(x,y) = \sigma^2 \exp\left(-\frac{\|x-y\|}{2\ell^2}\right) \tag{2.15}$$

- **Laplacian Kernel:** The Laplace Kernel is completely equivalent to the exponential kernel, except for being less sensitive for changes in the sigma parameter. Being equivalent, it is also a radial basis function kernel.

$$k(x,y) = \exp\left(-\frac{\|x-y\|}{\ell}\right) \tag{2.16}$$

- **Sigmoid kernel:** The Sigmoid Kernel is also known as Hyperbolic Tangent Kernel or Multilayer Perceptron kernel which comes from the neural Networks field, where the bipolar sigmoid function is often used as an activation function for artificial neurons.

$$k(x,y) = \tanh\left(\alpha x^T y + c\right) \tag{2.17}$$

- **Periodic Kernel:** The periodic kernel allows one to model functions that repeat themselves exactly.

$$k\left(x,x'\right) = \sigma^2 \exp\left(-\frac{2\sin^2\left(\pi\left|x-x'\right|/p\right)}{\ell^2}\right) \tag{2.18}$$

- **Rational Quadratic Kernel:** This kernel is equivalent to adding together many SE kernels with different lengthscales. So, GP priors with this kernel expect to see functions which vary smoothly across many lengthscales.

$$k\left(x,x'\right) = \sigma^2 \left(1 + \frac{\left(x-x'\right)^2}{2\alpha\ell^2}\right)^{-\alpha} \tag{2.19}$$

In addition to these kernel functions mentioned above, there are Multiquadric Kernel, Inverse Multiquadric Kernel, Wave Kernel, Power Kernel. Extensive effort has been spent on the study of these kernel function and

their parameters [10] [39] [40] [41] [42]. In addition to the separate kernel functions mentioned above, there is also a Composite kernel, which is composed of multiple simple kernels. By combining multiple kernel functions, the characteristics of different structures of the data can be captured, an example of the composite kernel is the Spectral Mixture (SM) kernel [43]. In recent years, deep neural networks have made breakthroughs in many aspects, [30] a kernel function has been developed which is equivalent to residual CNN [44]. Significant progress has been made in deep neural networks areas in recent years [4]. In particular, convolutional neural networks have made breakthroughs in the field of image processing. In order to make the GPs model suitable for processing image data, the convolution kernel is proposed, which will be introduced in detail in chapter 5.

2.3.1 Uncertainty

From the perspective of physics, the universe is full of uncertainty, this is determined by the physical nature of the universe. Furthermore, from the perspective of quantum mechanics, there is Heisenberg's uncertainty principle, the position, and velocity of a particle cannot be accurately measured. The quantum state of a particle is developed based on the above principle of uncertainty. In quantum mechanics, a single definite result is not predicted for observation in general. Instead, it predicts a different set of possible outcomes and tells us the probability of each outcome occurring. Back to the GPs model, because of the common practice zero-mean, the property of the data is entirely encoded in the covariance. GPs construct their covariance by the kernel function, and the correlation of the data is encoded into the covariance, and then a set of the probability of the output is given.

2.4 Multi-dimensional Gaussian Distribution and Parameter Estimation

Gaussian processes extend multivariate Gaussian distributions to infinite dimensionality, for a one-dimensional random variable x that obeys the Gaussian distribution of mean μ and variance σ, its probability density is

$$N\left(x|\mu,\sigma^2\right)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{1}{2\sigma^2}(x-\mu)^2\right] \tag{2.20}$$

Expanding to high dimensions, it becomes

$$N(\bar{x}|\bar{\mu},\Sigma)=\frac{1}{(2\pi)^{\frac{D}{2}}}\frac{1}{|\Sigma|^{\frac{1}{2}}}\exp\left[-\frac{1}{2}(\bar{x}-\bar{\mu})^T\Sigma^{-1}(\bar{x}-\bar{\mu})\right] \tag{2.21}$$

Where, $\bar{x}$ represents a D-dimensional vector, $\bar{\mu}$ is the average of this vector, Σ represents the covariance matrix of vector $\bar{\mu}$. Next we will introduce how to derive the high-dimensional probability density function 2.21 from the one-dimensional probability density function 2.20. This is useful in the later introduced Laplacian approximation for GPs.

For simplicity, it is assumed that all variables are independent. It means for probability distribution function $f\left(x_0,\ldots,x_n\right)$, we have $f\left(x_0,x_1,\ldots,x_n\right)=f\left(x_0\right)f\left(x_1\right)f\left(x_n\right)$. Now let's use the two-dimensional example to derive the above formula 2.21.

Assuming we have variable vector $\bar{x}=\begin{bmatrix}x_1\\x_2\end{bmatrix}$, their mean $\bar{\mu}=\begin{bmatrix}\mu_1\\\mu_2\end{bmatrix}$ and covariance $\bar{\sigma}=\begin{bmatrix}\sigma_1\\\sigma_2\end{bmatrix}$. Consider x_1 and x_2 are indenpendent, the Gaussian probability density function of $\bar{x}$ can be expressed as

$$
\begin{aligned}
f(\bar{x}) &= f\left(x_1, x_2\right) \\
&= f\left(x_1\right) f\left(x_2\right) \\
&= \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2}\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2\right) \times \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2}\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right) \\
&= \frac{1}{(2\pi)^{\frac{2}{2}}\left(\sigma_1^2\sigma_1^2\right)^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left[\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 + \left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right]\right)
\end{aligned}
\tag{2.22}
$$

Next, let's deal with the covariance matrix, for a two-dimensional vector $\bar{x}$, its covariance matrix is:

$$
\begin{aligned}
\Sigma &= \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} \\
&= \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}
\end{aligned}
\tag{2.23}
$$

Since x_1, x_2 are independent, so $\sigma_{12} = \sigma_{21} = 0$. In this way, Σ degrades into $\begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$. Determinant of Σ is $|\Sigma| = \sigma_1^2\sigma_2^2$, bringing it into formula 2.22 and replacing the corresponding parameters, we can get

$$
f(\bar{x}) = \frac{1}{(2\pi)^{\frac{2}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left[\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 + \left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right]\right) \tag{2.24}
$$

Until now, we have introduced the left half of the formula 2.21. Next, we will deal with the exp part on the right. The exp part of the high-dimensional Gaussian density function 2.21 is $\exp\left[-\frac{1}{2}(\bar{x}-\bar{\mu})^T\Sigma^{-1}(\bar{x}-\bar{\mu})\right]$, based on the previously calculated Σ, we can find its inverse: $\Sigma^{-1} = \frac{1}{\sigma_1^2\sigma_2^2}\begin{bmatrix} \sigma_2^2 & 0 \\ 0 & \sigma_1^2 \end{bmatrix}$.

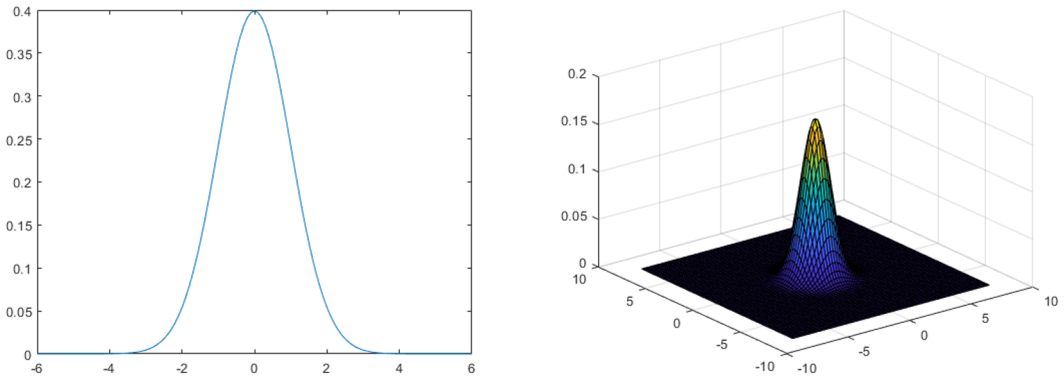

FIGURE 2.6: Gaussian distribution from one dimension expending to two dimension

Next, we perform a two-dimensional expansion on exp part of the formula eq.2.21

$$\begin{aligned}
\exp\left[-\frac{1}{2}(\bar{x}-\bar{\mu})^T\Sigma^{-1}(\bar{x}-\bar{\mu})\right] &= \exp\left[-\frac{1}{2}\begin{bmatrix}x_1-\mu_1 & x_2-\mu_2\end{bmatrix}\frac{1}{\sigma_1^2\sigma_2^2}\begin{bmatrix}\sigma_2^2 & 0\\ 0 & \sigma_1^2\end{bmatrix}\begin{bmatrix}x_1-\mu_1\\ x_2-\mu_2\end{bmatrix}\right] \\
&= \exp\left[-\frac{1}{2}\begin{bmatrix}x_1-\mu_1 & x_2-\mu_2\end{bmatrix}\frac{1}{\sigma_1^2\sigma_2^2}\begin{bmatrix}\sigma_2^2(x_1-\mu_1)\\ \sigma_1^2(x_1-\mu_2)\end{bmatrix}\right] \\
&= \exp\left[-\frac{1}{2\sigma_1^2\sigma_2^2}\left[\sigma_2^2(x_1-\mu_1)^2+\sigma_1^2(x_2-\mu_2)^2\right]\right] \\
&= \exp\left[-\frac{1}{2}\left[\frac{(x_1-\mu_1)^2}{\sigma_1^2}+\frac{(x_2-\mu_2)^2}{\sigma_2^2}\right]\right]
\end{aligned} \tag{2.25}$$

It is the same as the exp part of formula 2.22. The formula 2.21 is the extension of the Gaussian distribution in high-dimension. It is the computable basis for GPs, for a higher dimension can be inferences in the same way. This process can be explained by FIGURE 2.6, as can be seen from the figure the one-dimensional Gaussian distribution is a symmetrical uni-modal curve, and the bell shape is the Gaussian distribution expending to two-dimension space.

2.4.1 Parameter Estimation

Assume we have a set of data $(x_1, x_2, x_3, \ldots, x_m)$ that follows a Gaussian distribution, we need to solve the parameters (μ, Σ) of the Gaussian distribution, the most commonly used method is the Maximum Likelihood Estimate. The one-dimensional Gaussian probability density function is:

$$p\left(x|\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.26}$$

First, we write the likelihood function:

$$\begin{aligned} f\left(x_1, x_2, \ldots, x_m\right) &= \prod_{i=1}^{m}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\left(x_i-\mu\right)^2}{2\sigma^2}\right) \\ &= \left(2\pi\sigma^2\right)^{-\frac{m}{2}}\exp\left(-\frac{\sum_{i=1}^{n}\left(x_i-\mu\right)^2}{2\sigma^2}\right) \end{aligned} \tag{2.27}$$

Then take the logarithm:

$$\ln f\left(x_1, x_2, \ldots, x_m\right) = -\frac{m}{2}\ln\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2 \tag{2.28}$$

Find the derivative and set the derivative to 0 to get the likelihood equation:

$$\frac{\partial \ln f}{\partial \mu} = \frac{1}{\sigma^2}\sum_{i=1}^{n}\left(x_i-\mu\right) = 0 \tag{2.29}$$

$$\frac{\partial \ln f}{\partial \sigma} = -\frac{m}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{n}\left(x_i-\mu\right)^2 = 0 \tag{2.30}$$

Then we can get $\mu = \frac{1}{m}\sum_{i=1}^{m}\left(x_i-\mu\right)$, $\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(x_i-\mu\right)^2}$. As can be seen, this is actually the definition of the mean and standard deviation in the Gaussian function. For high-dimensional cases, the mean and covariance matrix can also be calculated in a similar way.

2.5 Related Work

2.5.1 Multiple-Person Detection and Tracking Technology

This section gives an overview of pedestrian detection in the surveillance system which is the most frequently-used technique. Over the recent years, detecting pedestrian in video scene of a surveillance system is attracting an increasing attention due to its wide range of applications in abnormal event detection [45] [46], human gait characterization [47] [48] [49] [50], person counting in a dense crowd [51] [52] [53], person identification [54] [55] [56], gender classification [57], fall detection for elderly people [58], etc. The scenes obtained from a surveillance video are usually with low resolution and most of the scenes captured by a static camera are with minimal change of background. Objects in outdoor surveillance are often detected in far-field, most existing digital video surveillance systems rely on human observers for detecting specific activities in a real-time video scene. However, there are limitations in the human capacity to monitor simultaneous events in surveillance displays. Hence, human motion analysis in automated video surveillance has become one of the most active and attractive research topics in the area of computer vision and pattern recognition. To achieve this goal, an intelligent system that detects and captures motion information of moving targets for accurate object classification is necessary, the classified object is being used for high-level analysis. In this study, we focus on detecting humans and do not consider the recognition of their complex activities. Pedestrian detection is a difficult task, from a machine vision perspective as it is influenced by a wide range of possible appearance due to changing articulated pose, clothing, lighting, and background. Using appropriate prior knowledge on these limitations can effectively improve the detection performance.

The detection process generally occurs in two steps: object detection and object classification, a general flowchart of human detection is as shown in

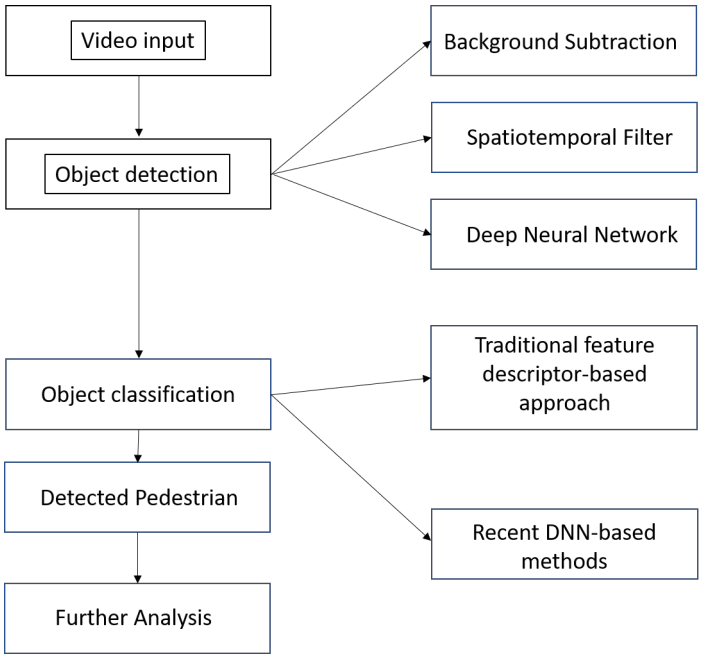

FIGURE 2.7: Flow chart of pedestrian detection

FIGURE 2.7. Object detection could be performed by background subtraction, optical flow, spatio-temporal filtering, and recent deep learning-based detection technologies. Background subtraction is a popular technology for object detection where it attempts to detect moving objects from the difference between the current frame and the background frame in a pixel-by-pixel or block-by-block fashion. There are a few available approaches to performing background subtraction, some of the most commonly used methods are adaptive Gaussian mixture [59], non-parametric background [60], temporal differencing [61], warping background and hierarchical background models [62] [63]. The optical flow-based object detection technique [64] uses characteristics of flow vectors of moving objects over time to detect moving regions in an image sequence. Apart from their vulnerability to image noise, color, and non-uniform lighting, most of the flow computation methods have large computational requirements and are sensitive to motion discontinuities. For motion detection based spatio-temporal filter methods [65], the motion is characterized via the entire three-dimensional (3D) spatio-temporal data volume spanned by the moving person in a series of image sequences. Their

advantages include low computational complexity and a simple implementation process. However, they are susceptible to noise and variations of the timings of movements.

The object classification methods could be divided into two main categories: the traditional feature descriptor-based methods [66] [67] [68] [69] and recent DNN-base methods [70] [71] [72] [73] [74] [75] [76] [77] . Traditional ways of object classification include extract features or descriptors for different classes first and then use classifiers to classify objects. Typical features used are Histogram of Oriented Gradients (HOG) [67], Scale-Invariant Feature Transform (SIFT) [68], Haar-like features [66], Speeded Up Robust Feature (SURF) [69] etc. An object classification algorithm takes images as input and output what objects it contains. Nowadays, with the rapid development of deep learning technologies in recent years, object detection algorithms have also shifted from traditional manual features-based algorithms to deep neural network-based detection technologies. As a result, the detection accuracy has been greatly improved. From the R-CNN [70] and OverFeat [71] originally proposed in 2013, to the subsequent Fast / Faster R-CNN [72] [78], SSD [73], and YOLO [74] [75] [76]series. In less than five years, the deep learning-based object detection technologies blowout, in the network structure from two stage to one stage, from bottom-up to top-down, from single scale network to feature pyramid network [79], from PC-oriented to mobile phones [80] [81] [82]. Many good algorithm have emerged, and these algorithms have excellent detection performance on open object detection datasets.

A successful multi-person tracking algorithm greatly relies on the precision of pedestrian detection, multi-person tracking in a smart surveillance system aims at tracking multiple different people among a series image sequence. The tracking process occurs in two steps: object detection and object association. Previous works such as a Kalman filter [83] and particle filter

[84] take into account only the previous one state information for current state estimation and optimize each trajectory independently. Many tracking-by-detection methods employ background subtractions from one or more cameras [85] [86]. Based on the progress in the field of object detection, the tracking-by-detection method has gradually become the mainstream tracking paradigm. Many works combine detection and association [87] [88] [89], and further link the detection results together to form a tracking trajectory. While other tracking-by-detection methods rely on the mean-shift tracker [90], which finds in the detections that best match to the target appearance by computing a weight image via a gradient ascent procedure. Avidan [91] employed a weak classifier to differentiate the background foreground pixels and a strong classifier to generate a confidence map, which is normally utilised by the mean-shift model in finding a target. Moreover, Benfold and Ian [92] proposed a part-based representation method by using head tracking instead of body tracking to handle the partial occlusion problems. An effective sampling method is proposed by Blake and Isard [93] and applied to visual tracking. Great progress has been made in detection technology, and data association technology has become very important. The data association method used in SORT [3] combines common algorithms such as Kalman Filter and Hungarian algorithm and can greatly improves the computation and data association efficiency of the algorithm. Later the proposed Deep_sort [2] association technique uses more reliable metrics instead of association metrics, meanwhile using a well trained CNN network on large-scale pedestrian datasets to extract features, this has greatly increased the network's robustness to the loss and obstacles. The observation model in these methods fully relies on the detected objects. Different from these methods, the novel observation model presented in Chapter 3 learns in a semi-supervised manner. More importantly, the model not only utilises the previous image patch information for tracking but also considers the current frame data. As a result,

this work reduces the potential risk of drifting. Matching the newly detection to the established object is closely related to the data association problem. The Hungarian algorithm [94] is a classic but effective method, which can be used to find the maximum matching of the detection-tracker pairs in a bipartite graph at runtime. In our work, we employ the Hungarian algorithm to deal with data association problems.

2.5.2 Gaussian Processes Dynamic Systems

Gaussian processes model is noticed by the machine learning community, due to recently people are beginning to realise that a single-layer and fully-connected neural network with i.i.d. prior over its parameters is equivalent to a Gaussian process (GP), in the limit of infinite network width [28] [27] [26]. GPs are a class of non-parametric statistical probability models that can be used to model complex data and then used for classification or regression tasks. Due to the recent success of deep learning, the GP model seems to have to go deeper. It has proven that the deep GP model has a higher model complexity than the shallow model, but training these deep models are often a tricky problem. The first GPs dynamic system [95] was proposed by Lawrence in 2011. This work shows how two GPs can be stacked together using a variational approximation method to train a two-layer deep GPs model. Based on the variational approximate framework, [21] proposed a kind of doubly stochastic variational inference method for training the deep Gaussian processes model. This method samples data from the variational posterior while subsampling the data in mini-batches. They subsample data in minibatch, makes it possible to extend the model to a larger dataset. [24] used Stochastic Gradient Hamiltonian Monte Carlo method to generate samples, to efficiently optimise the hyperparameters. [96] proposed a probabilistic backpropagation algorithm, which is similar to the error backpropagation

in neural networks and is used to train probability statistical models.

Because of the non-parametric nature of the GPs, the fitting function will vary depending on the data, which causes computation problems when fitting a large data set. The time complexity of training the GPs model is $\mathcal{O}\left(n^3\right)$, where n is the number of training points. Inside of the GPs model, there is a large number of expensive matrix inversion operations when constructing the covariance matrix. Such a time complexity severely hinders the expansion of GPs models to large data sets. In recent years, a lot of research work has been devoted to building sparse inferencing GPs models, the main idea behind these works is using a small set pseudo data set of m elements to perform inference. This can reduce computation time from $\mathcal{O}\left(n^3\right)$ to $\mathcal{O}\left(m^2 n\right)$, where m is the number of pseudo input and $m \ll n$. Among them the representative works include [97] [14] [15] [16] [17] [98] [99] [100] [18] [19], of which [97] proposed a sparse inference method with m pseudo-inputs, where the pseudo-inputs are lying in the same domain as the n available data elements. [15] proposed an improved pseudo-input method, first it maps input points to different domains through some specific linear transformations, thus the pseudo-inputs are lying in a different domain from the training points. This can greatly enhance the feature space of the data, improving the performance of the algorithm, and reducing the computational complexity. [16] generalized the m pseudo-input sparse inference method to multiscale for Gaussian processes regression. The latest work [14] has proven that in certain cases the pseudo-input m is increased with the training point n, the relationship $m = \mathcal{O}\left(\log^D n\right)$ is enough, where D is dimension.

In addition to the variational inference method, there is amortized inference [101], expectation propagation [22], and random Fourier features [102], all of those methods can be used to optimize the deep hierarchy Gaussian process model. In Chapter 4 we follow the variational inference method.

2.5.3 Gaussian Processes with Convolutional Kernel

Recently, CNNs have made important breakthroughs in image processing. The success can be attributed to the introduction of convolution operations in neural networks. The convolutional kernel not only reduces the parameters of the neural networks but also captures specific patterns in images. The CNNs with multiple structures have been proposed for different uses, these have been mentioned above. Convolutional kernel has good non-local scalability, there is very few research work that focuses on combining the GPs model with the convolutional kernel. The first one that introduced the convolutional kernel into the GPs model is presented in [103]. The Convolutional Gaussian processes model is still in its infancy, the performance and sparse inferencing skills of the model are far from mature and need further exploration. With an increase in computing power, it is believed that more potential will be released. Compared with CNNs, GPs have clear statistical probability theory, which may provide a direction for probabilistic interpretation for interpretable deep learning.

2.6 Summary

This section first introduces the basic Gaussian process model knowledge, and then explanatory examples. It further explains the common causes of GPs in nature and the commonly used covariance functions. And discussed related work.

Chapter 3

Transfer Learning Based Gaussian Processes Regression for Multiple-Person Tracking

There are many high-resolution cameras installed every day in different parts of the world for surveillance. This provides us with a large amount of image data. The demand for algorithms to automatically process such image data has surged. One particular research area is to deal with multi-person tracking. Even though this area has been intensively studied [91] [104] [105] [106], robust and efficient tracking of multiple persons still remains unsolved taking into account pedestrian occlusions, dynamic background changing, and real-time processing. Building on the tremendous progress of object detection, tracking-by-detection is the most popular paradigm for multi-person tracking in recent years. Compared with background modeling-based trackers tracking-by-detection approaches are more robust to changing backgrounds. However, the object detector, which is used for tracking, usually yields false positive and missing detections making the association between targets and detections difficult to build. In addition, these methods always build trajectories based on two neighboring frames during a long-term occlusion or abrupt human pose changes. As a result, the risk of the tracked target tends

to drift increases.

To address these challenges, many tracking-by-detection approaches combine dynamic and observation models. A dynamic model, like an early Kalman filter [83] or a particle filter [84] takes pedestrian behavior into account for estimating the current state of pedestrians and improves the data association. However, most existing dynamic models utilize only the previous one state for predicting without considering prior information leading to an incorrect estimation when the pedestrian walking direction changes abruptly. An observation model takes into account the pedestrian's appearance changes, particularly when an observation model updates adaptively in a causal way. Most observation models utilize the past appearance information over time but do not consider the current state information of the pedestrian's appearance leading to drifting problems.

To deal with these issues, we present a novel multi-person tracking framework by introducing a new Gaussian Process Regression-based observation model, which learns in a semi-supervised manner. The background information is taken into consideration to build the discriminative tracker, training samples are re-weighted appropriately to ease the impact of the potential sample misalignment and noisy during model updating. Experimental results show that the proposed approach outperforms a number of state-of-the-art methods on some benchmark datasets.

3.1 Transfer Learning

There are several major types of machine learning

- Supervised learning. The goal of the supervised learning is to learn a function from a given labeled training set pair, which is denoted as $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, and when a new data point x^* arrives, it can predict the corresponding label y^* based on this function. The labels

in the training set are always labeled by experienced human annotators and is very expensive. Common supervised learning algorithms include regression analysis and statistical classification.

- Unsupervised learning. For unsupervised learning the training set has no artificially labeled results denote as $\{x_1, \ldots, x_n\}$ compared with supervised learning. The purpose of unsupervised learning is finding a specific structure of the training set. Common unsupervised learning algorithms include Generative Adversarial Networks (GAN) and clustering.

- Semi-supervised learning. Semi-supervised learning is a learning method that combines supervised learning with unsupervised learning. Semi-supervised learning uses a large amount of unlabeled data and a small amount of labeled data simultaneously for the machine learning tasks. When using semi-supervised learning, it will require as few people as possible to work, and at the same time, it will bring higher accuracy.

- Transfer learning. Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem [107]. For example, the knowledge in the current image frame can affect the attributes of the samples in the previous frame, which is exactly the transfer learning strategy used in this thesis.

In addition to the several types mentioned above there are also Reinforcement learning, Self learning, Feature learning, Sparse dictionary learning, etc. Explaining these type of machine learning in detail is far beyond the scope of this thesis. In this chapter, we propose a Gaussian Process Regression based transfer learning strategy, which extracts information from the current frame and then use it for re-weighting the previous sample for final tracking decision making.

3.2 The Gaussian Processes Regression Based Observation Model

This section presents the design details of the multi-person framework, especially the observation model. At each frame I_t, we obtain $D_t = \{d_t^i\}_{i=1}^N$ detections by using Dalal and Triggs'Histogram of Oriented Gradients (HOG) detector [67], which is one of the most successful detectors, especially for pedestrian detection. Here, d_t^i refers to a bounding box at each frame I_t. A tracker T_i is defined as $\{kf_t, X_t, M_t\}$, where kf_t is the Kalman filter used to model the tracker's dynamics. Tracker's center point position and the 2D velocity are used to define Kalman filter's state (x, y, dx, dy). We set process and measurement noise covariance matrices to $\left(\frac{w_t^i}{r_f}\right)^2 \cdot \text{diag}(0.025, 0.025, 0.25, 0.25)$ and $\left(w_t^i\right)^2 \bullet diag(1,1)$ respectively. Here r_f refers to the frame rate of the input sequence, $X_t = \left[x_t^i, y_t^i, w_t^i, h_t^i\right]$ s a bounding box, which stores the current state of the target estimated by the observation model at $\left(x_t^i, y_t^i\right)$ with a size of $\left(w_t^i, h_t^i\right)$. M_t is a set of templates collected through time used for tracker's maintenance.

From the perspective of Bayesian incremental learning for visual tracking, once the tracker is initialized, the state variable $\mathbf{X}_\mathrm{t}^{T_i}$, which describes the location of a tracker T_i at time t can be inferred recursively

$$p\left(\mathbf{X}_t^{T_i}|\mathcal{I}_t^{T_i}\right) \propto p\left(\mathbf{I}_t|\mathbf{X}_t^{T_i}\right) \int p\left(\mathbf{X}_t^{T_i}|\mathbf{X}_{t-1}^{T_i}\right) p\left(\mathbf{X}_{t-1}^{T_i}|T_{t-1}^{T_i}\right) d\mathbf{X}_{t-1}^{T_i} \tag{3.1}$$

where $\mathbf{I}_t^{T_i} = \{I_1, \ldots, I_t\}$ is a set of observed images up to the t-th frame of the tracker T_i, and the distribution of the object location in the current frame $\mathcal{X}_U^{T_i} = \left\{\mathbf{X}_t^{T_i,j}, j = 1, 2, \ldots, n_U\right\}$ is stochastically generated with Kalman filter's prediction as input, which is a small image patch include background information we call n_u the tracking candidates of the tracker T_i. The tracking

result of tracker T_i can be estimated by MAP

$$\hat{\mathbf{X}}_{\mathrm{t}}^{\mathrm{T_i}} = \underset{\mathbf{x}_{\mathrm{t}}^{\mathrm{T}},j}{\arg\max \mathrm{P}} \left(\mathbf{X}_{\mathrm{t}}^{\mathrm{T_t},j} | T_{\mathrm{t}}^{T_{\mathrm{i}}} \right) \tag{3.2}$$

For each sample, we introduce an indicator variable $y_j \in \{-1, +1\}$ to indicate a positive sample with indicator of $(y_j = +1)$ or a negative sample with indicator $(y_j = -1)$ corresponding to $\mathbf{X}_t^{T_t j}$. $\mathcal{X}_U^{T_i}$ is an unlabeled sample set of tracker T_i from the tracking result $\left\{ \mathbf{X}_f^{T_i}, f = 1, 2, \ldots, t-1 \right\}$ up to the (t-1)-th frame. For each tracker, we extract n_L labeled training samples with indicator variables, and then we divide the n_L labeled training samples into two groups, ie, one is the target sample set including n_T samples gathered from the most recent frame denoted as $\mathcal{D}_T = \left\{ (\mathbf{X}, y_j), j = 1, 2, \ldots, n_T \right\}$ which is updated quickly and aggressively. The other auxiliary sample set $D_A = \left\{ (\mathbf{X}; y_j), j = n_T + 1, n_T + 2, \ldots, n_T + n_A \right\}$ is collected at every few intervals and updated slowly and carefully, where $n_L = n_T + n_A$, and y_j refers to a label. Let $\mathbf{1} = [+1, +1, \ldots, +1]^\top$, $y_U = [y_1, y_2, y_3, \ldots, y_{n_u}]^\top$, and then the regression function for the indicators of unlabeled samples y_u can be expressed as

$$\mathcal{R} = P\left(y_U = 1 | \mathcal{X}_U, \mathcal{D}_A, \mathcal{D}_T\right) \tag{3.3}$$

For each sample, we have indicator variable $y_j \in \{-1, +1\}$. Actually each prediction of a GPs on an unlabeled sample set $\mathcal{X}_U^{T_i} = \left\{ \mathbf{X}_t^{T_i, j}, j = 1, 2, \ldots, n_U \right\}$ is a real-valued output of a mean vector in a fixed feature space $\mathcal{K}$, ie, the distance to the hyperplane with the normal vector [108]. To use GPs Regression for classification, we introduce two real-valued latent vectors $l_A \in \mathbb{R}^{n_A}$ and $l_U \in \mathbb{R}^{n_U}$ corresponding to the label y_A and y_U respectively. Intuitively, the further away an unlabeled sample is from the hyperplane (ie, the larger the value of l), the more likely it is that the sample is from the class $y = \mathrm{sign}(l)$. We model this intuitive notion by

$$\mathrm{P}\left(y_i|l_i\right) = \frac{e^{\gamma_i|y_i}}{e^{\lambda_i y_i} + e^{-\gamma_i l_i y_i}} = \frac{1}{1+e^{-2\gamma_i y_i}}, \quad \forall_i = 1, 2, \ldots, n_U \tag{3.4}$$

where $l_u = [l_1, l_2, \ldots, l_{n_u}] \top$, γ is the noise level of the sigmoid noise label output model.We set γ to 10, and for auxiliary samples, we use the same label output model. First of all, we feed the auxiliary data to the GPs model and then get the corresponding latent real-valued l_A, which is the output of Gaussian processes regression. Furthermore, by using the sigmoid noise label generation model, we get the indicator label y_A. We construct the prior covariance matrix depending on all the samples, the correlated structure of the labeled samples, and unlabeled samples have a significant effect on the latent real-valued output. The latent variable l_A is the re-weighted knowledge extracted from the regression, which can be a soft replacement of the indicator label y_A. The latent variable is better for ameliorating sample misalignment problems and is less sensitive to noise compared with the indicator variable.

3.3 Gaussian Processes Latent Variable Model

In order to exploit latent variables $l_\cup$ and l_A, we marginalize over all their possible values (l_U, l_A)

$$\begin{aligned} P\left(y_U = 1|\mathcal{X}_U, \mathcal{D}_A, \mathcal{D}_T\right) &= \iint P\left(y_U = 1, l_A, l_U|\mathcal{X}_U, \mathcal{D}_A, \mathcal{D}_T\right) dl_A dl_U \\ &= \iint P\left(y_U = 1|l_U\right) P\left(l_A, l_U|\mathcal{X}_U, D_A, \mathcal{D}_T\right) dl_A dl_U \end{aligned} \tag{3.5}$$

The latent variable l and labels $y \in (-1, +1)$ are connected via the sigmoid noise label output model. Applying the posterior $P\left(l_A, l_U|\mathcal{X}_U, \mathcal{D}_A, \mathcal{D}_T\right)$ by Bayes theorem, we have

$$
\begin{aligned}
P\left(l_{\mathrm{A}}, l_{\mathrm{U}} | \mathcal{X}_{\mathrm{U}}, \mathcal{D}_{\mathrm{A}}, \mathcal{D}_{\mathrm{T}}\right) &= P\left(l_{\mathrm{A}}, l_{\mathrm{U}} | y_{\mathrm{A}}, \mathcal{X}_{\mathrm{U}}, \mathcal{X}_{A}, \mathcal{D}_{\mathrm{T}}\right) \\
&= \frac{P\left(y_{\mathrm{A}} | l_{\mathrm{U}}, l_{\mathrm{A}}, \mathcal{D}_{\mathrm{T}}, \mathcal{X}_{\mathrm{A}}, \mathcal{X}_{\mathrm{U}}\right) \cdot P\left(l_{\mathrm{U}}, l_{\mathrm{A}} | \mathcal{D}_{\mathrm{T}}, \mathcal{X}_{\mathrm{A}}, \mathcal{X}_{\mathrm{U}}\right)}{P\left(y_{\mathrm{A}} | D_{\mathrm{T}}, \mathcal{X}_{\mathrm{A}}, \mathcal{X}_{\mathrm{U}}\right)}
\end{aligned}
\tag{3.6}
$$

The GPs model $P\left(l_A, l_{\cup} | \mathcal{X}_A, \mathcal{X}_U, \mathcal{D}_{\top}\right)$ is restricted to the auxiliary data and unlabeled data with the mode μ and the covariance matrix $\tilde{\Delta}^{-1} \in \mathbb{R}^{(n_{\mathrm{A}}+n_{\mathrm{U}})\times(n_A+n_{\cup})}$, leading to the regression of the latent variables l_A and l_U

$$
P\left(l_A, l_U | \mathcal{X}_A, \mathcal{X}_U, \mathcal{D}_T\right) \sim \mathcal{N}\left(\mu, \tilde{\Delta}^{-1}\right) \tag{3.7}
$$

3.4 Advantages of The Gaussian Processes Regression Based New Observation Model

We explore how future information can be used to assist in generating a tracking decision, this is inspired by the label propagation algorithm proposed by zhu [109] in 2003. The detailed description of the label propagation algorithm is added in Appendix B. To achieve this goal, we employ the Gaussian Processes Regression based observation model, which learns in a semi-supervised fashion. The knowledge of the current frame tracking candidates was extracted used for re-weighting the previous sample's weight. The current state of the tracking targets has a significant influence on the final tracking decision. Because a dynamic model utilises only the previous one state to estimate the current state of the tracking targets, which tends to drift, we fuse prior information to enhance tracking inference, rather than utilise the previous one frame information, thereby to alleviate drift. The new observation is to update adaptively to avoid the loss of sample diversity. The

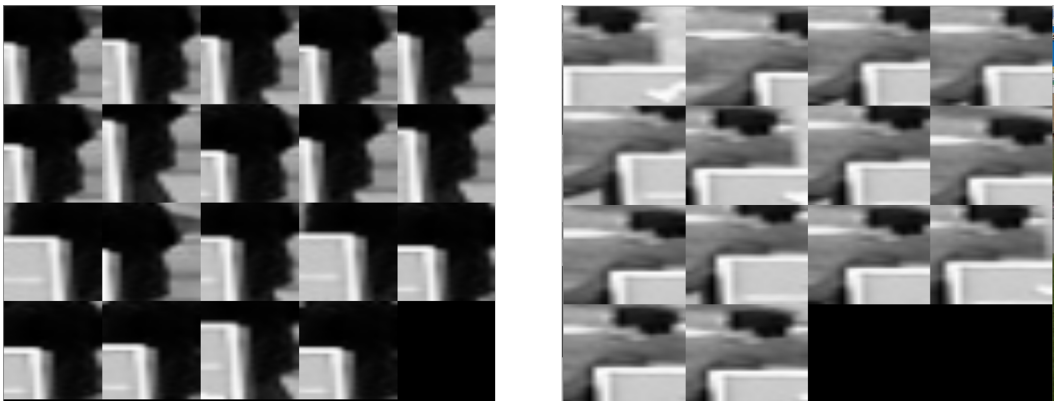

FIGURE 3.1: Unlabeled samples generated from the current frame, image patches containing background information. The first one is positive sample, and the rest are negative samples

tracking candidates set $\mathcal{X}_U^{T_i}$ is stochastically generated from a Normal distribution as shown in FIGURE 3.1. These tracking candidates are small image patches containing background information, which is stable for dynamic background tracking. All of these features of the new observation model help alleviate the problem of drifting.

3.5 Constructing Covariance Matrix

To prevent the binary classification $y \in \{-1, +1\}$ from losing generality, we need to define the kernel matrix $\mathbf{G}_{all}$ properly, which has to be symmetric with non-negative entries. The Gram matrix $\mathbf{G}_{all}$ is based on the weighted graph $\mathcal{G} = (V, E)$ with node set V corresponding to all the samples includes target samples, auxiliary samples, and unlabeled samples. Intuitively, the weighted matrix W of $\mathcal{G}$ specifies the local similarity.

Following this intuition, we use the method proposed in the work of Hu et al [110] to construct the weighted matrix. We further explore the manifold structure between all the samples as suggested in the work of Zhu et al [109]. Gaussian random fields are equivalent to GPs that are restricted to a finite set of points [111]. Following this connection, we establish the link between the graph Laplacian and the kernel method in general. We compute sparse

matrix W here, which empirically tends to have a good performance. Eq. 3.7 can be viewed as a GPs restricted to the auxiliary and unlabeled data. The Laplacian is defined as $\Delta \equiv \mathbf{D} - W$ and degree matrix $\mathbf{D}$ is the row sum of W because the Laplacian Δ has a zero eigenvalue with constant eigenvector $\mathbf{1}$ and as covariance matrix is an improper prior. To get rid of the eigenvalues, a regular Laplacian is obtained by $\tilde{\Delta} = \Delta + \mathbf{I}/\tau^2$, where τ is a small smoothing parameter, $\tilde{\Delta}^{-1}$ is the inverse function of the Laplacian $\tilde{\Delta}$. Therefore, the covariance between any two points i and j. In general, depending on all the data. This is the way semi-supervised learning working and how unlabeled data influences the Prior Knowledge can be viewed as a transfer learning strategy. Detailed Laplace approximate inference for the Gaussian Processes Latent Variable Model is shown in Appendix C.

3.6 Experiments

We test our new approach on a 2.8GHz octa-core CPU, 16GB memory computer; our system is implemented in C++ using OpenCV and Eigen library. There is no unified accepted benchmark available for multi-person tracking. Most of the recent publications have tested their approaches on their own sequence, which varies from the viewpoint, density of pedestrian, and amount of occlusion.We combined them together for the evaluation of our algorithm; these sequences include TUD-Campus [112] and TUD-Stadtmitte [113], PETS'09 S2.L1 - S2.L2 - S2.L3 [114] and TownCenter [92]. Runtime performance depends on the different sequences; most of the sequences can achieve real-time performance. In the following section, we will detail the different several experiments.

3.6.1 Evaluation Metrics

To measure the performance of the proposed work, we employed the CLEAR MOT metrics proposed by Kasturi et al [115]. The Multiple Object Tracking Accuracy (MOTA) considers false positive, missed targets, and identity switches. The Multiple Object Tracking Precision (MOTP) takes into account the average distance between the estimated location and ground truth. Note that the higher value of these metrics stands for better performance. Furthermore, we also compute the metrics described in the work of Bernardin and Stiefelhagen [116], which considers the partially tracked (PT), the counts the number of mostly tracked (MT), mostly lost (ML) trajectories, number of track fragmentations (FM), and identity switches (IDS).

3.6.2 Experiments Analysis

The PETS 2009 Dataset [114] was introduced by the Computational Vision Group University of Reading; this dataset includes four subsets used for different purposes of visual analysis. Among them, the S2 subset sequence is designed for testing the performance of the tracking algorithm. We tested our algorithm on the S2L1, L2, and L3 sequence; the density of pedestrians is raising according to the order of the dataset. S2L1 and S2L2 sequences exhibits randomly walking sparse crowd. The S2L3 sequence shows two individuals, which are bystanders in an empty scene and later join a moving crowd, who walk in the same direction. While this sequence has a very crowded crowd, the crowd is occluded heavily even for pedestrian detection. A brief description of the PETS sequences is as shown in TABLE 3.1. We compared our algorithm with different approaches; the results of these approaches are obtained from the MOTChallenge, which is a part of the famous VideoNet challenge and public available [1] A comprehensive comparison is shown in

[1] https://motchallenge.net/

TABLE 3.1: A brief description of the PETS S2 sequence

	Frame Rate (fps)	Number of frames	Number of Id	Our Method	
				Precision (%)	Recall (%)
PETS2009 S2L1	7	795	19	87	81
PETS2009 S2L2	7	436	43	90	59
PETS2009 S2L3	7	240	44	88	33

TABLE 3.2.

As can be seen from TABLE 3.2 our method produces higher recall and precision in all three datasets. For the S2L1 sequence, the MOTA score success surpasses all other methods. We also get a comparable MOTP score. We believe the slightly lower MOTP score was due to the update of the sample set not perfectly adapt the scale change over time; our method has less missing detection and potentially increases the number of false positive. We also compared our method with the work of Leal-Taixé et al [122], our method gets a higher MOTA score than the work [122]. For the S2L2 sequence, we have the best MOTA score and comparable MOTP score. Besides, we also test our algorithm on the S2L3 dataset, which featured a crowd with heavy occlusions; only a few persons can be tracked accurately. Note that the results of the work of Zhang et al [121] are slightly different from the original paper; we did the test ourselves, which it could be influenced by the parameter tuning or pretreatment optimization. We used the same evaluation metrics as the work of Zhang et al [121] and achieved comparable performance. It is noticed that with the increase of density of people in the sense there are few veterans, the ratio of veterans is much higher in sequence S2L1 than S2L2; it can be explained by the fact that there is more occlusion issues in S2L2 than

TABLE 3.2: Results comparison of selected PETS dataset

Sequence	Tracker	Rcll(%)	Prcn(%)	FP	IDSW	MOTA(%)	MOTP(%)
PETS2009-S2L1	DP_NMS [117]	83.3118	80.9783	910	348	56.2581	71.119
	Ours	81.957	87.6495	537	126	67.6989	62.5369
	TC_ODAL [118]	81.6559	83.4139	755	31	64.7527	70.4317
	TBD [119]	81.2903	84.2434	707	239	60.9462	71.1903
	SMOT [120]	75.1828	91.5663	322	99	66.129	71.5962
	[121]	61.2043	99.0257	15	15	60.2796	68.2216
	[122]	-	-	-	-	67	-
PETS2009-S2L2	[121]	25.2235	98.8199	31	47	24.4656	61.3475
	MotiCon [123]	54.8387	90.4224	560	238	46.5616	67.6273
	Ours	59.9883	90.2896	664	420	49.4559	52.9748
	JPDA_m [124]	49.611	82.4797	1016	139	37.631	65.9038
	SORT [3]	38.2014	82.045	806	240	27.3519	67.361
	RMOT [125]	50.8039	81.3081	1126	190	37.1538	67.6956
	GSCR [126]	35.5461	78.3673	946	162	24.0535	67.5983
PETS2009-S2L3	[121]	26.4168	96.2531	45	30	24.7029	57.1675
	Ours	33.0439	88.5487	187	38	27.9022	53.0151

TABLE 3.3: Results comparison on TownCenter dataset

Sequence	Tracker	Rcll(%)	Prcn(%)	FP	IDSW	MOTA(%)	MOTP(%)
TownCenter	Ours	82.1854	85.5537	6528	247	67.7827	55.827
	GSCR [126]	23.2233	79.1607	437	42	16.5221	67.8075
	SORT [3]	45.0196	74.3359	1111	162	27.2104	67.4241
	EAMTTpub [127]	32.8903	72.0723	911	201	17.3335	68.549
	OMT_DFH [128]	39.6335	69.3513	1252	52	21.3906	67.805
	RNN_LSTM [129]	34.4992	67.1569	1206	299	13.4443	68.7955
	TSDA_OAL [130]	45.0196	64.489	1772	105	18.7605	67.3565
	oICF [131]	38.2065	63.8233	1548	82	15.4029	67.542

S2L1.

We also test our algorithm on the TownCenter [92] dataset (see TABLE 3.3), which initially designed for head tracking, but nowadays widely used for the performance measurement corresponding to multi-person tracking algorithms. This video is high definition (1920×1080 at 25fps),with an average of sixteen people visible at any time [92]. Both the ground truth for this sequence and dataset are public available [2].

The results show that our algorithm outperforms other algorithms in the recall, precision, and MOTA categories, while more detection increases the risk of false positive. PETS'09 S2.L1, S2.L2, and TownCenter dataset, all of these datasets are a part of the MOT challenge 2015 benchmark [132]. In addition, we also selected two additional datasets from this benchmark, which is TUD-Campus [112] and TUD-Stadtmitte [113]. These two datasets are public available [3]. There is also a development kit public available [4], which is used for fair comparison.

[2]http://www.robots.ox.ac.uk/ActiveVision/Publications/benfold_reid_cvpr2011/benfold_reid_cvpr2011.html
[3]https://motchallenge.net/data/2D_MOT_2015/
[4]https://bitbucket.org/amilan/motchallenge-devkit/src/default/

TUD-Campus sequence is filmed at a horizontal view with people walking in two opposite directions occluding each other; the TUD-Stadtmitte sequence is filmed at a lower view, pedestrians dressed similar dresses, walking disorderly, and occlusion happens a lot; this makes the color-based observation model very hard to track people accurately. Results comparison is shown in TABLE 3.4. Our algorithm achieves not so good results. We believe that in the TUD-campus and TUD-Stadtmitte dataset, there are a large number of pedestrians staggered. This makes our algorithm prone to errors when sampling. In future work, we will continue to improve the algorithm.

TABLE 3.4: Results comparison between our algorithm and others

Sequence	Tracker	Rcll(%)	Prcn(%)	FAR	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA(%)	MOTP(%)
TUD-Stadtmitte	DP_NMS [117]	79.58	81.56	1.16	10	8	2	0	208	236	40	25	58.13	69.87
	Ours	54.8	77.4	1.03	10	2	8	0	185	522	6	32	38.3	61.6
	TBD [119]	82.26	82.91	1.09	10	8	2	0	196	205	28	13	62.88	69.51
	TC_ODAL [118]	78.46	86.38	0.79	10	7	3	0	143	249	6	17	65.57	69.91
	CEM [132]	74.91	93.11	0.35	10	6	4	0	64	290	11	9	68.42	69.65
	SMOT [122]	71.10	92.98	0.34	10	4	6	0	62	334	16	26	64.35	70.16
TUD-Campus	DP_NMS [119]	72.14	75.29	1.19	8	3	5	0	85	100	44	22	36.21	74.17
	Ours	27	67.4	0.66	8	0	5	3	47	262	4	18	12.8	66.7
	TBD [121]	76.32	86.43	0.60	8	5	3	0	43	85	9	12	61.83	74.85
	TC_ODAL [120]	58.77	86.83	0.45	8	1	7	0	32	148	6	14	48.18	74.02
	CEM [132]	65.18	89.65	0.38	8	3	5	0	27	125	15	8	53.48	74.73

3.7 Summary

This chapter first introduces the transfer learning, and then proposed a new kind of Gaussian Processes Regression Based observation model. Next, some details of the new model are introduced. At last, through experiments to verify the effectiveness of the new observation model.

Chapter 4

Deep Learning with Gaussian Processes Building Blocks

The most popular sub-field of the machine learning area is deep learning in recent years. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [4], deep learning can be supervised, semi-supervised or unsupervised. Around 2007 Professor Geoffrey Hinton shows how to effectively train feedforward neural networks, this algorithm treats each layer in the network as an unsupervised restricted Boltzmann machine, which is then tuned using a supervised backpropagation algorithm [133] [134]. As computing power continues to grow, and the availability of massive labeled data, deep learning unlocks its potential. Deep architectures such as deep neural networks (DNN), recurrent neural networks (RNN) and convolutional neural networks (CNN) have been applied to many fields including computer vision, natural language processing, machine translation, drug design even for playing game Go [135]. In these areas, the machine has achieved comparable or better performance than human experts. In this section, we investigate a new kind of deep learning architecture which is constructed with the building blocks of GPs, this kind of new model has a very high model complexity and is capable of the ability to learn abstract features. Most importantly it has many advantages compared with other structures of deep learning.

4.1 The Advantages of Gaussian Processes

Modern machine learning experts utilize powerful models for feature extraction of which DNN is only one of many. GPs are another of these models and their primary distinction is their relation to uncertainty. The success of DNN has its foundation on the growth of computing power and the availability of a large amount of data. However, it is still very difficult to obtain a large amount of labeled data in many fields, when applying DNN to a small dataset which tends to overfit. Meanwhile, Neural Networks that require different structures based on different tasks, and there is no theoretical guidance to find a suitable structure. When applying neural networks for image processing tasks, it can be attacked easily by adversarial samples, adding random noise to the image may lead to completely different recognition results. Compare with Neural Networks, GPs has several advantages as follows

- GPs directly capture the model uncertainty, e.g. in the regression case, the GPs model gives the distribution of the latent function instead of giving specific parameter values like weights in a neural network. The uncertainty is fully encoded in the mean and covariance of normal distribution.

- Compared with neural networks GPs are fully controlled by its mean and covariance, much less parameter to learn.

- When using the GPs model, based on different tasks, we can incorporate different prior knowledge by selecting different kernel functions.

- Due to the non-parametric nature of the GPs model, it is proved to prevent overfitting when data is scarce.

In the next few sections, we will introduce a new deep learning structure, which is constructed based on the Gaussian Process building blocks.

This kind of deep structure can benefit from Bayesian inferencing, and also integrate prior knowledge based on different tasks, finally provide stable uncertainty estimates based on Bayesian inference.

4.2 GPs vs RBM

4.2.1 Restricted Boltzmann Machine

A restricted Boltzmann machine (RBM) is a generative stochastic artificial neural network that can learn a probability distribution over its set of inputs. RBM was initially invented by Paul Smolensky in the 1980s [136] until twenty years later Geoffrey Hinton and his collaborators who invented the fast training algorithms make it practical [134]. RBM can be used in dimensionality reduction, classification, feature engineering, and recommendation systems. And it can be extended to deep belief networks by stacking multiple restricted Boltzmann machines, RMB is the cornerstone of modern mainstream deep learning.

RBM as a variant of Boltzmann machine (BM), RBM must be a bipartite graph, the model has two types of nodes: visual node V and hidden node H. In the RBM each node in the visible layer is connected to every node in the hidden layer, but no two nodes in the same layer share a connection. That is the restriction in the RMB, a picture interpretation is as shown in FIGURE 4.1. For an RBM with V visible nodes and H hidden nodes, it is governed by the energy function:

$$E(v,h) = -\sum_{i=1}^{V}\sum_{j=1}^{H} v_i h_j w_{ij} - \sum_{i=1}^{V} v_i b_i^v - \sum_{j=1}^{H} h_j b_j^h \tag{4.1}$$

Each element in the weight matrix $W = \left(w_{i,j}\right)$ specifies the weight between a hidden layer node h_j and a visible node v_j, b_i^v and b_j^h are the real-valued bias of the visible node v_i and hidden node h_j respectively. (v,h)

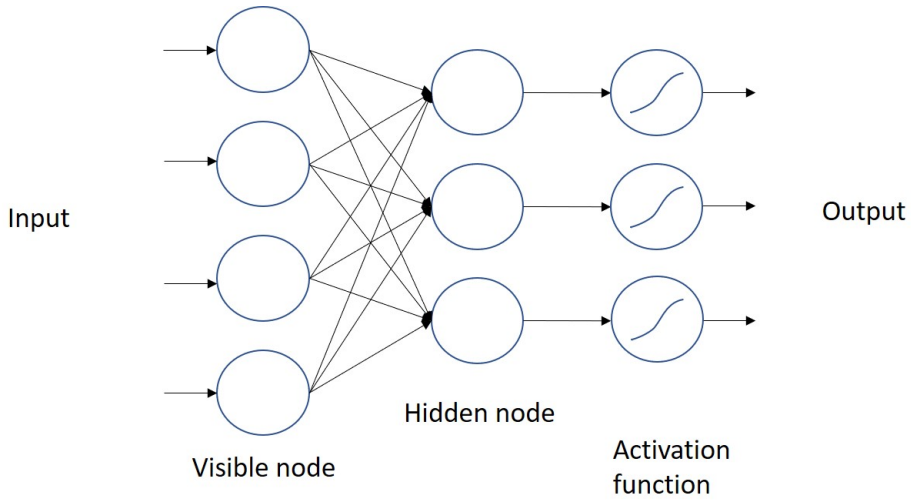

FIGURE 4.1: Schematic diagram of a restricted Boltzmann machine with four visible nodes and three hidden nodes.

is a vector of a collection of visible nodes v_i and hidden node h_j. The joint probability distribution between the hidden and visible layers is given by the energy function

$$P(v,h) = \frac{1}{Z}e^{-E(v,h)} \tag{4.2}$$

Where Z is a partition function defined as the sum of $e^{-E(v,h)}$ over all possible configurations. The marginal probability of a visible node is the sum over all possible hidden layer configurations

$$P(v) = \frac{1}{Z}\sum_{h} e^{-E(v,h)} \tag{4.3}$$

Similarly, we can get the marginal probability of a hidden node with a conditions on all the visible nodes.

$$P(h) = \frac{1}{Z}\sum_{v} e^{-E(v,h)} \tag{4.4}$$

Because RBM is a bipartite graph, and there is no connection between the same layers. The hidden unit activations are mutually independent given the visible unit activations and conversely, the visible unit activations are mutually independent given the hidden unit activations [137]. That is for m visible

nodes and n hidden nodes, the conditional probability of a configuration of the visible nodes v, given a configuration of the hidden nodes h is

$$P(v|h) = \prod_{i=1}^{m} P\left(v_i|h\right) \tag{4.5}$$

Conversely, the conditional probability of h given v is

$$P(h|v) = \prod_{j=1}^{n} P\left(h_j|v\right) \tag{4.6}$$

The goal of training an RBM model is based on the training data to maximize the product of probabilities $\arg\max_{W} \prod_{v \in V} P(v)$, where D is the training set and visible node v. Train the model to find the optimal weight matrix W, the most commonly used algorithm is the contrastive divergence [137].

4.2.2 Gaussian Processes

GPs extend multivariate Gaussian distributions to infinite dimensionality. such that every finite collection of those random variables has a multivariate normal distribution, due to the property of normal distribution every finite linear combination of them is normally distributed. The distribution of a Gaussian process is the joint distribution of all those (infinitely many) random variables, and as such, it is a distribution over functions with a continuous domain. x For a simple regression case, given a training sample set $X = [x_1, x_2, \cdots, x_i]$ and its corresponding output $Y = [f(x_1), f(x_2), \cdots, f(x_i)]$, we have a new sample x^* and we want to prediction the corresponding output y^*. Given the prediction function $f(\cdot)$ a GPs prior, we can directly model the prediction process.

$$\begin{bmatrix} Y \\ y^* \end{bmatrix} \sim N\left(\mu, \begin{bmatrix} K & K_*^{\mathrm{T}} \\ K_* & K_{**} \end{bmatrix}\right) \tag{4.7}$$

Where K is the covariance between training sample X, K_* is the covariance between the training sample and the new sample x^*, K_{**} is the covariance of the new sample x^*. The posterior probability $p\left(y^*|Y, X, x^*\right)$ is a joint normal distribution given by the following formula.

$$p\left(y^*|Y, X, x^*\right) \sim N(\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^{i+1}|\Sigma|}} \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right) \tag{4.8}$$

4.2.3 Comparing GPs with RBM

For the regression case, both the RBM and GPs model aim to fit a prediction function $f(\cdot)$, which must encode some regularity of the dataset. RBM finds the optimal parameters by maximizing the probability. Different from RBM, GPs model can effectively estimate uncertainty which encods in the covariance matrix. GPs tend to prevent overfitting due to their non-parametric nature when data is scarce. Selecting different kernel functions according to the characteristics of the data can integrate different prior knowledge. In RBM, maximizing the conditional probability of the output simply through a linear weighted over the inputs. Compared with RBM, GPs are more expressive of data in a probability way. It has long been known that an infinite wide single layer fully-connected neural network is equivalent to a GPs [26] [28] [27]. By stacking multiple GPs together, we can form a deep hierarchy Gaussian process model (DHGP). Naturally, a two-layer deep hierarchy Gaussian process model is equivalent to a two-layer infinite-wide neural network, and a deeper deep hierarchy Gaussian process is equivalent to a deep infinite-wide neural network. Obviously, the fitting ability of the deep hierarchy Gaussian process model is greater than the neural networks. In the next few sections, we will introduce the deep hierarchy Gaussian process model in detail.

4.3 Sparse Approximations for Gaussian Processes

The GPs is a simple, powerful, flexible and fully probabilistic tool, and can be applied to many fields. The main obstacle to the GPs model is the computational issues, whose computational complexity is cubic with training points. In order to overcome these computational issues, many researchers have proposed sparse inference methods [97] [16] [17] [98] [138] [99] [100]. The main idea behind the sparse approximation for GPs is to select a subset of M inducing points which can be chosen from the training set or other domain. Jointly consider training set and inducing points to establish *effective prior* and then use it to do prediction on the test set. This sparse approximation technique can effectively reduce the computational complexity from $\mathcal{O}\left(N^3\right)$ to $\mathcal{O}\left(NM^2\right)$, where N and M are the numbers of training samples and inducing points respectively. We briefly introduce a general sparse Gaussian Processes inference framework which can also scale to deep learning structure with the Gaussian Process building blocks.

Consider a GP prior function $f(x)$ and real-value *feature extraction function* $g(x,z)$ with *inducing feature* z. After a series of linear transformations, we can get inducing variable $u(z)$

$$u(z) = \int f(x)g(x,z)dx \tag{4.9}$$

The inducing variable $u(z)$ must be linear transform $f(x)$, due to the property of GPs, linear Gaussian is still Gaussian. The inducing variable $u(z)$ can be seen as a projection of the target function $f(x)$ to other domain through the *feature extraction function* $g(x,z)$ over the input sample. This cross-domain inducing variable allows more compact feature space expression and can integrate prior knowledge which is encoded in the *inducing feature* z and feature extraction function $g(x,z)$. Joint consider $f(x)$ and $u(z)$, we

can build *effective prior* which is then used to predict on test dataset more efficiently and accurately. Assume the original Gaussian Process prior $f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right)$. The mean of cross-domain joint Gaussian distribution $f(x)$ and $u(z)$ are given by

$$m(\mathbf{z}) = \mathbb{E}[u(\mathbf{z})] = \int \mathbb{E}[f(\mathbf{x})] g(\mathbf{x}, \mathbf{z}) d\mathbf{x} = \int m(\mathbf{x}) g(\mathbf{x}, \mathbf{z}) d\mathbf{x} \tag{4.10}$$

The covariance of cross-domain $f(x)$ and $u(z)$ are given by

$$\begin{aligned} k\left(\mathbf{x}, \mathbf{z}'\right) &= \mathbb{E}\left[f(\mathbf{x}) u\left(\mathbf{z}'\right)\right] = \mathbb{E}\left[f(\mathbf{x}) \int f\left(\mathbf{x}'\right) g\left(\mathbf{x}', \mathbf{z}'\right) d\mathbf{x}'\right] \\ &= \int k\left(\mathbf{x}, \mathbf{x}'\right) g\left(\mathbf{x}', \mathbf{z}'\right) d\mathbf{x}' \end{aligned} \tag{4.11}$$

$$\begin{aligned} k\left(\mathbf{z}, \mathbf{z}'\right) &= \mathbb{E}\left[u(\mathbf{z}) u\left(\mathbf{z}'\right)\right] = \mathbb{E}\left[\iint f(\mathbf{x}) g(\mathbf{x}, \mathbf{z}) d\mathbf{x} \int f\left(\mathbf{x}'\right) g\left(\mathbf{x}', \mathbf{z}'\right) d\mathbf{x}'\right] \\ &= \int \int k\left(\mathbf{x}, \mathbf{x}'\right) g(\mathbf{x}, \mathbf{z}) g\left(\mathbf{x}', \mathbf{z}'\right) d\mathbf{x} d\mathbf{x}' \end{aligned} \tag{4.12}$$

The mean and covariance are defined both by the original input domain and its augment domain.

4.3.1 Augmented Domain

The *inducing feature* and *feature extraction function* $g(x, z)$ define a transformed augment domain on the input sample set. By select proper $g(x, z)$, the feature space of the original input sample set can be better expressed. The selection of proper feature extraction function $g(x, z)$ can use prior knowledge about the sample set or use some famous family of functions like Fourier transform or Laplace transform. Here are just some examples of using separate augment domain. In fact, we can use an infinite number of augmenting domains

with different *feature extraction function*. The augment domain provides all possible auxiliary information about the input sample set, therefore provide better prediction performance.

4.4 Deep Hierarchy Gaussian Process

When trying to fit the data with a single GP model, we may not know which one is the correct covariance function, considering an improper covariance function may lead to bad results. When fitting complex structured data like audio and image, the data may not easily express as a covariance. Due to the success of deep learning in recent years, a natural idea is to develop a deep structure model. The deep structure can increase model complexity and extract different level abstract features from data at the same time. A series of decision tree-based deep model has been proposed to form a deep forest model and achieve good performance on various datasets [6] [7] [8]. By stacking multiple GPs, we can build a deep hierarchy Gaussian process model. In this deep structure, each node is used as the input of the next layer nodes in the network structure until the output is finally obtained and the connection between different layers is governed by a GP. A picture explanation of the deep hierarchy Gaussian process model is as shown in FIGURE 4.2 with the augmented feature domain and parameter.

In the standard GPs probabilistic inference framework, we have a set of training data $\mathbf{X}$ and the corresponding output $\mathbf{Y}$ respectively, the goal is to estimate the latent function $f = f(x)$. GPs place a nonparametric prior distribution over the latent function f. By adding a Gaussian noise ϵ we have the general GPs generation model, i.e.

$$y_n = f(x_n) + \epsilon_n, \quad \epsilon \sim \mathcal{N}\left(0, \sigma_\epsilon^2 I\right) \tag{4.13}$$

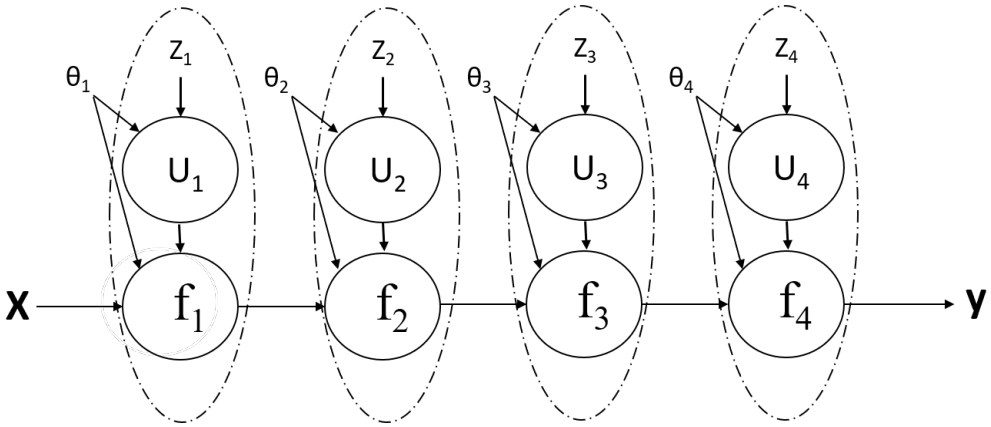

FIGURE 4.2: The general architecture of a deep hierarchy Gaussian process model, the dotted circle represents an augment domain with inducing variable and parameter.

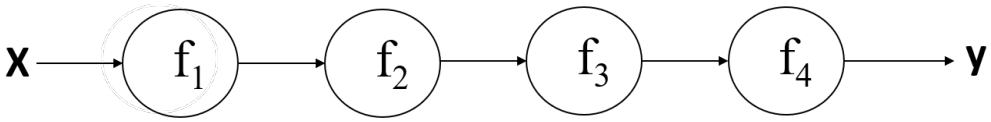

FIGURE 4.3: Deep hierarchy Gaussian process obtained by extending the standard Gaussian processes.

We follow the common practice with zero-mean, i.e. $f(\mathrm{x}) \sim \mathcal{GP}\left(0, k\left(x, x'\right)\right)$. The correlation information of paired data in the dataset is encoded in the covariance function, the GP prior to the latent function f is fully depended on the selection of basis kernel function. We can choose different kernel functions according to the characteristics of data to make the data more effectively expressed. We assume that each output y_n is generated independently by the input x_n, sum all the latent f_n, we have $F = \{f_n\}_n^N$. Due to that the latent function is normal distributed, the marginal likelihood is computed tractably.

$$\begin{aligned} p(\mathbf{Y}|\mathbf{X}) &= \int \prod_{n=1}^{N} p\left(y_n|f_n\right) p\left(f_n|x_n\right) \mathrm{d}\mathbf{F} \\ &= \mathcal{N}\left(\mathbf{Y}|\mathbf{0}, \mathbf{K}_{NN} + \sigma_\epsilon^2 \mathbf{I}\right), \mathbf{K}_{NN} = k(\mathbf{X}, \mathbf{X}) \end{aligned} \tag{4.14}$$

Extending the standard GPs model directly, we get a deep hierarchy Gaussian process model as shown in FIGURE 4.3.

$$p(y|x) = \int p\left(y|f_4\right) p\left(f_4|f_3\right) p\left(f_3|f_2\right) p\left(f_2|f_l\right) p\left(f_l|x\right) d\mathbf{F} \tag{4.15}$$

Or in a recursive form

$$y(x) = f_4\left(f_3\left(f_2\left(f_1(x)\right)\right)\right) \tag{4.16}$$

As the connection between each layer is governed by a GP, the deep hierarchy Gaussian process can be easily expanded to a deeper model. Here we introduce automatic relevance determination (ARD) covariance function for the GP.

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sigma_{ard}^2 e^{-\frac{1}{2}\sum_{q=1}^{Q} w_q\left(x_{i,q}-x_{j,q}\right)^2} \tag{4.17}$$

This covariance function specifies a different weight w_q for different latent dimensions. This weight is the parameter of a deep hierarchy Gaussian process model and gets the optimal value under the framework of Bayesian variational inferencing. Compared with deep neural networks, deep hierarchy Gaussian process have far fewer parameters and the GPs is a kind of stochastic process, which is computable and analyzable. Because of the $p\left(y|f\right)$ and covariance function introduces non-linearity, making the direct Bayesian inference challenging. In the next section, we will introduce an approximate Bayesian variational inference framework, which uses a variational distribution to approximate the true distribution.

4.5 Variational Bayesian Inference

Due to the introduction of non-linearities in the model, direct calculations are difficult. Variational Bayesian inference provides a type of approximate

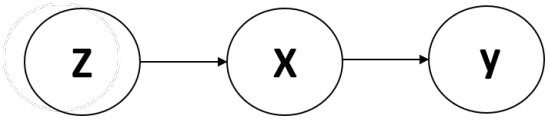

FIGURE 4.4: A two hidden layer hierarchy case and each connection is governed by separate Gaussian processes.

method for intractable integrals. Variational inference typically used in complex statistical models which consist of observed variables as well as parameters and latent variables. The parameters and latent variables usually group together as unobserved variables. Variational Bayesian inference is primarily used for approximating the posterior probability of the unobserved variables and derives evidence lower bound (ELBO) for the marginal likelihood of the observed data. The higher the marginal likelihood indicates that the given model fits the data better, hence that the higher the probability that the model can generate the data. A usual approach is to use a variational distribution q to approximate the true distribution P, and then minimise the Kullback-Leibler divergence between q and P. Minimising KL divergence between the variational distribution q and true distribution P is equivalent to maximising ELBO. (The detailed derivation and explanation of KL divergence is given in Appendix A)

Many previous works have focused on the variational inference for training complex statistic model [139] [95] [20]. In the deep hierarchy Gaussian process model training procedure, for a two hidden layer simple case as shown in FIGURE 4.4, we introduce the inducing feature $\mathbf{Z}$ to enhance the feature space of the input observation data $\mathbf{X}$, this process is governed by a GPs $\mathbf{F}^X$. The output $\mathbf{Y}$ is generated by another independent GPs $\mathbf{F}^Y$. We need to optimise the corresponding evidence lower bound.

$$\log p(\mathbf{Y}) = \log \int_{\mathbf{X},\mathbf{Z}} p(\mathbf{Y}|\mathbf{X}) p(\mathbf{X}|\mathbf{Z}) p(\mathbf{Z}) d\mathbf{X} d\mathbf{Z} \tag{4.18}$$

In this case, the evidence lower bound (ELBO) $L(q) \leq \log p(\mathbf{Y})$ with

$$L(q) = \int_{\mathbf{X},\mathbf{Z},\mathbf{F}^Y,\mathbf{F}^X} q \log \frac{p\left(\mathbf{Y},\mathbf{F}^Y,\mathbf{F}^X,\mathbf{X},\mathbf{Z}\right)}{q} \tag{4.19}$$

Where q is the variational distribution used to approximate the posterior probability of the unobserved variables, notice the joint distribution of the numerator in the integral has the expanded form.

$$p\left(\mathbf{Y},\mathbf{F}^Y,\mathbf{F}^X,\mathbf{X},\mathbf{Z}\right) = p\left(\mathbf{Y}|\mathbf{F}^Y\right) p\left(\mathbf{F}^Y|\mathbf{X}\right) p\left(\mathbf{X}|\mathbf{F}^X\right) p\left(\mathbf{F}^X|\mathbf{Z}\right) p(\mathbf{Z}) \tag{4.20}$$

Further, we augment the input and output layers with pseudo-input $\tilde{\mathbf{Z}}$ and $\tilde{\mathbf{X}}$ respectively, we have the corresponding inducing variable $\mathbf{U}^X$ and $\mathbf{U}^Y$. Then we can have the augmented probability space.

$$\begin{aligned} p\left(\mathbf{Y},\mathbf{F}^Y,\mathbf{F}^X,\mathbf{X},\mathbf{Z},\mathbf{U}^Y,\mathbf{U}^X,\tilde{\mathbf{X}},\mathbf{Z}\right) =& p\left(\mathbf{Y}|\mathbf{F}^Y\right) p\left(\mathbf{F}^Y|\mathbf{U}^Y,\mathbf{X}\right) p\left(\mathbf{U}^Y|\tilde{\mathbf{X}}\right) \\ & \cdot p\left(\mathbf{X}|\mathbf{F}^X\right) p\left(\mathbf{F}^X|\mathbf{U}^X,\mathbf{Z}\right) p\left(\mathbf{U}^X|\tilde{\mathbf{X}}\right) p(\mathbf{Z}) \end{aligned} \tag{4.21}$$

The inducing variable $\mathbf{U}^X$ and $\mathbf{F}^X$ are joint Gaussian distribution and can effectively enhance the feature space of the observed data. This is true for $\mathbf{U}^Y$ and $\mathbf{F}^Y$. From now we can define the variational distribution q with the augmented domain.

$$\begin{aligned} q = & p\left(\mathbf{F}^Y|\mathbf{U}^Y,\mathbf{X}\right) q\left(\mathbf{U}^Y\right) q(\mathbf{X}) \\ & \cdot p\left(\mathbf{F}^X|\mathbf{U}^X,\mathbf{Z}\right) q\left(\mathbf{U}^X\right) q(\mathbf{Z}) \end{aligned} \tag{4.22}$$

Substitute the variational distribution q back to the evidence lower bound $L(q)$ with the augmented domain, we have the following quantity.

$$L(q) = \int q \log \frac{p\left(\mathbf{Y}|\mathbf{F}^{\mathbf{Y}}\right) p\left(\mathbf{U}^{\mathbf{Y}}\right) p\left(\mathbf{X}|\mathbf{F}^{\mathbf{X}}\right) p\left(\mathbf{U}^{\mathbf{X}}\right) p(\mathbf{Z})}{q'} \tag{4.23}$$

Where $q' = q\left(\mathbf{U}^{\mathrm{Y}}\right) q(\mathbf{X}) q\left(\mathbf{U}^{\mathrm{X}}\right) q(\mathbf{Z})$ and the above $L(q)$ is corresponds to $\left\{\mathbf{X}, \mathbf{Z}, \mathbf{F}^{\mathbf{Y}}, \mathbf{F}^{\mathbf{X}}, \mathbf{U}^{\mathbf{Y}}, \mathbf{U}^{\mathbf{X}}\right\}$. Further, we can factorize the above $L(q)$ by grouping related variables together, the evidence lower bound can be written as follows:

$$L(q) = \mathrm{g}_Y + \mathbf{r}_X + \mathcal{H}_{q(\mathbf{X})} - \mathrm{KL}(q(\mathbf{Z}) \| p(\mathbf{Z})) \tag{4.24}$$

Where the symbol $\langle \cdot \rangle$ is used for denoting the expectations, $\mathcal{H}$ represents the information entropy of the distribution. *KL* represents the Kullback Leibler divergence.

$$\begin{aligned} \mathbf{g}_Y &= g\left(\mathbf{Y}, \mathbf{F}^Y, \mathbf{U}^Y, \mathbf{X}\right) \\ &= \left\langle \log p\left(\mathbf{Y} | \mathbf{F}^Y\right) + \log \frac{p\left(\mathbf{U}^Y\right)}{q\left(\mathbf{U}^Y\right)} \right\rangle_{p\left(\mathbf{F}^Y | \mathbf{U}^Y, \mathbf{X}\right) q\left(\mathbf{U}^Y\right) q(\mathbf{X})} \end{aligned} \tag{4.25}$$

$$\begin{aligned} \mathbf{r}_X &= r\left(\mathbf{X}, \mathbf{F}^X, \mathbf{U}^X, \mathbf{Z}\right) \\ &= \left\langle \log p\left(\mathbf{X} | \mathbf{F}^X\right) + \log \frac{p\left(\mathbf{U}^X\right)}{q\left(\mathbf{U}^X\right)} \right\rangle_{p\left(\mathbf{F}^X | \mathbf{U}^X, \mathbf{Z}\right) q\left(\mathbf{U}^X\right) q(\mathbf{X}) q(\mathbf{Z})} \end{aligned} \tag{4.26}$$

The term $\mathbf{g}_Y$ is only respected to the output. Even this is a simple case the model can be expanded vertically by adding more hidden layers with respect to more term $\mathbf{r}_X$. The above is a general framework for optimizing complex statistic models by maximizing evidence lower bound (ELBO) which is equivalent to minimize the KL divergence between a variational distribution and true distribution. Base on the general framework, efficient variant optimization methods have been proposed [21] [22] [23] [25] [24] used for training deep hierarchy Gaussian process model. [21] proposed a method of double stochastic variational inference optimization approach, which sampling form variational posterior meanwhile subsample the data in mini-batches allow the model scale to a large dataset. In the next section, we will verify the effectiveness of the inducing variables and the performance of the deep hierarchy Gaussian process model with different size datasets.

4.6 Experiments

In order to test the performance of the deep hierarchy Gaussian process Model, as well as the computational complexity, we tested DHGP model on eight UCI datasets, which cover small to medium scales. In all experiments, we employed the same hyperparameters for model initialization. And all layers adopted RBF kernels. We further tested the deep hierarchy Gaussian process model on the MNIST dataset, which was previously considered to be difficult for GPs models because of the high-dimensional input and computational complexity of the image dataset.

UCI Dataset for Regression

We compared DHGP model with other sparse Gaussian processes inferences on eight UCI datasets. We initialized the deep hierarchy Gaussian process model of 1-4 layers each with 100 inducing points. We compared with SGPR [138], SVGP [140], FITC [97] with 100 and 300 inducing points respectively. We also compared the DHGP model with a Bayesian neural network with a hidden layer, which uses the PBP [96] method for inferencing, and it is the most efficient Bayesian neural network inferencing method. The experimental results are directly observed from this work. All experiments used the same parameter configurations. We split the data into 80% for training and 20% for testing, and with 20 - fold cross-validation. TABLE 4.1 shows the comparison of the DHGP model with other methods on 8 UCI datasets of Regression task. TABLE 4.2 shows the RMSE results on eight UCI datasets, smaller numbers indicate better results.

In all of these datasets, the DHGP model always has comparable or better

TABLE 4.1: Comparison of DHGP model with other methods on regression task with eight UCI datasets. Test log-likelihood results, the number highter indicates better performance.

	N	D	SGPR100	SVGP100	FITC100	SGPR300	SVGP300	FITC300	DHGP1	DHGP2	DHGP3	DHGP4	BNN(PBP)
boston	506	13	-2.3847	-2.3847	**-2.3207**	-2.3265	-2.3275	-2.2659	-2.3848	-2.3973	-2.4093	-2.4159	-2.57
energy	768	8	-1.0719	-1.2410	-0.7524	**-0.5359**	-0.6990	-0.8101	-1.1691	-0.7188	-0.8446	-0.9013	-2.04
concrete	1030	8	-3.1331	-3.1391	**-2.9466**	-3.0608	-3.0599	-3.0501	-3.1418	-3.1603	-3.1849	-3.2169	-3.16
wine-red	1599	11	-0.8954	-0.8955	-0.9139	-0.8941	-0.8941	**-0.8737**	-0.8959	-0.8966	-0.8985	-0.8993	-0.97
wine-white	4898	11	-1.0320	-1.0321	-1.0254	-1.0234	-1.0244	**-0.9465**	-1.0341	-1.0317	-1.0317	-1.0311	-
power	9568	4	-2.8197	-2.8200	-2.8400	**-2.8049**	-2.8061	-2.8141	-2.8311	-2.8205	-2.8152	-2.8084	-2.84
naval	11934	16	9.2958	7.0464	9.3879	9.6962	6.8880	**9.8028**	6.6176	6.3520	6.7213	6.7632	3.73
protein	45730	9	-2.8997	-2.9128	**-2.7673**	-2.8444	-2.8554	-2.6613	-2.9183	-2.8384	-2.7888	-2.7969	-2.97

results compared to the sparse Gaussian Processes inference model. We observed that as the number of layers of deep hierarchy Gaussian process models increases, there is often better performance, especially for relatively larger data sets. On a relatively small data set like "Boston", "Wine", the advantage of the deep hierarchy Gaussian process model is not particularly obvious. Because the "Boston" dataset is very small, the wine data set is near-linear, "Naval" dataset is highly tested likehoods, which we can observe in the experiment. The Root Mean Square Error (RMSE) is almost close to 0 for all the models with the "Naval" dataset. These characteristics make the advantages of the deep Gaussian model not obvious. The more inducing points always lead to better performance. SGPR300, SVGP300, FITC300 always output better performance than SGPR100, SVGP100, FITC100, where the number refers to the number of inducing points used in the model. We believe this is due to more inducing points the model has richer feature representation space. At the same time, more inducing points also bring huge computation time. All the deep hierarchy Gaussian process models were given 100 inducing points. We also tested the deep hierarchy Gaussian process model with 500 inducing points. The computation time of a 4 layers DHGP model with 500 inducing points was prohibitive. However, we need to weigh the computational complexity and performance to choose the appropriate number of

TABLE 4.2: Comparison of DHGP model with other methods on regression task with eight UCI datasets. Test RMSE results, the number closer to 0 indicates better performance

	N	D	SGPR100	SVGP100	FITC100	SGPR300	SVGP300	FITC300	DHGP1	DHGP2	DHGP3	DHGP4	BNN(PBP)
boston	506	13	2.5356	2.5319	2.6098	**2.4338**	2.4352	2.6209	2.5346	2.5517	2.5557	2.5217	3.01
energy	768	8	0.6126	0.7134	0.4768	**0.4112**	0.4682	0.4758	0.6601	0.4440	0.5061	0.5323	1.80
concrete	1030	8	5.8921	5.9269	6.3538	5.7267	5.7036	6.4477	5.9080	5.9900	6.1017	6.1793	**5.67**
wine-red	1599	11	0.5934	0.5934	0.6096	0.5949	0.5947	0.6286	**0.5930**	0.5935	0.5947	0.5951	0.64
wine-white	4898	11	0.6773	0.6774	0.6889	**0.6734**	0.6734	0.6825	0.6785	0.6769	0.6769	0.6765	-
power	9568	4	4.0528	4.0534	4.0564	3.9913	3.9960	**3.9096**	4.1024	4.0557	4.0375	4.0063	4.12
naval	11934	16	**0.0000**	0.0002	0.0000	0.0000	0.0002	0.0000	0.0003	0.0003	0.0002	0.0002	0.01
protein	45730	9	4.3970	4.4505	4.4090	4.1629	4.2104	4.1375	4.4775	4.1459	**3.9626**	4.0009	4.73

inducing points to initialize the deep hierarchy Gaussian process model in future work.

MNIST Dataset for Multiclass Classification Task

We tested the deep hierarchy Gaussian process model on the MNIST dataset of handwritten digits, which is considered to be difficult for GPs based models before, because of its high input dimensions. We tested the deep hierarchy Gaussian process model using 1-5 and 10 layers, respectively. Each set of experiments was conducted 3 times with 100 inducing points, 300 inducing points, and 500 inducing points respectively. In order to observe the impact of the different inducing points on performance. We followed [141] to use the robust-max multiclass likelihood and with training/test split of 60**K**/10**K**. TABLE 4.3 shows the performance of the deep hierarchy Gaussian process model using 1-5 and 10 layers with 100, 300 and 500 inducing points respectively.

From TABLE 4.3, we can see that with the deep hierarchy Gaussian process model of 1-5 layers, as the number of layers increases, the accuracy keeps rising. We further tested on a 10 layer DHGP model. We found that, the accuracy does not increase but decreases. The same phenomenon also occurs in

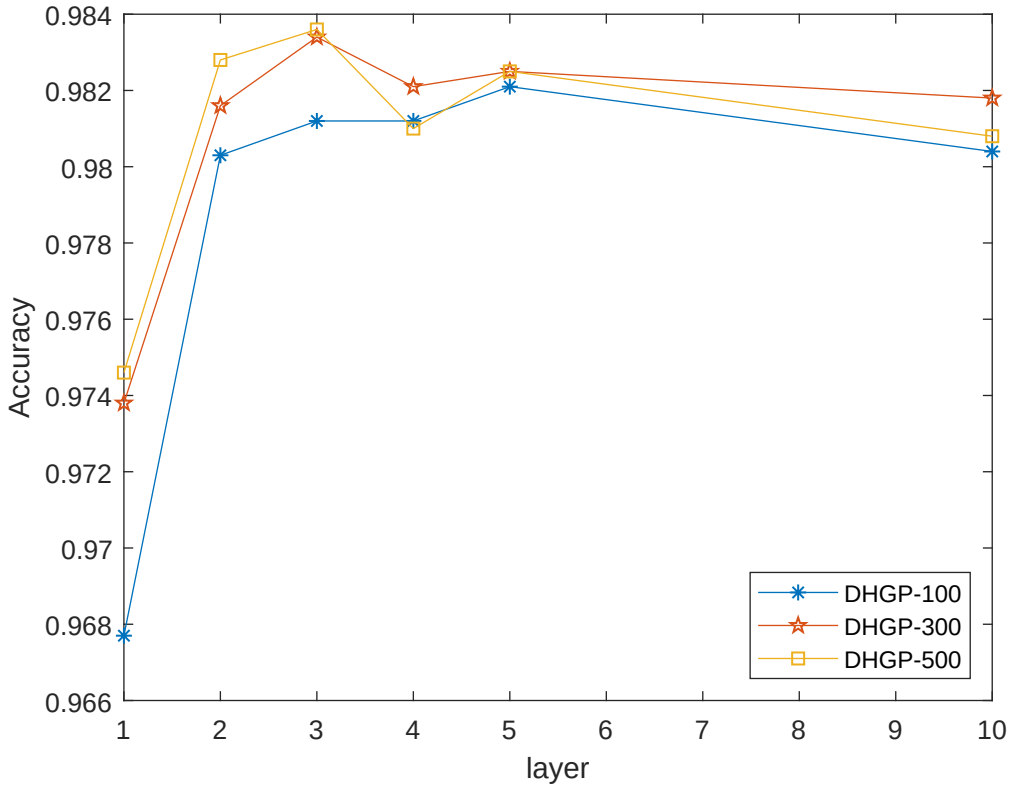

FIGURE 4.5: Accuracy of deep hierarchy Gaussian process model with a number of layer and inducing points

deep neural networks. Excessive layers have redundant information, which leads to the birth of the residual neural networks and also points the direction for our further work. The cost of computation increases dramatically as the number of layers increases. From FIGURE 4.5 we can see with more inducing points there is better performance. More inducing points make the feature space richer, which is better for better expression of features. There are very few GPs based models that can be compared, the previous single-layer 1000 inducing GPs model we found have achieved 98.1% [11] and 98.4% [142] respectively. We believe this is due to there are more inducing points, and the same phenomenon also appears in our experiments.

TABLE 4.3: MNIST tested results, deep hierarchy Gaussian process model 1-5 and 10 layers with 100, 300 and 500 inducing points respectively.

	100 inducing points			300 inducing points			500 inducing points		
	Running time	Accuracy	Test likelihood	Running time	Accuracy	Test likelihood	Running time	Accuracy	Test likelihood
DHGP1	0:29:43	0.9746	-0.0855	0:14:30	0.9738	-0.0929	0:29:43	0.9746	-0.0855
DHGP2	1:35:54	0.9828	-0.0601	0:40:15	0.9816	-0.0620	1:35:54	0.9828	-0.0601
DHGP3	2:41:52	0.9836	-0.0523	1:05:23	0.9834	-0.0562	2:41:52	0.9836	-0.0523
DHGP4	3:49:46	0.9810	-0.0599	1:30:35	0.9821	-0.0607	3:49:46	0.9810	-0.0599
DHGP5	4:54:34	0.9825	-0.0631	1:58:16	0.9825	-0.0621	4:54:34	0.9825	-0.0631
DHGP10	10:31:02	0.9808	-0.0673	4:05:29	0.9818	-0.0626	10:31:02	0.9808	-0.0673

4.7 Summary

This chapter compares the Gaussian process model with the restricted Boltzmann machine model, and the Gaussian process model has obvious advantages over the RBM model. Further the deep hierarchical Gaussian process model is introduced, followed by is sparse approximation and Variational Bayesian Inference optimization method for the deep hierarchical Gaussian process model, and finally is the experimental comparison results.

Chapter 5

Gaussian Processes with Convolutional Kernel

The convolutional structure is useful for image data input, this can be seen in the success of the convolutional neural network (CNN) in recent years. In this chapter we introduce the convolution structure into the GPs model, thereby enhancing the Gaussian process models non-local scalability, especially for image data.

5.1 Constructing Convolutional Kernel

Convolution is a mathematical operation on two functions that produce a third function expressing how the shape of one is modified by the other. In the image processing domain, convolution is the core of a convolutional neural network that takes a two-dimensional signal filtered to produce a new signal. e.g., a two-dimensional image x with width W and height H and a convolutional filter g, the convolutional operation is defined as:

$$(\mathrm{x} * \mathrm{g})[i,j] = \sum_{w=0}^{W-1} \sum_{h=0}^{H-1} \mathrm{x}[i+w, j+h]\mathrm{g}[w,h] \tag{5.1}$$

Where $\mathrm{x}[i,j] \in \mathbb{R}^2$ and g is the size of $\mathbb{R}^{H\times W}$ filter which slids over the

image to produce a new signal. Above is the way how convolution operation works in convolutional neural work. The convolutional kernel is constructed in a similar way of convolutional neural network, starting with a *patch response function* $g(\cdot) : \mathbb{R}^E \to \mathbb{R}$,which maps an image patch to a real number. The *patch response function* $g(\cdot)$ is non-linear and nonparametric, compare with CNN convolutional gaussian process, which does not need the linear connection followed by an activation function to achieve nonlinearly. For patches of size $E = w \times h$ and image of size $D = W \times H$, we have a total of $P = (W - w + 1) \times (H - h + 1)$ patches. Sum all the patch response together we get $f : \mathbb{R}^D \to \mathbb{R}$, a pictorial interpretation is shown in FIGURE 5.1 Given $g(\cdot)$ a GP prior, $f(\cdot)$ is consistent with a joint Gaussian distribution due to the elegant property of Gaussian process: Linear transformation of Gaussian is still Gaussian.

$$\begin{aligned} g \sim \mathcal{GP}\left(0, k_g\left(\mathbf{z}, \mathbf{z}'\right)\right), \quad f(\mathbf{x}) = \sum_p g\left(\mathbf{x}^{[p]}\right) \\ \Longrightarrow f \sim \mathcal{GP}\left(0, \sum_{p=1}^{P} \sum_{p'=1}^{P} k_g\left(\mathbf{x}^{[p]}, \mathbf{x}'^{[p']}\right)\right) \end{aligned} \tag{5.2}$$

The $\mathbf{x}^{[p]}$ refers to the p^{th} patch in the input image. The same *patch response function* $g(\cdot)$ is applied to all patches, the kernel function $k_g(\cdot, \cdot)$ measures the distance between patches, it means that the kernel function will output similar patches response for similar patches.

5.2 Inter-domain Inducing Patches

In the previous section we have introduced the inducing points used in sparse gaussian process inference, which avoids the $\mathcal{O}\left(N^3\right)$ training time with dataset

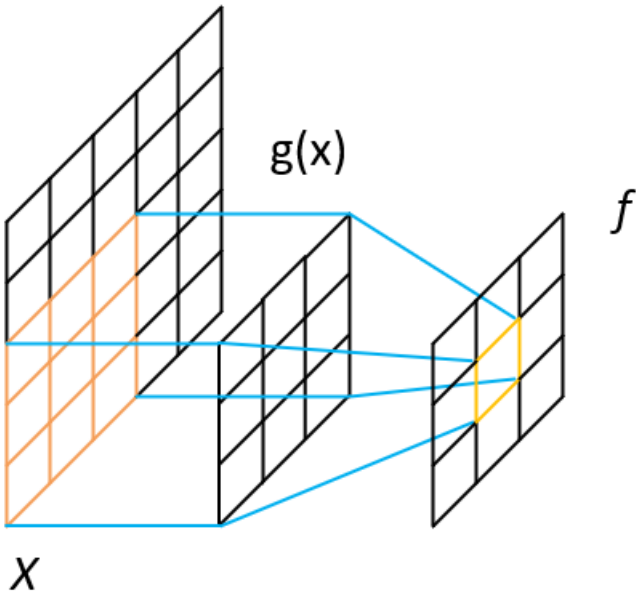

FIGURE 5.1: A pictorial interpretation of how a convolutional kernel works

size N reducing the computational cost to $\mathcal{O}\left(NM^2\right)$ with $M \ll N$, the number of inducing points [14]. The inducing points perform inference across domain, this makes the model more scalable and can encode more prior knowledge, making the data feature space more expressive. This has proven to be effective in Chapter 4 experiments. Consider a real-valued GP function $f(x)$ and a *feature extraction function* $h(x,z)$, the *feature extraction function* $h(x,z)$ projects $f(x)$ to other domain leading to inter-domain inference and z is the *inducing feature*. The inter-domain inducing variable u is derived after some linear transformation.

$$u_m = \int \mathrm{h}\left(x, z_m\right) f(x) \mathrm{d}x \tag{5.3}$$

The covariance between inducing variable u_m and $f(x_n)$ is then

$$\begin{aligned} \operatorname{cov}\left(u_m, f\left(\mathbf{x}_n\right)\right) &= \mathbb{E}\left[u_m f\left(\mathbf{x}_n\right)\right] = \mathbb{E}\left[f\left(\mathbf{x}_n\right) \int h\left(\mathbf{x}, \mathbf{z}_m\right) f(\mathbf{x}) \mathrm{d}\mathbf{x}\right] \\ &= \int h\left(\mathbf{x}, \mathbf{z}_m\right) k\left(\mathbf{x}, \mathbf{x}_n\right) \mathrm{d}\mathbf{x} \end{aligned} \tag{5.4}$$

The covariance between two inducing variables is as follows

$$\begin{aligned}\operatorname{cov}\left(u_m, u_{m'}\right) &= \mathbb{E}\left[u_m u_{m'}\right] = \mathbb{E}\left[\int f(\mathbf{x}) h\left(\mathbf{x}, \mathbf{z}_m\right) d\mathbf{x} \int f\left(\mathbf{x}'\right) h\left(\mathbf{x}', \mathbf{z}_{m'}\right) d\mathbf{x}'\right] \\ &= \iint h\left(\mathbf{x}, \mathbf{z}_m\right) h\left(\mathbf{x}', \mathbf{z}_{m'}\right) k\left(\mathbf{x}, \mathbf{x}'\right) \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{x}'\end{aligned} \tag{5.5}$$

Inter-domain inducing patch as the promotion of inducing points, the inducing inference takes place in the input of *patches* instead of *points*. An illustrative example from [140], consider a GP prior $f(x)$ the model is written as

$$f(\cdot)|\theta \sim \mathcal{GP}(0, k(\cdot, \cdot)) \tag{5.6}$$

Where θ summaries all the parameters, we introduce *inducing feature* $\mathbf{Z} = \{\mathbf{Z}_m\}_{m=1}^M$ and the inducing variable is constructed as $\mathbf{u} = \{h\left(\mathbf{z}_m\right)\}_{m=1}^M$, $h(\cdot)$ is a kind of linear transformation to ensure inducing variable $\mathbf{u}$ and $f(x_n)$ follow normal distribution. Then $f(\cdot)$ conditional on the inducing variable $\mathbf{u}$ is as follows

$$p(\begin{bmatrix} \mathrm{f} \\ \mathrm{u} \end{bmatrix} |\mathbf{X}, \mathbf{Z}) \sim gp(mean \begin{bmatrix} \mathrm{f} \\ \mathrm{u} \end{bmatrix}, cov(\begin{bmatrix} \mathbf{K}_{\mathrm{ff}} & \mathbf{K}_{\mathrm{fu}} \\ \mathbf{K}_{\mathrm{uf}} & \mathbf{K}_{\mathrm{uu}} \end{bmatrix})) \tag{5.7}$$

what we need to do is to find the appropriate cross-domain covariance $\mathbf{K}_{\mathrm{fu}}$ and $\mathbf{K}_{\mathrm{uu}}$ in the patches space. From the convolutional kernel construction function, the cross-domain covariance $\mathbf{K}_{\mathrm{fu}}$ can be found as

$$k_{fu}(\mathrm{x}, \mathrm{z}) = \mathbb{E}_g[f(\mathrm{x})\mathbf{u}(\mathrm{z})] = \mathbb{E}_g\left[\sum_p g\left(\mathrm{x}^{[p]}\right)\mathbf{u}(\mathrm{z})\right] = \sum_p k_g\left(\mathrm{x}^{[p]}, \mathrm{z}\right) \tag{5.8}$$

And the covariance between inducing patches $\mathbf{K}_{\mathrm{uu}}$ is

$$k_{uu}\left(\mathbf{z}, \mathbf{z}'\right) = \mathbb{E}_g\left[\mathbf{u}(\mathbf{z})\mathbf{u}\left(\mathbf{z}'\right)\right] = k_g\left(\mathbf{z}, \mathbf{z}'\right) \tag{5.9}$$

By using inducing patches inference, the computation cost can be effectively reduced. While the computational cost seems to be linear in N the number of training data, by using minibatch optimization N can be small.

5.3 Computational Issues with Gaussian Process Model

The problem that hinders the GP model from being applied to practical applications is the computational issues. The most important parameter of the Gaussian Process model is the covariance matrix. There are many computationally expensive inversions operations when constructing the covariance matrix. Exact computations for large data generally require $\mathcal{O}\left(N^3\right)$ computation and $\mathcal{O}\left(N^2\right)$ memory, where N is the number of training data samples. In recent years, many efforts have been made to explore sparse Gaussian Process inferencing. Including earlier sparse Gaussian process inference frameworks like SGPR [138], FITC[97]. By introducing inducing points, they avoid $\mathcal{O}\left(N^3\right)$ computation time scaling with data size N and reduce the computational cost to $\mathcal{O}\left(NM^2\right)$ with $M \ll N$, which is the number of *inducing variables*. The Inter-domain GPs [15] can be used to find a (possibly more compact) representative set of features lying in a different domain, at the same computational cost. The work [14] pointed out the true complexity of the algorithm depends on how M must increase to ensure a certain quality of the approximation. For a particular case, in a D-dimensional input regression task with Squared Exponential kernel, $M = \mathcal{O}\left(\log^D N\right)$ is sufficient. They proved a kind of relationship between M and N. (For a more detailed, please refer to [14])

5.4 Variant of Convolutional Kernel

Mentioned above from Eq. 5.2 for the construction of convolutional kernel, the same *patch response function* $g(\cdot)$ is applied to all patches in the image regardless of the type and location of the image. Image data often contains many different patterns. Depending on the task, a strict form of invariance may or may not be beneficial [143]. [103] introduced a series of variant convolution kernels for different tasks. First variant is the weighted convolutional kernel, the same *patch response function* $g(\cdot)$ is used in all the patches is too strong a constraint. Because the same features can be different categories of things in different areas of the image. Give different weights for the *patch response function* from different patches. Denote the *patch response function* $g(\cdot)$ again, the sum of $g(\cdot)$ $f(x)$ can be constructed as

$$f(\mathbf{x}) = \sum_{p} w_p g\left(\mathbf{x}^{[p]}\right) \tag{5.10}$$

The weights $\left\{w_p\right\}_{p=1}^{P}$ indicates that different patches have different weights. And the covariance between observed sample $\mathbf{K}_{\text{ff}}$ and cross-domain $\mathbf{K}_{\text{fu}}$ can be found via a minor modification from the invariant case

$$k_{ff}(\mathbf{x}, \mathbf{x}) = \sum_{pq} w_p w_q k_g\left(\mathbf{x}^{[p]}, \mathbf{x}_q\right) \tag{5.11}$$

$$k_{fu}(\mathrm{x}, \mathrm{z}) = \sum_{p} w_p k_g\left(\mathrm{x}^{[p]}, \mathrm{z}\right) \tag{5.12}$$

This introduces more parameters the patch weight w is considered as kernel hyperparameters and can be optimized in the same manner as other kernel parameters. We will verify the validity of this weighted convolutional kernel in the next section. Our experiment follows the same optimization method as [103] The second variant of the convolutional kernel is designed to handle the multi-channel image. Multi-channel images bring richer image features,

while also increasing the image input dimension, requiring more efficient feature processing approach to handle redundant information. One conceivable way is to apply the same response function to all image channels. This will produce C times more patch responses. [103] proposed a middle ground convolutional kernel construction approach that does not increasing input dimension so much, it is apply different *patch response function* $g_c(\cdot)$ for different channel refer it as the *multi-channel* convolutional kernel. The sum of the patch response function $f(x)$ is constructed as follows

$$f(\mathrm{x}) = \sum_{p=1}^{P} \sum_{c=1}^{C} w_{pc} g_c \left(\mathrm{x}^{[pc]} \right) \tag{5.13}$$

The inducing patches are shared among all channels. Sum each patch response function $g_c \left(\mathrm{x}^{[pc]} \right)$ together we get the image-based response function $f(x)|u(z)$. The function $f(x)|u(z)$ is governed by the covariance $\mathbf{K}_{\text{fu}}$ and $\mathbf{K}_{\text{uu}}$. As we have N observed points and M inducing patches the covariance matrix $\mathbf{K}_{\text{fu}}$ and $\mathbf{K}_{\text{uu}}$ is of size $N \times MC$ and $MC \times MC$ respectively, where C is the number of channels. The covariance between observed points and inducing patches is given by

$$k_{fg_c}(\mathbf{x}, \mathbf{z}) = \mathbb{E}_{\{g_c\}_{c=1}^{C}} \left[\sum_p w_{pc} g_c \left(\mathbf{x}^{[pc]} \right) g_c(\mathbf{z}) \right] = \sum_p w_{pc} k_g \left(\mathbf{x}^{[pc]}, \mathbf{z} \right) \tag{5.14}$$

And the covariance between inducing patches is as follows

$$k_{uu} \left(\mathbf{z}, \mathbf{z}' \right) = \mathbb{E}_g \left[g(\mathbf{z}) g \left(\mathbf{z}' \right) \right] = k_g \left(\mathbf{z}, \mathbf{z}' \right) \tag{5.15}$$

In the next section, we will verify the performance and benefits of convolution kernels through experiments.

5.5 Experiments

To test the power of the new convolution kernel, we tested the performance of convolutional kernels on three featured image datasets MNIST, Rectangle and colored image datasets Cifar-10 respectively. All of these datasets are considered to be difficult for kernel-function based methods because of the large computational complexity caused by high dimensional input and the need to identify different patterns of multipixel formation.

We first tested Gaussian Process with a convolutional kernel on the MNIST dataset. MNIST is a handwritten digit dataset for recognition and is widely used in the computer vision field. Because of the standard data folds, the performance-tested can be directly compared in a variety of methods. We set up a Gaussian process with the convolutional kernel to multi-class classification task with 10 output corresponding to 10 different classes. We trained the Gaussian Process with the convolutional kernel using Adam [144] with different inducing points. There are few existing works using Gaussian-based kernel methods for image processing. We compared the previous start-of-the-arts Gaussian process methods and also deep hierarchy Gaussian process model. As can be seen in TABLE 5.1, the translation invariance convolutional kernel underperforms compared with the previous reported state-of-the-art 98.4% [142]. This could be caused by applying the same *patch response function* to all the patches regardless of the location and its adjacent area. By introducing weights to the convolutional kernel, adjacent pixels that may form a pattern are more likely to be detected, thereby improving the detection performance. We compared with a 4-layer deep hierarchy Gaussian process model with 500 inducing points, a single layer Gaussian Process with the convolutional kernel can outperform a 4-layer DHGP model. This shows that the convolutional kernel is effective for processing image data.

We further tested the convolutional kernel Gaussian process on Cifar-10

TABLE 5.1: Test accuracy and nlpp(negative log predictive probability) comparsion of GPs model with different kernels with Neural network and other GPs based models on MNIST and Cifar-10

		Inducing	Test Accuracy		Test nlpp	
Gaussian process models	Layers	points	**MNIST**	**CIFAR-10**	**MNIST**	**CIFAR-10**
RBF AutoGP [142]	1	200	98.4%	55.05%	N/A	N/A
conv GP	1	750	97.89%	N/A	0.0792	N/A
Wconv GP	1	750	98.77%	56.88%	2.3305	2.3305
Multi-channel conv GP	1	1000	N/A	64.6%	N/A	2.0691
DHGP4	4	500	98.10%	N/A	N/A	N/A
Neural network models	Layers	#params				
Deep kernel learning [145]	5	$2.3M \cdots 4.6M$	99.2%	77.0%	N/A	N/A
ResNet-20 [44]	20	0.27M	N/A	91.25%	N/A	N/A
ResNet-56 [44]	56	0.85M	N/A	93.03%	N/A	N/A
ResNet-110 [44]	110	1.7M	N/A	93.57%	N/A	N/A
DenseNet [146]	100	27.2M	N/A	94.17%	N/A	N/A
Giant Neural Networks [147]	N/A	N/A	N/A	99.00%	N/A	N/A

[148] dataset. The Cifar-10 dataset is a collection of colored images that are commonly used to train machine learning algorithms, it contains 60000 of 32×32 low-resolution image in 10 classes. We used the same setup used in MNIST. As can be seen in TABLE 5.1. The Gaussian method using a weighted convolution kernel successfully increased the accuracy of the previous reported 55.05% [142] to 56.88%, which was further increased to 64.6% by applying a multi-channel convolution kernel. However, this is still far underperformed compared with the current state-of-the-art neural networks like ResNet [44], DenseNet [146] and Giant neural networks [147] which has reduced the error rate to 1%. Compared to neural networks Gaussian process model has far fewer parameters and can utilise the marginal likelihood to find the optimal parameter. With the increase in the number of network layers and the development of more skills, it is believed that the Gaussian model can release greater potential. The visualisation of inducing patches used in inference and error rate optimization is shown with FIGURE 5.2 and

FIGURE 5.3 respectively.

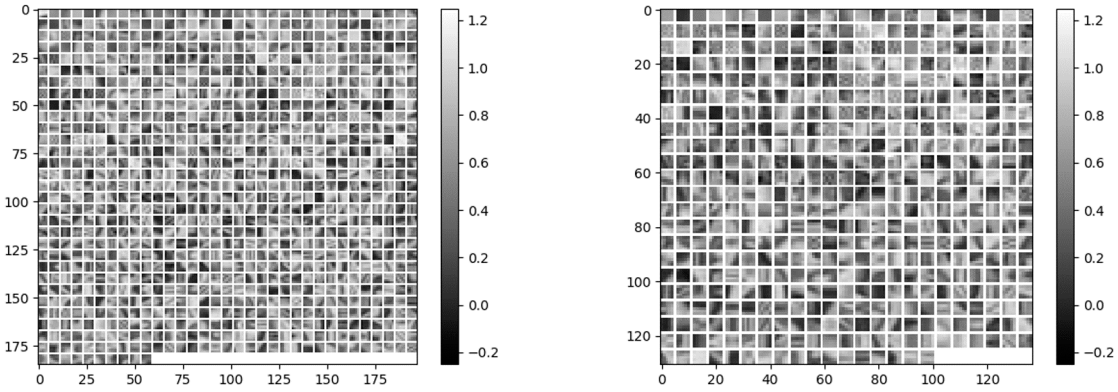

FIGURE 5.2: Visualisation of inducing patches used in Gaussian process with weighted convolutional kernel (left) and multi-channel convolutional kernel (right) test on CIFAR-10 dataset.

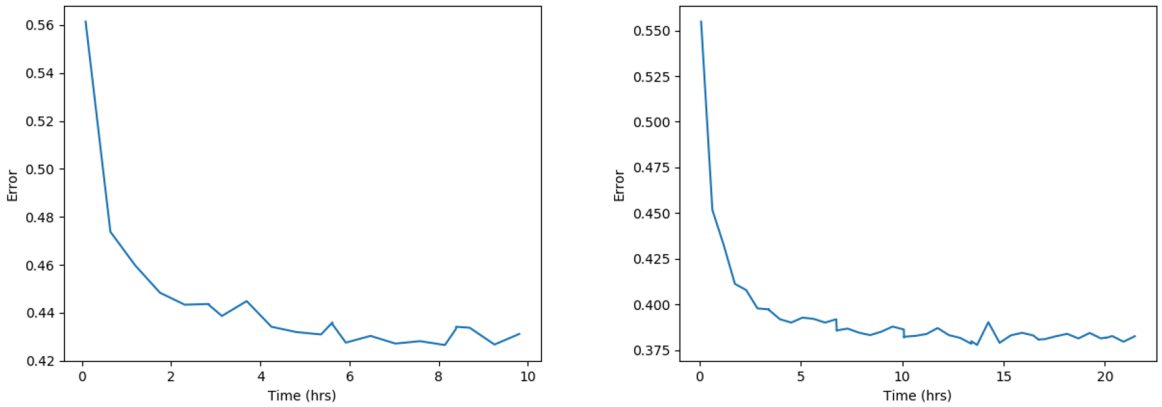

FIGURE 5.3: Visualisation of error rate optimization in Gaussian Process with weighted convolutional kernel(left) and multi-channel convolutional kernel(right) test on CIFAR-10 dataset.

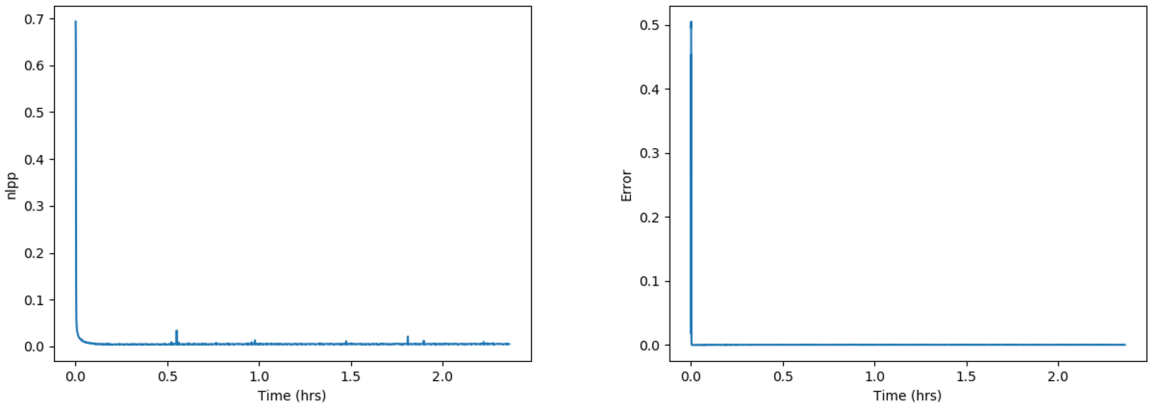

FIGURE 5.4: Visualisation of error rate and nlpp optimization in Gaussian Process with weighted convolutional kernel on Rectangles dataset.

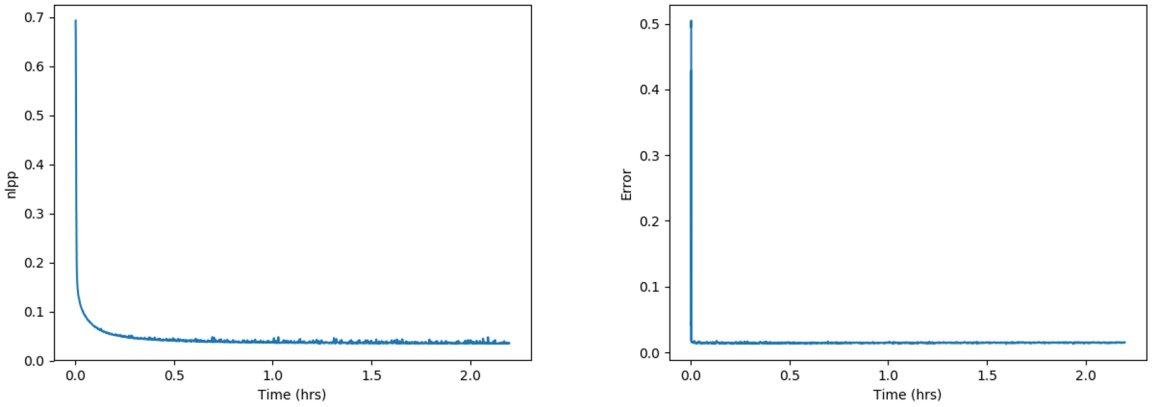

FIGURE 5.5: Visualisation of error rate and nlpp optimization in Gaussian Process with translation invariance convolutional kernel on Rectangles dataset.

We did another test on the Rectangles dataset, which is an artificial dataset composed of 1200 of 28×28 low-resolution image with randomly generated rectangles. The goal is to distinguish whether a rectangle is a long rectangle with a larger length or wide rectangle with a large width. We used the same setup as used in MNIST, the Gaussian process with convolutional kernel give 98.67% error rate and 0.039 nlpp (negative log predictive probability). This has a large performance boost compared with the Gaussian Process with RBF kernel which gives 95% error rate and 0.2581 nlpp. We attribute the improvement of performance to the introduction of convolution kernels. As the convolution kernel can detect patterns consisting of pixels in adjacent regions, which is also one of the key features for the success of CNN in recent years. The performance improvement obtained by introducing the convolution kernel into the Gaussian model is also expected. The weighted convolutional kernel further reduces the error rate to 0.0005, the weighted convolution kernel can detect patterns of different scales composed of adjacent pixels, which is more flexible and brings further performance improvement. The visualisation of error rate and nlpp optimization of weighted convolutional kernel and translation invariance convolutional kernel are shown in FIGURE 5.4 and FIGURE 5.5 respectively.

5.6 Summary

This chapter introduces convolution operation into the Gaussian process model, and introduces several variants of convolutional kernels, and finally the experimental comparison results.

Chapter 6

Conclusion and Future Work

This section summarises the work of the thesis and gives directions for future development.

6.1 Conclusion

In this thesis, we have introduced a semi-supervised tracking algorithm with a new observation model, which adopts graph Laplacian. Furthermore, the prior gram matrix is constructed based on all samples. In this way, future information has a strong influence on the tracking decision and can be viewed as a transfer learning strategy. We devise multiple-person tracking by using a tracker hierarchy. Trackers are classified into two groups based on the template they owe and the different types of trackers adopt different update strategies during the tracking process. The new model can effectively deal with various problems in the process of multiple-person tracking from multiple aspects, greatly improving the performance of the algorithm.

Furthermore, we explore the possibility of deep models other than deep neural networks and propose a deep hierarchy Gaussian process model based on the building block of GPs. This is a kind of statistical probability model that not only has a high model complexity similar to neural networks, which can extract different levels of abstract features, but also is capable of causal inference and fit uncertainty that neural networks lack. The deep hierarchy

Gaussian process model was optimised by means of variational inferencing, which uses a variational distribution to approximate the true posterior distribution. We further verified that the inducing points can not only accelerate the model inference speed but also improve the model performance. Since the convolutional kernel is very useful for image tasks, we introduced the convolutional kernel into the GPs model to construct the convolutional Gaussian processes model which is more suitable for image processing tasks. We further verified that inducing image patches can speed up model inference and improve model performance. Through experimental comparisons, the convolutional Gaussian process model significantly improves performance for image processing tasks compared to non-convolutional Gaussian processes. The main limitation of the Gaussian process model is the computation cost, because there are many expensive matrix inversion operations when constructing the covariance matrix. There have been many studies focusing on the sparse approximation problem of Gaussian process model in order to improve the inference speed of the model.

6.2 Future Work

Because an infinitely wide single-layer neural network is equivalent to GPs, the model complexity of a deep hierarchy Gaussian process model is much higher than that of a neural network at the same layer. Current deep neural networks still have problems such as lack of causal reasoning capabilities and poor interpretability, these are the advantages of statistical models. During our experiments, we also found the same phenomenon as found in deep neural networks. That is, by increasing the number of network layers does not necessarily lead to performance improvement. Sometimes it will lead to performance degradation, which is the reason for the birth of residual networks. It is believed that applying the same ideas to the deep hierarchy Gaussian process model will also bring performance improvement. The deep model based on Gaussian process is still in its infancy and has not been thoroughly explored. A translation invariance kernel [143] is proposed, and the experiments show that the deep model with Gaussian process building block already surpasses simple deep neural networks in terms of performance on some specific tasks, and has a good ability to estimate uncertainty. However, a large amount of computation cost is a big problem that hinders the development of Gaussian process models. Many companies are now developing dedicated artificial intelligence chips, and computing power continues to show exponential growth. It is believed that with the increase in computing power in the near future, this will not be a big problem. The deep model based on the Gaussian process is far from mature and needs further exploration to release its powerful potential. Traditional machine learning models have clear theory foundations, good interpretability, and causal inferencing capabilities. Deep neural networks are highly complex and can automatically extract features. A potentially feasible solution is to replace the special layer of the deep neural network with a Gaussian process model to study the

interpretability of the neural network from a statistical point of view.

Because a single-layer infinite-wide neural network is equivalent to GPs, a deep hierarchy Gaussian process model is equivalent to a deep infinite-wide deep neural networks. The capacity of the deep hierarchy Gaussian process model is much higher than that of ordinary neural networks. Because neural networks have various operation techniques, such as Dropout, Convolution operation, etc., and various optimization techniques. The deep hierarchy Gaussian process model lacks corresponding operations and optimization skills, and its potential is far from being released. It is believed that investing more manpower and resources to develop the deep hierarchy Gaussian process model will further release its potential. The deep hierarchy Gaussian process model is a statistical probability model, which has both the characteristics of causal inference and the high complexity of neural networks, which can extract abstract features layer by layer. It is believed that combining the advantages of traditional models and neural networks will be the development direction of future machine learning models.

References

[1] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim, "Multiple object tracking: A literature review", *arXiv preprint arXiv:1409.7618*, 2014.

[2] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric", in *2017 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2017, pp. 3645–3649.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking", in *2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2016, pp. 3464–3468.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai", *Information Fusion*, vol. 58, pp. 82–115, 2020.

[6] Z. Zhou and J Feng, "Deep forest: Towards an alternative to deep neural networks. arxiv 2017", *arXiv preprint arXiv:1702.08835*,

[7] Z.-H. Zhou and J. Feng, "Deep forest", *arXiv preprint arXiv:1702.08835*, 2017.

[8] J. Feng, Y. Yu, and Z.-H. Zhou, "Multi-layered gradient boosting decision trees", in *Advances in neural information processing systems*, 2018, pp. 3551–3561.

[9] M. N. Gibbs, "Bayesian gaussian processes for regression and classification", PhD thesis, Citeseer, 1998.

[10] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, 3. MIT press Cambridge, MA, 2006, vol. 2.

[11] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data", *arXiv preprint arXiv:1309.6835*, 2013.

[12] G. Manogaran and D. Lopez, "A gaussian process based big data processing framework in cluster computing environment", *Cluster Computing*, vol. 21, no. 1, pp. 189–204, 2018.

[13] A. Y. Sun, D. Wang, and X. Xu, "Monthly streamflow forecasting using gaussian process regression", *Journal of Hydrology*, vol. 511, pp. 72–81, 2014.

[14] D. R. Burt, C. E. Rasmussen, and M. Van Der Wilk, "Rates of convergence for sparse variational gaussian process regression", *arXiv preprint arXiv:1903.03571*, 2019.

[15] M. Lázaro-Gredilla and A. Figueiras-Vidal, "Inter-domain gaussian processes for sparse inference using inducing features", in *Advances in Neural Information Processing Systems*, 2009, pp. 1087–1095.

[16] C. Walder, K. I. Kim, and B. Schölkopf, "Sparse multiscale gaussian process regression", in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 1112–1119.

[17] L. Csató and M. Opper, "Sparse on-line gaussian processes", *Neural computation*, vol. 14, no. 3, pp. 641–668, 2002.

[18] J. QuiÃ±onero-Candela, C. E. Rasmussen, A. R. Figueiras-Vidal, *et al.*, "Sparse spectrum gaussian process regression", *Journal of Machine Learning Research*, vol. 11, no. Jun, pp. 1865–1881, 2010.

[19] M. Bauer, M. van der Wilk, and C. E. Rasmussen, "Understanding probabilistic sparse gaussian process approximations", in *Advances in neural information processing systems*, 2016, pp. 1533–1541.

[20] A. Damianou and N. Lawrence, "Deep gaussian processes", in *Artificial Intelligence and Statistics*, 2013, pp. 207–215.

[21] H. Salimbeni and M. Deisenroth, "Doubly stochastic variational inference for deep gaussian processes", in *Advances in Neural Information Processing Systems*, 2017, pp. 4588–4599.

[22] T. Bui, D. Hernández-Lobato, J. Hernandez-Lobato, Y. Li, and R. Turner, "Deep gaussian processes for regression using approximate expectation propagation", in *International Conference on Machine Learning*, 2016, pp. 1472–1481.

[23] Y. Wang, M. Brubaker, B. Chaib-Draa, and R. Urtasun, "Sequential inference for deep gaussian process", in *Artificial Intelligence and Statistics*, 2016, pp. 694–703.

[24] M. Havasi, J. M. Hernández-Lobato, and J. J. Murillo-Fuentes, "Inference in deep gaussian processes using stochastic gradient hamiltonian monte carlo", in *Advances in Neural Information Processing Systems*, 2018, pp. 7506–7516.

[25] K. Vafa, "Training deep gaussian processes with sampling", in *NIPS 2016 Workshop on Advances in Approximate Bayesian Inference*, 2016.

[26] D. J. MacKay, "A practical bayesian framework for backpropagation networks", *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[27] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[28] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes", *arXiv preprint arXiv:1711.00165*, 2017.

[29] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning", in *Artificial Intelligence and Statistics*, 2016, pp. 370–378.

[30] A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison, "Deep convolutional networks as shallow gaussian processes", *arXiv preprint arXiv:1808.05587*, 2018.

[31] R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Bayesian deep convolutional networks with many channels are gaussian processes", 2018.

[32] S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov, and R. Wang, "On exact computation with an infinitely wide neural net", *arXiv preprint arXiv:1904.11955*, 2019.

[33] S. Waugh, "Extending and benchmarking cascade-correlation", *Dept of Computer Science, University of Tasmania, Ph. D. Dissertation*, 1995.

[34] W. J. Nash, T. L. Sellers, S. R. Talbot, A. J. Cawthorn, and W. B. Ford, "The population biology of abalone (haliotis species) in tasmania. i. blacklip abalone (h. rubra) from the north coast and islands of bass strait", *Sea Fisheries Division, Technical Report*, vol. 48, 1994.

[35] A Frank, "Uci machine learning repository. irvine, ca: University of california, school of information and computer science", *http://archive. ics. uci. edu/ml*, 2010.

[36] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, and L. E. Meester, *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media, 2005.

[37] R. Durrett, *Probability: theory and examples*. Cambridge university press, 2019, vol. 49.

[38] Y. Filmus, "Two proofs of the central limit theorem", *Recuperado de http://www. cs. toronto. edu/yuvalf/CLT. pdf*, 2010.

[39] M. G. Genton, "Classes of kernels for machine learning: A statistics perspective", *Journal of machine learning research*, vol. 2, no. Dec, pp. 299–312, 2001.

[40] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning", *The annals of statistics*, pp. 1171–1220, 2008.

[41] H.-T. Lin and C.-J. Lin, "A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods", *submitted to Neural Computation*, vol. 3, pp. 1–32, 2003.

[42] R. Herbrich, *Learning kernel classifiers: theory and algorithms*. MIT press, 2001.

[43] A. G. Wilson, "Covariance kernels for fast automatic pattern discovery and extrapolation with gaussian processes", PhD thesis, University of Cambridge, 2014.

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

[45] Y. Cong, J. Yuan, and J. Liu, "Sparse reconstruction cost for abnormal event detection", in *CVPR 2011*, IEEE, 2011, pp. 3449–3456.

[46] ——, "Abnormal event detection in crowded scenes using sparse representation", *Pattern Recognition*, vol. 46, no. 7, pp. 1851–1864, 2013.

[47] S. Zheng, J. Zhang, K. Huang, R. He, and T. Tan, "Robust view transformation model for gait recognition", in *2011 18th IEEE International Conference on Image Processing*, IEEE, 2011, pp. 2073–2076.

[48] J. Zhang, J. Pu, C. Chen, and R. Fleischer, "Low-resolution gait recognition", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 4, pp. 986–996, 2010.

[49] Z. Wu, Y. Huang, L. Wang, X. Wang, and T. Tan, "A comprehensive study on cross-view gait based human identification with deep cnns", *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 2, pp. 209–226, 2016.

[50] T. Tan, L. Wang, Y. Huang, and W. Zifeng, *Gait recognition method based on deep learning*, US Patent App. 10/223,582, 2019.

[51] C. Wang, H. Zhang, L. Yang, S. Liu, and X. Cao, "Deep people counting in extremely dense crowds", in *Proceedings of the 23rd ACM international conference on Multimedia*, ACM, 2015, pp. 1299–1302.

[52] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2547–2554.

[53] M. Marsden, K. McGuinness, S. Little, and N. E. O'Connor, "Resnetcrowd: A residual deep learning architecture for crowd counting, violent behaviour detection and crowd density level classification", in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, IEEE, 2017, pp. 1–7.

[54] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 152–159.

[55] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep metric learning for person re-identification", in *2014 22nd International Conference on Pattern Recognition*, IEEE, 2014, pp. 34–39.

[56] E. Ahmed, M. Jones, and T. K. Marks, "An improved deep learning architecture for person re-identification", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3908–3916.

[57] G. Levi and T. Hassner, "Age and gender classification using convolutional neural networks", in *Proceedings of the iEEE conference on computer vision and pattern recognition workshops*, 2015, pp. 34–42.

[58] M. Shoaib, R. Dragon, and J. Ostermann, "View-invariant fall detection for elderly in real home environment", in *2010 Fourth Pacific-Rim Symposium on Image and Video Technology*, IEEE, 2010, pp. 52–57.

[59] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction", in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, IEEE, vol. 2, 2004, pp. 28–31.

[60] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction", in *European conference on computer vision*, Springer, 2000, pp. 751–767.

[61] G. Tesauro, "Temporal difference learning and td-gammon", *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[62] T. Ko, S. Soatto, and D. Estrin, "Warping background subtraction", in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 2010, pp. 1331–1338.

[63] Y.-T. Chen, C.-S. Chen, C.-R. Huang, and Y.-P. Hung, "Efficient hierarchical method for background subtraction", *Pattern Recognition*, vol. 40, no. 10, pp. 2706–2715, 2007.

[64] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks", *CoRR*, vol. abs/1504.06852, 2015. arXiv: 1504.06852. [Online]. Available: http://arxiv.org/abs/1504.06852.

[65] Y. Tian, R. Sukthankar, and M. Shah, "Spatiotemporal deformable part models for action detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2642–2649.

[66] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", 2001, pp. 511–518.

[67] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", 2005.

[68] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[69] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features", in *European conference on computer vision*, Springer, 2006, pp. 404–417.

[70] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn", in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[71] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks", *arXiv preprint arXiv:1312.6229*, 2013.

[72] R. Girshick, "Fast r-cnn", in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[73] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector", in *European conference on computer vision*, Springer, 2016, pp. 21–37.

[74] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[75] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[76] ——, "Yolov3: An incremental improvement", *arXiv preprint arXiv:1804.02767*, 2018.

[77] R. J. Wang, X. Li, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices", in *Advances in Neural Information Processing Systems*, 2018, pp. 1963–1972.

[78] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", in *Advances in neural information processing systems*, 2015, pp. 91–99.

[79] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[80] F. Chollet, "Xception: Deep learning with depthwise separable convolutions", *CoRR*, vol. abs/1610.02357, 2016. arXiv: 1610.02357. [Online]. Available: http://arxiv.org/abs/1610.02357.

[81] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications", *CoRR*, vol. abs/1704.04861,

2017. arXiv: 1704.04861. [Online]. Available: http://arxiv.org/abs/1704.04861.

[82] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[83] J. Black, T. Ellis, and P. Rosin, "Multi view image surveillance and tracking", in *Workshop on Motion and Video Computing, 2002. Proceedings.*, IEEE, 2002, pp. 169–174.

[84] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Robust tracking-by-detection using a detector confidence particle filter", in *2009 IEEE 12th International Conference on Computer Vision*, IEEE, 2009, pp. 1515–1522.

[85] X. Song, J. Cui, H. Zha, and H. Zhao, "Vision-based multiple interacting targets tracking via on-line supervised learning", in *European Conference on Computer Vision*, Springer, 2008, pp. 642–655.

[86] J. Berclaz, F. Fleuret, and P. Fua, "Robust people tracking with global trajectory optimization", in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, IEEE, vol. 1, 2006, pp. 744–750.

[87] C. Huang, B. Wu, and R. Nevatia, "Robust object tracking by hierarchical association of detection responses", in *European Conference on Computer Vision*, Springer, 2008, pp. 788–801.

[88] S. Tang, B. Andres, M. Andriluka, and B. Schiele, "Multi-person tracking by multicut and deep matching", in *European Conference on Computer Vision*, Springer, 2016, pp. 100–111.

[89] E. Insafutdinov, M. Andriluka, L. Pishchulin, S. Tang, E. Levinkov, B. Andres, and B. Schiele, "Arttrack: Articulated multi-person tracking in the wild", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6457–6465.

[90] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift", in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, IEEE, vol. 2, 2000, pp. 142–149.

[91] S. Avidan, "Ensemble tracking", *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 2, pp. 261–271, 2007.

[92] B. Benfold and I. Reid, "Stable multi-target tracking in real-time surveillance video", in *CVPR 2011*, IEEE, 2011, pp. 3457–3464.

[93] A. Blake and M. Isard, "The condensation algorithm-conditional density propagation and applications to visual tracking", in *Advances in Neural Information Processing Systems*, 1997, pp. 361–367.

[94] J. Munkres, "Algorithms for the assignment and transportation problems", *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[95] A. Damianou, M. K. Titsias, and N. D. Lawrence, "Variational gaussian process dynamical systems", in *Advances in Neural Information Processing Systems*, 2011, pp. 2510–2518.

[96] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks", in *International Conference on Machine Learning*, 2015, pp. 1861–1869.

[97] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs", in *Advances in neural information processing systems*, 2006, pp. 1257–1264.

[98] A. Naish-Guzman and S. Holden, "The generalized fitc approximation", in *Advances in Neural Information Processing Systems*, 2008, pp. 1057–1064.

[99] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression", *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005.

[100] M. Seeger, C. Williams, and N. Lawrence, "Fast forward selection to speed up sparse gaussian process regression", Tech. Rep., 2003.

[101] Z. Dai, A. Damianou, J. González, and N. Lawrence, "Variational auto-encoded deep gaussian processes", *arXiv preprint arXiv:1511.06455*, 2015.

[102] K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone, "Random feature expansions for deep gaussian processes", in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 884–893.

[103] M. Van der Wilk, C. E. Rasmussen, and J. Hensman, "Convolutional gaussian processes", in *Advances in Neural Information Processing Systems*, 2017, pp. 2849–2858.

[104] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera", *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 9, pp. 1820–1833, 2010.

[105] A. Andriyenko, S. Roth, and K. Schindler, "An analytical formulation of global occlusion reasoning for multi-target tracking", in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, IEEE, 2011, pp. 1839–1846.

[106] H. B. Shitrit, J. Berclaz, F. Fleuret, and P. Fua, "Tracking multiple people under global appearance constraints", in *2011 International Conference on Computer Vision*, IEEE, 2011, pp. 137–144.

[107] J. West, D. Ventura, and S. Warnick, "Spring research presentation: A theoretical foundation for inductive transfer", *Brigham Young University, College of Physical and Mathematical Sciences*, vol. 1, p. 32, 2007.

[108] R. Herbrich, *Learning Kernel Classifiers: Theory and Algorithms (Adaptive Computation and Machine Learning)*. MIT Press, 2002.

[109] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions", in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, 2003, pp. 912–919.

[110] W. Hu, X. Li, W. Luo, X. Zhang, S. Maybank, and Z. Zhang, "Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2420–2440, 2012.

[111] X. Zhu, J. Lafferty, and R. Rosenfeld, "Semi-supervised learning with graphs", PhD thesis, Carnegie Mellon University, language technologies institute, school of, 2005.

[112] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking", in *2008 IEEE Conference on computer vision and pattern recognition*, IEEE, 2008, pp. 1–8.

[113] ——, "Monocular 3d pose estimation and tracking by detection", in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 2010, pp. 623–630.

[114] J. Ferryman and A. Shahrokni, "Pets2009: Dataset and challenge", in *2009 Twelfth IEEE international workshop on performance evaluation of tracking and surveillance*, IEEE, 2009, pp. 1–6.

[115] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 319–336, 2008.

[116] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics", *Journal on Image and Video Processing*, vol. 2008, p. 1, 2008.

[117] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects", in *CVPR 2011*, IEEE, 2011, pp. 1201–1208.

[118] S.-H. Bae and K.-J. Yoon, "Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1218–1225.

[119] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3d traffic scene understanding from movable platforms", *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 5, pp. 1012–1025, 2013.

[120] C. Dicle, O. I. Camps, and M. Sznaier, "The way they move: Tracking multiple targets with similar appearance", in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2304–2311.

[121] J. Zhang, L. L. Presti, and S. Sclaroff, "Online multi-person tracking by tracker hierarchy", in *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance*, IEEE, 2012, pp. 379–385.

[122] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn, "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker", in *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, IEEE, 2011, pp. 120–127.

[123] L. Leal-Taixé, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese, "Learning an image-based motion context for multiple people tracking", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3542–3549.

[124] S. Hamid Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid, "Joint probabilistic data association revisited", in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3047–3055.

[125] J. H. Yoon, M.-H. Yang, J. Lim, and K.-J. Yoon, "Bayesian multi-object tracking using motion context from multiple objects", in *2015 IEEE Winter Conference on Applications of Computer Vision*, IEEE, 2015, pp. 33–40.

[126] L. Fagot-Bouquet, R. Audigier, Y. Dhome, and F. Lerasle, "Online multi-person tracking based on global sparse collaborative representations", in *2015 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2015, pp. 2414–2418.

[127] R. Sanchez-Matilla, F. Poiesi, and A. Cavallaro, "Online multi-target tracking with strong and weak detections", in *European Conference on Computer Vision*, Springer, 2016, pp. 84–99.

[128] J. Ju, D. Kim, B. Ku, D. K. Han, and H. Ko, "Online multi-object tracking with efficient track drift and fragmentation handling", *JOSA A*, vol. 34, no. 2, pp. 280–293, 2017.

[129] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks", in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[130] J. Ju, D. Kim, B. Ku, D. K. Han, and H. Ko, "Online multi-person tracking with two-stage data association and online appearance model learning", *IET Computer Vision*, vol. 11, no. 1, pp. 87–95, 2016.

[131] H. Kieritz, S. Becker, W. Hübner, and M. Arens, "Online multi-person tracking using integral channel features", in *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, IEEE, 2016, pp. 122–130.

[132] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking", *arXiv preprint arXiv:1504.01942*, 2015.

[133] G. E. Hinton, "Learning multiple layers of representation", *Trends in cognitive sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[134] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets", *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[135] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge", *Nature*, vol. 550, no. 7676, p. 354, 2017.

[136] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory", Colorado Univ at Boulder Dept of Computer Science, Tech. Rep., 1986.

[137] M. A. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning.", in *Aistats*, Citeseer, vol. 10, 2005, pp. 33–40.

[138] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes", in *Artificial Intelligence and Statistics*, 2009, pp. 567–574.

[139] M. Titsias and D Lawrence, "Bayesian gaussian process latent variable model", in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 844–851.

[140] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable variational gaussian process classification", 2015.

[141] D. Hernández-Lobato, J. M. Hernández-Lobato, and P. Dupont, "Robust multi-class gaussian process classification", in *Advances in neural information processing systems*, 2011, pp. 280–288.

[142] K. Krauth, E. V. Bonilla, K. Cutajar, and M. Filippone, "Autogp: Exploring the capabilities and limitations of gaussian process models", *arXiv preprint arXiv:1610.05392*, 2016.

[143] V. Dutordoir, M. van der Wilk, A. Artemev, M. Tomczak, and J. Hensman, "Translation insensitivity for deep convolutional gaussian processes", *arXiv preprint arXiv:1902.05888*, 2019.

[144] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[145] A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P. Xing, "Stochastic variational deep kernel learning", in *Advances in Neural Information Processing Systems*, 2016, pp. 2586–2594.

[146] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[147] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism", in *Advances in Neural Information Processing Systems*, 2019, pp. 103–112.

[148] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images", Citeseer, Tech. Rep., 2009.

Appendix A

Bayesian Variational Inference

A.1 Jensens Inequality

Theorem 1. *In the context of probability theory, it is generally stated in the following form: if X is a random variable and ϕ is a convex function, then*

$$\varphi(\mathbb{E}[X]) \leq \mathbb{E}[\varphi(X)]$$

A.2 Variation and Evidence Lower Bound (ELBO)

For the ordinary function $f(x)$, we can think of f as a real number operator about x, and its role is to map the real number x to the real number $f(x)$. Then analogy to this pattern, suppose there is a function operator F, which is a function operator about $f(x)$, can map $f(x)$ to a real number $F(f(x))$. For $f(x)$ we find the extreme value of $f(x)$ by changing x. In the variation, this x will be replaced by a function $y(x)$. We change $y(x)$ by changing x finally, make $F(y(x))$ to find the extreme value. Variation is the expansion of differentiation in the function space.

For a class of data x (whether it is audio or picture), the feature data obtained after encoding them often obeys a certain distribution $q(z)$. Where z is a hidden variable and the hidden distribution $q(z)$ is not known, but we can

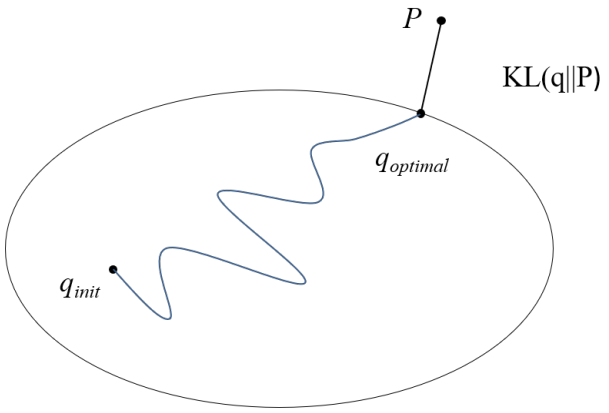

FIGURE A.1: Minimize the KL divergence so that the variational distribution q continuously approaches the true distribution P

infer $q(z)$ from the existing data X and it is $P(z|x)$. *KL divergence* is used to measure the distance between two distributions $P(z|x)$ and $q(z)$. When the distance is 0, it means that the two distributions are completely consistent. $P(x)$ does not change, so you want to make $KL(q(z)||P(z|x))$ smaller, which makes $ELBO$ larger, vice versa. A picture explanation is as shown in FIGURE A.1. Because $KL \geq 0$, $logP(x) \geq ELBO$. This conclusion can also be obtained by the following formula:

$$\begin{aligned}
\log P(x) &= \log P(x,z) - \log P(z|x) \\
&= \log \frac{P(x,z)}{q(z)} - \log \frac{P(z|x)}{q(z)} \\
&= \log P(x,z) - \log q(z) - \log \frac{P(z|x)}{q(z)} \\
&= \log P(x,z) - \log q(z) + \log \frac{q(z)}{P(z|x)}
\end{aligned} \tag{A.1}$$

Now both sides of the equation can be expected from $q(z)$.

$$\begin{aligned}
\int q(z) \log P(x) dz = &\int q(z) \log P(x,z) dz - \int q(z) \log q(z) dz \\
&+ \int q(z) \log \frac{q(z)}{P(z|x)} dz
\end{aligned} \tag{A.2}$$

Since $q(z)$ is not related to $P(x)$, so we have

$$\int q(z) \log P(x) dz = \log P(x) \tag{A.3}$$

The original formula eventually became.

$$\log P(x) = \underbrace{\int q(z) \log P(x,z) dz - \int q(z) \log q(z) dz}_{L(q(z)),\ \text{ELBO(Evidene Lower Bound)}} + \underbrace{\int q(z) \log \frac{q(z)}{P(z|x))} dz}_{KL(q(z) \| P(z|x))} \tag{A.4}$$

This conclusion can also be obtained by the following formula. The key to this formula is the middle inequality part, which is Jensen's inequality and has mentioned above.

$$\begin{aligned}
\log P(x) &= \log \left(\int P(x,z) dz \right) \\
&= \log \left(\int \left(\frac{P(x,z)}{q(z)} q(z) \right) dz \right) \\
&= \log \mathbb{E}_{zq(z)} \left[\frac{P(x,z)}{q(z)} \right] \\
&\geq \mathbb{E}_{zq(z)} \log \left(\frac{P(x,z)}{q(z)} \right) \\
&= \mathbb{E}_{zq(z)} \log P(x,z) - \mathbb{E}_{zq(z)} \log q(z) \\
&= \int q(z) \log P(x,z) dz - \int q(z) \log q(z) dz
\end{aligned} \tag{A.5}$$

A.3 Kullback-Leibler Divergence

Because *KL divergence* is not commutative, it cannot be understood as the concept of "distance". It does not measure the distance between two distributions in space. It is more accurate to measure the loss of information from one distribution compared to the other. In probability theory or information

theory, *Kullback-Leibler divergence* also known as relative entropy, is a way to describe the difference between two different probability distributions like P distribution and q distribution. *KL divergence* has its own clear physical meaning in information theory. It is used to measure the number of additional bits required to average the samples from the P distribution using a q distribution-based coding. The physical meaning in the field of machine learning is used to measure the degree of similarity or similarity of two functions. If it is related to the minimum code. The *KL divergence* can also be rewritten as *KL divergence* calculation formula for discrete probability distribution:

$$KL(P\|q) = \sum P(x) \log \frac{P(x)}{q(x)} \tag{A.6}$$

KL divergence calculation formula for continuous probability distribution

$$KL(P||q) = \int P(x) \log \frac{P(x)}{q(x)} dx \tag{A.7}$$

In Shannon's information theory, a P distribution-based coding method is used to encode samples from P distribution. The average number of bits required for its optimal encoding (ie, the entropy of this character set) is:

$$\mathrm{H(x)} = \sum_{x \in X} P(x) \quad * \quad \log\left(\frac{1}{P(x)}\right) \tag{A.8}$$

Using P distribution-based coding to encode samples from q distribution, the number of bits required becomes:

$$\mathrm{H'(x)} = \sum_{x \in X} P(x) \quad * \quad \log\left(\frac{1}{q(x)}\right) \tag{A.9}$$

Then we can get the *KL divergence* of P distribution and q distribution

$$\begin{aligned} KL(\mathrm{P}\|\mathrm{q}) = H'(x) - H(x) &= \sum_{x \in X} P(x) * \log\left(\frac{1}{q(x)}\right) - \sum_{x \in X} P(x) * \log\left(\frac{1}{P(x)}\right) \\ &= \sum_{x \in X} P(x) * \left[\log\left(\frac{P(x)}{q(x)}\right)\right] \end{aligned} \tag{A.10}$$

Appendix B

Label Propagation Algorithm

The Label propagation algorithm is initially proposed by zhu [109]. It is a graph-based semi-supervised learning method. The basic idea is to use the label information of labeled nodes to predict the label information of unlabeled nodes. Here is a brief introduction to the Label Propagation algorithm.

B.1 Similarity Matrix Construction

Assume we have a set of labeled dataset $\{(x_1, y_1) \dots (x_l, y_l)\}$ and the class lable $Y_L = \{y_1, \cdots, y_l\} \in \{1, \cdots, C\}$, the number of class C is known and all exist in the labeled data. There is another set of $\{x_{l+1} \dots x_{l+u}\}$ unlabeled data and the corresponding unobserved class label Y_U $\{y_{l+1}, \cdots, y_{l+u}\}$, where l$\ll$ u and $n = l + u$. Put the labeled and unlabeled data together we have dataset $X = \{x_1, \cdots, x_{l+u}\} \in R$. The problem turns into: from the data set X, use the labeled data's label information Y_L find a corresponding label Y_U for each data of the unlabeled dataset. Use all data as nodes (including labeled and unlabeled data) to create a graph. There are many ways to construct this graph. Here we assume that the graph is fully connected, and the edge weights of node i and node j are:

$$w_{ij} = \exp\left(-\frac{\left\|x_i - x_j\right\|^2}{\alpha^2}\right) \tag{B.1}$$

Where α is a hyperparameter. A common graph construction method is k-nn graph, which only keeps the k nearest neighbor weights of each node, and the others are 0. That is, there are no edges, so it is a sparse similarity matrix. This can effectively reduce the amount of computation.

B.2 Label Propagation

The label propagation algorithm is relatively simple: labels are propagated through the edges between nodes. The larger the weight of the edge, the more similar the two nodes are, and the easier the label is to propagate. We define an NxN probability transition matrix P:

$$P_{ij} = P(i \to j) = \frac{w_{ij}}{\sum_{k=1}^{n} w_{ik}} \tag{B.2}$$

P_{ij} represents the probability of transition from node i to node j. We merge Y_L and Y_U to get a $N \times C$ soft label matrix $f = \begin{bmatrix} Y_L \\ Y_U \end{bmatrix}$. The meaning of soft label is that we keep the probability that the sample i belong to each class, instead of being mutually exclusive. This sample belongs to only one class with probability 1. Of course, when the class of this sample i is finally determined, the class with max, which is the most probable, is taken as its class. Then there is a Y_U in f. It was unknown at first, and it doesn't matter, just set a value.

The LP algorithm is as follows:

1. Propagate $f \leftarrow Pf$
2. Fix the labeled data $f_L = Y_L$
3. Repeat step 1 and step 2 until f converges

Step 1 is to multiply the matrix P and the matrix f. In this step, each node propagates its own label to other nodes with a probability determined

by P. If the two nodes are more similar (the closer the distance is in Euclidean space), the more easily the other's label will be given by its own label. Step 2 is very critical. Because the label of the labeled data is determined in advance, it cannot be changed, so it must return to its original label after each propagation.

B.3 Variant of Label Propagation Algorithm

We all know that we calculate a soft label matrix $f = \begin{bmatrix} Y_L \\ Y_U \end{bmatrix}$ in each iteration. The Y_L is known in advance, calculating it is useless, and in step 2 we have to get it back. All we care about is Y_U, we can divide the matrix P as follows:

$$P = \begin{bmatrix} P_{LL} & P_{LU} \\ P_{UL} & P_{UU} \end{bmatrix} \tag{B.3}$$

Make $f = \begin{bmatrix} f_L \\ f_U \end{bmatrix}$. At this time, we can perform the following calculation to get the soft label of the unlabeled data.

$$f_U \leftarrow P_{UU} f_U + P_{UL} Y_L \tag{B.4}$$

Iterate this step until convergence is enough. It can be seen that f_U depends not only on the label of the labeled data and its transition probability but also on the current label and transition probability of the unlabeled data. Therefore, the LP algorithm can additionally use the distribution characteristics of unlabeled data.

B.4 Proof of Convergence

When n approaches infinity, there is

$$f_U = \lim_{n\to\infty} (P_{UU})^n f_U^0 + \left(\sum_{i=1}^{n} (P_{UU})^{(i-1)} \right) P_{UL} Y_L \tag{B.5}$$

Among them f_U^0 is the initial value of f_U, So we need to prove $(P_{UU})^n f_U^0 \to 0$. Because P is row-normalized and P_{UU} is a submatrix of P, so

$$\exists \gamma < 1, \sum_{j=1}^{u} (P_{UU})_{ij} \leq \gamma, \forall i = 1 \dots u \tag{B.6}$$

Therefore

$$\begin{aligned} \sum_j (P_{UU})_{ij}^n &= \sum_j \sum_k (P_{UU})_{ik}^{(n-1)} (P_{UU})_{kj} \\ &= \sum_k (P_{UU})_{ik}^{(n-1)} \sum_j (P_{UU})_{kj} \\ &\leq \sum_k (P_{UU})_{ik}^{(n-1)} \gamma \\ &\leq \gamma^n \end{aligned} \tag{B.7}$$

When n approaches infinity, $r^n = 0$ and $(P_{UU})^n$ converges to 0, which also means $(P_{UU})^n f_U^0 \to 0$. Therefore The initial value of f_U^0 is irrelevant. At the same time, f_U converges at the n-th iteration. For a more detailed explanation of the label propagation algorithm, please refer to Zhu's paper [109].

Appendix C

Laplace Approximation for Gaussian Processes

C.1 Laplace Approximation for Gaussian Processes with Multiple-Person Tracking Framework

We use $\mathbf{G} = \tilde{\Delta}^{-1}$ to denote the covariance matrix (the gram matrix). Considering the sigmoid noise label output model, the $P\left(l_{\mathrm{A}}, l_{\mathrm{U}} | \mathcal{X}_{\mathrm{U}}, \mathcal{D}_{\mathrm{A}}, \mathcal{D}_{\mathrm{T}}\right)$ is no longer a Gaussian and has no closed form solution, assuming $P\left(l_{\mathrm{A}}, l_{\mathrm{U}} | \mathcal{X}_{\mathrm{U}}, \mathcal{D}_{\mathrm{A}}, \mathcal{D}_{\mathrm{T}}\right)$ is a uni-modal function with mode $(l_{\mathrm{A}}, l_{\mathrm{U}}) \in \mathbb{R}^{n_{\mathrm{A}}+n_{\mathrm{U}}}$. We use its Laplace approximation with the mode $\mu' \in \mathbb{R}^{n_A+n_U}$ and covariance $\Sigma \in \mathbb{R}^{(n_A+n_U)\times(n_A+n_U)}$ instead the correct density.

Taking the logarithm will not change the maximum but render optimization easier; take the logarithm of Eq. 3.6, we have the following objective function to maximize:

$$\mathcal{J}(l_A, l_U) = \underbrace{\ln(\mathrm{P}(y_A|l_A))}_{\mathcal{Q}_1(l_A)} + \underbrace{\ln(\mathrm{P}(l_A, l_U|\mathcal{X}_A, \mathcal{X}_U, \mathcal{D}_T))}_{\mathcal{Q}_2(l_A, l_U)} - \ln(\mathrm{P}(l_A|\mathcal{X}_A, \mathcal{X}_U, \mathcal{D}_T)) \tag{C.1}$$

The last term is normalization constant has no influence on the maximization can be omitted from the optimization.

Let's focus first on $Q_2(l_A, l_U)$, which builds the link between Gaussian processes regression and classification. According to Eq. 3.7, this term is given by

$$Q_2(l_A, l_U) = -\frac{1}{2}(\ln(2\pi)^{n_A+n_U} + \ln|\mathbf{G}| + (l-\mu)^\top \mathbf{G}^{-1}(l-\mu)) \tag{C.2}$$

We define $l^\top = (l_A^\top l_U^\top)$, $\mathbf{y}^\top = (\mathbf{y}_T^\top l_A^\top)$, $\mathbf{y}_T = [y_1, y_2, ..., y_{n_T}]$. Moreover $\mathbf{G}_{all}$ $= \begin{pmatrix} \mathbf{G}_{LL} & \mathbf{G}_{LU} \\ \mathbf{G}_{UL} & \mathbf{G}_{UU} \end{pmatrix} = \begin{pmatrix} \mathbf{G}_{TT} & \mathbf{G}_{TZ} \\ \mathbf{G}_{ZT} & \mathbf{G}_{ZZ} \end{pmatrix}$ is $(n_L + n_U) \times (n_L + n_U)$ Gram matrix (symmetric, non-singular), which is defined over all samples. Its inverse is $\mathbf{G}_{all}^{-1} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{M} \end{pmatrix}$ (according to the Partitioned Matrix Inversion Theorem) derivate from the latter one. Where $\mathbf{A} = \mathbf{G}_{TT}^{-1} + \mathbf{G}_{TT}^{-1}\mathbf{G}_{TZ}\mathbf{M}\mathbf{G}_{ZT}\mathbf{G}_{TT}^{-1}$, $\mathbf{B} = -\mathbf{G}_{TT}^{-1}\mathbf{G}_{TZ}\mathbf{M}$, $\mathbf{M} = (\mathbf{G}_{ZZ} - \mathbf{G}_{ZT}\mathbf{G}_{TT}^{-1}\mathbf{G}_{TZ})^{-1}$. According to the works of Herbrich [108] and Zhu et al [111], we can determine μ and $\mathbf{G}$ in Eq. 3.7 as $\mu = -\mathbf{M}^{-1}\mathbf{B}^\top \mathbf{y}_\top$, $\mathbf{G} = \mathbf{M}^{-1}$. Hence, Eq. C.2 can derivate as follows:

$$\begin{aligned} Q_2(l_A, l_U) &= -\frac{1}{2}(\ln(2\pi)^{n_A+n_U} + \ln|\mathbf{G}| + (l-\mu)^\top \mathbf{G}^{-1}(l-\mu)) \\ &= -\frac{1}{2}(\ln|\mathbf{G}_{\text{all}}| + \mathbf{y}_T^\top \mathbf{A}\mathbf{y}_T + l^\top \mathbf{B}^\top \mathbf{y}_T + \mathbf{y}_T^\top \mathbf{B} l + l^\top \mathbf{M} l) + c_1 \\ &= -\frac{1}{2}(\ln|\mathbf{G}_{\text{all}}| + (\mathbf{y}_T^\top l^\top)\mathbf{G}_{\text{all}}^{-1} \begin{pmatrix} \mathbf{y}_T \\ l \end{pmatrix}) + c_1 \qquad \text{(C.3)} \\ &= -\frac{1}{2}(\ln|\mathbf{G}_{\text{all}}| + (\mathbf{y}^\top l_U^\top)\mathbf{G}_{\text{all}}^{-1} \begin{pmatrix} \mathbf{y} \\ l_U \end{pmatrix}) + c_1 \qquad \text{(C.4)} \end{aligned}$$

where $c_1 = -\frac{1}{2}(\mathbf{y}_T^\top(\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^\top - \mathbf{A})\mathbf{y}_T + \mathbf{ln}|\mathbf{G}| - \mathbf{ln}|\mathbf{G}_{\text{all}}| + \mathbf{ln}(2\pi)^{n_A+n_U})$ summarizes all terms independent of l, we can see that $Q_1(l_A)$ does not depend on l_U. Thus we can derive the optimal value $\hat{l_U}$ of l_U by maximizing $Q_2(l_A, \bullet)$, taking the derivative of $Q_2(l_A, \bullet)$ w.r.t. l_U, setting this function to

zero. According to [111] the optimal value $\hat{l_U}$ can be derived as:

$$\hat{l_U} = \mathbf{G}_{UL}\mathbf{G}_{LL}^{-1}\begin{pmatrix}\mathbf{y}_T\\ \hat{l_A}\end{pmatrix} = \mathbf{G}_{UL}\mathbf{G}_{LL}^{-1}\mathbf{y} \tag{C.5}$$

Substituting this expression into Eq. (C.4) shows that this term equals

$$\begin{aligned} Q_2(l_A, l_U) &= -\frac{1}{2}(\mathbf{y}_T^\top l_A^\top)\mathbf{G}_{LL}^{-1}\begin{pmatrix}\mathbf{y}_T\\ l_A\end{pmatrix} + c_1 - \frac{1}{2}\ln|\mathbf{G}_{\text{all}}| \\ &= -\frac{1}{2}(\mathbf{y}_T^\top l_A^\top)\mathbf{G}_{LL}^{-1}\begin{pmatrix}\mathbf{y}_T\\ l_A\end{pmatrix} + c_2 \end{aligned} \tag{C.6}$$

Let's turn our attention to the first term $Q_1(l_A)$ of $\mathcal{J}(l_A, l_U)$. We define $\pi(l_j) = (1 + \mathrm{e}^{-2\gamma l_j})^{-1}$, where $j = n_T + 1, n_T + 2, \ldots, n_T + n_A$, the sigmoid noise label generation model can be written as :

$$\begin{aligned} P(y_j|l_j) &= \frac{e^{\gamma l_j y_j}}{e^{\gamma l_j y_j} + e^{-\gamma l_j y_j}} \\ &= \left(\frac{e^{\gamma l_j}}{e^{\gamma l_j} + e^{-\gamma l_j}}\right)^{\frac{1+y_j}{2}}\left(1 - \frac{e^{\gamma l_j}}{e^{\gamma l_j} + e^{-\gamma l_j}}\right)^{\frac{1-y_j}{2}} \\ &= \pi(l_j)^{\frac{1+y_j}{2}}(1 - \pi(l_j))^{\frac{1-y_j}{2}} \end{aligned} \tag{C.7}$$

Therefore

$$\begin{aligned} Q_1(l_A) &= \ln(\mathrm{P}(\mathbf{y}_A|l_A)) \\ &= \sum_{j=n_T+1}^{n_L}\ln(\mathrm{P}(y_j|l_j)) \\ &= \gamma(\mathbf{y}_A - \mathbf{1})^\top l_A - \sum_{j=n_T+1}^{n_L}\ln(1 + \mathrm{e}^{-2\gamma l_j}) \end{aligned} \tag{C.8}$$

Combine $Q_1(l_A)$ and $Q_2(l_A, l_U)$ together, we obtain the following revised objective function $\mathcal{J}(l_A)$. Maximize $\mathcal{J}(l_A)$ over $l_A \in \mathbb{R}^{\mathrm{n_A}}$ we get

$$\mathcal{J}(l_A) = \gamma(\mathbf{y}_A - \mathbf{1})^\top l_A - \sum_{j=n_T+1}^{n_L} \ln(1 + \mathrm{e}^{-2\gamma l_j}) - \frac{1}{2}(\mathbf{y}_T^\top l_A^\top)\mathbf{G}_{LL}^{-1} \begin{pmatrix} \mathbf{y}_T \\ l_A \end{pmatrix} + c_2 \tag{C.9}$$

The gradient vector $\hat{l_A}$ given by a straightforward calculation

$$\left.\frac{\partial \mathcal{J}(l_A)}{\partial l_A}\right|_{l_A=\hat{l_A}} = \gamma(\mathbf{y}_A - \mathbf{1}) + 2\gamma(1 - \pi(\hat{l_A})) - \mathrm{G}_{LL}^{-1}\hat{l_A} \tag{C.10}$$

Furthermore, let $\mathbf{G}_{LL}^{-1} = \begin{pmatrix} \mathbf{B}_{TT} & \mathbf{B}_{TA} \\ \mathbf{B}_{AT} & \mathbf{B}_{AA} \end{pmatrix}$, Eq. C.10 can be written as

$$\left.\frac{\partial \mathcal{J}(l_A)}{\partial l_A}\right|_{l_A=\hat{l_A}} = \gamma(\mathbf{y}_A - \mathbf{1}) + 2\gamma(1 - \pi(\hat{l_A})) - \mathrm{B}_{AA}\hat{l_A} - \mathrm{B}_{AT}\mathrm{y}_T \tag{C.11}$$

Where $\pi(\hat{l_A}) = (\pi(\hat{l_{\mathrm{n}_T+1}}), \ldots, \pi(\hat{l_{\mathrm{n}_L}}))^\top$. We can see from this expression, the term $\pi(\hat{l_A})$ make it impossible to compute $\hat{l_A}$ in a closed form, we use the Newton-Raphson method.

$$l_A^{i+1} \leftarrow l_A^i - \eta \mathbf{H}^{-1} \cdot \left.\frac{\partial \mathcal{J}(l_A)}{\partial l_A}\right|_{l_A=l_A^i} \tag{C.12}$$

Where $\mathbf{H}$ is $n_A \times n_A$ Hessian matrix defined as :

$$\mathbf{H}_{\hat{l_A}} = \begin{pmatrix} \left.\frac{\partial^2 \mathcal{J}(l_A)}{\partial l_{n_T+1} \partial l_{n_T+1}}\right|_{l_{n_T+1}=\hat{l_{n_T+1}}} & \cdots & \left.\frac{\partial^2 \mathcal{J}(l_A)}{\partial l_{n_T+1} \partial l_{n_L}}\right|_{l_{n_T+1}=\hat{l_{n_T+1}}, l_{n_L}=\hat{l_{n_L}}} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial^2 \mathcal{J}(l_A)}{\partial l_{n_L} \partial l_{n_T+1}}\right|_{l_{n_L}=\hat{l_{n_L}}, l_{n_T+1}=\hat{l_{n_T+1}}} & \cdots & \left.\frac{\partial^2 \mathcal{J}(l_A)}{\partial l_{n_L} \partial l_{n_L}}\right|_{l_{n_L}=\hat{l_{n_L}}} \end{pmatrix} = -\mathbf{P} - \mathbf{B}_{AA} \tag{C.13}$$

Where $\mathbf{P}$ is a diagonal matrix with elements $P_{ii} = 4\gamma^2\pi(l_i)(1 - \pi(l_i))$ and $\eta \in \mathbb{R}^+$ has to be chosen such that $\mathcal{J}(l_A^{i+1}) > \mathcal{J}(l_A^i)$. We set η 0.4 and the number of iterations used in Newton-Raphson method is 7.

As the latent variable l can be the soft substitution of label y, we build two trackers by using auxiliary and target samples respectively. For each tracker based on the derivation above we compute the soft label vector l_U, hence we get two candidates set V_A and V_T. We check the similarity of two sets, if the similarity is high we can use anyone, if the similarity is low we rely more on the target decision to ensure the tracking consistency. When the similarity is zero, we use the auxiliary decision to handle the heavy occlusion or the severe appearance change.