Few-shot Image Classification with Gaussian Process Classifier

Zhu Morui
NEPTUN CODE: HHF14P

Abstract

In the computer vision field, it is very important to incorporate with computational uncertainty, which could make AI/machine learning technologies more reliable. Deep neural networks perform the state-of-art on various computer vision tasks (e.g. image classification, semantic segmentation, object detection), while there are many drawbacks behind this, such as long trainning time, overfitting and uninterpretable. To solve these issues, Gaussian processes (GPs) provide the uncertainty estimation and perform well only with few-shot data. In this report, I explore hw GPs could be applied in image classification task and how to incorporate with uncertainty. The results show that the GPs classifier is reliable when I set an acceptable uncertainty threhold.

1. Introduction

Deep neural networks (DNNs) are widely used in computer vision fileds, becaues of they have many advantages, such as high precision, powerful feature extraction and real-time infernecing. Recently, many papers adopt DNNs to solve the few-shot image classification task, e.g. they use data augmentation to increase the number of samples. In addition, transfer learning, meta learning and bayesain learning are also used in few-shot image classification task. In practice, DNNs have the following drawbacks.

1. DNNs need plenty of labeled data for training, high qualiy dataset collection and labelling is expensive and time-consuming.
2. Trainning a DNN requires many computation resources and plenty of time.
3. DNNs have a powerful fitting performance, while often leads to overfitting on training data.

4. Due to the complexity of deep neural networks, it is hard to explain the prediction of DNNs.

To solve above issues, I apply gaussian process in the few-shot image classification tasks. The advantages of GPs are following:

1. Flexibility: GPs could be applied in various tasks with different data types. It could be used in regression, classification, time serise analysis, etc.
2. Uncertainty approximation: GPs provides the uncertainty approximation for predicted results. It gives related probabilistic information about prediction, which helps use evalute the reliabilty and risks of models.
3. Well performance on few-shot learning, GPs has a powerful prior assumption about data, which makes it perform well on few data.

But note that the property of GPs (continues space) leads to inferior accuracy, especially doing multi-calsses classification task.

The GPs observes variables in a continuous space. When observed variable space of GPs is within the realm of real numbers, we can conduct regression to make predictions. This is so called Gaussian Process Regression (GPR). Conversely, when the observation variable space is within the realm of integers (where observation points are discrete), we can conduct classification. And it is called Gaussian Process Classification (GPC). Thus, we can apply the method of GPC to few-shot image classification task. In this way, we aviod a large architecture and data requirements typically associated with DNNs, while still achieving considerable classification results.

In this report, I will introduce the mathematical background of GPs and how it designed for classification task. And I will show related experiments on fashion-mnist and cifar-10 dataset, then incooperate with uncertainty to make prediction reliable. Finally, I will eplain the results and give an conclusion.

2. Gaussian process classifier

The applications of GPs include regression and classification. It's called regression problem when both input and output data are continuous, while prediction with finite set of discrete output variables is known as classification. In fact, Gaussian Process Classification (GPC) is an extension of Gaussian Process regression (GPR). From the output of GPR, we can use a link function to get a corresponding label, based on an appropriate threshold value, different category results can be output.

We assume dataset D contains n obeserved variables $D = \{(x_i, y_i)|i = 1, \ldots, n\}$, where x denotes the D-dimensional input vectors (covariates), and y represents the output vectors (dependent variables). The column vector inputs for all n cases can be combined into a design matrix X of size $D \times n$. We aim to construct a model $y = f(x)$ based on the data from the training set, where $f(.)$ represents the mapping relationship between the covariates and the dependent variable, which is the latent variable function. This model can then be applied to a new dataset $\{x^*, y^*\}$ for testing purposes. Given x^*, we can use the model to predict the corresponding output y^*. In more realistic modeling scenarios, it is common to encounter noise or uncertainty, making it challenging to obtain the exact function $f(x)$ itself.

$$y = f(x) + \epsilon, \epsilon \sim N(0, \delta_n^2 I_n) \tag{1}$$

Assume that $f(x)$ follows GPs, we can express it as $f(x) \sim GP(m(x), K(x, x'))$, then we get

$$y \sim GP(m(x), K(x, x') + \delta_n^2 I_n) \tag{2}$$

Where $m(x)$ is mean function, $K(x, x')$ is kernel function. Then we get joint distribution of y^* and y.

$$\begin{pmatrix} y \\ y^* \end{pmatrix} \sim GP\left(m(X, X^*), \begin{bmatrix} K(X, X) + \delta_n^2 I_n & K(X, X^*) \\ K(X^*, X) & K(X^*, X^*) \end{bmatrix}\right) \tag{3}$$

Then we conduct the conditional distribution, which is the key prediction function of the GPR.

$$\begin{aligned} y^*|y \sim GP(K(X^*, X)[K(X, X) + \delta_n^2 I_n]^{-1} y, \\ K(X^*, X^*) - K(X^*, X)[K(X, X) + \delta_n^2 I_n]^{-1} K(X, X^*)) \end{aligned} \tag{4}$$

But in this case, we can only get continuous predictions not labels. In a task of binary classification, we place a GPs prior on function $f(x)$ and use a function σ to compress the results within the range of $[0, 1]$.

$$p(y = 1|x) = \sigma(f(x)) \tag{5}$$

Next, we proceed with Bayesian inference by computing the distribution of the latent variable function applied to the test set data.

$$p(f^*|X, y, X^*) = \int p(f^*|X, X^*, f) p(f|X, y) df \tag{6}$$

Where $p(f|X, y) = \frac{p(y|f)p(f|X)}{p(y|X)}$ is the posterior distribution of the latent variable function, which can be used to generate a probability prediction based on the dataset.

$$p(y^* = 1|X, y, X^*) = \int \sigma(f^*) p(f^*|X, y, X^*) df^* \tag{7}$$

In the task of regression, the prediction can be directly calculated. However, in the classification problems, the Non-Gaussian likelihood makes the calculation more challenging. It is very hard to deal with posterior distribution and probability prediction. Thus, we need some approximation technologies like Laplace approximation and Variational Inference.

In this report, I use Laplace approximation algorithm for GPC. We can use Laplace algorithm to approximate posterior $p(f|X,y)$ with gaussian function $q(f|X,y)$. Apply a 2nd Taylor expansion on maximum posterior of $\log p(f|X,y)$, we can get gaussian approximation,

$$q(f|X,y) = N(f|\hat{f}, A^{-1}) \propto exp(-\frac{1}{2}(f-\hat{f})^T A(f-\hat{f})) \tag{8}$$

Where $\hat{f} = \arg max_f p(f|X,y),\ A = -\nabla\nabla \log p(f|X,y)|_{f=\hat{f}}$ is the Hessian matrix of the negative log of posterior at that point.

Based on the Bayesian theorem, we can get the marginal likelihood function toward $p(f|X,y)$

$$\Psi(f) = \log p(y|f) + \log p(f|X) = \log p(y|f) - \frac{1}{2}f^T K^{-1} f - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi \tag{9}$$

Then differentiating the equation 9, we can get,

$$\begin{aligned} \nabla\Psi(f) &= \nabla \log p(y|f) - K^{-1}f, \\ \nabla\nabla\Psi(f) &= \nabla\nabla \log p(y|f) - K^{-1} = -W - K^{-1} \end{aligned} \tag{10}$$

Where $W = -\nabla\nabla \log p(y|f)$ is a diagonal matrix. After finding the maximum the posterior of $\hat{f}$, we can specify the Laplace approximation of the posterior as a Gaussian function. The mean and covariance matrix are given by the negative inverse Hessian of Ψ.

$$q(f|X,y) = N(\hat{f}, (K^{-1}+W)^{-1}) \tag{11}$$

3. Few-shot image classification

In this section, I compare GPC on the image classification benchmarks of Fashion-MNIST and CIFAR-10. GPC implementation is from Python library scikit-learn. Currently, the implementation is restricted to using the logistic link function. For multi-class classification, several binary one-versus rest classifiers are fitted, i.e., this function thus does not implement a true multi-class Lapace approximation. Due to that issue, I both try binary and multi-class image classification on two dataset, and compare the precisions between two tasks.

Note that, in this task, I use kernel $1.0 \times RBF(1.0)$, and the kernel's hyper parameters are optimized during fitting.

3.1 Data preprocess

3.1.1 Fashion MNIST

The Fashion-MNIST dataset consists of a collection of grayscale images depicting various clothing and accessory items. It comprises 10 different categories, each containing 6,000 images, resulting in a total of 60,000 training images and 10,000 test images. Each image has a resolution of 28×28 pixels.

In this task, I just simply normalize the images within the range of $[0, 1]$ by deviding 255 and convert the 2D matrix into an 1D vector with size of 784.

3.1.2 Cifar-10

The CIFAR-10 dataset consists of 60,000 color images, with each image having a resolution of $32 \times 32 \times 3$ pixels. These images are divided into 10 different classes, each representing a distinct object category, including airplanes, automobiles, birds, etc. Each class contains 6,000 images.

In this task, I just simply normalize the images within the range of $[0, 1]$ by deviding 255 and convert the RGB image into an 1D vector with size of 3072.

3.2 Binary image classification

I select 2 labels randomly for this task. After trying PCA, I found results are not improved a lot, to simply the preprocess, I directly use all feature variables. In this example, label 1 and 4 are randomly selected in Fashion MNIST dataset, and I filter selected data from all dataset which have label 1 or 4. Then I randomly select 40 images from the filtered dataset. Finally, 40 images are preprocessed and randomly splitted into train data with 28 images and test data with 12 images, respectively.

This figure shows the test results on test data in the Fashion MNIST and Cifar-10 dataset.

Figure 1. Binary classification results on 28 training images with GPC in Fashion

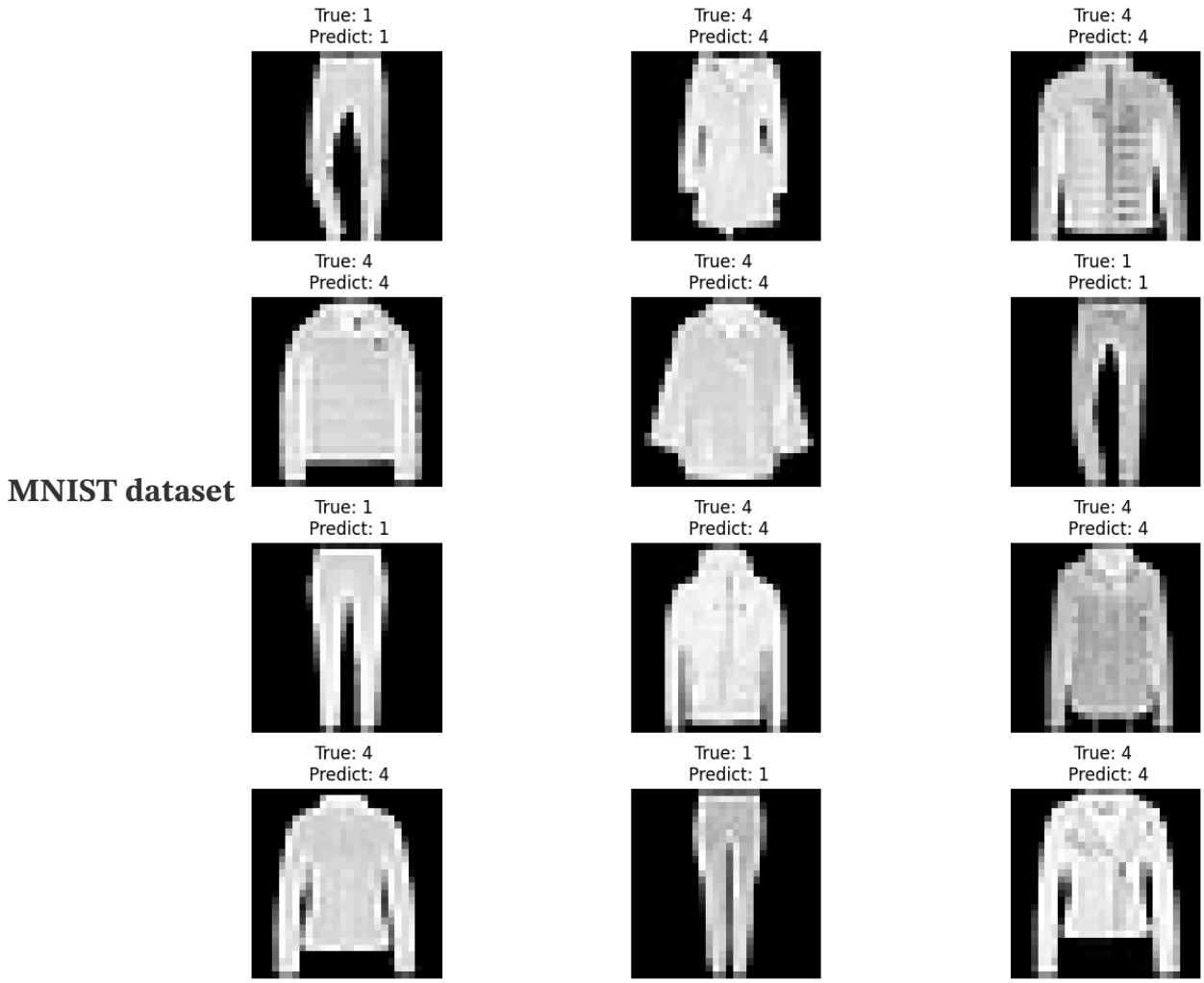
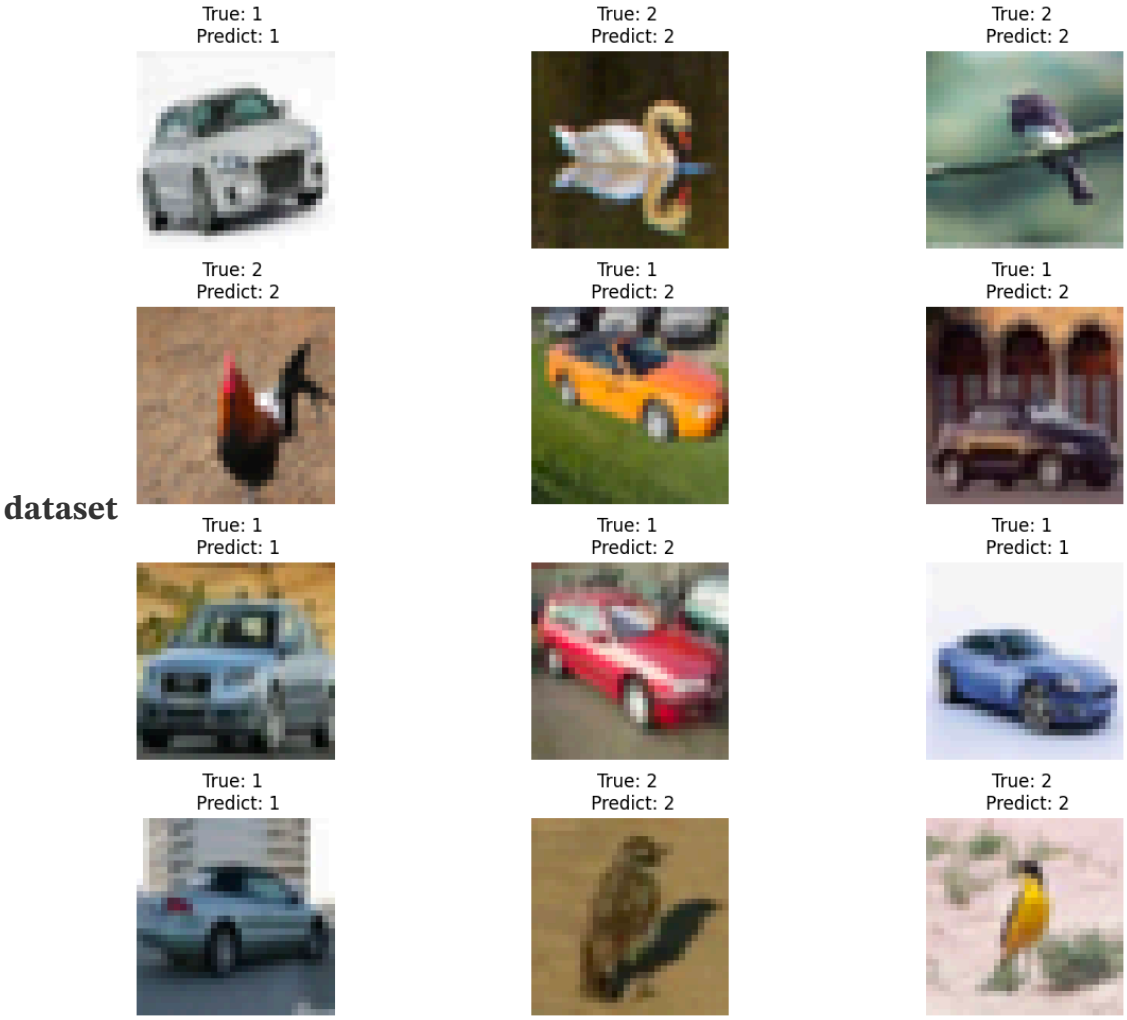


Figure 2. Binary classification results on 28 training images with GPC in CIFAR-10



To compare with different number of training images, start from 28 training images, I gradually increase the number of training images up to 140 images. The following table shows the prediction accuracy of the model with different number of training images on all dataset with corresponding labels.

Number of training images	Fashion MNIST (label 1 and 4)	CIFAR-10 (label 1 and 2)
28	0.947	0.6
56	0.980	0.65
84	0.987	0.63
112	0.980	0.66
140	0.986	0.652

From above table, we can find that GPC performs well on Fashion MNIST dataset, while terrible on CIFAR-10 dataset. The reason is that I simply convert RGB images into 1D vector. In addition, GPs has the issue of curse of dimensionality, and I did not apply feature dimensionality reduction in the data preprocess. Because we are focusing on how to incorporate uncertainty, next, GPC will mainly performs on Fashion MNIST dataset.

3.3 Multi-class image classification

The following figure shows the results with 112 training images on 4 classes.

Figure 3. 4-class classification results on 112 training images with GPC in Fashion MNIST dataset

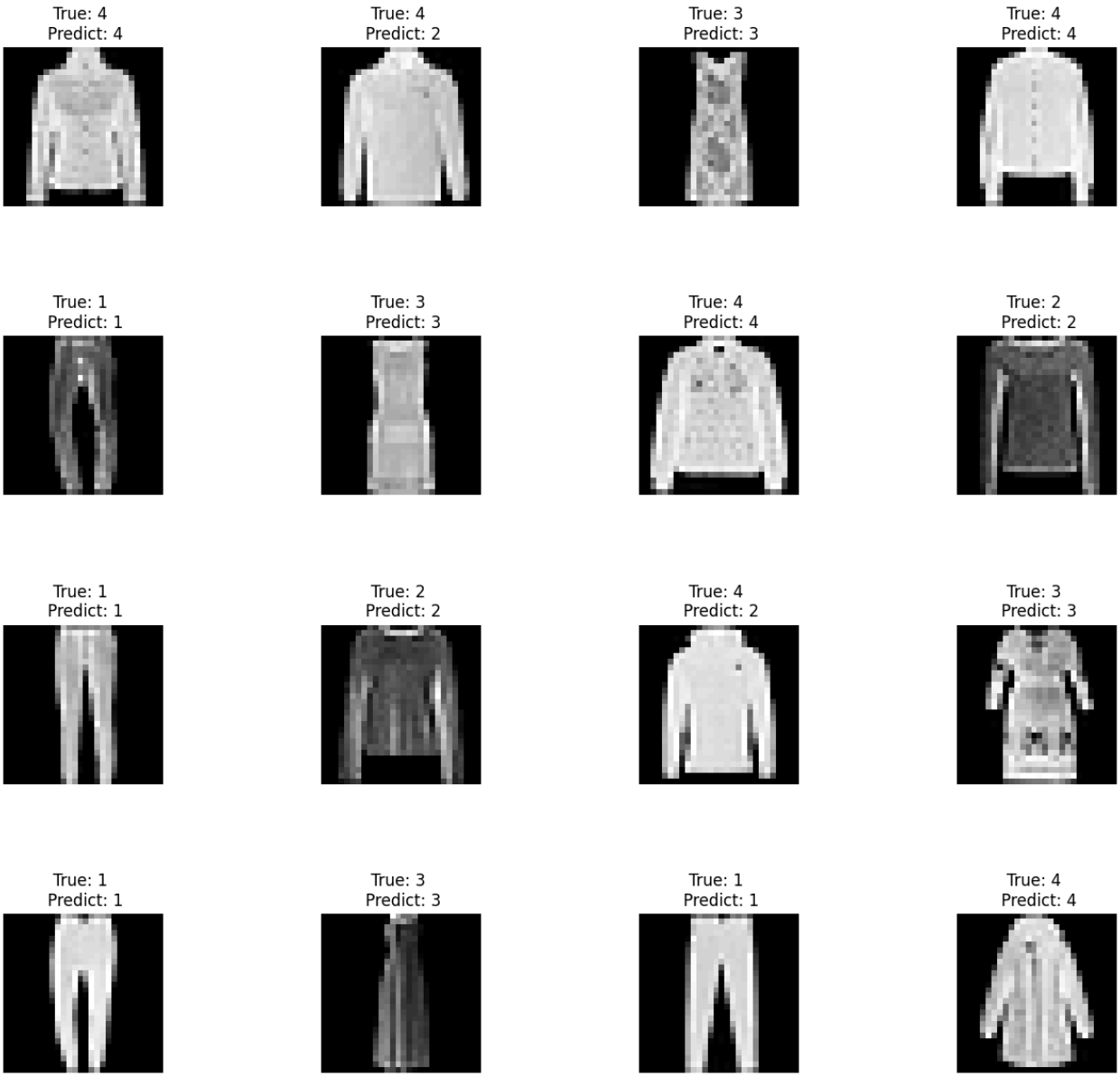
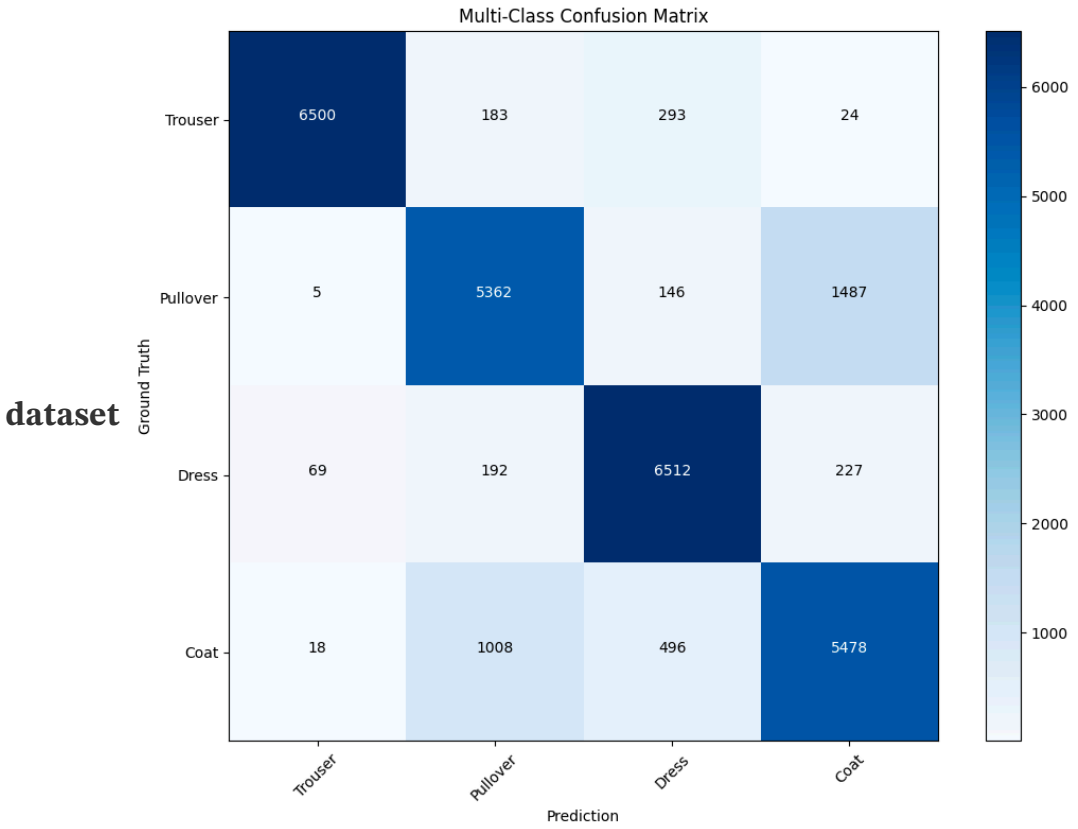


Figure 4. 4-class confusion matrix on 112 training images with GPC in Fashion MNIST



For multi-class image classification, I gradully increase the number of training images from 84 to 196 . Meanwhile, I increase the number of classes from 2 to 4. The following table shows the prediction accuracy of the model with different number of training images and labels on all dataset with corresponding labels.

Num: training imgs	Classes: 1, 2	Classes: 1, 2, 3	Classes: 1, 2, 3, 4	Classes: 1, 2, 3, 4, 5
112	0.984	0.95	0.846	0.80
140	0.988	0.952	0.844	0.83
168	0.985	0.953	0.851	0.83
196	0.986	0.955	0.865	0.86
224	0.983	0.955	0.864	0.867

From above table, we can find that the task becomes harder while we have more classes, because GPC perform more binary classifcation which makes prediction accuracy lower. Meanwhile, if we increase the number of training images, the results are better, while it needs more computation resources and more time-consuming.

3.4 Incorporate with uncertainty

To incorporate with uncertainty, we can define the uncertainty u by variance of classification probability distribution. When σ^2 is low, which means a uniform distribution of the prediction. Assume we have n sample, for ith samples, u_i can be calculated:

$$u_i = \left(1 - \frac{v_i - \min(V)}{\max(V) - \min(V)}\right)^k \tag{12}$$

Where u_i is the uncertainty of ith sample's prediction, V is the set of all variances in the test dataset. k is a scale factor, which is set to be 10 based experiments.

To visualize the results, We use the example in section 2, we train GPC on 112 images with 4 classes, and the prediction accuracy is 0.834 on all filtered dataset with label 1, 2, 3 and 4.

In this experiment, we intentionally introduced some datasets that the model has never seen before, including labels that have not been trained on.

Figure 5. 4-class classification results on dataset with labels have never seen with GPC in Fashion MNIST dataset

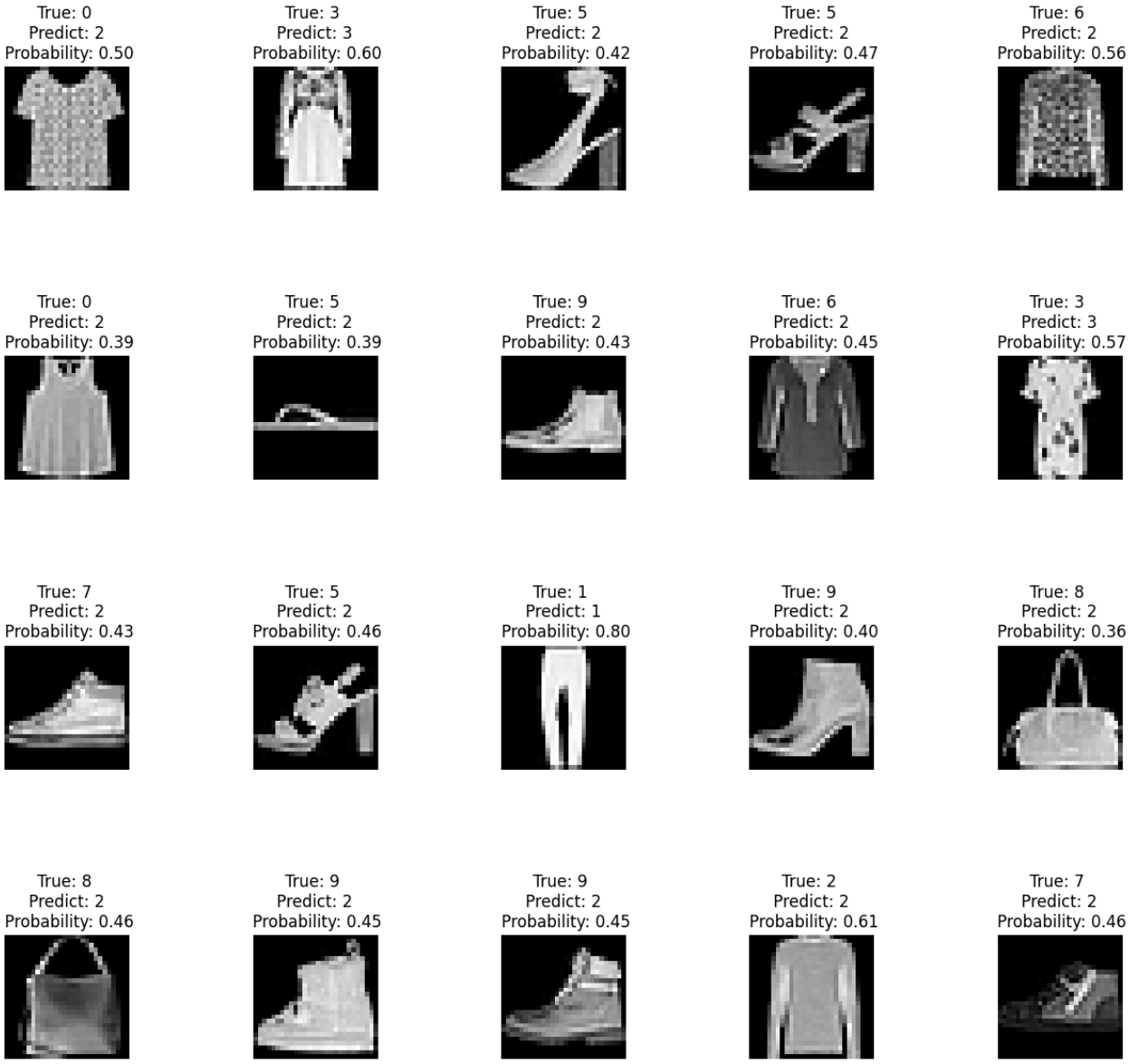
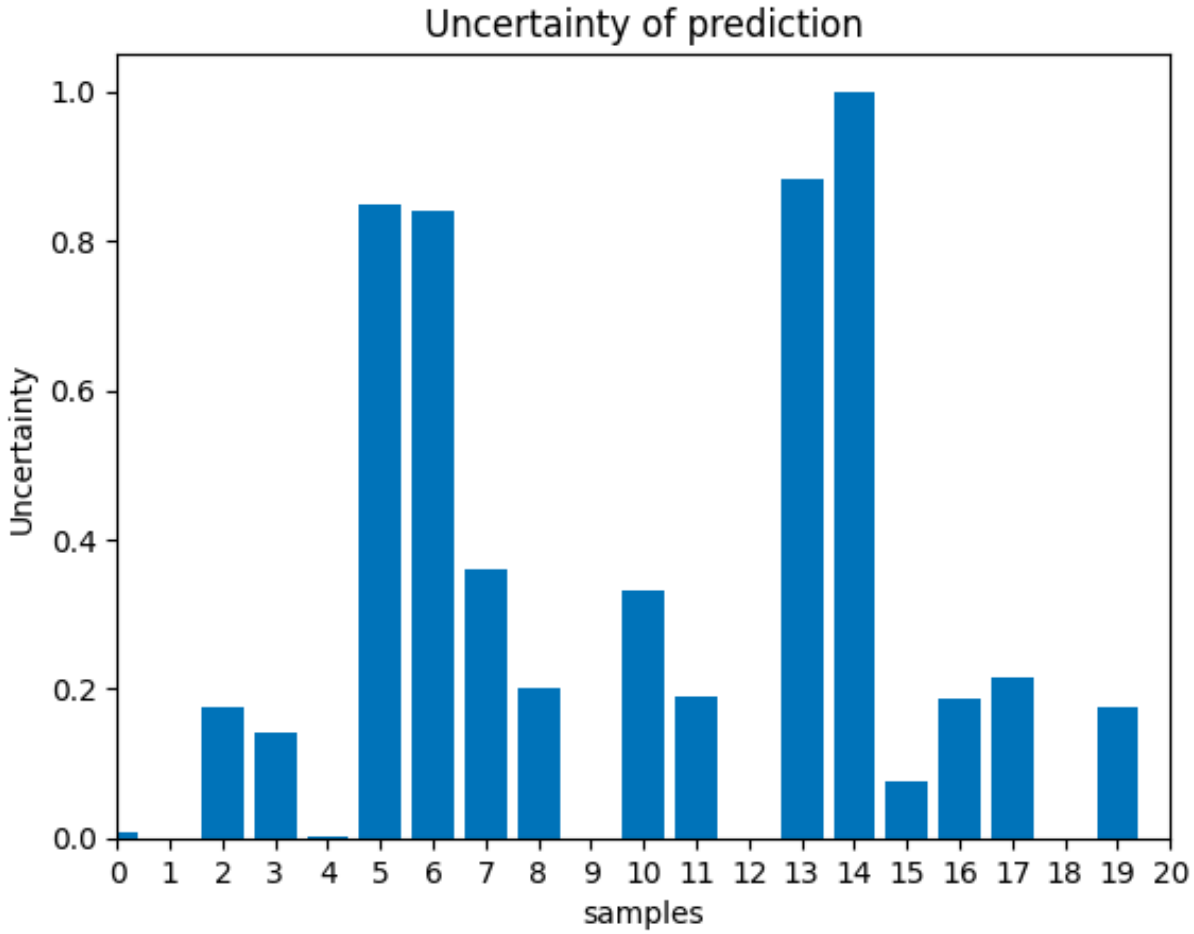


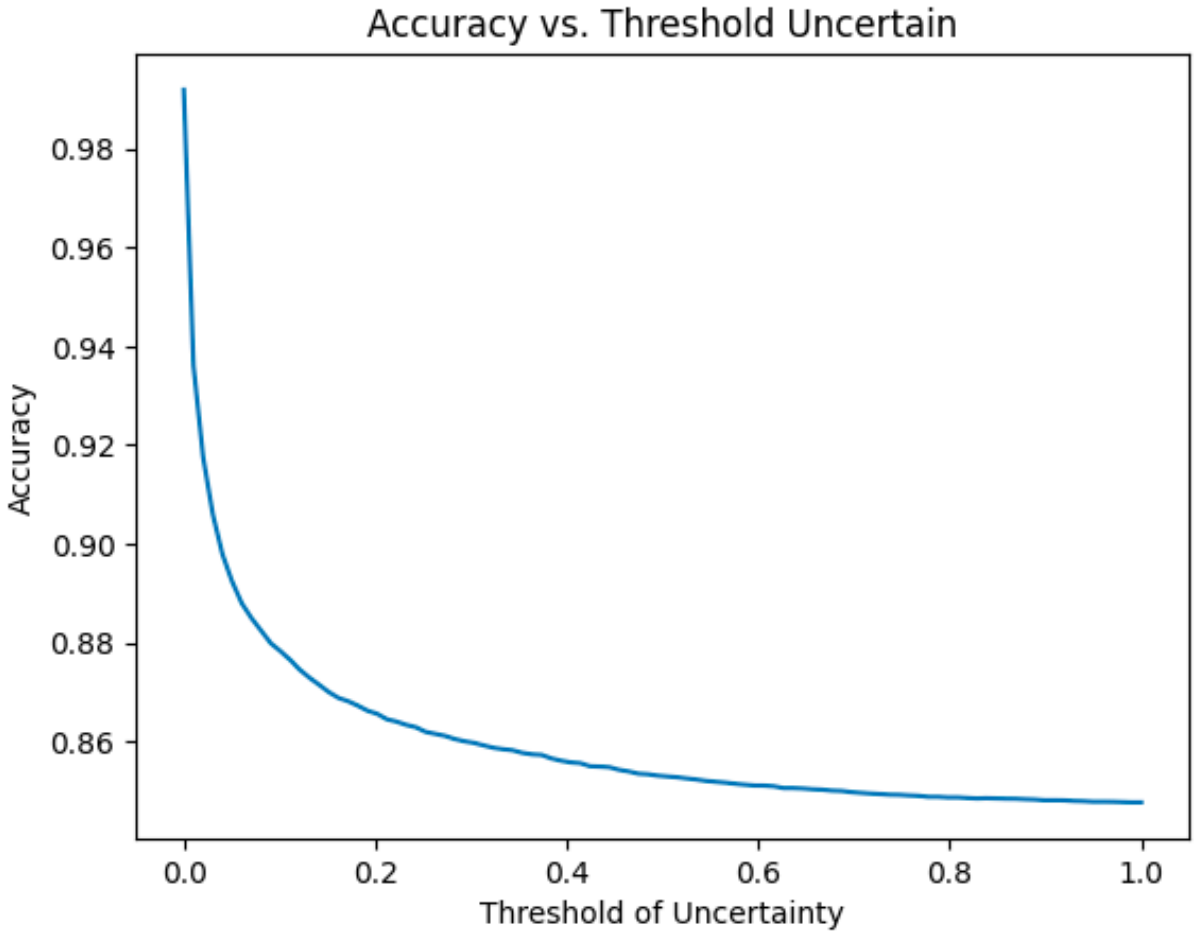
Figure 6. Corresponding uncertainties of prediction



From figure 5 and 6, when uncertainty of predition is very high, the prediction normally is wrong, while it is very low uncertainty, the prediction is normally correct (but sometimes it might be wrong, see failur cases).

Thus, a strategy is come up, if we setup a threshold of uncertainty manully, we can control the prediction accuracy. When uncertainty prediction is lower than threshold, we accept the results. Otherwise, we refuse the results. Figure 7 shows prediction vs. threshold of uncertainty.

Figure 7. Prediction accuracy vs. Threshold of Uncertainty



3.5 Failure cases

From figure 5 and 5, we find sample 0 has a very low prediction uncertainty, while the prediction is still wrong, this might happen when we choose a bad scale factor.

4. Conclusion

In this homework, I evaluated the performance of Gaussian Process Classification (GPC) for image classification tasks using the Fashion-MNIST and CIFAR-10 datasets, with a particular emphasis on integrating uncertainty into the classification process. The results revealed a strong performance of GPC in binary and multi-class classification contexts, especially with the Fashion-MNIST data.

In the binary classification task, GPC showed excellent performance on the Fashion-MNIST dataset but struggled with the CIFAR-10 dataset due to a simplistic preprocessing approach and high data dimensionality. This led me to focus subsequent investigations solely on the Fashion-MNIST dataset. In multi-class classification, the accuracy of the GPC model decreased as the number of classes increased. However, enhancing the number of training images improved

accuracy, albeit at the cost of additional computational resources.

Besides, the introduction of uncertainty into the classification process allowed for a more understanding of the model's predictions. By calculating the variance of the classification probability distribution, I assigned an uncertainty value to each prediction. Predictions with high uncertainty tended to be incorrect, prompting me to establish an uncertainty threshold. Predictions below this threshold were accepted, effectively improving overall prediction accuracy. Despite these promising results, the model had certain failure cases where low uncertainty corresponded to incorrect predictions. This could be attributed to an inappropriate choice of scale factor in the uncertainty calculations.

In conclusion, GPC is a powerful tool for introducing uncertainty, which can be widely applied in real-world situations that require reliable and stable output. For example, in autonomous driving or medical image analysis fields, predictions with high uncertainty can be flagged for additional review or analysis, thereby enhancing the safety and reliability of these computer vision systems.

5. References

Carl E. Rasmussen and Christopher K.I. Williams, “Gaussian Processes for Machine Learning”, MIT Press 2006

Blomqvist, K., Kaski, S. & Heinonen, M. (n.d.). *Deep convolutional Gaussian processes*.

Kapoor, A., Grauman, K., Urtasun, R. & Darrell, T. (2010). Gaussian Processes for Object Categorization. *International Journal of Computer Vision*, *88*(2), 169–188. https://doi.org/10.1007/s11263-009-0268-3

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Xiao, H., Rasul, K. & Vollgraf, R. (n.d.). *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*.

Krizhevsky, A., Nair, V. & Hinton, G. (n.d.). *CIFAR-10 (Canadian Institute for Advanced Research)*. http://www.cs.toronto.edu/~kriz/cifar.html