Add Uncertainty to Convolutional Neural Network by Using Gaussian Process

Andi Zhang [1]

Abstract

This work proposes a model that combines the convolutional neural network and gaussian process classifier in order to imbue the uncertainty of classification. Aided the uncertainty, the classifier will be able to reject the uncertain cases as opposed to returning a wrong answer, thereby enhancing the reliability of the system. During the experiment, our combined model was found to have an incredible "acute" property, which will enable the uncertainty to become much more useful than the original gaussian process classifier. This work also defines a "punitive" metric simulating the diagnosis scenario wherein the punishment of wrong classification is much greater than the rejection. In entirety, our new combined model is better than any single model in terms of both accuracy and our "punitive" metric.

1. Introduction

Uncertainty plays an important role in real-life scenarios. Our classifier can attain very high levels of accuracy, but some of the tasks have zero or little tolerance to errors. This requires our classifiers to be able to reject some "uncertain cases". For example, in the realm of automated medical diagnosis, we would like to output "reject" when we are uncertain, as opposed to outputting a wrong result, which may lead to an incalculable loss to a patient's life.

Convolutional neural networks (CNNs) do achieve state-of-the-art performance on image classification tasks, but they do not inspire confidence in their predictions. In contrast, while the gaussian process classifier (GPC) does not attain such a high level of performance on image classification tasks, the output can indeed predict both probability and the variance. To maximise the advantages of these two models, this work regards CNN as a feature extractor and integrates these features into the GPC, in order to obtain the predict probability and variance.

This work uses the GPC's output variance as uncertainty. In addition, we claim a "punitive" metric to simulate a practical scenario which enforces a bigger punishment on wrong classification than rejection. In this metric, our combined model yields much better results than any single model (CNN or GPC) on both MNIST and n-MNIST datasets.

Exploring the usage of this variance as uncertainty is another important aspect of this work. When conducting our experiment, we found that our combined model is much "acuter" than CNN or GPC in both probability and variance; this "acute" property inspires an idea to best utilise the variance. In fact, the idea is decide a threshold and cut off the cases trivially. However, owing to the "acute" property, the cut off on our combined model becomes much better than CNN or GPC only.

In essence, our work can be regarded as implicitly using CNN as a complex kernel, because it maps the raw features (pixels) into other features spaces (hidden layers), wherein the data points can be separated easier. Correspondingly, van der Wilk proposed a "Convolutional gaussian process" (2017) which explicitly uses a variant of CNN as the kernel of gaussian process.

2. Related Work

2.1. Convolutional Neural Network (CNN)

Convolutional neural networks (CNNs) (LeCun et al., 1998) are a class of artificial neural network, which is usually used in analyzing visual imagery.

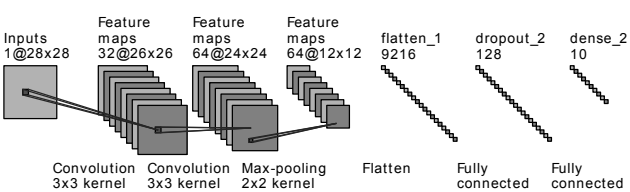


Figure 1. Structure of the CNN used in this work

As illustrated in Figure 1, a CNN comprises of several different layers (convolution, pooling, flatten, fully-connected), which denote the different kind of connections

[1]University of Cambridge, Cambridge, United Kingdom. Correspondence to: Andi Zhang <az381@cam.ac.uk>.

Assessment of LE49 Probabilistic Machine Learning, University of Cambridge, United Kingdom, 2017.

between the input and output. Please notice that the 'flatten_1', 'dropout_2', 'dense_2' refer to the names of layers in the program; however, we use these names here to indicate the output of the layers. Furthermore, the 'flatten_1', 'dropout_2' and 'dense_2' also signify our combined models in the remainder of this report.

2.2. Gaussian Process (GP)

A gaussian process is a stochastic process (a collection of random variables) such that every finite linear combination of these random variables is normally distributed, which can then be used as a prior probability distribution over functions in Bayesian inference. Gaussian process regression (GPR) refers to the inference of continuous values with a Gaussian process prior. The left hand side of Figure 2 depicts the posterior of GPR, where the blue line denotes mean and the light blue area represents the confidence interval.

With sigmoid, regression can be converted to classification, or the so-called gaussian process classification (GPC). In binomial classification, the likelihood obeys Bernolli distribution is as illustrated in the right hand side of Figure 2.

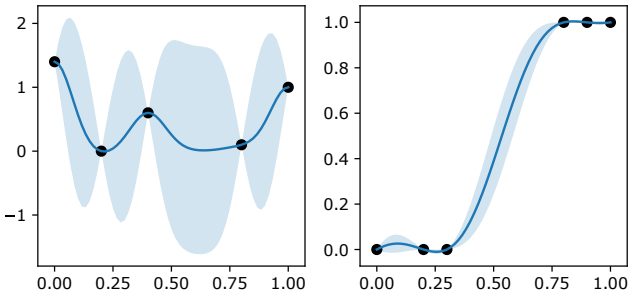


Figure 2. Posterior of GPR and GPC

For GPR, the combination of a GP prior with a Gaussian likelihood gives rise to a posterior which is again, a gaussian process. While the likelihood is non-Gaussian in the classification case, the posterior process can be approximated by a GP. Several approximation methods have been discussed by Nickisch and Rasmussen (2008).

3. Method

Feature Extraction As illustrated in Figure 1, 'flatten_1', 'dropout_2' and 'dense_2' refer to the output of the correspond layers in the convolutional neural network. This work attempts to use all of them as the features extracted from CNN and compares the performance of these features.

Classification After acquiring the features from CNN, we feed them into the GPC. In this work, we have excessive training data points for a normal GPC to handle, given that the complexities of approximate inference algorithms introduced by Rasmussen (2008) are $O(N^3)$ where N refers to the number of training cases (training datapoints). In order to address this problem, Hensman proposed "Scalable Variational Gaussian Process" (SVGP) (2015), cutting down the complexity to $O(NM^2)$ where M denotes the number of inducing points.

Hence this work uses SVGP as our gaussian process implementation and Bernoulli distribution as the likelihood through which, we obtain a binary GPC. When performing multinomial classification task in this work, we use the **one-vs-all** strategy to generalize our binary classifier.

Algorithm 1 Evaluation by "punitive" metric

Input: label y_i, prediction p_i, uncertainty u_i, ratio r, threshold t
Output: score s
Initialize $s = 0$.
for $i = 1$ **to** m **do**
  if $u_i < t$ **then**
    if $y_i = p_i$ **then**
      $s = s + 1$
    else
      $s = s - r$
    end if
  end if
end for

"Punitive" Metric Here, we claim a simple metric to simulate the "small tolerance scenario" based on the following rule: let r be the ratio of punishment to reward; for each case, if the prediction is correct, we gain a score of 1 as a reward; if the prediction is incorrect, we lose r scores as punishment. If we reject the prediction, we do not get any reward or punishment.

Set the threshold Inspired by the "acute" property of our combined models (as discussed in chapter 5.2, and depicted in Figure 9), it can be surmised that setting a threshold on variance is a very effective way to weed out most incorrect cases and retain as many correct cases as possible.

When combined with the threshold, a process to evaluate our trained model by "punitive" metric is illustrated in Algorithm 1. While this is a very simple strategy, it performs very well on our combined models for which, the "acute" property plays a very crucial role.

In practice, since we cannot get the test cases in advance, using the validation set to determine the threshold is a good choice.

Table 1. Results in accuracy

	CNN	RAW_GPC	FLATTEN_1	DROPOUT_2	DENSE_2
MNIST	0.9920	0.9778	0.9906	0.9919	0.9927
AWGN	0.9444	0.7689	0.9525	0.9552	0.9446
BLUR	0.9543	0.9585	0.8878	0.9499	0.9661
CONTRAST	0.7267	0.5659	0.8119	0.8081	0.6250

4. Experiment

4.1. Dataset

MNIST (Modified National Institute of Standards and Technology database) is a database of handwritten digits commonly used for image classification experiments. The MNIST database contains as many as 60,000 training images and 10,000 testing images, which are nomalized to 28×28 greyscale pixels.

n-MNIST (Basu et al., 2017) (noisy MNIST), as is shown in Figure 3, is generated using the MNIST dataset by adding

- additive white gaussian noise (AWGN)
- motion blur (BLUR)
- a combination of additive white gaussian noise and reduced contrast to the MNIST dataset (CONTRAST)

The size of n-MNIST is same as MNIST. In this work, we only use the MNIST database to train our classifier. Then we use both MNIST and n-MNIST to test our classifier, in order to evaluate the robustness of our model under the noise.

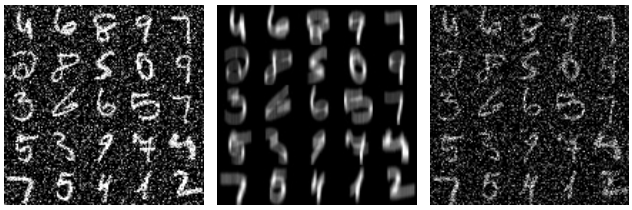


Figure 3. n-MNIST dataset

Hence, we now have the combinations of five methods and four datasets; for convenience, we use a letter to indicate the dataset (m for MNIST, a for AWGN, b for BLUR, c for CONTRAST). In addition, we use another letter or a number to indicate the method (c for CNN, r for raw-GPC, 128 for dropout_2, 10 for dense_2 and 9216 for flatten_1). In the published code, the names of variable names can be found, such as c128, mr, b9216, which stands for models.

4.2. Implementation

Here, we use the Keras (Chollet et al., 2015) MNIST example as our CNN implementation (as shown in Figure 1), from which, the features of input data are generated. Then, we transfer the features into the GPC. The implementation of GPC used within our experiment is GPFlow (Matthews et al., 2017), which is a Python package using Tensorflow (Abadi et al., 2015) as its computational backend. This work is based on van der Wilk's code (2017), which contains a parameter adjusted GPC model for MNIST.

4.3. Parameters

The hyper parameters of CNN are already illustrated in Figure 1. In addition, we use two dropout layers to avoid overfitting, which are omitted in Figure 1. The two dropout layers have a probability 0.25 and 0.5, which are inserted before 'flatten_1' and 'dense_1', respectively. The optimizer used in our CNN model is ADADELTA (Zeiler, 2012) whereas the batch-size is 128.

Table 2. Some hyper parameters of GPC (SVGP)

	DIM	INDUCING-PTS	NUM_ITER
RAW_GPC	784	750	20000
FLATTEN_1	9216	1000	15000
DROPOUT_2	128	100	30000
DENSE_2	10	10	30000

Some of the hyper parameters of GPC are depicted in Table 2. Additionally, the optimizer we used here is Adam (Kingma & Ba, 2014), and the batch-size is 200.

Further details of the parameters are shown in the published code of this work.

5. Results

5.1. A better performance in accuracy

Table 1 demonstrates the accuracy of our five trained models on the four datasets. On the original MNIST dataset, no major difference is found in the accuracy of all methods. From the second line of Table 1, it can be seen that the probability is labelled as red, if it is significantly lower than others. Hence, we can notice that CNN does not perform

well on CONTRAST; the raw GPC does not do well on AWGN and CONTRAST, flatten_1 is not good in BLUR, and dense_2 yields poor results on CONTRAST. Hence, the dropout_2 is the most balanced model in accuracy, which is not ridden with any significant drawbacks in each of the four datasets. As our models are shown to behave the worst on CONTRAST, we direct our focus on the CONTRAST dataset in the remainder of this work.

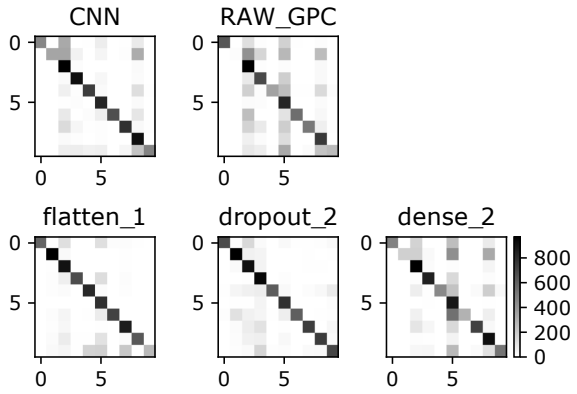


Figure 4. Confusion matrices of our models on CONTRAST

As is already known, flatten_1 and dropout_2 perform similarly in accuracy. However, based on the confusion matrices shown in Figure 4, we find that flatten_1 is not good at recognizing the digit '9'. Hence, we can deduce that dropout_2 has a better behaviour in F-measure.

5.2. A much more "acute" classifier

When using our trained classifier to do classification on CONTRAST, we find that in many cases, our combined models are "more confident" in its decision, i.e. they have a much higher probability and a much smaller variance than CNN and the raw GPC. Figure 5 illustrates an example of this phenomenon: for this digit '0', even though our CNN provides the right answer; its predict probability of '0' is not significantly higher than other results. The raw GPC is similar, hesitating between '0' and '2'. However, our combined model has a predict portability of 90% on the correct answer, demonstrating a more "acute" behaviour in comparison to other models. In fact the "CNN_GPC" in Figure 5 is dropout_2, since the two other combined models have very similar results in this case.

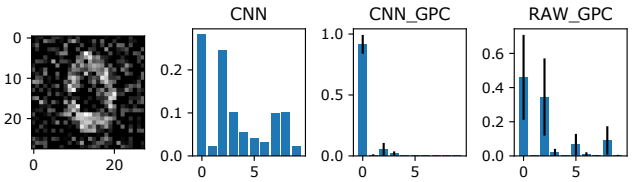


Figure 5. A '0' from CONTRAST and its results

In order to demonstrate that Figure 5 is not "cherry picking", we plot the histograms in Figure 6. The x axis of these plots in Figure 6 denotes the probability, and y axis shows how many cases have the probability as their highest predict probability. The blue bars represent correct cases and the yellow bars signify incorrect cases. Therefore, it is evident that our combined model is more "acute" - it has about 3500 cases with approximately 100% predict probability, thus getting the correct answer.

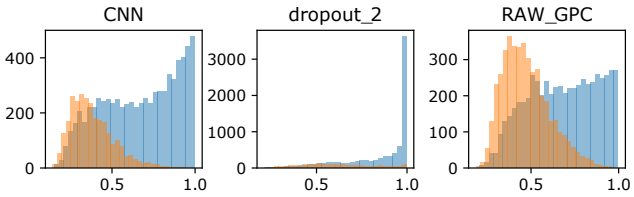


Figure 6. Histograms of probability

For comparison purposes, we also plot the histograms for flatten_1 and dense_2 in Figure 7. It can be seen that the dense_2 does not have the "acute" property any more, whereas the flatten_1 shows a similar behaviour as dropout_2. In addition, we think that flatten_1 has an advantage: it seldom provides incorrect cases when the predict probability is above 90%.

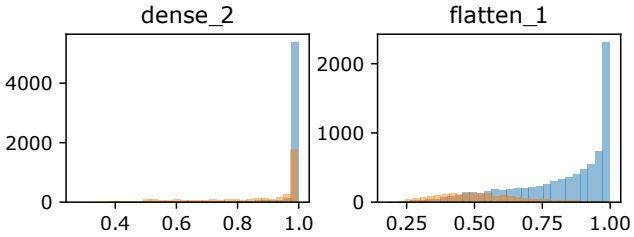
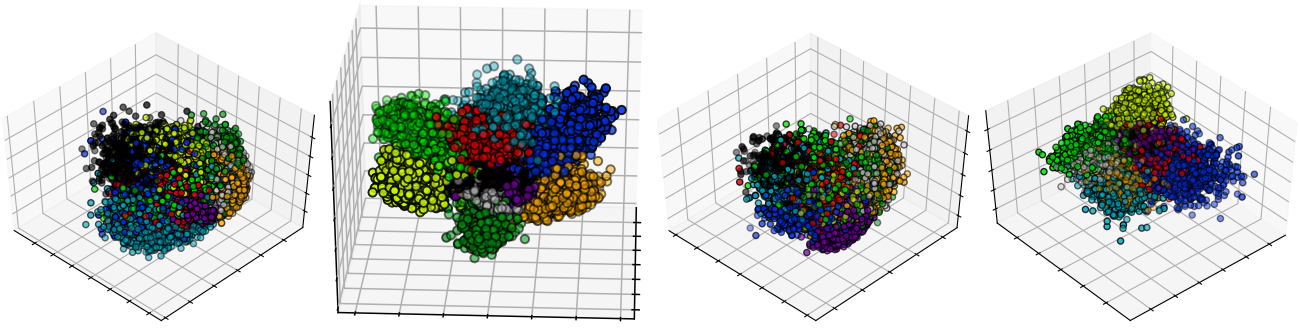


Figure 7. Histograms of probability (continue)

The property of "acute" is evidenced not merely in probability, but also in variance. Figure 9 illustrates the histograms of variance. Similar to the histograms of probability, the y axis represents the number of cases, whereas the x axis denotes variance. Again, we that find dropout_2 and flatten_1 is "acute" here, since the number of correct cases is extremely large when the variance is approximately 0.

There is another interesting pattern that must be noticed in Figure 9: the tips on the right hand side. These tips contain both correct and incorrect cases. Barring dense_2, the incorrect cases are clustered while the variance is larger than 0.18. This is a very strong evidence of the hypothesis that we can use this variance as the uncertainty.

More interestingly, the right hand side's tips of our combined models are found to be much more useful than the raw GPC's: in Figure 9, it is found that almost all cases in raw GPC are clustered towards the right hand side's tip. If we attempt to drop the cases with a high uncertainty, most of these cases will be dropped and the classifier will be ren-



(a) MNIST, raw (b) MNIST, dropout_2 (c) CONTRAST, raw (d) CONTRAST, dropout_2

Figure 8. Visualization of the features (by PCA)

Table 3. Results in our new metric (ratio=3), format: reject / not reject at all

	CNN	RAW_GPC	FLATTEN_1	DROPOUT_2	DENSE_2
MNIST	9680/9680	9374/9112	9708/9624	9740/9676	9722/9708
AWGN	7776/7776	4481/756	8373/8100	8620/8208	8194/7784
BLUR	8172/8172	8698/8340	6878/5512	8607/7996	8858/8644
CONTRAST	-932/-932	1770/-7364	5119/2476	4721/2324	718/-5000

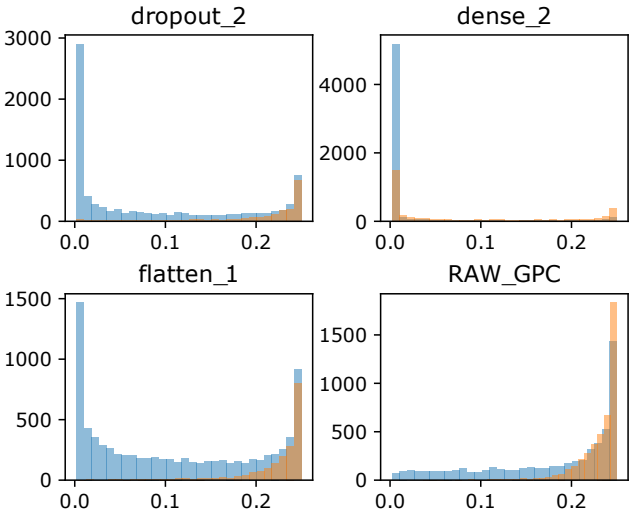


Figure 9. Histograms of variance

dered useless. However, in our dropout_2 and flatten_1, the correct cases are clustered at the left hand side. Hence if we choose to drop the cases with large uncertainty, **most of the correct cases will remain**.

5.3. A much better performance in "punitive" metric

We can call the tip on the right hand side as "tail". As mentioned in chapter 3, we set up a threshold and cut off the "tail" to obtain a higher mark in our "punitive" metric. Subsequently, the results are shown in Table 3, where a/b means: if the model does not reject any case, it will obtain b scores in the "punitive" metric; meanwhile if the model rejects the cases by selected threshold, it will get a scores in the metric. As illustrated in Table 3, our combined models yield a better performance than any single model in all the datasets. Moreover, the ratio is only 3 in this test. If the ratio is larger, the gap between the models will also increase. Hence in the real condition that have a lower tolerance in errors, our combined models will perform far better.

5.4. Explanation of "acute" property

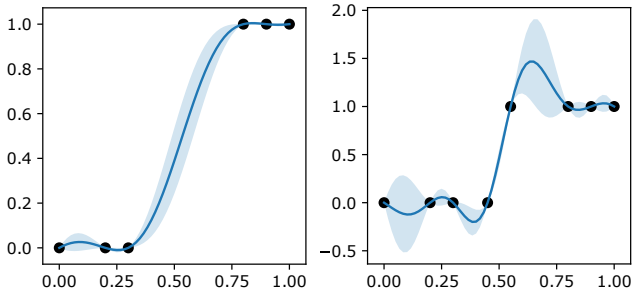


Figure 10. Schematic diagram of GPC

As shown above, the raw GPC does not have any "acute" property. The difference between raw GPC and combined models is the feature extraction process by CNN. This feature extraction process can be visualized in Figure 8. As illustrated in Figure 8, the cases with the same class are clustered: their distance becomes shorter and the margin between different classes increases. This change causes the "acute" property. To explain why this change improves

the behaviour of GPC, we plot Figure 10, which shows the posteriors of gaussian process with bernoulli likelihood - binary gaussian process classifiers. Figure 10 demonstrates that the variance of the model will reduce if the data points are more intensive and the margin between two classes is increased. Hence this study posits that CNN offers a "better arrangement" of the features, which subsequently leads to the "acute" property.

In essence, the "better arrangement" is a map from one feature space to another. From this viewpoint, CNN can be regard as an implicit kernel of the GPC. Hence our "acute property" is based on the complex kernel.

6. Conclusion

Dropout_2 should be regarded as the most optimal model in this work, since it does not have any shortcomings in accuracy, F-measure and our "Punitive metric" on the four datasets. Although flatten_1 is also good, the results suggest that flatten_1 is not efficient at handling BLUR dataset. In addition, it only takes a few minutes to train a dropout_2 model, instead of the 2 hours training cost by flatten_1.

"Punitive" metric, "acute" property and "cut off tail" operation is the core of this work: The "acute" property of our combined models inspire the idea of a "cut off tail" operation, which then yields a high performance on the "punitive" metric. The biggest accomplishment of this work is in finding the "acute" property of the combined models and claiming a suitable scenario for such models to work well with. In addition, we explain the rationale behind why our models can be so "acute" - in essence, the power of complex kernels.

In summation, this work is a good practice of combining the Bayesian method with the neural network.

7. Future Work

While our work has obtained satisfactory results, there are still some side-effects and unexamined factors that must be taken into consideration.

Noise and Blur According to Table 1, raw GPC does very poorly in AWGN and CONTRAST, which contains the additive white gaussian noise. However, the raw GPC performs very well in BLUR, which signifies a kind of morphology transformation.

A plausible explanation for this is that the raw GPC (with RBF kernel only) is sensitive to noise; hence, it performs badly. In contrast, the CNN feature extraction can filter the noise, enabling good performance to the combined models. However, this does not explain why RBF kernel remains sensitive to noise, but not to morphology transformation, which might be another story.

On the other hand, flatten_1 (whose feature has **9216** dimensions) is poor in BLUR, which indicates that the convolutional layers lose focus on the morphology, but the following full connect layers can capture it back. It is worth to explore what happened in this process, by some visualisation methods.

Compare with the convolutional gaussian process As mentioned above, this work essentially uses CNN as an implicit kernel of GPC. Recently, van der Wilk proposed the "convolutional gaussian process" (2017), which entails the use of CNN as an explicit kernel. It will be interesting to compare the performance of these two models.

Replication

All the code of this work is uploaded to Github:

`https://github.com/andiac/convgp`

To replicate the experiments, please follow the instructions of the repository.

Acknowledgement

I would like to thank Dr. Damon Wischik for his advice and support in this work.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dandelion, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Basu, Saikat, Karki, Manohar, Ganguly, Sangram, DiBiano, Robert, Mukhopadhyay, Supratik, Gayaka, Shreekant, Kannan, Rajgopal, and Nemani, Ramakrishna. Learning sparse feature representations using probabilistic quadtrees and deep belief nets. *Neural Processing Letters*, 45(3):855–867, 2017.

Chollet, François et al. Keras. https://github.com/keras-team/keras, 2015.

Hensman, James, Matthews, Alexander G de G, and Ghahramani, Zoubin. Scalable variational gaussian process classification. 2015.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Matthews, Alexander G. de G., van der Wilk, Mark, Nickson, Tom, Fujii, Keisuke., Boukouvalas, Alexis, León-Villagrá, Pablo, Ghahramani, Zoubin, and Hensman, James. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, apr 2017. URL http://jmlr.org/papers/v18/16-537.html.

Nickisch, Hannes and Rasmussen, Carl Edward. Approximations for binary gaussian process classification. *Journal of Machine Learning Research*, 9(Oct):2035–2078, 2008.

van der Wilk, Mark, Rasmussen, Carl Edward, and Hensman, James. Convolutional gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 2845–2854, 2017.

Zeiler, Matthew D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.