Using Gaussian Processes for Deep Neural Network Predictive Uncertainty Estimation

Dmitry Kazhdan

January 25, 2019

Abstract

Deep Neural Networks (DNNs) have achieved impressive performance on a wide range of tasks. However, their applicability is limited by their lack of uncertainty estimations, which is crucial in many domains. As a result, techniques for quantifying DNN uncertainty have been receiving increasing interest. This report describes a way of extracting uncertainty estimates from DNNs by combining them with Gaussian Process Classifiers, producing hybrid models. Results presented in this report demonstrate that these hybrid models achieve high predictive accuracy on normal samples, whilst reporting high uncertainty on noisy samples. Furthermore, this report demonstrates that uncertainty estimations of these hybrid models may be used for adversarial sample detection.

1 Introduction

Machine Learning (ML) has achieved groundbreaking results in a wide variety of fields, and is currently one of the most active research areas of Artificial Intelligence. One particular subfield of ML that is experiencing a surge of interest is Deep Learning (DL), which uses Deep Neural Network (DNN) models for learning data representations, and has been applied in numerous fields including: image recognition [10], speech recognition [8] and bioinformatics [21].

Unfortunately, applicability of DNNs has been limited by their inability to provide uncertainty estimates of their predictions. The lack of uncertainty estimates, coupled with uninterpretability of such models, often leads to a lack of confidence in their results, which is crucial in many safety-critical applications, such as healthcare, or self-driving cars.

Furthermore, DNNs have been shown to be susceptible to adversarial attacks [28], in which an attacker is able to trick a DNN into producing erroneous output, by applying human-imperceptible perturbations to input data. This is a major security concern that severely limits DNN applicability. Existing work suggests that some types of these adversarial attacks may be prevented by models which can provide uncertainty estimates [18].

Work described in this report aims to address these issues by combining a DNN with a Gaussian Process Classifier (GPC), in order to develop an *Augmented Deep Neural Network* (ADNN) that is capable of quantifying its predictive uncertainty. The contributions of this report are the following:

1. Creating an ADNN that is capable of producing classification predictions with uncertainty estimates
2. Verifying that this model is capable of achieving high classification accuracy on normal data
3. Verifying that this model reports higher uncertainty on noisy data

4. Verifying that this model reports higher uncertainty on adversarial data

Section 2 introduces the relevant background material. Section 3 discusses some of the related work. Section 4 describes the implementation of the ADNN model. Section 5 discusses the model evaluation. Possible directions for future work are discussed in Section 6. Finally, Section 7 gives some concluding remarks.

2 Background

2.1 Gaussian Process Classification

A Gaussian Process (GP) is a stochastic process, such that every finite collection of random variables in this process has a multivariate normal distribution. A GP defines a prior over functions (i.e. it is a *non-parametric* method), which can be converted into a posterior over functions given the input data. GPs are a Bayesian approach, meaning that they provide uncertainty measures with their output. GP classification may be achieved by mapping the values of the underlying modelled functions (referred to as *latent functions*) into a unit interval using sigmoidal inverse-link transformations, such that the transformed function values represent probabilities of class membership for a given point. The variances of the latent functions at a given point reflect the model's classification uncertainty. GPs are flexible and efficient, providing good results even with small datasets. These properties make them a popular choice for regression and classification tasks that involve relatively small datasets and require uncertainty estimation. Further details regarding GPs and GPCs are outside the scope of this report, but may be found in [27].

2.2 Adversarial Samples

As discussed in Section 1, adversarial samples typically refer to legitimate NN input that has been altered in a way that is imperceptible to the human eye, but can cause a trained DNN to produce incorrect output. Adversarial samples are typically considered in the context of image-recognition tasks. One well-know, simple and efficient adversarial method is the *Fast Gradient Sign Method* (FGSM) [7]. This method perturbs an input image by performing a one-step update of every image pixel in the direction determined by the gradient of the corresponding DNN. The *Basic Iterative Method* (BIM) is a powerful extension of FGSM, which runs the FGSM update for several iterations, instead of one [14]. Another sophisticated adversarial attack is the *DeepFool* attack [23]. This attack aims to minimise the changes required to push a DNN across decision boundaries, by iteratively perturbing an input image and linearlizing the classification space to move it to the closest decision boundary.

Vulnerability to such samples is a major security concern, given that it affects all applications that rely on DNNs. As a result, a plethora of attack and defence strategies has been explored by existing work (a comprehensive overview can be found in [2, 28]).

3 Related Work

A wide range of existing work has explored ways of making DNNs more Bayesian, in order to extract predictive uncertainty estimates from them [5, 15, 24]. This is typically achieved by treating parameters (weights) of a DNN as random variables, and computing suitable probability distributions over them. Whilst achieving high performance on many tasks, many of these approaches rely on complicated DNN re-training procedures or significant structural changes to the original DNN architecture, and often have a high computational cost.

Existing work has also demonstrated the synergy between GPs and DNNs. For instance, work in [12] combined GPs and DNNs by using DNNs for the mean functions of GPs. Work in [1] implemented a hybrid model similar to the one introduced in this report, in which a GP is used instead of a softmax layer of a CNN. However, their approach requires training the whole hybrid architecture from scratch in an end-to-end fashion, and is considerably more complex than the approach proposed in this report.

Work in [4] describes an approach for detecting adversarial samples using predictive uncertainties. This approach relies on dropout neural networks and kernel density estimators for obtaining uncertainty estimates. Similarly to the work described here, they rely on higher-layer DNN features when computing uncertainty estimates. Work in [9] describes the construction of *High-Confidence-Low-Uncertainty* adversarial samples, which simultaneously maximize confidence and minimize uncertainty, and evaluates them on plain GPs and Dropout DNNs.

4 Augmented Deep Neural Networks

4.1 Model Architecture

The implementation described in this report focuses on image-recognition Convolutional Neural Networks (CNNs), however, it can be seamlessly extended to many other DNN architectures. An ADNN model is constructed by taking a trained CNN, and replacing its output softmax layer with a GPC. The GPC is initially trained on the high-level features extracted by the penultimate CNN layer using a subset of the CNN's training data. Thus, during classification of an image, the modified CNN is effectively used as a feature extractor for the GPC, which then classifies the given input and reports uncertainty estimates. The model architecture is summarised in Figure 1. The *Decision Module* determines whether the given sample should be flagged as *unknown*, given it's predictive probability and uncertainty.

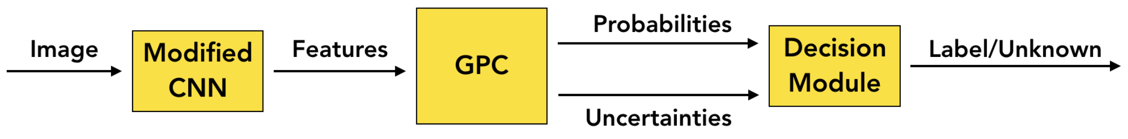


Figure 1: ADNN Model Architecture

The above approach offers several advantages:

- It requires minimal modifications to the original CNN (only removing the last layer)
- It does not require CNN re-training (the CNN weights remain unaltered)
- It is fast to train (the GPC is trained on considerably fewer samples than the original CNN)
- It is simple to implement

4.2 Classification

The decision module employs a simple thresholding strategy for flagging an input sample as *unknown*. For a given input sample, the GPC outputs a classification probability vector **p**, containing class probabilities $(p_1, ..., p_C)$ for the given sample (obtained from the latent functions, as discussed in Section 1), and a corresponding vector of latent function variances $\mathbf{v} = (v_1, ..., v_C)$, which represent the uncertainty of the probability estimates. The sample is flagged as unknown if the largest probability score is below a set threshold μ, or if the variance of the latent function with the largest probability score is above a set threshold σ, as shown in Algorithm 1.

Algorithm 1 Classification($\boldsymbol{p}$, $\boldsymbol{v}$)

```
1: $m \leftarrow \underset{i}{max}(p_i)$          ▷ Obtain the largest probability index
2: if  $p_m < \mu$ or $v_m > \sigma$  then
3:     return $-1$                               ▷ Flag as unknown
4: else
5:     return $m^{th}$ label                     ▷ Return class label
6: end if
```

Many high-dimensional datasets, such as images, are believed to lie on a low-dimensional manifold [17]. Furthermore, work in [6] demonstrated that the deeper layers of a NN extract high-level features, and thus provide much simpler, 'unwrapped' data manifolds, compared to those of the input space. Thus, we expect our GPC to be able to learn the data manifold using only a small subset of the original CNN training samples, and be able to classify new test cases with a low uncertainty and a high predictive probability. Hence, the value for μ is expected to be close to 1, and the value for σ is expected to be close to 0.

4.3 GPC

The GPC was implemented using GPFlow [19] (a package for building GP models in python, using TensorFlow), which uses variational inference for approximations [11]. In practice, many design decisions associated with the GPC involved considering sets of options with no obvious optimal choice. Thus, it was necessary to experiment with different configurations in order to determine which ones give better performance. The configurations investigated in this report are discussed below.

- **GPC Type**: Both the Variational Gaussian Approximation (VGP) and the Sparse Variational Gaussian Approximation (SVGP) offered by GPFlow were considered. SVGP resulted in considerable reductions in training time (as expected), whilst not impacting overall performance. Thus SVGP was chosen as the GPC type.
- **Kernel Type**: The RBF, Matern52 and Linear kernels were considered (periodic and constant kernels were deemed unsuitable for this task). The Linear kernel showed comparatively poor performance, whilst the RBF and Matern52 kernels demonstrated similar performance results. However, RBF performance was slightly less stable, and thus the Matern52 kernel was selected in the end.
- **Kernel Noise**: Adding a White kernel harmed overall performance, and was thus avoided.
- **Data Features and Automatic Relevance Determination (ARD)**: The relative importance of different features was investigated by running the GPC with random subsets of features, with and without separate kernel parameters for each feature. Results showed that the GPC was

capable of achieving high predictive accuracy even when using a small subset of the available features. However, performance improvements with ARD were unsubstantial based on preliminary results. Thus, for simplicity, the chosen kernel used all of the available features and did not use ARD.

Overall, the GPC used for this report was an SVGP classifier using the Matern52 Kernel with all of the available features and without ARD.

4.4 Ensemble Extensions

Section 4.3 demonstrated that there are many potential GPC configurations that all achieve high predictive accuracy on normal samples. Thus, a possible extension of the above approach is to incorporate multiple GPCs with different configurations into a single ADNN model, in order to improve the model robustness. Due to time constraints and a lack of convincing preliminary results, this report focused on the single GPC case. However, a discussion of the potential merits of the ensemble GPC approach is given in Appendix A.

5 Evaluation

The approach described in the previous section was evaluated using the CNN architecture found in [22], trained and tested using the MNIST dataset [16], and fitted with the GPC described in Section 4.3. Uncertainty of a prediction was defined to be the variance of the latent function with the highest predictive probability score. Given the limited computational resources available for this report, all experiments dealt with relatively small GPC training dataset sizes. Experimenting with larger training dataset sizes could thus be the subject of future work.

Firstly, the effect of the GPC training dataset size on the predictive accuracy and predictive uncertainty of the model was investigated by evaluating the ADNN on the test set, with the GPC classifier trained on randomly picked training data subsets of sizes 50, 100, 200 and 400. This experimental setup was re-run five times. The results are shown in Figure 2 and Figure 3.

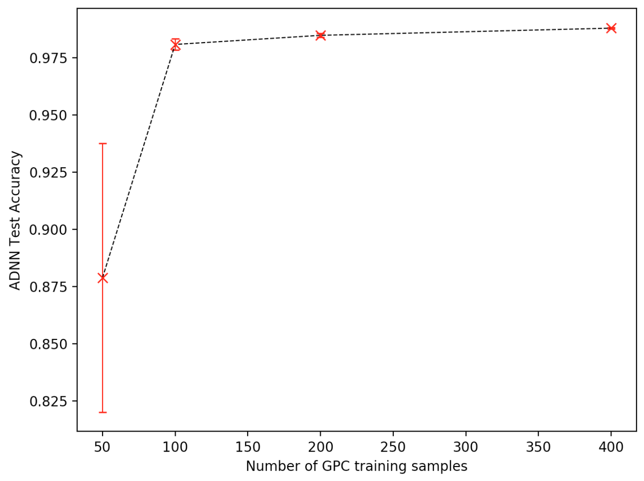


Figure 2: Predictive accuracy of the ADNN model

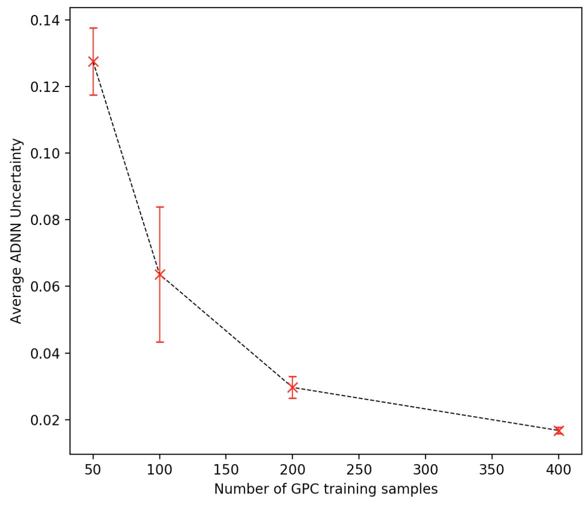


Figure 3: Predictive uncertainty of the ADNN model (averaged over all test sample predictions)

Overall, test accuracy of the ADNN increased with the number of GPC training samples, and the average test sample predictive uncertainty fell, indicating that the ADNN became more confident and more accurate in its predictions, given more GPC training data. As anticipated, the ADNN was capable of achieving high accuracies even with small GPC training sets, achieving test accuracies of over 97% using as few as 200 GPC training samples. In all other experiments discussed below, the GPC was trained using 400 randomly-picked clean training samples.

Next, the model's uncertainty was evaluated on abnormal input. The model was tested on the N-MNIST dataset [25], which consists of three different perturbed versions of the MNIST dataset, altered by additive white gaussian noise (AWGN), motion blur, and a combination of AWGN and reduced contrast. In every case, the GPC reported the same uncertainty value for all test samples (indicating that all test samples reached the maximum possible GPC uncertainty). To account for randomness in GPC training data selection, the experimental setup was re-run 5 times. The average test sample uncertainty (mean ± standard deviation) for the three datasets is given in Table 1.

N-MNIST Dataset	**Uncertainty**
AWGN	1.48 ± 0.03
Motion Blur	1.51 ± 0.04
Reduced Constrast and AWGN	1.51 ± 0.07

Table 1: N-MNIST Test Sample Uncertainties

Table 1 shows that the average uncertainty of N-MNIST samples was considerably higher than for normal samples. Furthermore, whilst the original CNN achieved predictive accuracies of over 60% on all three datasets, ADNN's accuracy was around 10% in all three cases (roughly as good as random guessing). These results demonstrate that the ADNN does not extrapolate beyond it's experience as confidently as an ordinary CNN, and is much more sensitive to changes in the input sample distribution.

Finally, the ADNN was evaluated on adversarial samples generated from the MNIST test samples with the CleverHans [26] toolbox, by applying the FGSM, BIM and DeepFool attacks to the original CNN model. The largest prediction probability and corresponding uncertainty distributions of the samples for the three attacks are shown in Figure 4 and Figure 5 below. The boxplot whiskers were set to extend to the 5th and 95th percentiles of the given data. FGSM was run with $\epsilon = 0.3, 0.4, 0.5$. BIM was run with $\epsilon = 0.05$ and iteration number $i = 5, 20, 50$. DeepFool was run with the maximum iteration number $i = 10, 50, 100$.

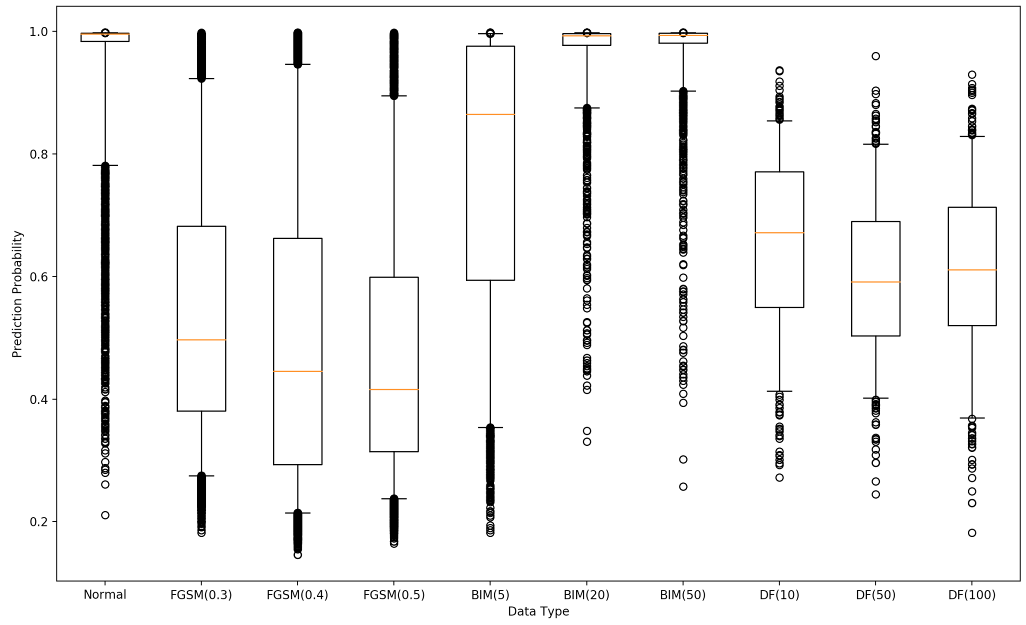


Figure 4: Distribution of the largest class probabilities (obtained by extracting the largest class probability from every test sample)

Overall, both predictive probabilities and uncertainty measures are important for recognising adversarial samples. FGSM and DeepFool attacks reported a very high confidence, but also a low predictive probability. This might indicate that these adversarial samples were close to the training data points, where the latent functions had a low variance, but also close to decision boundaries, where the latent function means were close together, thus producing similar predictive probability estimates. On the other hand, BIM attacks with 20 and 50 iterations reported a high predictive confidence and high uncertainty, possibly indicating that samples generated with these attacks were far from the training data, where the latent functions have a high variance.

These results demonstrate that a combination of uncertainty and predictive probability estimates of the ADNN can be used to detect abnormal samples which are either perturbed by various forms of noise, or by adversarial perturbations.

6 Future Work

Firstly, future work may explore more intelligent strategies for training data subset selection. For instance, using *active learning* [13] techniques for dynamically selecting more informative training data points for the GPC will likely lead to better performance compared to the simple random selection strategy used in this report, whilst still being computationally feasible. Informed training data selection strategies could potentially allow the GPC to learn manifold decision boundaries more accurately, making it even more robust to abnormal samples and reducing the amount of normal sample outliers

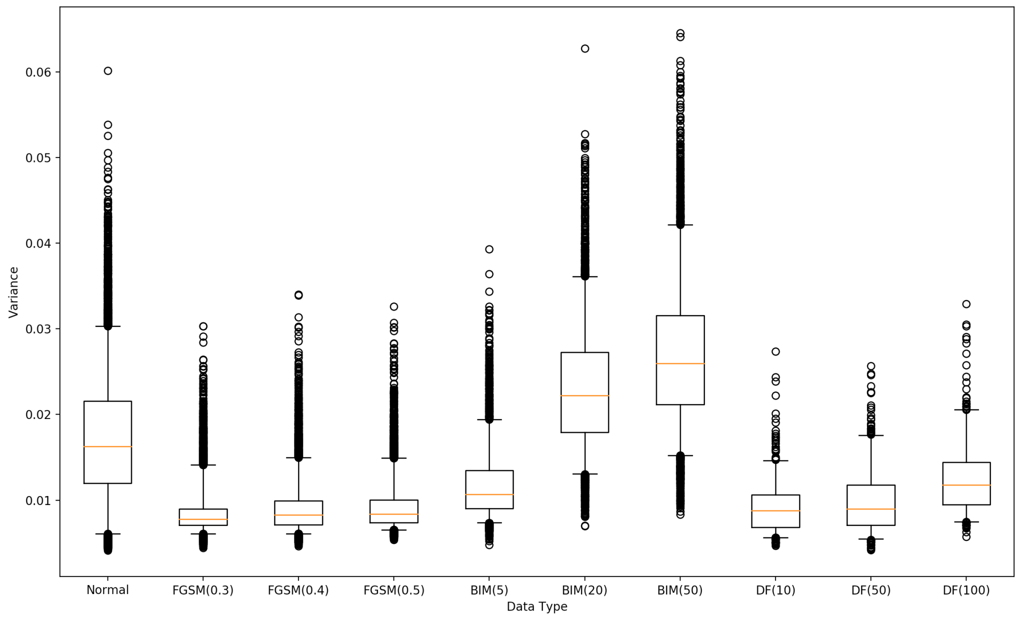


Figure 5: Distribution of predictive uncertainty (obtained by extracting the predictive uncertainty of every test sample)

classified with a low probability or low confidence.

Secondly, future work may focus on using ADNN-specific adversarial samples, instead of using adversarial samples generated for the original CNN. Unfortunately, crafting model-specific adversarial samples for hybrid models, such as the one discussed in this report, is a non-trivial task, and was thus outside the scope of this report. However, applying adversarial attacks that are specifically tailored to ADNN models would make adversarial detection results more meaningful.

Another direction for future work is investigating the defensibility of ADNNs. Many of the design choices for the proposed system, such as the chosen kernels, training data features, training data points etc. do not heavily impact performance on normal data. However, by exploring their effect on data lying off the manifold, it may be possible to identify sets of GPC configurations that differ in their off-manifold behaviour. This can then be used to improve the model defensibility (e.g. by randomly picking a GPC configuration during model construction).

Finally, the approach described in this report did not rely on assumptions that were specific to image-recognition or CNNs. Thus, future work may focus on exploring the generalisability of this approach by applying it to more tasks and using different types of DNNs, thus further demonstrating its wide applicability.

7 Conclusions

To conclude, this report successfully demonstrated how GPCs can be used for obtaining uncertainty estimates from DNN predictions. This was achieved by combining GPCs with DNNs in order to develop an ADNN model that produced classification results enhanced with uncertainty estimations. This ADNN model was shown to achieve a high predictive accuracy on normal test samples, whilst reporting lower prediction estimates and/or higher uncertainty on abnormal samples, such as adversarial samples or samples perturbed by noise.

References

[1] John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. "Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks". In: *arXiv preprint arXiv:1707.02476* (2017).

[2] Anirban Chakraborty et al. "Adversarial Attacks and Defences: A Survey". In: *CoRR* abs/1810.00069 (2018). arXiv: 1810.00069. URL: http://arxiv.org/abs/1810.00069.

[3] Thomas G Dietterich. "Ensemble methods in machine learning". In: *International workshop on multiple classifier systems.* Springer. 2000, pp. 1–15.

[4] Reuben Feinman et al. "Detecting adversarial samples from artifacts". In: *arXiv preprint arXiv:1703.00410* (2017).

[5] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning.* 2016, pp. 1050–1059.

[6] Jacob R. Gardner et al. *Deep Manifold Traversal: Changing Labels with Convolutional Features.* 2015. arXiv: 1511.06421 [cs.LG].

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and harnessing adversarial examples. CoRR (2015).*

[8] A. Graves, A. Mohamed, and G. Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing.* May 2013, pp. 6645–6649. DOI: 10.1109/ICASSP.2013.6638947.

[9] Kathrin Grosse et al. "The Limitations of Model Uncertainty in Adversarial Settings". In: *CoRR* abs/1812.02606 (2018). arXiv: 1812.02606. URL: http://arxiv.org/abs/1812.02606.

[10] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[11] James Hensman, Alexander Matthews, and Zoubin Ghahramani. *Scalable variational Gaussian process classification.* 2015.

[12] Tomoharu Iwata and Zoubin Ghahramani. "Improving Output Uncertainty Estimation and Generalization in Deep Learning via Neural Network Gaussian Processes". In: *arXiv preprint arXiv:1707.05922* (2017).

[13] Ashish Kapoor et al. "Active learning with gaussian processes for object categorization". In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on.* IEEE. 2007, pp. 1–8.

[14] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *CoRR* abs/1607.02533 (2016). arXiv: 1607.02533. URL: http://arxiv.org/abs/1607.02533.

[15] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in Neural Information Processing Systems.* 2017, pp. 6402–6413.

[16] Yann Lecun and Corinna Cortes. *The MNIST database of handwritten digits.* URL: http://yann.lecun.com/exdb/mnist/.

[17] John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[18] Yingzhen Li and Yarin Gal. *Dropout Inference in Bayesian Neural Networks with Alpha-divergences*. 2017. arXiv: 1703.02914 [cs.LG].

[19] Alexander G. de G. Matthews et al. "GPflow: A Gaussian process library using TensorFlow". In: *Journal of Machine Learning Research* 18.40 (Apr. 2017), pp. 1–6. URL: http://jmlr.org/papers/v18/16-537.html.

[20] Dongyu Meng and Hao Chen. "MagNet: a Two-Pronged Defense against Adversarial Examples". In: *CoRR* abs/1705.09064 (2017). arXiv: 1705.09064. URL: http://arxiv.org/abs/1705.09064.

[21] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. "Deep learning in bioinformatics". In: *Briefings in Bioinformatics* 18.5 (2017), pp. 851–869. DOI: 10.1093/bib/bbw068. eprint: /oup/backfile/content_public/journal/bib/18/5/10.1093_bib_bbw068/2/bbw068.pdf. URL: http://dx.doi.org/10.1093/bib/bbw068.

[22] *MNIST CNN*. URL: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py.

[23] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "DeepFool: a simple and accurate method to fool deep neural networks". In: *CoRR* abs/1511.04599 (2015). arXiv: 1511.04599. URL: http://arxiv.org/abs/1511.04599.

[24] Radford M. Neal. *Bayesian Learning for Neural Networks*. Berlin, Heidelberg: Springer-Verlag, 1996. ISBN: 0387947248.

[25] Garrick Orchard et al. "Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades". In: *Frontiers in Neuroscience* 9 (2015), p. 437. ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00437. URL: https://www.frontiersin.org/article/10.3389/fnins.2015.00437.

[26] Nicolas Papernot et al. "Technical Report on the CleverHans v2.1.0 Adversarial Examples Library". In: *arXiv preprint arXiv:1610.00768* (2018).

[27] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN: 026218253X.

[28] Xiaoyong Yuan et al. "Adversarial Examples: Attacks and Defenses for Deep Learning". In: *CoRR* abs/1712.07107 (2017). arXiv: 1712.07107. URL: http://arxiv.org/abs/1712.07107.

A Gaussian Process Classifier Ensembles

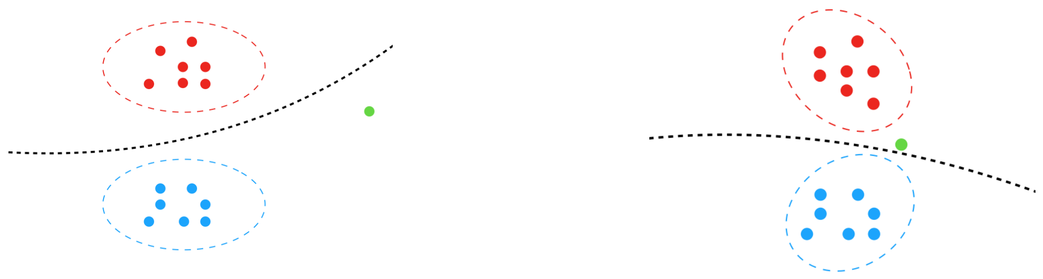
A.1 Motivation

A.1.1 Ensemble Learning

Ensemble learning is a ML approach in which multiple learners are trained to solve the same problem, constructing and combining a set of hypotheses, instead of learning a single one [3]. A common ensemble learning strategy is Bootstrap Aggregation (often abbreviated as *bagging*), in which a set of models are trained independently in parallel, typically using randomly drawn subsets of the training data (thus promoting model variance). Predictions of individual models are combined using a voting scheme, such as majority voting, to arrive to the final prediction.

A.1.2 Adversarial Sample Interpretation

As discussed in Section 4.2, many high-dimensional datasets are believed to lie on a low-dimensional manifold. Furthermore, work in [7] demonstrates that DNNs perform correctly only near the small manifold of training data. Thus, adversarial samples are commonly assumed to lie off of the data manifold. Existing work [4, 20] typically divides adversarial samples into two categories: samples that lie far off the manifold (exploiting poor generalizability to unseen portions of the input space), and samples that lie close to decision boundaries in areas of low-confidence. These cases are summarized in Figure 6a and Figure 6b, respectively.



(a) Adversarial sample far from the manifold

(b) Adversarial sample close to a decision boundary

Figure 6: Off-manifold adversarial samples. Red and blue dots and dashed lines represent data points and manifolds (respectively) of two different classes, the black dashed line represents the decision boundary, and the green dot represents an off-manifold adversarial sample.

A.1.3 Ensemble Classifiers

A GPC of an ADNN relies on only a small amount of training samples, and thus risks producing poor estimates for off-manifold adversarial samples. Unfortunately, using more training samples is computationally costly and may not lead to improvements in off-manifold behaviour. Instead, a *Gaussian Process Ensemble* (GPE) can be used for better adversarial sample detection. Such a GPE will consist of several GPC models, varying in the training samples, kernels, input features etc. Training a set of models in parallel on several sets of training data is considerably faster than training a single model on all of that data, thus avoiding the computational overhead of using extra samples. At the same time, introducing variance to the individual models means that the GPCs will learn differing decision boundaries, which may make the ensemble collectively more robust to adversarial samples, as shown in Figure 7.

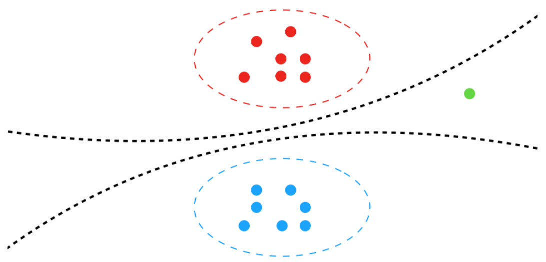


Figure 7: The adversarial sample may be detected by noticing an inconsistency between the two decision boundary classifications

A.2 Preliminary Results

The GPE functionality described above was implemented as part of this report. Furthermore, a simple setup consisting of 3 GPCs with different kernel types using a majority voting strategy was evaluated and resulted in slight improvements in detection rates of BMI adversarial samples (FGSM and DeepFool detection rates remained the same). However, producing a convincing proof of the validity of the assumptions discussed in the previous section would involve careful experimentation with a range of GPE configurations and adversarial samples, in order to ensure that the results are not arising due to inherent noisiness of the training and testing processes. Due to time constraints, the utility of GPEs was thus not explored further in this report, but may be the subject of future work.