Gaussian Processes and CNNs: The utility of Prediction Variance

Joshua Send [1]

Abstract

Convolutional neural networks (CNNs) have long been used to achieve high classification accuracies on a huge variety of tasks. However, they struggle provide information beyond class probabilities, which have to be accepted at face value. Gaussian processes (GPs) are fully Bayesian constructs which can be used to predict both class probabilities and variances for those probabilities, indicating confidence. Using MNIST and related datasets, this paper investigates the effectiveness of combining GPs with CNNs, considering variance for interpretation of results, and as a tool for merging predictions from the base CNN and the GP. This merging process is shown to slightly increase test set accuracy in non-adversarial settings, and points towards future research directions.

1. Introduction

Convolutional Neural Nets (CNNs) have seen a steep rise in use for a wide range of computer vision and machine learning tasks since Krizhevsky's success at the ImageNet challenge in 2012 (Krizhevsky et al., 2012). Given enough training data, they excel in domain specific applications and are efficient to train, but may confidently mis-predict classes, provide no feedback to users about variability of predictions, and have been shown to be vulnerable to adversarial attacks (Szegedy et al., 2013). On the other hand, Gaussian processes are capable of classification and providing a variance about the prediction. This can be interpreted as a confidence in the probability of a prediction.

This work combines CNNs with Gaussian Processes (GPs), using features extracted by the CNN as input to a GP. This combination takes advantage of neural networks' ability to approximate high dimensional data efficiently and reduce it to a lower dimensional representation; GPs can then produce Bayesian uncertainty estimates along with predictions.

Focusing on MNIST datasets, this work contributes an analysis of variance and statistically significant predictions, proposes a method for ensembling CNN *softmax* and GP predictions steered by prediction variance, and discusses promising future research directions.

2. Background

2.1. Convolutional Neural Nets

CNNs are normally used for classification. The following *softmax* activation is applied to the final layer to convert activations into probabilities:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k=m} e^{z_j}}$$

Combined with categorical-cross entropy loss during training, the resulting values between 0 and 1 from each of the m units represent the posterior distribution over m classes (Bridle, 1990). In other words, we obtain the probability of any class using a fully connected softmax layer trained this way. Note that this method reinforces large values and squashes small ones due to the exponentiation, exaggerating differences between classes and not taking into account absolute activation prior to the softmax.

2.2. Gaussian Processes

Gaussian processes are non-parametrized models of functions, defined by a mean and covariance function. The latter determines how related two points in space are – generally the further away, the less related. The covariance function is built out of a set of *kernels*, and generates a covariance matrix. By incorporating the mean and covariance functions, along with training data, GPs predict both a mean and a variance when given input points. For a simplified example see Figure 1.

By design, trained Gaussian processes have lower variance around known data, while moving away from these points results in much higher uncertainty. This is the property we are most interested in exploiting: when the variance is high, we are in a relatively unknown region of the input space and should put little trust into the predictions.

Another key benefit of GPs is that little hyperparameter tuning is needed – since the model is fully Bayesian, pa-

[1] University of Cambridge. Correspondence to: Joshua Send <js2173@cam.ac.uk>.

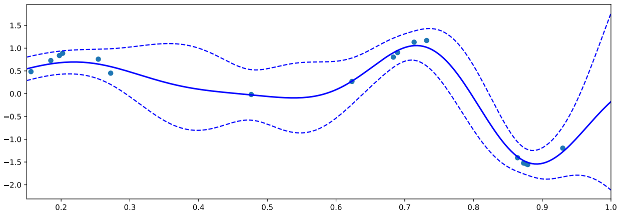


Figure 1. A simple 1-D GP regression. The dashed lines represent $\pm 2\sigma$ from the mean, and blue points are training data. Notice how the variance increases away from known datapoints.

rameters can be learned directly by maximizing marginal likelihood. However, traditional GPs are costly to train – covariance matrices are $O(n^2)$ in size, and inversion and decomposition operations have an $O(n^3)$ cost, where n is the number of training samples.

To deal with the resulting issue of scalability[1], various approximation methods have been developed. One approach that is used here is called Sparse Variational Gaussian Processes (Hensman et al., 2015). This classification approach effectively uses a set of *inducing points* optimized from the training data as an estimate of the full data set; these are then used to fit a GP.

Gaussian processes classification is described in Rasmussen and Williams (2006). This book also describes a variety of kernels, the choice of which strongly affects performance – some are explored in Section 5.

2.3. Adversarial Attacks

In this paper, an adversarially perturbed MNIST dataset is tested. The finite-gradient sign method (Goodfellow et al., 2014) (FGSM) is used to perform non-targeted attacks. This approach requires access to the trained model to maximize a loss function, performing gradient ascent away from correct classifications by adding small amounts of noise at each step. The amplitude of this noise is controlled by a parameter ϵ.

Other attacks, such as Jacobian-based saliency maps (Papernot et al., 2016) are not explored here, but the work could easily extended to include it.

3. Related Work

The key component of this work is using a fully Bayesian inference method to derive predictions and uncertainty. This is not a new idea, with proposals for predicting variance dating back to the 90s.

One basic idea is to train an ensemble of neural networks, using the set of predicted probabilities as a basis for calculating mean and variance of predictions (Paass, 1993). To obtain consistent results however, many networks need to be trained, thus leading to a tradeoff between prediction interval accuracy and computational intensity.

An alternative approach, Bayesian neural networks (MacKay David, 1992), place a distribution over each parameter in the network. While this makes calculating the full posterior distribution theoretically possible, it is practically intractable. Variational methods, and more recent work making inference backpropagation and GPU compatible (Blundell et al., 2015) have made Bayesian NNs more realistic, though they are still an expensive approach.

Alternative approaches exist that are more promising. One recent contribution is a dropout-based approach (Gal & Ghahramani, 2016), using dropout at *test* time to sample a variety of predictions and computing a mean and variance from this. This is method provides principled, but inherently sampling-based results. Thus the produced mean and variances are subject to fluctuations and highly dependent on the size of the network.

The most direct predecessor to this work is Bradshaw et al. (2017), which investigate GPDNNs (Gaussian Process Deep Neural Nets). However, the authors emphasize end-to-end trained models, comparing neural nets with softmax classification to ones with a GP replacing the softmax layer and retraining from scratch. In contrast, this work explores using pre-trained neural nets in combination with Gaussian processes which are trained post-hoc.

In terms of resisting adversarial attacks, this work will evaluate the normal CNN and the GP model on perturbed MNIST images, showing modest gains using the GP, though ultimately both are easily fooled with the same examples. Most existing methods for protecting machine learning models from such attacks are either based on data augmentation (which does not scale), or training multiple networks with high overhead. Recently alternative approaches have been proposed, such as using GANs (Samangouei et al., 2018)(Zantedeschi et al., 2017).

4. Implementation

The core libraries used were Keras (Chollet et al., 2015) with a TensorFlow[2] backend for the MNIST CNN. GPFlow (Matthews et al., 2017) was used to implement the Gaussian process model due to its support for large, high dimensional training data[3]. Scipy's machine learning toolkit was also tested but was found to be unsuitable.

[1] The training data here is a 60000x128 vector which is formed into a square matrix, far exceeding what could fit into memory

[2] https://www.tensorflow.org/

[3] GPFlow implements sparse variational GPs described in Section 2.2 as well as batched training

The models tested are

1. Keras MNIST CNN trained on 60,000 training images.
2. 3 different GPs trained on features extracted from the CNN.
3. A hybridized model merging CNN and GP predictions, motivated and developed in Section 5.6.

The CNN structure is as follows. A convolutional layer with 32 filters, using 3x3 pixel kernels is followed by a 64 channel convolutional layer with 3x3 pixel kernels. Max pooling with dropout is applied to avoid overfitting. The resulting outputs are flattened into a 1D array and connected to a 128-node layer with dropout. The final layer is a 10-node *softmax* activated fully connected layer. All other layers are *relu* activated.

5. MNIST

This section focuses on the grayscale MNIST test data set, consisting of 10,000 examples, each 28x28 pixels.

5.1. Accuracy and Model Motivation

Table 1 shows the test-set accuracy of the CNN, along with three different Gaussian processes: one using a *Polynomial* kernel, one using the *Matern12* kernel, and the last a *Matern32*Linear* kernel (referred to as *combined* kernel from now on). For explanation of the standalone kernels please refer to Rasmussen & Williams (2006).

Table 1. MNIST test set accuracy of of various models. The Hybrid model will be developed in Section 5.6.

MODEL	ACCURACY
CNN	99.11%
MATERN12 GP	99.03%
POLYNOMIAL GP	98.91%
LINEAR*MATERN32 GP	**99.14%**
HYBRID	**99.14%**

It will be shown later that the *Polynomial* kernel performs well in adversarial situations, as does *Matern12*, while the *combined* kernel appears to inherit properties from its two components: *Linear* kernels exhibit strong performance on non-adverserial datasets, while *Matern32* performs worse across the board but is most effective at merging with CNN predictions. This will exploited in Section 5.6. Of 17 different single and combined kernels, these had the most interesting properties.

While exploring the GP design space, many configurations were tested. The most influential by far was the number of inducing points used. All kernels in this paper were created using every 25th data point as a possible inducing point and allowed to be automatically optimized during training. Minibatch sizes were set at 8000. A Gaussian noise kernel was added to all kernels with constant variance 0.1, though its inclusion nor its variance have much impact on results. Allowing the GP to optimize a lengthscale parameter per input dimension did not improve performance but increased training time and was left out.

5.2. Distribution of Variance

Accuracy is a useful statistic to keep in mind. However, the goal of this paper is to examine the utility of prediction variance, which the CNN does not provide. Figure 2 examines the distribution of variance for correctly, and incorrectly classified examples using the *combined* kernel.

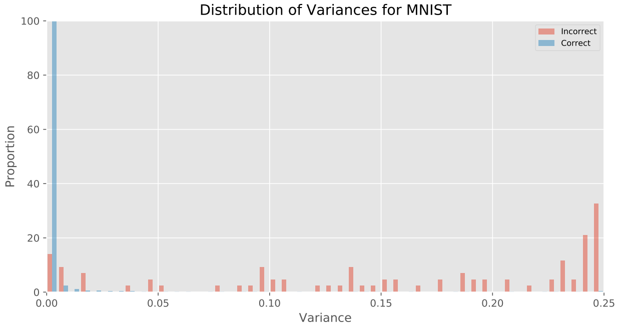


Figure 2. Distribution of Variances for both incorrect and correctly classified examples.

This plot clearly shows that the variance of incorrectly predicted classes is skewed higher than when correctly classified. One might expect correct classifications are on average confidently predicted. Indeed, this is supported here, showing inputs that are incorrectly classified have a lower confidence (higher variance).

5.3. Statistically Significant Predictions

We can combine both probability and variance to determine statistical significance using a two sample t-test. We look for 95% confidence ($t = 1.96$) that the top two predicted classes are different, i.e. the top class is the correct one. Using this we can obtain Table 2.

Table 2. Statistically Significant Predictions on MNIST using 95% confidence for the *combined* kernel.

	COMBINED GP
% SIGNIFICANT	97.55%
% CORRECT AND SIGNIFICANT	99.82%
% INCORRECT AND NOT SIGNIFICANT	28.16%

From this we can conclude that a user can be extremely sure of statistically significant responses being correct, and should be more weary of insignificant ones, as there is a 28% of it being incorrect on standard MNIST inputs.

5.4. SVGP Variance

The GP implementation used in this paper is the sparse variational gaussian process (SVGP) implemented in GPFlow. It is worth double checking that the variance being predicted makes sense and is providing additional information.

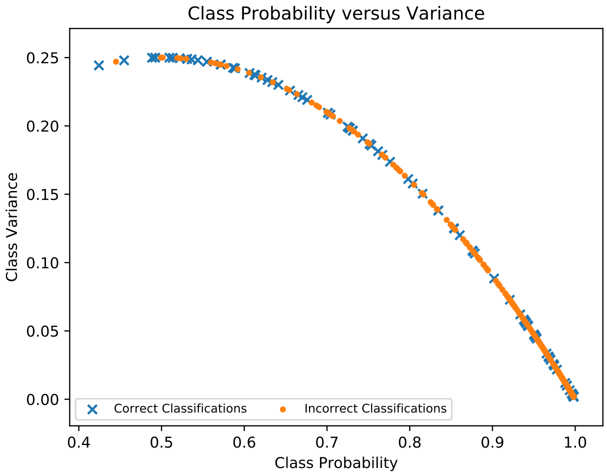


Figure 3. Plotting class probability versus variance on the standard MNIST dataset. They are totally correlated and thus one completely determines the other.

The above figure shows that, actually, with this technique and implementation, the class probability completely determines the predicted variance. In fact, a peek into the implementation reveals that variance σ^2 is calculated from the predicted mean p as $\sigma^2 = p - p^2$.

Note that this observation does not invalidate the usefulness of variance – it simply says that there is no point getting both mean and variance from the SVGP and analyzing them separately. Calculating the variance, in turn standard deviation, and performing statistical tests are still very useful results. Variance will also be used to merge *softmax* and GP predictions.

A human operator not realizing how these values are tied together may try to interpret the class probabilities and variances independently. While not harmful on its own, it could certainly lead to confusion and doubt about the system being presented to them.

It is tempting to calculate variance from CNN predictions in a similar way. However, there is no mathematical reasoning (that I am aware of) that suggests variances calculated this way have any meaning. On the other hand, GPs are accepted as being mathematically sound, so either using $\sigma^2 = p - p^2$ results in proper variance, or GPFlow contains a faulty implementation.

5.5. Examining Misclassifications

I further examine misclassifications on the MNIST dataset, for both CNNs and the *combined* kernel GP.

Table 3. Mean and Standard Deviation of probability of incorrect predicted classes for the *combined* kernel GP and the CNN.

MODEL	MEAN PROBABILITY	STANDARD DEVIATION
GP	74.5%	17.7%
CNN	75.3%	18.5%

From Table 3 we see that the CNN produces slightly higher probabilities when mis-classifying examples. From this we can loosely conclude that we should put more trust into the GP's prediction probabilities than the CNN's, a fact used later.

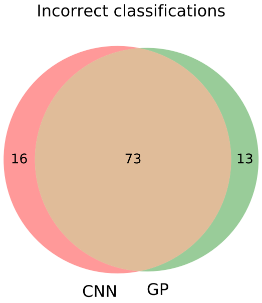


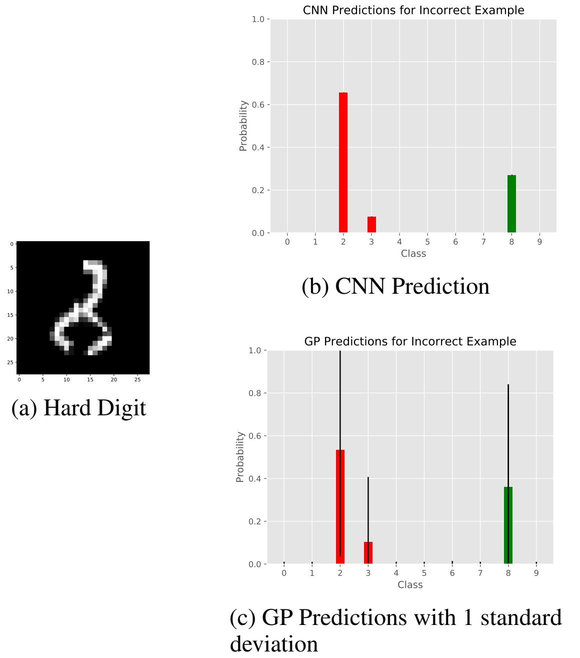
Figure 4. Overlap in misclassifications of the CNN and the *combined* GP model.

The Venn diagram in Figure 4 illustrates that both models largely fail on the same 73 test images. These are digits such as the one in Figure 5, and can be seen as the truly difficult or ambiguous images. Nothing much can be done about them.

However, more interesting is the non-overlapping region of the Venn diagram. From Figure 6 we see there is hope for boosting accuracy by combining the results from the CNN and the GP: the true class, in green, has a high probability but is not the highest. This inspires the hybridized model in the next section.

5.6. Hybridization

Because we have a (small) ensemble of classifiers when using both the CNN and a GP, a disagreement amounts to an extra bit of information. We can try to choose the best result, steered by the variance predicted by the GP.



(a) Hard Digit

(b) CNN Prediction

(c) GP Predictions with 1 standard deviation

Figure 5. An example of a truly difficult image that both models fail on. The green bar represents the true class, while the tallest red one is the predicted class.

The driving principle in this heuristic is that we trust the Gaussian process model more than the CNN.

5.6.1. Algorithm

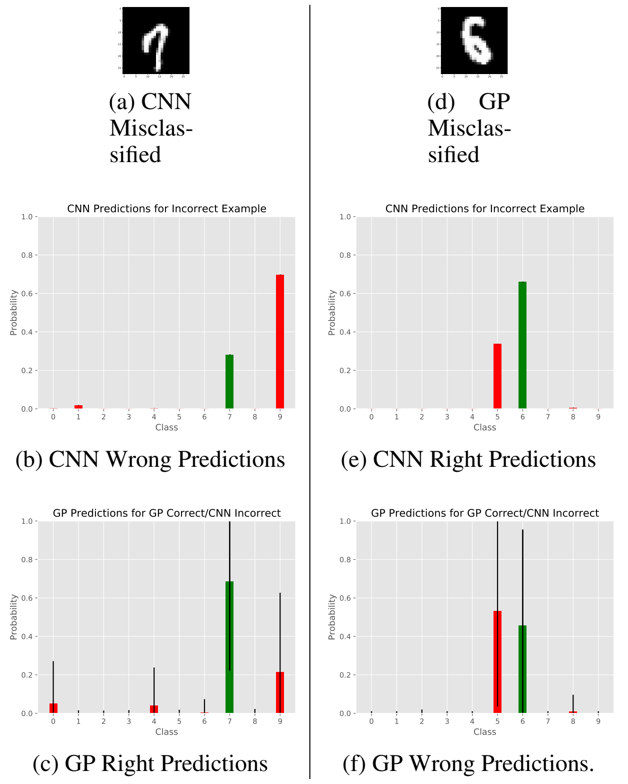
When merging CNN and GP predictions, we default to the GP prediction, only choosing the CNN's result if the following conditions hold:

1. The CNN's top prediction is within d standard deviations of the corresponding GP's probability (calculated using corresponding GP variance). This ensures that the GP agrees the CNNs prediction is plausible.

2. The CNN's top prediction is within d standard deviations of the top GP's prediction (calculated using top GP variance).

3. The GP's top prediction is not d or more standard deviations from the GP's corresponding prediction.

The first condition pulls the GP accuracy toward the CNN to either increase or decrease total accuracy. The latter two conditions mitigate worsened accuracy to an extent, ensuring that the GP's decision is accepted when it is very confident compared to the CNN and that the CNN's probability isn't too low, even if plausible according to the GP.



(a) CNN Misclassified

(d) GP Misclassified

(b) CNN Wrong Predictions

(e) CNN Right Predictions

(c) GP Right Predictions

(f) GP Wrong Predictions.

Figure 6. Examining cases of misclassifications. The left column corresponds to one of the 16 examples the CNN classified incorrectly, while the right column stems from the GP's incorrect classifications. With the right heuristic these case, and others, may be salvageable.

5.6.2. Discussion

When accepting a CNN prediction, we lose a meaningful variance and the resulting insight a human might utilize. However this could easily be fixed by flagging the result as uncertain, due to disagreement, as well as presenting the GP predictions.

Secondly, in this situation some kernels boost accuracy more than others. I believe this stems from larger non-overlapping regions as compared to the Venn diagram presented before. Through experimentation the *Matern32*, and in turn the combined *Linear*Matern32*, kernels were found to provide the best results.

Lastly, assuming the CNN and the GP outputs are available this cheap: linear in the number of classes.

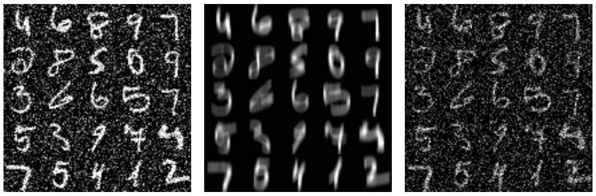
From here on I also present hybridized model results using the *combined* kernel with an acceptance criteria of $d = 0.2$ standard deviations.

Table 4. Accuracy across all models and n-MNIST data sets.

MODEL	AWGN	MOTION BLUR	LOW CONTRAST + AWGN
CNN	95.38%	94.58%	77.15%
MATERN12 GP	96.10%	87.46%	71.52%
POLYNOMIAL GP	95.68%	93.39%	**81.18%**
LINEAR*MATERN32 GP	96.13%	94.61%	78.95%
HYBRID	**96.19%**	**94.71%**	*79.27%*

6. n-MNIST

The ‘noisy’-MNIST dataset stems from Basu et al. (2017). The authors provide three transformations of the MNIST images: added white noise, motion blurred, and lowered contrast and white noise.



(a) (b) (c)

Figure 7. Added white noise, motion blurred, and low contrast + white noise samples. Images from http://csc.lsu.edu/~saikat/n-mnist/

6.1. Accuracy

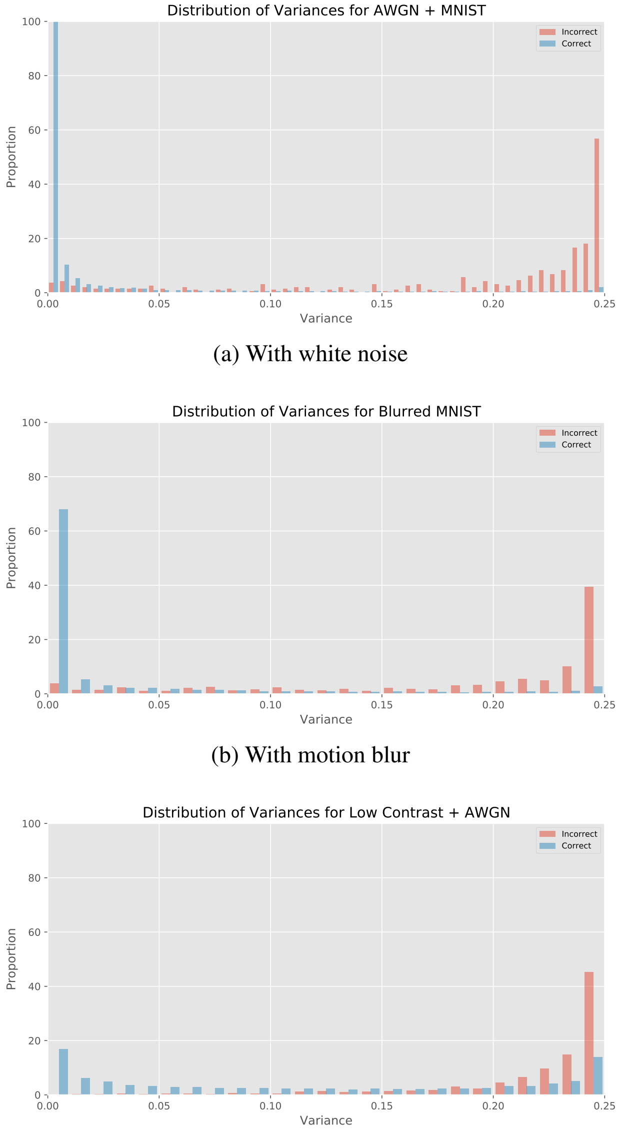
Table 4 shows how each model performs across the different n-MNIST datasets. The hybrid model consistently outdoes the standard *Linear * Matern32* kernel by a small amount. It is reassuring to note that the heuristic devised is not harming performance, and boosting it by 0.3% in the best case.

Most kernels outperform the CNN excepting *Matern12*. This kernel allows quite a lot of movement, i.e. the covariance between close data points is low, so it is possible it overfits the training data and suffers in very different situations.

From this experiment, it is quite evident that each kernel has some strengths, and none of them outperform the others across the board. This could be the basis of larger ensemble methods, and an idea discussed in Section 8.

6.2. Distribution of Standard Deviations

The distribution of variances on the various n-MNIST test sets are as expected, with Figure 8 showing the correct examples having much lower variance in general than the incorrect examples, though on more difficult datasets, such as the low contrast + white noise version, the distributions become more and more similar.



(a) With white noise

(b) With motion blur

(c) Low contrast and with white noise

Figure 8. Variances across the n-MNIST dataset

7. Adverserial Attacks

Adverserial attacks are described in Section 2.3. Cleverhans (Nicolas Papernot, 2017) generates perturbations using FGSM on the 10,000 MNIST images via the trained CNN.

Table 5. Mean and Standard Deviation of probability of incorrect predicted classes for the *combined* kernel GP and the CNN.

MODEL	ACCURACY
CNN	62.16%
MATERN12 GP	64.14%
POLYNOMIAL GP	**65.67%**
LINEAR*MATERN32 GP	62.58%
HYBRID	62.47%

Using an $\epsilon = 0.2$, all the models show large performance degradations from the 99% accuracy presented in Section 5.1. Returning to analyzing 95% confident predictions, we see in Table 6, we can see that adversarial attacks produce a large number of very confident incorrect predictions, successfully confusing the GP as well as the CNN.

Table 6. Statistically Significant Predictions on Perturbed MNIST using 95% confidence.

	COMBINED
% SIGNIFICANT	51.39%
% CORRECT AND SIGNIFICANT	72.43%
% INCORRECT AND NOT SIGNIFICANT	47.82%

7.1. Varying ϵ

Previous work by Bradshaw et al. (2017) showed that for GPDNNs, which have GP's trained end to end with CNNs from scratch, targeting the plain CNN architecture does not harm the GPDNN much. This conclusion is rather obvious since they learn different structure. Vice versa, their CNN is resistant to attacks on the GPDNN.

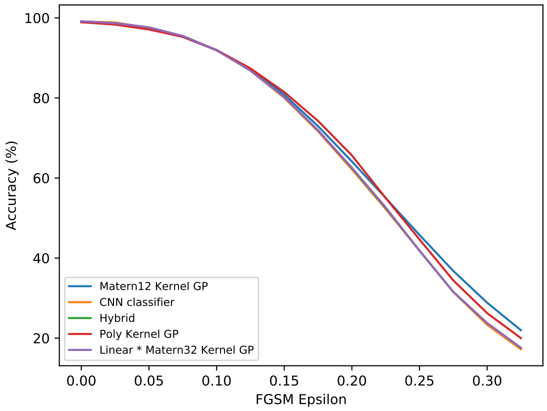


Figure 9. Classification Accuracies on FGSM attacked MNIST images across varying ϵ

Figure 9 shows, in the best case with *Polynomial* or *Matern12* kernels, at most 5-10% boosts can be expected as ϵ becomes large, but generally GPs degrade just as much as the CNN.

From these results we conclude GPs do not provide the hoped resistance to adversarial attacks. This is rather expected, since most of the targeted CNN structure is shared between models and produces features for the GP that appear like an incorrect class. One idea to mitigate this could be to incorporate features from lower levels in the neural network, at the cost of dimensionality.

8. Discussion

The best presented use of variance is to identify statistically significant predictions, which users can trust to an extremely high degree. For instance, a system could a statistical test to flag uncertain results for further inspection. This is especially useful in situations where false negatives are costly, such as in the medical field.

As noted in Section 5.4, in this implementation variance is actually just as informative as class probability. This does not mean variance is useless information since we can use it for inter-class statistical tests, as well as presenting it to human users who may find it useful when reasoning about predictions.

An alternative use of variance, the hybridizing approach, was found to provide incremental gains over the *Linear * Mater32* kernel. Unfortunately, it also slightly worsened performance in adversarial situations. It is possible that more consistent accuracy gains are within reach with more sophisticated methods. One obvious approach is to further refine the acceptance criteria described in Section 5.6.1, or experiment with different acceptance parameter values, or even using 3 different parameters, one for each part of the criterion. However, none of these are likely to lead to particularly novel algorithms or dramatically improved results. Even worse, all examples that were rescued with the hybridized approach turned out to be statistically insignificant predictions, so this approach is not useful at all if using statistical tests to eliminate bad predictions in the first place.

On the other hand, if only trying to marginally increase accuracy, creating the hybridization is almost free: most of the CNN needs to be evaluated to extract features to feed into the GP. For instance, in TensorFlow, executing the entire CNN and saving an intermediate value incurs almost no cost. If the Gaussian process is going to be executed in $O(n^2)$ anyway, the hybridization is only $O(m)$ (for m classes) more expensive.

Whether training the GP at all is worth it is another question: even with sparse variational implementations, GPs still do not scale as well as CNNs, and this investigation has revealed that classification accuracy is generally on par or only slightly better than CNNs. Where interpretation and confidence measures are important, it is certainly worth re-evaluating the entire training set to train the GP. However, if this is not a concern, I argue using a GP is not useful.

GPs are also detrimental when dealing with large scale datasets and time-bound execution. A small CNN can be made to execute quickly and at a constant speed regardless of training data size – GPs predict quadratically in the number of training samples, and even with variational methods are slow to train.

9. Future Work

9.1. Ensembles

Hybridization between a single GP and a CNN ultimately provides unimpressive results. A more interesting idea could be take the entire process a step further: hybridize based on a multitude of kernels, effectively forming an ensemble.

As mentioned before, each kernel appears to have different strengths, and it may be possible to come up with a heuristic for merging many predictions. Part of such an investigation would be to explore the role of variance. More heuristics could be developed, or reinforcement learning could be used to decide on an optimal decision strategy.

Finally, this entire investigation depends critically on a closed set of examples to perform well (i.e. we expect that all examples actually belong to one of the 10 classes). The introduction hinted that since softmax strongly accentuates small differences between prediction probabilities, and does not take into account absolute activation of the final layer, CNNs can return high probabilities for examples that are not any of the expected classes. I expect GPs can also be made to exhibit such behavior. This is related strongly to the research area of open set classification (Scheirer et al., 2013). Using an ensemble including classifiers specifically trained on irrelevant data could lead to an effective rejection mechanism.

9.2. Automatic Kernel Building

It was shown in Section 7 that kernels have varying adversarial resistance. Another investigative avenue could thus be automatically creating the best kernel for both a data set and adversarial resistance. Automatic kernel choice is already an active research area (for example (Abdessalem et al., 2017), (Duvenaud, 2014)), but without the adversarial effects taking into account. Further, it might be interesting to consider creating kernels that can effectively be ensembled together, i.e. err on maximally different parts of the same dataset.

10. Conclusion

This paper examined the utility of variance in classification. To do this, a CNN was trained on the MNIST dataset, and compared to various Gaussian processes on a variety of datasets. The main benefit of variance is interpretability: having access to an uncertainty in a prediction means consumers of classifications can make further judgements about results and don't need to accept predictions at face value; statistical tests can be used to make principled decisions about significance of data.

This paper also presented the idea of merging predictions from the base CNN and the GP, using variance as a guide for choosing one decision of the other. A heuristic was presented which achieved slightly increased test set accuracy on non-adversarial examples. Overall however, a simple two-way merge does not yield impressive results, and further research could be conducted into ensembling multiple different gaussian processes.

10.1. Software and Data

Code, results, and this report can be found at:

https://github.com/flyingsilverfin/CNN_GP_MNIST

Please note that intermediate results are not saved but can be recomputed, and that some of the configurations are specific to the development machine.

References

Abdessalem, Anis Ben, Dervilis, Nikolaos, Wagg, David J, and Worden, Keith. Automatic kernel selection for gaussian processes regression with approximate bayesian computation and sequential monte carlo. *Frontiers in Built Environment*, 3:52, 2017.

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.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Bradshaw, John, Matthews, Alexander G. de G., and Ghahramani, Zoubin. Adversarial Examples, Uncertainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep Networks. pp. 1–33, 2017. URL http://arxiv.org/abs/1707.02476.

Bridle, John S. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pp. 227–236. Springer, 1990.

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

Duvenaud, David. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

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

Goodfellow, Ian J, Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

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

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

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.

MacKay David, JC. A practical bayesian framework for backprop networks. *Neural computation*, 1992.

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.

Nicolas Papernot, Nicholas Carlini, Ian Goodfellow Reuben Feinman Fartash Faghri Alexander Matyasko Karen Hambardzumyan Yi-Lin Juang Alexey Kurakin Ryan Sheatsley Abhibhav Garg Yen-Chen Lin. cleverhans v2.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2017.

Paass, Gerhard. Assessing and improving neural network predictions by the bootstrap algorithm. In *Advances in Neural Information Processing Systems*, pp. 196–203, 1993.

Papernot, Nicolas, McDaniel, Patrick, Jha, Somesh, Fredrikson, Matt, Celik, Z Berkay, and Swami, Ananthram. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387. IEEE, 2016.

Rasmussen, Carl Edward and Williams, Christopher KI. *Gaussian process for machine learning*. MIT press, 2006.

Samangouei, Pouya, Kabkab, Maya, and Chellappa, Rama. Defense-gan: Protecting classifiers against adversarial attacks using generative models. 2018.

Scheirer, Walter J, de Rezende Rocha, Anderson, Sapkota, Archana, and Boult, Terrance E. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Zantedeschi, Valentina, Nicolae, Maria-Irina, and Rawat, Ambrish. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 39–49. ACM, 2017.