Reproducibility of the Gaussian Process Equivalence to Neural Networks

Niklas Brake[1], Nicholas Dudek[2], and Samuel Whaite[3]

Abstract— **Ensuring reproducibility in the rapidly expanding field of machine learning is a great scientific challenge. In this report, we review the reproducibility of a paper recently submitted to the 2018 ICLR Conference, *Deep Neural Networks as Gaussian Processes*. We find that although this paper provides much of the information necessary for reproducing the results, certain important information is lacking. Furthermore, the computational requirements of reproducing all the paper's results were too high to verify without access to more powerful machines. The results that we could faithfully reproduce were similar, but not exact to those presented in the original paper. In conclusion, the paper's results were fairly reproducible, but the paper's conclusions could not be completely verified.**

I. INTRODUCTION

The paper *Deep Neural Networks as Gaussian Processes* [1] concerns the equivalence between a Gaussian Process and a deep fully-connected neural network with i.i.d. priors over all its weights and biases in the limit of infinite network width.

The paper proposes a computationally efficient way of computing the kernel matrices used to define these Gaussian process models. This method employs the use of a pre-computed lookup table for a given activation function. This lookup table contains numerically estimates values for the covariance of post-activation variables, given that the pre-activations can be modelled as a Gaussian process.

The success of this technique, as well as the similarities between Gaussian processes and neural networks, is then demonstrated in the paper by testing both approaches on the Modified National Institute of Standards and Technology [2] (MNIST) handwriting database, and the Canadian Institute for Advanced Research [3] (CIFAR-10) object identification database. The neural networks were benchmarked using both Rectified Linear Units (ReLU) and tanh activation functions, in addition to varying other aspects, such as the number of training samples, the width, and the depth of the network.

The paper's conclusion is that neural network accuracy approaches that of the corresponding Gaussian process as the width of these networks gets larger. Furthermore, the paper claims this new Gaussian process prediction method is competitive with multi-layer neural networks trained with stochastic gradient descent.

The purpose of this report is to investigate and discuss the findings of the paper mentioned above, determine how reproducible the results are, and, if possible reproduce the results presented. In particular, there are two main goals: Duplicate the simplified process for calculating the Gaussian process convolution matrix, and test the resulting Gaussian process models performance against a variety of neural network tests that are similar to those in the paper. We find that the accuracies of Gaussian process predictions were comparable to those from neural networks, but we could not reproduce the high accuracy of predictions made in the original paper. Furthermore, we conclude that although the paper does a good job providing the details required to reproduce their results, it falls short of a complete description of the kernel matrix computation algorithm. For this reason, we had difficulties in reproducing the prediction accuracy of Gaussian processes using a kernel matrix computed in this manner.

[1] Email: niklas.brake@mail.mcgill.ca, ID: 260602499
[2] Email: nicholas.dudek@mail.mcgill.ca, ID: 260480902
[3] Email: samuel.whaite@mail.mcgill.ca, ID: 260656949

II. METHODOLOGY

A. Neural Networks

The neural network benchmarking used in the paper consisted of both ReLU and tanh models. For each of these activation functions, model hyperparameters were chosen using Googles Vizier service. Vizier is a hyperparameter optimizer which runs multiple trials of a certain model with various values for a selection of hyperparameters. In the paper, depth, width, initial weights, initial bias, and learning rate were all optimized in this way. Depths were tested between 1 and 5, with widths ranging from 100 to 5000 nodes per layer, with all layers using the same width. The learning rates tested ranged from 10^{-4} to 0.2. This hyperparameter optimization was performed multiple times for different sizes of training data. On average, the researchers ran 250 Vizier trials to find the optimal hyperparameters for each training data size. In terms of reproducibility, only the depth, weights, and weight averages were included in their report.

To generate results for benchmarking, neural networks using both ReLu and Tanh activation functions were created in Python by using the Keras and Tensorflow libraries. In order to use identical hyperparameter optimization and duplicate the methodology exactly, access to Vizier would be required, which was not possible. Instead, the models were used in conjunction with scikit-learns GridSearchCV, which trains multiple versions of a model with all possible permutations of a set of hyperparameters. The goal of this hyperparameter optimization was to get relatively similar values to those obtained in the paper itself, and then to compare them to results from the Gaussian process. Due to limitations of computing resources, some models were omitted, in particular those involving the full CIFAR data set. The results shown in Table I use the best-case model depth

TABLE I

Reproduction of Table 2 from the original paper [1]. Certain accuracies display NA. Due to limited computation capacity, these configurations were not replicated.

Num training	Model (ReLU)	Test accuracy	Model (tanh)	Test accuracy
MNIST:100	NN-2-5000-0.10-0.11	0.6857	NN-1-500-1.48-0.61	0.7625
	GP-100-1.79-0.83	0.6897	GP-100-3.14-0.97	0.1135
MNIST:200	NN-2-2000-0.52-0.00	0.7785	NN-2-1000-1.80-1.99	NA
	GP-100-1.79-0.83	0.7309	GP-100-3.99-2.00	0.1135
MNIST:500	NN-2-5000-1.82-0.77	0.8626	NN-1-5000-3.74-2.18	NA
	GP-100-1.79-0.83	0.8187	GP-50-3.48-1.86	0.1135
MNIST:1k	NN-2-5000-3.19-0.00	0.8692	NN-2-1000-0.60-0.00	0.8613
	GP-20-1.45-0.28	0.8544	GP-20-1.96-0.62	0.1028
MNIST:2k	NN-2-5000-2.88-0.01	0.9285	NN-1-2000-0.98-1.30	NA
	GP-10-1.11-0.55	0.8431	GP-10-1.79-1.45	0.1028
MNIST:5k	NN-3-500-2.92-0.22	NA	NN-2-1000-4.12-2.18	0.9339
	GP-7-0.61-0.07	0.3804	GP-3-1.11-0.00	0.8609
MNIST:10k	NN-2-2000-0.42-0.16	0.9671	NN-2-2000-2.41-1.84	0.9583
	GP-7-0.61-0.07	0.7533	GP-2-1.62-0.28	0.9333
MNIST:20k	NN-3-1000-2.45-0.98	NA	NN-2-2000-0.21-0.10	NA
	GP-5-1.62-0.83	NA	GP-1-2.63-0.00	NA
MNIST:50k	NN-2-2000-0.60-0.44	0.9832	NN-2-5000-0.28-0.34	0.9666
	GP-1-0.10-0.48	NA	GP-1-1.28-0.00	NA
CIFAR:100	NN-5-500-1.88-1.00	0.2064	NN-2-200-3.22-2.09	0.2012
	GP-3-4.49-0.97	0.1637	GP-10-3.65-1.17	0.1
CIFAR:200	NN-3-200-0.17-0.00	NA	NN-3-200-1.41-0.21	NA
	GP-3-3.99-1.72	0.1719	GP-7-3.65-0.55	0.1
CIFAR:500	NN-1-100-1.26-0.63	NA	NN-1-2000-0.11-0.90	NA
	GP-20-1.79-0.21	0.2655	GP-7-3.65-0.62	0.1
CIFAR:1k	NN-5-500-1.29-0.28	0.3289	NN-1-200-1.45-0.12	0.3211
	GP-7-1.28-0.00	0.2091	GP-50-2.97-0.97	0.1
CIFAR:2k	NN-3-5000-5.59-0.57	NA	NN-5-1000-0.86-1.28	NA
	GP-3-4.16-1.17	0.1883	GP-5-4.66-1.03	0.1
CIFAR:5k	NN-5-2000-5.26-1.74	NA	NN-1-5000-0.07-0.22	NA
	GP-3-4.66-1.03	0.1207	GP-10-3.65-1.38	0.1
CIFAR:10k	NN-5-2000-1.60-1.07	0.4103	NN-1-500-1.48-1.59	0.4197
	GP-5-2.97-0.28	0.1246	GP-7-3.48-2.00	NA
CIFAR:20k	NN-3-5000-4.18-0.18	NA	NN-2-5000-0.02-1.12	NA
	GP-3-5.00-0.83	NA	GP-7-3.14-1.93	NA
CIFAR:45k	NN-3-5000-0.53-0.01	NA	NN-2-2000-1.05-2.08	NA
	GP-3-3.31-1.86	NA	GP-3-3.48-1.52	NA

and widths provided by the report, and optimized values for the remaining parameters which were not specified. The models were trained on the same datasets as the paper: The MNIST and CIFAR-10 sets.

B. *Gaussian Process*

The modelling of a neural network as a Gaussian process relies critically on the activation function used at each hidden node. In general, the kernel of the Gaussian process is deterministic and depends only on the activation function. In the paper, an algorithm for numerically evaluating this kernel via bilinear interpolation into a pre-computed lookup table is given for the general case. An analytic solution for the special case of the ReLU activation function is also provided. In reproducing the results from the paper, we used this analytic solution for reproducing the Gaussian processes modelling networks with ReLU activation functions, and the lookup table for reproducing those modelling networks with the tanh activation function. These algorithms were implemented in MATLAB and loops were either converted into tensor multiplication or implemented as a parallel loop to reduce computation time. LUT interpolation was done with MATLAB's *griddedInterpolant* function.

The given method for prediction presents the formalism for predicting a single test point. This method was implemented such that all test points were evaluated at once with one-hot label vectors. Furthermore, we computed the test kernel $K_{x*,\mathcal{D}}$ using the same recursive algorithm used to calculate the kernel K^L, with the same depth as that used to calculate the kernel matrix on the training data.

III. Results

A. *Availability of Requisite information*

One of the main challenges in reproducing the results of the paper came from information that was omitted regarding the construction of the Gaussian process. Ideally, it would be possible to replicate their results without needing to understand the theoretical basis which lead to their algorithm. This is not the case, however, as the lookup table used in the algorithm is not well-defined. In computing the lookup table, is it necessary to loop over two vectors $s = [0, ..., s_{max}]$ and $c = [-1, ..., 1]$. The lookup table at each value is calculated

TABLE II
BASELINE RESULTS FOR NEURAL NETWORKS.

Num training	Model	Reported Accuracy	Reproduced Accuracy	Difference
MNIST:100	NN-2-5000-ReLU	0.7786	0.6857	0.0929
MNIST:200	NN-2-2000-ReLU	0.8298	0.7785	0.0513
MNIST:500	NN-2-5000-ReLU	0.9028	0.8626	0.0402
MNIST:1000	NN-2-5000-ReLU	0.9252	0.8692	0.0560
MNIST:2000	NN-2-5000-ReLU	0.9485	0.9285	0.02
MNIST:10000	NN-2-2000-ReLU	0.9771	0.9671	0.01
MNIST:45000	NN-2-2000-ReLU	0.9864	0.9832	0.0032
MNIST:100	NN-1-500-TanH	0.7766	0.7625	0.0141
MNIST:1000	NN-2-1000-TanH	0.9254	0.8613	0.0641
MNIST:5000	NN-2-1000-TanH	0.9693	0.9339	0.0354
MNIST:10000	NN-2-2000-TanH	0.9745	0.9583	0.0162
MNIST:45000	NN-2-5000-TanH	0.9857	0.9666	0.0191
CIFAR:100	NN-5-5000-ReLU	0.2586	0.2064	0.0522
CIFAR:1000	NN-5-500-ReLU	0.3235	0.3289	-0.0054
CIFAR:10000	NN-5-2000-ReLU	0.4545	0.4103	0.0442
CIFAR:100	NN-2-200-TanH	0.2470	0.2012	0.0458
CIFAR:1000	NN-1-100-TanH	0.3378	0.3211	0.0167
CIFAR:10000	NN-1-500-TanH	0.4429	0.4197	0.0232

as:

$$F_{i,j} = \frac{\sum_{a,b} \phi(u_a)\phi(u_b) \exp\left(\begin{bmatrix} u_a \\ u_b \end{bmatrix}^\top \begin{bmatrix} s_i & c_j s_i \\ c_j s_i & s_i \end{bmatrix}^{-1} \begin{bmatrix} u_a \\ u_b \end{bmatrix}\right)}{\sum_{a,b} \exp\left(\begin{bmatrix} u_a \\ u_b \end{bmatrix}^\top \begin{bmatrix} s_i & c_j s_i \\ c_j s_i & s_i \end{bmatrix}^{-1} \begin{bmatrix} u_a \\ u_b \end{bmatrix}\right)}$$

Note this definition requires the inversion of a matrix that is singular for the values $s_i = 0$, $c_j = -1$, and $c_j = 1$. While the paper does have a footnote which explains that a second lookup vector should be made for $c_j = 1$, the paper does not contain a method for computing this second vector, nor do the address the other values that cannot satisfy their definition. Furthermore, to generate the lookup table, one sums over the values $[-u_{max}, ..., u_{max}]$ where this range represents the pre-activations. However the paper does not explain what u_{max} value was used in the experiments performed, nor how the value might have been determines. Since the pre-activations are distributed as Gaussians, there is no theoretical maximum value they may achieve. In the paper, it is specified that $s_{max} < u_{max}^2$, so for our calculation of the LUT, we chose an arbitrary value that satisfied this constraint, namely $u_{max} = 18$

Finally, for prediction, the following matrix appears:

$$\left(K_{\mathcal{D}\mathcal{D}} + \sigma_\epsilon^2 \mathbb{1}\right)^{-1}$$

However, the value of σ_ϵ^2 used by the authors is neither stated nor discussed. For our implementation, we ignored the noise model mentioned in the paper and let $\sigma_\epsilon^2 = 0$.

B. Computational Resources Required

An important factor in the reproducibility of machine learning results is the computational resources required to run the presented algorithms. For this paper we found that, in the case of the MNIST data set, the kernel matrix for the full training set requires over 20GB of RAM. Additionally, this matrix has to be inverted when actually being used for predictions. The Gaussian process algorithms were run on a laptop with 8GB of RAM, and a dual core 2.9GHz processor. Due to the memory constraints, the necessary kernels for the 20,000 and 50,000 sample training sets could not be computed. Due to the processing constraints, hyperparameter optimization for the variances σ_w and σ_b could not be reproduced. Using parallelized matrix multiplications, generating the lookup table with values of $n_g = 500$, $n_v = 501$, and $n_c = 500$ took slightly under an hour to calculate. In theory, this operation is $O(n_g^2 n_v n_c)$. The kernel matrices were calculated for each Gaussian process configuration stated in Table 2 of the original paper [1]. The test kernel vectors were then computed and the accuracy of the predictions evaluated. In this manner, all the Gaussian process accuracies in Table I were computed overnight.

The deep signal propagation heat maps were more computationally intensive. To fully reproduce Figure 7 from the paper [1], which shows heatmaps of the 5,000 training sample MNIST accuracies with $d = 50$ and $d = 100$, we estimate it would require several hundred hours using the resources available. As a result, no attempt was made to verify this aspect of the paper.

The algorithms presented in this paper were applied to a wide range of training set sizes. As such, simpler computational tasks could still be verified, even when reproducing all results was too computationally expensive.

C. Data

The main focus of our report is the reproduction of Table 2 presented in the appendix of the original paper [1]. We found that overall our baseline results were very similar to those presented in the paper (see Table II). Our results, while comparable, are mostly lower than the presented results, with the average difference being 4.1% lower. The worst result obtained is the MNIST ReLU with 100 training samples, which scored 9.29% lower, with the best being the CIFAR 1,000 samples using ReLU, which scored 0.64% higher than the paper's result. In general, these discrepancies are likely due to the hyperparameter optimization the the authors were

TABLE III

RELU ACCURACIES WITH 100 DEPTH TEST-KERNEL.

Num training	Model	Test Accuracy
MNIST:100	GP-100-1.79-0.83	0.6897
MNIST:200	GP-100-1.79-0.83	0.7309
MNIST:500	GP-100-1.79-0.83	0.8187
MNIST:1000	GP-20-1.45-0.28	0.8844
MNIST:2000	GP-10-1.11-0.55	0.9449
MNIST:5000	GP-7-0.61-0.07	0.9600
MNIST:10000	GP-7-0.61-0.07	0.9608

able to perform. As is mentioned above, Vizier was used to train hundreds of models for each data set size, and return the best performing one. Without access to similar optimization methods or computing power, we were unable fine-tune the models to same degree, resulting in worse performance.

Our implementation of the Gaussian process did not yield the same accuracies as those presented in the original paper. Our best accuracies were achieved on the MNIST dataset using the ReLU activation function. When originally varying the Gaussian Process depth, the $K_{x*,\mathcal{D}}$ kernel was computed to a depth of 100. These results are seen in Table III. Surprisingly, these results were more accurate than those seen when the $K_{x*,\mathcal{D}}$ kernel depth was the same as the $K_{\mathcal{D},\mathcal{D}}$ depth (see Table I).

For the tanh activation function, our predication accuracy on the CIFAR data set was only as good as random. As mentioned above, the kernels for the Gaussian processes modelling the ReLU activation function were computed using the analytic function

$$K^l(x,x') = \sigma_b^2 + \frac{\sigma_w^2 \xi_{x,x'}}{2\pi}\sqrt{K^{l-1}(x,x)K^{l-1}(x',x')}$$

$$\xi_{x,x'} = \sin\theta_{x,x'}^{l-1} + (\pi - \theta_{x,x'}^{l-1})\cos\theta_{x,x'}^{l-1}$$

$$\theta_{x,x'}^{l} \cos^{-1}\left(\frac{K^l(x,x')}{\sqrt{K^l(x,x)K^l(x',x')}}\right)$$

whereas the tanh kernel matrices were computed by interpolating into the pre-computed LUT. The significant difference in the accuracies between tanh and ReLU may be because the LUT algorithm did not work as expected, or that the implementation of the algorithm was not the same as in the original paper. However, using the algorithm, we achieved 93% accuracy on the MNIST 10K data set (see Table I). This is surprising because the Gaussian processes using the LUT-derived kernel only had good prediction accuracy on one other dataset, namely MNIST 5K. It is possible that the algorithm only worked with a large enough dataset and the reason it did not work on CIFAR is because the input vector size was larger and thus would require even more training data.

The accuracies of the Gaussian process predications were compared against those made by our implementation of the MNIST neural network (see Table IV). The Gaussian process obtained a maximum of 7.6% higher than the neural network, and a minimum of 4.8% lower (see Table IV). Overall, the Gaussian processes and neural networks achieved similar accuracies. Notably, for 10,000 training examples on the MNIST data set, the different between the two methods was only 0.63%. However, in our reproduction, the Gaussian process did not consistently out-perform the neural networks.

TABLE IV

COMPARISON OF REPRODUCED GP AND NN RESULTS.

Num training	GP Accuracy	NN Accuracy	Difference
MNIST:ReLU 100	0.6897	0.6857	0.004
MNIST:ReLU 200	0.7309	0.7785	-0.0476
MNIST:ReLU 500	0.8187	0.8626	-0.0439
MNIST:ReLU 1000	0.9449	0.8692	0.0757
MNIST:ReLU 2000	0.9600	0.9285	0.0315
MNIST:ReLU 10000	0.9608	0.9671	-0.0063

IV. DISCUSSION

In summary, the overall process and results detailed in the paper are reproducible, with some caveats. In particular, the paper lacks information regarding the generation of the lookup table. It does not provide sufficient information to obtain u_{max} nor does it adequately explain how to deal with cases where $c = \pm 1$ or $s = 0$. Furthermore, it does not specify what value of σ_ϵ^2 was used for the final prediction of the Gaussian processes. Aside from this, the process for generating the Gaussian process is explained well, as is the procedure for testing. The baseline results with a neural network for both datasets were reproducible within an average of 4.1%, with the deficiency likely explained by differences in hyperparameter optimization.

Apart from the problems implementing the kernel matrix lookup table, a main difficulty in reproducing the entire paper stems from a lack of computational resources. The lookup tables for the full MNIST data set, for example, require over 20GB of memory. Calculating these tables was not possible with the resources available. Furthermore, inverting such a big matrix each time a hyper-parameter is to be tested was simply not feasible. As such, the hyper-parameter optimization could not be reproduced, and the results for the larger sample sizes, as well as the deep signal propagation heatmaps could not replicated. There is no implications, other than those previously stated, that would imply these results could not theoretically be replicate given additional resources.

With respect to the conclusions of the original paper, we find that although Gaussian processes do outperform neural networks after a certain point, the neural network techniques were not state-of-the-art, nor were they necessarily competitive with the best approach for each data set. A neural network that makes use of advanced techniques such as dropout, or choke layers could potentially see an increase in performance that puts them over the Gaussian processes. Furthermore, a core proposition of the paper is that ”the trained neural network accuracy approaches that of the corresponding GP-based computation.” It is indicated in the paper that the equivalent Gaussian process to an L-layer deep neural network has itself a composition kernel of depth L. However all comparisons in the paper between Gaussian process accuracies and neural network accuracies are done between the best performing models of each class. The paper does not discuss the accuracies of Gaussian processes and neural networks of the same depth.

We conclude that the results in this paper were fairly reproducible, however specifying u_{max} and numerically stable solutions to lookup table algorithm would have aided in the papers reproducibility. Furthermore, although the Gaussian process method for predication is effective, we did not find they performed significantly better than neural networks, nor were they easier to work with, especially in the context of memory requirements. Moving forward, the implications of these results on other data sets is a question worth considering. The polynomial growth of the kernel matrix size relative to the size of the data set suggests that this approach may not be feasible depending on the task.

ACKNOWLEDGMENT

Niklas Brake implemented the Gaussian process in MATLAB. Nicholas Dudek and Sam Whaite implemented the baseline neural networks. All three authors participated in running the algorithms to obtain test accuracies, and all three contributed to the writing of this report. We hereby state that all the work presented in this report is that of the authors.

REFERENCES

[1] Lee, J., Bahri, Y., Novak, R., Schoenholz S., Pennington, J., Sohl-Dickstein J. (2017): Deep Neural Networks as Gaussian Processes. (https://arxiv.org/abs/1711.00165)

[2] LeCun, Y., Cortes, C., & Burges C. (1999): The MNIST Dataset of Handwritten Digits.

[3] Krizhevsky, A. (2009): Learning Multiple Layers of Features from Tiny Images.

APPENDIX

A. Public Review

Available at: https://openreview.net/forum?id=B1EA-M-0Z

1) Availability of Requisite Information: The appendix of the paper does a good job outlining most of the necessary information for reproducing the results. The computation of the Gaussian process is thorough and well described, with the exception of some information regarding the lookup table. The values n_g,n_v, and n_c were all provided, however a value for u_{max} was not provided, nor was it explained how it was determined. With regards to testing, the authors provide the neural network and Gaussian process configurations used to obtain their best results, as well as their random search ranges for various hyper-parameters. Furthermore, the data sets on which they trained and tested their algorithms are publicly available and readily accessible.

2) Computational Resources Required: The Gaussian process algorithms were run on a laptop with 8 GB of RAM and a dual core 2.9GHz processor. The kernel matrix for the full MNIST training set is over 20 GB. Furthermore, this matrix needs to be inverted for prediction. Due to memory requirements, the results for 20,000 and 50,000 training points could not be computed. With matrix multiplication and parallel computing, the lookup table, which theoretically is $O(n_g^2 n_v n_c)$, took just under an hour to calculate for the values $n_g = 500$,$n_v = 501$, and $n_c = 500$. The accuracies for the Gaussian processes in Table 2 were all computed (with the exception of 20,000, and 50,000 sizes) overnight. Finally we estimated that reproducing Figure 7, MNIST=5,000, d=50, activation=tanh would have taken us approximately 60 hours.

3) Prior Knowledge Required: The paper does a good job giving an overview of the theoretical results at the beginning of the paper. They outline how to construct the kernel, as well as how to use the kernel to do prediction. However, the value for u_{max} perhaps required a deeper theoretical understanding to determine. Further, the lookup table algorithm is not well-defined. For values of $c = \pm 1$ and $s = 0$, the lookup table is defined with the inverse of a singular matrix. The case of $c = 1$ is addressed, however the solution to these singularities is not explicitly given.

4) Results: The Gaussian process calculations were reproducible, although the accuracies we obtained did not perfectly match those provided in the paper. Notably, the LUT only gave good accuracy on MNIST with training data size larger than 2000.

The replicated baseline accuracies overall were quite close to those stated in the paper. They typically were slightly lower, likely due to the differences in hyperparameter optimization. The authors of the paper had access to Google's Vizier technology, which is not publicly available. Despite this, using the same model depths and widths yielded values within 5% .

5) Practical Applications: As the paper described, the Gaussian process seems to form something like an upper bound on the performance of fixed-width fully connected neural networks. However, the gap between the neural network baselines and the gaussian process were more often than not very minimal (See Table IV). This brings in to doubt the practical applications of this gaussian process: It performed approximately equivalently to the neural networks but required a polynomial space lookup table as well as being more complex to implement. As it stands, this new method brings only disadvantages and a possible a minor boost to accuracy. Unless some special machine learning application can be found where the gaussian process performs significantly better compared to fully connected neural networks it is hard to justify every using this approach. The implications of the paper at the moment seem to be purely theoretical and will hopefully yield insight into relations between gaussian processes and other current machine learning approaches in the future.

6) Conclusion: The time for reproducing Table 2 was reasonable, though we would not have been able to reproduce the optimization of the GP hyper-parameters given our computational resources. We were also unable to reproduce the deep-signal propagation heatmaps given our resources and time constraints. There is nothing to suggest these results would not have been reproducible, however, given additional resources. The numerical kernel implementation is arguably the central contribution of this paper. Given the information provided in the paper, we could not reproduce the results from this numerical algorithm. It would have helped to have the original source code to guide us, especially with respect to handling the singular cases of the lookup table. The source code does not seem to be available online. We have contacted the authors to ask whether it was or could be made available, but we had not received an answer by the time of writing In conclusion, this paper was fairly reproducible but requires a high level of computational power and theoretical knowledge. Specifying u_{max} and numerically stable solutions to their lookup table would have aided in the papers reproducibility.