methodologies employed in computational experiments, notably in climate and weather prediction [30], [31].

III. Results and Discussion

A. Selection of hyperparameters

The optimization of the NN-SVGP-based radiative physics emulator required meticulous analysis of error outcomes across various hyperparameter settings, a critical process for enhancing prediction accuracy while maintaining computational efficiency. The primary hyperparameters under scrutiny were the size of the induction points (Ips), the number of neurons in the hidden layer (Hidn), and the number of neurons in the output layer (Fout) of the neural network.

We computed the Root Mean Squared Error (RMSE) for an array of hyperparameter configurations to assess the model's performance. RMSE, the square root of the average squared differences between predicted and actual values, is a widely recognized measure for evaluating predictive accuracy. Beyond assessing the overall RMSE, this study delved into the errors associated with Long Wave (LW) Heating Rate, Short Wave (SW) Heating Rate, LW Flux, and SW Flux.

Table 2 presents the RMSE values across various hyperparameter settings, facilitating an in-depth comparison of their effects on model accuracy. Notably, the configuration with Ips=128, Hidn=256, and Fout=128 emerged as the most effective, demonstrating the lowest overall error. Such insights were instrumental in refining our emulator's design, guiding us toward configurations that promise precision and efficiency. This table enumerates the RMSE across various configurations, highlighting the nuanced impact of each hyperparameter on model accuracy. Notably, the configuration with Ips=128, Hidn=256, and Fout=128 emerged as the most effective, demonstrating the lowest overall error. Such insights were instrumental in refining our emulator's design, guiding us toward configurations that promise precision and efficiency.

TABLE II
DIFFERENT RMSE VALUES FOR HYPER-PARAMETER SETTINGS

Hyper-Parameters	**Root Mean Squared Error**				
Ips-Hidn-Fout	**Total**	**LW Heating Rate**	**SW Heating Rate**	**LW Flux**	**SW Flux**
64-64-32	2.839	1.399	0.958	1.687	16.795
64-64-64	2.851	1.374	0.972	1.709	16.898
64-64-128	2.551	1.287	0.894	1.455	15.020
64-64-256	2.696	1.315	0.907	1.557	15.983
64-128-32	2.412	1.291	0.872	1.466	14.017
64-128-64	2.709	1.305	0.901	1.565	16.100
64-128-128	2.576	1.245	0.893	1.441	15.278
64-128-256	2.400	1.225	0.844	1.465	14.072
64-256-32	2.498	1.226	0.873	1.457	14.760
64-256-64	2.359	1.174	0.824	1.477	13.882
64-256-128	2.530	1.252	0.894	1.475	14.922
64-256-256	2.347	1.229	0.882	1.438	13.636
128-64-32	2.460	1.259	0.877	1.438	14.430
128-64-64	2.420	1.213	0.835	1.395	14.275
128-64-128	2.383	1.232	0.849	1.436	13.941
128-64-256	2.429	1.217	0.872	1.443	14.271
128-128-32	2.364	1.200	0.832	1.381	13.890
128-128-64	2.246	1.179	0.821	1.356	13.083
128-128-128	2.768	1.322	0.958	1.614	16.417
128-128-256	2.456	1.211	0.878	1.485	14.464
128-256-32	2.374	1.258	0.838	1.437	13.841
128-256-64	3.720	1.692	1.198	3.273	21.891
128-256-128	**2.162**	1.141	0.792	1.354	**12.567**
128-256-256	2.318	1.166	0.837	1.449	13.588
256-64-32	2.346	1.215	0.840	1.376	13.725
256-64-64	2.206	1.174	0.821	**1.337**	12.799
256-64-128	2.499	1.196	0.863	1.425	14.835
256-64-256	2.560	1.203	0.856	1.436	15.267
256-128-32	2.351	1.184	0.814	1.422	13.830
256-128-64	2.169	1.149	0.801	1.344	12.593
256-128-128	2.956	1.343	0.992	1.805	17.649
256-128-256	2.185	**1.132**	**0.784**	1.341	12.766
256-256-32	2.518	1.257	0.853	1.526	14.855
256-256-64	2.259	1.171	0.799	1.388	13.211
256-256-128	2.283	1.133	0.807	1.358	13.449
256-256-256	2.253	1.135	0.820	1.397	13.196

B. Evaluation of NN-SVGP Emulator Performance

The emulator was trained using the Gpytorch Framework within Python, utilizing the Adam optimization algorithm with an initial learning rate of 0.01. We designated 10% of our roughly 270,000 training samples for validation, reducing the learning rate by half whenever the validation data's MSE did not improve over five epochs. Training occurred on an NVIDIA A100-PCI-40GB system with 256 CPUs and 1TB of memory.

To compare the performance of our NN-SVGP emulator with that of Roh and Song's NN emulator [9], we examined the accuracy and the computational efficiency. Figure 2 shows our NN-SVGP emulator demonstrates an RMSE of 2.162, approximately 28% better than the NN emulator's RMSE of 3.003. This improvement in accuracy is quantified by the Brier Skill Score (BSS).

$$BSS = \left(1 - \frac{NN\text{-}SVGP_{rmse}}{NN_{rmse}}\right) \times 100\,\% \tag{9}$$

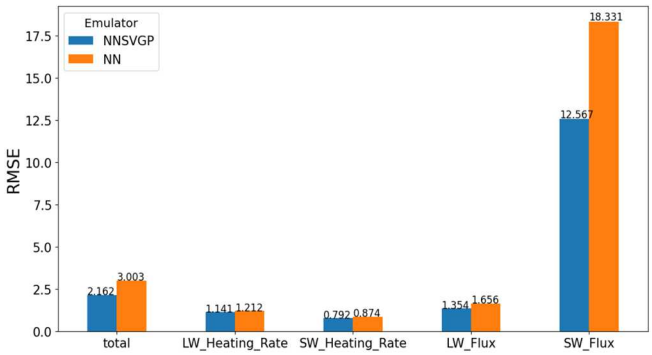

Fig. 2 Comparison of RMSE between NN emulator and NN-SVGP emulator

Additionally, to assess the computational demand of each emulator, we measured the average calculation speed, leveraging a GPU for enhanced performance analysis. As depicted in Figure 3, we found the average computation speed for the NN to be 0.7 ms, while the NN-SVGP averaged 3.2 ms. This increase in computation time for the NN-SVGP emulator is attributed to its greater computational complexity. However, considering the higher accuracy and the ability of the NN-SVGP to quantify uncertainty, the trade-off for increased computation time is justified in applications where predictive precision is crucial.

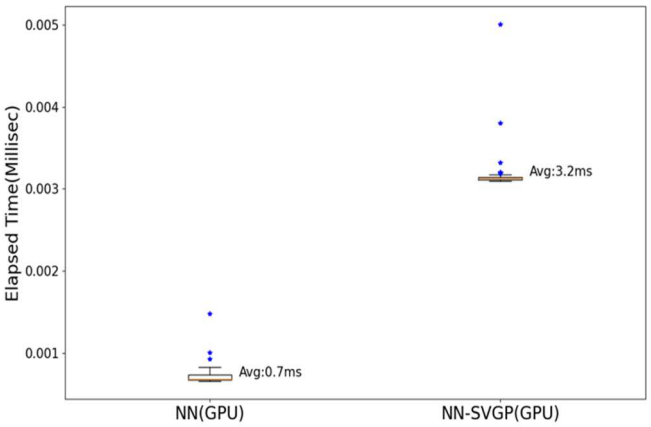

Fig. 3 Comparative Boxplot of Computation Speeds for Neural Network (NN) and Neural Network with Stochastic Variational Gaussian Process (NN-SVGP) Emulators

IV. Conclusion

This study developed an NN-SVGP emulator based on the Gaussian Process Regression (GPR) model using simulation data of radiative physics processes in a numerical forecast model. The model significantly improves prediction accuracy and uncertainty estimation compared to conventional NN emulators. In particular, NN-SVGP can provide confidence intervals for prediction results, an important tool for managing and understanding uncertainty in complex physical processes.

However, regarding computational speed, NN-SVGP is much slower than the NN model. This means that NN-SVGP may be limited in environments requiring high-speed computations. Nevertheless, it is noteworthy that applying the Gaussian process regression model and its potential use in limited environments confirms the possibility of developing an emulator capable of fast computation. These results suggest that NN-SVGP emulators have the potential to contribute to improving the performance of numerical forecasting models. In addition, the ability to estimate uncertainty opens the possibility of application in various fields, such as climate change research and disaster prediction systems.

In assessing the computational efficiency of our models, we noted a significant difference in calculation speeds between NN and NN-SVGP, with NN-SVGP averaging slower computation times. Despite this, the precision, robustness, and ability of NN-SVGP to provide valuable uncertainty estimates far outweigh its computational demands. The enhanced predictive performance and confidence supplied by NN-SVGP affirm its utility, especially in applications where accuracy is critical. Therefore, while the NN model's faster computation may be advantageous in time-sensitive scenarios, the accuracy and reliability benefits of NN-SVGP support its use even with its slower computational speed.

In future research, we will explore ways to improve the NN-SVGP emulator's computational efficiency and increase the uncertainty estimation accuracy. We aim to develop an emulator that can deliver faster and more accurate predictions, thereby contributing to the advancement of numerical forecasting models and their applications in diverse fields. This suggests that integrating sophisticated modeling techniques, such as those employed in NN-SVGP, can lead to more stable and reliable forecasting tools, notwithstanding the comparative slowdown in computation times.

Computationally, it took about three times longer than the NN-SVGP emulator but produced results faster than the radiative physics simulator (RRTMG-K). Despite losing computational time compared to the NN emulator, the NN-SVGP emulator proposed in this study delivered more accurate results. This underlines the potential benefits of adopting advanced emulators over traditional simulators, ensuring more stable operations even with increased computational loads.

Acknowledgment

This work was supported by a research grant from Jeju National University in 2022.

References

[1] C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker. “A Bayesian approach to the design and analysis of computer experiments,” Oak Ridge National Lab., TN, USA, Tech. Rep. ORNL-6498, Sep. 1988.

[2] T. J. Santner, B. J. Williams, W. I. Notz, and B. J. Williams. “The design and analysis of computer experiments,” New York: Springer, 2003.

[3] A. O'Hagan. “Curve fitting and optimal design for prediction,” *Journal of the Royal Statistical Society: Series B (Methodological)*., vol. 40, no. 1, pp. 1-24, 1978, 10.1111/j.2517-6161.1978.tb01643.x.

[4] T. Beckers. “An introduction to Gaussian process models,” arXiv preprint, arXiv:2102.05497, 2021.

[5] J. Hensman, N. Fusi, and N. D. Lawrence. “Gaussian processes for big data,” arXiv preprint, arXiv:1309.6835, 2013.

[6] P. Ghasemi, M. Karbasi, A. Z. Nouri, M. S. Tabrizi, and H. M. Azamathulla. “Application of Gaussian process regression to forecast multi-step SPEI drought index,” Alexandria Engineering Journal., vol. 60, no. 6, pp. 5375-5392, 2021.

[7] S. Rasp, P. D. Dueben, S. Scher, J. A. Weyn, S. Mouatadid, and N. Thuerey. “WeatherBench: a benchmark data set for data-driven weather forecasting,” Journal of Advances in Modeling Earth Systems, vol. 12, no. 11, 2020, doi: 10.1029/2020MS002203.

[8] Y. Lee and J. S. Park. “Generalized Nonlinear Least Squares Method for the Calibration of Complex Computer Code Using a Gaussian Process Surrogate,” Entropy., vol. 22, no. 9, pp. 985, 2020.

[9] Y. A. Seo and J. S. Park. “Expectation-Maximization Algorithm for the Calibration of Complex Simulator Using a Gaussian Process Emulator,” Entropy., vol. 23, no. 1, pp. 53, 2020.

[10] S. Roh and H. J. Song. “Evaluation of neural network emulations for radiation parameterization in cloud-resolving model,” Geophysical Research Letters., vol. 47, no. 21, 2020.

[11] J. Luo, X. Ma, Y. Ji, X. Li, Z. Song, and W. Lu. “Review of machine learning-based surrogate models of groundwater contaminant modeling,” Environmental Research., vol. 238, part. 2, 2023, doi:10.1016/j.envres.2023.117268.

[12] M. S. Go, J. H. Lim, and S. Lee. “Physics-informed neural network-based surrogate model for a virtual thermal sensor with real-time simulation,” International Journal of Heat and Mass Transfer., vol. 214, 2023, doi: 10.1016/j.ijheatmasstransfer.2023.124392.

[13] R. M. Slot, J. D. Sørensen, B. Sudret, L. Svenningsen, and M. L. Thøgersen. “Surrogate model uncertainty in wind turbine reliability assessment,” Renewable Energy., vol. 151, pp. 1150-1162, 2020, doi:10.1016/j.renene.2019.11.101.

[14] M. Tang, Y. Liu, and L. J. Durlofsky. “A deep-learning based surrogate model for data assimilation in dynamic subsurface flow problems,” Journal of Computational Physics., vol. 413, 2020, doi:10.1016/j.jcp.2020.109456.

[15] P. Jiang, Q. Zhou, X. Shao, P. Jiang, Q. Zhou, and X. Shao. “Surrogate-model-based design and optimization,” Springer Singapore., pp. 135-236, 2020.

[16] J. S. Park. “Tuning complex computer codes to data and optimal designs,” University of Illinois at Urbana-Champaign, 1991.

[17] D. D. Cox, J. S. Park, and C. E. Singer. “A statistical method for tuning a computer code to a database,” Computational statistics & data analysis., vol. 37, no. 1, pp. 77-92, 2001, doi:10.1016/S0167-9473(00)00057-8.

[18] M. D. Hoffman, D. M. Blei, and J. Paisley. “Stochastic variational inference,” Journal of Machine Learning Research., 2013.

[19] J. Quinonero-Candela and C. E. Rasmussen. “A unifying view of sparse approximate Gaussian process regression,” The Journal of Machine Learning Research., vol. 6, pp. 1939-1959, 2005.

[20] C. Ding, H. Rappel, T. Dodwell. “Full-field order-reduced Gaussian Process emulators for nonlinear probabilistic mechanics,” Computer Methods in Applied Mechanics and Engineering., vol. 405, 2023, doi:10.1016/j.cma.2022.115855.

[21] Y. A. Seo, Y. Lee, and J. S. Park. “Iterative method for tuning complex simulation code,” Communications in Statistics-Simulation and Computation., vol. 51, no. 7, pp. 3975-3992, 2022, doi:10.1080/03610918.2020.1728317.

[22] H. Liu, Y. S. Ong, X. Shen, and J. Cai. “When Gaussian process meets big data: A review of scalable GPs,” IEEE transactions on neural networks and learning systems., vol. 30, no. 11, pp. 4405-4423, 2020, doi: 10.1109/TNNLS.2019.2957109.

[23] M. M. Noack, H. Krishnan, M. D. Risser, and K. G. Reyes. “Exact Gaussian processes for massive datasets via non-stationary sparsity-discovering kernels," Scientific reports., vol. 13, no. 1, pp. 3155, 2023.

[24] S. Damm, D. Forster, D. Velychko, Z. Dai, A. Fischer, and J. Lücke. “The ELBO of Variational Autoencoders converges to a Sum of Three Entropies,” arXiv preprint, arXiv:2010.14860, 2020.

[25] I. Torroba, C. I. Sprague, and J. Folkesson. “Fully-probabilistic Terrain Modelling with Stochastic Variational Gaussian Process Maps,” arXiv preprint, arXiv:2203.10893, 2022.

[26] M. Ketenci, A. Perotte, N. Elhadad, and I. Urteaga. “A Coreset-based, Tempered Variational Posterior for Accurate and Scalabel Stochastic Gaussian Process Inference,” arXiv preprint, arXiv:2311.01409, 2023.

[27] I. Torroba, M. Cella, A. Teran, N. Rolleberg, and J. Folkesson. “Online stochastic variational gaussian process mapping for large-scale bathymetric slam in real time,” IEEE Robotics and Automation Letters., vol. 8, no. 6, 2023, doi:10.1109/LRA.2023.3264750.

[28] H. Yu and Y. Chen. “Stochastic Motion Planning as Gaussian Variational Inference: Theory and Algorithms,” arXiv preprint, arXiv:2308.14985, 2023.

[29] R. Meng, H. K. Lee, and K. Bouchard. “Stochastic Collapsed Variational Inference for Structured Gaussian Process Regression Networks,” In Conference of the International Federation of Classification Societies, Cham: Springer International Publishing, pp. 253-261, 2022.

[30] C. Hua, X. Cao, B. Liao, and S. Li. “Advances on intelligent algorithms for scientific computing: an overview,” Frontiers in Neurorobotics, vol. 17, 2023, doi:10.3389/fnbot.2023.1190977.

[31] J. Chen, Y. Saad, and Z. Zhang. “Graph coarsening: from scientific computing to machine learning,” SeMA Journal., vol. 79, pp. 187-223, 2022, doi:10.1007/s40324-021-00282-x.