2023 International Conference on Innovations in Intelligent Systems and Applications (INISTA) | 979-8-3503-3890-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/INISTA59065.2023.10310471

A fully automatic Bayesian model for adaptive activation functions in artificial neural networks

Mohamed Fakhfakh[1] and Lotfi Chaari[2]

[1] Toulouse INP, IRIT, University of Toulouse, France
mohamed.fakhfakh@toulouse-inp.fr

[2] Toulouse INP, IRIT, University of Toulouse, France
lotfi.chaari@toulouse-inp.fr

***Abstract*—There is a significant focus in deep neural networks on finding activation functions that can enhance network performance. Scientists are also interested in developing flexible activation functions that can adapt during training to avoid overfitting and provide reliable parameter learning.**
This paper presents a Bayesian algorithm that employs Markov Chain Monte Carlo (MCMC) to estimate activation function parameters and model weights. This algorithm accelerates convergence through effective sampling and evaluates its effectiveness on two datasets, achieving high accuracy with low complexity.

***Index Terms*—** M CMC, Deep Learning, Trainable activation functions, neural networks, Optimization

I. Introduction

Over the past two decades, Convolutional Neural Networks (CNNs) [1, 2] have gained popularity due to their ability to reduce complex and high-dimensional input data into easily understandable low-dimensional concepts. CNNs consist of hierarchical layers, where each layer is built on the features of the layer below, resulting in increasing data abstraction with each layer. The activation function plays a vital role in discovering meaningful features and allowing for nonlinearity, which ultimately improves precision by providing more accurate outcomes. The activation function operates by comparing the input value to a threshold value. If the input value is greater than the threshold value, the neuron is activated, and otherwise, it is disabled.

Despite years of research, the effectiveness of neural networks remains an ongoing area of study. One of the most significant challenges is identifying the optimal activation function, with various functions having been suggested. These functions can be classified into three categories: fixed-shape, trainable, and trainable non-standard neuron activation functions, according to [3].

Bayesian methods have seen increasing use in various fields due to their ability to integrate prior knowledge into models and parameter probabilities. Recent developments in Markov Chain Monte Carlo (MCMC) methods [4, 5] have made it easier to apply Bayesian analyses to complex and multidimensional data, as mentioned in [6]. Our previous work showed that utilizing a Bayesian formulation resulted in more efficient resolution of the optimization problem related to the use of CNNs than traditional gradient-based methods, according to [7].

In this study, we introduce a novel approach based on Markov Chain Monte Carlo (MCMC) to estimate both the parameters of a trainable activation function and the weights of a deep model simultaneously. This approach builds upon our previous work presented in [8], where we utilized non-smooth Hamiltonian methods to fit sparse artificial neural networks.

The proposed optimization procedure in [8] offers several key advantages. Firstly, it achieve efficient and rapid sampling procedures, even when dealing with non-differentiable energy functions that often arise from the utilization of sparse regularization functions. Additionally, unlike gradient-based techniques, the method guarantees convergence towards the global minimum of the defined cost function.

The rest of this paper is organized as follows. In Section II, we present the problem statement. The adopted hierarchical Bayesian model is detailed in Section III. The proposed Bayesian inference scheme is developed in Section IV and validated in Section V. Finally, the conclusion and future work are drawn in Section VI.

II. Problem formulation

The activation function plays a crucial role in neural networks and can take various forms, including commonly used functions such as sigmoid [9] and ReLU [10], as well as trainable functions like FReLU [11] and MeLU [12], where the parameters are optimized using gradient descent. Alternative methods like the Maxout network [13] have also been explored.

However, these approaches have their limitations. They can be computationally expensive and suffer from issues like vanishing gradients, which can lead to getting stuck in local minima and lower performance [14]. Furthermore, there are unresolved concerns regarding the flexibility of these functions and parameter estimation.

In this paper, we propose a modification to the MeLU activation function [12] and incorporate parameter estimation into a global Bayesian optimization framework. In this paper, we propose a modification to the MeLU activation function [12] and incorporate parameter estimation into a global Bayesian optimization framework. The MeLU activation function, with its Mexican hat shape resembling a bell but with a peak in the center, offers an advantage for smaller input values. Compared to other activation functions, MeLU provides a stronger response for these small values, effectively avoiding the problem of gradient vanishing. Our objective is to address

the limitations of standard activation functions and enhance the flexibility and performance of our model. By incorporating trainable activation functions, we enable dynamic learning and optimization of activation function parameters alongside the network weights, thereby improving representation and modeling capabilities.

Moreover, The combination of multiple trainable activation functions provides an opportunity to leverage the strengths of different functions and exploit their complementary properties. Each activation function may better poerform in modeling specific types of data or capturing certain types of non-linearities. By combining them, our aim is to enhance the overall modeling capacity of the network and achieve better performance across various tasks. Furthermore, this combination helps address the issue of vanishing gradients, which is a common challenge in deep neural networks.

The motivation behind the development of the Modified Mexican ReLU (MMeLU) activation function is rooted in its ability to address the limitations of the MeLU method while providing enhanced performance and reduced computational demands. One drawback of the MeLU method is its speed and memory requirements, primarily due to the utilization of numerous parameters for estimation. In contrast, MMeLU requires fewer parameters to estimate and employs a Bayesian framework, distinguishing it from other activation functions.

Let us define the function :

$$f_{\gamma,b}(x) = max(b - |x - \gamma|, 0) \tag{1}$$

where x is the input of a neural network layer and γ, b are real numbers. This function is null when $|x - \gamma| > b$. In addition, it increases with a derivative of 1 between $\gamma - b$ and γ and then decreases utilizing a derivative of minus 1 between γ and $\gamma + b$. When plotted, this function has the shape of a Mexican hat.

Using $f_{\gamma,b}$, the proposed activation function MMeLU can be defined as

$$MMeLU(x) = ReLU(x) + c\, f_{\gamma,b}(x) \tag{2}$$

where c is a real number. c, b, and γ are the parameters to be estimated.

As indicated above, this contribution is an extension of our recent work [8] which introduced a new method for sparse optimization of deep neural network weights based on the non-smooth Hamiltonian Monte Carlo algorithm [15]. Compared to the MeLU function, the novelty behind the definition of our MMeLU function resides in a combination of the popular activation function ReLU with the Mexican hat function, which is estimated only once. The novelty of this study is that it suggests a new trainable activation function and adjusts all the parameters of the artificial neural network in a Bayesian framework. Furthermore, our method can be applied to both regression and classification problems. In what follows, a classification problem will be adopted to formulate the proposed model.

Let us denote the ground truth by y, and the estimated label by the proposed activation function MMeLU(x, W) as a non-linear function of the input x and weights vector $W \in \mathbb{R}^N$. The weight vector can be determined during the training phase using a generic distance D (euclidean, Minkowski,...) based on the error applied to the M input data.

$$\begin{aligned}\widehat{W} &= \arg\min_{W} \mathcal{L}(W) \\ &= \arg\min_{W} \sum_{m=1}^{M} D\left(MMeLU(x^m; W) - y^m\right) + \sum_{l=1}^{L} \lambda \|W^l\|_1,\end{aligned} \tag{3}$$

where λ is a regularization parameter to be estimated that balances the solution between the data fidelity and ℓ_1 regularization terms.

III. Hierarchical Bayesian Model

As mentioned earlier, the estimation of trainable activation function parameters is formulated in a Bayesian framework. In this approach, all parameters and hyperparameters are considered as realisations of random variables following probability distributions. Specifically, a likelihood distribution is defined to model the relationship between the target weights, activation function parameters, and the data, while a prior distribution is used to represent prior knowledge about the target weights and activation function parameters.

A. Likelihood

According to the principle of minimizing the error between the reference vector y (label or continuous values) and its estimate $\widehat{y}$, we define the likelihood law as

$$\begin{aligned}&f(y, x; W, c, \gamma, b) \propto \times \\ &\prod_{m=1}^{M} \exp\left[-D(MMeLU(x^m; W, c, \gamma, b) - y^m)\right].\end{aligned} \tag{4}$$

where $x = \{x^1, \ldots, x^M\}$, and $y = \{y^1, \ldots, y^M\}$. It is important to mention that when the Euclidean distance is employed as a measure for D (distance), the corresponding likelihood function is essentially a Gaussian distribution.

B. Priors

In our model, the unknown parameters are gathered in the unknown vector $\theta = \{W, c, \gamma, b, \lambda\}$.

Prior for W :

As regards the prior knowledge of the weights vector W, we propose to use a Laplace distribution in order to promote the sparsity of the neural network :

$$f(W; \lambda) \propto \prod_{l=1}^{L} \prod_{k=1}^{K_l} \left[\frac{1}{\lambda} \exp\left(-\frac{|W_k^l|}{\lambda}\right)\right], \tag{5}$$

where K_l is the number of weights of the l layer of the network, λ is a parameter to be estimated.

Prior for λ :
Since λ is a positive value real positives, an inverse-gamma (IG) distribution has been used :

$$f(\lambda;\delta,\mu) = IG(\lambda;\delta,\mu) \propto (\lambda)^{-1-\delta} \exp\left(-\frac{\mu}{\lambda}\right) \tag{6}$$

where $\delta = \mu = 10^{-3}$.

Prior for c :
Regarding the parameter c, we consider a uniform distribution over the interval $[0,1]$, denoted as

$$c \sim U_{[0,1]}(c). \tag{7}$$

Prior for γ :
Since γ is a real value, a Gaussian distribution is used as follows

$$f(\gamma;\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\gamma^2}{2\sigma^2}\right), \tag{8}$$

where σ^2 is a hyperparameter to be estimated.

Priori for b:
Since b is a positive real number, an exponential distribution is used as follows:

$$f(b;\lambda_b) \propto \begin{cases} \frac{1}{\lambda_b} \exp\left(-\frac{b}{\lambda_b}\right); \text{if}; b \geq 0 \\ 0; \text{otherwise.} \end{cases} \tag{9}$$

where λ_b is a hyperparameter to be estimated. This prior penalizes large values of b. It is worth noting that this prior helps promote low values for b, thereby encouraging sparsity in the neural network by increasing the number of deactivated neurons.

C. Hyperpriors

Since λ_b and σ^2 are positive real numbers, an inverse gamma (IG) distribution was used as a hyper-*a priori*:

$$f(\lambda_b;\delta,\mu) = IG(\lambda_b;\delta,\mu) \propto (\lambda_b)^{-1-\delta} \exp\left(-\frac{\mu}{\lambda_b}\right), \tag{10}$$

and

$$f(\sigma^2;\delta,\mu) = IG(\sigma^2;\delta,\mu) \propto (\sigma^2)^{-1-\delta} \exp\left(-\frac{\mu}{\sigma^2}\right), \tag{11}$$

where δ and μ are positive parameters that were fixed at 10^{-3}.

IV. Bayesian inference scheme

By adopting a Maximum *a Posteriori* (MAP) approach, we first need to express the posterior distribution. Let $\Phi_e = \{\sigma^2, \lambda_b\}$ be the hyperparameters to be estimated, and $\Phi_m = \{\delta, \mu\}$ be the hyperparameters. Using the likelihood, the prior distributions, and the defined hyperpriors, we can write the posterior distribution as:

$$f(\theta,\Phi_e;y,\Phi_m) \propto f(y;\theta) f(\theta;\Phi_e) f(\Phi_e;\Phi_m) \tag{12}$$

which can be reformulated in a detailed version as

$$\begin{aligned} &f(\theta,\Phi_e;\mathbf{y},\mathbf{x},\Phi_m) \propto \\ &\prod_{m=1}^{M} \exp\left[-D(MMeLU(x^m;W,c,\gamma,b) - y^m)\right] \times \\ &\prod_{l=1}^{L}\prod_{k=1}^{K_l} \left[\frac{1}{\lambda}\exp\left(-\frac{|W_k^l|}{\lambda}\right)\right] \times (\lambda)^{-1-\delta} \exp\left(-\frac{\mu}{\lambda}\right) \times \\ &\exp\left(-\frac{\gamma^2}{2\sigma^2}\right) \times \frac{1}{\lambda_b}\exp\left(-\frac{b}{\lambda_b}\right) 1_{\mathbb{R}_+}(b) \times 1_{[0,1]}(c) \times \\ &(\lambda_b)^{-1-\delta} \exp\left(-\frac{\mu}{\lambda_b}\right) \times (\sigma^2)^{-1-\delta} \exp\left(-\frac{\mu}{\sigma^2}\right). \end{aligned} \tag{13}$$

The conditional posterior related to W writes

$$\begin{aligned} &f(W;c,\gamma,b,\lambda) \propto \exp\left[-\sum_{l=1}^{L}\sum_{k=1}^{K_l}\frac{|W_k^l|}{\lambda}\right] \times \\ &\exp\left[-\sum_{m=1}^{M}\left(D(MMeLU(x^m;W,c,\gamma,b) - y^m)\right)\right]. \end{aligned} \tag{14}$$

The conditional distribution for the parameter c is given by:

$$\begin{aligned} &f(c;W,b,\gamma) \propto 1_{[0,1]}(c) \times \\ &\exp\left[-\sum_{m=1}^{M}\left(D(MMeLU(x^m;W,c,\gamma,b) - y^m)\right)\right]. \end{aligned} \tag{15}$$

For the parameter b, the conditional distribution is given by:

$$\begin{aligned} &f(b;W,c,\gamma,\lambda_b) \propto \exp\left(-\frac{b}{\lambda_b}\right) \times \\ &\exp\left[-\sum_{m=1}^{M}\left(D(MMeLU(x^m;W,c,\gamma,b) - y^m)\right)\right]. \end{aligned} \tag{16}$$

As regards γ, the conditional distribution writes:

$$\begin{aligned} &f(\gamma;W,b,c,\sigma^2) \propto \exp\left(-\frac{\gamma^2}{2\sigma^2}\right) \times \\ &\exp\left[-\sum_{m=1}^{M}\left(D(MMeLU(x^m;W,c,\gamma,b) - y^m)\right)\right]. \end{aligned} \tag{17}$$

The conditional distribution for the parameter λ is given by:

$$f(\lambda;\delta,\mu) \propto \lambda^{-1-(\delta+K)} \exp\left(-\frac{\mu}{\lambda}\right) \propto IG(\delta+K,\mu). \quad (18)$$

For the hyperparameter vector Φ_e, it is necessary to calculate the conditional distributions from which it is possible to sample based on the likelihood and adopted priors.

The conditional distribution for the hyperparameter λ_b is given by:

$$f(\lambda_b;b,\mu,\delta) \propto \lambda_b^{-2-\delta} \exp\left(-\frac{b+\mu}{\lambda_b}\right) \propto IG(\delta+1,b+\mu). \quad (19)$$

The conditional distribution for the hyperparameter σ^2 is given by:

$$f(\sigma^2;\mu,\gamma,\delta) \propto (\sigma^2)^{-1-\delta} \exp\left(-\frac{\gamma^2+2\mu}{2\sigma^2}\right) \propto IG(\delta,\gamma+2\mu). \quad (20)$$

The sampling scheme is summarized in Algorithm 1, where the model weights W and the parameters of the proposed MMeLU function are sampled, in addition to all hyperparameters which need to be estimated.

Algorithm 1: Gibbs sampler for the proposed method.

Fix the hyperparameters Φ_m ;
for $r = 1, \ldots, S$ **do**
  * Sample c according to $f(c; W, b, \gamma)$;
  * Sample γ according to $f(\gamma; W, b, c, \sigma^2)$;
  * Sample b according to $f(b; W, c, \gamma, \lambda_b)$;
  * Sample σ^2 according to $f(\sigma^2; \mu, \gamma, \delta)$;
  * Sample λ_b according to $f(\lambda_b; b, \mu, \delta)$;
  * Sample λ according to $f(\lambda; \delta, \mu)$;
  * Sample W as described in [8] ;
end

In Algorithm 1, S denotes the number of MCMC sampling iterations. After the burn-in period, the sampled coefficients are used to calculate the estimators MMSE (Minimum...) $\widehat{W}$, $\widehat{c}$, $\widehat{b}$, $\widehat{\gamma}$, in addition to $\widehat{\sigma^2}$, $\widehat{\lambda}$ and $\widehat{\lambda_b}$.

V. Experimental validation

To validate our proposed approach, we conducted experiments on three different datasets: one regression dataset and two image classification datasets. The first dataset involved using the California Housing dataset [16], which is a commonly used dataset for regression tasks. It provides information about housing in California with the aim of predicting the median value of houses in different neighborhoods based on various features. The second dataset consisted of CT (Computed Tomography) images for COVID-19 classification, aimed at detecting whether a patient is positive for COVID-19 based on a CT scan. The third dataset utilized the CIFAR-10 dataset, which is a widely used benchmark dataset for image classification tasks. It contains images belonging to 10 different classes.

These experiments were designed to assess the performance of our proposed approach in both regression and image classification tasks, demonstrating its effectiveness and versatility across different domains.

We compared our method to five widely used activation functions in the literature. We employed the ADAM optimization technique with a learning rate of 10^{-3}. The activation functions we compared with were ReLU [10], FReLU [11], ELU [10], PReLU [17], and MeLU [12].

Our convolutional neural network (CNN) architecture consisted of nine convolutional layers, specifically 3XConv-32, 3XConv-64, and 3XConv-128. Additionally, we incorporated two fully connected layers, FC-128 and FC-64x. Each convolutional layer utilized 3×3 kernel filters, and we applied 2×2 max-pooling layers with a stride size of 1. To enhance the model's performance, we implemented two regularization techniques, namely Batch Normalization, and Dropout. The dropout rate was determined through cross-validation and set to $p = 0.35$.

For coding purposes, we utilized the Python programming language along with the Keras and TensorFlow libraries. The experiments were conducted on a system with an Intel(R) Core(TM) i7-2720QM CPU 2.20GHZ architecture and 16 GB of memory.

A. Experiment 1: Regression

This section focuses on a regression task that utilizes the California Housing dataset, which consists of 20,640 instances. The dataset includes 8 predictive numerical attributes and a target variable. These attributes encompass various factors such as the median income in the block group (MedInc), the median house age in the block group (HouseAge), the average number of rooms per household (AveRooms), the average number of bedrooms per household (AveBedrms), the block group population (Population), the average number of household members (AveOccup), the latitude of the block group (Latitude), and the longitude of the block group (Longitude).

TABLE I
Experiment 1: Regression results (activation functions (Act Fcts), mean squared error with train set (MSE_{train}), mean squared error with test set (MSE_{test}).

Act. Fcts	MSE_{train}	MSE_{test}
MMeLU	**0.12**	**0.14**
ReLU	0.13	0.26
ELU	0.19	0.25
PReLU	0.15	0.28
FReLU	0.13	0.27
MeLU	0.14	0.28

Table I presents the results of Experiment 1, which compares different activation functions in the regression task. The mean squared error (MSE) is used as an evaluation metric, and the results are reported for the training set (MSE_{train}) and the test set (MSE_{test}).

The proposed MMeLU activation function achieves the best performance, outperforming all the competing activation functions. It achieves a significantly lower MSE_{test} value of 0.14, which is half the value obtained by the other activation functions. Moreover, the MSE value is more stable with MMeLU between train and test, indicating better generalization properties.
In contrast, the ReLU activation function shows a relatively high MSE_{test} value of 0.26, indicating that it may not be the most suitable choice for this regression task. Similarly, the ELU, PReLU, FReLU, and MeLU activation functions also exhibit higher MSE_{test} values compared to MMeLU.
These results highlight the importance of selecting an appropriate and flexible activation function for regression tasks. The MMeLU activation function demonstrates promising performance in accurately predicting the median house values in the California Housing dataset.

B. Experiment 2 : COVID-19 classification using CT images

In this section, we evaluate the effectiveness of our approach in classifying Covid-19 infections from other pneumonias in CT data. The COVID-CT dataset consists of 349 Covid-19 positive CT images from 216 patients and 397 Covid-19 negative CT images, which is publicly available[1]. We split the dataset into 566 training images and 180 test images, each with a size of 230×230. This task is challenging due to the complex nature of CT images and the similarity between Covid-19 infections and other types of pneumonia.
Table II presents the accuracy, loss, sensitivity, specificity, and computation time for the classification experiments. The reported scores indicate that our proposed Bayesian method outperforms all competing activation functions significantly. The results also demonstrate that the computation time is shorter with our proposed model. Additionally, even when employing various regularization techniques, there are notable drops in performance for all competing functions. This can be attributed to the inherent difficulty of classifying CT scan images due to their intricate content and the similarities between images of Covid-19 infections and other types of pneumonia.

TABLE II
EXPERIMENT 2: CT CLASSIFICATION RESULTS (ACTIVATION FUNCTIONS (ACT FCTS), COMPUTATION TIME IN MINUTES, ACCURACY (ACC), LOSS, SENSITIVITY (SENS) AND SPECIFICITY (SPEC)).

Act. Fcts	Time	Acc.	Loss	Sens.	Spec.
MMeLU	**61.92**	**0.91**	**0.21**	**0.87**	**0.87**
ReLU	81	0.77	0.39	0.74	0.72
ELU	97	0.76	0.46	0.75	0.75
PReLU	119	0.70	0.76	0.68	0.67
FReLU	123	0.77	0.52	0.76	0.75
MeLU	146	0.80	0.38	0.80	0.80

Figure 1 visually represents the behavior of the algorithms and demonstrates a significant improvement in accuracy observed with most activation functions. When compared to our proposed Bayesian approach, MMeLU, there are distinct differences in the accuracy and loss curves for all competing methods.
For instance, the LReLU function introduces a negative bias that suppresses excessive activations. However, if the bias value is inappropriate, it can result in underfitting. The ELU function allows for negative activations, but its exponential shape can cause activation values to explode for large values of x. The Swish function accelerates learning convergence but can lead to overfitting by being more sensitive to outliers. Lastly, the FReLU function is capable of capturing complex data patterns, but if the parameters are not well chosen, it can suffer from overfitting.
These significant differences are reduced with our MMeLU method, confirming its effectiveness and efficiency in challenging datasets.

C. Experiment 3 : CIFAR-10 image classification

In this section, we assess the performance of our approach using the standard CIFAR-10 dataset. This dataset comprises 60,000 color images, each sized 32×32 pixels, divided into 10 classes. Out of these, 50,000 images are used for training, while the remaining 10,000 images are used for testing.
The classification results for the CIFAR-10 dataset are presented in Table III. The superior performance of MMeLU can be attributed to its capability to capture more complex and diverse features, resulting in lower loss rates and higher accuracy. Moreover, MMeLU introduces a regularization effect that encourages sparsity within the network, leading to a more efficient model, as evident in Table III. This is reflected in the significantly lower computation time of our proposed method compared to other activation functions. As a result, the model is faster to train and requires less memory. The lower loss rate, higher accuracy, and sparsity nature of our global optimization method make it an excellent choice for real-world applications that require efficient and accurate deep learning models.

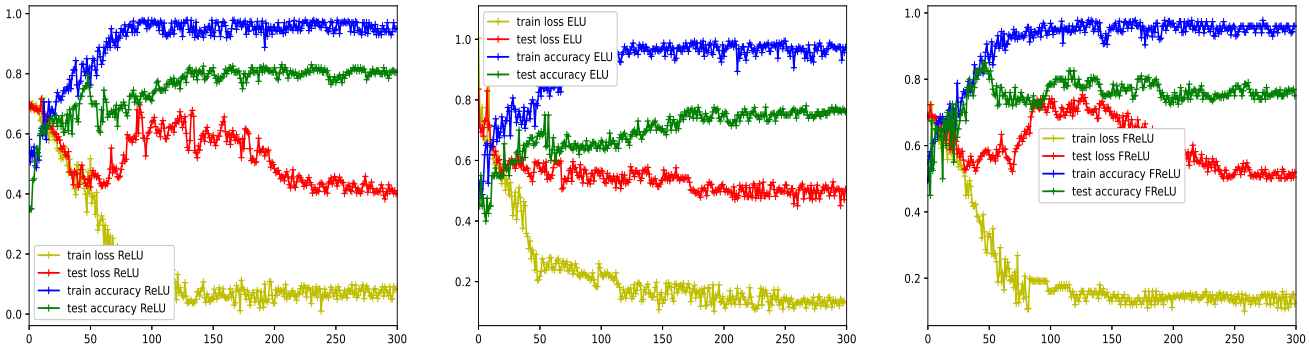
TABLE III
EXPERIMENT 3: CIFAR-10 CLASSIFICATION RESULTS (ACTIVATION FUNCTIONS (ACT FCTS), COMPUTATION TIME IN MINUTES, ACCURACY (ACC), LOSS, SENSITIVITY (SENS) AND SPECIFICITY (SPEC)).

Act. Fcts	Time	Acc.	Loss	Sens.	Spec.
MMeLU	**332.5**	**0.93**	**0.20**	**0.90**	**0.88**
ReLU	429	0.90	0.36	0.87	0.86
ELU	438	0.88	0.39	0.86	0.85
PReLU	466.8	0.84	0.47	0.81	0.79
FReLU	431.7	0.86	0.38	0.83	0.81
MeLU	485.6	0.90	0.34	0.88	0.87

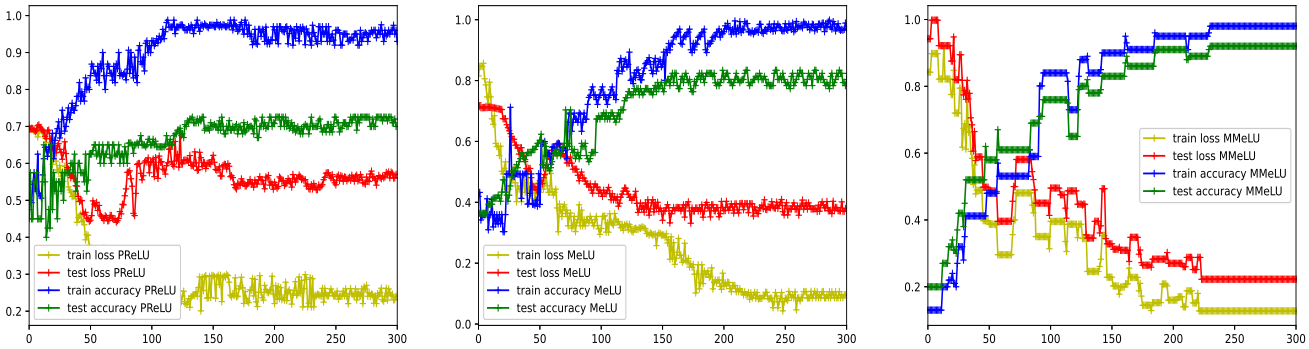
VI. CONCLUSION

The present paper introduces a novel Bayesian approach for enhancing the performance of deep neural networks with trainable activation functions. The proposed method achieves favorable classification results, high generalization properties, and improved computational efficiency when compared to

[1]https://www.kaggle.com/luisblanche/covidct



(a): ReLU (b): ELU (c): FReLU



(d): PReLU (e): MeLU (f): MMeLU

Fig. 1. Experiment 2: Train and test curves.

other activation functions and standard optimization methods. Future studies will concentrate on parallelizing the algorithm to enable GPU calculations and further reduce computational time.

REFERENCES

[1] M. Fakhfakh, B. Bouaziz, F. Gargouri, and L. Chaari, "Prognet: Covid-19 prognosis using recurrent and convolutional neural networks," *The Open Medical Imaging Journal*, vol. 12, no. 1, 2020.

[2] J. Giuxiang, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al., "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[3] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, "A survey on modern trainable activation functions," *Neural Networks*, vol. 138, pp. 14–32, 2021.

[4] C. Andrieu, A. Doucet, and R. Holenstein, "Particle markov chain monte carlo methods," *Journal of the Royal Statistical Society: Series B*, vol. 72, no. 3, pp. 269–342, 2010.

[5] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré, "Markov chain monte carlo without likelihoods," *Proceedings of the National Academy of Sciences*, vol. 100, no. 26, pp. 15324–15328, 2003.

[6] L. Chaari, H. Batatia, N. Dobigeon, and J.-Y. Tourneret, "A hierarchical sparsity-smoothness bayesian model for l0+l1+l2 regularization," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 1901–1905.

[7] M. Fakhfakh, B. Bouaziz, F. Gargouri, and L. Chaari, "Bayesian optimization using hamiltonian dynamics for sparse artificial neural networks," in *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2022, pp. 1–4.

[8] M. Fakhfakh, L. Chaari, B. Bouaziz, and F. Gargouri, "Non-smooth bayesian learning for artificial neural networks," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–24, 2022.

[9] A. C Marreiros, J. Daunizeau, S. J. Kiebel, and K. J. Friston, "Population dynamics: variance and the sigmoid activation function," *Neuroimage*, vol. 42, no. 1, pp. 147–157, 2008.

[10] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE transactions on cybernetics*, vol. 50, no. 8, pp. 3668–3681, 2019.

[11] S. Qiu, X. Xu, and B. Cai, "Frelu: flexible rectified linear units for improving convolutional neural networks," in *2018 24th international conference on pattern recognition (icpr)*. IEEE, 2018, pp. 1223–1228.

[12] G. Maguolo, L. Nanni, and S. Ghidoni, "Ensemble of convolutional neural networks trained with different activation functions," *Expert Systems with Applications*, vol. 166, pp. 114048, 2021.

[13] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *International conference on machine learning*, 2013, pp. 1319–1327.

[14] X. Wang, Y. Qin, Y. Wang, S. Xiang, and H. Chen, "Reltanh: An activation function with vanishing gradient resistance for sae-based dnns and its application to rotating machinery fault diagnosis," *Neurocomputing*, vol. 363, pp. 88–98, 2019.

[15] L. Chaari, J.-Y. Tourneret, C. Chaux, and H. Batatia, "A Hamiltonian Monte Carlo method for non-smooth energy sampling," *IEEE Trans. on Signal Process.*, vol. 64, no. 21, pp. 5585 – 5594, Jun. 2016.

[16] R. K. Pace and R. Barry, "Sparse spatial autoregressions," *Statistics & Probability Letters*, vol. 33, no. 3, pp. 291–297, 1997.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.