A Novel Hybrid Grid Search and Tree Parzen Estimator for Deep Learning Hyperparameters Optimization

Souhir Khessiba
Laboratoire de Technologie et Imagerie Médicale, Faculté de Médecine de Monastir, Université de Monastir
Institut Supérieur d'Informatique et des Technologies de Communication de Hammam Sousse, Université de Sousse
U2IS, ENSTA Paris, Institut Polytechnique de Paris,
Palaiseau, France
Sousse, Tunisie
souhir.khessiba@ensta-paris.fr

Antoine Manzanera
U2IS, ENSTA Paris, Institut Polytechnique de Paris,
Palaiseau, France,91120
antoine.manzanera@ensta-paris.fr

Ahmed Ghazi Blaeich
Laboratoire de Technologie et Imagerie Médicale, Faculté de Médecine de Monastir, Université de Monastir
Institut Supérieur des Sciences Appliquées et de Technologie de Sousse, Université de Sousse
Sousse, Tunisie
ahmedghazi.blaiech@issatso.u-sousse.tn

Khaled Ben Khalifa
Laboratoire de Technologie et Imagerie Médicale, Faculté de Médecine de Monastir, Université de Monastir
Institut Supérieur des Sciences Appliquées et de Technologie de Sousse, Université de Sousse
Sousse, Tunisie
khaled.benkhalifa@issatso.rnu.tn

Asma Ben Abdallah
Laboratoire de Technologie et Imagerie Médicale, Faculté de Médecine de Monastir, Université de Monastir
Institut supérieur d'informatique et de Mathématiques, Université de Monastir,
Sousse, Tunisie
assoumabab@gmail.com

Mohamed Hédi Bedoui
Laboratoire de Technologie et Imagerie Médicale, Faculté de Médecine de Monastir, Université de Monastir
Monastir, Tunisie
medhedi.bedoui@fmm.rnu.tn

***Abstract*—Hyperparameter optimization plays a crucial role in maximizing the performance of Deep Learning (DL) models, particularly in the medical field. In this study, we propose a novel hybrid approach called GS-TPE, which combines Grid Search (GS) and Tree Parzen Estimator (TPE) for optimizing the hyperparameters of DL architectures in order to enhance the vigilance states classification from the EEG signals. Our experiments demonstrate that the GS-TPE approach competes with the state of the art on multiple performance metrics, leading to significantly improved classification results. The obtained accuracy with combined one-Dimensional Convolutional Neural Network and Long Short-Term Memory (1D-CNN-LSTM) and with combined Auto-Encoder and LSTM (AE-LSTM) architectures reach 93.74 and 93.53%, respectively. The proposed GS-TPE approach shows great promise for advancing the field of medical signal analysis and enhancing the accuracy of EEG-based diagnostic systems.**

Keywords—Hyperparameter Optimization, Deep Learning, Tree Parzen Estimator (TPE), Grid Search (GS), Vigilance State Classification.

I. INTRODUCTION

Deep neural networks (DNNs) have successfully been applied across various data-intensive applications ranging from computer vision, bioinformatics and biomedical applications. Hyperparameters of a DNN are defined as parameters that remain fixed during model training and heavily influence the DNN performance. Hence, regardless of application, the design-phase of constructing a DNN model becomes critical. Framing the selection and tuning of hyper-parameters is an expensive black-box optimization problem, and obstacles encountered in manual by-hand tuning could be addressed by taking instead an automated algorithmic approach.

Recently, many researches were proposed for Hyperparameter Optimization (HPO) using different methods and several novel approaches were suggested. Grid Search (GS) [1], Bayesian Optimization (BO), Random Search (RS) [2] [3] and many heuristic algorithms have been used for HPO of CNN, such as Particle Swarm Optimization (PSO) and Tree Parzen Estimator (TPE) [4]. In addition to these algorithms, there are many studies in the literature that try to improve an existing optimization technique or develop a hybrid one by combining some of these optimization techniques. The simplest algorithm for HPO is GS which is used in [5], the authors specify a finite set of values for each hyperparameter, and GS evaluates the whole Cartesian product of these sets. This is very inefficient because the required number of function evaluations grows exponentially with the dimensionality of the hyperparameter space. In [6], an OLPSO (Orthogonal Learning Particle Swarm Optimization) approach was presented in which hyperparameters values were optimized for VGGNet network on plant disease diagnosis. The batch size and the dropout rate were used as hyperparameters. Through experiments, they proved that their approach achieves better performance and higher accuracy than other methods for the same data. In [7] a search approach was proposed to find the optimal model through hyperparameter tuning for a neural network using the uDEAS

method in order to minimize the cost. The proposed method was applied on two neural networks: Auto-Encoder (AE) and CNN, and was able to find the optimal hyperparameters setting with higher convergence rate and lower computational complexity than the classical random search. [8] studied a PSO and GS hybrid method for parameter optimization of SVM. They used GS to narrow down the search space and used PSO for detailed research in this confined search space. In [9], a hybrid optimization approach of Genetic Algorithm (GA) and PSO using Random Forest (RF), called GAPSO-RF, was developed in order to select optimized hyperparameters that improve the accuracy of cardiac disease prediction. In [10], the TPE algorithm has been proposed through the Hyperas tool in order to optimize CNN hyperparameters for classifying pulmonary nodules at an early stage.

In order to improve the performance of DL models, our study presents an innovative contribution by introducing a new hybrid method called Grid Search-Tree Parzen Estimator (GS-TPE) that combines GS and TPE, inheriting their strengths and compensating their weaknesses. The primary contribution of our research is to demonstrate the efficacy of GS-TPE in improving the classification performance of DL models on EEG signals. To validate this claim, the study compares the impact of the proposed GS-TPE approach with that of GS, Random Search (RS) or TPE alone on the same dataset. Furthermore, the study evaluates the performance of the DL models in their default state (i.e. with empirically defined hyperparameters), in order to highlight the added value of the GS-TPE hybrid approach.

The remainder of this paper is organized as follows. Section 2 introduces the DL architectures implemented for vigilance state classification, their hyperparameters and the HPO algorithms used. In Section 3, we describe the proposed new approach for hyperparameter optimization. Section 4 presents the dataset and the theoretical background describing EEG signal preprocessing for vigilance state detection and then provides experiments focusing on vigilance classification and performs a comparative study, comparing our approach with the baseline HPO algorithms namely GS, RS and TPE. The last section concludes the paper and gives some perspectives.

II. THEORETICAL BACKGROUND

In this section, we focus on the theoretical bases of our study. We first introduce the DL algorithms employed, providing a brief overview in terms of architecture and functionality. Next, we will detail the hyperparameters, highlighting the ones to be optimized. We will also present the HPO techniques used in our study.

A. DL algorithms

DL is a powerful subset of Machine Learning (ML) that is inspired by the structure and function of the human brain. It involves training artificial neural networks with multiple layers to learn complex patterns and make accurate predictions or decisions. DL models have shown exceptional performance in various domains. In this study, three neural network models were used for vigilance classification: an LSTM-based model, a 1D-CNN-LSTM model and an AE-LSTM model.

1) LSTM Architecture

In this study, we used a stacked four-layer LSTM network to process temporal data. The choice of using four LSTM layers is empirical. More specifically, we feed features vectors of size 21. These features vectors are derived from pre-processing applied to each 4-second segment of the raw signal. Each feature represents a frequency band. The network in Fig. 1 illustrates the details. Each feature vector is fed into the first LSTM layer (LSTM-1). This layer processes the data by calculating activations based on the current inputs and the hidden state of this layer, which is initialized to zero. The activations generated by LSTM-1 are then passed on to the second LSTM layer (LSTM-2). The LSTM-2 layer carries on this processing, using the activations received from LSTM-1 as well as the hidden state from the previous time step for its calculations. This process is repeated for the third (LSTM-3) and fourth layers (LSTM-4). At each time step, a decision is made. To classify the data, we use a fully connected layer (FC). In addition, between the LSTM-2 and LSTM-3 layers, we have added a dropout layer. This helps prevent overfitting. To process subsequent vectors, the same process is performed, but this time the hidden states of the LSTM layers are those calculated from the previous vector. That means that the relevant information extracted from the feature vector in step $n-1$ is used to process the feature vector at step n. By transferring hidden states from one step to another, we can capture temporal dependencies. The HPs selected for optimization are the number of units in the LSTM layers for the LSTM network and the Dropout rate. Also, the HPs related in particular to the network training process such as learning rate and batch size.

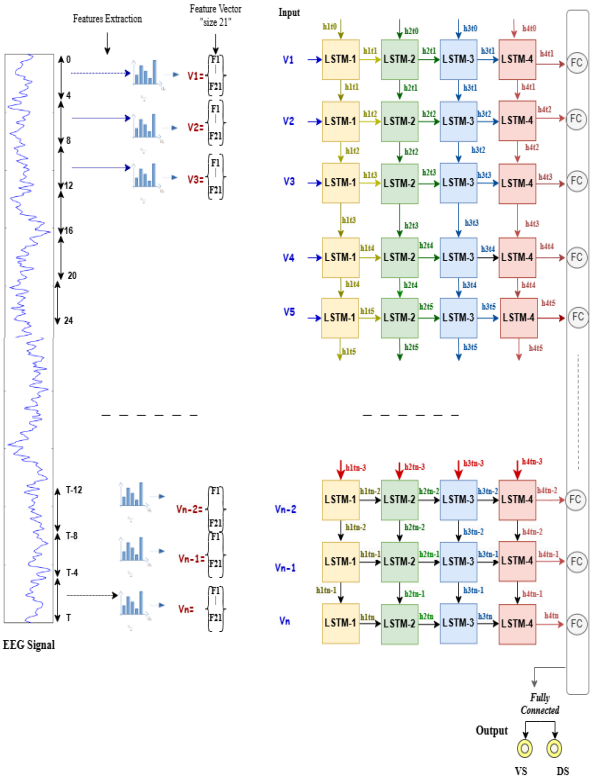

Fig. 1: LSTM Architecture

2) 1D-CNN-LSTM Architecture

The hybrid 1D-CNN-LSTM architecture is an interesting choice for vigilance state classification, by combining the advantages of CNN and LSTM networks [11] (Fig. 2). Firstly,

CNNs are used to identify and extract specific features from input data. Feature vectors are processed by a CNN made up of three blocks, each one integrating convolution-1D and pooling layers. The first two blocks also contain Dropout layers to prevent overfitting and enhance model generalization. The features extracted by the CNN are then fed into a stacked LSTM network, composed of three layers. LSTM are particularly effective in the capture of temporal dependencies in EEG data sequences, due to their ability to memorize information over long periods and to handle sequences with long-term dependencies. Such networks use memory cells and gate mechanisms to prevent the problem of vanishing gradients, allowing better storage of pertinent information over time. The main HPs selected for optimization are the number of filters in each of three convolution layers for the CNN, and the number of units in the LSTM layers for the LSTM network.

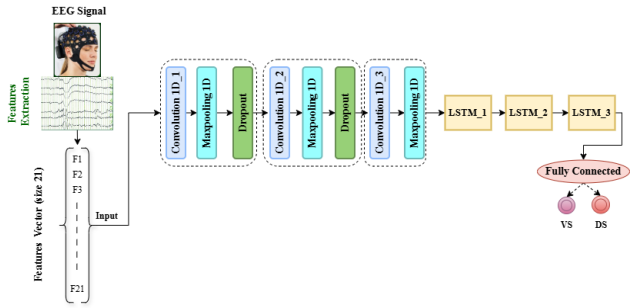

Fig. 2: 1D-CNN-LSTM Architecture

3) *AE-LSTM Architecture*

The AE-LSTM architecture combines the encoder element of an auto-encoder with an LSTM network. First, the auto-encoder is trained to learn a latent representation of the EEG signals. Where each vector within this representation has 6 features. The encoder output (the latent space) then serves as input to the LSTM network, which is composed of four layers (Fig.3).

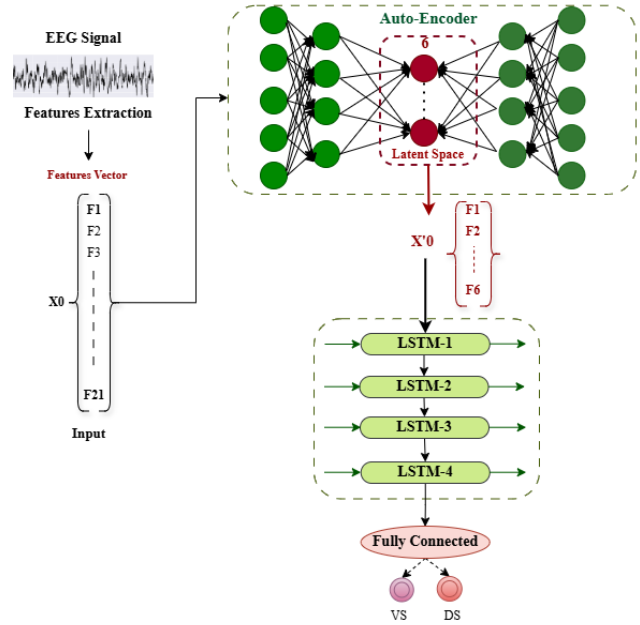

Fig. 3: AE-LSTM Architecture

One of the interesting aspects of this combination is that auto-encoders are designed to learn latent representations from input data. The size of the input data can be considerably reduced, while relevant information is retained. On the other hand, the use of LSTM enables the temporal dependencies within the EEG signal to be captured.
The HPs to be adjusted are mainly related to the LSTM network. They include the number of units in each LSTM layer and the HPs related in particular to the network training process such as learning rate and batch size.

B. *Hyperparameters (HP)*

HP are key variables that impact the behavior and performance of a ML algorithm. They are set before the training process and influence the learning process and the resulting model's capacity and complexity. Finding the optimal values of HP helps to improve performance and avoid problems such as overfitting. Hence, HPO is critical for the performance gain of any ML technique, particularly in DL, in which the number of HP is extremely large according to the number of layers. Therefore, the goal of HPO is to find out the best values of HP in any of the DL techniques as well as the best architecture size to reach the best performance in a test set. This optimization can be formalized as equation (1):

$$v = arg\,min\, f(x)_{x \in T} \qquad (1)$$

where $f(x)$ represents an "objective function" that is precisely one or more metrics that we want to minimize (such as error rate) or maximize (such as accuracy) evaluated in the validation set, v represents the set of HP that will fetch the optimal value from the objective function, and x can be any value in the domain T, which covers the range of values defined for the specified HP to be explored.

In this paper, we will use several optimization algorithms, notably GS, RS and TPE, in order to achieve HPO across different architectures. Table I details the HPs that need to be optimized for the three architectures presented above, as well as the defined search space used for GS, RS and TPE independently, and for GS-TPE and RS-TPE. It also shows the step size for GS, as well as the bounds used for each HP to create new search spaces for TPE while using TPE as part of the GS-TPE and RS-TPE approaches. The setting of the search space was based essentially on the assumption that empirical parameters used in the non-optimization versions were in the middle of each range. Among the HPs chosen for all architectures is the learning rate, the number of filters for convolution network architectures such as 1D-CNN-LSTM, and the number of units in LSTM layers for LSTM, AE-LSTM and 1D-CNN-LSTM. The batch size was also selected.

C. *Hyperparameters optimization algorithms*

In this section, we will present the hyperparameter optimization (HPO) algorithms employed in our study: Grid Search (GS), Random Search (RS) and Tree-structured Parzen Estimator (TPE).

1) *Grid Search algorithm (GS)*

The first algorithm used is Grid search (GS) which is an optimization method used in machine learning to optimize hyperparameters. It systematically explores all possible combinations of hyperparameter values within defined ranges to find the optimal configuration.

2) Random Search algorithm (RS)

Random search is also a method for optimizing hyperparameters in machine learning. But unlike grid search, it randomly selects and evaluates different combinations of hyperparameters within the search space [2].

3) Tree Parzen Estimator algorithm (TPE)

TPE is an enhanced version of the Bayesian Optimisation (BO) algorithm that addresses the limitations of traditional BO in handling classification and conditional parameters, resulting in improved efficiency. It is widely used for HPO in DL models, including CNN [4].
[12]. The primary procedure of the TPE algorithm involves initially transforming the HP space into a non-parametric density distribution and subsequently modeling the process $p(x|y)$. As shown in equation (2), TPE uses two density distributions of Equation to define $p(x|y), y < y*$, indicates that the value of the objective function is less than the threshold, and $y \geq y*$ denotes that the value of the objective function is greater than or equal to the threshold.

$$p(x|y) = \begin{matrix} l(x) & if \quad y < y* \\ g(x) & if \quad y \geq y* \end{matrix} \qquad (2)$$

The calculation of Expected Improvement (EI) is shown in Equations (3–5).

$$E(x) = \int_{-\infty}^{y*} (y* - y) \frac{p(x|y)\, p(y)}{p(x)}\, dy \qquad (3)$$

$$\alpha = p(y < y*) \qquad (4)$$

$$P(x) = \int P(x|y)\ P(y)\, dy \qquad (5)$$

Substitute (4), (5) into (3) to get the final (6)

$$EI_{y*}(x) = (\alpha + \frac{g(x)}{l(x)}(1-\alpha))^{-1} \qquad (6)$$

It can be seen from (6) that point x^* with the largest EI is the point with the smallest $g(x)/l(x)$. The TPE algorithm evaluates the improvement points according to $g(x)/l(x)$ in each iteration, and finally returns a point x^* with the largest EI.

III. PROPOSED METHODOLOGY

This section demonstrates a novel approach to building automated DL models using a metaheuristic algorithm for classification problems. We will detail the conceptual foundations and motivations that guided its design.

We propose a novel approach named GS-TPE based on a hierarchical combination of GS and TPE, which exploits the advantages of both. Initially, GS is used for a coarse grid exploration in hyperparameter space, in which a set of values is defined for each HP that needs to be optimized, and then a grid is generated that covers all possible combinations. Each combination is then evaluated on a validation set using a predefined metric, such as accuracy. Once all the combinations are evaluated, the 5 best configurations are selected. This number was selected taking into account computational time constraints to maintain the optimization process within a reasonable resource limit. Then, each one of these 5 HP configurations is used as initialization of a TPE algorithm, that refine the optimization by searching a better solution around each candidate HP. The HPs space, denoted by x in the flowchart, is then divided into five subspaces that will be explored independently by TPE. Basically, the TPE search strategy is divided into two phases: The first phase, called "warm-up", is a random exploration of a given space of hyperparameters (HPs), performing 20 iterations (n_init). Each HP combination builds a model which is then evaluated on a validation set to determine its performance in terms of accuracy. A function based on the Bayesian rule $p(x|y)$ is then constructed, where y denotes the validation accuracy and x the HP set. These HP sets are divided into two categories: good $l(x)$ and bad $g(x)$, based on a parameter γ fixed at 0.25, meaning that 25% of the combinations are considered as good.

TABLE I. HYPERPARAMETERS AND EXPLORATION RANGES BASED ON GS-TPE OPTIMIZATION APPROACH

Architectures	HP	Search Space	Step for GS	HP bounds used by TPE
LSTM	LSTM_1_unit	{32,128}	2^n, n∈{5,6,7}	± 20
	LSTM_2_unit	{32,128}	2^n, n∈{5,6,7}	± 20
	LSTM_3_unit	{16,64}	2^n, n∈{4,5,6}	± 20
	LSTM_4_unit	{8, 32}	2^n, n∈{3,4,5}	± 10
	Dropout	{0.1, 0.5}	0.2	± 0.1
	Batch size	{32,128}	2^n, n ∈{4,5,6,7}	± 16
	Learning rate	$\mathbb{R}[10^{-2}, 10^{-4}]$	10^{-1}	± 10^{-5}
1D-CNN-LSTM	Conv_1D-1 filters	{32,128}	2^n , n∈ {5,6,7}	± 12
	Conv_1D-2 filters	{32,128}	2^n , n∈ {5,6,7}	± 12
	Conv_1D-3 filters	{64,256}	2^n , n∈ {6,7,8}	± 12
	LSTM_1 unit	{10,50}	20	± 5
	LSTM_2 unit	{10,50}	20	± 5
	LSTM_3 unit	{5,25}	10	± 3
	Learning rate	$\mathbb{R}[10^{-2}, 10^{-4}]$	10^{-1}	± 10^{-5}
AE-LSTM	LSTM_1 unit	{30, 50}	10	± 6
	LSTM_2 unit	{30, 50}	10	± 6
	LSTM_3 unit	{30,50}	10	± 6
	LSTM_4 unit	{5, 25}	10	± 3
	Batch size	{16,128}	2^n, n∈{4,5,6,7}	± 10
	Learning rate	$\mathbb{R}[10^{-2}, 10^{-4}]$	10^{-1}	± 10^{-5}

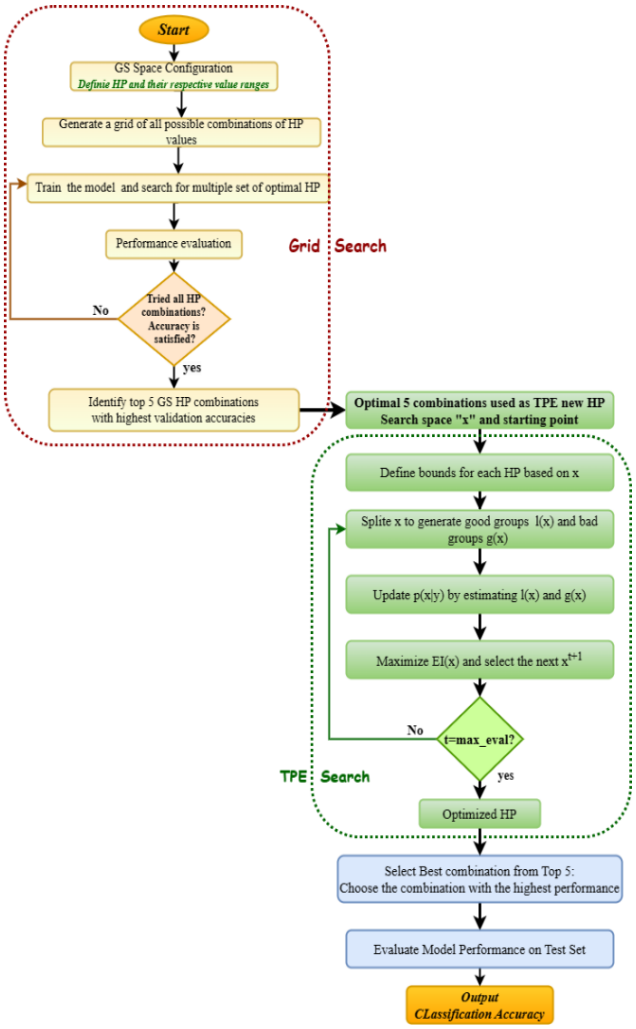

Fig. 4: Flowchart of the proposed GS-TPE approach

The aim of the second phase is to maximize the expected improvement (EI) ratio by selecting HPs x with high probability under $l(x)$ and low probability under $g(x)$. This is done by sampling n_EI combinations of HP values (n_EI =24) and the one offering the greatest EI improvement is selected, then the process is repeated, including all previous combinations, until the specified number of tests is reached. The process is repeated for the 5 HP search spaces defined in the study, and the optimum configuration is selected for evaluation on test data. The default TPE parameters used are: n_init = 20, γ=0.25, n_EI =24.

IV. EXPERIMENT AND RESULTS

The primary goal of this section is to illustrate the effectiveness of the proposed GS-TPE in enhancing the accuracy and effectiveness of vigilance state classification. We first describe the database we used, as well as how we prepared the data, the experimental setting and the results obtained.

A. EEG: Data and pre-processing

In this study, we focus on EEG signals. Such signals are highly relevant in the medical field. The traditional acquisition techniques are based on measuring the potential variation of the cerebral cortex activity on the scalp surface. The intensity and the shape of the EEG electrical activity depend strongly on the level of brain activity. This signal is a dynamic, stochastic and non-stationary electrical activity whose recording depends essentially on the position of the electrodes and also on the vigilance or sleep state of the subject. The EEG data are collected at the center of vigilance and sleep at the Faculty of Medicine of Monastir. We started from a collected set of six healthy subjects aged 18–23 which were used in previous work of our team [12] [13] [14]. For each subject, we carried out three 24-h recordings with a 15-day interval. Recordings are done for two states: vigilance state (VS) and drowsiness state (DS). Expert labeling of the EEG recordings is performed, reviewed and approved. In the preprocessing phase, we were interested mainly in achieving a size-reduced acquisition system: Only one EEG signal derivation was analyzed [the right parieto–occipital (Pz–Oz)]. For each 4s portion from an EEG signal, the spectral power was calculated by the Fast Fourier Transform (FFT) with a Hamming window and a 512-point resolution. From this spectrum, only the frequencies included in the interval [0.1–21 Hz] were conserved. This interval corresponds to the field of adherence of the physiological waves (lower than 21 Hz) while eliminating the continuous component (0 Hz frequency). Then, the power spectrum was subdivided into elementary frequency bands. Each band represents the sum of the spectral amplitudes included in the spectral interval corresponding to a frequency band. For each frequency band, we calculated the percentage of its relative spectral power (PRSP), which is equal to the spectral band power (SBP) divided by the total spectral power (TSP). Computing these PRSPs is done by equation (7):

$$PRSP_i = \frac{SBP_{[U_i + U_{i+1}]}}{TSP} * 100 \qquad (7)$$

where $u_i = 0.1 + (i-1) * \triangle u; \quad i \in [1, \ldots, k]$

and $\triangle u = \frac{(21-0.1)}{k}$

where $\triangle u$ is the length of the frequency band and k the number of bands. Actually, the [0.1-21 Hz] interval was discretized into k regular sub-intervals of $\triangle u$ length. Thereby, the $PRSP$ will be the input to the classification tool for vigilance state detection using DL architectures.

B. Database processing

In the case study, LSTM models require sequences as input, since they are designed to deal specifically with temporal dependencies within temporal sequences. In order to be able to train our LSTM-based models, i.e., LSTM, CNN-LSTM and AE-LSTM, it is important to generate sequences based on extracted EEG features. We therefore adopted the sliding window technique. A fixed sequence size of 5 was used for each patient. Table II describes the sequence creation process. We have 1494 feature vectors of 21 length, from which 1470 sequences were created. Next, these sequences were divided into training, validation and test sets.

TABLE II. SEQUENCE CREATION AND DISTRIBUTION OF EEG DATA

Number of samples	*Sequence length*	*Total sequences*	*Training sequences (60%)*	*Validation sequences (20%)*	*Test Sequences (20%)*
1494	5	1470	882	294	294

C. Experimental setting

1) Working environment

We have evaluated HPO algorithms using different architectures. That were implemented using Keras, whose libraries are written in Python. The hardware configuration included an NVIDIA GeForce RTX 3090 GPU with 32 GB and an 11th Gen Intel® core™ i9-11900F processor. In order to deal with HP optimization problems, we used the Optuna Framework, which provides a variety of algorithms including TPE.

2) Experimental setup for HP optimization

In this section, we describe the settings used in the optimization experiments for the different architectures. Each architecture was associated with a given set of GS combinations and a specified set of RS and TPE iterations. In order to provide a balanced and meaningful comparison between the different optimization methods, we established standardized criteria for each of them. For example, we decided to set the number of iterations for RS to be one-third of the total number of GS combinations. Similarly, we opted for 350 iterations for the TPE algorithm (Table III). We used the same settings for the GS-TPE algorithm.

TABLE III. EXPERIMENTAL CONFIGURATION

Architectures	*GS combinations*	*RS iterations*	*TPE iterations*
LSTM	2187	729	350
CNN-LSTM	2187	729	350
AE-LSTM	972	324	350

D. Discussion and results

In the context of HPO, we opted to evaluate the different baseline optimization algorithms, i.e., RS, GS, TPE, as well as our novel GS-TPE approach. Table IV presents the results obtained in terms of accuracy, which was evaluated on both validation and test data.

To further evaluate the effectiveness of RS and GS, we have compared the results obtained with these two algorithms across all models. The results show that, overall, GS performs slightly better than RS. As an example, for LSTM, GS achieved a validation accuracy of 92.99% and a test accuracy of 89.38%, compared to 91.16% and 87.55% for RS. For CNN-LSTM, significant differences were observed between the different optimization methods. Using RS, we achieved a validation accuracy of 94.56% and a test accuracy of 91.02%. Comparatively, GS improved on this, achieving a validation accuracy of 95.03% and a test accuracy of 92.38%

As for the TPE algorithm, we noticed that its use yielded better results. For 1D-CNN-LSTM architecture, the best performance reached 95.37% on validation set and 93.12% on test set compared to 89.79% with no optimization process. These results underline the effectiveness of TPE in refining HPs and improving model performance. The table clearly shows the positive impact of TPE integration on model performance for GS-TPE approach. When comparing GS with GS-TPE, it is clear that TPE integration significantly improves results by refining the HPs search. For the LSTM model, the improvement is more marked, with test accuracy increasing from 89.38% to 91.15% for GS and GS-TPE respectively. Also, for 1D-CNN-LSTM, there is an improvement (from 92.38% to 93.74%).

Table V shows the validation results for the different architectures using the GS-TPE approach, detailed for the 5 different configurations pre-selected by GS. The results demonstrate that integrating TPE systematically improves the performance obtained by GS.

For the LSTM model, the best accuracy performance is achieved in configuration 4, rising from 92.78% to 94.01% after TPE integration. Similarly, for the CNN-LSTM model, the best performance is obtained in configuration 3, reaching 97.48% (Fig.5). And for the AE-LSTM model, configuration 4 registered the best performance, upgrading to 95.31% from 94.76%.

TABLE IV. SUBJECTS VIGILANCE STATE CLASSIFICATION ACCURACY

	Without HPO	*RS*		*GS*		*TPE*		*GS-TPE*	
	Test Acc	*Validation ACC*	*Test Acc*	*Validation ACC*	*Test Acc*	*Validation ACC*	*Test Acc*	*Validation ACC*	*Test Acc*
LSTM	**86.39**	91.16	**87.55**	92.99	**89.38**	92.45	**90.20**	94.01	**91.15**
1D-CNN-LSTM	**89.79**	94.56	**91.02**	95.03	**92.38**	95.37	**93.12**	97.48	**93.74**
AE-LSTM	**88.43**	94.76	**91.97**	95.24	**92.58**	90.07	**88.37**	95.31	**93.53**

TABLE V. PERFORMANCE OF GS-TPE CONFIGURATIONS IN TERMS OF ACCURACY ON VALIDATION DATA FOR LSTM, CNN-LSTM, AND AE-LSTM ARCHITECTURES

	LSTM		*CNN-LSTM*		*AE-LSTM*	
	Accuracy/Validation_data (%)		*Accuracy/Validation_ data (%)*		*Accuracy/Validation_data (%)*	
	GS	*TPE*	*GS*	*TPE*	*GS*	*TPE*
Configuration-1	92.99	93.74	95.03	96.73	95.24	95.24
Configuration-2	92.85	93.19	94.89	97.00	94.89	95.17
Configuration-3	92.78	93.67	94.76	**97.48**	94.89	95.10
Configuration-4	92.78	**94.01**	94.62	96.32	94.76	**95.31**
Configuration-5	92.78	93.67	94.28	96.80	94.76	95.17

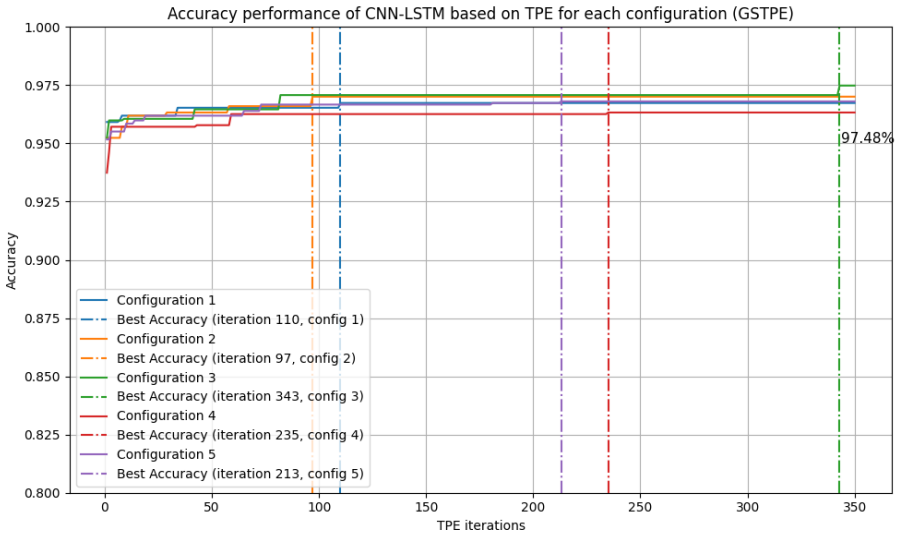

Fig. 5: Accuracy evolution of top 5 configurations with TPE iterations on validation data for 1D-CNN-LSTM (GS-TPE)

Table VI presents a comparison of performance metrics between LSTM and 1DCNN-LSTM, evaluated in terms of Recall, Precision, and F1-Score. The 1DCNN-LSTM model outperforms the LSTM with a Recall of 93.56% compared to 89.88%, a Precision of 94.30% versus 92.80%, and an F1-Score of 93.92% against 91.31%. Despite these improvements, GSTPE further boosts performance, enhancing these metrics even more for both models.

TABLE VI. PERFORMANCE METRICS COMPARISON BETWEEN LSTM AND 1DCNN-LSTM MODELS

	Recall (%)	***Precision (%)***	***F1-Score (%)***
LSTM	89.88	92.80	91.31
1DCNN-LSTM	93.56	94.30	93.92

Table VII presents the best HP configurations found using the GS-TPE approach for the 1D-CNN-LSTM architecture. It is divided into two parts: the first presents the initial configurations found by GS, and the second demonstrates the optimal configurations after TPE fine-tuning. the five best initial GS configurations were optimized and improved Among them, the initial configuration 3 (config 4), identified by the GS approach, was optimized into a new configuration (config 3') using the TPE algorithm, giving the best results. The improvements resulting from the TPE consist in re-evaluating and adjusting the units in the various LSTM layers. This change improves the results.

CONCLUSION AND PERSPECTIVES

This study proposed a novel hybrid approach GS-TPE that combines GS and TPE for optimizing hyperparameters in DL models. The results obtained from this approach clearly demonstrate its effectiveness in improving models performance. The results also demonstrate its superiority over non-optimized models and its improvement over both TPE and the basic GS algorithm. As a result, this study highlights the importance of advanced optimization techniques in maximizing the performance of DL models. Moving forward, future research can explore several perspectives to build upon the findings of this study. Firstly, investigating the scalability of the hybrid GS-TPE approach to more subjects and more complex DL architectures would provide valuable insights into its adaptability and generalizability.

TABLE VII. OPTIMUM HP CONFIGURATIONS ACHIEVED WITH THE GS-TPE APPROACH FOR 1D CNN-LSTM ARCHITECTURE

	Best configuration found by GS				
	Config-1	*Config-2*	*Config-3*	*Config-4*	*Config-5*
Conv_1D-1 filters	128	128	128	128	128
Conv_1D-2 filters	64	64	32	32	128
Conv_1D-3 filters	128	128	128	64	128
LSTM_1 unit	30	50	30	50	10
LSTM_2 unit	50	50	10	30	50
LSTM_3 unit	25	25	15	25	15
Learning rate	0.001	0.001	0.001	0.001	0.001
	Optimal new configurations resulting from TPE				
	Config-1'	*Config-2'*	*Config-3'*	*Config-4'*	*Config-5'*
Conv_1D-1 filters	131	127	131	119	131
Conv_1D-2 filters	52	63	42	27	137
Conv_1D-3 filters	116	129	130	70	124
LSTM_1 unit	34	54	28	55	9
LSTM_2 unit	52	45	10	26	48
LSTM_3 unit	24	25	14	26	14
Learning rate	0.001	0.001	0.0009	0.0009	0.0009

REFERENCES

[1] D. Belete et M. D H, « Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results », *Int. J. Comput. Appl.*, vol. 44, p. 1-12, sept. 2021, doi: 10.1080/1206212X.2021.1974663.

[2] J. Bergstra et Y. Bengio, « Random search for hyper-parameter optimization », *J Mach Learn Res*, vol. 13, nº null, p. 281-305, févr. 2012.

[3] R. Andonie et A.-C. Florea, « Weighted Random Search for CNN Hyperparameter Optimization », *Int. J. Comput. Commun. CONTROL*, vol. 15, nº 2, mars 2020, doi: 10.15837/ijccc.2020.2.3868.

[4] S. Watanabe, « Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance », 2023, doi: 10.48550/ARXIV.2304.11127.

[5] B. H. Shekar et G. Dagnew, « Grid Search-Based Hyperparameter Tuning and Classification of Microarray

Cancer Data », in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, févr. 2019, p. 1-8. doi: 10.1109/ICACCP.2019.8882943.

[6] A. Darwish, D. Ezzat, et A. E. Hassanien, « An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis », *Swarm Evol. Comput.*, vol. 52, p. 100616, févr. 2020, doi: 10.1016/j.swevo.2019.100616.

[7] Y. Yoo, « Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches », *Knowl.-Based Syst.*, vol. 178, p. 74-83, août 2019, doi: 10.1016/j.knosys.2019.04.019.

[8] T. Xiao, D. Ren, S. Lei, J. Zhang, et X. Liu, « Based on grid-search and PSO parameter optimization for Support Vector Machine », in *Proceeding of the 11th World Congress on Intelligent Control and Automation*, juin 2014, p. 1529-1533. doi: 10.1109/WCICA.2014.7052946.

[9] M. G. El-Shafiey, A. Hagag, E.-S. A. El-Dahshan, et M. A. Ismail, « A hybrid GA and PSO optimized approach for heart-disease prediction based on random forest », *Multimed. Tools Appl.*, vol. 81, nº 13, p. 18155-18179, mai 2022, doi: 10.1007/s11042-022-12425-x.

[10] J. Konar, P. Khandelwal, et R. Tripathi, *Comparison of Various Learning Rate Scheduling Techniques on Convolutional Neural Network*. 2020, p. 5. doi: 10.1109/SCEECS48394.2020.94.

[11] S. Khessiba, A. Blaiech, K. Ben Khalifa, A. Ben Abdallah, et M. H. Bedoui, « Correction to: Innovative deep learning models for EEG-based vigilance detection », *Neural Comput. Appl.*, vol. 34, janv. 2022, doi: 10.1007/s00521-021-06187-0.

[12] S. Khessiba, A. G. Blaiech, A. Manzanera, K. Ben Khalifa, A. Ben Abdallah, et M. H. Bedoui, « Hyperparameter Optimization of Deep Learning Models for EEG-Based Vigilance Detection », in *Advances in Computational Collective Intelligence*, vol. 1653, C. Bădică, J. Treur, D. Benslimane, B. Hnatkowska, et M. Krótkiewicz, Éd., in Communications in Computer and Information Science, vol. 1653. , Cham: Springer International Publishing, 2022, p. 200-210. doi: 10.1007/978-3-031-16210-7_16.

[13] B. K. Khalifa, M. H. Bedoui, M. Dogui, et F. Alexandre, « Analysis of vigilance states by neural networks », in *Proceedings. 2004 International Conference on Information and Communication Technologies: From Theory to Applications, 2004.*, avr. 2004, p. 429-430. doi: 10.1109/ICTTA.2004.1307815.

[14] A. G. Blaiech, K. Ben Khalifa, M. Boubaker, et M. H. Bedoui, « LVQ neural network optimized implementation on FPGA devices with multiple-wordlength operations for real-time systems », *Neural Comput. Appl.*, vol. 29, nº 2, p. 509-528, janv. 2018, doi: 10.1007/s00521-016-2465-7.