



## Energy Efficient Hyperparameter Tuned Deep Neural Network to Improve Accuracy of Near-Threshold Processor

K. Chanthirasekaran and Raghu Gundaala\*

Department of Electronics and Communication Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, 602105, Tamilnadu, India

\*Corresponding Author: Raghu Gundaala. Email: raghu.gundaala44@gmail.com

Received: 18 September 2022; Accepted: 13 January 2023

**Abstract:** When it comes to decreasing margins and increasing energy efficiency in near-threshold and sub-threshold processors, timing error resilience may be viewed as a potentially lucrative alternative to examine. On the other hand, the currently employed approaches have certain restrictions, including high levels of design complexity, severe time constraints on error consolidation and propagation, and uncontaminated architectural registers (ARs). The design of near-threshold circuits, often known as NT circuits, is becoming the approach of choice for the construction of energy-efficient digital circuits. As a result of the exponentially decreased driving current, there was a reduction in performance, which was one of the downsides. Numerous studies have advised the use of NT techniques to chip multiprocessors as a means to preserve outstanding energy efficiency while minimising performance loss. Over the past several years, there has been a clear growth in interest in the development of artificial intelligence hardware with low energy consumption (AI). This has resulted in both large corporations and start-ups producing items that compete on the basis of varying degrees of performance and energy use. This technology's ultimate goal was to provide levels of efficiency and performance that could not be achieved with graphics processing units or general-purpose CPUs. To achieve this objective, the technology was created to integrate several processing units into a single chip. To accomplish this purpose, the hardware was designed with a number of unique properties. In this study, an Energy Efficient Hyperparameter Tuned Deep Neural Network (EEHPT-DNN) model for Variation-Tolerant Near-Threshold Processor was developed. In order to improve the energy efficiency of artificial intelligence (AI), the EEHPT-DNN model employs several AI techniques. The notion focuses mostly on the repercussions of embedded technologies positioned at the network's edge. The presented model employs a deep stacked sparse autoencoder (DSSAE) model with the objective of creating a variation-tolerant NT processor. The time-consuming method of modifying hyperparameters through trial and error is substituted with the marine predators optimization algorithm (MPO). This method is utilised to modify the hyperparameters associated with the DSSAE model. To validate that the proposed EEHPT-DNN model has a higher degree of functionality, a full simulation study is conducted, and the



This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

results are analysed from a variety of perspectives. This was completed so that the enhanced performance could be evaluated and analysed. According to the results of the study that compared numerous DL models, the EEHPT-DNN model performed significantly better than the other models.

**Keywords:** Deep learning; hyperparameter tuning; artificial intelligence; near-threshold processor; embedded system

## 1 Introduction

Their use for real-time inference on sensor-rich platforms such as the Internet of Things (IoT), wearables, personal biomedical devices, and drones is hampered by the energy and latency costs of existing machine learning (ML) approaches [1]. These applications [2] include devices that collect and analyse data to provide interpretations and actions that may be used to automate or monitor a variety of activities without human intervention. These types of methods must be able to include computationally intensive machine learning techniques while adhering to severe limitations on form factor, energy, and latency [3]. Memory access is known to reduce both the latency and the energy cost associated with machine learning algorithm comprehension. In recent years, several machine learning systems that utilise energy-efficient digital architectures and integrated circuits (IC) have been created. These applications employ methods like as efficient data flow, data reuse, and the optimization of computations to reduce the amount of memory accesses [4].

In order to conserve the needed amount of energy, the authors have reached a consensus on using this complete technique. This is owing to the stringent energy restrictions put on them. This technique covers every component of a computer system, from hardware to software [5]. Near-exact computing, also known as approximation computing, is one of the most cost-effective ways to preserve energy. This type of computing is sometimes referred to as “approximate computing.” This technology sacrifices application precision in order to reduce energy consumption. The assumption that certain applications, but not many, required the level of accuracy afforded by programming methods or current circuits was the impetus for the invention of this technology [6]. As an illustration, video decryption can tolerate a substantial amount of mistake since image variations cannot be seen by the human eye. This enables the utilisation of video decryption. It is anticipated that multicore architectures (embedding and increased performance) would rely largely on accelerators for many computationally intensive tasks (heterogeneous multicores). This is because the lack of voltage scaling leads to a phenomenon known as Dark Silicon. This architectural trend was motivated by the increasing limits put on accessible energy. Beyond typical accelerator solutions such as Application Specific Integrated Circuits (ASICs), Graphics Processing Units (GPUs), and Field Programmable Gate Arrays (FPGAs), writers are studying co-processors or accelerators with an intermediate scope in terms of generality. These co-processors or accelerators contain more applications than ASICs but less than general-purpose processors. In compared to FPGAs and GPUs, this is done in order to maximise energy efficiency. In particular, the authors [7] Coprocessors and accelerators whose capabilities fall somewhere in the centre. The present practise of integrating many accelerators into a single chip may lead the combined application scopes of these accelerators to encompass a vast amount of territory. This development is a result of the present practice of incorporating many accelerators on a chip. The second observed trend may be attributable to applications that are getting more computationally intensive.

Deep neural networks (DNNs) have demonstrated exceptional performance in cognitive tasks such as image and speech recognition [8]. This achievement has transformed the field of artificial intelligence (AI), and DNNs are also known as DNNs. Edge computing applications have a great interest in discovering platforms that could deploy the well-trained approach of these networks and perform inference in an energy-efficient manner. Specifically, battery-powered Internet of Things devices and autonomous vehicles benefited greatly from the speed, energy economy, and reliability of DNN's inference engines [9]. [Bibliography required] With the development of specialised hardware for inference that runs at reduced digital precision (four to eight bits), such as low-power graphics processing units and Google's tensor processing unit1 like NVIDIA T42, significant progress was achieved in this field. This was a significant advance in the correct direction. These are a few examples of something known as a low-power graphics processing unit. Even while such platforms have the potential to be extremely flexible, they nonetheless require a framework in which the processor and memory units are physically separated [10]. Even if such platforms have the potential to be very adaptable, they still require such a framework. There was a continuous flow of data between the memory and the processors since the procedures were typically kept in off-chip memory. This resulted in frequent memory access and modification. As a result, the maximum level of energy efficiency that could be achieved with the available resources reduced. In this study, an Energy Efficient Hyperparameter Tuned Deep Neural Network (EEHPT-DNN) model for Variation-Tolerant Near-Threshold Processor was developed. In order to improve the energy efficiency of artificial intelligence (AI), the EEHPT-DNN model employs several AI techniques. This model focuses mostly on the consequences of incorporating AI into embedded devices at the network's edge. A deep stacked sparse auto encoder (DSSAE) model is used to meet the objective of constructing a variation-tolerant NT processor, according to the model provided. Due to the tedious nature of the trial-and-error approach of tuning hyperparameters, the marine predators' optimization (MPO) algorithm is used to optimise the DSSAE model's hyperparameters. This is because the conventional approach for adjusting hyperparameters involves trial and error. To demonstrate that the suggested EEHPT-DNN model has enhanced performance, an extensive number of simulation evaluations have been conducted, and the results have been analysed in a variety of methods.

## 2 Related Works

Lin et al. [11] have developed a variation-tolerant framework for Convolutional Neural Networks (CNNs) that can operate from near threshold voltage (NTV) systems to maximise energy efficiency. The researchers introduced this framework to CNNs. This article focused on the statistical error compensation (SEC) method, sometimes referred to as rank decaying statistical error compensation (RD-SEC). Utilizing the intrinsic redundancy found in matrix-vector multiplication, a technique that consumes a great deal of power in CNNs, was the driving force behind the development of a less costly estimator for error identification and compensation. Utilizing the intrinsic redundancy found in matrix-vector multiplication, this was done. Gundi and his coworkers[12] are investigating the PREDITOR, a low-power TPU that functions in a Near-Threshold Computing (NTC) environment. PREDITOR employs scientific research to mitigate undetectable timing errors by boosting the voltages of select multiplier-and-accumulator parts at specific intervals to increase the efficiency of NTCTPU, which helps to ensure a greater inference precision at lower voltages. PREDITOR does this by increasing the voltages of chosen multiplier-and-accumulator components at specific intervals in order to improve the performance of NTCTPU. Lin et al. [13] present a variation-tolerant framework for CNN that is capable of robust functions supplied by an NTC system for the purpose of improving energy efficiency. In particular, the authors produce effective CNNs by employing two methods that

are both affordable and inaccurate. Each of these strategies has its own error statistics, such as an NTC technique with a precision of one hundred percent and a K-means approximation strategy that clusters weighted vectors from CNN to simplify the procedure.

Wang et al. [14] study a 3D-optimized approach that, when correctly applied, efficiently calculates the optimal classification configuration for the goal of achieving a balance between energy, reliability, and efficiency. The introduction of an efficient recognition system in the method makes this feasible. The authors employ a technique known as dynamic programming to determine the optimal voltage and approximation level by basing their conclusion on three different predictors: the output quality, the system's efficiency, and the amount of energy consumed. The authors provide a resultant quality predictor that utilises a hardware or software co-design error injection platform to analyse the effect of errors on resultant quality in NTC. This was done so that the writers could evaluate if flaws influenced the quality of the final product. Using this platform, one may make predictions regarding the eventual product's quality. The authors of the work [15] describe a model that reduces the number of pessimistic margins by limiting the number of temporal failures. As a result, the margin of error will be diminished. In particular, the authors investigate a model with the objective of minimising, to the greatest extent feasible, the number of processor pipeline stages with limitations and the number of stages with greater latency pathways. Because of this strategy, we are not only able to reduce the number of time failures, but also the number of sites in the pipeline that are susceptible to effective errors.

Zhu et al. [16] develop an error-resilient integrated clock gate (ERICG) and a technique for incorporating it into the design flow of an error detection and repair mechanism. ERICCG is an abbreviation for error-resilient integrated clock gate (EDAC). The ERICG may provide the EDAC with the capability of in-situ timing with only four additional transistors above and above what is typically required for integrated clock gates. This would be an improvement above the existing circumstance. Zhu et al. [17] provide a self-gated error-resilient cluster of sequential cells referred to as SGERC for gathering potentially hazardous data during wide-voltage manipulation for the EDAC method. This cluster of cells is error-tolerant and self-gating. In error-resistant circuits, the SGERC will for the first time introduce a latch-related clock gating approach and produce clock gates with the capability of timing error self-correction with the addition of just two extra transistors. This will be an unprecedented occurrence. In addition, it eliminates the timing error recognition circuits that were previously required for every main register and replaces them with data-driven clock gating circuits that create timing error data. These circuits were required for every previous significant register, but this register eliminates them entirely.

### **3 The Proposed Model**

In this article, a new EEHPT-DNN method was formulated for Variation-Tolerant Near-Threshold Processor. The presented EEHPT-DNN model makes use of AI techniques for energy efficient AI, concentrated on the implication of embedded system at the network edge.

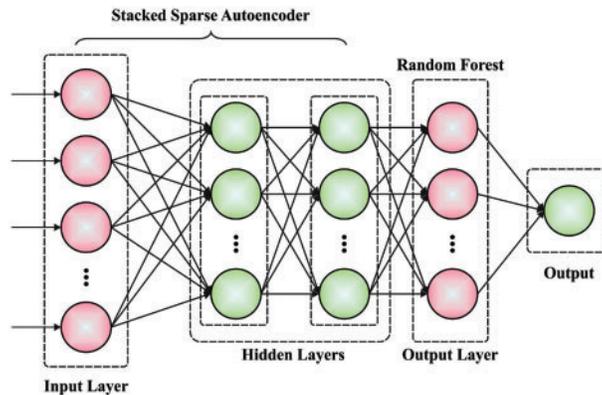
#### ***3.1 Design of DSSAE Model***

For the development of a variation-tolerant NT processor, the suggested model utilised the DSSAE paradigm. AE-NN refers to an unsupervised learning technique that modifies the objective value to match the input value [18] by employing the BP learning methodology. This method of learning matches the input value to the objective value. The AE was trained to study the function  $y_i = i$ , and by setting limitations on the number of hidden layers (HLs) and the sparsity of hidden units, it discovered amazing patterns in the input dataset. In reality, this easy AE is identical to Principal

Component Analysis (PCA); however, although PCA provides linear conversion, AE is nonlinear and has the capacity to discover intricate correlations between hidden and visible layers; the network gains a condensed presentation of input. Hidden units are subject to the sparsity restrictions set by the AE. This enables the identification of noteworthy traits within the dataset. As soon as the deep network receives the input dataset, it will first “change” the data to a compact representation to “compress” it (using an encoding technique), and then attempt to reconstruct it (using a decoding technique). The purpose of training a network is to enhance its reconstruction accuracy and identify an efficient compressed representation (encode) of the dataset it is tasked with processing. There are several levels of AE, each containing layers that are apparent, hidden, and reconstructive. These layers are present on every level. The visible (input) layer of the second level of AE corresponds to the HL of the first level of AE, whereas the visible layer of the third level of AE corresponds to the HL of the second level of AE. After the initial level AE has been trained to deliver the required output of the initial level feature in the HL, the feature is transferred to the second level AE as input. Similarly, each AE receives training to reach high levels of functioning at the greatest degree feasible. Fig. 1 illustrates the SSAE’s architecture. AE training may consist of: The encoder extracted the h features from the input, and the decoder recreates the y output using the h extracted features, as seen below:

$$h = f(w_e x + b_e) \tag{1}$$

$$f = f(w_d h + b_d) \tag{2}$$



**Figure 1:** Framework of SSAE

Then,  $w_e$  and  $w_d$  denotes weight matrix of encoder and decoder  $b_e$  and  $b_d$  characterizes the bias vector.  $f(\cdot)$  represents the activation function of AE.

The objective is to minimize the error function among the reconstructed dataset of reconstruction layer and input dataset of the visible layer. The primary term represents the average summation of square error and the next term represents weight degradation term, whereas the 3rd term represents sparse penalty term:

$$c(x_t, y) = \frac{1}{2n} \sum_{j=1}^n \|x^j - y^j\|_2^2 + \frac{\lambda}{2} (\|w_e\|_2^2 + \|w_d\|_2^2) + \beta \sum_{j=1}^m KL(\rho \| \hat{\rho}_j) \tag{3}$$

whereas:

$n$  = the number of input dataset

$\lambda$  = the weight degradation variable that controls relative momentousness of primary and 2nd second terms

$\beta$  = variable control weight of sparse penalty term

$m$  = number of neurons in hidden unit

$KL(\rho||\hat{\rho})$  indicates kullback-leibler divergence between mean  $\rho$  and mean  $\hat{\rho}_j$  bernoulli random parameter.  $\hat{\rho}_j$  represents the average activation of  $j$  hidden neuron or averaged over the trainedset and  $\rho$  indicates a sparsity variable, generally, the smallest value nearest to 0 ( $\rho = 0.05$ )

$KL(\rho||\hat{\rho}_j)$  can be evaluated as follows:

$$KL(\rho||\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{(1 - \rho)}{(1 - \hat{\rho}_j)} \quad (4)$$

whereas

$$\hat{\rho}_j = \frac{1}{n} \sum_{i=1}^n h_j(x^{(i)}) \quad (5)$$

The number of AE levels and hidden neurons in every level were defined using trial and error.

The higher-level feature or hidden neurons are provided to the softmax regression method, i.e., supervised learning model in DNN, and categorizes the input dataset.

Assume  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(n)}, y)\}$  tuples as input dataset, the output  $y$  could take on

$f$  distinct classes. Then  $y^{(i)} \in \{1, 2, 3, f\}$ . The aim was to forecast the probability  $p(y = z|x)$  for  $z = 1, 2, f$ :

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y)^1 = 1|x^{(i);\theta} \\ p(y)^2 = 2|x^{(i);\theta} \\ \vdots \\ p(y)^f = f|x^{(i);\theta} \end{bmatrix} = \frac{1}{\sum_{z=1}^f e^{\theta_z^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_f^T x^{(i)}} \end{bmatrix} \quad (6)$$

whereas  $\theta_1, \theta_2, \theta_f$  are model parameter,  $\theta = \begin{bmatrix} \theta_1^T \\ \theta_2^T \\ \vdots \\ \theta_f^T \end{bmatrix}$ , the term  $\frac{1}{\sum_{z=1}^f e^{\theta_z^T x^{(i)}}}$  cause the distribution to sum to one and normalize it.

The cost functions of softmax regression classification mechanism are shown below:

$$J(\theta) = -\frac{1}{n} \left[ \sum_{i=1}^n \sum_{z=1}^f 1\{y^{(i)} = z\} \log \frac{e^{\theta_z^T x^{(i)}}}{\sum_{l=1}^f e^{\theta_l^T x^{(i)}}} \right] + \frac{\lambda}{2} \sum_{i=1}^f \sum_{j=0}^u \theta_{ij}^2 \quad (7)$$

whereas  $1$  indicates the indicator functions,  $1\{\text{true expression}\} = 1$  and  $1\{\text{false expression}\} = 0$ . The weight decayed term  $\frac{\lambda}{2} \sum_{i=1}^f \sum_{j=0}^u \theta_{ij}^2$  punishes higher value of the parameter, and  $u+1$  indicates dimension

of feature vector  $\chi$ . The cost function  $J(\theta)$  is certain to have a single solution as gradient descent, L-BFGS, and so on. Because it is completely convex with the weight decay term.

### 3.2 Hyperparameter Tuning Using MPO Algorithm

At this stage, the MPO method is used to fine-tune the various hyperparameters of the DSSAE model. MPO, a metaheuristic influenced by nature, was created [19]. It was inspired by the vast foraging strategy adopted by ocean predators. The predator utilises both the Lévy flight and the Brownian motion techniques when hunting. Both the prey and the predator will update their location in MPO based on Lévy flight or Brownian motion. The MPO comprises four essential phases, which are outlined below.

#### Initialization Phase

In this section, population individual is randomly generated using the uniform distributed technique, represented  $X_0 = X_{\min} + \text{rand}(X_{\max} - X_{\min})$ .  $X_{\min}$  and  $X_{\max}$  is lower and upper limits of the parameter, correspondingly.  $\text{rand}$  denotes a random number within  $[0, 1]$ . Next, evaluate the fitness function value of each individual. At last, create Prey matrix and Elite matrix. The population individual updates its location according to the Elite and Prey matrixes [20].

#### MPO Optimization Scenarios

The method of optimization known as MPO can be broken down into three primary stages, each of which is determined by the distinct velocity ratios of the prey and the predator. At first, the prey is able to move more quickly than the predator. After that, both the predator and the prey move at an almost identical speed. When this occurs, the predator is able to outpace the prey. According to the rules that govern the behaviour of prey and predator movements, a particular amount of iteration time is specified and allotted to each phase.

**Phase 1:** In the early phase of optimization iteration, in higher velocity ratio ( $v \geq 10$ ), the better strategy for predator isn't to move at all. The mathematical modelling of this stage is given below.

Where  $\text{Iter} < \frac{1}{3} \text{Max\_Iter}$ ,

$$\text{stepsize}_i = R'_B \otimes (\text{Elite}_i - R'_B \otimes \text{Prey}_i) \quad i = 1, \dots, n \quad (8)$$

$$\text{Prey}_i = \text{Prey}_i + P.R \otimes \text{stepsize}_i$$

$R$  denotes a vector of uniform random numbers within  $[0, 1]$ .  $\text{Iter}$  denotes the existing iteration number. Indicates a vector comprising random numbers according to the standard distribution demonstrating the Brownian movement. The standard Brownian movement was a random process. The step size has the features of unit variance ( $\sigma^2 = 1$ ) and zero mean ( $\mu = 0$ ). The notation  $\otimes$  illustrates the entry-wise multiplication.

**Phase 2:** In unit speed ratio, predators try to make transition from exploration to exploitation. Subsequently, half of the organism is earmarked for exploration and other half for exploitation.

Where  $\frac{1}{3} \text{Max\_Iter} < \text{Iter} < \frac{2}{3} \text{Max\_Iter}$ , the initial half of individual updates its position according to the subsequent formula:

$$\text{stepsize}_i = R'_B \otimes (\text{Elite}_i - R'_B \otimes \text{Prey}_i) \quad i = 1, \dots, \frac{n}{2} \quad (9)$$

$$\text{Prey}_i = \text{Elite}_i + P.CF \otimes \text{stepsize}_i$$

And next half of individual updates the location according to the subsequent formula:

$$stepsize_i = R_B^r \otimes (Elite_i - R_B^r \otimes Prey_i) \quad i = \frac{n}{2}, \dots, n \quad (10)$$

$$Prey_i = Elite_i + P.CF \otimes stepsize_i$$

$$CF = \left(1 - \frac{Iter}{Max\_Iter}\right)^{\left(2 \frac{Iter}{Max\_Iter}\right)}$$

$R_L$  indicates a vector of arbitrary value according to the Lévy distribution demonstrating Lévy movement. CF denotes an adaptive variable for controlling step size for predator movement.  $R_B \otimes Elite$  denotes the Brownian motion of predator chasing its prey. Also, the prey upgrades its location according to the predator in Brownian movement.

**Phase 3:** Lower velocity ratio or if the predator moves faster than prey.

Where  $Iter > \frac{2}{3} Max\_Iter$ , individual updates its location according to the subsequent formula.

$$stepsize_i = R_L^r \otimes (Elite_i - R_L^r \otimes Prey_i) \quad i = 1, \dots, n \quad (11)$$

$$Prey_i = Elite_i + P.CF \otimes stepsize_i$$

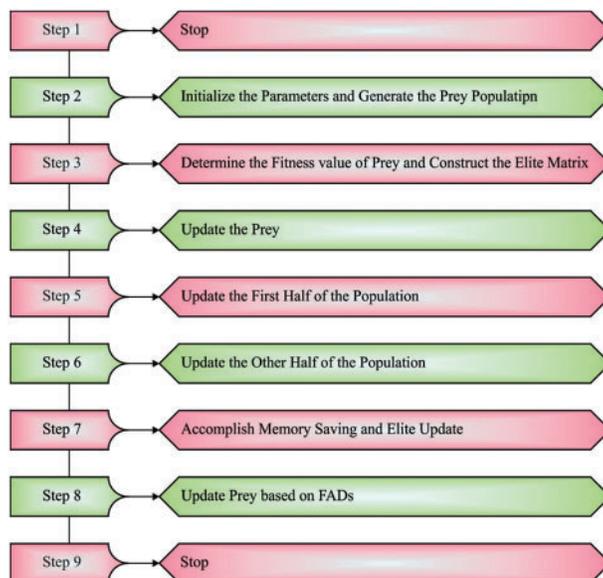
$R_L \otimes Elite_i$  simulated the movement of predators in the Lévy strategy.

The environmental problem could change the behavior of Marine predators, namely FADs' effect and eddy formation. Based on the study, predators often move around FAD. The FAD is regarded as local optimal and its effects as trapping at this point. Consideration of this long jumps during simulation prevents stagnation in local optimal. Consequently, FAD effects are shown in the following.

$$Prey_i = \begin{cases} prey_i + CF [x_{min}^r + R^r \otimes (x_{max}^r - x_{min}^r)] \otimes U^r & \text{if } r \leq FADs \\ Prey_i + [FADs(1-r) + r](Prey_{r1} - Prey_{r2}) & \text{if } r \geq FADs \end{cases} \quad (12)$$

where  $FAD = 0.2$  represents probability of FAD effects on the optimization technique.  $U^r$  denotes binary vector.  $r$  indicates a uniform random value within  $[0, 1]$ .  $x_{max}^r$  and  $x_{min}^r$  indicates upper and lower boundaries of containing dimension.  $r1$  and  $r2$  subscript denotes random index of prey matrixes.

Usually, marine predators have better memories and remember places where they have been prosperous in foraging. Afterward, upgrade prey and perform FAD effects, these matrixes are estimated for fitness to upgrade the Elite. Every solution in the existing iteration is compared to the preceding iteration to define its fitness. When the present solution is more fitted, it is replaced. Also, these algorithm enhances the solution quality with the lapse successful foraging. Fig. 2 displays the steps involved in MPO algorithm [21].



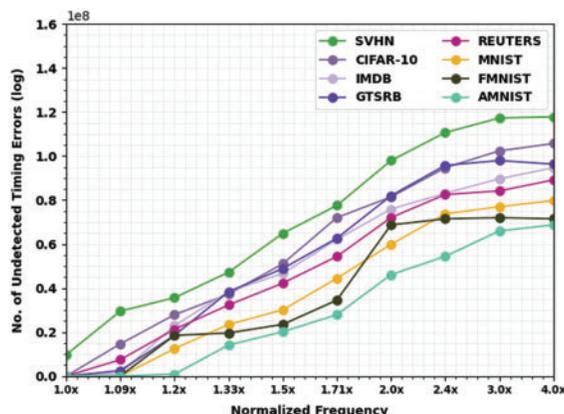
**Figure 2:** Steps involves in MPO algorithm

#### 4 Performance Validation

In this section, the experimental validation of the EEHPT-DNN model is tested using eight datasets. Table 1 and Fig. 3 provides NUTE examination of the EEHPT-DNN model with recent models on eight datasets. These results inferred that the EEHPT-DNN model has offered the least values of NUTE under all normalized frequencies. At the same time, it is noticed that the EEHPT-DNN model has obtained increasing NUTE values with an increase in normalized frequencies. i.e., the value of NUTE is minimal at 1.0x normalized frequency and reaches the maximum at 4.0x normalized frequency.

**Table 1:** NUTE analysis of EEHPT-DNN approach with eight distinct datasets

Normalized frequency	SVEN	CIFAR-10	IMDB	GTSRB	REUTERS	MOST	FEMINIST	AMNIST
1.0x	9694398	9652	6810	6898	9383	8991	7915	8593
1.09x	29550492	14658421	1421025	2524141	7488165	8568	8950	5565
1.2x	35617632	27895817	22931794	18519328	21277119	12452189	18519328	869467
1.33x	47200353	37272306	38375423	38375423	32308283	23483352	19622445	14106863
1.5x	64850215	51061261	46648795	48855028	42236330	30102050	23483352	20174003
1.71x	77536053	72020471	62092424	62643982	54370610	44442563	34514515	27895817
2.0x	97943705	81396960	75881378	81948518	72020471	59886191	68711122	46097237
2.4x	110629543	94634356	83051635	95737473	82500077	73675146	71468913	54370610
3.0x	117248242	102356171	89670333	97943705	84154751	76984495	72020471	65953331
4.0x	117799800	105665520	94634356	96289031	89118775	79742286	71468913	68711122



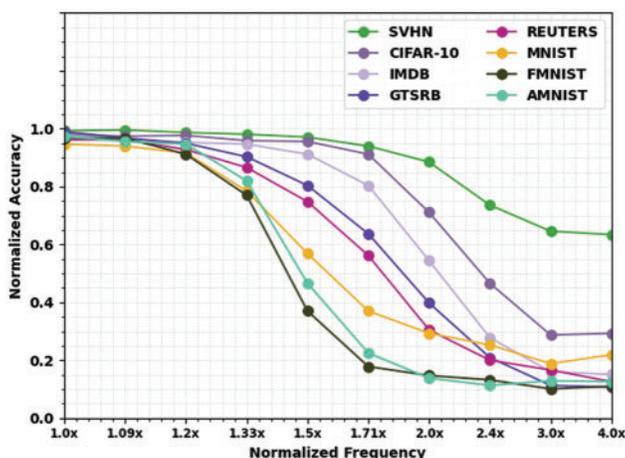
**Figure 3:** NUTE analysis of EEHPT-DNN approach with eight distinct datasets

Table 2 and Fig. 4 deliver NACC inspection of the EEHPT-DNN technique with recent approaches on eight datasets. These results denoted the EEHPT-DNN algorithm has the least values of NACC in all normalized frequencies. Meanwhile, it is noted that the EEHPT-DNN method has gained reduced NACC values with an increase in normalized frequencies. i.e., the value of NACC was maximum at 1.0x normalized frequency and will reach minimum at 4.0x normalized frequency.

Table 3 provides a comparative NACC examination of the EEHPT-DNN model with existing models under SVHN dataset. The experimental outcomes implied that the EEHPT-DNN model has attained maximum NACC values on SVHN dataset at all normalized frequencies. For instance, on normalized frequency of 1.0x, the EEHPT-DNN model has offered increased NACC of 0.998 whereas the IA(PRED), IA(MRFF), BE(PRED), IA(TED), and IA(B-TPU) models have obtained reduced NACC values of 0.974, 0.973, 0, 0.979, and 0.978 respectively. Similarly, on normalized frequency of 5.0x, the EEHPT-DNN model has attained maximum NACC of 0.990 whereas the IA(PRED), IA(MRFF), BE(PRED), IA(TED), and IA(B-TPU) models have reached minimal NACC values of 0.970, 0.312, 0, 0.965, 0.965, and 0.018 respectively.

**Table 2:** NACC analysis of EEHPT-DNN approach with eight distinct datasets

Normalized accuracy								
Normalized frequency	SVEN	CIFAR-10	IMDB	GTSRB	REUTERS	MOST	FEMINIST	AMNIST
1.0x	0.993	0.981	0.977	0.990	0.962	0.947	0.968	0.974
1.09x	0.996	0.974	0.962	0.968	0.962	0.940	0.968	0.956
1.2x	0.987	0.977	0.953	0.950	0.928	0.916	0.912	0.947
1.33x	0.981	0.959	0.947	0.903	0.866	0.785	0.770	0.820
1.5x	0.971	0.956	0.912	0.804	0.748	0.569	0.370	0.466
1.71x	0.940	0.912	0.804	0.637	0.562	0.370	0.178	0.225
2.0x	0.885	0.711	0.544	0.398	0.305	0.293	0.147	0.138
2.4x	0.736	0.466	0.278	0.209	0.200	0.253	0.132	0.113
3.0x	0.646	0.287	0.160	0.113	0.166	0.188	0.101	0.129
4.0x	0.634	0.293	0.151	0.107	0.126	0.219	0.110	0.126



**Figure 4:** NACC analysis of EEHPT-DNN approach with eight distinct datasets

**Table 3:** NACC analysis of EEHPT-DNN approach with existing algorithms under SVHN dataset

Normalized accuracy-SVHN						
Normalized frequency	EEHPT-DNN	IA (PRED)	IA (MRFF)	BE (PRED)	IA (TED)	IA (B-TPU)
1.0x	0.998	0.974	0.973	0.000	0.979	0.978
1.075x	0.997	0.980	0.973	0.000	0.978	0.970
1.15x	0.997	0.981	0.976	0.000	0.974	0.974
1.255x	0.997	0.973	0.971	0.000	0.978	0.973
1.36x	0.996	0.981	0.976	0.000	0.973	0.975
1.515x	0.995	0.980	0.968	0.000	0.979	0.968
1.67x	0.994	0.978	0.977	0.888	0.972	0.974
1.875x	0.992	0.970	0.971	0.908	0.973	0.971
2.14x	0.992	0.974	0.966	0.974	0.968	0.781
2.57x	0.992	0.970	0.852	0.972	0.969	0.612
3.0x	0.991	0.970	0.584	0.963	0.970	0.390
4.0x	0.991	0.972	0.394	0.961	0.970	0.191
5.0x	0.990	0.970	0.312	0.965	0.965	0.018

**Table 4** grants a brief NACC inspection of the EEHPT-DNN approach with existing methodologies under CIFAR-10 dataset. The experimental outcomes indicate EEHPT-DNN approach has achieved maximal NACC values on SVHN dataset at all normalized frequencies. For example, on normalized frequency of 1.0x, the EEHPT-DNN technique has presented increased NACC of 0.995, whereas the IA(PRED), IA(MRFF), BE(PRED), IA(TED), and IA(B-TPU) methodologies have attained reduced NACC values of 0.959, 0.956, 0.000, 0.962, and 0.961 correspondingly. Similarly, on normalized frequency of 5.0x, the EEHPT-DNN technique has obtained maximum NACC of 0.976

whereas the IA(PRED), IA(MRFF), BE(PRED), IA(TED), and IA(B-TPU) approaches have reached minimal NACC values of 0.959, 0.292, 0.910, 0.928, and 0.004 correspondingly.

**Table 4:** NACC analysis of EEHPT-DNN approach with existing algorithms under CIFAR-10 dataset

Normalized accuracy-CIFAR-10						
Normalized frequency	EEHPT-DNN	IA (PRED)	IA (MRFF)	BE (PRED)	IA (TED)	IA (B-TPU)
1.0x	0.995	0.959	0.956	0.000	0.962	0.961
1.075x	0.992	0.970	0.954	0.000	0.968	0.951
1.15x	0.996	0.969	0.966	0.000	0.957	0.964
1.255x	0.977	0.962	0.958	0.000	0.961	0.961
1.36x	0.997	0.962	0.959	0.000	0.960	0.960
1.515x	0.986	0.964	0.951	0.000	0.965	0.949
1.67x	0.975	0.965	0.961	0.876	0.962	0.964
1.875x	0.984	0.959	0.957	0.892	0.959	0.953
2.14x	0.996	0.958	0.953	0.958	0.949	0.765
2.57x	0.988	0.955	0.856	0.953	0.956	0.592
3.0x	0.983	0.959	0.573	0.943	0.957	0.379
4.0x	0.978	0.955	0.480	0.949	0.952	0.176
5.0x	0.976	0.959	0.292	0.910	0.928	0.004

**Table 5** offers a brief NACC check of the EEHPT-DNN approach with existing models under the IMDB dataset. The experimental outcomes implied that the EEHPT-DNN technique has attained maximum NACC values on SVHN dataset at all normalized frequencies. For example, on normalized frequency of 1.0x, the EEHPT-DNN approach has rendered increased NACC of 0.981 whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) methodologies have acquired reduced NACC values of 0.975, 0.965, 0.000, 0.977, and 0.959 correspondingly. Also, on normalized frequency of 5.0x, the EEHPT-DNN technique has reached maximum NACC of 0.977 whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) approaches have reached minimal NACC values of 0.968, 0.309, 0.905, 0.857, and 0.002 correspondingly.

**Table 6** offers the detailed NACC analysis of the EEHPT-DNN approach with existing techniques under GTSRB dataset. The experimental outcomes implied that the EEHPT-DNN approach has reached maximum NACC values on SVHN dataset at all normalized frequencies. For example, on normalized frequency of 1.0x, the EEHPT-DNN model has offered increased NACC of 0.978 whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) methodologies have acquired reduced NACC values of 0.971, 0.980, 0.000, 0.973, and 0.955 correspondingly. Similarly, on normalized frequency of 5.0x, the EEHPT-DNN technique has attained maximum NACC of 0.976 whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) approaches have reached minimal NACC values of 0.956, 0.287, 0.846, 0.890, and 0.008 correspondingly.

**Table 5:** NACC analysis of EEHPT-DNN approach with existing algorithms under IMDB dataset

Normalized accuracy-IMDB						
Normalized frequency	EEHPT-DNN	IA (PRED)	IA (MRFF)	BE (PRED)	IA (TED)	IA (B-TPU)
1.0x	0.981	0.975	0.965	0.000	0.977	0.959
1.075x	0.982	0.979	0.955	0.000	0.971	0.967
1.15x	0.986	0.983	0.955	0.000	0.969	0.962
1.255x	0.997	0.979	0.951	0.000	0.970	0.962
1.36x	0.996	0.982	0.975	0.000	0.972	0.968
1.515x	0.995	0.954	0.962	0.000	0.954	0.958
1.67x	0.998	0.953	0.978	0.880	0.958	0.970
1.875x	0.979	0.970	0.971	0.911	0.968	0.968
2.14x	0.991	0.953	0.963	0.951	0.956	0.779
2.57x	0.989	0.957	0.893	0.967	0.967	0.608
3.0x	0.977	0.954	0.675	0.956	0.954	0.378
4.0x	0.979	0.954	0.412	0.950	0.969	0.177
5.0x	0.977	0.968	0.309	0.905	0.857	0.002

**Table 6:** NACC analysis of EEHPT-DNN approach with existing algorithms under GTSRB dataset

Normalized accuracy-GTSRB						
Normalized frequency	EEHPT-DNN	IA (PRED)	IA (MRFF)	BE (PRED)	IA (TED)	IA (B-TPU)
1.0x	0.978	0.971	0.980	0.000	0.973	0.955
1.075x	0.984	0.971	0.959	0.000	0.970	0.958
1.15x	0.980	0.977	0.958	0.000	0.975	0.969
1.255x	0.995	0.953	0.979	0.000	0.980	0.958
1.36x	0.992	0.973	0.965	0.000	0.965	0.959
1.515x	0.982	0.973	0.966	0.000	0.968	0.958
1.67x	0.975	0.964	0.965	0.871	0.969	0.972
1.875x	0.991	0.973	0.959	0.906	0.961	0.961
2.14x	0.973	0.963	0.961	0.951	0.950	0.759
2.57x	0.979	0.967	0.892	0.962	0.960	0.590
3.0x	0.981	0.968	0.573	0.953	0.965	0.387
4.0x	0.981	0.963	0.328	0.955	0.963	0.178
5.0x	0.976	0.956	0.287	0.846	0.890	0.008

[Table 7](#) offers a comprehensive NACC analysis of the EEHPT-DNN technique with existing methodologies under REUTERS dataset. The experimental outcomes denoted the EEHPT-DNN

algorithm has achieved maximum NACC values on SVHN dataset at all normalized frequencies. For example, on normalized frequency of 1.0x, the EEHPT-DNN approach has rendered increased NACC of 0.978 whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) algorithms have reached reduced NACC values of 0.972, 0.975, 0.000, 0.962, and 0.967 correspondingly. Also, on normalized frequency of 5.0x, the EEHPT-DNN method has obtained maximum NACC of 0.994 whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) approaches have reached minimal NACC values of 0.945, 0.298, 0.945, 0.944, and 0.179 correspondingly.

**Table 7:** NACC analysis of EEHPT-DNN approach with existing algorithms under REUTERS dataset

Normalized accuracy-REUTERS						
Normalized frequency	EEHPT-DNN	IA (PRED)	IA (MRFF)	BE (PRED)	IA (TED)	IA (B-TPU)
1.0x	0.978	0.972	0.975	0.000	0.962	0.967
1.075x	0.987	0.951	0.957	0.000	0.958	0.946
1.15x	0.981	0.964	0.959	0.000	0.970	0.974
1.255x	0.983	0.964	0.961	0.000	0.963	0.947
1.36x	0.991	0.952	0.963	0.000	0.966	0.945
1.515x	0.975	0.951	0.966	0.000	0.966	0.963
1.67x	0.981	0.946	0.950	0.867	0.955	0.957
1.875x	0.981	0.962	0.967	0.904	0.966	0.955
2.14x	0.981	0.945	0.952	0.957	0.970	0.754
2.57x	0.975	0.972	0.830	0.951	0.964	0.598
3.0x	0.979	0.966	0.573	0.945	0.966	0.383
4.0x	0.971	0.956	0.374	0.959	0.944	0.179
5.0x	0.994	0.945	0.298	0.945	0.940	0.100

**Table 8** presents a comparative NACC examination of the EEHPT-DNN method with existing approaches under MINIST dataset. The experimental outcomes implied that the EEHPT-DNN approaches have reached maximum NACC values on SVHN dataset at all normalized frequencies. For example, on normalized frequency of 1.0x, the EEHPT-DNN method has rendered increased NACC of 0.999, whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) approaches have achieved reduced NACC values of 0.962, 0.969, 0.000, 0.960, and 0.952 correspondingly. Similarly, on normalized frequency of 5.0x, the EEHPT-DNN method has obtained maximum NACC of 0.979 whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) methods have reached minimal NACC values of 0.953, 0.294, 0.942, 0.968, and 0.216 correspondingly.

**Table 9** portrays a comparative NACC analysis of the EEHPT-DNN approach with existing approaches under FMINIST dataset. The experimental outcomes denote the EEHPT-DNN approach has obtained maximum NACC values on SVHN dataset at all normalized frequencies. For example, on normalized frequency of 1.0x, the EEHPT-DNN algorithm has presented increased NACC of 0.997 whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) approaches have obtained reduced NACC values of 0.972, 0.976, 0, 0.969, and 0.973 correspondingly. Also, on normalized frequency of 5.0x, the EEHPT-DNN approach has attained maximum NACC of 0.992

whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) algorithms have reached minimal NACC values of 0.953, 0.292, 0.952, 0.949, and 0.007 correspondingly.

**Table 8:** NACC analysis of EEHPT-DNN approach with existing algorithms under SVHN dataset

Normalized accuracy-MNIST						
Normalized frequency	EEHPT-DNN	IA (PRED)	IA (MRFF)	BE (PRED)	IA (TED)	IA (B-TPU)
1.0x	0.999	0.962	0.969	0.000	0.960	0.952
1.075x	0.985	0.972	0.969	0.000	0.973	0.966
1.15x	0.996	0.968	0.968	0.000	0.957	0.948
1.255x	0.998	0.955	0.953	0.000	0.958	0.960
1.36x	0.989	0.964	0.963	0.000	0.952	0.959
1.515x	0.976	0.963	0.967	0.000	0.969	0.950
1.67x	0.985	0.950	0.964	0.883	0.956	0.957
1.875x	0.989	0.943	0.952	0.909	0.952	0.968
2.14x	0.984	0.957	0.959	0.956	0.970	0.778
2.57x	0.985	0.968	0.837	0.961	0.961	0.694
3.0x	0.987	0.944	0.581	0.958	0.962	0.365
4.0x	0.974	0.943	0.481	0.957	0.948	0.195
5.0x	0.979	0.953	0.294	0.942	0.968	0.216

**Table 9:** NACC analysis of EEHPT-DNN approach with existing algorithms under FMNIST dataset

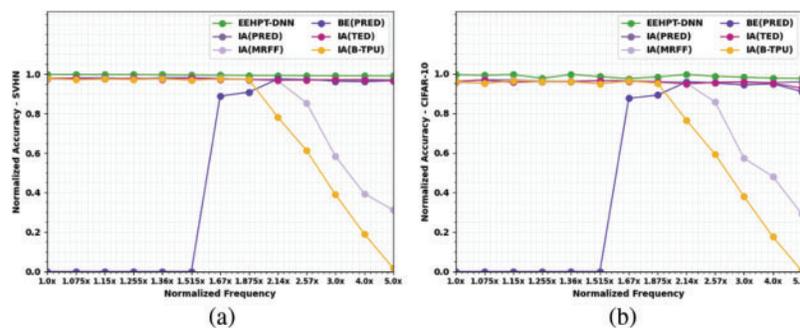
Normalized accuracy-FMNIST						
Normalized frequency	EEHPT-DNN	IA (PRED)	IA (MRFF)	BE (PRED)	IA (TED)	IA (B-TPU)
1.0x	0.997	0.972	0.976	0.000	0.969	0.973
1.075x	0.991	0.976	0.966	0.000	0.971	0.953
1.15x	0.983	0.968	0.952	0.000	0.951	0.962
1.255x	0.992	0.950	0.963	0.000	0.967	0.959
1.36x	0.978	0.964	0.954	0.000	0.969	0.959
1.515x	0.983	0.951	0.976	0.000	0.966	0.968
1.67x	0.982	0.953	0.955	0.861	0.951	0.968
1.875x	0.976	0.952	0.967	0.902	0.962	0.967
2.14x	0.995	0.949	0.950	0.953	0.962	0.778
2.57x	0.996	0.958	0.937	0.956	0.968	0.591
3.0x	0.979	0.959	0.564	0.957	0.959	0.469
4.0x	0.971	0.949	0.382	0.955	0.957	0.185
5.0x	0.992	0.953	0.292	0.952	0.949	0.007

Table 10 demonstrates a complete NACC analysis of the EEHPT-DNN algorithm with existing approaches under AMINIST dataset. The experimental outcomes indicated the EEHPT-DNN approach has acquired maximum NACC values on SVHN dataset at all normalized frequencies. For example, on normalized frequency of 1.0x, the EEHPT-DNN method has presented increased NACC of 0.981, whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) algorithms have attained reduced NACC values of 0.961, 0.951, 0.000, 0.952, and 0.971 correspondingly. Likewise, on normalized frequency of 5.0x, the EEHPT-DNN technique has obtained maximum NACC of 0.991 whereas the IA (PRED), IA (MRFF), BE (PRED), IA (TED), and IA (B-TPU) approaches have reached minimal NACC values of 0.960, 0.284, 0.940, 0.944, and 0.118 correspondingly.

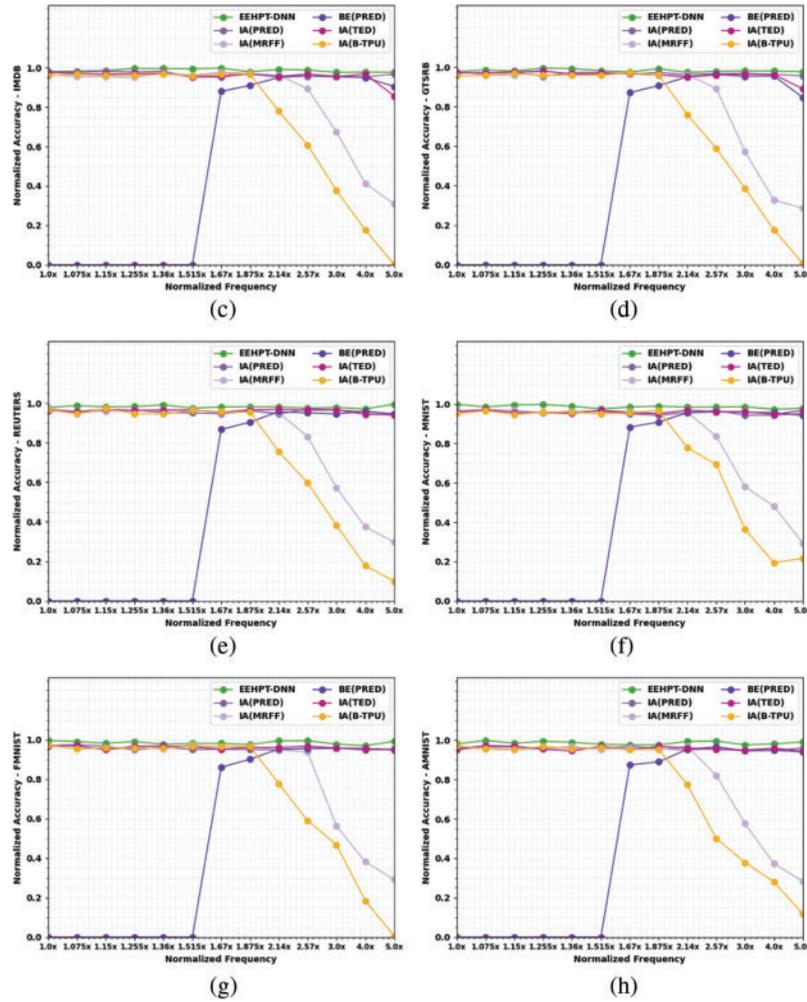
**Table 10:** NACC analysis of EEHPT-DNN approach with existing algorithms under AMINIST dataset

Normalized accuracy-AMINIST						
Normalized frequency	EEHPT-DNN	IA (PRED)	IA (MRFF)	BE (PRED)	IA (TED)	IA (B-TPU)
1.0x	0.981	0.961	0.951	0.000	0.952	0.971
1.075x	0.998	0.969	0.965	0.000	0.973	0.955
1.15x	0.984	0.966	0.952	0.000	0.969	0.952
1.255x	0.995	0.959	0.957	0.000	0.954	0.969
1.36x	0.989	0.962	0.968	0.000	0.945	0.957
1.515x	0.980	0.956	0.953	0.000	0.970	0.962
1.67x	0.976	0.969	0.961	0.874	0.951	0.954
1.875x	0.976	0.958	0.953	0.890	0.969	0.954
2.14x	0.993	0.952	0.968	0.953	0.961	0.773
2.57x	0.996	0.957	0.821	0.966	0.953	0.502
3.0x	0.977	0.946	0.577	0.946	0.950	0.379
4.0x	0.981	0.949	0.373	0.949	0.960	0.281
5.0x	0.991	0.960	0.284	0.940	0.944	0.118

An overall comparison study of the EEHPT-DNN model with recent models under eight datasets is given in Fig. 5. The figure implied that the EEHPT-DNN model has shown outperforming performance over other models on all the test datasets.



**Figure 5:** (Continued)



**Figure 5:** Comparative analysis of EEHPT-DNN approach (a) SVHN, (b) CIFAR-10, (c) IMDB, (d) GTSRB, (e) REUTERS, (f) MNIST, (g) FMNIST, (h) AMNIST

**5 Conclusion**

This paper proposes a novel developed EEHPT-DNN technique for Variation-Tolerant Near-Threshold Processor. The presented EEHPT-DNN model employs AI techniques to improve the energy efficiency of AI. The model emphasises the effects of embedded systems near the network’s edge. For the development of a variation-tolerant NT processor, the suggested model utilised the DSSAE paradigm. Due to the time-consuming and arduous nature of tuning hyperparameters through trial and error, the MPO approach is used to alter the hyperparameters associated with the DSSAE model. In order to demonstrate that the presented EEHPT-DNN model has a higher degree of functionality, a thorough simulation assessment has been conducted, and the results of this evaluation have been analysed with respect to a number of criteria. Compared to other DL models, the EEHPT-DNN model delivered superior outcomes, according to the findings of the comparison study.

**Funding Statement:** The authors received no specific funding for this study.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

- [1] W. Shan, X. Shang, X. Wan, H. Cai, C. Zhang *et al.*, “A wide-voltage-range half-path timing error-detection system with a 9-transistor transition-detector in 40-nm CMOS,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 6, pp. 2288–2297, 2019.
- [2] P. Pandey, P. Basu, K. Chakraborty and S. Roy, “GreenTPU: Improving timing error resilience of a near-threshold tensor processing unit,” in *Proc. of the 56th ACM/IEEE Design Automation Conf. (DAC)*, IEEE, Las Vegas, NV, USA, pp. 1–6, 2019.
- [3] S. Wang, C. Chen, X. Y. Xiang and J. Y. Meng, “A Variation-tolerant near-threshold processor with instruction-level error correction,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 7, pp. 1993–2006, 2017.
- [4] P. Pandey, P. Basu, K. Chakraborty and S. Roy, “GreenTPU: Predictive design paradigm for improving timing error resilience of a near-threshold tensor processing unit,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 7, pp. 1557–1566, 2019.
- [5] J. Koo, E. Park, D. Kim, J. Park, S. Ryu *et al.*, “Low-overhead, one-cycle timing-error detection and correction technique for flip-flop based pipelines,” *IEICE Electronics Express*, vol. 2019, pp. 16-20190180–1620190196, 2019.
- [6] R. G. Rizzo, V. Peluso, A. Calimera, J. Zhou and X. Liu, “Early bird sampling: A short-paths free error detection-correction strategy for data-driven VOS,” in *Proc. of the IFIP/IEEE Int. Conf. on Very Large Scale Integration (VLSI-SoC)*, IEEE, Abu Dhabi, United Arab Emirates, pp. 1–6, 2017.
- [7] V. Bhatnagar, M. K. Pandey and S. Pandey, “A variation tolerant nanoscale sram for low power wireless sensor nodes,” *Wireless Personal Communications*, vol. 124, pp. 3235–3251, 2022.
- [8] E. Mani, E. Abbasian, M. Gunasegeran and S. Sofimowloodi, “Design of high stability, low power and high speed 12 T SRAM cell in 32-nm CNTFET technology,” *AEU-International Journal of Electronics and Communications*, vol. 154, pp. 154308, 2022.
- [9] S. Wang, C. Chen, X. Xiang and J. Meng, “A Metastability-immune error-resilient flip-flop for near-threshold variation-tolerant designs,” *IEICE Electronics Express*, vol. 14, no. 11, pp. 20170353–20170353, 2017.
- [10] P. Sharma and B. P. Das, “Design and analysis of leakage-induced false error tolerant error detecting latch for sub/near-threshold applications,” *IEEE Transactions on Device and Materials Reliability*, vol. 20, no. 2, pp. 366–375, 2020.
- [11] Y. Lin, S. Zhang. and N. R. Shanbhag, “A rank decomposed statistical error compensation technique for robust convolutional neural networks in the near threshold voltage regime,” *Journal of Signal Processing Systems*, vol. 90, no. 10, pp. 1439–1451, 2018.
- [12] N. D. Gundi, P. Pandey, S. Roy and K. Chakraborty, “Implementing a timing error-resilient and energy-efficient near-threshold hardware accelerator for deep neural network inference,” *Journal of Low Power Electronics and Applications*, vol. 12, no. 2, pp. 1–32, 2022.
- [13] Y. Lin and J. R. Cavallaro, “Energy-efficient convolutional neural networks via statistical error compensated near threshold computing,” in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, Florence, Italy, pp. 1–5, 2018.
- [14] J. Wang, W. W. Liang, Y. H. Niu, L. Gao and W. G. Zhang, “Multi-dimensional optimization for approximate near-threshold computing,” *Frontiers of Information Technology & Electronic Engineering*, vol. 21, no. 10, pp. 1426–1441, 2020.
- [15] I. Tsiokanos, L. Mukhanov, D. S. Nikolopoulos and G. Karakonstantis, “Significance-driven data truncation for preventing timing failures,” *IEEE Transactions on Device and Materials Reliability*, vol. 19, no. 1, pp. 25–36, 2019.

- [16] T. T. Zhu, J. Y. Meng, X. Y. Xiang and X. L. Yan, "Error-resilient integrated clock gate for clock-tree power optimization on a wide voltage IoT processor," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1681–1693, 2019.
- [17] T. Zhu, X. Xiang, C. Chen and J. Meng, "SGERC: A self-gated timing error resilient cluster of sequential cells for wide-voltage processor," *IEICE Electronics Express*, pp. 20170214–20170218, 2017.
- [18] S. Akhila, S. Zubeda and R. Shashidhara, "DS2AN: Deep stacked sparse autoencoder for secure and fast authentication in HetNets," *Security and Privacy*, vol. 5, no. 3, pp. e208, 2022.
- [19] M. A. Soliman, H. M. Hasanien and A. Alkuhayli, "Marine predators algorithm for parameters identification of triple-diode photovoltaic models," *IEEE Access*, vol. 8, pp. 155832–155842, 2020.
- [20] M. Mustaqeem and S. Kwon, "A CNN assisted enhanced audio signal processing for speech emotion recognition," *Sensors*, vol. 20, no. 1, pp. 1–15, 2020.
- [21] Z. Tarek, M. Y. Shams, A. M. Elshewey, E. M. El-kenawy, A. Ibrahim *et al.*, "Wind power prediction based on machine learning and deep learning models," *Computers, Materials & Continua*, vol. 74, no. 1, pp. 715–732, 2023.