Context-Aware Backpropagation Using Neuron Activation History

Pranjala G. Kolapwar
CSE Department
SGGSIET
Nanded(MH), India
pgkolapwar@sggs.ac.in

Jaishri M. Waghmare
CSE Department
SGGSIET
Nanded(MH), India
jmwaghmare@sggs.ac.in

***Abstract*—Backpropagation is the cornerstone of neural network training, yet its reliance on instantaneous neuron activations makes gradient updates sensitive to noise and short-term fluctuations. This motivates the need for incorporating neuron-level temporal context into the learning process. This paper proposes context-aware backpropagation (CABP), a variant of the standard backpropagation algorithm that augments gradient computation with neuron activation history, enabling temporally contextualized updates while preserving the original forward pass, loss function, and optimization framework. The proposed method maintains a lightweight activation history for each neuron using an exponential moving average (EMA), capturing long-term activation behavior across training iterations. During the backward pass, gradients are adaptively modulated based on this activation history, reinforcing neurons that consistently contribute to learning while attenuating updates for neurons that contribute weakly or are noisy. As a result, CABP achieves lower training loss in fewer iterations, indicating faster convergence and improved optimization efficiency, particularly for large-scale and high-dimensional datasets. Importantly, CABP preserves the same asymptotic computational complexity as standard backpropagation and requires only minimal additional memory for storing activation history. Unlike adaptive moment-based optimizers, CABP avoids parameter-wise moment estimation while delivering stable, robust, and well-generalized learning dynamics.**

***Index Terms*—Gradient contextualization, backpropagation, neuron activation history, adaptive gradient scaling, neural network optimization, dynamic neural network**

I. Introduction

Backpropagation remains the fundamental learning mechanism for training neural networks; however, its gradient updates depend solely on instantaneous activations and error signals [1]. Such short-term dependency makes learning sensitive to noise, data imbalance, and unstable neuron participation—particularly in deep networks where some neurons become consistently active while others remain passive or underutilized. Passive neurons, especially under ReLU-like activations, often receive weak or zero gradients and thus contribute little to representation learning [2].

Recent advances have focused on reshaping gradient behavior to improve stability and generalization. These include sharpness-aware optimization [3], gradient centralization [4], adaptive clipping [5], sign-based optimizers [6], and lightweight second-order methods [7]. While effective, these approaches primarily operate on gradient statistics (norms, moments, and curvature) or optimizer dynamics, without explicitly considering neuron-level activation behavior over time.

Consequently, existing methods overlook a critical aspect of deep learning: the temporal consistency of individual neuron activations across training iterations. Prior studies have shown that sparse and intermittent neuron firing—common in ReLU-based deep networks—can lead to uneven gradient flow and biased feature learning, where only a subset of neurons dominates representation learning [2], [8]. Neurons that are sporadically active tend to introduce high-variance or noisy gradients, while consistently informative neurons are treated indistinguishably from transient contributors under standard backpropagation [9]. This absence of neuron-wise temporal context restricts backpropagation's ability to differentiate stable contributors from weak or noisy participants, a limitation that becomes more pronounced in deep and over-parameterized architectures [10].

This paper proposes CABP, an efficient backpropagation variant that incorporates historical neuron activation information to enable context-aware gradient modulation. By contextualizing gradient updates using historical neuron activity, CABP reinforces consistently useful neurons while suppressing noisy or weak contributors, thereby reducing gradient variance and improving learning stability—without altering the forward pass, loss function, or optimizer.

The remainder of this paper is organized as follows. Section II reviews recent advances in backpropagation and optimization techniques, highlighting their strengths and limitations with respect to gradient stability and neuron-level learning dynamics. Section III introduces the proposed CABP framework and details its integration into the backward pass. Section IV presents experimental results and comparative analyses against standard and recent optimization methods. Finally, Section V concludes the paper and outlines potential directions for future research.

II. Literature Review

Gradient-based learning through backpropagation remains the foundational mechanism for training deep neural networks due to its generality and computational efficiency [1]. Over the years, numerous extensions have been proposed to improve

convergence speed, stability, and generalization. Stochastic gradient–based methods introduce mini-batch sampling to improve scalability and robustness in large-scale learning settings [11], [12], while momentum-based techniques accumulate past gradient information to accelerate optimization along consistent descent directions and reduce oscillations [16]. Adaptive optimization algorithms such as AdaGrad [12], RMSProp [13], and Adam [14], [15] further enhance training by dynamically adjusting learning rates using first- and second-order gradient statistics.

To mitigate gradient instability in deep or recurrent architectures, techniques such as gradient clipping [26] and layer-wise adaptive scaling methods, including LARS and LAMB [27], have been introduced to control update magnitudes under large batch or high-curvature settings. Recent extensions, including per-example gradient clipping for differentially private learning and fast clipping frameworks, further refine gradient control at the sample level but remain focused on norm-based constraints rather than neuron behavior [24], [25]. In parallel, normalization and regularization strategies such as batch normalization [25], layer normalization, and dropout [28] stabilize training by reshaping activation distributions or reducing neuron co-adaptation. While effective, these approaches primarily influence the forward pass or operate on global, layer-wise, or sample-level gradient statistics, without explicitly modeling the temporal contribution of individual neurons during learning. Separately, studies on sparse activations and rectified networks have shown that neuron participation during training is highly uneven, with a small subset of neurons dominating representation learning while many remain intermittently active or persistently passive [2], [23]. This imbalance is particularly pronounced in deep and over-parameterized networks, where sporadically active neurons can inject noisy gradients yet are treated equivalently to consistently informative neurons under standard backpropagation. Existing optimization and regularization techniques do not explicitly capture this neuron-wise temporal behavior.

More recent efforts have explored structural modifications to backpropagation itself, including dense backpropagation for sparse Mixture-of-Experts routing and combined gradient-shaping strategies [28]- [30]. However, these methods are often architecture-specific or operate at coarse granularity, and none leverage per-neuron activation history as a guiding signal for gradient computation.

This gap highlights a fundamental limitation of current learning algorithms: the absence of neuron-level temporal context in the backward pass. While momentum-based methods accumulate historical gradients and adaptive optimizers rescale updates using gradient moments, they do not incorporate the historical activation behavior of individual neurons. Consequently, gradient updates remain sensitive to short-term fluctuations rather than being guided by long-term neuron usefulness.

To address this limitation, CABP introduces a lightweight and general mechanism that integrates neuron-level activation history directly into the backpropagation process. It maintains a per-neuron exponential moving average of activations and uses this historical context to modulate gradient updates, reinforcing consistently informative neurons while attenuating noisy or weak contributors. Unlike optimizer-level methods such as Adam or SAM, CABP operates within the backpropagation mechanism itself and does not alter the forward pass, loss function, or optimization objective. As a result, CABP provides a principled, interpretable, and complementary enhancement to existing gradient-based learning techniques, particularly suited for deep supervised networks with sparse and imbalanced neuron activations.

III. Proposed Method

This section introduces Context-Aware Backpropagation (CABP), a variant to the standard backpropagation that incorporates neuron activation history into the gradient update process. By maintaining an EMA of neuron activations, CABP adaptively scales neuron-wise gradients during the backward pass, enabling more stable and context-aware learning without altering the forward computation or the optimizer. The basic working of CABP is shown in fig. 1.

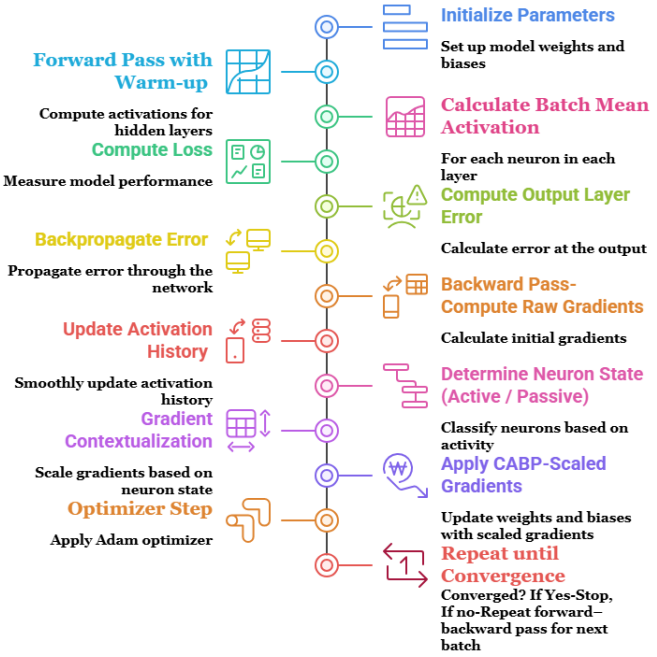

Fig. 1. CABP training workflow with gradient contextualization based on neuron activation history

Let $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ denote a supervised dataset of N samples, where $\mathbf{x}_i \in \mathbb{R}^d$ is the input vector and $\mathbf{y}_i \in \mathbb{R}^C$ is the target vector. Consider an L-layer feedforward neural network with layer index set $\mathcal{I} = \{1, 2, \ldots, L\}$ and parameters, $\boldsymbol{\theta} = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l \in \mathcal{I}}$, where $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ and $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ denote the weight matrix and bias vector of layer l, respectively. Here $n_l \in \mathbb{N}$ denotes the number of neurons in layer l, i.e., layer l contains n_l neurons. For an input sample

$\mathbf{x}_i$, the pre-activation and activation vectors at layer l are given by:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \quad \forall\, l \in \mathcal{I}, \tag{1}$$

$$\mathbf{a}^{(l)} = \phi\Big(\mathbf{z}^{(l)}\Big), \quad \forall\, l \in \mathcal{I}, \tag{2}$$

where $\mathbf{a}^{(0)} \equiv \mathbf{x}_i$ represents the activation of the input layer ($l = 0$), which contains no learnable parameters and directly passes the input features to the network. This definition enables the forward propagation equations to be written in a unified form for all layers, while $\phi(\cdot)$ denotes a nonlinear activation function (e.g., ReLU [2]). During training, each layer l maintains an activation history vector, $\mathbf{H}^{(l)} \in \mathbb{R}^{n_l}$, which summarizes the long-term activation behavior of individual neurons and is subsequently used to contextualize gradient updates in the backward pass.

The following subsections describe CABP in detail, including forward propagation, activation history modeling, gradient contextualization, and optimization.

A. Forward Pass and Warm-up Phase

The forward pass in CABP is identical to that of standard backpropagation. For a mini-batch $\mathcal{B} \subset \mathcal{D}$ of size B, activations are computed layer-wise for all layers $l = 1, \ldots, L$ and for all samples in $\mathcal{B}$. The network output at the final layer, $\mathbf{a}^{(L)}$, is then used to evaluate the mini-batch loss as:

$$\mathcal{L}_{\mathcal{B}} = \frac{1}{B} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{B}} \ell(\mathbf{a}_i^{(L)}, \mathbf{y}_i), \tag{3}$$

where $\ell(\cdot, \cdot)$ denotes a suitable loss function, such as mean squared error for regression tasks or cross-entropy loss for classification problems. The corresponding output layer error is computed as:

$$\boldsymbol{\delta}^{(L)} = \frac{\partial \mathcal{L}_{\mathcal{B}}}{\partial \mathbf{z}^{(L)}}, \tag{4}$$

which serves as the starting point for the backward propagation of errors. To ensure stable neuron activity statistics, CABP uses a warm-up phase of T_w iterations during which gradient scaling is disabled. During this phase, neuron activations are accumulated to initialize the activation history, but all gradients remain unscaled. Here, T_w denotes the number of initial training iterations (mini-batches) dedicated to stabilizing the exponential moving averages of neuron activations prior to the application of CABP scaling, with training steps indexed as $t = 1, 2, \ldots, T_w$. This warm-up phase allows the activation history to reflect stable neuron behavior rather than noisy early-training responses, thereby improving optimization stability and avoiding premature amplification or suppression of gradients.

B. Activation History Modeling

Following the forward pass, CABP models neuron activity by maintaining a neuron-wise exponential moving average of activations. At iteration t, given a mini-batch $\mathcal{B}$, the mean absolute activation of neuron j in layer l is computed as:

$$\bar{a}_j^{(l)}(t) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \Big| a_{i,j}^{(l)}(t) \Big|, \quad \forall\, l = 1, \ldots, L, \tag{5}$$

where $a_{i,j}^{(l)}(t)$ denotes the activation of neuron j in layer l for sample i at iteration t. The batch-averaged quantity $\bar{a}_j^{(l)}(t)$ provides a stable estimate of neuron activity used to update the activation history. The activation history is then updated using an exponential moving average:

$$H_j^{(l)}(t) = \beta H_j^{(l)}(t-1) + (1-\beta)\, \bar{a}_j^{(l)}(t), \quad \forall\, l = 1, \ldots, L, \tag{6}$$

where $\beta \in (0, 1)$ controls the temporal memory of past activations. To mitigate initialization bias during early training iterations, a bias-corrected activation history is computed as

$$\hat{H}_j^{(l)}(t) = \frac{H_j^{(l)}(t)}{1 - \beta^t}, \quad \forall\, l = 1, \ldots, L. \tag{7}$$

This activation history provides a stable estimate of long-term neuron activity and is subsequently used to contextualize neuron-wise gradient updates during backpropagation.

C. Gradient Contextualization and Neuron State Identification

Building on the activation history modeling described in the previous subsection, CABP leverages neuron-wise historical activity to both identify neuron states and contextualize gradient updates during backpropagation.

Fig. 2 presents a network-level view of CABP, illustrating how activation histories accumulated across layers are used to modulate gradient flow during the backward pass without altering the forward computation or loss function. After the warm-up phase ($t \geq T_w$), the bias-corrected activation history $\hat{H}_j^{(l)}(t)$ of each neuron j in layer l is utilized to guide neuron-specific learning dynamics.

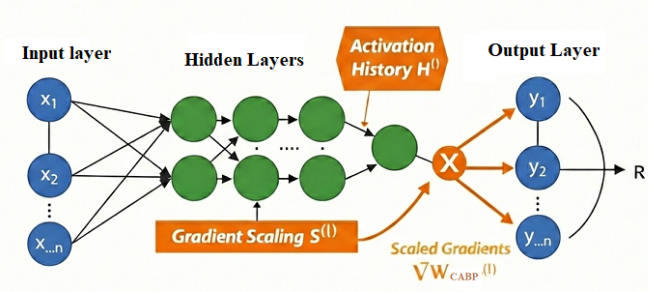

Fig. 2. Network-level illustration of CABP showing how neuron activation history is used to scale layer-wise gradients during backpropagation in a fully connected neural network

a) Active and Passive Neuron Identification.: Neurons are classified as either *active* or *passive* based on their sustained activity patterns. Specifically, for each neuron j in layer l ($j = 1, \ldots, N_l$; $l = 1, \ldots, L$) at iteration t, the state is defined as:

$$\text{State}_j^{(l)}(t) = \begin{cases} \text{Active}, & \text{if } \hat{H}_j^{(l)}(t) \geq \tau_H \\ & \quad \wedge\, |a_j^{(l)}(t)| \geq \tau_A \\ & \quad \wedge\, \|\mathbf{W}_{j,:}^{(l)}\| \geq \tau_W\ , \\ \text{Passive}, & \text{otherwise}, \end{cases} \tag{8}$$

where the thresholds τ_H, τ_A, and τ_W are typically set based on the mean and standard deviation of historical activations,

current activations, and weight norms, respectively. In practice, these thresholds are implemented as fixed empirical values. In this study, τ_H, τ_A, and τ_W are set to a common value of 0.05 for simplicity while maintaining effective neuron state separation.

b) Gradient Contextualization.: Following neuron state identification, CABP computes a neuron-specific gradient scaling factor:

$$S_j^{(l)}(t) = \begin{cases} 1 + \alpha\, \hat{H}_j^{(l)}(t), & \text{if Active,} \\ \beta, & \text{if Passive,} \end{cases} \quad \forall l, \forall j, \tag{9}$$

where $\alpha > 0$ amplifies updates for consistently influential neurons and $0 < \beta < 1$ attenuates gradients associated with weakly active or noisy neurons.

Let θ denote a generic trainable parameter associated with neuron j in layer l. The standard backpropagation gradient $\nabla\theta(t) = \partial\mathcal{L}(t)/\partial\theta$, computed via the chain rule, is contextually modulated as:

$$\nabla\theta_{\text{CABP}}(t) = S_j^{(l)}(t)\, \nabla\theta(t). \tag{10}$$

In practice, this contextual scaling is applied explicitly to the weight vectors and bias terms of each neuron:

$$\nabla \mathbf{W}_{j,:}^{(l)}(t) \leftarrow S_j^{(l)}(t)\, \nabla \mathbf{W}_{j,:}^{(l)}(t), \tag{11}$$

$$\nabla b_j^{(l)}(t) \leftarrow S_j^{(l)}(t)\, \nabla b_j^{(l)}(t). \tag{12}$$

The resulting CABP-scaled gradients are subsequently passed to the optimizer, together with the effective learning rate $\eta_{\text{effective}}(t)$.

Fig. 3 provides a neuron-level view of this mechanism, highlighting how activation history directly influences gradient scaling for individual neurons.

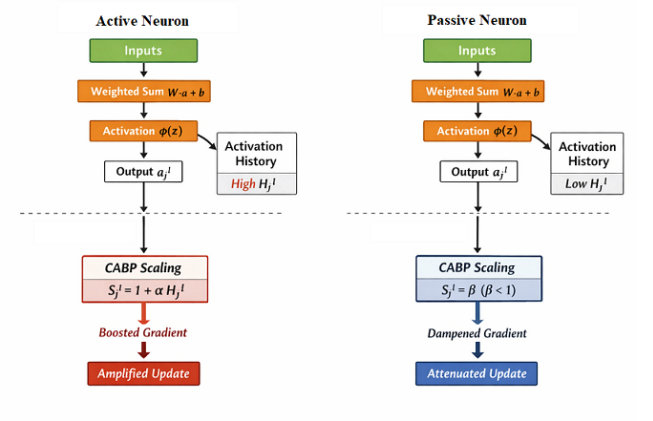

Fig. 3. Neuron-level view of CABP where neurons with high activation history receive amplified gradient updates, while low-activity neurons undergo dampened updates

D. Hyperparameter Settings

CABP introduces four key hyperparameters that jointly control how historical neuron activity influences learning dynamics: the EMA coefficient β_j, the gradient scaling strength $\lambda(t)$, the learning rate η, and the warm-up duration T_w.

- The EMA coefficient β_j, which determines how much past activation history is retained for each neuron, is constrained to lie between 0 and 1, $0 < \beta_j < 1$.

$$\beta_j = \begin{cases} \beta_{\text{high}}, & \text{if activation variability is high (noisy neuron),} \\ \beta_{\text{low}}, & \text{if activation variability is low (stable neuron),} \end{cases} \tag{13}$$

- The warm-up duration T_w specifies the initial iterations used to gather reliable activation statistics.

$$T_w = \begin{cases} k \cdot \frac{N}{B}, & \text{for large or high-variance datasets,} \\ \frac{N}{B}, & \text{otherwise,} \end{cases} \tag{14}$$

  where N is the total number of samples, B is the batch size, and $k \in [1, 5]$ scales the warm-up for high-variance datasets.
- The scaling factor $\lambda(t)$ modulates the impact of activation history on gradients, starting minimal during the warm-up phase and gradually increasing as neuron activity stabilizes.

$$\lambda(t) = \begin{cases} 0, & t < T_w \quad \text{(warm-up)} \\ \frac{t - T_w}{\tau}, & t \geq T_w \quad \text{(gradual ramp-up)} \end{cases} \tag{15}$$

  where τ is the ramp-up duration and t is the current training iteration. The scaling factor $\lambda(t)$ governs the gradual introduction of gradient modulation after the warm-up phase, ensuring stable early training while progressively enabling CABP as neuron activation statistics become reliable.
- The learning rate η is adjusted dynamically to account for gradient amplification introduced by CABP [15].

$$\eta_{\text{effective}}(t) = \frac{\eta_0}{\max(1, \max_j S_j^{(l)}(t))}, \tag{16}$$

  where $S_j^{(l)}(t)$ is the scaling factor for neuron j in layer l. Large neuron-wise scaling factors reduce the effective learning rate proportionally, maintaining stable optimization and compatibility with standard optimizers.

Together, these hyperparameters enable CABP to adaptively regulate memory, scaling strength, and learning stability in a data-driven manner, ensuring robust performance across diverse network architectures and training conditions.

E. Optimization with Adam

After gradient contextualization, CABP-scaled gradients are fed into the Adam optimizer. For each parameter θ, the updates are computed as [14]- [16]:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \nabla\theta_{\text{CABP}}(t), \tag{17}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \big(\nabla\theta_{\text{CABP}}(t)\big)^2, \tag{18}$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t}, \quad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t}, \tag{19}$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}, \tag{20}$$

where $\mathbf{m}_t$ and $\mathbf{v}_t$ are the first- and second-moment estimates, β_1 and β_2 are the decay rates, $\nabla\theta_{\text{CABP}}(t)$ is the CABP-scaled gradient, and ϵ ensures numerical stability.

By modifying gradients only after backpropagation, CABP integrates seamlessly with existing optimizers and training pipelines without altering the network architecture, forward pass, or loss function. Conceptually, CABP acts as a form of temporal regularization on gradient updates by leveraging neuron activation history through an exponential moving average. This enables differentiation between consistently informative and sporadically active neurons, attenuating high-variance updates from noisy neurons while reinforcing stable feature contributors. As a result, CABP smooths the optimization process, improving convergence stability and reducing sensitivity to mini-batch fluctuations. For clarity and ease of implementation, the complete training procedure of the proposed Context-Aware Backpropagation (CABP) method is summarized in Algorithm 1.

Algorithm 1 Context-Aware Backpropagation (CABP)

Initialize network parameters θ
Initialize activation history $H^{(l)} = 0,\ \forall l$
for each training iteration t **do**
  Perform forward pass and compute loss $\mathcal{L}$
  Compute gradients $\nabla_\theta \mathcal{L}$ using standard backpropagation
  Update activation history (EMA):

$$H^{(l)} \leftarrow \beta H^{(l)} + (1-\beta)\, a^{(l)}$$

  if $t > T_w$ **then**
    Determine neuron states (active/passive) based on threshold τ
    Compute scaling factors $S_j^{(l)}(t)$
    Scale gradients:

$$\nabla_\theta^{CABP} \leftarrow S_j^{(l)}(t) \cdot \nabla_\theta$$

  end if
  Update parameters using optimizer (e.g., Adam):

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta^{CABP}$$

end for
return θ =0

IV. Performance and Result Analysis

This section evaluates the proposed CABP method on benchmark datasets, including Iris, Wine, Breast Cancer, and Digits. CABP is compared with standard backpropagation under identical network architectures, training configurations, and optimization settings. Performance is assessed in terms of classification accuracy, generalization, convergence behavior, and hyperparameter sensitivity. All datasets are represented as feature matrices $\mathbf{X} \in \mathbb{R}^{N \times d}$ with corresponding label vectors $\mathbf{y} \in \mathbb{R}^N$, and evaluation is performed using both k-fold cross-validation and hold-out testing splits. The selected datasets are widely used benchmarks that enable controlled analysis of optimization dynamics and gradient behavior. This study focuses on validating the fundamental effectiveness of CABP on such datasets before extending it to more complex, large-scale scenarios. The EMA decay parameter β was fixed at 0.99 across all experiments, with consistent performance observed across datasets.

A. Case Study: CABP on Iris Dataset

CABP was applied to the Iris dataset to evaluate its efficiency in minimizing training error while reducing computational complexity. The Iris dataset consists of 150 samples with 4 features each, classified into 3 classes. Data was standardized using z-score normalization and split into 80% training and 20% testing sets. The model architecture consists of a three-layer Multi-Layer Perceptron (MLP) with 4 input neurons, 16 neurons in the first hidden layer, 8 neurons in the second hidden layer, and 3 output neurons corresponding to the three classes. ReLU activation was applied in the hidden layers.

Given the sensitivity of CABP to hyperparameters, we conducted an analysis to identify optimal values that enhance training stability and generalization. Hyperparameter analysis is crucial for understanding and optimizing CABP performance, as the choice of scaling factors, warm-up periods, and EMA parameters directly influences gradient adaptation. Figure 4 illustrates test accuracy across different combinations of λ (scaling factor), T_w (warm-up epochs), and β (EMA factor). The results indicate that CABP achieves optimal performance at $\beta = 0.99$, $\lambda = 0.5$, and $T_w = 5$, while values above or below these thresholds lead to reduced accuracy. This emphasizes the importance of carefully tuning these hyperparameters to maximize the benefits of gradient contextualization. For this case study, CABP was applied to

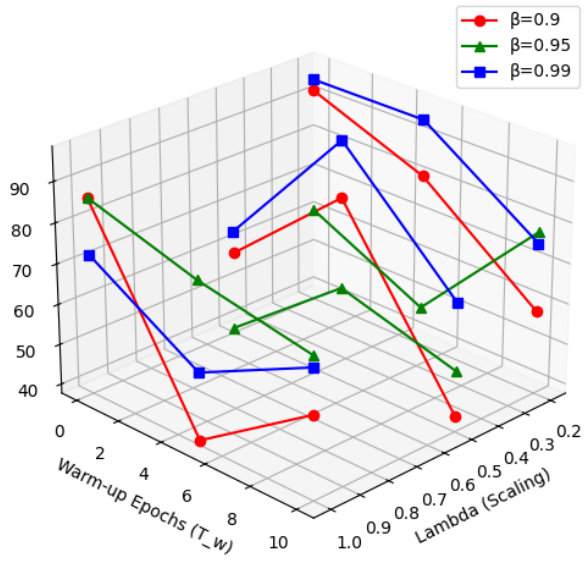

Fig. 4. CABP test accuracy on the Iris dataset across λ, T_w, and β, showing optimal performance at $\beta = 0.99$, $\lambda = 0.5$, $T_w = 5$

the MLP with the following configuration: $\alpha = 0.1$ for EMA smoothing of neuron activation history, $\tau = 0.05$ as the active neuron threshold, and $\epsilon = 10^{-6}$ for numerical stability. A

warm-up of $T_w = 5$ epochs was used before applying gradient scaling. The network was trained with a learning rate of 0.01 using the Adam optimizer for 15 epochs with a batch size of 16. This setup allowed CABP to adaptively scale gradients based on neuron activation history during training, enhancing both convergence and stability.

Table I summarizes the training loss, percentage of active neurons in the hidden layers, and test accuracy across all 15 epochs. CABP demonstrates rapid convergence, achieving high test accuracy while only a subset of neurons is active during backpropagation. This selective gradient update reduces computational complexity without compromising performance, highlighting CABP's efficiency. In the warm-up phase (epochs 1–5), CABP is disabled and only standard Adam updates are applied, resulting in a gradual decrease in loss with minor fluctuations due to mini-batch variance. From epoch 6 onward, CABP is enabled, scaling gradient updates based on neuron activation history, which leads to lower and more stable loss values and accelerates convergence. The stabilization of active neuron percentages in FC1 and FC2 after a few epochs is due to the low feature dimensionality and limited complexity of the Iris dataset; for larger, high-featured datasets, CABP exhibits more pronounced and sustained activation dynamics across layers, leading to greater performance gains.

TABLE I
CASE STUDY OF CABP ON THE IRIS DATASET USING AN MLP ARCHITECTURE, ILLUSTRATING CONVERGENCE BEHAVIOR AND NEURON ACTIVATION DYNAMICS

Epoch	Training Loss	Active Neurons FC1 (%)	Active Neurons FC2 (%)	Test Accuracy (%)
1	1.0366	7.88	1.63	70.00
2	0.8078	11.75	5.75	83.33
3	0.5259	13.00	6.63	86.67
4	0.3663	13.63	7.00	93.33
5	0.2994	14.00	7.00	93.33
6	0.2265	14.88	7.00	100.00
7	0.1795	15.00	7.00	96.67
8	0.1322	15.00	7.00	100.00
9	0.1101	15.00	7.00	96.67
10	0.0995	15.00	7.00	96.67
11	0.0795	15.00	7.00	96.67
12	0.0788	15.00	7.00	96.67
13	0.0704	15.00	7.00	96.67
14	0.0727	15.00	7.00	100.00
15	0.0677	15.00	7.00	96.67

B. Cross-Validation Analysis

To further assess CABP's robustness, five-fold cross-validation was performed on four datasets for all four methods: Stochastic Gradient Descent (SGD) [11], Root Mean Square Propagation (RMSProp) [13], Adaptive Moment Estimation (Adam) [14], and CABP. Table II presents the mean accuracy across the datasets for different dataset splits, including 90–10 and 70–30 train-test splits, as well as testing on unseen data. CABP (highlighted in **bold**) consistently achieves the highest mean accuracy across all dataset splits, demonstrating superior generalization, stability, and robustness compared to the standard optimizers. The results indicate that CABP effectively improves predictive performance while maintaining consistent performance across different data partitioning schemes.

TABLE II
AVERAGE ACCURACY (%) ACROSS FOUR DATASETS FOR DIFFERENT DATASET SPLITS

Method	90–10 Split	70–30 Split	Unseen Data	Average
SGD	91.5	89.6	89.0	90.0
RMSProp	94.7	93.8	92.8	93.8
Adam	95.2	94.5	93.9	94.5
CABP	**97.7**	**96.7**	**95.9**	**96.8**

Fig. 5 presents the performance of CABP on the Iris dataset. The training loss decreases rapidly, indicating fast convergence and efficient learning. The percentage of active neurons in FC1 and FC2 shows that only a subset of neurons contributes to gradient updates, reducing computational complexity without affecting performance. Test accuracy remains consistently high across epochs, demonstrating CABP's ability to maintain robust generalization while minimizing error with fewer active neurons. This visualization highlights the efficiency and effectiveness of CABP in accelerating training and optimizing network utilization.

C. Comparative Analysis

To assess the effectiveness of the proposed CABP approach, its performance is compared with conventional optimization methods such as SGD, RMSProp, and Adam across different datasets.

Table III compares the performance of CABP with standard optimizers across four benchmark datasets. CABP consistently achieves the highest training and testing accuracy on all datasets while also yielding the lowest loss, indicating improved convergence behavior. The performance gain of CABP is modest on simpler datasets such as Iris, reflecting stable learning without overfitting, but increases progressively with dataset complexity. Notably, CABP records the largest improvement on the high-dimensional Digits dataset, achieving a 7.8% gain in test accuracy, demonstrating its effectiveness in handling complex feature spaces. These results confirm that CABP enhances generalization and scalability compared to conventional optimization methods.

CABP, a variant of standard backpropagation, achieves efficient error minimization in fewer training iterations by selectively updating actively contributing neurons, thereby reducing computational overhead. By incorporating neuron activation history into gradient scaling, CABP synergizes with adaptive optimizers such as Adam, smoothing updates near local minima and mitigating oscillations, which enhances convergence stability. Empirical results across multiple benchmark datasets demonstrate that CABP consistently improves convergence speed, predictive accuracy, and generalization performance compared to conventional methods. Experiments were conducted over multiple runs with consistent performance trends observed, indicating the robustness of the proposed approach. These findings establish CABP as a lightweight and effective training paradigm that integrates seamlessly with existing

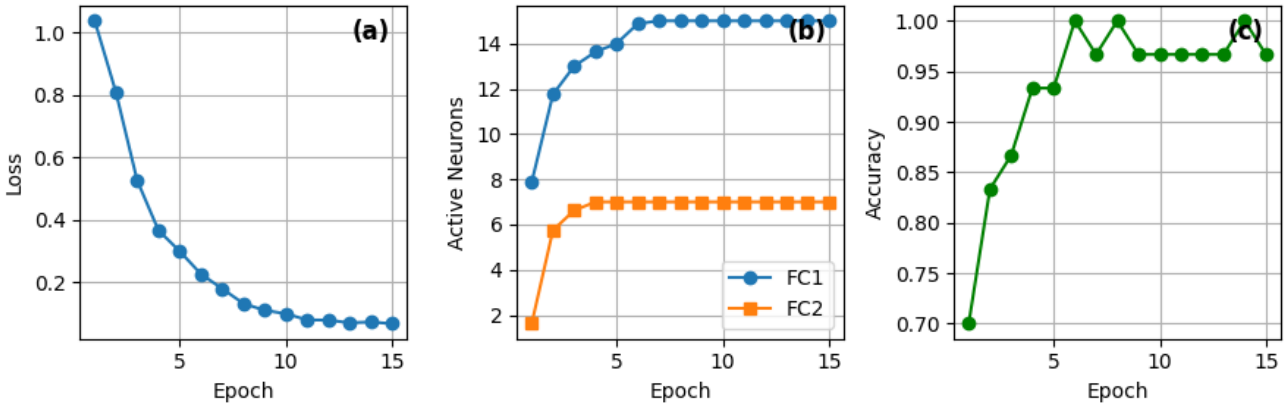

Fig. 5. CABP performance on the Iris dataset: (a) Training loss over epochs, (b) Percentage of active neurons in hidden layers FC1 and FC2, (c) Test accuracy over epochs

TABLE III
COMPARISON ACROSS OPTIMIZERS WITH TRAINING AND TESTING ACCURACY

Dataset	Feature	Method	Train Acc. (%)	Test Acc. (%)	Loss	Gain (%)
Iris	4	SGD	95.10	95.33	0.22	0.00
		RMSprop	95.40	95.67	0.20	+0.34
		Adam	95.30	95.50	0.21	+0.17
		CABP	**96.20**	**96.67**	**0.18**	**+1.34**
Wine	13	SGD	95.10	96.20	0.28	0.00
		RMSprop	96.05	97.10	0.24	+0.90
		Adam	96.30	98.05	0.26	+1.85
		CABP	**99.20**	**100.00**	**0.19**	**+3.80**
Breast Cancer	30	SGD	92.40	93.10	0.26	0.00
		RMSprop	93.60	94.20	0.23	+1.10
		Adam	94.50	95.30	0.24	+2.20
		CABP	**96.80**	**97.40**	**0.17**	**+4.30**
Digits	64	SGD	85.20	85.80	0.36	0.00
		RMSprop	88.40	89.10	0.30	+3.30
		Adam	89.10	90.00	0.28	+4.20
		CABP	**93.20**	**93.60**	**0.21**	**+7.80**

neural network architectures without modifying forward computations or optimization frameworks. Although the current evaluation is limited to small-scale datasets, the method is architecture-agnostic and can be extended to deep and high-dimensional tasks, including convolutional neural networks and large-scale image classification. Preliminary observations indicate that both the warm-up phase and neuron-state-based gradient scaling contribute to performance improvements; a detailed ablation study isolating these components is left for future work. The evaluation is limited to small-scale datasets, and hyperparameter sensitivity and computational overhead in deep architectures remain to be explored. Future work will address large-scale settings with comprehensive ablation and statistical analysis.

V. CONCLUSIONS

This paper presented a backpropagation variant, CABP, a variant of the standard backpropagation algorithm that incorporates neuron activation history to achieve context-aware gradient updates. By maintaining a lightweight exponential moving average of neuron activations, CABP selectively amplifies gradients for consistently informative neurons while attenuating updates for weak or noisy contributors. This mechanism reduces gradient variance, smooths updates near local minima and maxima, and mitigates oscillations, thereby enhancing convergence stability and overall learning efficiency. Extensive experiments on benchmark datasets—including Iris, Wine, Breast Cancer, and Digits—demonstrate that CABP consistently accelerates convergence, improves predictive accuracy, and enhances generalization on unseen data compared to conventional optimizers such as SGD, RMSProp, and Adam. Importantly, CABP achieves these improvements while maintaining the same asymptotic computational complexity as standard backpropagation and seamlessly integrating with existing network architectures and optimization frameworks without altering the forward pass or loss function. These results establish CABP as a lightweight yet robust backpropagation variant, offering superior stability, efficiency, and scalability for deep neural network training. Future work will extend CABP to large-scale convolutional and recurrent networks and evaluate it on complex, high-dimensional benchmarks (e.g., image classification tasks) to validate scalability and generalization, while also exploring adaptive neuron-state thresholding strategies to further enhance learning dynamics.

ACKNOWLEDGMENT

This research was carried out independently and did not receive any financial support from public, commercial, or not-for-profit funding agencies.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams,"Learning representations by back-propagating errors,"*Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[2] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. AISTATS*, 2011, pp. 315–323.
[3] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," in *Proc. ICLR*, 2021.

[4] H. Yong, J. Huang, X. Hua, and L. Zhang, "Gradient centralization: A new optimization technique for deep neural networks," in *Proc. ECCV*, 2020.

[5] A. Brock, S. De, S. L. Smith, and K. Simonyan, "High-performance large-scale image recognition without normalization," in *Proc. ICML*, 2021.

[6] X. Chen, C. Wang, and J. Liang, "Symbolic discovery of optimization algorithms," in *Proc. NeurIPS*, 2022.

[7] G. Liu, T. Li, J. Chen, and Z. Wang, "Sophia: A scalable stochastic second-order optimizer for language model pre-training," in *Proc. ICLR*, 2024.

[8] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. ECCV*, 2014, pp. 818–833.

[9] S. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *arXiv preprint arXiv:1703.00810*, 2017.

[10] B. Neyshabur, R. Tomioka, and N. Srebro, "In search of the real inductive bias: On the role of implicit regularization in deep learning," in *Proc. ICLR*, 2015.

[11] Z. Kirori, "Performance analysis of stochastic gradient descent-based algorithms for time series sequence modeling," M.S. thesis, 2019.

[12] E. M. Dogo, A. F. Salami, N. N. Afolayan, and O. O. Afolayan, "A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks," in *Proc. Int. Conf. Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, IEEE, 2018, pp. 92–99.

[13] F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu, "A sufficient condition for convergence of Adam and RMSProp," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11127–11135.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] Z. Zhang, "Improved Adam optimizer for deep neural networks," in *Proc. IEEE/ACM 26th Int. Symp. Quality of Service (IWQoS)*, IEEE, 2018, pp. 1–6.

[16] F. Huang, S. Gao, J. Pei, and S. Huang, "Momentum-based policy gradient methods," in *Proc. Int. Conf. Machine Learning (ICML)*, PMLR, 2020, pp. 3722–3731.

[17] F. Huang, S. Gao, J. Pei, and S. Huang, "Momentum-based policy gradient methods," in *Proc. Int. Conf. Machine Learning (ICML)*, PMLR, 2020, pp. 3722–3731.

[18] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Communications*, vol. 7, no. 13276, 2016.

[19] D.-H. Lee, "Difference target propagation," in *Proc. European Conf. Computer Vision (ECCV)*, Springer, 2016, pp. 498–515.

[20] Y. You, I. Gitman, and B. Ginsburg, "Large batch training of convolutional networks," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2018.

[21] L. Li, K. Dey, and J. He, "Normalized gradient descent," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[23] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[25] I. Loshchilov and F. Hutter,"Decoupled weight decay regularization," in *Proc. ICLR*, 2019.

[26] J. Lee and D. Kifer, "Scaling up differentially private deep learning with fast per-example gradient clipping," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 4, pp. 1–21, 2021.

[27] W. Kong and A. Munoz Medina, "A unified fast gradient clipping framework for DP-SGD," in *Advances in Neural Information Processing Systems*, vol. 36, pp. 52401–52412, 2023.

[28] Z. Liu et al.,"Layer-wise natural gradient descent for deep networks,"in *Proc. NeurIPS*, 2024.

[29] R. Panda et al.,"Dense backpropagation improves training for sparse mixture-of-experts,"*arXiv preprint arXiv:2504.12463*, 2025.

[30] Y. You et al.,"Large batch optimization for deep learning: Training BERT in 76 minutes,"in *Proc. ICLR*, 2020.