On the Difficulty of Training Non-negative Neural Networks

M. Kirtas[3,1], N. Passalis[2], N. Pleros[1], and A. Tefas[1]
[1]*Dept. of Informatics*, [2]*Dept. of Chemical Engineering*,
Aristotle University of Thessaloniki, Greece
[3]*Centre de Nanosciences et de Nanotechnologies, CNRS, Université Paris-Saclay, France*
emmanouil.kirtas@universite-paris-saclay.fr, passalis@auth.gr, {npleros, tefas}@csd.auth.gr

***Abstract*—Although non-negative neural networks are equipped with intrinsic properties that can be utilized in emerging analog accelerators and interpretable models, their use is limited due to the inherent constraints of the parameters. In fact, existing work fails to generalize across a wide range of architectures, resulting in significant performance degradation compared to their regular counterparts. In this work, we study the difficulties of non-negative optimization, claiming that without the proper initialization of non-negative parameters, the model is a priori limited before even the optimization process takes place. To this end, we propose a post-initialization method that can be applied on top of any existing initialization scheme, preserving the variance of activation units and, as a result, the expressivity of the non-negative models. The proposed method can be applied directly to convolutional and recurrent architectures, with the work providing experimental results in demanding scenarios involving easily saturated neural layers.**

***Index Terms*—Non-negative Neural Networks, Constrained Neural Networks, Non-negative Initialization, Non-negative Optimization**

I. Introduction

Although Non-negative Neural Networks have been studied merely in machine learning literature, they have recently emerged due to their interesting representation properties, which arise from their ability to eliminate canceling neurons [1] and enforce sparsity [2], [3]. Early works have demonstrated the benefits of part-based learning in non-negative deep learning architectures, presenting advantages in terms of explainability [4] and post-hoc analysis [5], which can, in turn, be utilized to further improve the performance of models [6], [7]. However, as highlighted in those works, the existing non-negative architectures and the applied optimization methods face limitations regarding their universality and their ability to generalize in traditionally applied deep learning architectures (such as CNNs) and larger datasets (other than MNIST), resulting in significant performance degradation.

Furthermore, training non-negative neural networks is also essential in emerging technologies that target high speed analog accelerators, such as Optical Neural Networks [8], [9]. Only very recent photonic applications utilize non-negative neural networks to deploy them in analog hardware for machine learning applications [10].

In fact, non-negative training involves different dynamics in the optimization process due to the constraints that are enforced upon trainable parameters. Initialization schemes are among those mechanisms that crucially affect the performance of non-negative training, especially in the initial stages, since they define the magnitude of parameters that, in turn, affect the variance of activations. However, as shown in this work, applying non-negative optimization directly to typically initialized networks is catastrophic for their performance, introducing new difficulties for study in the variance preserving initialization literature [11], [12], [13].

In this work, we study such difficulties and propose a method for the effective post-initialization of non-negative neural networks. The proposed method can be applied on top of any existing initialization scheme, unlocking the potential of non-negative neural networks, as it ensures high variance in the activation distribution and gradient propagation during the initial stage of training when a non-negative optimization method is applied. The proposed method ensures the expressivity of the models in the non-negative constrained high-dimensional space, allowing high performance, while it can be easily generalized to deeper architectures.

The rest of the paper is structured as follows. In Section II, the necessary background and related work are discussed. In Section III, the proposed method is introduced, and in Section IV, the experimental evaluations are presented. The conclusions are drawn in Section V.

II. Background & Related work

To understand the difficulty of training non-negative DL models, a further examination of the limitations of traditional initialization schemes when applied to non-negative models is needed. In Table I, the existing initialization schemes used in regular neural networks are presented, including a non-negative initialization alternative that has been applied in the literature [4], [3], [2]. Finally, Table II reports various update terms that can be integrated into optimization methods, ensuring the non-negativity of trainable parameters during training.

Typically, initialization schemes aim to set weights using a distribution that maintains a high activation variance. This prevents neuron saturation and ensures consistent gradient flow during backpropagation, mitigating the phenomenon of vanishing gradients [13], [14]. This is especially critical when employing high-slope activation units with narrow activation

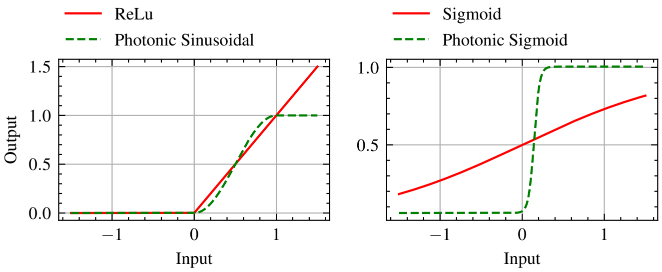

Fig. 1: Photonic activations functions can be easily saturated compared to typically ones used in most DL architectures

TABLE I: Initialization methods

Initialization	$w \sim$
(I.1) Xavier Norm.	$\mathcal{N}\left(0, \sqrt{\frac{2}{N_j+N_{j-1}}}\right) \in \mathbb{R}$
(I.2) Xavier Unif.	$U\left(-\sqrt{\frac{6}{N_j+N_{j-1}}}, \sqrt{\frac{6}{N_j+N_{j-1}}}\right) \in \mathbb{R}$
(I.3) Kaiming Norm.	$N\left(0, \sqrt{\frac{2}{N_j}}\right) \in \mathbb{R}$
(I.4) Kaiming Unif.	$U\left(-\sqrt{\frac{6}{N_j}}, \sqrt{\frac{6}{N_j}}\right) \in \mathbb{R}$
(I.3) Exponential	$\frac{\ln\left(U(0,1)\right)}{\lambda=100} \in \mathbb{R}_+$

windows, such as those found in hardware-implemented neurons in neuromorphic devices [15] and depicted in Figure 1. As has been extensively studied, a way to ensure gradient flow in the initial stage of training, especially in cases with easily saturated neurons, can be partially achieved by drawing weights from a random distribution, whether uniform or normal, whose variance is a factor of fan-in $N \in \mathbb{N}$ and fan-out $M \in \mathbb{N}$, as reported in Table I.

Given a typically initialized network, the optimization algorithms can be modified accordingly to ensure that, during training, the network preserves the non-negativity of the parameters. Such modifications include altering the update term, as presented in Table II. Even though there are some works [4], [3], [2] showing that such update terms when applied on top of typical initialization, can converge to a sub-optimal solution, they face challenges in scaling with deep architectures since they either saturate the neurons when weights approach zero, leading to vanishing gradients (e.g., Clip, Softplus, and Sigmoid), or they increase the weights, resulting in exploding gradients (e.g., Absolute and Squared). This leads non-negative neural networks to have significantly lower performance compared to regular models.

Alternatively, the weights of the network can be drawn directly from a non-negative distribution, allowing for the use of multiplicative optimization methods [16]. However, even if the optimization method ensures the best possible convergence, without the proper post-initialization of the network, the potential capability and expressivity of the model remain low. The optimization of non-negative trainable parameters based on the surface of the loss functions, which incorporates the non-negativity constraints, even if it converges to an optimal local minimum, does not ensure good performance in terms of accuracy, since the model is a priori limited.

TABLE II: Non-negative update terms

(U.0) **Update**	$\theta_t = \sigma_{nn}\left(\theta_{t-1} - \eta \frac{\partial L}{\partial \theta_{t-1}}\right)$
(U.1) Clip	$\sigma_{clip}(x) = \max\{0, x\} \in \mathbb{R}_+$
(U.2) Absolute	$\sigma_{abs}(x) = \lvert x \rvert \in \mathbb{R}_+$
(U.3) Sigmoid	$\sigma_{sig}(x) = \frac{1}{1+e^{-x}} \in (0,1)$
(U.4) Squared	$\sigma_{sqrt}(x) = x^2 \in \mathbb{R}_+$
(U.5) Softplus	$\sigma_{sft}(x) = \ln(1+e^x) \in \mathbb{R}_+$

In this work, we claim that this is attributed to the fact that conceptually, ANNs project the input features to a latent dimensional space, aiming to represent them in a more separable way (and, as a result, a more expressive way), seeking a hyperplane decision boundary that can classify them optimally. However, classifying non-negative features, even when they are linearly separable, is often impossible when using non-negative parameters in a classifier. For example, in a two-dimensional binary classification task, as depicted in Figure 2.a, the decision boundary of a traditional logistic regression classifier is given by:

$$x_1 = -\frac{w_0}{w_1}x_0 - \frac{b}{w_1} \in \mathbb{R}, \tag{1}$$

where $x_i \in [0, \alpha]$ is the input, $w_i \in \mathbb{R}$ and $b \in \mathbb{R}$ are the weights and biases of the classifier, respectively, and $i \in \{0, 1\}$. It can be easily concluded that there is a positive slope decision, with the slope given by $m = -w_0/w_1$, which requires a negative weight. In fact, training the linear classifier using SGD results in a positive slope decision boundary, as shown in Figure 2.a, achieving optimal performance with $w_0 < 0$. On the other hand, constraining the classifier to non-negative parameters results in a negative slope decision boundary that cannot discriminate between the two classes, as presented in Figure 2.b, even in this simple binary classification task.

III. Proposed Method

Inspired by the fact that challenging computational tasks, such as calculating the motion of our solar system's planets in a geocentric manner, can be easily solved by changing the coordinate system, for example, calculating the motion of the planets in a heliocentric system, we propose to initialize the network by rotating an equivalent portion of weights that are negative in the initial distribution and, in turn, associating them with the inverse input space. To this end, we claim that the proposed post-initialization scheme ensures the expressivity of the non-negative neural network, unlocking their universality and scalability capabilities while enabling performance close to that of their regular equivalents. The proposed scheme results in a non-negative initialized model, whose variance of each layer's activation distribution is the same to the original layer, ensuring that the performance of the model depends only on the optimization process, meaning that it is not a priori limited by the applied non-negative constraints.

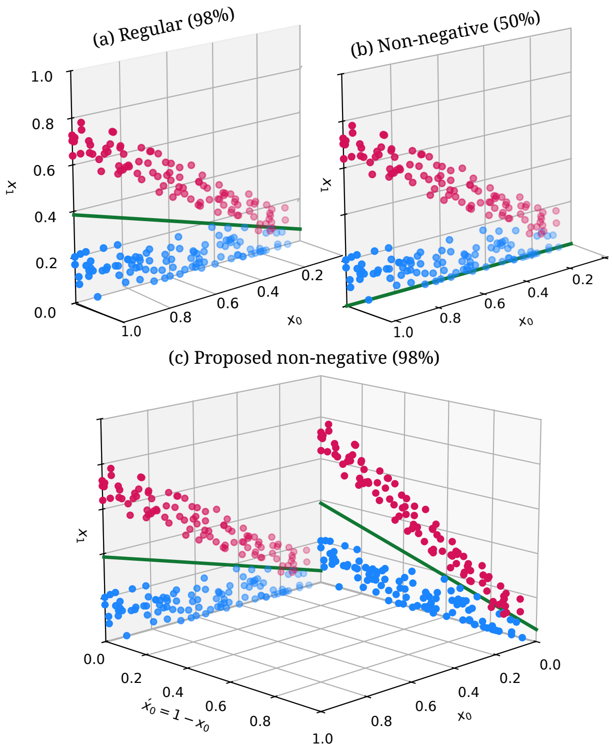

Fig. 2: (a) The resulting positive-slope decision boundary as acquired by the traditional optimization process. (b) Regular training of the non-negative classifier results in a non-positive slope decision boundary. (c) The proposed post-initialization.

Indeed, in the example of Figure 2.a, we can easily rotate the original problem and the classifier, by rotating and inverting the points and the decision boundary, getting an equivalent non-negative classifier, as depicted in Figure 2.c, using only non-negative parameters. The acquired non-negative classifier results in the decision boundary given by:

$$x_1 = -\frac{|w_0|}{w_1}x'_0 - \frac{b - a\,|w_0|}{w_1} \in \mathbb{R} \tag{2}$$

where $w_0 < 0$, $w_1 > 0$, $a = 1$ and $x'_0 = (a - x_0)$. The non-negative classifier leads to a flipped decision boundary and is equal to the original classifier performance.

To formulate this for fully connected multi-layered neural networks, let $\boldsymbol{z}^{(j)} \in \mathbb{R}^{N_j}$ be the output of the linear part of the j-th layer of a neural network architecture, where the linear function is denoted as $f^{(j)}(\cdot) : \mathbb{R}^{N_{j-1}} \to \mathbb{R}^{N_j}$, given by

$$\boldsymbol{z}^{(j)} = f^{(j)}(\boldsymbol{x}^{(j)}) = \boldsymbol{W}^{(j)}\boldsymbol{x}^{(j)} + \boldsymbol{b}^{(j)} \in \mathbb{R}^{N_j}, \tag{3}$$

where $\boldsymbol{W}^{(j)} \in \mathbb{R}^{N_j \times N_{j-1}}$, $\boldsymbol{b}^{(j)} \in \mathbb{R}^{N_j}$, and $\boldsymbol{x}^{(j)} \in \mathbb{R}^{N_{j-1}}$. Assuming that an activation function $\phi(\cdot) : \mathbb{R}^{N_j} \to \mathbb{R}_+^{N_j}$, where $\mathbb{R}_+$ denotes the set of positive real values, the output of the j-th layer is given by:

$$\boldsymbol{y}^{(j)} = \phi^{(j)}(\boldsymbol{z}^{(j)}) \in \mathbb{R}_+^{N_j}. \tag{4}$$

Typically, initializations applied to weights [12], [13] involve drawing weights from a distribution (e.g. $\boldsymbol{W} \sim \mathcal{N}_{N_j \times N_{j-1}}(\mu, \sigma)$, $\boldsymbol{W} \sim \mathcal{U}_{N_j \times N_{j-1}}(-a, a)$), where $\mu = 0$ denotes the mean of the distribution and $\sigma > 0$ defines the standard deviation, taking into account the fan-in $N_{j-1} \in \mathbb{N}$ and fan-out $N_j \in \mathbb{N}$ of the layer, as described in Table I. By directly applying any optimization algorithm that uses non-negative update functions, as reported in Table II, the negative weights are transformed into non-negative ones, with the update rule determining the acquired distribution. For instance, when the Clipping update (Equation U.1) is utilized, the initial normal distribution is transformed into a Rectified Gaussian Distribution. Similarly, when the Absolute update (Equation U.2) is applied, the network consists of the Folded Normal Distribution, as shown in Figure 3.

What is becoming clear is that after the first epoch of the optimization process, an aggressive reduction of variance is enforced in the weight distribution, leading a portion of neurons to saturation and significantly eliminating the expressivity of the network. In this work, we propose to rotate the input multidimensional-space to a non-negative axis, projecting the weights that were originally initialized as negative onto the $(\mathbb{1}_{N_{j-1}}\alpha^{(j)} - \boldsymbol{x}^{(j)}) \in [0, \alpha]^{N_{j-1}}$-axis, as conceptually depicted in Figure 3, with $\mathbb{1}_{N_{j-1}} \in [1, \ldots, 1]^{N_{j-1}}$. The $\alpha^{(j)} \geq max(\mathcal{X}^{(j)})$ denotes the rotation constant of the layer and *max* denotes the maximum element in the feasible set $\{x_i^{(j)} : \boldsymbol{x}^{(j)} \in [x_{min}, x_{max}]^{N_{j-1}} \subset \mathbb{R}_+^{N_{j-1}}\}$, defined as $\mathcal{X}^{(j)}$. The existence of the rotation constant $\alpha^{(j)}$ requires a closed interval of $\mathcal{X}^{(j)}$ that can be easily achieved in input through normalization or by considering a bounded activation function in the intermediate layers.

To this end, the initialized non-negative weights will retain their initial variance in those two axes as a projection. This can be expressed as

$$\boldsymbol{z}^{(j)} = \boldsymbol{W}_-^{(j)}(\mathbb{1}_{N_{j-1}}\alpha^{(j)} - \boldsymbol{x}^{(j)}) + \boldsymbol{W}_+^{(j)}\boldsymbol{x}^{(j)} + \boldsymbol{b}'^{(j)} \in \mathbb{R}_+^{N_j} \tag{5}$$

where $\boldsymbol{W}_+^{(j)} + \boldsymbol{W}_-^{(j)} = |\boldsymbol{W}^{(j)}|$ with $|\boldsymbol{W}^{(j)}| \in {\mathbb{R}_+}^{N_j \times N_{j-1}}$ denotes the absolute value of the original weight matrix and $\boldsymbol{b}'^{(j)} \in \mathbb{R}_+^{N_j}$ the bias vector. Both $\boldsymbol{W}_-^{(j)} = \max(0, -\boldsymbol{W}^{(j)}) \in \mathbb{R}_+^{N_j \times N_{j-1}}$ and $\boldsymbol{W}_+ = \max(0, \boldsymbol{W}^{(j)}) \in \mathbb{R}_+^{N_j \times N_{j-1}}$ are sparse matrices, each containing $\approx 50\%$ zero values.

The bias term can be initialized to any non-negative vector $\boldsymbol{b}'^{(j)} \in \mathbb{R}_+^{N_j}$, however, this will lead to a different variance in the output of the activation function, undermining the applied initialization scheme. To overcome this and ensure that the variance of the activation values is preserved during training, we propose initializing the biases based on the initialized weights and acquiring the same variance in the activation distribution. More specifically, we propose to accumulate the applied positive rotation of the weights in the $(\mathbb{1}_{N_{j-1}}\alpha^{(j)} - \boldsymbol{x}^{(j)})$ axis in the bias term and integrate it into a shifted activation function. Thus, the bias vector is initialized as

$$\boldsymbol{b}'^{(j)} = \tilde{\boldsymbol{b}}^{(j)} + \mathbb{1}_{N_j}c^{(j)} \in \mathbb{R}_+^{N_j}, \tag{6}$$

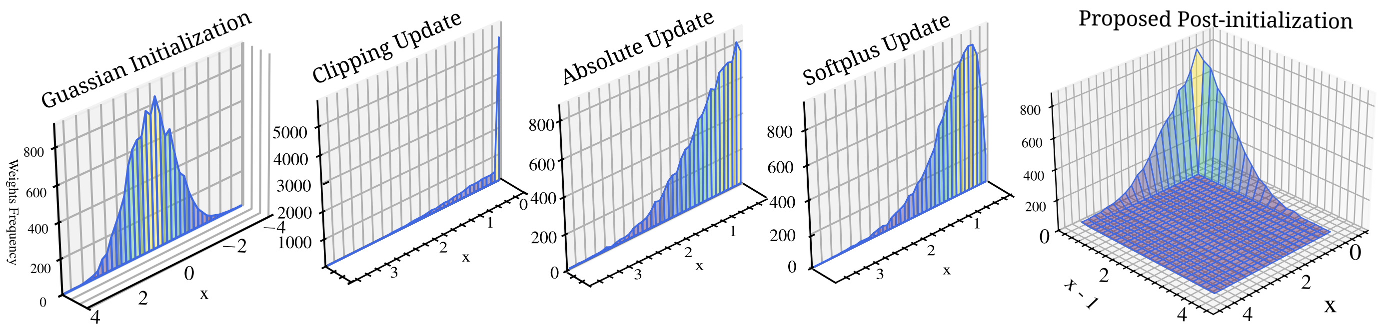

Fig. 3: From left to right: (a) Gaussian initialization, (b-d) After an epoch of non-training and (c) Proposed post-training initialization

where

$$\tilde{\boldsymbol{b}}^{(j)} = \boldsymbol{b}^{(j)} - \sum_{i=0}^{N_j} \sum_{l=0}^{N_l} w_{il-}^{(j)} \in \mathbb{R}_+^{N_j}, \tag{7}$$

and $c \in \mathbb{R}_+$ denotes the *activation shifting point* computed as

$$c^{(j)} = \max\{|b_0^{(j)}|, \ldots, |b_{N_j}^{(j)}\} \in \mathbb{R}_+. \tag{8}$$

The *activation shifting point* is applied to the original activation to slide it into the input domain, preserving the variance of the original initialization scheme. To ensure that the network will function in a non-negative manner, the original activation function must operate within a closed interval with a non-negative output space, i.e., $\phi^{(j)}(\boldsymbol{z}^{(j)}) : \mathbb{R}^{N_j} \to [\phi_{min}^{(j)}, \phi_{max}^{(j)}]^{N_j}$, where $0 \leq \phi_{min}^{(j)} < \phi_{max}^{(j)} < \infty$, with the shifted activation calculated as:

$$\boldsymbol{y}^{(j)} = \phi_c^{(j)}(\boldsymbol{z}^{(j)}) = \phi^{(j)}(\boldsymbol{z}^{(j)} - \mathbb{1}_{N_j} c^{(j)}) \in \mathbb{R}_+^{N_j}. \tag{9}$$

Remark. Consider a linear layer followed by an activation function. defined by Equations 3 and 4. Let the weights be initialized so that $w_{ij} \sim \mathcal{D}(-a, a)$, and let the resulting activation distribution be denoted $Y \sim \mathcal{A}(\mu, \sigma)$.

Given a post-initialization to non-negative weights $\boldsymbol{W}^{'} = |\boldsymbol{W}|$, where $\boldsymbol{W}_+ + \boldsymbol{W}_- = |\boldsymbol{W}|$, there exist a corrective non-negative bias vector corrective $\boldsymbol{b}^{'} \in \mathbb{R}_+^N$ and a shifted activation function $\phi_c(\cdot) : \mathbb{R}^N \to \mathbb{R}_+^N$ such that the new activation distribution $Y^{'}$ satisfies $Y^{'} \overset{d}{=} Y$.

Proof. Equation 3 can be expressed as

$$\boldsymbol{z} = -|\boldsymbol{W}_-|\boldsymbol{x} + \boldsymbol{W}_+\boldsymbol{x} + \boldsymbol{b} \in \mathbb{R}^{N_j}, \tag{10}$$

and considering that $\{x_i : \boldsymbol{x} \in [x_{min}, x_{max}]^M \subset \mathbb{R}_+^M\}$, the Equation 10 can be written as

$$\boldsymbol{z} = \mathbb{1}_N \alpha|\boldsymbol{W}_-| - |\boldsymbol{W}_-|\boldsymbol{x} + |\boldsymbol{W}_+|\boldsymbol{x} + (\boldsymbol{b} - \mathbb{1}_N \alpha|\boldsymbol{W}_-|) \in \mathbb{R}^N, \tag{11}$$

with the number of the layer omitted for simplicity.

This allows us to flip the input feature space, as shown in Fig. 2. To this end, the first two terms can be merged, while the last term can be integrated into an updated bias term. Therefore, Equation 11 can be written as:

$$\boldsymbol{z} = |\boldsymbol{W}_-|(\mathbb{1}_N \alpha - \boldsymbol{x}) + |\boldsymbol{W}_+|\boldsymbol{x} + \tilde{\boldsymbol{b}} \tag{12}$$

where $\tilde{\boldsymbol{b}} \in \mathbb{R}^N$ is given by Equation 7. In turn, the bias vector, which defines the threshold of neuron when an activation function follows, can be shifted in magnitude, Equation , while the activation is shifted on the opposite side of the input axis, Equation 9. Therefore, this confirms that $Y^{'} \overset{d}{=} Y$ is satisfied layer-wise. □

It should be mentioned that the introduced remark can be easily generalized to Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Furthermore, the proposed post-initialization can be applied on top of any initialization scheme and combined with any optimization method that preserves the sign of trainable parameters, unlocking also the integration of multiplicative updates to non-negative training [17], [18], [19].

In this work, we utilize the recently proposed Multiplicative Stochastic Gradient Descent (MSGD) [16]. The MSGD is applied after the post-initialization on and constrains the trainable parameter $\theta^{(j)} = \{w_{il}^{(j)}, b_i^{(j)}\}_{j=0}^{J}$ of j-th layer, consisting of weights $\boldsymbol{W}^{(j)} \in \mathbb{R}^{N_j \times N_{j-1}}$ and biases $\boldsymbol{b}^{(j)} \in \mathbb{R}^{N_j}$, to non-negative quantities. The optimization algorithm picks a point $\theta_\tau^{(j)}$, at each time step $\tau \in \mathbb{N}$, and updates it according to:

$$\theta_\tau^{(j)} = \theta_{\tau-1}^{(j)} + |\theta_{\tau-1}^{(j)}| \tanh\left(-\eta_{in} \frac{\partial J}{\partial \theta_{\tau-1}^{(j)}}\right) \eta_{out} \in \mathbb{R}_+, \tag{13}$$

where $\eta_{in} \in \mathbb{R}^+$ is the inner and $\eta_{out} \in (0, 1]$ the outer learning rate.

IV. Experimental Results

To evaluate the proposed post-initialization method, we deploy a benchmark with extremely easily saturated neurons as arise in Optical Neural Networks [15]. More specifically, before evaluating the proposed method in larger ReLU architectures, we apply different types of vanilla architectures, including RNNs. Two easily saturated photonic activation functions are evaluated, the Photonic Sigmoid [20] and the Photonic Sinusoidal [21].

A. Post-initialization ablation study

The role of variance in weights is vital during training, even in stages where the model converges to a suboptimal distribution in the early epochs. In Figure 4, the evolution of the variance, when non-negative post-initialization is applied, is presented with reference to regular training. More precisely, we report the variance of the classification layer of the

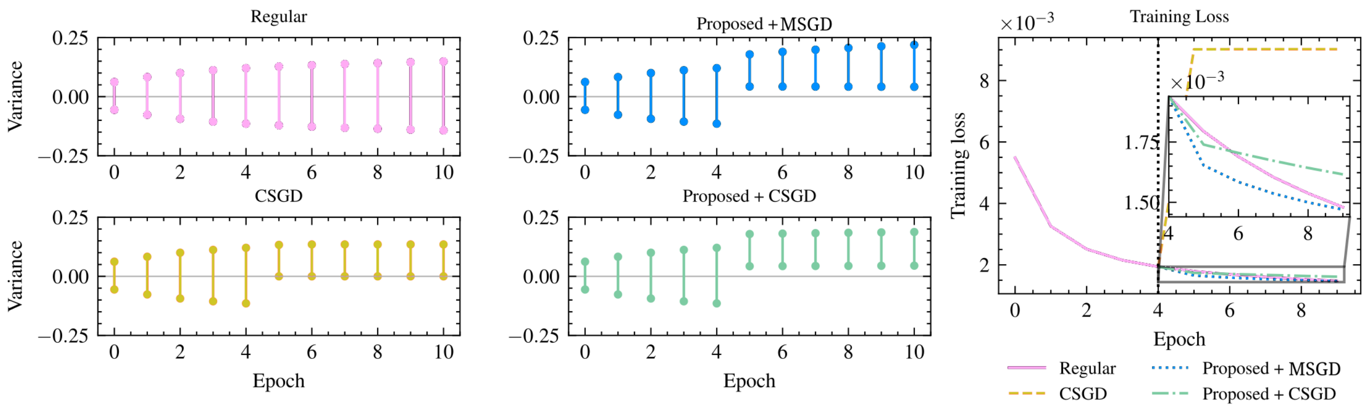

Fig. 4: Depicts the variance of classification layer in photonic sigmoid architecture applied on FMNIST dataset. Additionally, in the right subfigure, the loss during the training is presented. Networks are pre-trained for 5 epochs with SGD and then they are either perform the proposed post-training initialization or clipping is applied directly. The figure in the top left shows the variance of the regular training.

photonic sigmoid architecture used in the FMNIST dataset. In the upper left subfigure, the variance of the regular training is reported. The same network, after some epochs of training, is post-initialized in the non-negative space and trained in a non-negative manner by applying the three evaluated methods. Specifically, we compared three approaches after an initial 5-epoch training phase: (a) direct Clipping SGD (CSGD), (b) CSGD following our post-initialization, and (c) MSGD following our post-initialization.

The results indicate that applying clipping directly to a pre-trained model causes a precipitous drop in variance, leading to a significant loss of learned features and performance degradation. Applying the post-initialization before clipping ensures that the weights maintain sufficient distance from the zero-boundary. This prevents variance diminution and allows the training to proceed smoothly, preventing weight atrophy, even in demanding cases of narrow activation windows. Finally, it is demonstrated that the proposed initialization can work properly with multiplicative updates, such as the MSGD, converging faster to a better local minimum even compared to regular training.

To validate these findings across architectures, Tables III and IV report accuracy and variance for MLPs, CNNs, and RNNs. While direct clipping remains catastrophic, our post-initialization preserves the information density of the pre-trained parameters. Notably, this stability is achieved without Batch Normalization, highlighting the robustness of the method in unstable, non-negative training environments.

B. Non-negative training from scratch

The proposed post-initialization demonstrates remarkable flexibility across various initialization schemes (Table VII). Most notably, as evidenced in Table VIII, traditional methods—including Xavier and Kaiming completely fail to train deep architectures like ResNet34 under non-negative constraints, resulting in accuracies near 1.3%. In contrast, our method overcomes this initialization bottleneck, enabling the MSGD optimizer to reach 82.40% accuracy, effectively bridging the gap between constrained hardware-native training and regular unconstrained performance. These findings confirm that the framework generalizes across CNN and RNN architectures, consistently allowing models to converge and leading to a higher average evaluation accuracy over the baselines.

C. Initializations & Non-negative optimizations

The proposed non-negative post-initialization can be applied on top of any initialization scheme, as depicted in Table VII, demonstrating that it can be effectively combined with multiplicative updates [15]. Finally, in Table VIII, several non-negative optimization alternatives are applied on top of the traditionally used initialization methods without, however, providing any convergence to the models. On the other hand, even aggressive weight clipping leads to promising results when combined with the proposed post-initialization, with the MSGD performing the best after 200 epochs of training.

V. Conclusions, Limitation & Future work

In this work, we introduce a post-initialization framework designed to augment traditional initialization schemes while preserving activation variance in non-negative neural networks. We provide theoretical proofs demonstrating that through specific weight rotations and input-space inversions, a non-negative layer can be constructed that maintains the exact activation distribution of the original unconstrained model.

This study highlights the critical role of initialization in non-negative training, particularly for architectures employing saturating activation functions. While the specific dynamics of non-negative optimization require continued investigation, addressing the initialization bottleneck is a foundational step. By stabilizing the starting conditions, this work paves the way for deeper explorations into optimization behavior, potentially unlocking the unique computational advantages of constrained neural networks.

Acknowledgments: This work was partially supported by the European Commission (EC) through the H2020 Project PLASMONIAC (871391) and by the European Union's Horizon Europe research and innovation program under Grant Agreement No. 101135523 (SYMPHONY).

TABLE III: Non-negative post training (MLPs)

Dataset	Regular + Clipping	Proposed + Clipping	Proposed + Absolute	Proposed + MSGD
		Photonic Sigmoid		
MNIST	11.00 ± 0.00	97.31 ± 0.90	97.27 ± 0.09	**97.40 ± 0.06**
FMNIST	10.00 ± 0.00	84.77 ± 0.21	84.65 ± 0.19	**85.01 ± 0.12**
CIFAR10	10.00 ± 0.00	36.34 ± 0.37	36.65 ± 0.35	**36.81 ± 0.33**
		Photonic Sinusoidal		
MNIST	09.74 ± 0.00	97.85 ± 0.10	97.86 ± 0.09	**97.86 ± 0.07**
FMNIST	10.00 ± 0.00	86.62 ± 0.09	86.90 ± 0.13	**86.95 ± 0.17**
CIFAR10	10.00 ± 0.00	40.96 ± 0.30	41.09 ± 0.18	**41.19 ± 0.38**

TABLE IV: Non-negative post training (CNNs & RNNs)

Config.	Photonic Sigmoid			Photonic Sinusoidal		
Method	Proposed + Absolute	Proposed + Clipping	Proposed + MSGD	Proposed + Absolute	Proposed + Clipping	Proposed + MSGD
MNIST	96.98 ± 0.07	96.85 ± 0.96	**97.60 ± 1.83**	98.72 ± 0.05	98.58 ± 0.12	**99.04 ± 0.07**
FMNIST	83.92 ± 0.36	83.90 ± 0.46	**84.87 ± 0.33**	**87.69 ± 0.15**	87.59 ± 0.20	87.63 ± 0.19
CIFAR10	76.18 ± 0.25	74.30 ± 0.22	**77.39 ± 0.56**	78.21 ± 0.31	77.30 ± 0.73	**79.95 ± 0.21**
Malimg*	92.46 ± 3.2	93.53 ± 2.76	**96.60 ± 0.78**	92.18 ± 4.5	92.18 ± 4.52	**93.53 ± 4.30**
Names	**62.61 ± 1.7**	53.94 ± 1.44	58.37 ± 1.05	**72.83 ± 0.31**	66.64 ± 0.83	67.14 ± 0.54
FI2010†	0.1098 ± 0.011	0.1050 ± 0.0070	**0.1220 ± 0.0022**	0.1100 ± 0.0038	0.1434 ± 0.0014	**0.1840 ± 0.0018**

∗F1 score, †Cohen's kappa score are reported, since the datasets are highly unbalanced

TABLE V: Non-negative training from scratch (MLPs)

Dataset	Exp. Initialization + Absolute	Exp. Initialization + Clipping	Exp. Initialization + MSGD	Proposed + Absolute	Proposed + Clipping	Proposed + MSGD
			Photonic Sigmoid			
MNIST	11.35 ± 0.0	11.35 ± 0.00	11.35 ± 0.00	92.91 ± 0.25	92.52 ± 0.52	**93.34 ± 1.11**
FMNIST	10.00 ± 0.0	10.00 ± 0.00	10.00 ± 0.00	83.27 ± 0.21	82.31 ± 1.29	**84.09 ± 0.56**
CIFAR10	10.00 ± 0.0	10.00 ± 0.00	10.00 ± 0.00	38.21 ± 1.1	37.25 ± 1.57	**37.61 ± 1.76**
			Photonic Sinusoidal			
MNIST	85.12 ± 0.5	86.36 ± 1.39	35.31 ± 6.11	93.22 ± 0.15	92.51 ± 0.78	**94.14 ± 0.90**
FMNIST	69.04 ± 1.1	68.95 ± 3.71	9.97 ± 0.02	83.37 ± 0.26	82.91 ± 0.55	**83.57 ± 0.26**
CIFAR10	10.00 ± 0.0	10.00 ± 0.00	10.00 ± 0.00	**36.46 ± 0.75**	34.96 ± 2.44	35.41 ± 1.43

TABLE VI: Non-negative training from scratch (CNNs & RNNs)

Config.	Photonic Sigmoid			Photonic Sinusoidal		
Method	Proposed + Absolute	Proposed + Clipping	Proposed + MSGD	Proposed + Absolute	Proposed + Clipping	Proposed + MSGD
MNIST	**98.40 ± 0.07**	97.69 ± 0.20	98.14 ± 0.23	98.68 ± 0.033	98.54 ± 0.14	**98.71 ± 0.05**
FMNIST	**85.33 ± 0.57**	83.81 ± 0.85	83.87 ± 0.19	**88.08 ± 0.17**	87.77 ± 0.36	87.86 ± 0.36
CIFAR10	71.46 ± 0.88	71.61 ± 1.04	**72.16 ± 0.54**	73.62 ± 0.9	69.87 ± 1.73	**75.16 ± 1.83**
Malimg*	90.31 ± 6.31	91.47 ± 4.37	**91.98 ± 4.00**	97.19 ± 0.32	96.53 ± 0.42	**97.26 ± 0.31**
Names	**61.35 ± 1.80**	54.00 ± 2.13	56.68 ± 1.11	**69.53 ± 0.26**	64.17 ± 0.14	68.70 ± 0.78
FI2010†	0.1300 ± 0.0045	0.1338 ± 0.0045	**0.1498 ± 0.0112**	**0.1901 ± 0.014**	0.1732 ± 0.0175	0.1789 ± 0.0102

∗F1 score, †Cohen's kappa score are reported, since the datasets are highly unbalanced.

Manos Kirtas is currently affiliated with Centre de Nanosciences et de Nanotechnologies, CNRS, Université Paris-Saclay, France. This work is partially made when Manos Kirtas was in Aristotle University of Thessaloniki, Greece.

TABLE VII: Proposed post-initialization on top of different initialization schemes (MLPs)

	Proposed + Absolute				Proposed + MSGD			
	Photonic Sigmoid		Photonic Sinusoidal		Photonic Sigmoid		Photonic Sinusoidal	
Initialization	MNIST	FMNIST	MNIST	FMNIST	MNIST	FMNIST	MNIST	FMNIST
Kaiming Normal	0.9374	0.8362	0.962	0.8729	**0.9462**	**0.8410**	**0.9668**	**0.8771**
Kaiming Uniform	**0.9492**	0.8467	0.9607	0.8743	**0.9492**	**0.8482**	**0.9654**	**0.8753**
Xavier Normal	0.949	0.8419	0.9641	0.8695	**0.9513**	**0.8552**	**0.9651**	**0.875**
Xavier Uniform	0.9528	**0.8559**	0.9606	0.8715	**0.9567**	**0.8559**	**0.9682**	**0.873**

TABLE VIII: Average training accuracy on 5 runs at the 200th epoch of training applying ResNet34 on CIFAR100 dataset

	Default	Xavier Normal	Xavier Uniform	Kaiming Normal	Kaiming Uniform
			Regular Training		
Update	*91.59*	*90.47*	*90.33*	*89.94*	*89.59*
			Typical Initialization		
Softplus	1.35	1.81	1.00	1.87	1.50
Sigmoid	1.35	1.00	1.35	1.03	1.08
Squared	1.83	1.89	1.20	1.00	1.80
Clipping	1.45	1.38	1.30	1.47	1.39
Absolute	1.51	1.32	1.53	1.31	1.46
			Post-initialization (ours)		
Clipping	80.42	75.44	74.75	62.97	62.58
Absolute	81.92	79.89	80.01	62.09	62.17
MSGD	**82.40**	**80.22**	**81.12**	**63.12**	**64.52**

REFERENCES

[1] T. J. Piotrowski *et al.*, "Fixed points of nonnegative neural networks," *Journal of Machine Learning Research*, vol. 25, no. 139, pp. 1–40, 2024.

[2] B. O. Ayinde *et al.*, "Deep learning of constrained autoencoders for enhanced understanding of data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 3969–3979, 2018.

[3] J. Chorowski *et al.*, "Learning understandable neural networks with nonnegative weight constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, 2015.

[4] M. Kirtas *et al.*, "Using part-based representations for explainable deep reinforcement learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2023, pp. 420–432.

[5] V. Wargnier-Dauchelle *et al.*, "Constrained non-negative networks for a more explainable and interpretable classification," in *Medical Imaging with Deep Learning*, 2024.

[6] M. Ferro *et al.*, "Non-negative structured pyramidal neural network for pattern recognition," in *Proc. Intl. Joint Conf. on Neural Networks*, 2018, pp. 1–7.

[7] X. Hu *et al.*, "Enhancing uncertainty estimation and interpretability with bayesian non-negative decision layer," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=xJXq6FkqEw

[8] A. Prapas *et al.*, "Time-space-wavelength multiplexed photonic tensor core using wdm sige eam array chiplets," *Optics Express*, vol. 33, no. 17, pp. 36 960–36 972, 2025.

[9] C. Pappas *et al.*, "Reaching the peta-computing: 163.8 tops through multidimensional awgr-based accelerators," *Journal of Lightwave Technology*, 2025.

[10] L. Avramelou *et al.*, "Neuromorphic photonic neural networks for on-chip gas spectra classification," *Applied Soft Computing*, p. 114467, 2025.

[11] R. Pascanu *et al.*, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. Pmlr, 2013, pp. 1310–1318.

[12] X. Glorot *et al.*, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[13] K. He *et al.*, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[14] ——, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[15] M. Kirtas *et al.*, "Robust architecture-agnostic and noise resilient training of photonic deep learning models," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–10, 2022.

[16] ——, "Multiplicative stochastic gradient descent for fast and robust deep learning training," in *2025 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2025, pp. 1–8.

[17] Z. Yang *et al.*, "Multiplicative updates for non-negative projections," *Neurocomputing*, vol. 71, no. 1, pp. 363–373, 2007, dedicated Hardware Architectures for Intelligent Systems Advances on Neural Networks for Speech and Audio Processing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231207000318

[18] J. Bernstein *et al.*, "Learning compositional functions via multiplicative weight updates," in *Advances in Neural Information Processing Systems*, H. Larochelle *et al.*, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 13 319–13 330.

[19] M. Kirtas *et al.*, "Multiplicative update rules for accelerating deep learning training and increasing robustness," *Neurocomputing*, vol. 576, p. 127352, 2024.

[20] G. Mourgias-Alexandris *et al.*, "An all-optical neuron with sigmoid activation function," *Opt. Express*, vol. 27, no. 7, pp. 9620–9630, Apr 2019.

[21] N. Passalis *et al.*, "Training deep photonic convolutional neural networks with sinusoidal activations," *IEEE Trans. Emerg. Topics Comput. Intell.*, pp. 1–10, 2019.