Training Neural Networks by Optimizing Neuron Positions

Laura Erb[1,2], Tommaso Boccato[3](✉), Alexandru Vasilache[1,2], Juergen Becker[2], Nicola Toschi[3,4]
laura.erb1@gmx.de, tommaso.boccato@uniroma2.it, vasilache@fzi.de, juergen.becker@kit.edu, nicola.toschi@uniroma2.eu

[1] FZI Research Center for Information Technology, Karlsruhe, Germany
[2] Karlsruhe Institute of Technology, Karlsruhe, Germany
[3] Department of Biomedicine and Prevention, University of Rome Tor Vergata, Rome, Italy
[4] A.A. Martinos Center for Biomedical Imaging and Harvard Medical School, Boston, USA

Abstract. The high computational complexity and increasing parameter counts of deep neural networks pose significant challenges for deployment in resource-constrained environments, such as edge devices or real-time systems. To address this, we propose a parameter-efficient neural architecture where neurons are embedded in Euclidean space. During training, their positions are optimized and synaptic weights are determined as the inverse of the spatial distance between connected neurons. These distance-dependent wiring rules replace traditional learnable weight matrices and significantly reduce the number of parameters while introducing a biologically inspired inductive bias: connection strength decreases with spatial distance, reflecting the brain's embedding in three-dimensional space where connections tend to minimize wiring length. We validate this approach for both multi-layer perceptrons and spiking neural networks. Through a series of experiments, we demonstrate that these spatially embedded neural networks achieve a performance competitive with conventional architectures on the MNIST dataset. Additionally, the models maintain performance even at pruning rates exceeding 80% sparsity, outperforming traditional networks with the same number of parameters under similar conditions. Finally, the spatial embedding framework offers an intuitive visualization of the network structure.

Keywords: Neuron Position Optimization, Gradient Descent, Spatial Embeddings, Spiking Neural Networks, Network Pruning

1 Introduction

In scenarios requiring real-time inference, enhanced user privacy, and reduced reliance on cloud-based systems, it is often preferred to deploy Deep Neural Networks (DNNs) directly on edge devices. However, the increasing computational complexity and parameter counts of modern DNNs present a significant

challenge for deployment in resource-constrained environments [14]. Addressing this challenge requires energy-efficient neural architectures with reduced memory footprints. Spiking Neural Networks (SNNs) offer a promising solution to meet low-energy consumption requirements. Their biologically inspired, spike-based computations enable highly efficient processing while maintaining high accuracy [4]. Nonetheless, in addition to energy efficiency, memory constraints on edge devices require further optimization techniques to reduce the number of parameters.

Incorporating biologically inspired inductive biases into the network architecture can reduce network complexity and parameters while guiding learning and improving generalization [5, 7, 8]. For example, Convolutional Neural Networks (CNNs) effectively reduce parameters by mimicking the concept of local receptive fields in the visual cortex [13]. Recent research showed that optimizing wiring rules based on gene expressions, rather than training individual weights, can improve learning efficiency and generalization while maintaining performance [2]. This approach was inspired by the fact that certain neural circuits in the brain are prewired, which means their connectivity is genetically encoded rather than learned through experience [3]. However, the complexity of these circuits far exceeds the information capacity of the genome, a challenge known as the genomic bottleneck [15]. This implies that the genome likely encodes generalized wiring rules rather than specifying individual connections.

Inspired by the brain's embedding in space and its associated spatial constraints, previous work has explored embedding neurons of Artificial Neuronal Networks (ANNs) in three-dimensional Euclidean space. Studies have shown that biologically-informed spatial constraints on connection weights lead to connectivity patterns similar to those observed in the brain [1, 6, 11, 16]. However, in these previous approaches, neuron positions remained fixed throughout training and influenced computation only through regularization terms. This raises the question: How would allowing neuron positions to be dynamically optimized during training impact both network structure and function?

To address this question, we propose a parameter-efficient neural architecture in which artificial neurons are embedded within Euclidean space. Their positions are optimized during training, and synaptic weights between connected neurons are computed as the inverse of their distance. By replacing explicit weight matrices with a wiring rule that specifies the connections based on neuron distance, we achieve a more parameter-efficient model representation. In this work, we evaluate this approach for Multi-layer Perceptrons (MLPs) and SNNs.

2 Methods

We introduce a parameter-efficient neural architecture where each neuron is assigned a position in three-dimensional Euclidean space. During training, these positions are optimized and the connection weights between neuron pairs are computed based on their spatial distances. This approach differs from traditional ANNs, which learn by optimizing individual connection weights. Assigning each

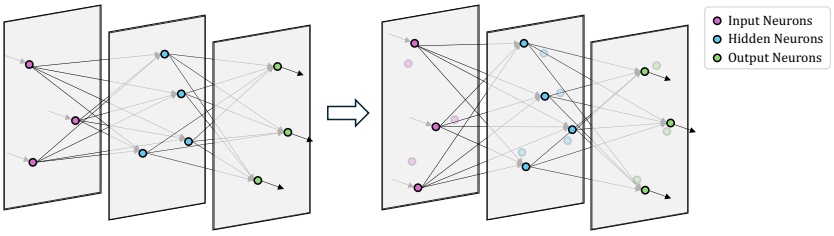


Fig. 1: An illustration of a three-layer feedforward network embedded in three-dimensional Euclidean space. Neurons optimize their positions within their respective two-dimensional layers.

neuron i a position p_i in Euclidean space, the weight of the connection between neurons i and j is then defined as a differentiable function of their distance:

$$w_{ij} = \frac{1}{\|p_i - p_j\|_2}. \tag{1}$$

This formula for the connection weights is inspired by the inverse relation between the conductance G, which is analogous to synaptic strength, and the length L of the resistive material in an electrical circuit:

$$G = \frac{A}{\rho \cdot L}, \tag{2}$$

where A is the cross-sectional area and ρ is the resistivity of the material.

We organize neurons into layers by keeping their z-coordinate fixed to correspond to their layer's index. This ensures a structured feedforward flow, where neurons can optimize their positions within their respective two-dimensional layers, as illustrated in Fig. 1. To enable negative connection weights, we distinguish between excitatory and inhibitory neurons by multiplying their output by a biologically inspired inhibition mask. Each neuron is assigned a continuous inhibition value, which is transformed by a scaled steep sigmoid function to ensure that neurons with positive inhibition values receive an inhibition mask close to 1 (excitatory) and those with negative values a mask close to -1 (inhibitory). The inhibition values are optimized during training, allowing the model to flexibly learn the distribution of excitatory and inhibitory neurons.

Computing the connection weights from neuron positions, rather than explicitly defining the entire weight matrix, significantly reduces the number of parameters in the model while preserving the same number of connections as in traditional architectures. Let N_L be the number of layers, where each layer $l \in \{1, ..., N_L\}$ contains n_l neurons. In a traditional MLP, the total number of parameters consists of both the connection weights and the bias terms. It is given by:

$$\sum_{i=1}^{N_L - 1} n_i n_{i+1} + \sum_{i=2}^{N_L} n_i. \tag{3}$$

In contrast, a spatially embedded MLP optimizes neuron positions instead of the connection weights. Its total number of parameters consists of the neuron coordinates, the bias terms, and the inhibition mask. This can be formulated as:

$$(d-1)\cdot\sum_{i=1}^{N_L} n_i + \sum_{i=2}^{N_L} n_i + \sum_{i=1}^{N_L-1} n_i, \tag{4}$$

where $d = 3$ is the dimensionality of the metric space in which the neurons are embedded. Each neuron optimizes only $d-1 = 2$ coordinates, because the z-coordinate remains fixed to correspond to the neuron's layer. With n representing the total number of neurons in the network, the number of parameters in a traditional MLP scales as $O(n^2)$, whereas a spatially embedded MLP reduces this complexity to $O(n)$. This parameter-efficient framework can be combined with existing compression techniques such as pruning or quantization, thereby complementing and extending prior work.

3 Experiments and Results

The proposed concept can be integrated into different neural architectures, such as MLPs or Recurrent Neural Networks (RNNs), without requiring fundamental changes to their core design. In this work, we specifically focus on embedding MLPs and fully connected SNNs in three-dimensional Euclidean space. For training and evaluation, we use the MNIST [13] dataset. All models in this work have three layers, with 10 input neurons and 784 output neurons. Models were trained with three different random seeds, with cross-entropy loss as the objective function. Training is performed in mini-batches of size 600 using the Adam optimizer [12]. The SNNs are implemented using Leaky Integrate-and-Fire (LIF) neurons from the SNNTorch library [10], with a threshold value of 1 and a beta value of 0.95, which are not modified during training.

3.1 Performance Evaluation

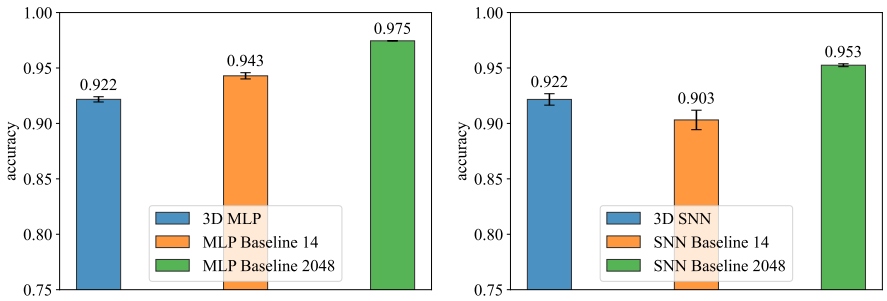
We implement an MLP embedded in a three-dimensional Euclidean space with fixed layer distances set to one across all layers. It has a single hidden layer with 2,048 neurons, resulting in a total of 10,574 learnable parameters. After being trained for 300 epochs with a learning rate of 0.005, the model achieves a test accuracy of 0.9018 ± 0.0042. Next, we relax the constraint that all layers must share the same fixed layer distance and implement a 3D MLP with independently learned layer distances, referred to as 3D MLP. This model achieves an improved test accuracy of 0.9217 ± 0.0024 with only a minor increase in the number of parameters (10,576). We compare the performance of the spatially embedded MLPs to two MLP baselines, which have the same architecture but vary in the number of hidden neurons. They are designed to match either the number of parameters or number of hidden neurons to more accurately assess the affect of the distance dependent wiring rule. MLP Baseline 2048 has 2,048 hidden neurons

Table 1: Accuracy comparison of the 3D MLP (with 2,048 hidden neurons) and the baseline MLPs (with 14 and 2,048 hidden neurons) on the MNIST [13] dataset. Models were trained using three different random seeds and accuracy is reported as the mean ± standard deviation.

Model	Test Accuracy	Number of Parameters	Learning Rate
3D MLP	0.9217 ± 0.0024	10,576	0.005
MLP Baseline 14	0.9429 ± 0.0028	11,140	0.001
MLP Baseline 2048	0.9745 ± 0.0003	1,628,170	0.001

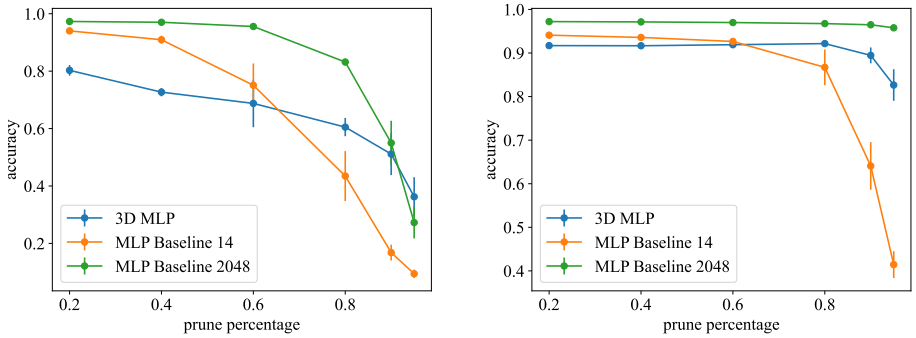
to match the number of neurons in the 3D MLP, but contains significantly more parameters. MLP Baseline 14 is a more compact MLP with only 14 hidden neurons, designed to have a parameter count comparable to that of the 3D MLP. After training for 300 epochs with learning rate 0.001, the two baselines outperform the spatially embedded MLPs (see Table 1 and Fig. 2a).

We further evaluate the spatial embedding framework for fully connected SNNs. Analogous to the evaluation of the 3D MLPs, we train an SNN embedded in 3D space with 2,048 hidden neurons with fixed layer distances. After training for 200 epochs with learning rate 0.005, the model achieves a test accuracy of 0.9187 ± 0.0027. Allowing the layer distances to be independently learnable for each layer slightly improves the performance of the 3D SNN to 0.9216 ± 0.0052. When compared to conventional fully connected SNNs trained for 200 epochs with learning rate 0.001, SNN Baseline 2048 with 2,048 hidden neurons outperforms the 3D SNNs, while having the same number of weights but significantly more parameters. In contrast, the 3D SNNs outperform the SNN Baseline 14, which has only 14 hidden neurons and a number of parameters comparable to that of the 3D SNNs (see Table 2 and Fig. 2b).



(a) MLP Performance Comparison (b) SNN Performance Comparison

Fig. 2: Accuracy comparison of the (a) 3D MLP and (b) 3D SNN (with 2,048 hidden neurons) with the baseline models (with 14 and 2,048 hidden neurons) on the MNIST [13] dataset. Models were trained using three different random seeds, with error bars representing the standard deviation across runs.



(a) Models are pruned after training. (b) Models are pruned during training.

Fig. 3: Accuracy comparison of the 3D MLP (with 2,048 hidden neurons) and the baseline MLPs (with 14 and 2,048 hidden neurons) on the MNIST [13] dataset for different pruning percentages. The models are pruned (a) after training and (b) during training by setting the smallest percentage of weights to zero. Models were trained using three different random seeds, with error bars representing the standard deviation across runs.

Table 2: Accuracy comparison of the 3D SNN (with 2,048 hidden neurons) and the baseline SNNs (with 14 and 2,048 hidden neurons) on the MNIST [13] dataset. Models were trained using three different random seeds, and accuracy is reported as the mean ± standard deviation.

Model	Test Accuracy	Number of Parameters	Learning Rate
3D SNN	0.9216 ± 0.0052	10,576	0.005
SNN Baseline 14	0.9031 ± 0.0088	11,140	0.001
SNN Baseline 2048	0.9525 ± 0.0013	1,628,170	0.001

3.2 Magnitude-Based Weight Pruning

Mimicking the brain's tendency to minimize wiring length [9], we prune the longest connections within our spatially embedded models. Since the connection weights are the inverse of the distance between neurons, this is equivalent to removing the connections with the smallest weights, effectively implementing magnitude-based weight pruning.

We first apply post-training pruning, where weights are pruned only after the model has been fully trained. We evaluate this approach for the models that were introduced in the previous Section 3.1, namely the 3D MLP with 10,576 parameters and 1,626,112 weights, MLP Baseline 14 with 11,140 parameters and 11,116 weights, and MLP Baseline 2048 with 1,628,170 parameters and 1,626,112 weights. For the baselines, removing connections is equivalent to removing parameters. The results of post-training pruning are visually presented in Fig. 3a.

Table 3: Accuracy comparison of the Relaxed 3D MLPs with learned z-coordinates, the 3D MLP, and the baseline MLPs (with 17 and 2,048 hidden neurons) on the MNIST [13] dataset. Models were trained using three different random seeds, and accuracy is reported as the mean ± standard deviation.

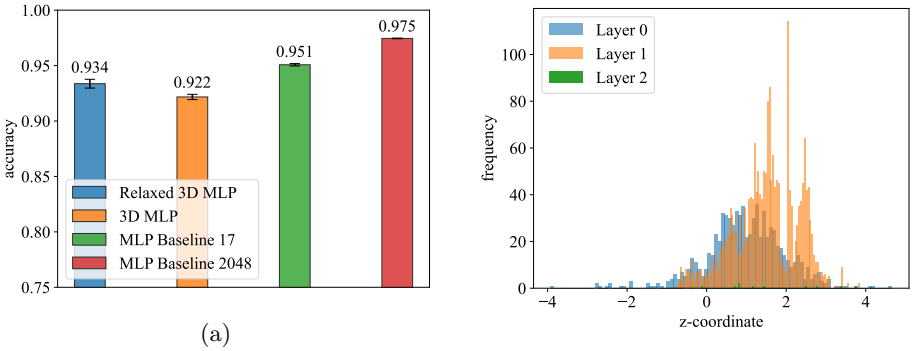
Model	Test Accuracy	Number of Parameters	Learning Rate
Relaxed 3D MLP	0.9337 ± 0.0040	13,416	0.005
3D MLP	0.9217 ± 0.0024	10,576	0.005
MLP Baseline 17	0.9507 ± 0.0011	13,525	0.001
MLP Baseline 2048	0.9745 ± 0.0003	1,628,170	0.001

Initially, MLP Baseline 14 outperforms the 3D MLP, but for pruning percentages of 80% or higher, the 3D MLP achieves better performance. However, despite both models having the same number of parameters, MLP Baseline 14 contains significantly fewer hidden neurons and connection weights. For a fair comparison, MLP Baseline 2048 has the same number of connection weights as the 3D MLP. It generally outperforms the spatially embedded MLP, but at an extreme pruning percentage of 95%, the 3D MLP outperforms even the MLP Baseline 2048, which has significantly more parameters.

In the second approach, pruning is integrated into the training process. In each forward pass, the smallest connection weights are identified and removed. We apply this strategy to 3D MLP, MLP Baseline 14, and MLP Baseline 2048. The results of pruning during training are illustrated in Fig. 3b. As observed in post-training pruning, MLP Baseline 14 initially outperforms the 3D MLP. However, for pruning percentages above 60%, the 3D MLP surpasses the MLP Baseline 14, which has fewer connections and neurons but the same number of parameters. When pruning during training, MLP Baseline 2048 consistently outperforms 3D MLP while also having significantly more parameters.

3.3 Optimizing Z-Coordinates

While structuring neurons into spatial layers provides a simple and organized framework, it limits the model's expressiveness by constraining neuron placement. Therefore, we relax the spatial layer hierarchy and allow neurons to optimize their z-coordinate during training while preserving the layer-based connectivity for computational simplicity. This relaxation adds an additional learnable parameter, namely the z-coordinate, for each neuron. The z-coordinates are initialized layerwise in the range $[l, l + 1)$ for layer l. We train a relaxed spatially embedded MLP with 2,048 neurons, referred to as Relaxed 3D MLP, for 300 epochs with learning rate 0.005. The model achieves a test accuracy of 0.9337 ± 0.0040, representing an improvement over 3D MLP. However, it is still outperformed by MLP Baseline 17, which has approximately the same number of parameters but only 17 hidden neurons (see Table 3 and Fig. 4a). Figure 4b shows the distribution of the learned z-coordinates for the Relaxed 3D MLP. Notably, the z-distributions of all layers exhibit significant overlap.



(a) (b)

Fig. 4: (a)Accuracy comparison of the Relaxed 3D MLPs with learned z-coordinates, the 3D MLP, and the baseline MLPs (with 17 and 2,048 hidden neurons) on the MNIST [13] dataset. Models were trained using three different random seeds, with error bars representing the standard deviation across runs. (b) The distribution of the z-coordinates in the Relaxed 3D MLP after training.

3.4 Higher-Dimensional Embedding Spaces

From a graph-theoretic perspective, increasing the dimensionality of the embedding space provides additional degrees of freedom. Although this departs from the motivation to mimic the brain's embedding in 3D space, higher-dimensional spaces allow neurons to form more complex geometric relationships. We embedded models in 5-, 8-, 16-, and 32-dimensional Euclidean spaces, each with 2,048 hidden neurons. Increasing the dimensionality also increases the number of parameters, so we compared the models' performance to MLP baselines with similar parameter counts with 20, 32, 60, and 117 hidden neurons, respectively. Notably, the models embedded in higher-dimensional spaces still have fewer parameters than the baseline with the same number of neurons. Although increasing the number of dimensions of the embedding space improves performance, the spatially embedded models are still outperformed by their conventional MLP counterparts (see Fig. 5).

4 Discussion

Spatially embedded models demonstrated competitive performance. Notably, the spatially embedded SNN outperformed the spiking baseline model with a comparable parameter count, indicating a more efficient parameter utilization. This particularly strong performance of the spatially embedded SNN likely stems from its ability to leverage temporal dynamics for information encoding. This allows the model to compensate for the reduced weight flexibility through precise spike timing. In contrast, MLPs rely primarily on weight adjustments and may face challenges with restricted connectivity.

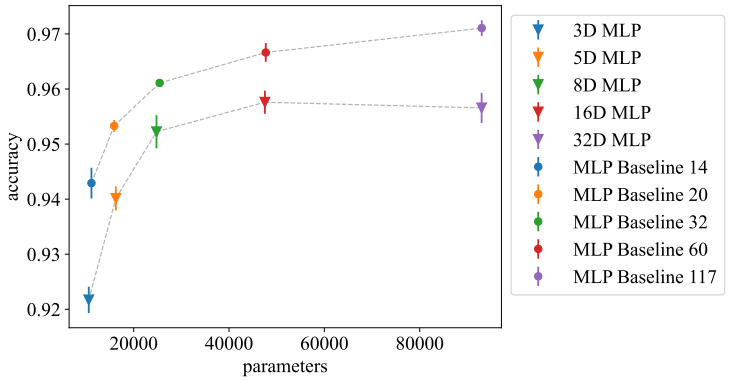
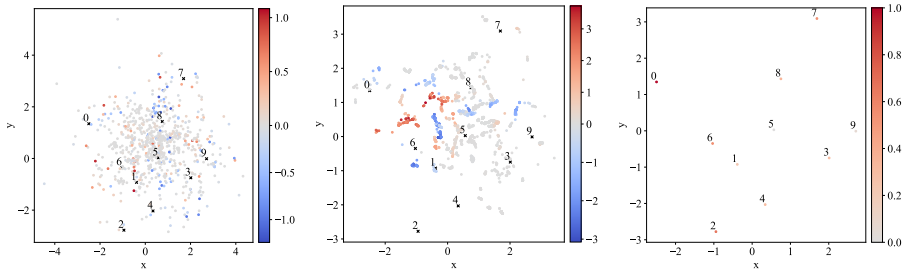


Fig. 5: Accuracy comparison of MLPs with 2,048 hidden neurons embedded in 3-, 5-, 8-, 16-, and 32-dimensional space, and the baseline MLPs with 14, 20, 32, 60, and 117 hidden neurons on the MNIST [13] dataset. Models were trained using three different random seeds, with error bars representing the standard deviation across runs.

Omitting the spatial layer structure improved performance, likely because it allowed the model to strengthen connections between specific neurons by moving them closer together. The significant overlap of z-coordinate distributions across layers confirms that the models leveraged this additional flexibility. Increasing the dimensionality of the embedding space improved performance, while the embedded models maintained a lower parameter count compared to conventional models with the same number of neurons. The performance plateaued beyond 16 dimensions, indicating that only a limited number of dimensions are required to express the weight configurations necessary for complex computations.

Spatially embedded models further demonstrated robustness to pruning. At sparsity levels above 60%, they outperformed baselines with similar parameter counts under the same conditions. When pruned after training with an extreme pruning percentage of 95%, the 3D MLP even outperformed the baseline with the same number of connection weights but significantly more parameters. Notably, when pruning 80% of the connections during training, the performance of the 3D MLP slightly improved again, nearly matching the accuracy of the unpruned model. These findings can be explained by the distance-dependent weight computation. Since distances cannot be infinite, the spatial embedding inherently prevents connections from having zero weights. As a result, each neuron receives input from all neurons in the preceding layer, even if some of these inputs are noisy or irrelevant. Pruning the smallest weights effectively enables zero-weighted connections, eliminating those that would otherwise introduce noise. Additionally, pruning reduces the dependencies between connection weights imposed by spatial embedding. Since modifying the position of a single neuron affects all its associated connection weights, removing less critical connections allows neurons to adjust positions more freely with fewer unintended side effects.



(a) Input Neurons (b) Hidden Neurons (c) Output Neurons

Fig. 6: The final neuron positions of a spatially embedded MLP with three layers. The color indicates the strength of the neurons' activation in response to the average input image for digit 0. Blue indicates inhibitory activation while red indicates excitatory activation. The positions of the output neurons are also included in the visualizations of the input and hidden layer as reference points.

Making neuron positions learnable and deriving connection weights from their distances establishes a direct link between geometry and connectivity. As a result, spatial constraints are naturally integrated into the architecture itself. Similarly to neural circuits in the brain, which evolve under pressure to minimize wiring length [9], the proposed model inherently favors spatially compact connectivity patterns by assigning stronger weights to connections between nearby neurons. Moreover, by structuring neurons in 2D layers, the spatial embedding framework offers an intuitive way of visualizing the spatial arrangement of neurons and their activations, as illustrated in Fig. 6. These visualizations help explain why the model makes certain decisions and identify neurons that are essential for particular features. Beyond interpretability, this framework is also useful for debugging. Outlier neurons positioned far from input or output clusters may indicate underutilized components, enabling targeted architectural adjustments.

Although spatially embedded MLPs demonstrated competitive performance, they were consistently outperformed by conventional MLPs, even when compared to baseline models with a similar number of parameters but significantly fewer neurons. This performance gap can be attributed to the dependencies between connection weights. Since the connection weights are determined by distance, modifying the position of a single neuron affects all connection weights linked to that neuron. This interdependence restricts the range of possible weight configurations, as adapting individual connection weights is not possible. As a consequence, the spatial embedding limits the model's expressiveness and likely prevents spatially embedded MLPs from outperforming traditional models when evaluated on the MNIST dataset. However, the spatial embedding can be a useful constraint in other specific tasks and future research is needed to investigate the effectiveness of spatially embedded models on more diverse datasets.

5 Conclusion

Motivated by the need for resource-efficient models with a low memory footprint and to bridge the gap between artificial and biological neural systems, we proposed a neural architecture in which neurons are embedded in three-dimensional Euclidean space, mimicking the spatial embedding of the brain. By optimizing neuron positions rather than individual connection weights, we reduced the model's parameter complexity from $O(n^2)$ to $O(n)$, where n is the number of neurons in the model. Spatially embedded models achieved competitive performance compared to baseline MLPs and even outperformed the baseline SNN with the same number of parameters. Increasing the embedding dimensionality or relaxing the spatial layer hierarchy further improved performance. Moreover, the spatially embedded architecture demonstrated robustness to pruning and maintained accuracy even when 80% of the weights were pruned during training – a valuable property for deployment on resource-constrained devices. Beyond efficiency, visualizing neuron positions and their activations offers deeper insights into the inner workings of spatially embedded neural networks.

The current experimental validation is limited to shallow networks trained on the MNIST dataset. To assess the scalability of the spatial embedding framework and to evaluate its performance in more challenging scenarios, future work should investigate its application to deeper architectures and more complex, diverse datasets. Moreover, a promising direction for future research is to explore more flexible spatial embeddings that better emulate biological neural systems. For example, relaxing rigid layer-wise connectivity structure and allowing the network to learn its connectivity could provide greater flexibility. Introducing sparse long-range connections, similar to the brain's heavy-tailed degree distributions, could further increase model expressiveness while maintaining efficiency through structural constraints. Furthermore, since the application to SNNs proved to be successful, the spatial embedding approach may also be well-suited for other neural architectures, such as Graph Neural Networks (GNNs), where input data often naturally exhibit spatial or topological structure. In summary, this work established spatially embedded neural networks as a novel framework for designing compact and biologically inspired AI systems.

Disclosure of Interests The authors have no competing interests to declare that are relevant to the content of this article.

Acknowledgment The idea for this work arose from a collaboration at the Capocaccia Cognitive Neuromorphic Engineering Workshop 2024.

References

1. Achterberg, J., Akarca, D., Strouse, D.J., Duncan, J., Astle, D.E.: Spatially embedded recurrent neural networks reveal widespread links between structural and functional neuroscience findings. Nature Machine Intelligence **5**(12), 1369–1381 (2023). https://doi.org/10.1038/s42256-023-00748-9

2. Barabási, D.L., Beynon, T., Katona, A., Perez-Nieves, N.: Complex computation from developmental priors. Nature Communications **14**(1) (2023). https://doi.org/10.1038/s41467-023-37980-1
3. Barabási, D.L., Schuhknecht, G.F.P., Engert, F.: Functional neuronal circuits emerge in the absence of developmental activity. Nature Communications **15**(1), 364 (2024). https://doi.org/10.1038/s41467-023-44681-2
4. Bartolozzi, C., Indiveri, G., Donati, E.: Embodied neuromorphic intelligence. Nature Communications **13** (2022). https://doi.org/10.1038/s41467-022-28487-2
5. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., Pascanu, R.: Relational inductive biases, deep learning, and graph networks (2018)
6. Blauch, N.M., Behrmann, M., Plaut, D.C.: A connectivity-constrained computational account of topographic organization in primate high-level visual cortex. Proceedings of the National Academy of Sciences **119**(3), e2112566119 (2022). https://doi.org/10.1073/pnas.2112566119
7. Boccato, T., Ferrante, M., Duggento, A., Toschi, N.: 4ward: A relayering strategy for efficient training of arbitrarily complex directed acyclic graphs. Neurocomputing **568**, 127058 (2024). https://doi.org/10.1016/j.neucom.2023.127058
8. Boccato, T., Ferrante, M., Duggento, A., Toschi, N.: Beyond multilayer perceptrons: Investigating complex topologies in neural networks. Neural Networks **171**, 215–228 (2024). https://doi.org/10.1016/j.neunet.2023.12.012
9. Cherniak, C.: Component placement optimization in the brain. The Journal of Neuroscience **14**(4) (1994). https://doi.org/10.1523/JNEUROSCI.14-04-02418.1994
10. Eshraghian, J.K., Ward, M., Neftci, E.O., Wang, X., Lenz, G., Dwivedi, G., Bennamoun, M., Jeong, D.S., Lu, W.D.: Training spiking neural networks using lessons from deep learning. Proceedings of the IEEE **111**(9), 1016–1054 (2023). https://doi.org/10.1109/JPROC.2023.3308088
11. Jacobs, R.A., Jordan, M.I.: Computational consequences of a bias toward short connections. Journal of Cognitive Neuroscience **4**(4), 323–336 (1992). https://doi.org/10.1162/jocn.1992.4.4.323
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017)
13. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2323 (1998). https://doi.org/10.1109/5.726791
14. Liu, S., Lin, Y., Zhou, Z., Nan, K., Liu, H., Du, J.: On-demand deep model compression for mobile devices: A usage-driven model selection framework. In: Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services. p. 389–400 (2018). https://doi.org/10.1145/3210240.3210337
15. Shuvaev, S., Lachi, D., Koulakov, A., Zador, A.: Encoding innate ability through a genomic bottleneck. Proceedings of the National Academy of Sciences **121**(38) (2024). https://doi.org/10.1073/pnas.2409160121
16. Stiso, J., Bassett, D.S.: Spatial embedding imposes constraints on neuronal network architectures. Trends in Cognitive Sciences **22**(12), 1127–1142 (2018). https://doi.org/10.1016/j.tics.2018.09.007