Dynamic neural network with matrix-extended residual connections*

Szymon Świderski[a] and Agnieszka Jastrzębska[a,b]

[a]Warsaw University of Technology, Warsaw, Poland
[b]The John Paul II Catholic University of Lublin, Poland

Abstract. The issue of adjusting neural network structure is one of the core problems in artificial intelligence. The issue of adjusting neural network structure is one of the core problems in artificial intelligence. A highly desirable scenario is a dynamic architecture that evolves structurally during the training process. In this paper, we propose a new and powerful tool that facilitates dynamic changes in network structure. We introduce a novel form of residual connections based on matrix extensions, enabling adaptable weight matrices and enhancing structural flexibility. The approach enhance the potential for structural modifications. We conducted a series of comprehensive experiments confirming that the new residual connections scheme behaves very well. The new type of connection improves performance by enabling better error flow during the error backpropagation phase, resulting in more efficient training. Our method demonstrates superior performance and enhanced trackability during the training process. The paper is supplemented by Python source code to ensure reproducibility. This method marks a significant starting point, showing immense potential for more advanced dynamic neural network models and transfer learning with dynamic models.

Keywords: dynamic neural network · changing architecture · training · shrinking · growing · residual connections with matrix extensions · self-changing neural networks.

1 Introduction

The architecture of a neural network plays a critical role in determining its efficiency and overall performance. Designing an effective architecture requires making decisions about the number of layers, their connections, and other hyperparameters. While many networks rely on fixed structures, our focus is on models that can modify their architecture during training—a challenging task that requires innovative solutions. To support this dynamic adaptation, we introduce a new way of connecting layers, called *matrix-extended residual connections*. This method significantly improves the process of dynamically modifying network structures during training.

* The research was funded by the National Science Centre, Poland, grant number 2024/53/B/ST6/00021.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14

Inspired by the mechanism of *skip connections* present in ResNet [6], this method has been designed to better accommodate networks with evolving structures. While the primary goal is to improve the performance of dynamically changing architectures, these connections are also valuable for static architectures, providing broader applicability. They enhance the traceability and interpretability of the learning process, offering deeper insights into model behavior.

The proposed modification to the classical skip connections is particularly effective for dynamic neural networks, where the structure evolves during training. The newly designed connections help preserve learned information while seamlessly integrating new layers, essential for maintaining stability and performance in such networks.

In our previous work, titled "Dynamic Growing and Shrinking of Neural Networks with Monte Carlo Tree Search" [14], we presented a basic method for modifying network architecture during training. That approach used standard skip connections, which performed as initially described in the ResNet paper [6]. In this paper, we introduce a new way of connecting layers that significantly enhances the efficiency of dynamic structural changes. Beyond dynamic networks, these connections show promise for applications such as transfer learning, where preserving and extending learned information is critical. The proposed design for residual connections handling enables seamless architecture expansion without any loss of previously learned knowledge, offering a powerful tool for both dynamic and static neural networks.

To support reproducibility and practical use, we have released our implementation as an open-source package named `growingnn`. The package is available on PyPi at https://pypi.org/project/growingnn/, where all relevant links, including the GitHub repository with the source code and documentation, can be found. This tool aims to simplify the development and experimentation of dynamic neural networks for researchers and practitioners.

2 Literature survey

The design of neural network architectures has always been challenging for developers. This led to the development of many new approaches over the years. Two key papers were particularly influential for this work, namely "Deep Residual Learning for Image Recognition" [6] and "Dynamic Growing and Shrinking of Neural Networks with Monte Carlo Tree Search" [14]. The first paper introduced the concept of skip connections, also known as residual connections, in neural networks.

The second paper, published by our team, presented an algorithm that allows dynamic modification of the network structure during training in a highly efficient manner. By integrating the best aspects of these two approaches, we developed a powerful new method that introduces exciting possibilities in the field of growing neural networks.

One of the most well-known methods for dynamically changing network structures is GradMax [3]. This method can adjust the architecture during training

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14

without losing previously learned data. While the general concept is similar to our approach, the procedures for modifying the network architecture differ significantly. In GradMax, changes are based on SVD (Singular Value Decomposition). This approach ensures that new neurons do not interfere with existing knowledge. In contrast, our method uses specialized connections, referred to as residual connections with matrix extensions.

A recent paper, "Dynamic Neural Network Structure: A Review of Its Theories and Applications" [5], provides a detailed overview of dynamic neural network structures. This paper describes a taxonomy and associated naming convention for dynamic neural networks. The group most relevant to our approach is the *adaptive layer* group, within which three main types of methods exist.

The first type of an adaptive layer method is called *stacking layer*, where the architecture is modified by adding layers sequentially [12]. The second type is called *residual approximation*. It originates from ResNet [6]. We can classify our new contribution to this subgroup. They adjust residual block connections to alter the network structure dynamically. The third type, *shortcut connections*, originates from the ResNeXt [17] architecture. These building blocks are more complex, with an emphasis on connections that bypass multiple layers. The paper "Going Deeper with Convolutions" [15] introduced a method that enables support for very deep architectures with residual-like connections. However, this approach is limited by concatenation layers, which combine inputs from various layers into a unified structure. Our approach addresses these limitations by introducing new types of connections. Additionally, shortcut-based structures often rely on predefined building blocks that restrict their ability to expand flexibly.

We believe that the methods belonging to the adaptive layer group [5] are all effectively unified in our algorithm. This highlights the significance of our work, as it provides a straightforward and efficient way to integrate these ideas. In our algorithm, we generate a set of actions in each iteration. These actions allow us to add sequential or residual layers, encompassing the capabilities of all the methods described in the adaptive layer group. The method presented in this paper is not limited to building blocks or any type of concatenate layer. The new way of residual connections allows to easily create new connections without any data loss on the connection.

In the field of dynamic neural networks, two key approaches focus on optimizing network structure. The first is reinforcement learning, which has demonstrated auspicious results [18]. The core idea behind this approach involves evaluating and grading structural changes to determine the most optimal adjustments. The second approach is evolutionary algorithms. Neuroevolution [13] is an approach to developing neural networks by leveraging evolutionary algorithms, optimizing not just the weights but also the architectures, hyperparameters, activation functions, and even learning rules of neural networks.

It is also worth mentioning papers that present non-standard, interesting ideas, such as the Cascade-Correlation Architecture [4], which adds new neurons for each unrecognized pattern; the work by Kilcher et al. [8], which explores escaping flat regions through structural modifications; and the Convex Neural

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14

Networks [1], which demonstrates how neural networks can adapt to various linear structures by adding neurons to a single hidden layer at each step.

3 The method

The described algorithm is based on Stochastic Gradient Descent (SGD) and Adam optimizer [10]. It operates with a model represented as a directed acyclic graph of layers. Each layer is an independent node that manages its own incoming and outgoing connections to other layers. In this algorithm, the training procedure is divided into generations and then further into epochs. One epoch consists of one forward and backward propagation.

A generation consists of a training phase with multiple epochs, followed by a structure modification phase. After training, all possible changes to the current structure are generated. We refer to these changes as actions in this paper. The simulation provides information on which action is the best, and the best action is then applied to the structure. The overall flow of our method is in Figure 1.

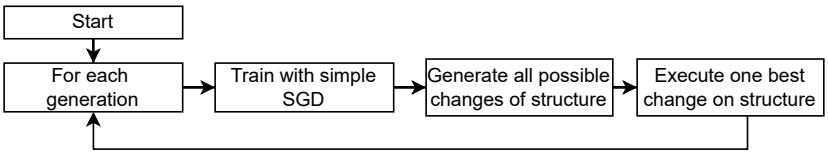


Fig. 1: Block diagram showing the flow of the method addressed in this study.

We modify the architecture using three types of actions. The first type involves adding a sequential layer between two directly connected layers. The second type adds a layer with residual connections; a layer is added between two not directly connected layers through skip connections. The third type consists of actions that remove an existing layer. In each generation, the algorithm generates all possible actions for the current state and evaluates them using a Monte Carlo simulation to determine and execute the best action.

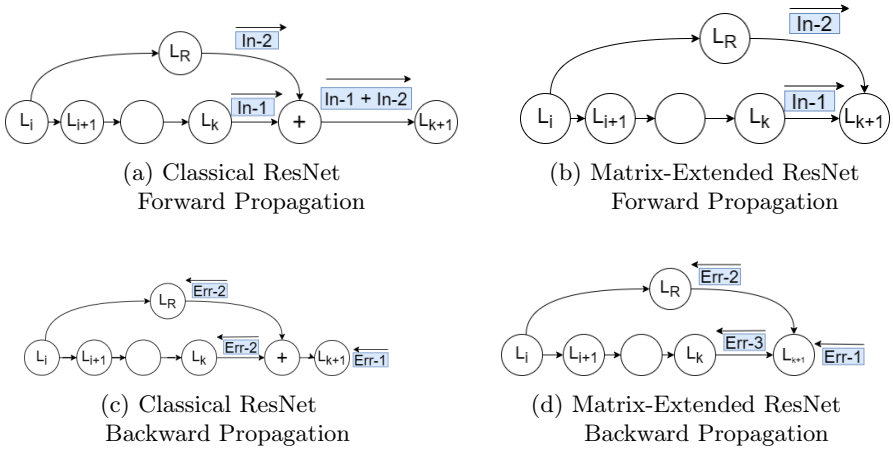
The graph starts from the smallest possible setup: an input and output layer linked by a single connection and evolves over generations by adding or removing layers. Architecture changes in a given generation only if there is no improvement in accuracy, which means that if a given structure is capable of learning a given dataset, training continues without a change in the structure.

3.1 Data flow in dynamic neural structure

Residual connections in deep learning bring a significant change in training capabilities, but these solutions have their own limitations. If we analyze the data flow in residual connections [6], we can see that building blocks are limited in

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14

terms of learning capabilities. The error that is backpropagated is partially lost due to the building block structure. Data flowing through the building block at some point needs to go through a summation phase to combine outputs from two different layers into one.



(a) Classical ResNet Forward Propagation

(b) Matrix-Extended ResNet Forward Propagation

(c) Classical ResNet Backward Propagation

(d) Matrix-Extended ResNet Backward Propagation

Fig. 2: Comparison of data flow in forward and backward propagation in Classical ResNet and Matrix-Extended ResNet.

The building block structure in ResNet consists of one residual (skip) connection that skips a given number of layers. We will use examples illustrated in Figure 2 to present the limitations of classical ResNet connections. In those examples, layers $l_i, l_{i+1}, \ldots, l_{k+1}$ are connected sequentially, with each l_i connected to l_{i+1}. The skip connection is connected from l_i to an additional layer l_R and from l_R to l_{k+1}.

The signal flow is depicted by blue rectangles labeled with In and arrows, as shown in Figure 2. During the forward propagation phase, the output from layer L_i is forwarded to two subsequent layers: one via a skip connection and the other sequentially, as illustrated in Figure 2a. After this step, data flows through all layers without loss, except at layer l_{k+1}. In this layer, inputs from l_k and l_R are combined through summation, expressed as $In1 + In2$. Although this input combination is valid for layer l_{k+1}, we hypothesize that a significant portion of the interpretable data for l_{k+1} is lost. In our proposed approach, referred to as the *residual matrix extended connection* (Figure 2b), we omit the summation step. Instead, the matrices are integrated to generate a new input signal for layer l_{k+1}, as visualized in Equation 3.

In the back-propagation phase for the classical ResNet, layer l_{k+1} sends the same error to both l_R and l_{k+1}. When both layers receive the same error, their potential to detect distinct features is limited. While this approach effectively addresses exploding or vanishing gradients, allowing the network to have a much

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14

deeper structure, we aim to retain these advantages while enhancing efficiency. Our method does not backpropagate the same error to all connected layers. After extending the matrix for the new connection, we know which part belongs to each layer and send only the error assigned to that part. The layer that backpropagates the error calculates it individually for each connected layer, based on the corresponding part of the main weight matrix, as shown in Equation 3.

3.2 Matrix-extended residual connections

New residual connections with matrix extension are based on reshaping matrices. This helps the neural network adjust to the new connections without losing information.

In the first epoch, after adding a new connection, the neural network will return exactly the same output, but the number of all trainable weights will increase. This means that the layer will not forget any data it has already learned, but it will have a larger number of weights that are initially set to zero.

– **Before adding new connection**:

$$\mathbf{z}^{[l]} = \mathbf{W}^{[l]}\mathbf{a}^{[0]} + \mathbf{b}^{[l]} \tag{1}$$

– **After adding n connections with classical ResNet approach**:

$$\mathbf{z}^{[l]} = \mathbf{W}^{[l]} \sum_{i=0}^{n} \mathbf{a}^{[i]} + \mathbf{b}^{[l]} \tag{2}$$

– **After adding n connections with Matrix-extended ResNet**:

$$\mathbf{z}^{[l]} = \begin{bmatrix} W^{[l]} \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} a^{[0]} \cdots a^{[n]} \end{bmatrix} + \mathbf{b}^{[l]} \tag{3}$$

In the presented formulas, $\mathbf{a}$ represents the data forwarded from the preceding layer, $\mathbf{W}$ is the weight matrix, and $\mathbf{b}$ is the bias. $\mathbf{z}^{[l]}$ denotes the output of a layer l, which is subsequently processed by an activation function and passed to the next layers. To handle multiple input signals without data loss, we extend the weight matrix by adding rows filled with zero values and combining all incoming signals into a single matrix in a column-wise manner. The output remains the same as before during the first epoch after introducing this change (i.e., after adding the new connections). However, in subsequent backpropagation steps, these newly added zero weights become trainable, enabling additional flexibility for the layer. This approach allows us to add or remove a specific number of layers without losing data or functionality.

With our approach, the backpropagated error through multiple residual connections does not need to be duplicated. Instead, only the error related to the

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14

specific part of the weight matrix corresponding to a given layer's input is propagated. This ensures that each layer receives a backpropagated error that is directly calculated for it.

Our new approach can be interpreted as increasing the number of connections but not the number of neurons, which brings a new field of possibilities to the algorithms that change the structure while training. Adding those connections can not only bring changes to the structure but also increase the number of trainable weights inside the layer without causing any loss in network memory.

3.3 Trackability of neural network learning process

We extend traditional residual connections with a novel approach called matrix-extended residual connections. This method offers improved traceability by distributing the back-propagated error more effectively. Our approach ensures that each layer receives a unique error signal that is tailored specifically to it. This allows for a more precise analysis of the error propagation and enables a deeper understanding of the learning process. This brings a significant change in our algorithm since the structure can grow to extensive and complicated graphs. Under the assumption that each layer analyzes a distinct set of features, our method facilitates a clearer interpretation of learning dynamics.

Figure 3 illustrates the evolution of the network structure on the CIFAR-10 dataset. Each dot represents a single layer. The color of the dot indicates the error propagated through that layer: red corresponds to the highest propagated error, while blue indicates the smallest error. The error for each layer is calculated and normalized according to the following formula: $\min\left(\max\left(\frac{\sum|E|}{E_{\max}}, 0\right), 1\right)$, where E is the error received in a given epoch by a given layer. In the presented images, we only show the state from the last epoch of a given generation, just before the change in structure, and the maximum error $E_{\max}$ is defined as $E_{\max} = \prod(\text{shape of } E)$.

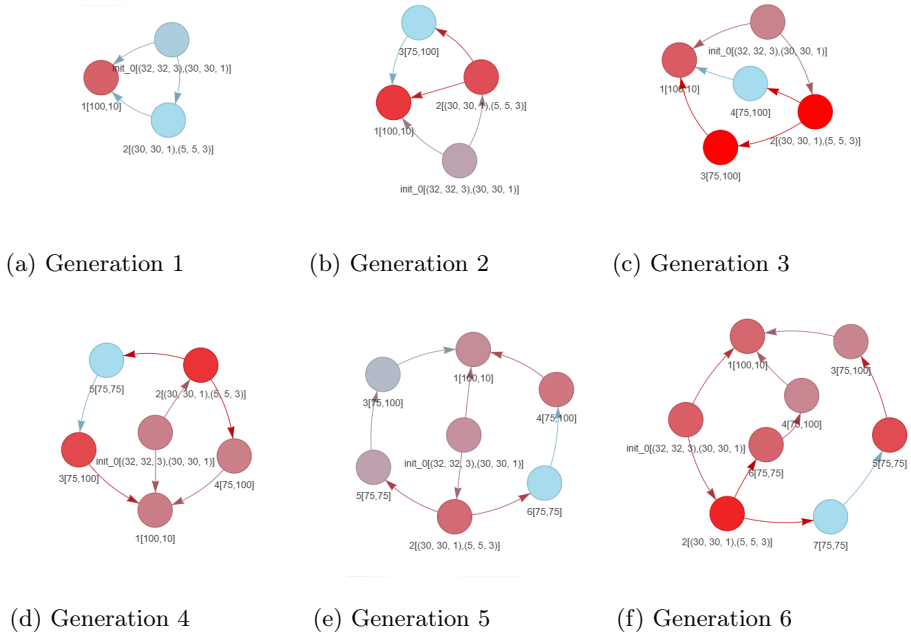
After analyzing how the error evolves during training and structural changes, we can apply methods such as MDA [7] to visualize and understand the features learned by each layer. This approach provides a deeper insight into the overall learning process of the neural network.

4 Empirical analysis

4.1 Dataset and empirical setup

In the conducted experiments, we used three widely recognized datasets: MNIST [2], Fashion-MNIST (FMNIST) [16], and CIFAR-10 [11]. MNIST contains handwritten digits. FMNIST consists of images of various clothing items, providing a more complex challenge. CIFAR-10 includes images of objects from 10 different classes, such as animals and vehicles, offering a more diverse and challenging dataset.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14



(a) Generation 1 (b) Generation 2 (c) Generation 3

(d) Generation 4 (e) Generation 5 (f) Generation 6

Fig. 3: History of structural changes during CIFAR-10 training. Error magnitudes are shown in red (high) and blue (low). Classical ResNet connections were used.

We conducted a series of experiments to validate the three hypotheses outlined in Section 4.2. To encourage structural growth, each experiment started with a minimal configuration: a single 3×3 convolutional input layer and a dense output layer with 10 neurons. For MNIST and FMNIST, we used three generations with 10 epochs per generation and a hidden size of 50; for CIFAR-10, six generations with 50 epochs per generation and a hidden size of 100. All convolutional layers used 3×3 kernels, ReLU activations in hidden layers, and a sigmoid output. Training used a batch size of 128 and multiclass cross-entropy loss. A simulation scheduler guided the search, with each simulation running for 60 minutes to find the best action. Proposed actions were graded via 20-epoch training runs, using 20 random samples per class. The simulation algorithm was a modified Monte Carlo Tree Search (MCTS), optimized for efficient action space exploration.

In our experiments, we partitioned each dataset into training and testing subsets, with the testing set comprising 20% of the data. We ensured an even distribution of images per class using stratified sampling via the *train_test_split* function from the sklearn library. Consequently, for MNIST, the training set contained 48,000 images, and the testing set had 12,000 images. Fashion MNIST was

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14

divided similarly, with 48,000 training and 12,000 testing images. For CIFAR-10, which consists of 60,000 images in total, we allocated 40,000 images for training and 10,000 for testing.

4.2 Classification quality of the new approach

The conducted experiments aimed to empirically validate three hypotheses concerning the proposed residual connection mechanism. We compare it with the classical residual connections from the ResNet model.

RH1 The first hypothesis states that dynamically modifiable residual connections with matrix extensions tend to guide the weight distribution in the network toward a normal distribution.

RH2 The second hypothesis states that residual connections with matrix extensions improve learning capabilities by allowing the error signal to propagate more effectively through the network.

RH3 The third hypothesis states that dynamically modifiable residual connections with matrix extensions perform better than traditional residual connections and are particularly effective in networks that dynamically change their architecture during training.

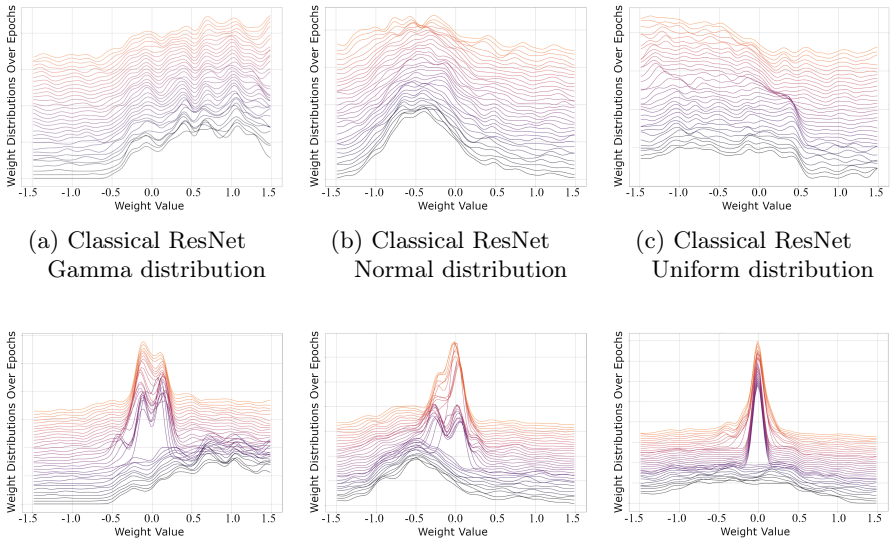
The first hypothesis (**RH1**) proposes that dynamically modifiable residual connections with matrix extensions guide the weight distribution toward a normal distribution. In this experiment, we initialized neural networks with three types of general weight distributions: uniform, Gaussian, and gamma. After this, we run the experiments with our algorithm, which modifies the network structure.

Table 1: Comparison of mean values for three repetitions across three datasets and distribution types between classical ResNet and matrix-extended ResNet, based on skewness and kurtosis of the weight distribution in the final epoch.

Dataset	**Distribution Type**	**Classical ResNet**		**Matrix-extended ResNet**	
		Skewness	**Kurtosis**	**Skewness**	**Kurtosis**
CIFAR-10	Gamma	0.11	-1.48	3.67	12.84
	Normal	0.34	-0.86	0.85	0.31
	Uniform	0.26	-1.32	1.45	2.25
FMNIST	Gamma	0.20	-1.33	3.60	12.74
	Normal	0.58	-0.90	0.51	-0.70
	Uniform	-0.08	-1.31	1.02	1.13
MNIST	Gamma	0.38	-1.12	0.45	-0.47
	Normal	0.34	-1.08	2.06	4.39
	Uniform	-0.06	-1.48	1.70	4.89
Mean		0.23	-1.21	1.70	4.15

The graphs in Figure 4 illustrate how weight distribution changes over the course of training. Each line in a graph represents the weight distribution for an

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14



(a) Classical ResNet Gamma distribution

(b) Classical ResNet Normal distribution

(c) Classical ResNet Uniform distribution

(d) Matrix-extended ResNet Gamma distribution

(e) Matrix-extended ResNet Normal distribution

(f) Matrix-extended ResNet Uniform distribution

Fig. 4: Comparison of weight distribution changes throughout the neural network structure between the classical ResNet and the matrix-extended ResNet connections on MNIST dataset.

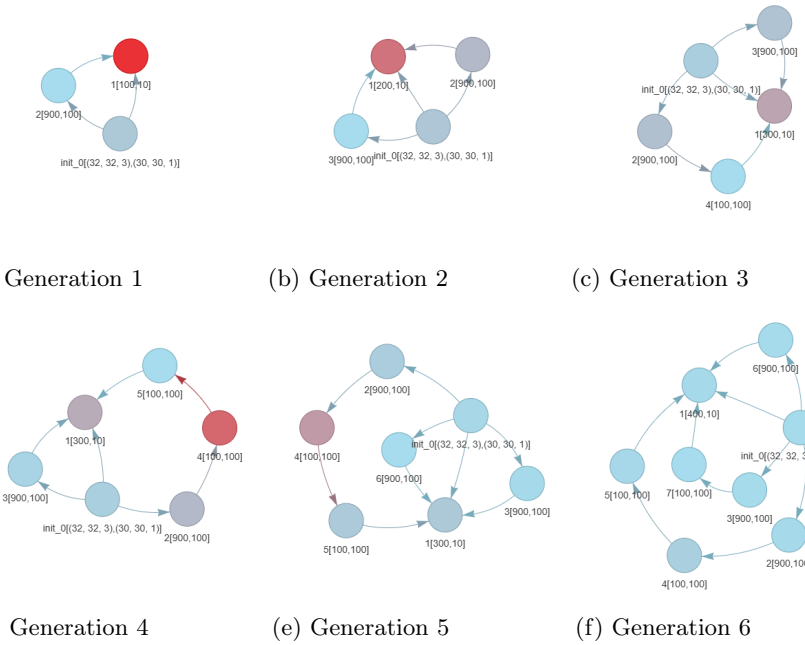
epoch. The earliest epoch is displayed at the bottom, and the final distribution is shown at the top. Our analysis focuses on the input and output layers, as they consistently reflect changes in the overall weight distribution of the network.

As illustrated in Figure 4, the proposed method consistently exhibits spikes near zero in the weight distributions, with the number of weights close to zero increasing over the course of training. In comparison, the classical ResNet approach shows minimal variation in its weight distribution across epochs. Table 1 presents the average distribution metrics across all seeds. While both methods display some asymmetry, the proposed approach yields a mean kurtosis of 4.15, which is closer to the value of a normal distribution [9]. In contrast, the classical method has an average kurtosis of -1.21, indicating a significantly flatter distribution. These findings suggest that our method produces distributions that more closely resemble normality, a result influenced by the growing number of zero weights introduced after each structural modification.

The second hypothesis (**RH2**) states that residual connections with matrix extensions improve learning capabilities by propagating the error signal more effectively through the network. During each epoch, we calculated a normalized error for every layer. The formula for calculating this normalized error is detailed in Section 3.3. This error does not directly reflect the overall performance of

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14

the network but rather indicates the magnitude of the error passing through a specific layer.



(a) Generation 1 (b) Generation 2 (c) Generation 3

(d) Generation 4 (e) Generation 5 (f) Generation 6

Fig. 5: History of structural changes during CIFAR-10 training. Error magnitudes are shown in red (high) and blue (low). Matrix-extended ResNet connections were used.

Figure 5 illustrates the learning process in the proposed solution, where the error stabilizes more effectively, as indicated by the predominance of blue nodes representing low error. In contrast, Figure 3 shows the traditional approach, showing a higher overall error between layers. In the standard ResNet architecture in Figure 3, each layer transmits identical error signals to all connected layers via residual connections. In contrast, our approach utilizes a matrix-extended connection in Figure 5, allowing for layer-specific error propagation. In the conventional approach, when a high error occurs in one layer, it propagates through the residual connections to all subsequent layers, causing a cascade of high errors across many layers. This results in inefficient error transmission, making it harder for the network to learn effectively. This is supported by Table 2, which shows that the mean and variance of errors across all layers are lower for the matrix-extended ResNet connections.

The third hypothesis (**RH3**) states that dynamically modifiable residual connections with matrix extensions outperform traditional residual connections and

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14

Table 2: Mean and variance for normalized error distribution across the whole neural network structure.

Dataset	Seed	Classical ResNet		Matrix-extended ResNet	
		Mean	Variance	Mean	Variance
CIFAR-10	0	0.485	0.058	0.007	0.000
	1	0.199	0.010	0.000	0.000
	2	0.011	0.000	0.074	0.014
FMNIST	0	0.081	0.003	0.029	0.001
	1	0.056	0.003	0.052	0.004
	2	0.079	0.016	0.022	0.001
MNIST	0	0.027	0.001	0.009	0.000
	1	0.013	0.000	0.011	0.000
	2	0.031	0.001	0.007	0.000
Mean		0.109	0.010	0.024	0.002

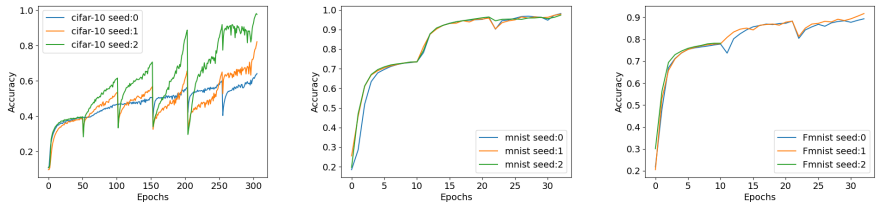
are particularly effective in networks that change their architecture dynamically during training. The experiments for the first and second hypotheses demonstrated the benefits of our method. To further support this, we compared the accuracy of a model trained with matrix-extended connections to that of a classical ResNet with skip connections.

Table 3: Comparison of training accuracy between the classical ResNet model and the model with matrix-extended connections.

Dataset	Seed	Classical ResNet	Matrix-extended ResNet
CIFAR-10	0	64.16%	99.96%
	1	82.20%	100.0%
	2	98.15%	98.13%
FMNIST	0	89.30%	88.19%
	1	91.68%	86.66%
	2	78.14%	90.75%
MNIST	0	98.14%	97.83%
	1	97.84%	98.31%
	2	97.36%	98.06%
Mean		88.55%	95.32%

The most significant differences were observed with the CIFAR-10 dataset. In this case, the matrix-extended ResNet connections demonstrated superior learning capabilities, delivering better results. As shown in Table 3, our method generally led to better results. The training history, illustrated in Figures 6a and 6d, shows that each structural change in the classical approach caused considerable instability. In contrast, our method exhibited visible instability only in seed 0 of the third generation. In the matrix-extended connections, we observe several smooth spikes following each generation, which occur every 50 epochs. There is

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14



(a) Classical ResNet CIFAR-10 (b) Classical ResNet MNIST (c) Classical ResNet FMNIST

(d) Matrix-extended ResNet, CIFAR-10 (e) Matrix-extended ResNet, MNIST (f) Matrix-extended ResNet, FMNIST

Fig. 6: Comparison of training history with accuracy per epoch between the classical ResNet and the matrix-extended ResNet connections.

a significant improvement compared to the old method, indicating that the new approach has substantial potential for modifying the learning structure without losing any previously acquired data. The results shown in Table 3 indicate that the matrix-extended ResNet connections outperform the classical ResNet connections. The mean accuracy for the matrix-extended ResNet connections is 95.32%, compared to 88.55% for the classical ResNet connections. It is important to note that all training was conducted using our code written from scratch and that we employed basic tools for training, such as Stochastic Gradient Descent. These results are based on the training dataset, as our focus is on the training method rather than the model's performance on the test dataset. Our goal is to analyze the training procedure itself, not the test performance. To validate the method, we had to address specific features that are observable when a model is trained.

5 Conclusion

This paper presents a novel solution to common challenges in dynamic neural networks. We introduce matrix-extended ResNet connections, allowing structural changes without losing previously learned knowledge. These connections improve the efficiency of information flow by eliminating the summation step typically found in ResNet skip connections. This design enables easier and more

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14

flexible modification of the network structure during training, supporting dynamic adaptation without disrupting existing representations.

Moreover, our approach goes beyond simply addressing the summation step. Many other methods discussed in the literature also involve some form of merging, which has similar limitations. Our method simplifies this process by directly modifying the matrix, offering several advantages. No additional step is required to connect layers; the layer connections are straightforward and more closely resemble natural neural behaviors. The summation steps do not align with the foundational concept of artificial neurons, which was to emulate the behavior of natural neurons. In our approach, connecting a new layer allows neurons to establish additional connections that include the newly added layer. This opens new possibilities for algorithms that dynamically adjust their structure during training.

The conducted experiments validate the proposed thesis. Dynamically modifiable structures with matrix-extended residual connections enhance learning capabilities by providing better-adjusted error flow during backpropagation. Conducted experiments show that our method outperformed classical residual connections, achieving higher accuracy. When comparing the training history, it is clear that in the traditional approach, each change resulted in a visible drop in performance for the CIFAR-10 dataset. On the other hand, the training history for matrix-extended connections showed no negative impact in most changes, as there was no summation point or any additional step that could negatively affect the results. We also analyzed the error propagation through the network for both methods. In the classical approach, the summation step forces the error to be replicated, leading to significantly higher errors in most layers of the structure. In contrast, our solution maintains a low error throughout most of the structure, thereby making the learning process more efficient.

Additionally, our method offers a novel way to track the training process, enabling a more precise error propagation analysis and a deeper understanding of the learning dynamics. This is especially beneficial for large, complex networks.

All training code was written from scratch without additional frameworks, focusing on simple datasets. This demonstrates that effective and fast training can be achieved using only a CPU. The open-source implementation is available as a Python package named *growingnn* on PyPi: https://pypi.org/project/growingnn.

References

1. Bach, F.R.: Breaking the curse of dimensionality with convex neural networks. Journal of Machine Learning Research **18**, 1–53 (2017), https://jmlr.org/papers/volume18/14-546/14-546.pdf
2. Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012)
3. Evci, U., Vladymyrov, M., Unterthiner, T., van Merriënboer, B., Pedregosa, F.: GradMax: Growing neural networks using gradient information. In: Proc. of ICLR 2022 (2022), https://iclr.cc/virtual/2022/poster/7131

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14

4. Fahlman, S.E., Lebiere, C.: The Cascade-Correlation Learning Architecture, p. 524–532. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1990)
5. Guo, J., Chen, C.L.P., Liu, Z., Yang, X.: Dynamic neural network structure: A review for its theories and applications. IEEE Transactions on Neural Networks and Learning Systems pp. 1–21 (2024). https://doi.org/10.1109/TNNLS.2024.3377194
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
7. Islam, M.T., Zixia, Z., Ren, H., Badiei Khuzani, M., Kapp, D., Zou, J., Tian, L., Xing, L.: Revealing hidden patterns in deep neural network feature space continuum via manifold learning. Nature Communications **14** (12 2023). https://doi.org/10.1038/s41467-023-43958-w
8. Kilcher, Y., Bécigneul, G., Hofmann, T.: Escaping flat areas via function-preserving structural network modifications. In: Proc. of ICLR 2019 (2019), https://openreview.net/forum?id=H1eadi0cFQ
9. Kim, T.H., White, H.: On more robust estimation of skewness and kurtosis. Finance Research Letters **1**(1), 56–73 (2004). https://doi.org/https://doi.org/10.1016/S1544-6123(03)00003-5, https://www.sciencedirect.com/science/article/pii/S1544612303000035
10. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. International Conference on Learning Representations (12 2014)
11. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations (ICLR 2015). pp. 1–14. Computational and Biological Learning Society (2015)
13. Stanley, K., Clune, J., Lehman, J., Miikkulainen, R.: Designing neural networks through neuroevolution. Nature Machine Intelligence **1** (01 2019). https://doi.org/10.1038/s42256-018-0006-z
14. Świderski, S., Jastrzębska, A.: Dynamic growing and shrinking of neural networks with monte carlo tree search. In: Franco, L., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) Computational Science – ICCS 2024. pp. 362–377. Springer Nature Switzerland, Cham (2024)
15. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions . In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–9. IEEE Computer Society, Los Alamitos, CA, USA (Jun 2015). https://doi.org/10.1109/CVPR.2015.7298594, https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298594
16. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR **abs/1708.07747** (2017), http://arxiv.org/abs/1708.07747
17. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5987–5995 (2017). https://doi.org/10.1109/CVPR.2017.634
18. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8697–8710 (2018). https://doi.org/10.1109/CVPR.2018.00907

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_14