DOI: 10.15514/ISPRAS-2026-38(6)-30

On Activation Design for Improving Robustness and Effective Capacity in Neural Networks

M.A. Sentiurev, ORCID: 0000-0003-3311-6651 <michael_sentiurev@outlook.com>
I.A. Makarov, ORCID: 0000-0002-3308-8825 <iamakarov@hse.ru>

Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. Activation functions play a critical role in shaping both the robustness and effective capacity of neural networks, yet their impact can vary across architectural settings. This work investigates how activation design and simple training strategies influence model behavior under fixed parameter budgets, implemented in a custom C++ experimental framework. We consider two complementary approaches. First, we study non-monotonic gradient-preserving activation functions combined with an Equivalent Exchange (EqEx) training heuristic for fully connected networks. Second, we introduce a non-monotonic gradient-preserving activation variant aimed at improving training dynamics in convolutional architectures. Empirical evaluation on MNIST shows that EqEx with non-monotonic activations reduces the success rate of the evaluated targeted gradient-based input synthesis procedures by up to 37% for Carlini-Wagner L2 and 29% for the Basic Iterative Method, while maintaining comparable or slightly improved accuracy (up to +0.6%-0.8% accuracy improvement depending on configuration) under equal parameter budgets. On CIFAR-10, the proposed activation demonstrates improved optimization behavior, achieving higher accuracy (70.8%) compared to Leaky-ReLU (68.7%) and symmetric/mirrored activation variants (68.0%), approaching the performance of models with twice the number of channels (71.8%). These results suggest that different activation design principles are effective in different regimes: non-monotonicity combined with structured training improves robustness against the input synthesis in fully connected networks, while preserving gradient flow is more beneficial for convolutional architectures. Overall, activation design provides a lightweight mechanism for reducing the success rate of targeted input synthesis and improving effective capacity without increasing inference cost.

Keywords: activation functions; non-monotonic activations; Equivalent Exchange (EqEx); Immortal ReLU; robustness; input-synthesis resistance; gradient flow; convolutional neural networks

For citation: Sentiurev M.A., Makarov I.A. On Activation Design for Improving Robustness and Effective Capacity in Neural Networks. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 6, part 2, 2026, pp. 147-162. DOI: 10.15514/ISPRAS-2026-38(6)-30.

О разработке функций активации для повышения эффективной ёмкости и устойчивости нейронных сетей

М.А. Сентюрев, ORCID: 0000-0003-3311-6651 <michael_sentiurev@outlook.com>
И.А. Макаров, ORCID: 0000-0002-3308-8825 <iamakarov@hse.ru>

Институт системного программирования РАН им. В.П. Иванникова,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Функции активации играют критически важную роль в формировании как устойчивости, так и эффективной ёмкости нейронных сетей, однако их влияние может существенно различаться в зависимости от архитектурных условий. В данной работе исследуется, как дизайн функций активации и простые стратегии обучения влияют на поведение модели при фиксированном числе параметров, реализованные в рамках собственного экспериментального C++ фреймворка. Рассматриваются два взаимодополняющих подхода. Во-первых, изучаются немонотонные функции активации с сохранением градиента, объединённые с эвристикой обучения Equivalent Exchange (EqEx) для полносвязных сетей. Во-вторых, предлагается немонотонный вариант функции активации с сохранением градиента, направленный на улучшение динамики обучения в сверточных архитектурах. Эмпирическая оценка на наборе MNIST показывает, что EqEx в сочетании с немонотонными активациями снижает успешность процедур целевого синтеза входов на величину до 37% для Carlini-Wagner L2 и до 29% для Basic Iterative Method при сохранении сопоставимой или слегка улучшенной точности (до +0.6–0.8% в зависимости от конфигурации) при одинаковом бюджете параметров. На CIFAR-10 предложенная активация демонстрирует улучшенную динамику оптимизации, обеспечивая более высокую точность (70.8%) по сравнению с Leaky-ReLU (68.7%) и симметричными/зеркальными вариантами активаций (68.0%), приближаясь к результатам моделей с вдвое большим числом каналов (71.8%). Полученные результаты указывают, что различные принципы проектирования функций активации эффективны в разных режимах: немонотонность в сочетании со структурированным обучением улучшает устойчивость к исследованным атакам в полносвязных сетях, тогда как сохранение градиентного потока оказывается более важным для сверточных архитектур. В целом, дизайн функций активации представляет собой «лёгкий» механизм снижения успешности целевого синтеза входов и повышения эффективной ёмкости без увеличения вычислительных затрат на этапе инференса.

Ключевые слова: функции активации; немонотонные активации; обмен с эквивалентом; Бессмертный ReLU; устойчивость; устойчивость к синтезу входов; градиентный поток; сверточные нейронные сети

Для цитирования: Сентюрев М.А., Макаров И.А. О разработке функций активации для повышения эффективной ёмкости и устойчивости нейронных сетей. Труды ИСП РАН, 2026, том 38, вып. 6, часть 2, стр. 147–162 (на английском языке). DOI: 10.15514/ISPRAS–2026–38(6)–30.

1. Introduction

Neural networks are typically evaluated under a trade-off between predictive performance, robustness, and computational cost. In practical deployments, model size and inference cost are often fixed due to hardware or latency constraints, which limits the use of larger architectures as a solution to performance improvements. As a result, improvements must often come from more efficient design choices within a fixed parameter budget.

Activation functions play a critical role in determining gradient flow, effective capacity per unit, and training stability, yet they are often treated as a secondary design choice compared to architecture scaling. This raises the question of whether carefully designed activation functions, combined with training strategies, can improve both accuracy and robustness without increasing model complexity.

In this work, we investigate this question from two complementary perspectives. For fully connected networks, we introduce non-monotonic activation functions combined with Equivalent Exchange (EqEx), an algorithm for transferring weights between equivalent monotonic and non-monotonic network representations. In our experiments, EqEx is applied once during training (after an initial pre-training phase), allowing the model to benefit from stable monotonic optimization before transitioning to a non-monotonic representation. For convolutional networks, we focus on the role of gradient-

preserving ("immortal") activation functions, including their non-monotonic variants, applied directly without EqEx. In this setting, maintaining gradient flow is critical for stable optimization, and immortal activations improve learning dynamics compared to both standard ReLU and symmetric activation functions [1].

2. Related Work

There are several research directions that modify activation functions or training procedures to improve neural network performance, robustness, or optimization stability.

2.1 Activation function design

Zhao et al. [1] introduce symmetric activation functions that suppress unusually large internal activations, improving robustness against adversarial and noisy inputs while maintaining clean accuracy.

Misra [2] introduces Mish, a smooth non-monotonic activation function defined as $x \cdot \tanh(\mathrm{softplus}(x))$. Mish improves optimization stability and generalization, demonstrating that carefully designed non-monotonic activations can outperform ReLU-based variants on vision benchmarks.

He et al. [3] propose PReLU, a learnable variant of ReLU that adapts the negative slope during training. Combined with a rectifier-aware initialization scheme, their approach enables stable training of very deep networks and improves ImageNet classification performance.

Ramachandran et al. [4] propose a method for automatically searching activation functions, discovering Swish, a smooth non-monotonic activation that improves performance across deep architectures with minimal architectural modifications.

2.2 Network transformation and representation transfer

Chen et al. [5] propose Net2Net, a knowledge transfer method that accelerates training by transforming a trained network into a larger one while approximately preserving its function. This enables efficient initialization of deeper or wider models without retraining from scratch.

Wei et al. [6] introduce Network Morphism, a framework for continuously transforming neural network architectures while preserving functionality, enabling efficient reuse of learned representations across model variants.

2.3 Additional related approaches

Additional work explores adaptive, stochastic, and task-dependent activation functions, including attention-based nonlinearities and automated activation optimization methods [7-15].

2.4 Discussion

Prior work primarily focuses either on designing improved activation functions or on transforming network representations while preserving learned behavior. Activation-function research has explored learnable rectifiers, smooth non-monotonic nonlinearities, symmetry-based robustness mechanisms, and automatically discovered activation forms. Separately, representation-transfer methods such as Net2Net and Network Morphism investigate how trained networks can be transformed without restarting optimization from scratch.

In contrast, this work combines both directions. We study non-monotonic gradient-preserving activations together with Equivalent Exchange (EqEx), a procedure that transfers weights between monotonic and non-monotonic representations during training. Unlike architecture-morphism approaches, EqEx focuses on transitions between activation representations rather than changes in network depth or width.

In addition, we investigate gradient-preserving ("immortal") activations that maintain non-zero gradients in inactive regions, improving optimization stability in convolutional networks. Our results suggest that the effectiveness of activation design depends strongly on architectural context. In fully connected networks, non-monotonic activations benefit from structured representation transfer during training, while in convolutional architectures, gradient-preserving non-monotonic activations can improve optimization behavior directly.

3. Method

3.1 Overview

This work studies how activation function design and training procedures jointly influence neural network robustness and effective capacity. The proposed method consists of two key components:

- Gradient-preserving activation functions (Immortal ReLU and its variants),
- Equivalent Exchange (EqEx), a weight-transfer algorithm between monotonic and non-monotonic representations.

The overall training procedure follows a staged approach: the model is first trained using a monotonic (or gradient-preserving) activation, after which EqEx is applied once to obtain a non-monotonic representation. Training then continues in the non-monotonic space.

3.2 Immortal ReLU

Standard ReLU may lead to inactive neurons that stop contributing to learning ("dying ReLU" [16]). Existing approaches such as Leaky-ReLU [17] and PReLU [3] address this issue by preserving gradient flow in the negative region.

This work proposes a ReLU variant with separate definitions for the forward value and backward derivative. The proposed activation preserves the exact sparse output and computational simplicity of standard ReLU - $\max(0, x)$ - while maintaining non-zero gradients for inactive neurons, improving trainability similarly to Leaky-ReLU-based approaches.

The forward pass remains identical to ReLU:

$$y = \max(0, x)$$

However, the backward pass is modified:

$$\frac{dy}{dx} = \begin{cases} 1, & x > 0 \\ h_{error}, & \text{otherwise} \end{cases}$$

where h_{error} is defined as:

$$h_{error} = h_c \begin{cases} 1, & error < 0 \\ factor, & \text{otherwise} \end{cases}$$

Here, $error$ denotes the back-propagated error; the sign convention depends on the definition of the loss function. For example, $loss > 0$ may correspond to a required update in the negative direction. The parameter factor is a small positive value controlling growth in the zero-output region ($factor > 0$, typically $factor \approx 0$). The constant $h_c \in [0.01, 0.1]$ defines the derivative magnitude in ReLU's negative region (like Leaky-ReLU [17]), while h_{error} denotes the final overridden derivative value.

An important implementation detail is the decay of h_{error} during backpropagation. Specifically, when gradients propagate from an iReLU layer to layers that do not support derivative overrides (e.g., from iReLU to Leaky-ReLU), the overridden derivative is multiplied by a fading coefficient. This fading is applied only during gradient propagation beyond the iReLU layer; local weight optimization inside the iReLU layer still uses the original non-faded derivative.

During backpropagation, overridden derivatives are attenuated when gradients propagate into layers that do not support derivative overrides. This stabilization mechanism prevents uncontrolled gradient amplification across heterogeneous activation stacks while preserving the local optimization behavior

of iReLU layers. In practice, attenuation is implemented using a multiplicative fading coefficient applied only outside the originating iReLU layer.

3.3 Non-Monotonic Activation Functions

To increase effective capacity per neuron, we introduce non-monotonic activation functions.

A Leaky-ReLU-based [17] variant is defined as:

$$\mathrm{nmReLU}(x) = \max\big(-|x| + \beta, \alpha(-|x| + \beta)\big)$$

Here: $\alpha \in (0,1)$ is the Leaky-ReLU negative slope [17], $\beta > 0$ controls the peak value.

An advanced variant based on iReLU is:

$$\mathrm{nmiReLU}(x) = \max(-|x| + \beta, 0)$$

These functions introduce controlled non-monotonicity, allowing a single neuron to represent more complex patterns (e.g., XOR-like behavior).

3.4 Equivalent Exchange

3.4.1 Motivation

Non-monotonic activations increase expressiveness but make optimization more difficult due to increased non-convexity. To address this, we introduce Equivalent Exchange (EqEx), which enables training in a more stable monotonic space before transitioning learned weights to a non-monotonic representation. Weights optimized in a monotonic representation are generally suboptimal for non-monotonic activations due to differences in gradient structure and optimization landscapes; thus, the purpose of the equivalent representation is not to produce final weights directly, but to simplify optimization of the target non-monotonic network. In all cases, projection is followed by additional training in the non-monotonic space.

Although EqEx temporarily employs an expanded equivalent representation during optimization, the final deployed model preserves the original inference-time architecture and parameter budget. The equivalent decomposition further suggests that a single non-monotonic neuron may encode behaviors that would otherwise require multiple monotonic neurons, potentially increasing expressive efficiency and effective capacity under fixed inference-time constraints.

3.4.2 Equivalent Representation

The equivalent representation uses two monotonic neurons to approximate a single non-monotonic neuron. The intuition is that a non-monotonic activation can be decomposed into two complementary monotonic responses, corresponding to positive and negative correlation regimes. One neuron captures features aligned with the input, while the second models an inverted (anti-correlated) response. Their combination allows the equivalent system to approximate non-monotonic behaviors such as input-dependent change in effective response direction and XOR-like patterns, which cannot be represented by a single monotonic unit.

A non-monotonic neuron:

$$y = \mathrm{nmiReLU}\left(w_0 + \sum_{i=1..|L|} x_i\, w_i\right)$$

can be approximated using two monotonic neurons:

$$\hat{y} = \mathrm{iReLU}\left(w_0^f + \sum_{i=1..|L|} x_i\, w_i^f\right) + \mathrm{iReLU}\left(-w_0^s + \sum_{i=1..|L|} -x_i w_i^s\right)$$

Here: w^f and w^s are weights of the equivalent pair. It is noticeable that the sign for the w^s has a nominal value, since it can be leveled during training. However, it is useful for a better initialization of the weights.

A non-monotonic neuron and equivalent representation are connected by the following expression:

$$y \approx \hat{y} + \delta$$

Here δ represents projection error.

3.4.3 Practical Considerations

In practice, we use a bounded gradient-preserving activation (ilReLU) as the monotonic base:

$$\mathrm{ilReLU}(x) = \max(0, \min(x, \beta))$$

This improves compatibility with nmiReLU by matching output ranges, reducing projection error.

3.4.4 Exchange algorithm V2

EqEx V2 defines a mapping from the equivalent representation (w^f, w^s) to a non-monotonic weight w^{nm} as follows:

$$w_{i=1..|L|}^{nm} = \frac{1}{2}\begin{cases} w_i^f - w_i^s, & \mathrm{spread}(w_i^f, w_i^s) > \mu \\ |w_i^f| + |w_i^s|, & \mathrm{spread}(w_i^f, w_i^s) \le \mu \end{cases}$$

$$w_0^{nm} = \frac{(w_0^f - w_0^s)}{2} - \beta$$

$$\mathrm{spread}(w^f, w^s) = \left(\frac{\min(|w^f|, |w^s|)}{\max(|w^f|, |w^s|) + \epsilon}\right)^2 \cdot \mathrm{sign}(w^f w^s)$$

Here: w_0^{nm} is a bias weight of non-monotonic representation, $w_{i=1..|L|}^{nm}$ is inputs weights of non-monotonic representation, μ is a threshold parameter (common value is $\mu \approx -0{,}5$), $\varepsilon \approx 10^{-8}$ ensures numerical stability.

This mapping distinguishes between: complementary (anti-correlated) weights and aligned weights (additive). Anti-correlated equivalent weights indicate that the two monotonic neurons respond to opposing input patterns, corresponding to non-monotonic or XOR-like behavior. In this case, the subtraction term preserves the contrast between the two responses. Conversely, aligned weights indicate cooperative activation behavior, where additive combination better preserves the equivalent representation after projection.

This version of EqEx is applied once after an initial equivalent training phase (rewrites non-monotonic weights). After projection, training continues in the non-monotonic space to refine weights and reduce approximation error δ.

3.4.5 Exchange algorithm V4

EqEx V4 extends the equivalent exchange by introducing joint training of the equivalent (monotonic) and non-monotonic representations.

Unlike EqEx V2, where exchange is applied once after pretraining, EqEx V4 introduces periodic synchronization between equivalent and non-monotonic representations during training, enabling more stable optimization under noisy gradient updates.

The procedure consists of four steps: (1) neuron matching, (2) bias alignment, (3) regime-based weight transfer, and (4) parameter update. Some details have been omitted so as not to overload this section.

Neuron Matching. At each synchronization step, neurons in the equivalent and non-monotonic networks are matched based on similarity of their outputs over a mini-batch. We use cosine similarity as the matching criterion. A one-to-one assignment is then computed by maximizing total similarity.

Bias Alignment. Induced non-monotonic bias:

$$w_0^{bar} = (w_0^f + w_0^s) / 2$$
$$w_0^{\widehat{nm}} = w_0^{bar} - \beta$$

Here: β is the same offset parameter used in the non-monotonic activation function, $w_0^{\widehat{nm}}$is target bias values from equivalent.

Regime-Based Weight Transfer. For each matched weight pair$(\mathrm{w}_i^{\mathrm{f}}, \mathrm{w}_i^{\mathrm{s}})$, a structural similarity measure s is computed using a spread function:

$$s = \mathrm{spread}(\mathrm{w}_i^{\mathrm{f}}, \mathrm{w}_i^{\mathrm{s}})$$

The three transfer regimes correspond to different interaction patterns between equivalent neurons: cooperative activation (linear regime), dominance of one representation (dominant regime), and strongly complementary or anti-correlated behavior (XOR-like regime).

Based on s, weights are updated according to three regimes (μ is the same as in EqEx-V2):

- Linear regime ($s > 0$): $w_i^{\widehat{nm}} = w_i^f + w_i^s$
- Dominant regime ($s > \mu \wedge s \leq 0$): $w_i^{\widehat{nm}} = \begin{cases} w_i^f, & |w_i^f| \geq |w_i^s| \\ w_i^s, & |w_i^f| < |w_i^s| \end{cases}$
- XOR-like regime ($s \leq \mu$): $w_i^{\widehat{nm}} \to \beta$ and equivalent weights are adjusted to enforce approximate anti-correlation between representations (i.e., $w_i^f \approx -w_i^s$) rather than using direct projection.

Update Rule. The non-monotonic weights are updated as:

$$w_i^{nm} = w_i^{nm} - \left(\frac{\gamma}{2}\right) * (w_i^{nm} - w_i^{\widehat{nm}})$$

Here: γ is exchange intensity parameter (like learning rate). Equivalent weights receive corresponding corrections depending on the regime.

Statistical Stabilization. Batch statistics (mean and variance of activations) are used to normalize differences between representations, ensuring that bias and weight scaling remain consistent after exchange.

4. Experimental Setup

4.1 Datasets

We evaluate the proposed methods on two standard benchmarks: MNIST [18] and CIFAR-10 [19].

MNIST consists of grayscale handwritten digit images of size 28×28, used for evaluating fully connected architectures. CIFAR-10 contains natural color images of size 32×32×3 across 10 classes and presents a more challenging setting for convolutional networks.

4.2 Architectures

4.2.1 Fully Connected Networks (MNIST)

For MNIST experiments, we use fully connected networks with the following structure:

- input layer of size 784,
- first hidden layer with 32 or 64 neurons,
- second hidden layer with 16 neurons,
- output layer with 10 neurons and softmax activation.

All configurations share identical architecture except for activation functions and EqEx usage.

4.2.2 Convolutional Networks (CIFAR-10)

To examine whether activation-induced effects generalize to more complex visual domains, we evaluate compact convolutional architectures on CIFAR-10.

Each network consists of four convolutional layers followed by a dense classification head:

- input: 32×32×3,
- convolution (16/32 channels, 3×3), activation,
- convolution (16/32 channels), activation,
- 2×2 max pooling,
- convolution (32/64 channels), activation,
- convolution (32/64 channels), activation,
- 2×2 max pooling,
- flattening,
- dense layer (128/256 neurons),
- linear logits (10), softmax output.

The Leaky-ReLU networks naming convention reflects layer sizes. For example, ReLU-16/32/128 denotes 16 and 32 channels in convolutional blocks and 128 neurons in the dense layer. A larger configuration (ReLU-32/64/256) approximately doubles the number of parameters.

4.3 Evaluation Protocol

For MNIST inputs are normalized to [0,1]; batch size: 64; optimizer: Adam [20] with learning rate 10^{-3}; training duration: 10 epochs. EqEx-V2 or EqEx-V4 is applied once after an initial pre-training phase (2 epochs), after which training continues in the non-monotonic representation. The evaluation considers targeted gradient-based input synthesis. Unlike conventional adversarial evaluation, the synthesis starts from a randomly generated input vector rather than a test-set sample and uses a randomly selected target label. Two white-box gradient-based input synthesis procedures are evaluated: Carlini-Wagner L2 (CW-L2) [21]; Basic Iterative Method (BIM) [22] (an extension of FGSM [23]). In this experiment, robustness refers to the resistance of the model to targeted input synthesis. For each trained model, 128 independent synthesis attempts are conducted, and the targeted input synthesis success rate, defined as the fraction of attempts that reach the selected target label, is recorded.

For CIFAR-10 standard per-channel normalization (mean subtraction and division by standard deviation); optimizer: Adam [20] with learning rate 10^{-3}; batch size: 32; training duration: 10 epochs.

No data augmentation, weight decay, or dropout is applied, in order to isolate the effect of activation functions on optimization and performance. All results are averaged over multiple independent runs with different random seeds.

4.4 Activation Configurations

We evaluate several activation configurations: Leaky-ReLU (baseline); Immortal ReLU (iReLU); non-monotonic variants (nmReLU, nmiReLU); nmiReLU with EqEx (for fully connected networks).

For MNIST, both direct non-monotonic training and EqEx-based training are evaluated.

For CIFAR-10, EqEx is not applied due to architectural constraints, and activation functions are used directly.

5. Results

5.1 MNIST

5.1.1 Configurations

The following configurations were evaluated:

1. **ReLU-64**, the network using Leaky-ReLU was trained using a hidden layer of size N = 64 for the 10 epochs.
2. **ReLU-32**, the network using Leaky-ReLU was trained using a hidden layer of size N = 32 for the 10 epochs.
3. **iReLU-32**, the network like ReLU-32 but on "Immortal" ReLU was trained using a hidden layer of size N = 32 for the 10 epochs.
4. **nmReLU-JN**, the network on "non-monotonic" ReLU was trained directly (with no equivalent phase, just non-monotonic), N = 32 for the 10 epochs.
5. **nmiReLU-JN**, the network on "non-monotonic immortal" ReLU was trained directly (with no equivalent phase training, just non-monotonic), N = 32 for the 10 epochs.
6. **nmiReLU-EQ-V2**, a two-phase training was employed: initially, a network with ilReLU activation and a hidden layer of size N = 64 was trained for 2 epochs. The learned parameters were then projected with EqEx-V2 onto a non-monotonic network based on nmiReLU activation function with a reduced hidden layer size of N = 32, after which training continued for an additional 8 epochs. The projection procedure used the spread metric with a threshold value of -0.4.
7. **nmiReLU-EQ-V4**, a two-phase training similar to nmiReLU-EQ-V2 (6), except that first 2 epochs equivalent and non-monotonic networks are trained simultaneously and that projection is performed by EqEx-V4.

5.1.2 Training Results

Here we compare the final performance on the test set. As shown in Fig. 1, the configurations exhibit noticeable differences in both accuracy and convergence behavior. The corresponding numerical results are summarized in Table 1. ReLU-32 also exhibits higher variance across runs, reflected in its larger standard deviation. The correlation between accuracy and F1-score exceeded 0.999 for all configurations.

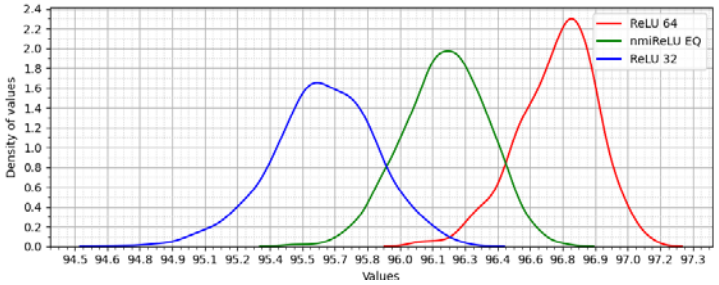


Fig. 1. Accuracy value density of ReLU-32, nmiReLU-EQ-V2, ReLU-64 (kernel density).

Thus, the two-phase training in nmiReLU-EQ-V2 appears to enhance the capacity of its internal representation. Depending on the baseline, this manifests as a slight decrease relative to ReLU-64 (−0.5%) or as an increase relative to ReLU-32 (+0.6%). Taken together, the results suggest that nmiReLU-EQ-V2 attains greater effective capacity and, consequently, higher internal entropy for the

same parameter budget. Setup 7 (nmiReLU-EQ-V4) shows even greater accuracy with the same inference cost, but requires more computation time at the training stage.

Table 1. Numerical results of accuracy in MNIST experiments.

Criteria Name	Results			
	Setup 1	**Setup 7**	**Setup 6**	**Setup 2**
Activation Function	Leaky-ReLU	ilReLU=>nmiReLU	ilReLU=>nmiReLU	Leaky-ReLU
Layer *N* size	64	64 => 32	64 => 32	32
Epochs	10	2 => 8	2 => 8	10
Accuracy (%)	96.724	96.411	96.208	95.649
Variance (Acc)	0.0338	0.0342	0.0387	0.0584
SD (Acc)	0.184	0.185	0.197	0.241

Next, we can compare the training results of ReLU-32 and iReLU-32, which proved to be very similar: on average, 95.649% and 95.613%, respectively, with standard deviations of 0.242 and 0.243. The corresponding distributions are shown in Fig. 2. Although ReLU-32 achieved slightly higher accuracy, the practical difference between the configurations is small. A Mann-Whitney [24] test nevertheless indicated statistical significance ($p < 0.001$), suggesting that the observed difference is consistent across runs. Despite this, iReLU remains preferable in settings involving equivalent exchange, since its gradient-preserving behavior enables optimization of the target non-monotonic representations.

Training results were compared for nmReLU-JN, nmiReLU-JN, and nmiReLU-EQ-V2, with average accuracies of 95.824% (standard deviation is 0.199), 95.814% (standard deviation is 0.206), and 96.208% (standard deviation is 0.197), respectively. As shown in Fig. 3, directly trained non-monotonic activations outperform ReLU-32, while EqEx-based training further improves performance under the same final architecture. This suggests that the equivalent monotonic representation simplifies optimization of non-monotonic networks and improves access to more effective solutions.

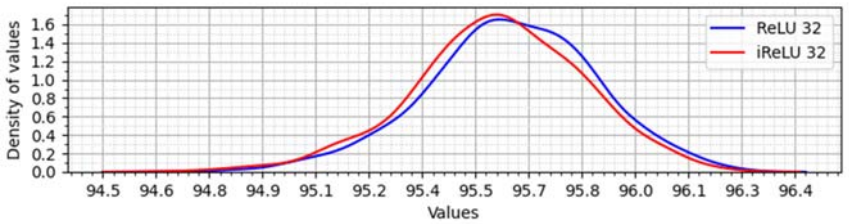


Fig. 2. Accuracy value density of ReLU-32 and iReLU-32 (kernel density).

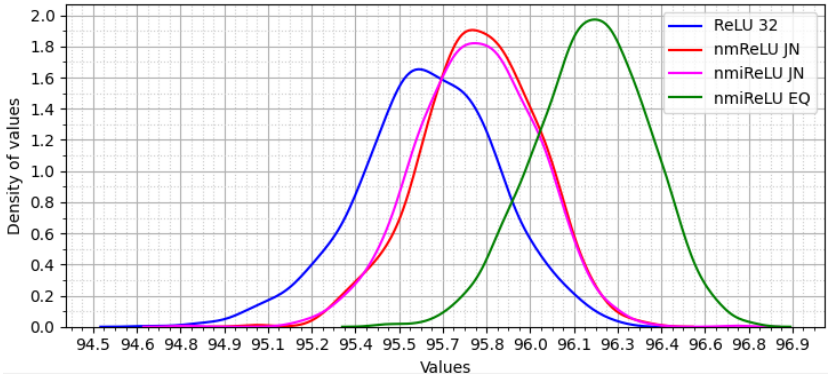


Fig. 3. Accuracy value density of ReLU-32, nmReLU-JN, nmiReLU-JN, nmiReLU-EQ-V2.

5.1.3 Learning Dynamics

To further analyze optimization behavior beyond final performance, we examine training dynamics over 20 epochs. Fig. 4 shows the evolution of train accuracy and Fig. 5 shows the evolution of test accuracy for different activation configurations, including direct non-monotonic training (nmReLU-JN, nmiReLU-JN), standard baselines (ReLU-32, ReLU-64), and EqEx-based methods (nmiReLU-EQ-V2, nmiReLU-EQ-V4). The bold line on the figures shows the weighted value for several launches, and the translucent area of the corresponding color is one standard deviation. For Fig. 4 note that the accuracy and F1 are highly-correlated.

All configurations demonstrate rapid initial learning within the first 3-5 epochs, followed by a slower convergence phase. However, clear differences emerge in both convergence stability and final performance.

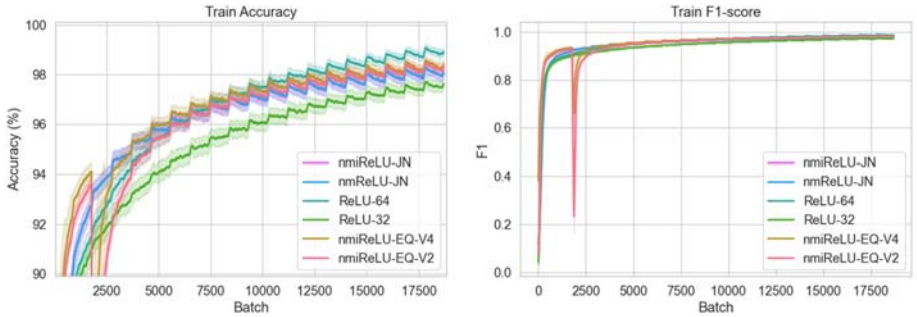


Fig. 4. MNIST train accuracy and F1 curves (different scale) over batches.

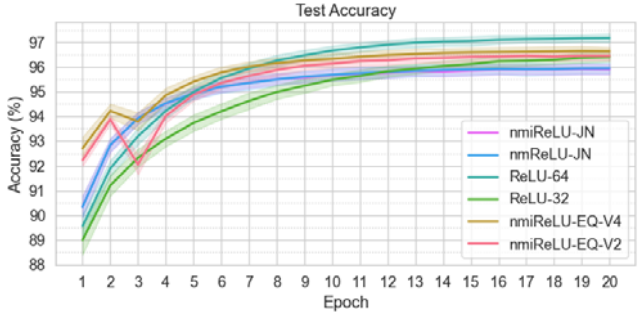


Fig. 5. MNIST test accuracy curve over epochs.

Directly trained non-monotonic models exhibit fast early improvement, but their optimization becomes unstable in later epochs. In particular, while training accuracy continues to increase, test accuracy tends to plateau or degrade, indicating overfitting and reduced generalization stability. As a result, their final performance remains below that of baseline ReLU models in later stages of training.

ReLU-based models show stable and predictable convergence behavior. ReLU-64 achieves the highest overall performance, while ReLU-32 converges more slowly and to a lower accuracy level, reflecting the effect of reduced effective capacity.

EqEx-based methods demonstrate improved optimization behavior compared to direct non-monotonic training. In particular, nmiReLU-EQ-V2 achieves consistently higher accuracy early in training and maintains a stable advantage over direct non-monotonic variants throughout optimization. The nmiReLU-EQ-V4 variant further improves stability, exhibiting smoother learning curves and reduced performance degradation associated with the representation transfer step. In comparison to V2, V4 shows a smaller post-exchange accuracy drop, indicating improved alignment between monotonic and non-monotonic representations.

Overall, these results suggest that non-monotonic activations alone may introduce optimization instability, while the EqEx procedure mitigates these effects by providing a structured transition between representations. This leads to more stable convergence behavior and improved final performance.

5.1.4 Input Synthesis Evaluation

First, consider the ReLU-32 configuration separately. For ReLU-32, Carlini-Wagner L2 (CW-L2) targeted input synthesis achieved a 99.685% success rate (standard deviation 1.426), while the Basic Iterative Method (BIM) achieved 99.987% (standard deviation 0.1). Thus, both evaluated gradient-based synthesis procedures were highly successful for ReLU-32.

For CW-L2, targeted input synthesis success rates on non-monotonic variants were as follows: 60.109% (standard deviation 10.156) for nmReLU-JN, 60.139% (standard deviation 10.342) for nmiReLU-JN, and 63.213% (standard deviation 13.01) for nmiReLU-EQ-V2 (Fig. 6). The nmReLU-JN and nmiReLU-JN plots are similar, while nmiReLU-EQ-V2 shows a wider spread across the distribution. The higher synthesis success rate of nmiReLU-EQ-V2 may be related to its lower non-monotonicity utilization (12.2% compared to about 17% utilization with nmReLU-JN and nmiReLU-JN). Although nmReLU-JN and nmiReLU-JN reduce synthesis success, their training is less controllable - potentially resisting not just input synthesis, but also effective learning (it was shown in learning dynamics section).

For BIM, targeted input synthesis success rates on non-monotonic variants were: 80.996% (standard deviation 13.35) for nmReLU-JN, 83.247% (standard deviation 13.155) for nmiReLU-JN, and 70.922% (standard deviation is 15.978) for nmiReLU-EQ-V2 (Fig. 7).

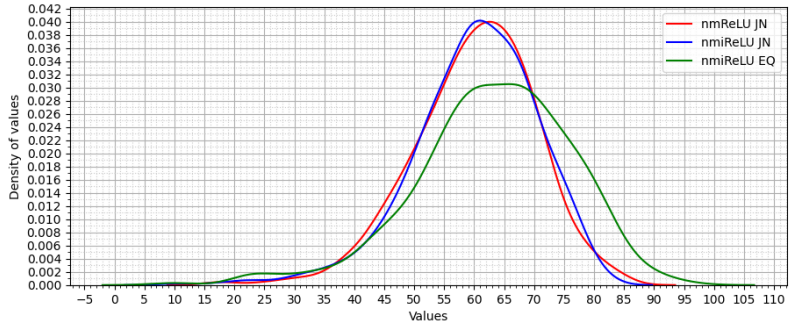


Fig. 6. Carlini-Wagner L2 targeted input synthesis success rate distribution (kernel density).

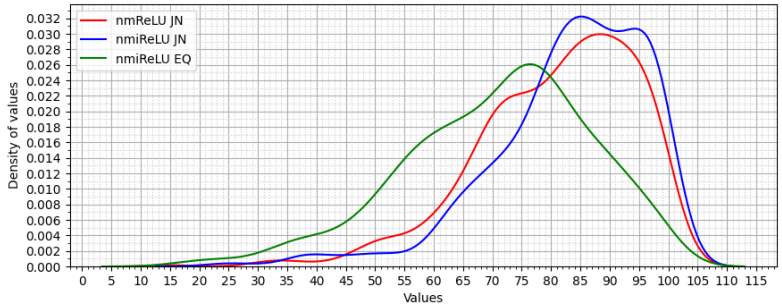


Fig. 7. Basic Iterative Method targeted input synthesis success rate distribution (kernel density).

Thus, nmiReLU-EQ-V2 achieves the lowest synthesis success rate among the evaluated models. The higher success rates of nmReLU-JN and nmiReLU-JN may be related to their direct iterative training in the non-monotonic representation. In contrast, nmiReLU-EQ-V2 is initialized through EqEx from a monotonic equivalent representation. After EqEx, optimization continues in the non-monotonic representation, whose optimization landscape differs from that of the initial monotonic representation. This change may affect the local optimization behavior of gradient-based synthesis procedures such as BIM and contribute to the observed reduction in synthesis success. This result indicates increased resistance to the evaluated targeted input-synthesis procedures. This setting is related to input synthesis [25] and model inversion [26], where inputs are generated to produce prescribed model responses. It represents a scenario in which an attacker attempts to synthesize an input producing a desired model output without starting from an existing sample.

5.1.5 Statistical Significance

Statistical significance was assessed using the Mann-Whitney U test [24], as the data are not consistently normally distributed. Results are shown in Table 2. Abbreviations: ReLU-32 (R32), Non-monotonic ReLU (nmReLU), Non-monotonic Immortal ReLU with equivalent-phase training (nmiEQ), Basic Iterative Method (BIM), Carlini-Wagner L2 (CW-L2).

Table 2. Mann-Whitney test results on MNIST.

Compare	Metric	U	P
R32 vs nmiEQ	Accuracy	$3.15 \cdot 10^6$	≈ 0
R32 vs nmiEQ	CW-L2	22.5	$2.09 \cdot 10^{-118}$
R32 vs nmiEQ	BIM	496	$1.72 \cdot 10^{-117}$
nmReLU vs nmiEQ	BIM	$1.36 \cdot 10^5$	$9.20 \cdot 10^{-33}$

5.2 CIFAR

5.2.1 Configurations

The following configurations were considered:

1. **ReLU-16/32/128** (ReLU1632), based on Leaky-ReLU smaller model with 16 and 32 channels in convolutional blocks and 128 neurons in the dense layer,
2. **ReLU-32/64/256** (ReLU3264), based on Leaky-ReLU larger model with doubled capacity,
3. **NM-M**, model uses non-monotonic mirrored activation proposed in [1],
4. **NM-DI**, model uses non-monotonic immortal activation (nmiReLU) trained directly.

5.2.2 Training results

Here we compare the final performance on the test set. The results show a clear separation between activation types in terms of achievable accuracy.

The larger baseline model (ReLU3264) achieved the highest performance, reaching a maximum mean test accuracy of 71.845% (standard deviation 0.649). The smaller baseline (ReLU1632) achieved 68.595% (standard deviation 0.879), reflecting the expected capacity gap.

Among non-monotonic variants, the mirrored activation (NM-M) reached 68.384% (standard deviation 0.728), slightly below the ReLU1632 baseline. In contrast, the non-monotonic immortal activation (NM-DI) achieved 71.927% (standard deviation 0.962), significantly outperforming both ReLU1632 and NM-M.

Notably, NM-DI approaches the performance of the larger ReLU3264 model despite having approximately half the number of parameters. This suggests that preserving gradient flow in non-monotonic activations substantially improves effective capacity (the ability of the network to use its

capacity). Across all configurations, the correlation between accuracy and F1 score exceeded 0.999, indicating consistent behavior across evaluation metrics.

5.2.3 Learning dynamics

Training curves functions (Fig. 8) reveal consistent differences in convergence behavior across activation. The larger baseline (ReLU3264) converges quickly and remains stable after reaching its peak performance. In contrast, the smaller ReLU1632 model saturates earlier and exhibits mild degradation in later epochs, indicating the onset of overfitting or limited capacity.

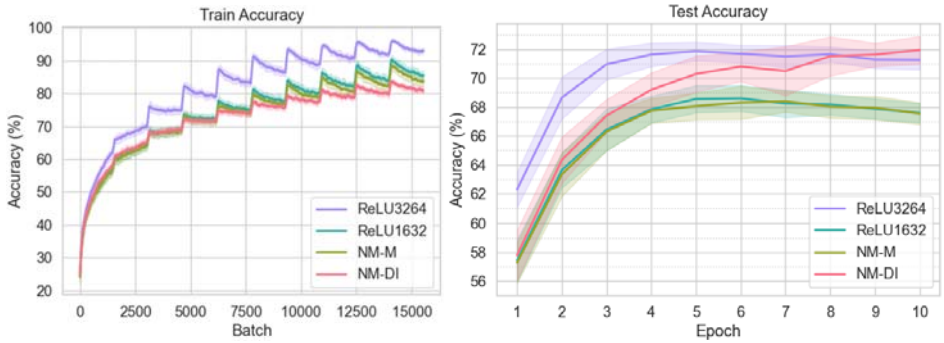


Fig. 8. CIFAR-10 train and test accuracy curves.

A similar but more pronounced effect is observed for the symmetric non-monotonic variant (NM-M), which peaks early and then gradually degrades. This suggests that non-monotonicity alone does not necessarily improve optimization behavior in convolutional settings.

In contrast, the gradient-preserving non-monotonic variant (NM-DI) demonstrates more stable behavior. It continues improving beyond the point where other compact models saturate and does not exhibit the same degradation trend. As a result, it approaches the performance of the larger baseline while maintaining a compact architecture.

Overall, these results indicate that gradient preservation plays a critical role in stabilizing training: while standard and symmetric activations tend to overfit or stagnate, the proposed NM-DI maintains consistent improvement and better generalization.

5.2.4 Statistical Significance

Statistical significance was assessed using the Mann-Whitney U test [24] on best-epoch accuracies. The improvement of NM-DI over ReLU1632 is statistically significant ($p \approx 4.6 \cdot 10^{-9}$), as well as over NM-M ($p \approx 4.8 \cdot 10^{-7}$). Differences between NM-M and ReLU1632 are less pronounced and not statistically significant at standard thresholds ($p \approx 0.604$), confirming that mirrored non-monotonicity alone does not reliably improve performance.

6. Limitations

This work is limited to relatively compact architectures and datasets (MNIST and CIFAR-10), and it remains unclear how well the proposed methods scale to larger models or datasets. The EqEx procedure also relies on several heuristic design choices, including spread thresholds and synchronization strategies, without formal convergence guarantees.

In addition, EqEx was substantially more effective for fully connected networks than for convolutional architectures. Preliminary experiments suggest that extending representation transfer to convolutional layers introduces structural difficulties related to channel interactions, reducing its effectiveness compared to direct non-monotonic gradient-preserving training in the evaluated settings.

Finally, robustness evaluation was limited to standard white-box gradient-based procedures on MNIST. Although related work on symmetric activation functions [1] reported robustness improvements in convolutional architectures, robustness of the gradient-preserving activations for CNNs was not directly evaluated in this work.

7. Conclusion

This work shows that activation design, combined with appropriate training strategies, can improve neural network performance under fixed parameter budgets.

On MNIST, non-monotonic activations with Equivalent Exchange (EqEx) improve resistance to the evaluated targeted gradient-based input synthesis procedures while maintaining or slightly increasing accuracy (up to +0.6–0.8%). EqEx enables stable training by transferring weights from a monotonic representation to a non-monotonic one.

On CIFAR-10, EqEx does not directly extend to convolutional networks, but gradient-preserving ("immortal") activations demonstrate clear benefits. They improve training stability, reduce late-stage degradation, and achieve higher accuracy than both ReLU and symmetric variants, approaching larger models.

Overall, the results indicate that non-monotonicity and gradient preservation are effective in different regimes: representation transfer benefits fully connected networks, while stable gradient flow is critical for convolutional architectures.

References

[1]. Zhao Q., Griffin L.D. Suppressing the Unusual: towards Robust CNNs using Symmetric Activation Functions. arXiv preprint, 2016. DOI: 10.48550/arXiv.1603.05145.

[2]. Misra D. Mish: A Self Regularized Non-Monotonic Activation Function. arXiv preprint, 2019. DOI: 10.48550/arXiv.1908.08681.

[3]. He K., Zhang X., Ren S., Sun J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv preprint, 2015. DOI: 10.48550/arXiv.1502.01852.

[4]. Ramachandran P., Zoph B., Le Q.V. Searching for Activation Functions. arXiv preprint, 2017. DOI: 10.48550/arXiv.1710.05941.

[5]. Chen T., Goodfellow I., Shlens J. Net2Net: Accelerating Learning via Knowledge Transfer. arXiv preprint, 2016. DOI: 10.48550/arXiv.1511.05641.

[6]. Wei T., Wang C., Rui Y., Chen C.W. Network Morphism. arXiv preprint, 2016. DOI: 10.48550/arXiv.1603.01670.

[7]. Agostinelli F., Hoffman M., Sadowski P., Baldi P. Learning Activation Functions to Improve Deep Neural Networks. arXiv preprint, 2015. DOI: 10.48550/arXiv.1412.6830.

[8]. Bingham G., Miikkulainen R. Discovering Parametric Activation Functions. Neural Networks, 2022, vol. 148, pp. 48-65. Available at: https://www.sciencedirect.com/science/article/pii/S0893608022000020/pdf, accessed 15.04.2026.

[9]. Guo Q., Juefei-Xu F., Zhou C., Feng W., Liu Y., Wang S. Sparta: Spatially Attentive and Adversarially Robust Activation. arXiv preprint, 2022. DOI: 10.48550/arXiv.2105.08269.

[10]. Xiao C., Zhong P., Zheng C. Enhancing Adversarial Defense by k-Winners-Take-All. In International Conference on Learning Representations (ICLR), 2020. Available at: https://openreview.net/pdf?id=Skgvy64tvr, accessed 15.04.2026.

[11]. Lee K., Yang J., Lee H., Hwang J.Y. Stochastic Adaptive Activation Function. arXiv preprint, 2022. DOI: 10.48550/arXiv.2210.11672.

[12]. Dai S., Mahloujifar S., Mittal P. Parameterizing Activation Functions for Adversarial Robustness. arXiv preprint, 2021. DOI: 10.48550/arXiv.2110.05626.

[13]. Bingham G., Miikkulainen R. Efficient Activation Function Optimization through Surrogate Modeling. arXiv preprint, 2023. DOI: 10.48550/arXiv.2301.05785.

[14]. Alexandridis K. P., Deng J., Nguyen A., Luo S. Adaptive Parametric Activation. In: Computer Vision – ECCV 2024, Lecture Notes in Computer Science. Springer, 2024. DOI: 10.48550/arXiv.2407.08567.

[15]. Chen D., Li J., Xu K. AReLU: Attention-based Rectified Linear Unit. arXiv preprint, 2020. DOI: 10.48550/arXiv.2006.13858.

[16]. Lu L., Shin Y., Su Y., Karniadakis G. E. Dying ReLU and Initialization: Theory and Numerical Examples. Communications in Computational Physics, 2020, vol. 28, no. 5, pp. 1671-1706. Available at: https://www.global-sci.org/intro/article_detail/cicp/18137.html, accessed 15.04.2026.

[17]. Maas A. L., Hannun A.Y., Ng A.Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. 2013. Available at: https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf, accessed 15.04.2026.

[18]. LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, vol. 86, no. 11, pp. 2278-2324.

[19]. Krizhevsky A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. Available at: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, accessed 15.04.2026.

[20]. Kingma D. P., Ba J. Adam: a method for stochastic optimization. arXiv preprint, 2017. DOI: 10.48550/arXiv.1412.6980.

[21]. Carlini N., Wagner D. Towards evaluating the robustness of neural networks. 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 39-57. DOI: 10.1109/SP.2017.49.

[22]. Kurakin A., Goodfellow I., Bengio S. Adversarial examples in the physical world. arXiv preprint, 2016. DOI: 10.48550/arXiv.1607.02533.

[23]. Goodfellow I. J., Shlens J., Szegedy C. Explaining and Harnessing Adversarial Examples. arXiv preprint, 2015. DOI: 10.48550/arXiv.1412.6572.

[24]. Mann H. B., Whitney D. R. On a test of whether one of two random variables is stochastically larger than the other. The Annals of Mathematical Statistics, 1947, vol. 18, no. 1, pp. 50-60.

[25]. Nguyen A., Dosovitskiy A., Yosinski J., Brox T., Clune J. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. arXiv preprint, 2016. DOI: 10.48550/arXiv.1605.09304.

[26]. Pattichis R., Janampa S., Pattichis C. S., Pattichis M. S. Inverting Neural Networks: New Methods to Generate Neural Network Inputs from Prescribed Outputs. arXiv preprint, 2026. DOI: 10.48550/arXiv.2603.20461.

Информация об авторах / Information about authors

Михаил Алексеевич СЕНТЮРЕВ является независимым исследователем в области теории алгоритмов и основ машинного обучения, магистром факультета прикладной математики и информатики Московского физико-технического института (МФТИ) по кафедре алгоритмов и технологий программирования. В сферу его научных интересов входят архитектуры нейронных сетей, функции активации, алгоритмы обучения, а также методы повышения устойчивости нейросетей к атакам.

Mikhail Alekseevich SENTIUREV is an independent researcher in the field of algorithm theory and foundations of machine learning, with a master's degree from the Phystech School of Applied Mathematics and Computer Science at the Moscow Institute of Physics and Technology (MIPT), where he studied in the Department of Algorithms and Programming Technologies. His research interests include neural network architectures, activation functions, learning algorithms, and methods for improving the robustness of neural networks against attacks.

Илья Андреевич МАКАРОВ является старшим научным сотрудником исследовательского центра доверенного искусственного интеллекта Института системного программирования РАН им. В.П. Иванникова (ИСП РАН), PhD по компьютерным наукам. Его научные интересы включают кооперативные мультиагентные LLM-системы, автономную генерацию научного знания.

Ilya Andreevich MAKAROV is a senior researcher at the Research Center for Trusted Artificial Intelligence of the Institute for System Programming of the Russian Academy of Sciences (ISP RAS), with a PhD in Computer Science. His research interests include cooperative multi-agent LLM systems and autonomous scientific knowledge generation.