Electronics and information technologies, 2026, 34,31–50
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC 004.94

MULTI-PARAMETER DYNAMIC LEARNING MAPS OF NEURAL NETWORKS TRAINED WITH ADAM OPTIMISER

***Sergiy Sveleba*[1]*, *Ivan Katerynchuk*[1], *Yaroslav Shmyhelskyy*[1], *Lucjan Pelc*[2], *Volodymyr Brygilevych*[2], *Nazar Sveleba*[3]**
[1]Ivan Franko National University of Lviv,
107 Gen. Tarnavskoho Str., Lviv, 79017, Ukraine
[2]The State Higher School of Technology and Economics in Jarosław,
ul. Czarnieckiego 16, Jarosław, 37-500, Poland
[3]Private Higher Education Establishment "European University",
16B, Academician Vernadsky Boulevard, Kyiv, Ukraine

Sveleba, S., Katerynchuk, I., Shmyhelskyy, Ya., Pelc, L., Brygilevych, V., Sveleba, N. (2026). Multi-Parameter Dynamic Learning Maps of Neural Networks Trained with Adam Optimizer. *Electronics and Information Technologies*, 34, 31-50. https://doi.org/10.30970/eli.34.2

ABSTRACT

Background. Understanding how hyperparameters of the Adam optimizer shape the learning dynamics of multilayer neural networks remains an open problem, particularly in regimes where training transitions from stable to chaotic behavior. Previous studies have shown that learning rate and network complexity can lead to underfitting, efficient learning, overfitting, or chaotic states; however, systematic visualization of these regimes in the β_1–β_2 parameter space is still lacking.

Materials and Methods. Dynamic regime maps are constructed for a multilayer neural network trained on binary-encoded digit patterns. The training process is analyzed using a recurrent two-dimensional mapping based on the error evolution of a neuron and its neighbor. Periodicities in the final iterations are identified via the convergence of periodicities method, enabling classification of learning regimes from fixed points to high-order cycles and chaos. The maps are generated for varying learning rates α, training duration, network depth, and neuron count, with β_1 and β_2 scanned over the interval [0, 0.999].

Results and Discussion. The results demonstrate strong sensitivity of Adam dynamics to the joint values of β_1 and β_2. Small learning rates ($\alpha = 0.001$) produce predominantly first-order periodicity, indicating underfitting. Moderate values ($\alpha = 0.1 - 0.4$) reveal structured transitions from rapid to slower learning as β_2 increases. In contrast, large α (≥ 0.6) leads to fragmented, non-monotonic patterns associated with overfitting and chaotic dynamics. Increasing network size introduces noisy gradients, shrinking stable regions and expanding chaotic zones, whereas smaller networks yield smoother regime transitions.

Conclusion. Dynamic regime mapping in the β_1–β_2 space serves as an effective diagnostic tool for identifying stable, inefficient, and chaotic learning regimes. The findings confirm the high sensitivity of Adam to hyperparameter interactions and network architecture, showing that inappropriate combinations can induce chaotic behavior even at moderate learning rates. The proposed framework provides a novel approach for analyzing optimizer dynamics and guiding hyperparameter selection in deep learning.

Keywords: Adam optimizer; learning dynamics; periodicity maps; hyperparameter tuning; chaotic learning; multilayer neural networks.

The present study should primarily be interpreted as a qualitative analysis of the dynamic properties of neural-network learning under controlled conditions, rather than as a benchmark comparison on large-scale datasets. The use of a binary 4×7 dataset was intentionally chosen to minimize the influence of high-dimensional factors and to improve the interpretability of optimizer dynamics. Preliminary experiments on MNIST and CIFAR-10 datasets demonstrated qualitatively similar fragmentation of stability regions, although with significantly higher noise levels and lower interpretability.

INTRODUCTION

Optimizers adjust network parameters to minimize a loss function and are central to training efficiency and generalization. Adam combines momentum on first moments with adaptive scaling by second moments, providing fast convergence in many settings but exhibiting sensitivity to the learning rate and moment decay factors (β_1, β_2). Here, we investigate how Adam's hyperparameters, together with iteration budget and model size, shape the qualitative dynamics of learning.

Background and related work

Neural network training optimizers are used to adjust model weights to minimize the loss function. They play a key role in the learning process, helping the network find the best parameter values to summarize the data and achieve high accuracy on the test set.

The main functions of optimizers can be considered:

- convergence acceleration – optimizers help to quickly find the minimum loss function;
- overcoming local minimums – optimizers that allow you to get out of local minimums and find better solutions;
- adaptability to different parameters – optimizers change the learning speed for different weights, improving the stability of learning;
- prevention of overtraining – optimizers or their modifications, which can contribute to better generalization of the model.

The main types of optimizers include gradient descent and adaptive methods.

Gradient Descent optimizers include:

- regular (Batch Gradient Descent), which uses the entire dataset to update the weights;
- stochastic (SGD – Stochastic Gradient Descent), which updates the weights after each cycle, which can speed up learning, but makes it unstable;
- minibatch (minibatch Gradient Descent), which is a compromise between the two previous methods.

Adaptive methods:

- Momentum – adds "inertia" to updates, helping to overcome local lows;
- Adagrad – adaptively changes the learning speed for different parameters;
- RMSprop is a modification of Adagrad that eliminates its main drawback – a decrease in the learning speed too quickly;
- Adam (Adaptive Moment Estimation) – combines the advantages of Momentum and RMSprop, one of the most popular optimizers;
- AdamW is an improved version of Adam with better regulation.

Each of the optimizers has its own advantages and is used depending on the task, the type of neural network, and the size of the data.

Adam is one of the most popular and effective optimizers for training neural networks. At its core, the Adam optimizer is an extension of stochastic gradient descent methods. It includes adaptive learning speed for each parameter, and dynamic step size adjustment during training. This adaptability allows the method to navigate

efficiently in an optimization environment, mitigating the problems associated with fixed learning rates. The algorithm's adaptability and momentum methods contribute to faster convergence and improved overall performance of deep neural networks [1]. Adam optimizer works as an extension of the traditional gradient descent algorithm, introducing adaptive learning rates to improve the optimization process. The key to its functionality lies in its ability to adapt the learning speed for each parameter individually. This adaptability is achieved by combining two important concepts: momentum methods and adaptive learning rates [1]

Adam momentum update.

The Adam optimization method calculates the average slope of the path (m_t) and the average steepness of the hills (v_t):

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{1}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2. \tag{2}$$

where β_1 and β_2 are hyperparameters controlling exponential velocities,
g_t is the gradient of the loss function with respect to the model parameters at iteration t.

Bias correction.

There is a slight correction to make sure that these averages are fair, especially at the beginning.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \tag{3}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{4}$$

where $\hat{m}_t$ represents the exponentially weighted average of past gradients,
$\hat{v}_t$ represents the exponentially weighted mean of past quadratic gradients,

Parameter update.

It then uses these averages to decide how large the steps should be for subsequent iterations. The model parameters are updated according to the equation below.

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t + \varepsilon}} \hat{m}_t, \tag{5}$$

where *α* is a learning rate,ε and is a small constant that prevents division by zero.

Let's consider the deeper meaning of these hyperparameters. Understanding and fine-tuning hyperparameters is critical to harnessing the full potential of the Adam optimizer in neural network projects. Properly tuning them ensures optimal convergence and performance for most machine learning or deep learning tasks.

- Learning step (α): controls the size of the step during optimization. A higher learning rate may result in faster convergence but may also result in missing the minimum. Requires careful adjustment.
- β_1 and β_2: exponential recession rates for past gradients and quadratic gradients, respectively. They affect the weight of recent and past information. Common default values: 0.9 for β_1 and 0.999 for β_2.
- Epsilon (ε): a small constant to prevent division by zero. As a rule, it has a small value, for example, 10^{-8}.

The speed of adaptive learning in Adam plays a crucial role in solving the problems faced by traditional optimization algorithms. Adam dynamically adjusts the step size of each parameter, ensuring a balanced and efficient convergence process. This adaptability mainly makes adaptive optimizers like Adam the best algorithms when dealing with sparse gradients, non-stationary targets, or different gradient sizes for different parameters.

Adam's performance may depend on the choice of learning speed. In some cases, an incorrect setting of the learning speed can lead to suboptimal convergence or method divergence. Thus, in particular, in the work [2] it was noted that depending on the size of the learning rate of the α, there may be different modes of learning, namely: non-additional training, the mode of satisfactory training, and overfitting with the transition to a chaotic mode of training. Logically, the interval of their existence in α depends on the number of iterations.

Adam supports additional state variables (moment estimates) for each parameter, which increases memory requirements. For large deep learning models or datasets, this can be a limitation, especially in environments with limited memory.

Unlike some simpler optimization algorithms, Adam does not have strong theoretical guarantees of convergence. Although in practice it often works well. The lack of clear theoretical guarantees can make it difficult to predict its behavior in specific scenarios.

In cases where the target function has a lot of noise, Adam may not work optimally. Noise-related gradients can affect the rate of adaptive learning and moment estimation, resulting in suboptimal convergence.

Adam's performance depends on the initial values of his moment estimates (m and v). In some cases, incorrect initialization can affect convergence.

The additional computations required to maintain and update moment estimates make Adam computationally more expensive compared to simpler optimizers, especially in scenarios with large models and datasets.

In some cases, Adam's impulse-like behavior can cause the minimum to be exceeded, especially in the large dimension parameter space.

A number of publications are devoted to the Adam optimization method. Here are some key publications devoted to this optimization method. In the paper [3], the authors first presented the Adam algorithm, which combines the advantages of the Momentum and RMSprop methods for adaptive adjustment of the learning rate during stochastic optimization. The author's article by Rahul Agarwal [4] provides a detailed overview of the Adam algorithm, its advantages, drawbacks, and practical aspects of use, which can be useful for machine learning practitioners. In the paper [5], the authors propose a modification of the Adam algorithm, known as AdamW, which improves the regularization of the model by separating the weight decay from parameter updates, which contributes to better generalization of the model. The paper [6] analyses the problem of large variation in adaptive learning rate in Adam-type algorithms at the initial stages of training and proposes a variant of RAdam, which fixes this issue and improves the stability and performance of optimization. The paper [7] presents the NosAdam algorithm, which gives more weight to the previous gradients in calculating the adaptive learning rate, which contributes to better convergence and stability of learning. The paper [8] investigates the choice of optimal learning rates for adaptive algorithms such as Adam and AMSGrad and their impact on the convergence and performance of deep neural networks.

In the paper [9], we present a new and complex framework for analyzing the convergence properties of Adam. This structure offers a universal approach to establishing Adam convergence. Adam has been shown to reach non-asymptotic limits of sample complexity similar to SGD.

The purpose of this work is to study maps of dynamic learning modes of a multilayer neural network using the Adam optimization method for the target learning error function.

The hyperparameters β_1 and β_2 were scanned within the interval [0; 0.999] using a uniform scanning step. The periodicity detection parameter q was chosen as q = 8 or q = 16, depending on the experiment. The neural network was trained using the mean squared error (MSE) loss function. Sigmoid activation was used in the output layer, while standard nonlinear activation functions were used in hidden layers. Weights were initialized randomly with a fixed random seed to ensure reproducibility. A regime was classified as chaotic if no periodicity up to order q was detected and convergence was absent within the observation window.

MATERIALS AND METHODS

The construction of dynamic mode maps was carried out in the coordinates of the hyperparameters at different values of the learning rate, the number of hidden layers, and the number of neurons in them and the number of iterations. Under these conditions, the learning error value of each neuron was analyzed in comparison with the others. The construction of a multilayer neural network was performed in the Python software environment. An array of printed digits represented as a set of zeros and ones with a dimension of 4×7 was used as the dataset. The sample contained 5 options for representing each digit. The sample for each digit contained variants of distortions of this digit. The amount of distortion was 5%. The construction of maps of dynamic modes was carried out according to the method described in the paper [10] and was as follows.

It is known that two-dimensional mappings can be specified by recurrence relations of the form:

$$\begin{aligned} x_{n+1} &= f(x_n, y_n), \\ y_{n+1} &= g(x_n, y_n). \end{aligned} \tag{6}$$

When building maps of dynamic modes, the function $f(x_n, y_n)$ was a function describing the learning error of a neuron in the output layer on the n-th neuron. The function $g(x_n, y_n)$ was a function describing the learning error of a neuron in the output layer on the $(n-1)$-th neuron.

The following algorithm was used to build maps of dynamic modes.

For some initial pair (x_0, y_0) at some pair of values (β_1, β_2), according to Eq. (6), a set of iterative values $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.

A few q last points stand out. For example, 8 or 16. q is the number that will denote the maximum period that can be determined using this algorithm. The selected points form a set of $(x_{n-q+1}, y_{n-q+1}), (x_{n-q+2}, y_{n-q+2}), \ldots, (x_n, y_n)$.

Working with a set of points that has 2 coordinates is not very convenient in our case. To do this, you need to replace 2 coordinates with a single number. For example, replace with a square the modulus of the vector that corresponds to this coordinate. That is, instead of a point (x_i, y_i), the value $d_i = x_i^2 + y_i^2$ is used. The result is a set of $d = \{d_1, d_2, \ldots\}$. For further actions, it will be necessary that d_i displays a specific value (x_i, y_i) and does not allow duplicates. But this cannot be achieved if the points form a certain symmetrical figure, for example, a circle – the distances to the middle of the center of the coordinates of which will be the same. Therefore, it makes sense to introduce some asymmetry into the above operation. For example, like this: $d_i = (x_i + k_1)^2 + (y_i + k_2)^2$.

If the last, q element, coincides with $q-1$, then it is stated that the resulting set d has a period of 1, which means that with the parameters (β_1, β_2), the system has a limit point. until the period is detected. That is, where a situation is reached when none of the numbers of the set d will coincide with the last element. In this case, the presence of chaos is asserted. Although in fact it may turn out to be a period of higher order or the system has not yet moved to stable behavior, and we ourselves have the right to choose with what accuracy the behavior of the system is considered. In a situation where the period has not been detected, it is wise to write it as zero. The above method of determining the period can be called the method of simple comparison or the method of convergence of periodicities.

Next, it is necessary to save the obtained result. In pre-created 3 arrays, values are stored according to the following rule: in the first array – the value of β_1, in the second – the value of β_2, and, finally, in the third – the value of the obtained period.

All the previous points are worked out at other values of β_1, β_2. It is necessary to go through all possible combinations (β_1, β_2) from the interval, going over with a certain step.

The last stage is the display of the image. Points corresponding to different period values must be displayed using different colors. To do this, you can use check loops or logical operations.

When building dynamic mode maps, we selected only the first fourteen possible periodicities (red – 1, orange – 2, yellow – 3, green – 4, cyan – 5, blue – 6, violet – 7, navy (#000080) – 8, light shade of purple-blue (#9370DB) – 9, dark -orchid (#9932CC) – 10, medium light shade of purple (#DDA0DD) – 11, purple red (#C71585) – 12, midnight blue (#191970) – 13, purple (#DB7093) – 14, white (white) – all others). The black color indicates the absence of any periodicity.

RESULTS AND DISCUSSION

Fig. 1 shows maps of dynamic training modes of a three-layer neural network for printed digits with 28 neurons in each layer, training step at 10 iterations. Maps of dynamic neuronal learning modes are characterized mainly by the existence of the first periodicity (red color) in the entire spatial area of change. Since the first periodicity is associated with the largest gradient change, this indicates that under these conditions of neural network training, its underfitting can be traced. Only in the spatial region of changes (0.7–0.9999) and (0–0.2) there is an appearance of periodicity with a longer period (tripling and above), that is, a decrease in the gradient of weight correction, and therefore in the speed of learning. Therefore, under these conditions (during the training step and 10 iterations), the neural network is in underfitting mode.

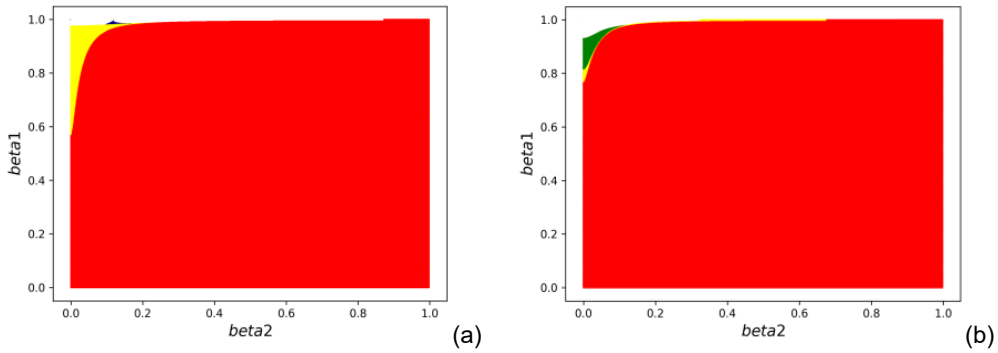

Fig. 1. Dynamic regime maps of a three-layer neural network trained on printed digits, with 28 neurons per layer, learning rate $\alpha = 0.001$, and 10 training iterations: (a) digit “0”; (b) digit “1”, illustrating the dependence of learning dynamics on the input pattern

Fig. 2 shows maps of dynamic learning modes of the neural network, provided that the learning rate (α) is equal to 0.1, with 10 iterations. With small values of (0–0.2) and (0–0.2), the existence of periodicity with a longer period can be traced.

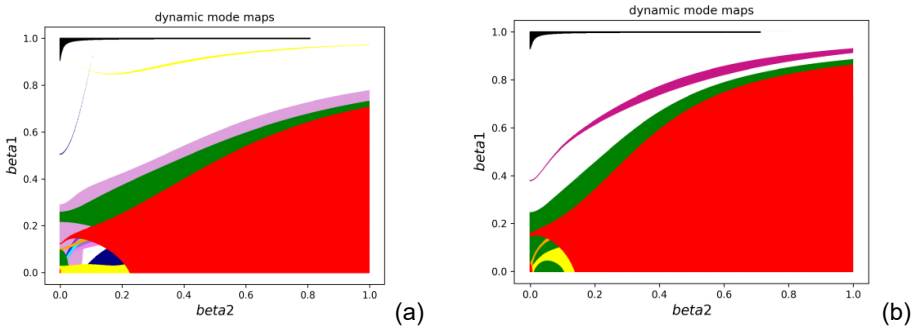

Fig. 2. Dynamic regime maps of a three-layer neural network trained on printed digits, with 28 neurons per layer, learning rate $\alpha = 0.1$, and 10 training iterations: (a) digit “0”; (b) digit “1”, illustrating the dependence of learning dynamics on the input pattern

The update of weights is more sensitive to gradients, and in this interval of changes in optimization parameters, an uneven learning process is observed. In this range of changes β_1 and β_2, the emergence and increase of the range of periodicity with the smallest period can be traced. That is, training takes place with a larger gradient of weight correction. With an increase in optimization parameters β_1 and β_2, there is an expansion of the spatial area of existence of periodicity with the shortest period. At values of β_1 (0.2–0.6) and β_2 (0.2–0.999), it (the periodicity with the smallest period) becomes decisive. That is, the learning process takes place at the same speed. Along with this, for some numbers ("0", "2", "4", "5", "6", "9"), there is a threefold (yellow) period, four (green) longer than the periodicity with the shortest period (red). At $\beta_1 > 0.9$ and $\beta_2 > 0.9$, according to the authors, there is a transition to periodicities with a period longer than the existing largest periodicity (white palette (white) - all other periodicities). Therefore, high values of β_1 and β_2 stimulate a smooth update of learning parameters with a transition to insignificant, large gradients of weight correction, or practically to their absence. That is, under these conditions $\beta_1 > 0.9$ and $\beta_2 > 0.9$ (white palette (white) - all other periodicities), there may be no process of correcting the weights. Based on the above, the following conclusion can be drawn. Provided that the control of inertia in the direction of the gradients is negligible ($\beta_1 < 0.4$) and the control of the "adaptive" learning rate is insufficient ($\beta_2 < 0.2$), it reacts quickly to gradient changes, resulting in unstable learning and oscillations. Under the same conditions, the speed adapts to new gradients, and this increases the risk of fluctuations (there is no change in the colors specified in the method, and therefore periodicity.

It is known [11] that in the Adam (Adaptive Moment Estimation) method, the parameters β_1 and β_2 control the smoothing of first- and second-order moments, respectively. They affect the speed of updating weights and the stability of learning. β_1 (usually 0.9) determines the smoothing of the first moment (mean gradient), expression (1.1)

A high β_1 (for example, 0.4–0.99) adds "inertia" to the update of the weights, which helps to avoid sudden changes in the direction of movement, as well as smoothes gradients, stabilizes learning.

Low β_1 results in less anti-aliasing effect, making weight updates more sensitive to current gradients, and making learning faster but less stable.

β_2 determines the smoothing of the second moment (the mean square of the gradients), the expression (1.2).

A high β_2 allows for smooth parameter updates, but can cause excessive anti-aliasing, which slows learning, as well as smoother refresh but slower adaptation of learning speed [12].

Low β_2 makes the model more sensitive to gradient changes, which can help when gradients change rapidly, as well as faster adaptation, but greater risk of instability.

Usually, the default is β_1 = 0.9, β_2 = 0.999 [11]. If the gradients change very quickly, you can reduce β_2. If the weights are very unstable, you can increase β_1.

So, based on **Fig. 2**, at $\beta_1 > 0.9$ and $\beta_2 > 0.9$, there is a smooth update of parameters, which is accompanied by a slowdown in learning or practically its absence. To identify the learning dynamics processes of a neural network under the condition $\beta_1 > 0.9$ and $\beta_2 > 0.9$, we will consider the influence of the number of iterations, the learning rate, the number of hidden layers and neurons in them on the learning process.

Influence of the number of iterations.

Fig. 3 shows the dependence of dynamic mode maps on the number of iterations. The obtained maps of dynamic modes of neuronal learning indicate that the process of difference in the learning of one neuron in relation to another is not very sensitive to the number of iterations.

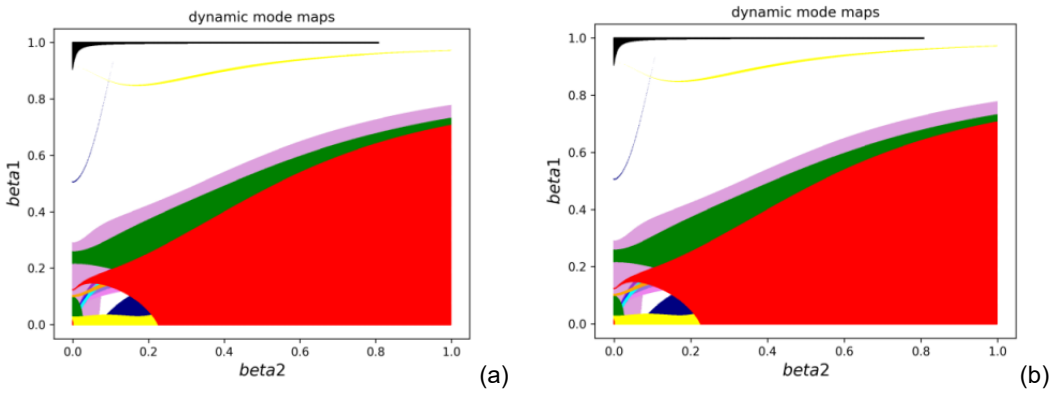

Fig. 3. Dynamic regime maps of a three-layer neural network trained on printed digits (digit "0"), with 28 neurons per layer and learning rate α = 0.1: (a) 3 training iterations; (b) 100 training iterations, illustrating the effect of iteration number on learning dynamics

Although it is known [13] that the number of iterations directly affects the role of the β_1 and β_2 parameters in the Adam method, especially because of their role in bias correction in the early stages of learning. Since the evaluation moments of m_t and v_t are calculated recursively, they are shifted towards zero in the first iterations. To compensate for this, bias correction is applied to expressions (1.3) and (1.4). This correction has a strong effect in the early stages of learning (small t). Since β_1 and β_2 are initially close to zero, the denominator is small, which amplifies updates in early iterations.

With a large number of iterations ($t \to \infty$), correction becomes almost unnecessary, since $1 - \beta_1^t \approx 1$ and $1 - \beta_2^t \approx 1$. On small iterations, low values β_1 make the weights update sensitive to gradient changes. High β_1 gives greater inertia, which stabilizes the

update. On large iterations, the effect of choosing β_1 is smoothed out because the middle gradient accumulates enough information.

If the training is short (a few iterations), it is worth reducing β_1 to respond faster to changes. If the training is long, high β_1 (0.9–0.99) improves stability.

On small iterations, high β_2 makes updating weights very careful, which can slow adaptation. On large iterations: v_t (RMS gradients) is already stabilized, and β_2 is affected less. If the learning is short (a few iterations), you can reduce β_2 it so that the adaptation of the learning speed is faster. If the training is long, high β_2 will help to avoid fluctuations in the learning speed.

Therefore, in the early iterations, the bias correction plays an important role by enhancing the effect of the gradients. In the later iterations, the effect of choosing β_1 and β_2 becomes less critical, and Adam works stably. If the training is short (a few iterations), it is possible to reduce β_1 and β_2 so that the network adapts faster. If the training is long, the standard values are β_1 = 0.9, β_2 = 0.9999, and usually work well [12, 13].

The difference between our results and the generally accepted statements about the influence of the number of iterations on the role of the parameters β_1 and β_2 in the Adam method may be related to the method for determining recurrence relationships, Eq. (1.6).

Impact of the learning rate.

The learning rate (α) in the Adam optimizer significantly affects the learning process of the neural network [3]. This manifests itself as follows.

The Large learning rate (α).

A quick update of the weights, which can accelerate convergence, while the network may "jump" the optimum and not stabilize. The error can fluctuate or even increase.

Small learning rate (α).

Learning is stable and smooth. Very slow convergence. Can get stuck at a local minimum.

The optimal value of the learning rate would be the balance between speed and stability. Normally, Adam works well at $\alpha = 0.001$ (recommended value for convolutional neural networks). A reduction in the learning rate can be used to improve convergence [3].

Choosing the right training step is critical for the neural network to train effectively.

According to the work [2], an increase in the learning rate (α) can lead to overfitting of the neural network, which ultimately leads to a chaotic learning mode of the neural network (for a three-layer neural network). **Fig. 4 a–c** shows maps of dynamic modes under the condition of $\alpha = 0.4, \alpha = 0.6, \alpha = 0.9$. According to **Fig. 4a**, an increase in the learning rate caused a decrease in the spatial region in the coordinates β_1 and β_2 the existence of periodicity. In particular, first of all, periodicity with the shortest period. Along with this, the appearance of new periodicities and the expansion of existing periodicities with an increased period take place. This indicates a change in the speed of learning in the direction of its decrease. In our opinion, this is due to an increase in the speed of weight correction and the rapid achievement of a local minimum.

Let's consider it in more detail. Under the condition of $\beta_1 < 0.1$ and $\beta_2 < 0.9$, there is practically no learning process. A similar situation is observed for $0.4 < \beta_1 < 0.99$ and $0 < \beta_2 < 0.99$. The origin of periodicity with different periods can be traced in the spatial region of changes in optimization parameters $0 < \beta_1 < 0.3$ and $0 < \beta_2 < 0.2$. A further increase in the β_2 parameter is an increase in the spatial regions of periodicity. In particular, this is well traced for the smallest periodicity, and for the periodicity of which

the period has increased by three, four, five, and six times (**Fig. 4a**). Ideally, the pattern of neural network learning when using an optimizer should be as follows.

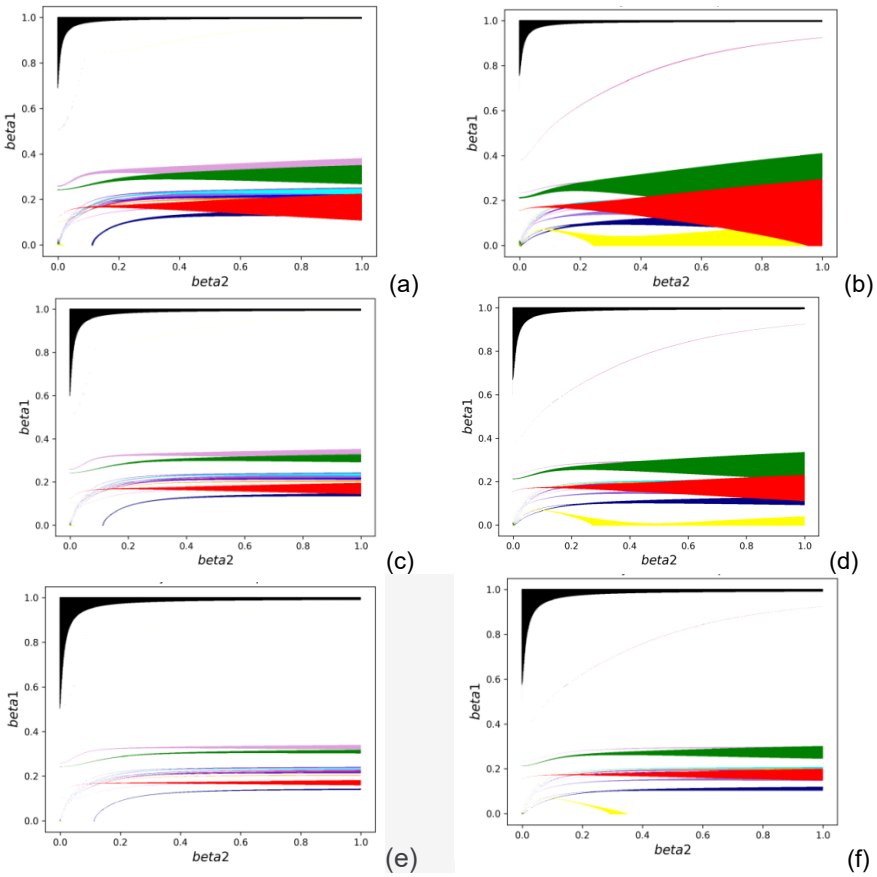

Fig. 4. Dynamic regime maps of a three-layer neural network trained on printed digits, with 28 neurons per layer and 10 training iterations: (a), (b) learning rate α = 0.4; (c), (d) α = 0.6; (e), (f) α = 0.9; for each pair, (a), (c), (e) correspond to digit "0" and (b), (d), (f) to digit "1", illustrating the effect of increasing learning rate on learning dynamics and the emergence of complex and chaotic regimes

At the beginning, as a result of large gradients, the highest learning speed should be traced, in our case, the correction Weights. As we approach the global minimum, the magnitude of the gradients decreases, so there will be an increase in periodicity. That is, ideally, we should observe the following sequence of color palettes with increasing values of optimization parameters β_1 and β_2: red – 1, orange – 2, yellow – 3, green – 4, cyan – 5, blue – 6, violet – 7, navy (#000080) – 8, light shade of purple-blue (#9370DB) – 9, dark – orchid (#9932CC) – 10, medium light shade of purple (#DDA0DD) – 11, purple red (#C71585) – 12, midnight blue (#191970) – 13, purple (#DB7093) – 14, white (white) – all others). That is, the first periodicity (red color) with the smallest period is described by the largest changes in the learning error gradient. Other, subsequent periodicities are characterized by a longer period multiple of the period of the first periodicity. In fact, according to **Fig. 4a**, at the initial stage, learning begins with not the biggest change in the target learning function of the neural network. That is, learning begins with periodicities, the period of which is three or six times longer than the period of the first periodicity. This indicates that learning does not begin with the highest learning speed, but three to six times less than the maximum possible. The sequence of periodicities with

an increase in the value of β_1 in the entire range of changes in the optimization parameter β_2 is also far from the ideal sequence of periodicities.

That is, for most digits, except for "1", "2", "3", "4", "8", the periodicity with a longer period is traced first, before that with a decreasing period. Therefore, there is a non-equilibrium learning process, which is accompanied by a different sequence of periodicities, i.e. uneven learning with different changes in speeds when changing parameters. In addition, it should be noted an increase in the spatial area of changes in the parameters β_1 and β_2, where periodicity is traced, the period of which is 14 times longer than the period of the first periodicity. In this area, chaotic fluctuations in the speed of neurons learning can also occur.

A further increase in the learning rate $\alpha = 0.6$ (**Fig. 4b**) is similar to that for **Fig. 4a**.

The sequence of periodicity with increasing magnitude β_1 in the entire range of changes in the optimization parameter β_2 is also far from its ideal sequence. This indicates a non-equilibrium learning process, which is accompanied by a different sequence of periodicities, i.e. uneven learning with different changes in speeds when optimizing parameters change. In addition, there is an increase in the spatial area of changes in parameters β_1 and β_2, and where periodicity is traced, the period of which is 14 times longer than the period of the first periodicity.

Fig. 4c shows maps of the dynamic learning modes of a neural network during a learning rate $\alpha = 0.9$. The resulting maps are dynamic, provided $\alpha = 0.9$, indicate the practical absence of the neural network learning process. The observed periodicities have an insignificant spatial area of existence and undergo their non-sequential change (with a decrease in the period). That is, the learning process is almost non-existent. It is known [2] that with a given value of the learning rate in multilayer neural networks, the existence of a chaotic learning mode can be traced, which is due to the overfitting of neurons. The optimizer's attempt to stabilize the learning process leads to the existence of a number of periodicities with a narrow area of their existence, which can be traced in **Fig. 4b** and **Fig. 4c**.

Additional studies of dynamic mode maps have been carried out, under these conditions, at different values ε (ε determines the existence of periodicities with a value of ε), testified to the absence of any periodicity in the spatial region, which is described by the existence of white. This state of the neural network can be described as a chaotic mode of the neural network learning process. This conclusion follows from the algorithm for building maps of dynamic modes using the method of convergence of periodicity. Namely, it is known that the chaotic regime is characterized by the fact that small changes in the initial conditions lead to radically different results. Under such conditions, periodicities occur that are not convergent, and the amplitude of such fluctuations can vary sharply. Therefore, in this method, a limit was set on the magnification of the learning error, and on the map of dynamic modes, this area is indicated in black. An area that is represented in white can be described as a region in which there are periodicities with a period greater than fourteen times the periodicity with the smallest period, zero period, or chaotic. The chaotic oscillations existing in this area (indicated in white) are amplitudes smaller in magnitude than in the specified black region. Therefore, in the white area of dynamic mode maps, a chaotic learning mode occurs with an amplitude less than the specified value in the black area.

Influence of the number of neurons

It is known [3–8] that the change in the number of neurons in the network affects the stability and efficiency of the neural network learning process.

If the number of neurons is small, Adam works stably, because there are fewer parameters and fewer gradient fluctuations. The model learns quickly. Possible

underfitting is insufficient expressiveness of the model. Adam updates the weights, but they are not enough for the model to summarize the data well.

You can use a larger learning rate (α), because the model is less sensitive to large updates. Gradients are more stable, so moments work efficiently [3].

If the number of neurons is large, the model can study complex patterns. The gradients become noisy, which can make it difficult for Adam to work. The model can be overtrained. Learning becomes slower due to more scale updates. Under these conditions, it is necessary to reduce the training step (α) to avoid sudden changes in large models. Adam adapts to gradient changes but can match more slowly due to the large size of the parameters.

So, when there are fewer neurons, Adam works more stably, and a larger learning rate can be used. More neurons – Adam can be unstable due to variable gradients, requiring a lower learning rate [3].

Fig. 5a–j presents dynamic regime maps for different numbers of neurons in the network. **Figures 5a–d** correspond to configurations with fewer neurons than the input dimensionality, whereas **Fig. 5e–j** illustrate cases with a larger number of neurons. For the configuration with 20 neurons per layer (i.e., fewer than the number of input features), the initial variation of the hyperparameters β_1 and $\beta_2$is associated with the emergence of first-order periodicity. As β_1 and β_2 increase, this periodicity expands and gradually transitions to higher-order periodic regimes. Notably, the observed periodicities may exceed the fundamental period by more than a factor of five. Thus, for small values of the optimization parameters ($\beta_1 < 0.2$ and $\beta_2 < 0.2$), the learning process is characterized by the dominance of low-order periodic behavior.

Learning is mostly characterized by an uneven learning speed. At low values of β_1 ($\beta_1 < 0.2$) and β_2 ($\beta_2 < 0.2$), rapid adaptation takes place, but with greater instability. Lower values of β_2 are accompanied by a smaller smoothing effect, and the update of weights is painfully sensitive to current gradients. A further increase in the optimization parameters stimulates the presence of the highest learning rate, the interval of existence of which mainly depends on the parameter β_1. That is, β_1 it affects the rate of renewal of weights and the stability of learning. At higher values of β_2, the intervals of the existence of periodicities are larger. This indicates that β_2 it causes excessive smoothing, and this results in homogeneous learning. This, in particular, is indicated by the sequence of existing periodicities. That is, according to the first periodicity, the presence of periodicity is observed, a period that significantly exceeds (five or more times) the period of the first periodicity. A further increase in β_1 and β_2 causes the disappearance of periodicity, which indicates the transition of the neural network to the overfitting mode or to the chaotic mode. Therefore, a sharp transition to the periodicity with the longest period indicates the presence of a smoothing process, which provokes a decrease in the learning speed and expansion of the spatial areas of the periodicity of periodicity on the map of dynamic modes.

A further decrease in the number of neurons (up to 18, **Fig. 5c-d**) is accompanied by the separation of the spatial region of changes β_1 ($\beta_1 < 0.4$) and β_2 ($\beta_2 < 0.4$), where the existence of a significant number of periodicities, which indicates the existence of a number of learning speeds. Further increase in the value of parameters β_1 and β_2 is accompanied by the expansion of the spatial regions of the first, second, and third periodicity. This indicates a gradual transition to a decrease in the value of the learning speed (periodicity determined by red, yellow, and orange). The next increase in the value of β_1 and β_2 contributes to the appearance of periodicity with an even longer period, which indicates a further decrease in the learning rate. In addition, there is a transition to the absence of any periodicity (areas of white color), which in some places are separated by periodicities with a long period. This behavior of dynamic mode maps at large values

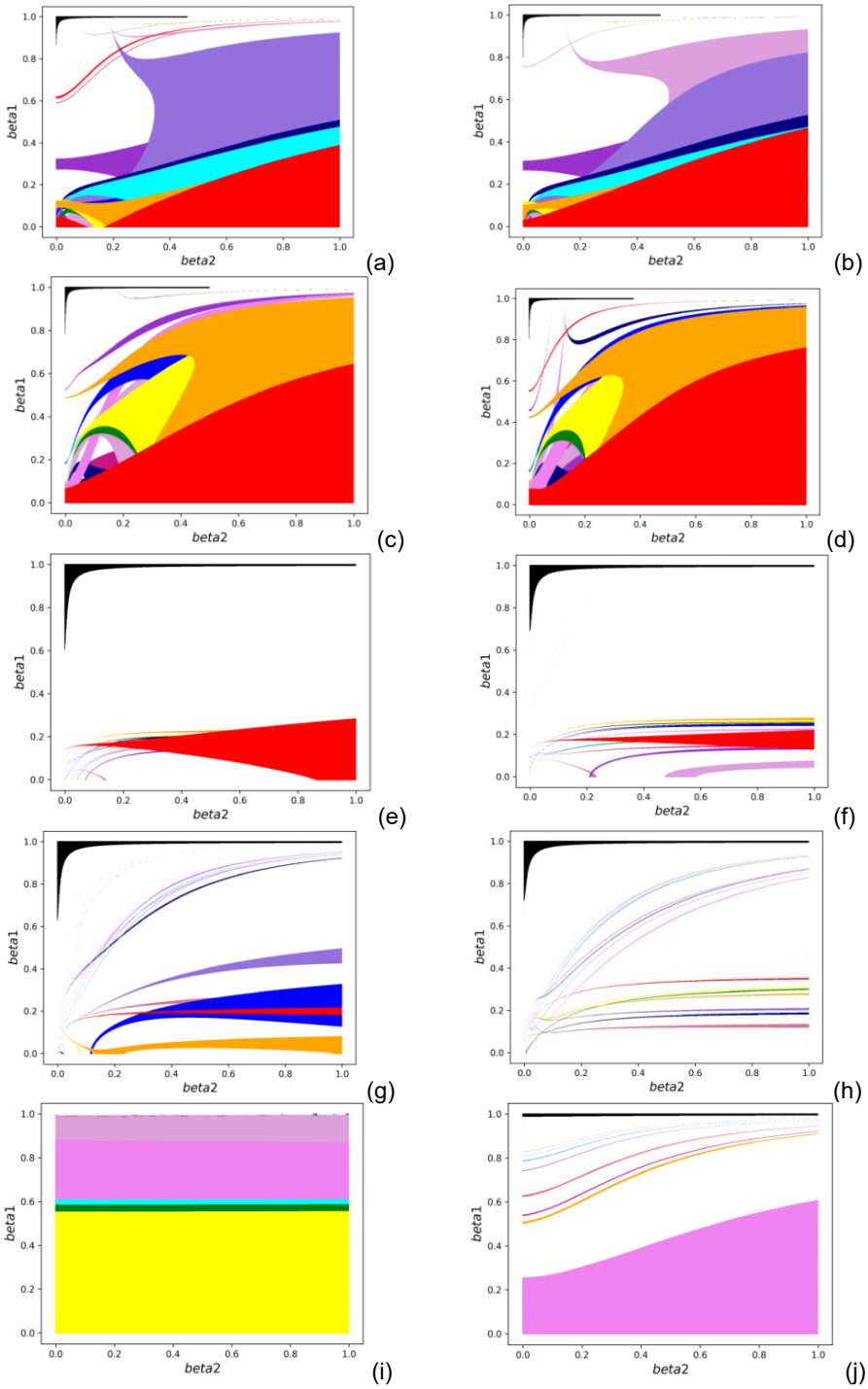

Fig. 5. Dynamic regime maps of a three-layer neural network trained on printed digits, evaluated over 10 training iterations: (a), (b) 20 neurons per layer, learning rate $\alpha = 0.1$; (c), (d) 18 neurons per layer, $\alpha = 0.1$; (e), (f) 50 neurons per layer, $\alpha = 0.4$; (g), (h) 75 neurons per layer, $\alpha = 0.4$; (i), (j) 100 neurons per layer, $\alpha = 0.4$; for each pair, the left panel corresponds to digit “0” and the right panel to digit “1”, illustrating the effect of network width on learning dynamics

of β_1 and β_2 indicates a significant decrease in the learning rate or its absence. That is, at large values of β_1 and β_2 reaches a global minimum, and neural network training can go into chaotic learning mode. Wedging of single periodicities with a long period may indicate the presence of a learning process confirming the existence of transparency windows on the branching diagram [2].

A slight further decrease in the number of neurons (up to 15 neurons) is accompanied by a lack of periodicity on the maps of dynamic learning modes of the neural network. This indicates the absence of a neuronal learning process. Therefore, a decrease in the number of neurons in comparison with the number of input variables leads to an improvement in the learning process in the early stages, making it more monotonous in terms of learning speed.

Let's consider how the process of increasing the number of neurons affects the dynamics of the learning speed. **Fig. 5e–j** shows maps of the dynamic modes of training of the neural network with an increase in the number of neurons in each layer (50, 75, 100 neurons). Choosing a learning rate equal α = 0.4 is due to the fact that, as shown in the paper [2], this training step corresponds to the optimal learning error of a three-layer neural network, with 28 neurons in each layer. An increase in the number of neurons is accompanied by a narrowing of the spatial areas of the existence of periodicities, the appearance of periodicities with longer periods. With 50 neurons (**Fig. 5e-f**), there is a sharp transition to periodicity with longer periods and to the mode of absence of the learning process. That is, there is a sharp decrease in the speed of learning and the transition to the mode of its absence. An increase in the number of neurons to 75 (**Fig. 5g-h**) leads to the separation of spatial areas of the existence of periodicity by areas of their absence, which indicates the appearance of a heterogeneous process of neuronal learning A further increase in the number of neurons (100 neurons, **Fig. 5i-j**) is accompanied by an increase in the heterogeneous process of neuronal learning, and possibly the appearance of a chaotic learning mode. This, in particular, is indicated by the results of work [2] on the influence of the number of neurons on the learning error of a three-layer neural network.

Consider how the number of hidden layers affects the performance of the Adam optimizer. **Fig. 6a–d** shows the influence of the number of hidden layers and the dynamics of the learning speed of the neural network.

An increase in the number of layers of a neural network from three to five is accompanied by a spatial expansion of the existence of periodicity with a shorter period, which indicates the existence of a slow learning process, compared to a three-layer neural network. At the same time, there is a decrease in the moment parameter ($\beta_1 < 0.4$), at which there is a more significant change in the learning speed. At $\beta_1 > 0.4$, the learning of the neural network goes into the mode of practically its absence. Wedging of single periodicities with a long period (**Fig. 6a-b**) may indicate the presence of a learning process with a low learning speed, which indicates the existence of a heterogeneous learning process. An increase in hidden layers from five to seven leads to the disappearance of almost periodicity with a short period (areas of red, orange, and yellow), and the appearance of periodicity with a longer period. That is, an increase in the number of layers of the neural network at this stage leads to a decrease in the learning speed of the neural network, and an increase in the heterogeneity of the learning process not only at $\beta_1 > 0.4$. Therefore, the learning process becomes more heterogeneous. This, according to the authors, leads to a decrease in learning speed. That is, the Adam optimization method becomes slower due to the fact that the gradients become noisier.

It is known [2] that the optimizer Adam adaptively changes the learning rate for each network parameter, taking into account the average and variance of the gradients.

Increasing the number of hidden layers affects the dynamics of gradients and, accordingly; the behavior of Adam.

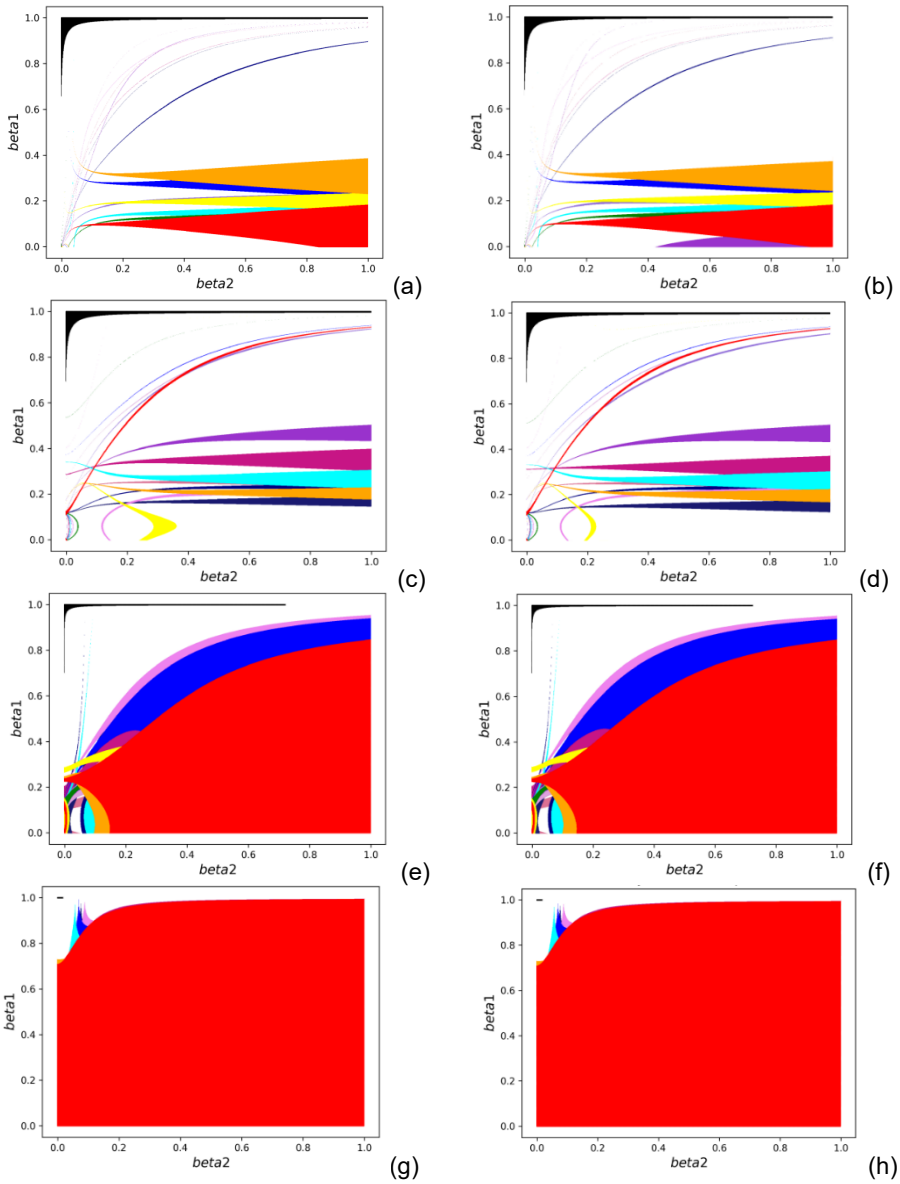

Fig. 6. Dynamic regime maps of a multilayer neural network trained on printed digits, with 28 neurons per layer, learning rate α = 0.4, and 10 training iterations: (a), (b) five-layer network; (c), (d) seven-layer network; (e), (f) nine-layer network; (g), (h) twelve-layer network; for each pair, the left panel corresponds to digit "0" and the right panel to digit "1", illustrating the effect of network depth on learning dynamics

Few hidden layers (fine mesh). A simpler network is easier to optimize. Gradients change predictably, Adam controls them well.

Lots of hidden layers (deep web). Better at detecting complex patterns in data. May suffer from decay or explosion of gradients. Increasing the volume of calculations, Adam may run more slowly. Gradients can become noisy, making it difficult to learn.

Shown in **Fig. 6e–h** are maps of dynamic modes of the nine- and twelve-layer non-grid. Their maps of dynamic modes are radically different from maps of dynamic modes

with fewer layers (**Fig. 6a–d**). This difference lies in the presence of periodicity with the shortest period, which indicates an increase in the speed of learning. Taking into account the results shown in Fig. 6 g, h, according to the authors, this is due to the processes of undertraining of the neural network with a small number of iterations. Further studies with more hidden layers show a slowdown in the learning process.

So, the paper considers maps of dynamic learning modes of the neural network depending on the number of iterations, the learning rate, the number of neurons, and the number of neural network layers. The construction of maps of dynamic modes was carried out in the coordinates of hyperparameter optimization β_1 and β_2, and since these parameters, according to the work [2], determine the learning speed of the neural network. In the role of recurrent correlations was the learning error of the neuron and its neighbor. This is the peculiarity of the task.

The relationship was intended to determine the heterogeneity of the learning of one neuron in relation to another. Considering the influence of iterations (3, 5, 10, 100 iterations) on the dynamics of neural network learning, we did not establish any features of neuron training in relation to each other. As for the impact of the learning rate on the learning process, depending on its magnitude, there are modes of underfitting, satisfactory training (when there is no overfitting of the neurons) and the mode of overfitting, which degenerates into a chaotic one.

Let us now consider these learning modes in the coordinates of the hyperparameters of optimization β_1 and β_2. In the mode of underfitting (in particular, when $\alpha = 0.001$) in the range $0 < \beta_1 < 0.99$ and $0 < \beta_2 < 0.2$, a monotonic decrease in the learning rate is observed with an increase in the value of β_1. With higher values of β_2, $\beta_2 > 0.2$, the appearance of the underfitting process is traced in 10 iterations. It is known that a high value β_2 leads to a significant smoothing of correction gradients, which leads to a slowdown in the learning process. In the mode of satisfactory learning (in particular at $\alpha = 0.1$ and $\alpha = 0.4$), the neural network achieves a global minimum faster and transitions to a mode of practically no learning process. In the shift interval $0 < \beta_1 < 0.4$ and $0 < \beta_2 < 0.2$ traces the existence of abrupt and heterogeneous changes in the learning rate. An increase in the magnitude of the second moment (β_2) is accompanied by a smoothing of gradients, which leads to a monotonous decrease in the learning rate and the achievement of a global minimum. Under these conditions, if there is a heterogeneity of learning, it manifests itself in a consistent decrease in the speed of learning.

Switching to overfitting mode (In particular, when $= 0.6$, $\alpha = 0.9$), then at the coordinates β_1 and β_2 it manifests itself in heterogeneous learning, which is described both in a non-monotonic change in the learning rate and the existence of intervals where it is practically absent. It should be especially noted that at small values of β_1 in a satisfactory learning mode ($0.1 < \alpha < 0.5$), the appearance of a heterogeneous learning process, or practically no learning, can be traced. This is due to the magnitude β_1 and β_2 are much smaller than one. Since the evaluation of moments m_t and v_t are calculated recursively, they are shifted towards zero in the first iterations. To compensate for this, displacement correction (expressions (1.3) and (1.4)). This correction has a strong effect on early stages of training (small t), since by default the value β_1 and β_2, in Adam method, respectively, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. In our case, the parameters β_1 and β_2 are variable quantities and vary in the range 0, up to 0.999. That is, at small values, especially β_1, it does not occur Strengthening Training Updates. An increase in value β_2, with lower values of the learning rate, in the mode of satisfactory learning improves the situation somewhat, contributing to the learning process.

A change in the number of neurons in a multilayer neural network does not unambiguously affect the learning process when they increase compared to their decrease relative to the number of input values. On maps of dynamic modes in coordinates β_1 and β_2, this manifests itself as follows. If the number of neurons is less than the number of input values, Adam works stably. This is most likely due to fewer parameters and a lower value of gradient oscillations. The model learns quickly. The gradients are more stable, so the moments m_t and v_t work efficiently. The change in the learning rate in the coordinates β_1 and β_2 becomes more monotonous as the number of neurons decreases. In the interval of changes $0.1 < \beta_1 < 0.4$ and $0 < \beta_2 < 0.3$, the existence of sharp and heterogeneous changes in the learning speed is traced. An increase in the magnitude of the smoothing of the second moment (β_2) contributes to the smoothing of gradients, which leads to a monotonous decrease in the learning speed and the achievement of a global minimum. The model is less sensitive to large updates. Provided that the number of neurons is greater than the number of input variables, it contributes to the complication of the work of Adam. The constructed maps of dynamic modes testify to the overfitting of the neural network (50 neurons), with the transition to the chaotic mode of learning the neural network (75, 100 neurons). This is due to the noisiness of the gradients.

An increase in the number of layers is accompanied by an increase in the depth of the neural network, and therefore an increase in the complexity of its optimization. Built maps of dynamic modes in coordinates β_1 and β_2, indicate that with an increase in the number of hidden layers, at the initial stages (3÷7 layers), gradients become noisy, which complicates the learning process. Under these conditions, a heterogeneous learning process begins to be traced with the emergence of a chaotic learning mode. A further increase in the number of layers leads to the normalization of the learning process with the advent of the underfitting regime. The subsequent increase in the layers of the neural network contributes to the appearance of a satisfactory learning mode first, and then, with the transition to an overfitting mode, which turns into a chaotic one. At the same time, both attenuation modes and gradient explosions can be traced on dynamic mode maps.

The maps offer a compact diagnostic of Adam's stability across architecture and hyperparameters. Large β_1 and β_2 promote smooth updates but can stall progress; small β_1 and β_2 adapt rapidly but risk oscillations. Increasing network width/depth amplifies gradient noise, shrinking stable regions. These findings align with reports on optimizer sensitivity and the importance of tuning α and decay rates. Practically, the maps can guide the selection of (β_1, β_2, α) for a given budget of iterations and model size.

The obtained results should be considered as a qualitative framework for studying the nonlinear dynamics of the Adam optimizer rather than as a direct benchmark evaluation of modern datasets. Extension of the proposed approach to larger-scale datasets and architectures represents an important direction for future research.

CONCLUSION

Therefore, training a multilayer neural network with the Adam optimization method is sensitive to both the learning rate, the number of neurons in the layer, and the number of layers of the neural network. Changing these parameters leads to the emergence of several neural network training modes. In particular, to the mode of undertraining, satisfactory training, and the mode of overfitting, which can turn into a chaotic mode.

Recurrence-based dynamic mode maps in (β_1, β_2) reveal distinct Adam learning regimes and their dependence on α, iterations, width, and depth. This approach helps

identify stable operating regions and anticipate chaotic behavior in multilayer neural networks.

The source code is available in the GitHub repository at the following link: incom2025**neuro_map** (https://github.com/incom2025/neuro_map).

ACKNOWLEDGMENTS AND FUNDING SOURCES

The authors received no financial support for the research, writing, and/or publication of this article.

COMPLIANCE WITH ETHICAL STANDARDS

The authors declare that the research was conducted in the absence of any conflict of interest.

AUTHOR CONTRIBUTIONS

Conceptualization, [S.S., L.P., V.B.]; formal analysis, [S.S., I.K., Y.S.,].; investigation, [I.K.]; resources, [I.K.]; writing – original draft preparation, [S.S., I.K.]; writing – review and editing, [I.K., Y.S.].

All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] Kingma, D.P., Ba, J. Adam: A Method for Stochastic Optimisation. arXiv:1412.6980 (2014). https://doi.org/10.48550/arXiv.1412.6980

[2] Sveleba, S., Katerynchuk, I., Kuno, I., Karpa, I., Semotyuk, O., Shmyhelsky, Y., Sveleba, N., Kunyo, V. Chaotic states of a multilayer neural network. *Electronics and Information Technologies*. 2021;16:20–35. https://doi.org/10.30970/eli.16.3

[3] Loshchilov, I., Hutter, F. Decoupled Weight Decay Regularization. *arXiv:1711.05101* (2017). https://doi.org/10.48550/arXiv.1711.05101

[4] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J. On the Variance of the Adaptive Learning Rate and Beyond. *arXiv:1908.03265* (2019). https://doi.org/10.48550/arXiv.1908.03265

[5] Huang, H., Wang, C., Dong, B. Nostalgic Adam: Weighting more of the past gradients when designing the adaptive learning rate. *IJCAI* (2019) 2532–2539. https://doi.org/10.24963/ijcai.2019/355

[6] Iiduka, H. Appropriate Learning Rates of Adaptive Learning Rate Optimisation Algorithms for Training Deep Neural Networks. *arXiv:2002.09647* (2020). https://doi.org/10.48550/arXiv.2002.09647

[7] Reddi, S.J., Kale, S., Kumar, S. On the Convergence of Adam and Beyond. In: *Proceedings of the International Conference on Learning Representations* (2018). https://doi.org/10.48550/arXiv.1904.09237

[8] Jin, R., Li, X., Yu, Y., Wang, B. A comprehensive framework for analyzing the convergence of Adam: bridging the gap with SGD. *arXiv:2410.04458* (2025). https://doi.org/10.48550/arXiv.2410.04458

[9] Choi, D., Shallue, C.J., Nado, Z., Lee, J., Maddison, C.J., Dahl, G.E. On Empirical Comparisons of Optimizers for Deep Learning. *arXiv:1910.05446* (2020). https://doi.org/10.48550/arXiv.1910.05446

[10] Sivaprasad, P.T., Mai, F., Vogels, T., Jaggi, M., Fleuret, F. Optimiser Benchmarking Needs to Account for Hyperparameter Tuning. *arXiv:1910.11758* (2020). https://doi.org/10.48550/arXiv.1910.11758

[11] Zhou, Z., Zhang, Q., Lu, G., Wang, H., Zhang, W., Yu, Y. AdaShift: Decorrelation and Convergence of Adaptive Learning Rate Methods. *arXiv:1810.00143* (2019). https://doi.org/10.48550/arXiv.1810.00143

[12] Sveleba, S., Katerynchuk, I., Kuno, I., Shmyhelsky, Y., Gut, S. Programs for calculating the map of dynamic modes using a system with chaotic modes. *Electronics and Information Technologies*. 2025; 30: 137-154. https://doi.org/10.30970/eli.30.11

БАГАТОПАРАМЕТРИЧНІ ДИНАМІЧНІ КАРТИ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ, НАВЧЕНИХ ЗА ДОПОМОГОЮ ОПТИМІЗАТОРА ADAM

Сергій Свелеба[1]*, Іван Катеринчук[1], Ярослав Шмигельський[1], Люціян Пельц[2], Володимир Бригілевич[2], Назар Свелеба[3]
[1]Львівський національний університет імені Івана Франка
вул. Ген. Тарнавського, 107, 79017 Львів, Україна
[2]Державна вища школа технологій та економіки в Ярославі
вул. Чарнецького 16, 37-500 Ярослав, Польща.
[3]Приватний вищий навчальний заклад «Європейський університет»,
бульвар академіка Вернадського 16 В, Київ, Україна

АНОТАЦІЯ

Вступ. Розуміння того, як гіперпараметри оптимізатора Adam формують динаміку навчання багатошарових нейронних мереж, залишається відкритою проблемою, особливо в режимах, де навчання переходить від стабільної до хаотичної поведінки. Попередні дослідження показували, що швидкість навчання та складність мережі можуть зумовлювати недонавчання, задовільне навчання, перенавчання або навіть хаотичні стани; однак систематичної візуалізації цих режимів у просторі параметрів β_1–β_2 досі бракує.

Матеріали та методи. Побудовано карти динамічних режимів для багатошарової нейронної мережі, навченої на двійково закодованих зразках цифр. Динаміку навчання аналізували за допомогою рекурентного двовимірного відображення, що базується на похибці нейрона та його сусіда. Періодичності у фінальних q ітераціях визначали методом збіжності періодичностей, що дало змогу класифікувати стани навчання – від стабільних фіксованих точок до циклів високого порядку та хаосу. Карти генерували для різних значень швидкості навчання α, кількості ітерацій, числа нейронів у шарі та глибини мережі, при цьому β_1 і β_2 сканували в інтервалі [0; 0.999].

Результати та обговорення. Дослідження показало, що поведінка Adam є сильно чутливою до спільних значень β_1 і β_2. Малі швидкості навчання ($\alpha = 0.001$) переважно приводять до періодичності першого порядку, що вказує на недонавчання. Помірні швидкості ($\alpha = 0.1 - 0.4$) формують структуровані переходи від швидкого навчання до повільніших режимів із зростанням β_2, тоді як надмірно великі значення α (≥ 0.6) породжують фрагментовані, немонотонні структури, характерні для перенавчання та хаотичної динаміки. Збільшення кількості нейронів або шарів посилює шум у градієнтах, звужуючи області стабільності та розширюючи хаотичні зони. Навпаки, зменшення числа нейронів нижче розмірності вхідного простору приводить до більш однорідних траєкторій навчання зі згладженими переходами періодичності.

Висновки. Картографування динамічних режимів у просторі β_1–β_2 є потужним діагностичним інструментом для визначення стабільних, неефективних або хаотичних режимів навчання в нейронних мережах, що навчаються за допомогою оптимізатора Adam. Результати підтверджують, що Adam є високочутливим до

взаємодії гіперпараметрів та архітектури мережі, і що некоректні їх комбінації можуть спричинити хаотичне навчання навіть за помірних швидкостей навчання. Запропонована методологія формує нову основу для аналізу динаміки оптимізаторів та оптимального вибору гіперпараметрів у моделях глибинного навчання.

Ключові слова: оптимізатор Adam; динаміка навчання; карти періодичності; налаштування гіперпараметрів; хаотичне навчання; багатошарові нейронні мережі.

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
05 March, 2026	15 May, 2026	18 May, 2026	29 June 2026