# Neural ODE Machine Learning Method with Embedded Numerical Method

Andrey Televnoy, Sergei Ivanov, Tatiana Zudilova, Tatiana Voitiuk
ITMO National Research University (ITMO University)
Saint Petersburg, Russian Federation
adtelev@mail.ru, serg_ie@mail.ru, zudilova@ifmo.spb.ru, tanya_4ever@mail.ru

***Abstract*—In the study, the authors examine the application of the Neural ODE machine learning method to reduce the number of neural network layers in order to minimize the problem of fading gradients. It is proposed to implement in Neural ODE a new scheme of the numerical Runge-Kutta method developed by the authors. The mathematical substantiation of the proposed numerical method, which guarantees the accuracy of the third order, is presented in detail. A comparative analysis of the training and computation time for Neural ODE with other machine learning methods using a computational experiment. The results of the experiment show that the proposed modification of the Neural ODE can reduce the computation time and the network training time.**

## I. Introduction

Currently, many different machine learning problems are solved using neural networks. The topic of research carried out in this area shows that the authors focus on a more substantive study of the so-called deep learning. This principle uses readily available big data to solve various kinds of tasks that previously seemed overly complicated in terms of computational resources. Increasing the depth of traditional neural networks made it possible to significantly improve the quality of their work, but also led to a number of computational difficulties.

In just the past few years, many studies have been published that illustrate the relationship between deep neural networks and differential equations. The similarity of the behavior of neural networks with the mathematical apparatus of differential equations was established during similar studies.

A direct analogy can be drawn between the residual layers of a neural network and differential equations. Computing the output of the residual network block can be considered as one iteration of the Euler method to solve the differential equation. Neural Ordinary Differential Equations (Neural ODEs) [1] are a new class of models that consider the limit of this discretization step, naturally giving rise to an ODE that can be optimised via black-box ODE solvers. The main distinguishing feature of Neural ODE is that the computational model has a continuous number of layers, thus setting a time-continuous latent state transformation. This distinctive feature can be useful in applications where information is received periodically and the exact timing of information is important. Thus, this advantage of Neural ODE can improve the performance of prediction models in such applications. Another advantage of this approach is the fewer parameters compared to residual networks, since new parameters are not required for each layer.

The fading gradient problem is one of the main problems encountered in neural networks of great depth. It lies in the fact that during training, the gradient practically does not reach the input. This is because the gradient gets very close to zero during the backpropagation algorithm. In the final result, it can lead to the fact that the neural network will no longer be taught. In this case, the input weight will be almost impossible to change because this gradient practically does not exist, it is not there.

The solution to this problem can be a different principle of building a neural network. In this case, the so-called “residual layer” is used as the main building block. Adding skip-connections minimizes the negative impact of this problem. The gradient has the ability to go through not only a non-linear transformation of the layer, but also through an identity transformation. This approach solves the problem of damped gradients due to the fact that each layer of the neural network has access to the untransformed gradient during the backpropagation of the error.

Later, in the course of extended work with this approach, the scientists found that the iterative update of the latent state of the neural network is very similar to using the Euler method when solving ordinary differential equations. Thus, in the article [1], the authors proposed to parameterize the dynamics of the latent state using a neural network. In this case, obtaining the final hidden state requires the need to solve the banal Cauchy problem. One of the obvious advantages of this approach is that to solve such a problem, you can use various accurate and efficient numerical methods that allow you to explicitly set the required solution accuracy. In general, the work of the authors [1] made a fundamental contribution to the development of this area, since the authors additionally described the ways of applying the approach proposed by them.

For example, Neural ODE can also be applied to supervised learning problems. In this case, the residual neural networks are replaced by neural ODEs. The authors conducted an experiment comparing a neural network built using several residual layers and a neural network with one neural ODE. Experimental results showed that Neural ODE trained with conjugate state computation performed better. In addition, it has constant memory allocation costs.

It should be noted that the authors did not ignore the normal distribution problem. For example, in image processing tasks, it is usually too simple. The distribution density of real data is usually much more complex than a Gaussian. The authors proposed to solve this problem by successively applying simple reversible transformations to the model outputs [1].

The advantage of more efficient use of memory is one of the benefits of using Neural ODE. By varying the uniform grid, we can obtain an arbitrarily large number of residual blocks with total weights. Moreover, you can use different step sizes for training and testing. The number of calculations grows with the number of grid elements, and the amount of allocated memory remains constant. It is interesting that the results of the presented author's experiment showed that the quality of the solution with a smaller number of parameters is even better [1]. This advantage is also very important when working with real-time systems, where computations are limited in time.

In modern generative models, the problem of made collapsing is also acute. With it, the generator produces a limited number of different samples, while not even covering the entire training set. Such a neural network can only sample points, while not giving out their density. The computational model in [1] allows finding the density at the point of a new distribution in linear time. Moreover, the transformations have no restrictions.

In general, a lot of research is carried out with the aim of improving methods for training neural networks. For example, the authors of the article [2] proposed the so-called discrete-time method of successive approximations (MSA), based on Portnyagin's maximization principle. MSA also helps to minimize the problem of fading gradients to some extent. This method is illustrated by the example of its use for training a class of non-traditional networks, i.e. networks with discrete weights. The experimental results showed that MSA builds fairly original trained models in the case of ternary networks. The method also showed itself well in solving the optimization problem (in terms of the error rate). All this makes its use in some machine learning tasks attractive to some extent.

An attempt to solve the problem of gradient decay was also found in work [3]. The authors of this article, approaching the issue from a biological point of view, proposed their Least Mean Squares (LMS) method, based on the accumulation of information between different iterations. A characteristic feature of the experiment is the authors' attempt to consider an online learning case, in which learning occurs continuously and the tests are sequential. The results showed that interference in the learning process of a neural network depends on a number of different factors and has a serious impact on the entire concept of problem solving. The authors proposed experimental protocols to minimize such interventions. Protocols allow inferences to be drawn about the underlying learning rules by observing the behavior of a neural network throughout the learning process.

Like many other approaches to solving a particular problem, Neural ODE may have its own disadvantages. In [4], the authors concluded that there are functions that Neural ODE is unable to provide. To address this limitation, so-called Augmented Neural ODEs (ANODEs) were introduced. They expand the space in which the ODE is solved. Thereby helping the model to use extra dimensions to study higher-order functions. Authors believe Augmented Neural ODEs are more empirically robust and have lower computational costs compared to traditional Neural ODEs. ANODEs overcome the representational weaknesses of NODEs while maintaining all their attractive properties. However, the authors noticed that ANODEs, when working with excessively large augmented dimensions, “the model tends to perform worse with higher losses”.

Work [5] became a logical continuation of research on ANODEs. The authors examined the behavior of Second Order Neural ODEs (SONODEs). The adjoint sensitivity method shows that first-order optimization is more computationally efficient. The extended theoretical methodology of the ANODEs class allowed us to consider higher-order dynamics with a minimal number of augmented dimensions. This opportunity allows us to conclude that the use of ANODEs is possible in larger perspectives than previously thought by scientists. SONODEs are especially useful when working with synthetic and real second-order dynamic systems.

The work [6] is also intended to expand the scope of Neural ODE with the help of stochastic processes that simulate discrete events. The authors introduce Neural Jump Stochastic Differential Equations (Neural JSDEs) that “provide a data-driven approach to learn continuous and discrete dynamic behavior”. The authors believe that the downside of continuous Neural ODE models is that they cannot include discrete events (or inputs). Such discrete states or inputs can dramatically change the latent state vector. A major advantage of Neural JSDEs is that they can be used to model a variety of marked point processes. During the experiment, the authors demonstrated the predictive capabilities of their approach on a number of synthetic and real data sets when solving various kinds of problems.

The authors of the article [7] proposed Dynamics of Attention for Focus Transition (DAFT) as a human prior for machine reasoning. DAFT streamlines reasoning by modeling it as a continuous dynamic system using ordinary neural differential equations. Experiments have shown that DAFT provides performance comparable to the original model with fewer steps. The step size is determined using a new metric, Total Length of Transition (TLT). TLT enables a direct quantitative comparison between the quality of reasoning of different models. The proposed approach is one of the forms of Neural ODE extension and allows to solve incomparably more complex problems.

The authors of the article [8] draw attention to the fact that the work [1] lacks some frequently used regulation mechanisms in discrete neural networks (for example, dropout, Gaussian noise). At the same time, these regulatory mechanisms are critical in reducing generalization errors as well as in improving the robustness of neural networks to adversarial attacks. They proposed a new continuous neural network framework called Neural Stochastic Differential Equation (Neural SDE) network, which “naturally incorporates various commonly used regularization mechanisms based on random noise injection”. According to the authors, “the Neural SDE network can achieve better generalization than the Neural ODE and is more resistant to adversarial and non-adversarial input perturbations”. The authors have developed a new efficient

backpropagation method for calculating the gradient and training the SDE neural network in a scalable way.

In [9], it is proposed to use spectral element methods for fast and accurate learning of the Neural ODE. Several unlimited subproblems have been proposed to be solved using coordinate descents. They alternately minimize coefficients and weights using standard backpropagation and gradient techniques. The result of the work done by the authors is an optimization scheme that is parallel in time and has low memory costs. The experiment showed that "on training surrogate models of small and medium-scale dynamical systems shows that it is at least one order of magnitude faster at reaching a comparable value of the loss function".

Neural ODEs are the latest example of continuous deep learning models that are primarily developed in the context of continuous recurrent networks. The topic of adapting Neural ODE for use in other tasks is raised in [10]. The work itself is dedicated to the study and improvement of learning standard recurrent neural networks (RNNs). The authors made a successful attempt to generalize RNNs to have continuous-time hidden dynamics defined by ordinary differential equations (ODE-RNNs). The proposed ODE-RNN model has more efficient parameterization with irregular sampling of data points. None of the models presented by the authors require importing data at the preprocessing stage. This characteristic makes them suitable for interfacing with irregularly sampled time series that are often used in many applications.

The authors of [11] set out to develop a Universal Differential Equations (UDE) methodology that complements scientific models using machine learning frameworks. Authors present a "software that allows for combining existing scientific simulation libraries with neural networks to train and augment known models with data-driven components". Hybrid machine learning models combine the best of different methodologies. They can lead to significant improvements in problem solving efficiency. The materials presented in this article cover many scientific disciplines and combine many different approaches to modeling. This really meets the universality requirement put forward by the authors.

Neural ODEs contribute to the theoretical bridging of the gap between dynamic systems and deep learning. This is accomplished by using a black box approach. However, we may face a situation, which the decoding of the internal state of the black box is required. The work [12] is devoted to handling this situation. The authors propose a system – "theoretic perspective with the aim of clarifying the influence of several design choices on the underlying dynamics". It allows processing in an infinite-dimensional space and led the authors to create another variant of the Neural ODE extension - Neural ODE variant based on a spectral discretization (GalNODE).

Another variant of the Neural ODE is proposed by the authors of the article [13], whose trajectories evolve on an energy functional parametrised by a neural network. This method provides robustness to input disturbances and low computational load. The proposed adaptation law is based on a gradient descent procedure covering both terminal and backpropagated settings. The authors also proposed the introduction of certain regularizers to facilitate the optimization process.

The authors of the article [14] find an explanation for the rather low performance of Neural ODE in comparison with discrete-layered models. In their opinion, the reason for this situation lies in the inaccuracy of the existing methods for assessing the gradient. The authors suggest using the Adaptive Checkpoint Adjoint (ACA) method for the solution. According to their idea, ACA removes redundant components for shallow computation graphs during the backtrack and supports an adaptive solution. Experiments have shown that ACA guarantees a two-fold reduction in the number of errors in half the training time. Thus, the ACA-trained Neural ODE outperforms traditional ResNet in both accuracy and reliability of repeated iterations.

The paper [15] is devoted to the application of Neural ODE in problems of modeling the dynamics of continuous time. According to the authors, their proposed new learning models "characterize continuous-time dynamics and enable us to develop high-performing policies using a small amount of data". In this case, the adaptation of Neural ODE to the solution of this applied problem led to the fact that it was possible to optimize the measurement schedules. And it also succeeded by minimizing interactions with the environment while maintaining optimal performance. According to the authors, they were the first to extend the applicability of neural ODEs to reinforcement learning (RL).

Finally, looking for an answer to the question "When are Neural ODE Solutions Proper ODEs?" the work of the authors is devoted [16]. The behavior of a trained neural network directly depends on the numerical method embedded in it. If the network is trained on a solver with too coarse sampling, then when the solver is replaced, the error rate can increase dramatically. Thus, it is necessary to carefully monitor the size of the adaptive stride, otherwise the stride indicator may take on some critical value. It can violate the whole concept of Neural ODE, as it can violate the interpretation of the flow - a combination of a vector field and a numerical method. The authors proposed a step-by-step adaptation algorithm that checks that the continuity property is preserved and adapts the step size if necessary. Moreover, the algorithm ensures that the proposed initial step is not too small. Failure to comply with this condition would make such an algorithm not very applicable in practice. To correctly determine the step size, the algorithm requires taking into account a certain multiplicative coefficient. Taking it into account, according to the authors, has successfully proven itself in practice.

When presenting their concept, the authors of the original article [1] noted the similarity of updating the hidden state of a neural network with solving a differential equation by Euler's method. However, today, it is generally known that Euler's method has significant drawbacks. This method has a large error, is computationally unstable and accumulates error at each step, deviating from the integral curve. Therefore, a more perfect modification must be used as a numerical method. It must meet modern requirements for computing resources and time resources.

The family of Runge-Kutta methods and their modifications are widely used to solve nonlinear differential equations of polynomial structure. As a rule, the Runge-Kutta methods of the fourth and higher orders are used. To solve nonlinear

differential equations, it is necessary to use methods that provide the required performance and accuracy of calculations [17]. Therefore, the new scheme of the numerical method proposed by the authors of this article was based on the Runge-Kutta family of methods, which includes more efficient and accurate numerical methods for solving differential equations. The scheme should allow you to successfully apply all the declared advantages of Neural ODE.

Thus, in the course of the research carried out by the authors of this article, the authors have developed a new scheme of the numerical method for implementation in Neural ODE, based on the Runge-Kutta family of methods. The proposed numerical method should provide guaranteed third-order accuracy.

The analysis of previous experiments allowed us to conclude that Neural ODE is a relevant and practically applicable approach. This concept is relatively new, but is already being actively studied and deserves closer study. The first experiments confirm the applicability of Neural ODE in practice and demonstrate results comparable to traditional models. They have shown great promise in solving a number of problems, including modeling continuous-time data and building stream normalization with low computational costs. Neural ODE achieves great success in free-form reversible generative models, time series analysis and system identification.

Using Neural ODE allows you to reach a certain trade-off between the desired accuracy and training time. In turn, this feature allows you to adapt the computational complexity of the model for specific tasks and available computing resources. The information we have so far about the results of experiments in the field of Neural ODE suggests that Neural ODE has promising prospects, despite the longer training time for the model.

Section 2 of this article contains a description of the implemented numerical method with a detailed presentation of the mathematical apparatus. Section 3 presents the results of the experiment, and Section 4 formulates the general conclusions of the study.

## II. Implemented numerical method

In this section, we present a description of the numerical method and an estimation of the numerical model error based on Runge-Kutta method.

The Runge-Kutta family of methods is a large class of numerical methods for solving the Cauchy problem for ordinary differential equations and their systems.

For Runge-Kutta methods, the calculation formulas are uniquely determined by the Butcher table [18]. For the Runge-Kutta method, the Butcher table has the form in Eq. (1):

$$\begin{array}{cccccc} 0 & & & & c_1 & \\ 1/2 & 1/2 & & & c_2 & a_{21} \\ 1 & -1 & 2 & & c_3 & a_{31} \quad a_{32} \\ 1 & 1/6 & 2/3 & 1/6 & c_4 & a_{41} \quad a_{42} \quad a_{43} \\ & 1/6 & 2/3 & 1/6 \quad 0 & & b_1 \quad b_2 \quad b_3 \quad b_4 \end{array} \tag{1}$$

The calculation formulas of the third order Runge-Kutta method in accordance with the table are presented in Eq. (2):

$$\begin{gathered} y'(x) = f(x,y), \\ y_{n+1} = y_n + k_1/6 + 2k_2/3 + k_3/6, x_{n+1} = x_n + h \\ k_1 = hf(x_n, y_n), k_2 = hf\left(x_n + h/2, y_n + k_1/2\right) \\ k_3 = hf\left(x_n + h, y_n + 2k_2 - k_1\right) \end{gathered} \tag{2}$$

Let us present the Butcher table of the proposed method based on the Runge-Kutta method. The Butcher table for the proposed numerical method is presented in Eq. (3):

$$\begin{array}{cccc} 0 & & & \\ 1/2 & 1/2 & & \\ 3/4 & 0 & 3/4 & \\ 1 & 1/6 & 1/3 & 1/2 \\ & 1/6 & 1/3 & 1/2 \quad 0 \end{array} \tag{3}$$

The calculation formulas of the proposed method in accordance with the Butcher table are presented in Eq. (4):

$$\begin{gathered} y'(x) = f(x,y), x_{n+1} = x_n + h_n, \\ y_{n+1} = y_n + k_1/6 + k_2/3 + k_3/2, \\ k_1 = h_n f(x_n, y_n), \\ k_2 = h_n f\left(x_n + h_n/2, y_n + k_1/2\right) \\ k_3 = h_n f\left(x_n + 3h_n/4, y_n + 3k_2/4\right), \\ k_4 = h_n f\left(x_n + h_n, y_n + k_1/6 + k_2/3 + k_3/2\right) \end{gathered} \tag{4}$$

To select an adaptive step, the author proposed calculation formulas presented in Eq. (5):

$$h_{n+1} = h_n 0.08\left(max\left|k_1/18 - k_3/18\right|\right)^{-0.2} \tag{5}$$

The error was determined by the method of Dormand and Prens [19].

For the formulas of the Runge-Kutta methods, the equality presented in Eq. (6):

$$y_{n+1} = y_n + h_n F\left(y_n, h_n\right) = y_n + \sum_{i=1}^{s} b_i k_i \tag{6}$$

where $k_i = h_n f(y_n + \sum_{j=1}^{i-1} a_{ij} k_j)$, $k_1 = h_n f\left(y_n\right)$, $a_{ij}, b_i$ - Butcher table coefficients.

Method error at point $x_{n+1}$ presented in Eq. (7):

$$\varepsilon = y_n + h_n F\left(y_n, h_n\right) - y_{n+1} \tag{7}$$

Let us expand the error in the vicinity of the point $x_n$ into a Taylor series (see Eq. (8)):

$$\varepsilon = h_n \left( F\left(y_n, h_n\right) - \sum_{i=1}^{\infty} \frac{h^{i-1}}{i!} y^{(i)} \right) \tag{8}$$

If $\varepsilon = O\left(h^p\right)$, then the error for the Runge-Kutta formulas of order *p* and presented in Eq. (9) [18]:

$$\varepsilon = \sum_{j=1}^{\infty} h_n^{p+j} \delta_{p+j-1} \tag{9}$$

where error function $\delta_r = \sum_{i=1}^{n_{r+1}} \alpha_i^{(j)} D_i^{(r+1)}$, $D_i^{(r+1)}$ - order differential (r+1) from $f\left(y\right)$, $i = 1, \ldots, n_{r+1}$.

Substituting the error function, we get the formula for the error (see Eq. (10)):

$$\varepsilon = \sum_{j=1}^{\infty} h_n^{p+j} \sum_{i=1}^{n_{p+j}} \alpha_i^{(j)} D_i^{(p+j)} \tag{10}$$

For the Runge-Kutta formulas of order *p* error function $\delta_r = 0,\ r = 1, \ldots, p-1$, which means (see Eq. (11)):

$$\alpha_i^{(r+1)} \approx 0,\ i = 1, \ldots, n_{r+1}, r = 1, \ldots, p-1 \tag{11}$$

D. Butcher in his work [18] gave formulas for calculating the coefficients $\alpha_i^{(r+1)}$ depending on the coefficient of the Butcher table.

The author [18] proposed the choice of the adaptive step in the form presented in Eq. (12):

$$\begin{aligned} h_{n+1} &= h_n 0.08 \left( max \left| \varphi_{n+1} \right| \right)^{-0.2} = \\ &= h_n 0.08 \left( max \left| \tilde{y}_{n+1} - y_{n+1} \right| \right)^{-0.2} = \\ &= h_n 0.08 \left( max \left| k_1 / 18 - k_3 / 18 \right| \right)^{-0.2} \end{aligned} \tag{12}$$

where $\varphi_{n+1} = \tilde{y}_{n+1} - y_{n+1} = (k_1 - k_3)/18$.

The coefficients of the Butcher table, taking into account the choice of the step, have the values presented in Eq. (13):

$$\begin{aligned} &c_1 = 0, c_2 = \frac{1}{2}, c_3 = \frac{3}{4}, c_4 = 1, \\ &\tilde{b}_1 = \frac{4}{18}, \tilde{b}_2 = \frac{1}{3}, \tilde{b}_3 = \frac{8}{18}, \tilde{b}_4 = 0 \\ &a_{21} = \frac{1}{2}, a_{31} = 0, a_{32} = \frac{3}{4}, a_{41} = \frac{1}{6}, a_{42} = \frac{1}{3}, a_{43} = \frac{1}{2}, \end{aligned} \tag{13}$$

Substituting into the formulas for $\alpha_i^{(r+1)}$ the Butcher coefficients we obtain zero values for the coefficients (see Eq. (14)).

$$\begin{aligned} &\alpha_1^{(1)} = \sum_i b_i - 1 = \mathrm{b}_1 + \mathrm{b}_2 + \mathrm{b}_3 + \mathrm{b}_4 - 1 = 0 \\ &\alpha_1^{(2)} = \sum_i b_i c_i - \frac{1}{2} = \mathrm{c}_1\mathrm{b}_1 + \mathrm{c}_2\mathrm{b}_2 + \mathrm{c}_3\mathrm{b}_3 + \mathrm{c}_4\mathrm{b}_4 - \frac{1}{2} = 0 \\ &\alpha_1^{(3)} = \frac{1}{2} \sum_i b_i c_i^2 - \frac{1}{6} = \\ &= \frac{1}{2}\left( \mathrm{c}_1^{\ 2}\ \mathrm{b}_1 + \mathrm{c}_2^{\ 2}\ \mathrm{b}_2 + \mathrm{c}_3^{\ 2}\ \mathrm{b}_3 + \mathrm{c}_4^{\ 2}\ \mathrm{b}_4 \right) - \frac{1}{6} = 0 \\ &\alpha_2^{(3)} = \sum_{ij} b_i a_{ij} c_j - \frac{1}{6} = \mathrm{b}_2\mathrm{a}_{21}\mathrm{c}_1 + \mathrm{b}_3\mathrm{a}_{31}\mathrm{c}_1 + \\ &+\mathrm{b}_3\mathrm{a}_{32}\mathrm{c}_2 + \mathrm{b}_4\mathrm{a}_{41}\mathrm{c}_1 + \mathrm{b}_4\mathrm{a}_{42}\mathrm{c}_2 + \mathrm{b}_4\mathrm{a}_{43}\mathrm{c}_3 - \frac{1}{6} = 0 \\ &\alpha_2^{(4)} = \sum_{ij} b_i c_i a_{ij} c_j - \frac{1}{8} = \mathrm{b}_2\mathrm{c}_2\mathrm{a}_{21}\mathrm{c}_1 + \mathrm{b}_3\mathrm{c}_3\mathrm{a}_{31}\mathrm{c}_1 + \\ &+\mathrm{b}_3\mathrm{c}_3\mathrm{a}_{32}\mathrm{c}_2 + \mathrm{b}_4\mathrm{c}_4\mathrm{a}_{41}\mathrm{c}_1 + \mathrm{b}_4\mathrm{c}_4\mathrm{a}_{42}\mathrm{c}_2 + \mathrm{b}_4\mathrm{c}_4\mathrm{a}_{43}\mathrm{c}_3 - \frac{1}{8} = 0 \\ &\alpha_3^{(4)} = \frac{1}{2} \sum_{ij} b_i a_{ij} c_j^2 - \frac{1}{24} = \\ &= \left( \begin{matrix} \mathrm{b}_2\mathrm{a}_{21}\mathrm{c}_1^{\ 2} + \mathrm{b}_3\mathrm{a}_{31}\mathrm{c}_1^{\ 2} + \mathrm{b}_3\mathrm{a}_{32}\mathrm{c}_2^{\ 2} + \\ +\mathrm{b}_4\mathrm{a}_{41}\mathrm{c}_1^{\ 2} + \mathrm{b}_4\mathrm{a}_{42}\mathrm{c}_2^{\ 2} + \mathrm{b}_4\mathrm{a}_{43}\mathrm{c}_3^{\ 2} \end{matrix} \right) / 2 - \frac{1}{24} = 0 \end{aligned} \tag{14}$$

For accuracy of the Runge-Kutta method, it is necessary and sufficient to satisfy the conditions for the smallness of the coefficients up to $\alpha_i^{(p)}$, since in the Taylor expansion of the error, only the terms of the order $h^{p+1}$.[18-21]

Thus, in the Taylor series expansion of the error function, the terms will be zero up to the fourth order. This confirms that our proposed numerical method based on a modification of the Runge-Kutta method has a guaranteed accuracy of the third order.

## III. Computational experiment

In this section, the experimental methods are presented, as well as the results of the calculations performed.

The purpose of the experiment is to compare our proposed Neural ODE with the integration of a numerical method based on a modification of the Runge-Kutta method with already existing standard machine learning methods.

The following methods were selected to participate in the experiment: Gaussian Progress, Linear Regression, Random Forest, Neural Network, Decision Tree, Neural ODE (with the proposed modification). These methods are standard and most commonly used in the field of machine learning.

The experiment was carried out on 300 examples of solving a standard forecasting problem. Figures 1 and 2 show the timing of training and calculations (units of time - seconds).

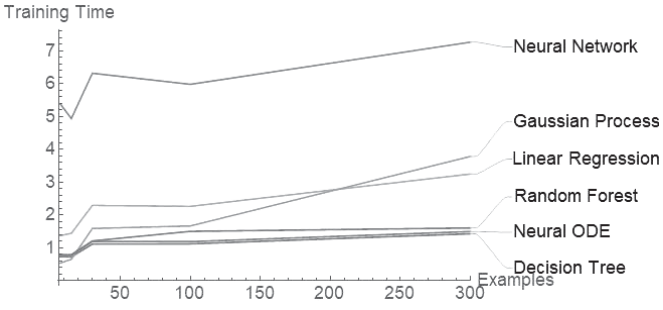


Fig. 1. Training time

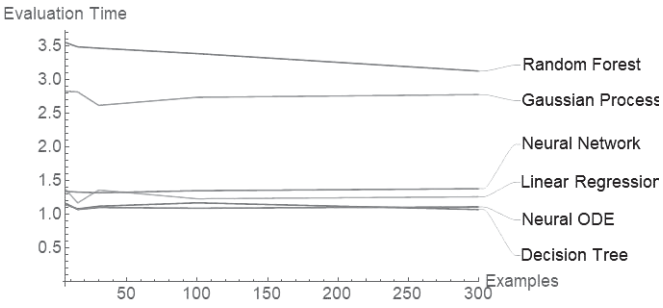


Fig. 2. Evaluation time

The results of the experiment allow us to conclude that the proposed modification of the Neural ODE (under the given experimental conditions) shows, relative to other methods, a shorter computation time, comparable only to the indicators of the Decision Tree method. This advantage of the method is easy to explain and is confirmed by the very concept of this approach: a decrease in the number of layers of the neural network in order to minimize the problem of fading gradients.

Standard Neural Networks showed the maximum training time, which is also explained by the large number of layers.

The Random Forest method shows one of the minimum training times. However, it takes the longest to calculate.

It should be noted that only the Decision Tree methods and the proposed modification of the Neural ODE showed comparable results of training and computation times.

Separately comparing specifically Neural Network and Neural ODE with an integrated numerical method, the following conclusion is obvious: Neural ODE is more preferable in terms of training time and computation time.

Thus, the experimental results have shown the operability and computational efficiency of Neural ODE with the proposed modification of the Runge-Kutta method.

## IV. Conclusion

Using Neural ODE allows you to reach a certain trade-off between the desired accuracy and training time. In turn, this feature allows you to adapt the computational complexity of the model for specific tasks and available computing resources.

Within the framework of this work, a numerical method based on a modification of the Runge-Kutta method was presented.

An experiment was carried out to compare the Neural ODE and the proposed numerical method integrated into it with other standard machine learning methods. The experimental results confirmed the practical applicability of the model proposed by the authors. Neural ODE with an integrated modification of the Runge-Kutta method with guaranteed third-order accuracy showed the minimum training time and computation time relative to other presented machine learning methods.

Neural ODE is a relevant and practically applicable approach. This concept is relatively new, but is already being actively studied and deserves a closer study. We hope that applying Neural ODE with our proposed numerical method will improve the robustness and interpretability of the Neural ODE concept. The results of our work are ways to make a direct practical contribution to the development of the Neural ODE methodology.

## References

[1] T.Q. Chen, Y. Rubanova, J. Bettencourt and D.K. Duvenaud, “Neural ordinary differential equations”, *in Advances in neural information processing systems*, 2018, pp. 6571-6583.

[2] Q. Li and S. Hao, “An optimal control approach to deep learning and applications to discrete-weight neural networks”, Web: https://arxiv.org/abs/1803.01299.

[3] C. Beer and O. Barak, “One Step Back, Two Steps Forward: Interference and Learning in Recurrent Neural Networks”, *in Neural Computation*, vol. 31, Aug. 2019, pp. 1985-2003.

[4] E. Dupont, A. Doucet and Y.W. Teh, “Augmented Neural ODEs”, Web: https://arxiv.org/abs/1904.01681.

[5] A. Norcliffe, C. Bodnar, B. Day, N. Simidjievski and P. Liò, “On Second Order Behaviour in Augmented Neural ODEs”, Web: https://arxiv.org/abs/2006.07220.

[6] J. Jia and A. Benson, “Neural Jump Stochastic Differential Equations”, Web: https://arxiv.org/abs/1905.10403.

[7] W. Kim and Y. Lee, “Learning Dynamics of Attention: Human Prior for Interpretable Machine Reasoning”, *in Advances in Neural Information Processing Systems*, 2019, pp. 6019-6030.

[8] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar and C.-J. Hsieh, “Neural SDE: Stabilizing Neural ODE Networks with Stochastic Noise”, Web: https://arxiv.org/abs/1906.02355.

[9] A. Quaglino, M. Gallieri, J. Masci and J. Koutn´ık, “SNODE: Spectral Discretization of Neural ODEs for System Identification”, Web: https://arxiv.org/abs/1906.07038.

[10] Y. Rubanova, T.Q. Chen and D. Duvenaud, “Latent odes for irregularly-sampled time series”, Web: https://arxiv.org/abs/1907.03907.

[11] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner and A. Ramadhan, "Universal Differential Equations for Scientific Machine Learning", Web: https://arxiv.org/abs/2001.04385.
[12] S. Massaroli, M. Poli, J. Park, A. Yamashita and H. Asama, "Dissecting Neural ODEs", Web: https://arxiv.org/abs/2002.08071.
[13] S. Massaroli, M. Poli, M. Bin, J. Park, A. Yamashita and H. Asama, "Stable Neural Flows", Web: https://arxiv.org/abs/2003.08063.
[14] J. Zhuang, N. Dvornek, X. Li, S. Tatikonda, X. Papademetris and J. Duncan, "Adaptive Checkpoint Adjoint Method for Gradient Estimation in Neural ODE", Web: https://arxiv.org/abs/2006.02493.
[15] J. Du, J. Futoma and F. Doshi-Velez, "Model-based Reinforcement Learning for Semi-Markov Decision Processes with Neural ODEs", Web: https://arxiv.org/abs/2006.16210.
[16] K. Ott, P. Katiyar, P. Hennig and M. Tiemann, "When are Neural ODE Solutions Proper ODEs?", Web: https://arxiv.org/abs/2007.15386.
[17] S.E. Ivanov, "A numerical method for solving differential equations of a polynomial structure based on modification of the Runghe-Kutta method", *in J. Scientific Horizons*, vol. 11, 2018, pp. 16-22.
[18] J.C. Butcher, "Coefficients for the study of Runge-Kutta integration processes", *in J. of Comp. and Appl. Math.*, 1963, pp. 185-201.
[19] J.R. Dormand and P.J. Prince, "A Family of Embedded Runge-Kutia Formulae", *in J. Australian Math. Soc.*, vol. 6, 1980, pp. 19-26.
[20] J.C. Butcher, "On Runge-Kutta processes of high order", *in J. Australian Math. Soc.*, vol. 4, 1964, pp. 179-194.
[21] J.R. Dormand and P.J. Prince, "New Runge-Kutta algorithms for numerical simulation in dynamical astronomy", *in Celestial Mechanics*, vol. 18, 1978, pp. 223-232.