УДК 004.852:519.85

ФУНКЦИИ АКТИВАЦИИ НЕЙРОННЫХ СЕТЕЙ, ИХ ВИЗУАЛИЗАЦИЯ И ПРИМЕНЕНИЕ В ПРОГРАММИРОВАНИИ НА ПРИМЕРЕ БИБЛИОТЕКИ TENSORFLOW

В. А. Максимук[1], Р. А. Максимук[2]

[1] Студент-магистрант факультета электронно-информационных систем, УО «Брестский государственный технический университет», Брест, Беларусь, e-mail: sashanik212@gmail.com

[2] Студент-магистрант факультета электронно-информационных систем, УО «Брестский государственный технический университет», Брест, Беларусь, e-mail: romamaksimuk3@gmail.com

Реферат

В представленной научной работе проводится комплексное исследование функций активации (AF) – фундаментального компонента искусственных нейронных сетей, определяющего выходной сигнал узла на основе входных данных. Работа последовательно раскрывает роль AF как критического элемента нелинейности, который позволяет глубоким архитектурам обучаться сложным паттернам и выступать в роли универсальных аппроксиматоров.

Статья показывает эволюцию математических методов преобразования сигнала: от простейших линейных функций (Identity), пригодных лишь для задач скалярной регрессии, до сложных адаптивных и немонотонных структур. В работе проанализированы S-образные кривые (Sigmoid, Tanh), семейство ReLU, экспоненциальные юниты, обучаемые функции и диверсифицированные экспериментальные образцы.

Произведен также анализ градиентных потоков. В работе рассмотрено, как математические свойства производных AF влияют на возникновение проблем затухающего и взрывного градиентов. Демонстрируется, как переход от насыщающихся функций (сигмоида) к ReLU-подобным позволил ускорить обучение глубоких сетей, а внедрение таких функций, как GELU, стало индустриальным стандартом для современных трансформерных архитектур (BERT, GPT).

Исследование раскрывает потенциал современных решений, таких как Swish и Mish. Эти функции, полученные путем автоматизированного поиска и систематического анализа, за счет своей гладкости и немонотонности обеспечивают более эффективную пропагацию информации в сверхглубоких сетях. В разделе специализированных функций рассматриваются инструменты вероятностной нормировки, такие как Softmax, и их разреженные альтернативы (Sparsemax, Entmax).

Практическая значимость работы подкреплена примерами реализации в среде TensorFlow. Статья показывает, как механизмы автоматического дифференцирования (GradientTape) и инструменты визуализации (TensorBoard) позволяют исследователю эффективно отслеживать динамику активаций и стабильность обучения.

Ключевые выводы работы подчеркивают, что выбор функции активации – это не просто технический параметр, а стратегическое архитектурное решение, напрямую влияющее на точность, скорость сходимости и обобщающую способность искусственного интеллекта.

Ключевые слова: нейронные сети, функция активации, глубокое обучение, нелинейность, затухающий градиент, адаптивность, ReLU, TensorFlow.

ACTIVATION FUNCTIONS OF NEURAL NETWORKS, THEIR VISUALIZATION AND APPLICATION IN PROGRAMMING USING THE TENSORFLOW LIBRARY

V. A. Maksimuk, R. A. Maksimuk

Abstract

This paper presents a comprehensive study of activation functions (AFs), a fundamental component of artificial neural networks that determines a node's output based on input data. The paper consistently reveals the role of AFs as a critical element of nonlinearity, enabling deep architectures to learn complex patterns and act as universal approximators.

The paper demonstrates the evolution of mathematical methods for signal transformation: from simple linear functions (Identity), suitable only for scalar regression problems, to complex adaptive and non-monotonic structures. The paper analyzes S-shaped curves (Sigmoid, Tanh), the ReLU family, exponential units, learnable functions, and diversified experimental samples.

Gradient flows are also analyzed. The paper examines how the mathematical properties of AF derivatives influence the emergence of vanishing and exploding gradient problems. It demonstrates how switching from saturating functions (sigmoid) to ReLU-like functions has accelerated the training of deep networks, and the implementation of functions such as GELU has become an industry standard for modern transformer architectures (BERT, GPT).

The study reveals the potential of modern solutions such as Swish and Mish. These functions, obtained through automated search and systematic analysis, provide more efficient information propagation in ultra-deep networks due to their smoothness and non-monotonicity. The section on specialized functions discusses probabilistic normalization tools such as Softmax and their sparse alternatives (Sparsemax, Entmax).

The practical significance of the work is supported by implementation examples in the TensorFlow environment. The article demonstrates how automatic differentiation mechanisms (GradientTape) and visualization tools (TensorBoard) allow the researcher to effectively monitor activation dynamics and training stability. The key findings of the paper highlight that the choice of activation function is not simply a technical parameter, but a strategic architectural decision that directly impacts the accuracy, convergence rate, and generalization ability of artificial intelligence.

Keywords: neural networks, activation function, deep learning, nonlinearity, decaying gradient, adaptivity, ReLU, TensorFlow.

Введение

Ключевым компонентом, обеспечивающим высокую эффективность работы искусственных нейронных сетей, являются функции активации (AF). Их основное назначение состоит в том, что они определяют выходной сигнал узла сети на основе набора входных данных [4].

В контексте математической модели нейронной сети, основная роль функции активации заключается во внедрении нелинейности [3]. Без использования нелинейных функций глубокая нейронная сеть, независимо от количества слоев, оставалась бы эквивалентной однослойному персептрону и могла бы выполнять лишь линейные преобразования, что ограничило бы её способность находить сложные нелинейные

https://doi.org/10.36773/1818-1112-2026-139-1-123-130

закономерности в данных. Математические свойства функций активации, такие как дифференцируемость, монотонность и диапазон значений, напрямую влияют на скорость обучения, стабильность градиентных потоков и общую обобщающую способность модели [4].

Эволюция функций активации прошла путь от классических S-образных кривых, таких как логистическая сигмоида и гиперболический тангенс, до семейства ReLU, которое стало стандартом де-факто благодаря своей вычислительной эффективности и способности бороться с проблемой затухающих градиентов. Современный этап развития глубокого обучения характеризуется появлением адаптивных и немонотонных функций, таких как Swish и Mish, а также разработкой обучаемых функций, параметры которых настраиваются одновременно с весами сети. Понимание характеристик различных категорий функций активации критически важно для эффективного проектирования и обучения нейросетевых архитектур.

Как и алгоритмы оптимизации, функции активации имеют строго определенную структуру и формульную имплементацию. Поэтому они широко представлены в различных инструментах, помогающих применять функции активации в программировании – например, в библиотеке TensorFlow. Использование таких модулей, как tf и tf.keras, сильно упрощает имплементацию функций активации, предоставляя, в том числе, экспериментальную свободу при поиске подходящего решения. Кроме вышеперечисленного, TensorFlow автоматизирует сложные вычисления градиентов, полностью контролируя даже такие сложные современные адаптивные функции, как Swish и Mish.

1 Общее описание функций активации

Как уже было описано ранее, задача функции активации заключается во внедрении нелинейности в вычисления нейронной сети. Без использования нелинейных функций любая многослойная сеть, независимо от её глубины, будет выполнять лишь последовательность линейных операций, а с математической точки зрения композиция линейных функций сама является линейной функцией, что делает глубокую сеть без функций активации эквивалентной однослойному персептрону. Аналогично, любой блок последовательных слоев без функций активации может быть сведен к одному линейному слою после перемножения соответствующих матриц весов. Таким образом, именно нелинейные функции активации позволяют сетям выступать в роли универсальных аппроксиматоров, способных изучать сложные иерархические представления и моделировать нелинейные зависимости в данных.

Различные характеристики функций активации существенно влияет на способность модели к обучению и её итоговую точность. К ключевым характеристикам функций активации можно отнести нелинейность, дифференцируемость, монотонность, ограниченность, центрирование относительно нуля, вычислительную сложность [4].

Функции активации (AF) на основе их характеристик и исторического предназначения могут быть разделены на несколько ключевых групп [3]. Линейные FA представляют собой прямую, делая градиент независимым от значений входа. Классические FA имеют S-образную форму, гладкие и ограниченные, исторически использовались первыми. ReLU-подобные FA характеризуются линейностью для положительных входов и занулением для отрицательных. Адаптивные FA содержат параметры, которые настраиваются вместе с весами сети. Вероятностные и специализированные FA используются преимущественно в выходных слоях для формирования распределения вероятностей классов.

Использование функций активации неразрывно связано с рядом фундаментальных проблем [4].

1. Проблема затухающего градиента (Vanishing Gradient) возникает, когда градиент становится крайне малым по мере распространения вглубь сети, что вредит обучению, так как веса сети не обновляются должным образом.

2. Проблема взрывного градиента (Exploding Gradient) описывает резкий рост значений градиента по мере распространения вглубь сети, ведущий к нестабильности.

3. Проблема умирающих нейронов (Dead Neuron Problem) – это ситуация, когда нейроны с ReLU-типом активации перестают обновляться из-за постоянного нулевого выхода при отрицательных входах.

4. Насыщение (Saturation) характеризуется предрасположенностью некоторых функций активации формировать плато значений, зачастую в отдалении от нуля, где производная близка к нулю.

Проблемы затухающего и взрывного градиентов связаны между собой и напрямую зависят от значения производной используемой функции активации при обратном распространении. Градиент вычисляется путем последовательного перемножения локальных производных согласно цепному правилу, соответственно, накапливая аномальные значения с процессом распространения по сети. Так, например, максимальное значение производной сигмоиды составляет всего 0,25. При перемножении таких малых величин в глубоких архитектурах (например, с 20 слоями) итоговый градиент в начальных слоях становится исчезающе малым, из-за чего веса практически не обновляются и обучение останавливается [14].

Различные функции активации позволяют частично и полностью решать отдельные проблемы, оставаясь подверженными остальным. Кроме того, помимо решения очевидных проблем, делающих обучение невозможным, стоит также принимать во внимание и общие характеристики FA, например, вычислительную сложность, ведь даже незначительное усложнение процесса расчета FA в конечном итоге может повлиять на общую скорость обучения из-за постоянного вычисления градиентов.

Стоит отметить, что в программировании функции активации зачастую представлены в виде отдельных функций или строковых параметров связанных функций. Так, воспользовавшись модулем tf.keras.activations библиотеки TensorFlow v2.16.1 можно из набора входных значений xt получить выходные значения yt, преобразованные в тензор. Преобразование можно осуществить функцией, имеющей схожее название с используемой функцией активации, например, GeLU:

```
import numpy as np
import tensorflow as tf

x = np.linspace(-2, 2, 5).astype(np.float32)
y = tf.keras.activations.gelu(x)
print("Input: ", x)
print("Output: ", y)
```

```
Input: [-2. -1.  0.  1.  2.]
Output: tf.Tensor([-0.04550026 -0.15865526 0.  0.8413447
1.9544997], shape=(5,), dtype=float32)
```

Полученные значения могут быть применены для визуализации функций активации.

При использовании модуля tf.keras.layers в процессе определения структуры нейронной сети присвоенная слою функция активации может быть передана в виде строкового параметра. Создание простого слоя на 16 нейронов с функцией активации GeLU выглядит следующим образом:

```
tf.keras.layers.Dense(16, activation='gelu')
```

Более подробная информация о полном наборе функций и возможных строковых значениях доступна в официальных документациях TensorFlow и Keras.

2 Линейные функции активации. Identity

Линейные функции активации – это простейший тип преобразования сигнала в нейронной сети. Они сохраняют пропорциональную зависимость между входным и выходным значениями [3].

Основным представителем этой группы является функция тождественности Linear (или Identity):

$$f(z) = z. \quad (1)$$

И ее масштабированная модификация Scaled Linear:

$$f(z) = a \cdot z, \quad (2)$$

где a – коэффициент наклона.

https://doi.org/10.36773/1818-1112-2026-139-1-123-130

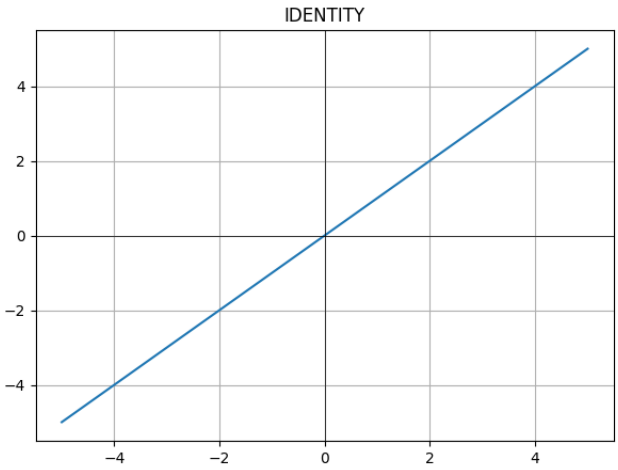

Рисунок 1 – Линейные функции активации: Identity

В TensorFlow Identity представлена как tensorflow.keras.activations.linear или 'linear'. Кроме того, Identity – значение по умолчанию для класса слоя Dense.

Из формул (1) и (2) видно, что функцию активации Linear можно представить как Scaled Linear с коэффициентом наклона a = 1. Соответственно, производная таких AF постоянна и равна a:

$$f'(z) = a. \tag{3}$$

Такая особенность производных этих функций активации делает их самыми простыми для вычислений при расчете градиентов среди всех доступных AF.

Использование линейных функций несет ряд серьезных ограничений, которые стоит учитывать при определении структур нейронных сетей. Как уже неоднократно упоминалось ранее, композиция нескольких линейных функций сама является линейной функцией, сводя несколько слоев с линейной функцией активации к одному эквивалентному, то есть без введения нелинейности увеличение глубины сети не расширяет её пространство гипотез. Поэтому линейные модели не могут аппроксимировать нелинейные зависимости, такие как классическая задача XOR [22].

Несмотря на множественные ограничения, линейные функции активации незаменимы в определенных частях нейронных сетей. Так, линейные AF могут быть применены для снижения размерности – линейные скрытые блоки могут использоваться для уменьшения количества параметров через факторизацию [24]. Кроме этого, линейные AF активно применяются для задач регрессии: если целью сети является предсказание непрерывного числового значения, которое не ограничено конкретным диапазоном (0, 1), то на выходном слое применяется линейная активация. Также, на практике, Identity-функция исторически была основой алгоритмов ADALINE [16].

В современных глубоких сетях линейная активация практически никогда не используется в скрытых слоях, уступая место семейству ReLU, которое сохраняет преимущества линейности для положительных значений, но вводит необходимую нелинейность за счет зануления отрицательных входов [3].

3 Классические функции активации

3.1 Логистическая сигмоида (Sigmoid)

Классические функции активации характеризуются S-образной формой. Они были доминирующими на ранних этапах развития глубокого обучения и до сих пор широко применяются в специфических архитектурах. Эти функции являются гладкими, монотонными и имеют фиксированный математический вид без обучаемых параметров [3].

Логистическая сигмоида Sigmoid – одна из самых известных функций активации, формула имеет вид

$$f(x) = \frac{1}{1+e^{-x}}. \tag{4}$$

И ее производная

$$f'(x) = \frac{e^{-x}}{(1+e^{-x})^2}. \tag{5}$$

На рисунке 2 приведены классические функции активации: Sigmoid.

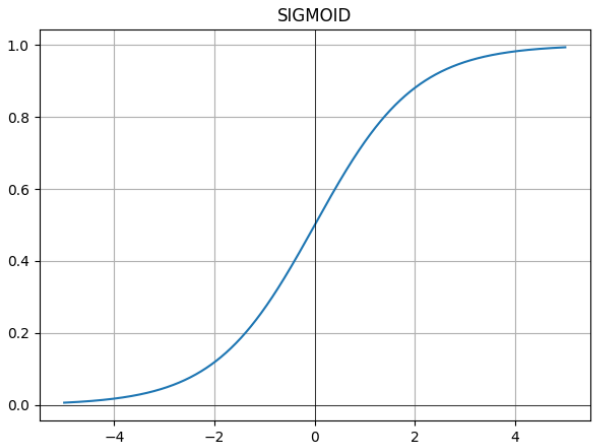

Рисунок 2 – Классические функции активации: Sigmoid

В TensorFlow Sigmoid представлена как tf.keras.activations.sigmoid или 'sigmoid'.

Как видно на рисунке 2, диапазон значений Sigmoid равен (0, 1), поэтому эта AF хорошо подходит для выходных слоев в задачах бинарной классификации. Однако, Sigmoid, как и все классические AF, сильно уязвима к проблеме насыщения и затухания градиента: при очень больших или малых значениях аргумента функция выходит на плато, где производная практически равна нулю. Кроме того, расчет экспоненты требует значительных ресурсов по сравнению с простыми операциями в ReLU [4].

3.2 Гиперболический тангенс (Tanh)

Гиперболический тангенс Tanh является альтернативой сигмоиде. Он определяется формулой

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{6}$$

Производная Tanh

$$f'(x) = \frac{4}{(e^{x} + e^{-x})^2}. \tag{7}$$

На рисунке 3 приведены классические функции активации: Tanh.

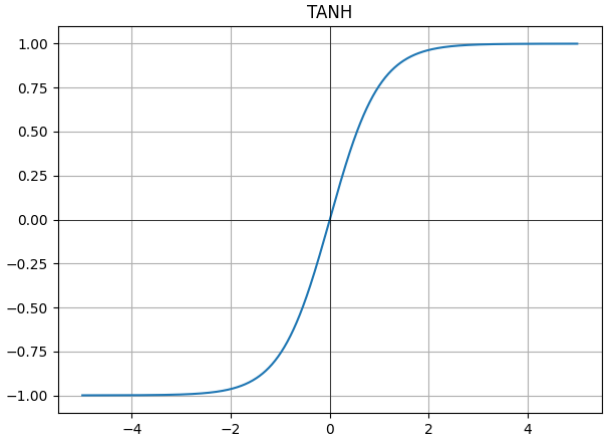

Рисунок 3 – Классические функции активации: Tanh

https://doi.org/10.36773/1818-1112-2026-139-1-123-130

В TensorFlow Tanh представлен как tf.keras.activations. tanh или 'tanh'.

Главная особенность, отличающая Tanh от Sigmoid, Tanh работает в интервале (–1, 1). Одним из недостатков сигмоиды было то, что, поскольку выходы Sigmoid всегда положительны, градиенты весов в процессе обновления будут иметь одинаковые знаки, что может вызывать нестабильность и замедлять сходимость. Tanh же помогает центрировать данные в скрытых слоях в начале обучения, что значительно ускоряет сходимость градиентного спуска [6]. Однако, как и сигмоида, Tanh подвержен насыщению градиента.

Tanh часто является предпочтительным выбором для рекуррентных нейронных сетей (RNN), так как помогает удерживать значения в стабильном диапазоне в течение многих временных шагов [5].

3.3 Кусочно-линейные варианты

Стоит упомянуть специфические вариации (hard-вариации) классических функций, такие как Hard-sigmoid и Hard-tanh. В основе своей изменения в их структуре по сравнению с оригинальными прототипами внесены для использования в системах с ограниченными вычислительными ресурсами или в аппаратных реализациях [5].

Hard-sigmoid представляет собой кусочно-линейную функцию, которая имитирует S-образную кривую, но состоит из простых арифметических операций без вычисления экспоненты. Она возвращает 0 при x≤−3, 1 при x≥3 и линейно изменяется между ними.

Hard-tanh является более простой версией Tanh, ограничивая линейный сигнал f(x) = x в диапазоне [–1, 1]. Она широко применяется в задачах обработки естественного языка (NLP) и робототехнике.

4 ReLU-подобные функции активации

4.1 Ректифицированный линейный юнит (ReLU)

Семейство функций активации на базе линейного ректификатора (ReLU), в отличие от классических S-образных функций, являются ненасыщающимися в положительной области, что критически важно для предотвращения затухания градиентов.

ReLU – простой стандарт для скрытых слоев современных нейронных сетей [5]. Формула

$$f(x) = \max(0, x). \tag{8}$$

Производная

$$f'(x) = \begin{cases} 0, & x < 0, \\ 1, & x > 0. \end{cases} \tag{9}$$

На рисунке 4 приведены ReLU-подобные функции активации: ReLU.

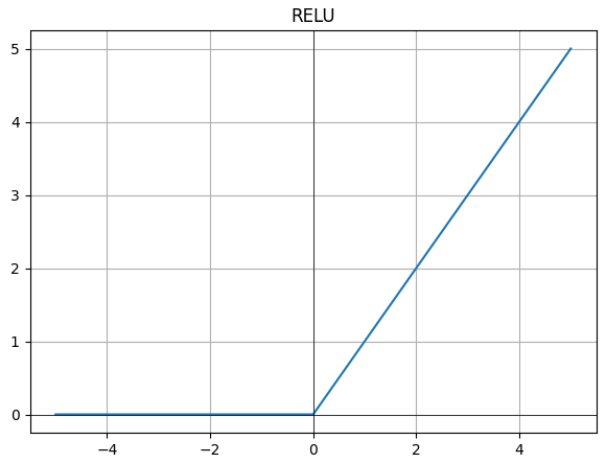

Рисунок 4 – ReLU-подобные функции активации: ReLU

В TensorFlow ReLU представлен как tf.keras.activations. relu или 'relu'.

К особенностям ReLU можно отнести ранее упомянутую ненасыщаемость: в положительной области производная всегда равна 1, что позволяет сигналу ошибки проходить через множество слоев без потерь. Кроме того, функция не требует расчета экспонент или деления, что значительно ускоряет обучение. Однако, если веса обновляются так, что вход всегда отрицателен, нейрон перестает активироваться и обновляться, так как его градиент становится нулевым.

4.2 Leaky ReLU

Leaky ReLU (LReLU) была предложена для решения проблемы смерти нейронов путем введения небольшого наклона в отрицательной области, заставляя градиенты никогда не затухать.

Математическая формула LReLU определяется как

$$f(x) = \max(\alpha x, x), \tag{10}$$

где α – коэффициент наклона (обычно малый, например, 0,01) [3].

Производная

$$f'(x) = \begin{cases} \alpha, & x < 0, \\ 1, & x > 0. \end{cases} \tag{11}$$

На рисунке 5 приведены ReLU-подобные функции активации: LReLU.

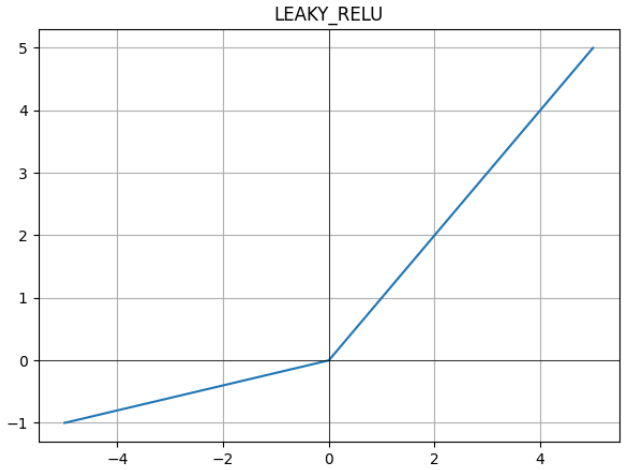

Рисунок 5 – ReLU-подобные функции активации: LReLU

В TensorFlow для использования LReLU можно применять tf.nn.leaky_relu.

Наличие коэффициента α гарантирует, что градиент никогда не станет нулевым, позволяя нейронам не зануляться в процессе обучения. Это особенно ценно в задачах, где важно сохранение информации об отрицательных входах, например, в генеративно-состязательных сетях (GAN) [22].

4.3 Гауссовский линейный юнит (GELU)

GELU – современная функция активации, которая, в отличие от ReLU, не просто зануляет отрицательные значения, а взвешивает входное значение его вероятностью в рамках стандартного нормального распределения. В формуле GELU зачастую используют аппроксимацию, поскольку точный расчет функции ошибок (erf) вычислительно сложен. Тогда аппроксимация формулы GELU:

$$f(x) = 0.5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right) \tag{12}$$

или выраженная через сигмоиду

$$f(x) = x \cdot \sigma(1.702x) \tag{13}$$

где σ – сигмоида.

Производная GELU:

$$f'(x) = \sigma(1.702x) + 1.702x\,\sigma(1.702x)(1 - \sigma(1.702x)) \tag{14}$$

На рисунке 6 приведены ReLU-подобные функции активации: GELU.

https://doi.org/10.36773/1818-1112-2026-139-1-123-130

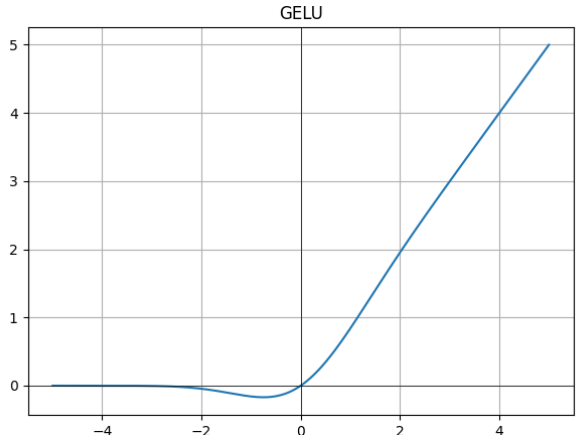

Рисунок 6 – ReLU-подобные функции активации: GELU

В TensorFlow GeLU представлена tf.keras.activations.gelu или 'gelu'.

В отличие от ReLU, имеющей резкий излом в нуле, GELU является полностью гладкой функцией. Также она немонотонна в небольшой области отрицательных значений (имеет локальный минимум), что может улучшать поток градиентов и обучение сложных структур [7]. GELU хорошо сочетается с пакетной нормализацией (Batch normalization), так как входы нейронов после неё часто имеют распределение, близкое к нормальному. GELU является основной функцией активации в большинстве современных языковых моделей, то есть в NLP и Трансформерах.

4.4 Прочие функции семейства ReLU

Среди прочих ReLU-подобных AF можно выделить Softplus, ELU и SELU.

Softplus определяется как f(x) = ln(1 + ex). Это полностью гладкая аппроксимация ReLU с диапазоном (0,∞). Несмотря на гладкость, на практике она часто уступает ReLU по качеству обучения [24].

ELU использует экспоненту для отрицательных значений: f(x) = α(ex − 1) при x ≤ 0. Это позволяет приблизить среднее значение активаций к нулю, что ускоряет сходимость и повышает устойчивость к шуму [3].

SELU является вариантом ELU f(x) = λ (ELU(x)) с фиксированными параметрами масштабирования (λ ≈ 1,0507, α ≈ 1,6733). Обладает уникальным свойством самонормализации: при соблюдении определенных условий выходы слоев автоматически сохраняют среднее 0 и дисперсию 1, решая проблему взрывных/затухающих градиентов без Batch Normalization [9].

5 Адаптивные функции активации

5.1 Swish

Адаптивные функции активации представляют собой класс функций, которые способны динамически настраивать свои параметры в процессе обучения в зависимости от характеристик входных данных или архитектуры нейронной сети [3].

Swish (или SILU) – это одна из гибридных функций активации, определяется формулой

$$f(x) = \frac{x}{1+e^{-\beta x}} = x \cdot \text{sigmoid}(\beta x)\,, \qquad (15)$$

где β – обучаемый параметр. Он может быть как фиксированной константой, так и обучаемым весом. При малых значениях β функция ведет себя как линейная, а при больших – приближается к ReLU. Это позволяет сети самостоятельно контролировать уровень вносимой нелинейности [6].

Производная Swish:

$$f'(x) = \frac{1}{1+e^{-\beta x}} + \frac{x}{1+e^{-\beta x}}\left(1 - \frac{1}{1+e^{-\beta x}}\right)\beta\,. \qquad (16)$$

На рисунке 7 приведены адаптивные функции активации: Swish.

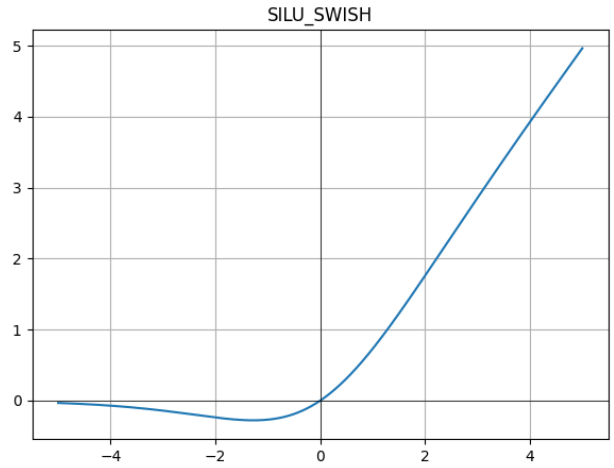

Рисунок 7 – Адаптивные функции активации: Swish

В TensorFlow для использования Swish можно применять tf.keras.activations.silu, либо строковые 'silu' и 'swish'.

Swish является гладкой, непрерывной и немонотонной функцией, что является её ключевым отличием от классических AF. Swish обладает плавной немонотонностью в области отрицательных значений, благодаря чему он не обнуляет информацию полностью, как это делает ReLU. Это позволяет сетям эффективнее передавать и градиенты, и сами активации через глубокие слои, что улучшает обучение и нередко приводит к более высокой точности на сложных задачах классификации [4].

5.2 Mish

Адаптивная AF Mish обладает более широким минимумом по сравнению со Swish и ReLU, что помогает избегать локальных минимумов и делает ландшафт функции потерь более гладким [4].

Формула Mish:

$$f(x) = x \cdot \tanh\left(\ln(1+e^{x})\right) = x \cdot \tanh(\text{softplus}(x))\,. \qquad (17)$$

Производная:

$$f'(x) = \tanh\left(\ln(1+e^{x})\right) + x \cdot \left(1 - \tanh^{2}\left(\ln(1+e^{x})\right)\right) \cdot \frac{1}{1+e^{-x}}\,. \qquad (18)$$

На рисунке 8 приведены адаптивные функции активации: Mish.

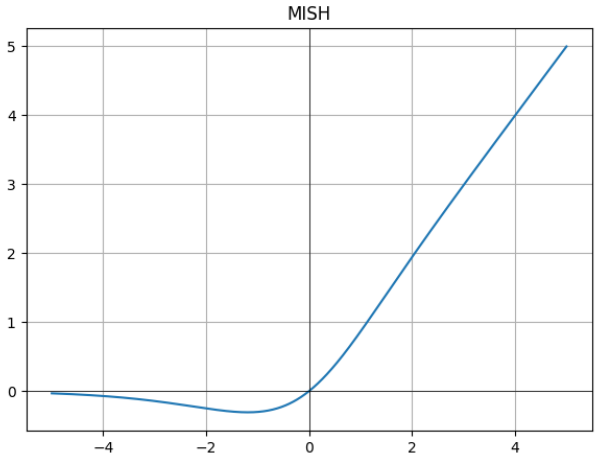

Рисунок 8 – Адаптивные функции активации: Mish

Для использования в TensorFlow Mish нужно определить самостоятельно, например, как return x * tf.math.tanh(tf.math.softplus(x)) либо использовать строковое представление 'mish'.

https://doi.org/10.36773/1818-1112-2026-139-1-123-130

За счет сохранения небольшого количества информации от отрицательных входов Mish значительно снижает риск появления проблемы Dying ReLU. Функция продемонстрировала высокую эффективность в задачах компьютерного зрения, в частности, она использовалась в архитектуре YOLOv4 для обнаружения объектов [6]. Тем не менее, её вычислительная сложность выше из-за использования нескольких функций (tanh, softplus).

5.3 Кусочно-линейные варианты

Как и в случае с классическими AF, адаптивные функции активации имеют свои альтернативные версии. Примером может послужить Hard-swish – линеаризованная версия swish для применения в мобильных и встроенных системах с ограниченными ресурсами [9]. В своей формуле Hard-swish заменяет вычислительно дорогую сигмоиду на её кусочно-линейный аналог ReLU6:

$$f(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6}. \quad (19)$$

Такая модификация значительно снижает затраты на расчет экспонент, сохраняя при этом основные преимущества Swish в плане точности.

6 Функции активации для вероятностных моделей и нормировки

6.1 Softmax

В отличие от функций активации для скрытых слоев, которые применяются к каждому нейрону независимо, функции для вероятностного моделирования и нормировки оперируют векторами значений, преобразуя их в распределения вероятностей. Проще говоря, функция активации Softmax, используемая при обучении вероятностных моделей и нормировке, применяется ко всем нейронам слоя одновременно, и значение каждого нейрона слоя зависит от всех остальных.

Для входного вектора x (называемого логитом) Softmax вычисляется как

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}. \quad (20)$$

Производная Softmax вычисляется по следующим формулам:

$$f_i'(x_j) = f_i(x_i)(\delta_{ij} - f_j(x_j)), \quad \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}. \quad (21)$$

Таким образом, Softmax преобразует набор входных значений в набор вероятностей, сумма которых равна 1. Тогда можно визуализировать работу этой функции активации, показав, как меняется итоговое распределение в зависимости от значения одного из входов x. Пусть входы AF Softmax представлены, как [x, 0,5, 0,1]. Тогда итоговые вероятности будут меняться следующим образом (рисунок 9).

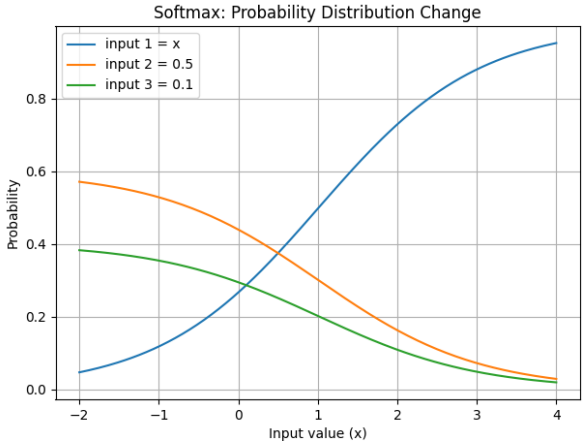

Рисунок 9 – Функции активации для вероятностных моделей и нормировки: пример Softmax

В TensorFlow Softmax доступен через tf.nn.softmax или 'softmax'.

Как уже упоминалось ранее, выход функции Softmax можно интерпретировать как распределение вероятностей по взаимоисключающим классам. Само собой, это делает Softmax идеальным кандидатом в задачах классификации.

Однако, при реализации Softmax часто возникает проблема переполнения из-за экспонент. Для её решения используется stable Softmax, где из каждого входа вычитается максимальное значение вектора: xi−max(x) [24].

6.2 Модификации Softmax

Gumbel-Softmax – это специализированная модификация, используемая для обучения моделей с дискретными переменными. Она позволяет проводить обратное распространение ошибки через процесс выборки из распределения, что невозможно с обычным Softmax. К логитам добавляется небольшой случайный шум, а затем применяется Softmax с настройкой "температуры".

Sparsemax превращает набор чисел в вероятности, оставляя часть элементов точно нулями, в отличие от softmax, где всё >0. Математически она определяется как: sparsemax(x)i = max(xi − т,0), где т – пороговое значение, вычисляемое адаптивно для каждого вектора. Это полезно в механизмах внимания, чтобы модель могла полностью игнорировать нерелевантные части входа [2].

Entmax является обобщением Softmax и Sparsemax, использующим энтропию Тсаллиса. Как softmax превращает числа в вероятности, а sparsemax иногда обнуляет часть значений, Entmax позволяет делать и то, и другое, но более гибко. С помощью параметра можно решать, сколько значений оставить нулями, а сколько – распределить как вероятности. Это делает эту AF эффективной в современных задачах обработки естественного языка (NLP) [2].

7 Экспериментальные и специализированные функции активации

Некоторые функции активации находятся на стадии активного исследования, либо были спроектированы для решения узкоспециализированных задач, таких как оптимизация вычислений на аппаратном уровне или улучшение градиентных потоков в сверхглубоких сетях [4].

Maxout – мощная обобщающая функция. В отличие от традиционных AF, она не просто применяет нелинейность к взвешенной сумме, а возвращает максимум из набора линейных функций

$$f(x) = \max\left(w_1^T x + b_1, w_2^T x + b_2, \ldots, w_k^T x + b_k\right). \quad (22)$$

Maxout является универсальным аппроксиматором любых выпуклых функций. Она наследует преимущества ReLU (отсутствие насыщения), но при этом избавляет сеть от проблемы умирающих нейронов. Главным минусом является значительное увеличение количества параметров (в k раз), что делает модель вычислительно дорогой и требующей усиленной регуляризации [5].

SIREN (Sinusoidal Representation Networks) использует функцию синуса

$$\mathrm{f}(x) = \sin(ax). \quad (23)$$

Было показано, что такие функции позволяют одиночным нейронам решать сложные задачи, такие как XOR, и эффективно моделируют сигналы с высокой частотой (звук, изображения) [4].

SQNL разработана для обеспечения максимальной вычислительной эффективности, особенно в аппаратных реализациях (FPGA, мобильные чипы). Она аппроксимирует форму сигмоиды или тангенса, используя только квадратичные операции, полностью исключая расчет экспонент:

$$\text{SQNL}(x) = \begin{cases} -1, & x \le -2 \\ [2mm]x + \frac{x^2}{4}, & -2 < x < 0 \\ [1mm]x - \frac{x^2}{4}, & 0 \le x < 2 \\ [1mm]1, & x \ge 2 \end{cases}. \quad (24)$$

https://doi.org/10.36773/1818-1112-2026-139-1-123-130

Производная SQNL является линейной, что на порядок ускоряет процесс обратного распространения ошибки по сравнению с традиционными AF [9].

LiSHT – это непараметрическая функция, предложенная для решения проблем затухающего градиента при сохранении преимуществ гиперболического тангенса:

$$f(x) = x \cdot \tanh(x). \quad (25)$$

Функция является симметричной и выдает только неотрицательные значения в диапазоне $[0,\infty)$. В отличие от ReLU, LiSHT использует информацию от отрицательных входов, масштабируя их, что может улучшить обучение в архитектурах вроде LSTM или глубоких остаточных сетях [9].

Bent Identity представляет собой изогнутую функцию тождественности, которая вводит нелинейность, сохраняя при этом общую линейную форму

$$f(x) = \frac{\sqrt{x^2+1}-1}{2} + x. \quad (26)$$

Эта AF действует как гладкая аппроксимация ReLU, обеспечивая наличие градиента даже при отрицательных значениях, что делает ландшафт функции потерь более стабильным для оптимизации [9].

Заключение

Таким образом, функций активации (AF) – критически важные компоненты нейронных сетей, определяющих их способность к обучению и обобщению. Основная роль функций активации заключается во внедрении нелинейности, что позволяет глубоким сетям аппроксимировать сложные зависимости и не превращаться в простые линейные модели.

AF прошли путь от простейших линейных функций, используемых преимущественно в задачах регрессии, до классических S-образных кривых (Sigmoid, Tanh). Переход к семейству ReLU стал революционным шагом, позволившим эффективно обучать сверхглубокие сети за счет устранения проблемы затухающих градиентов в положительной области.

Современный этап развития нейронных сетей характеризуется использованием адаптивных функций, таких как Swish и Mish, которые за счет своей немонотонности и гладкости обеспечивают лучшую сходимость и точность в сложных архитектурах.

Выбор конкретной AF должен быть строго продиктован спецификой задачи и типом слоя. Если функции семейства ReLU доминируют в скрытых слоях, то для выходных слоев незаменимыми остаются Softmax (в задачах многоклассовой классификации) и Sigmoid (для бинарного вывода), обеспечивающие вероятностную интерпретацию результатов.

На примере библиотеки TensorFlow можно увидеть, как современные фреймворки автоматизируют сложные вычисления производных, упрощая и визуализируя изменения данных. Применяя классы или строковые параметры, можно сокращать затраты времени на определение AF, фокусируясь на реализации и обучении нейронных сетей, позволяя TensorFlow определять и оптимизировать математические расчеты.

Список цитированных источников

1. Документация TesnorFlow. – URL: https://www.tensorflow.org/api_docs/python/tf (дата обращения: 13.02.2026).
2. Документация Keras. – URL: https://keras.io/api/layers/activations (дата обращения: 13.02.2026).
3. Bouraya, S. A comparative analysis of activation functions in neural networks: unveiling categories / S. Bouraya, A. Belangour // Bulletin of Electrical Engineering and Informatics. – 2024. – Vol. 13. – 8 c.
4. Gustineli, M. A survey on recently proposed activation functions for Deep Learning / M. Gustineli // Cornell University. – 2022. – 7 c.
5. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning / C. E. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall // Cornell University. – 2018. – 20 c.
6. Dubey, S. R. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark / S. R. Dubey, S. K. Singh, B. B. Chaudhuri // Cornell University. – 2022. – 18 c.
7. Titiya, M. D. Analyzing the Effect of Different Activation Functions in Deep Learning on Accuracy and Execution time / M. D. Titiya, A. V. Bala, S. Degadwala // International Journal of intelligent systems and applications in engineering. – 2024. – 8 c.
8. Демещенко, М. В. Функции активации / М. В. Демещенко, Р. С. Марковец, Т. А. Сугако, В. Д. Владымцев // Компьютерные системы и сети : сборник статей 59-й науч. конф. аспирантов, магистрантов и студентов, Минск, 17–21 апр. 2023 г. / БГУИР. – Минск, 2023. – С. 333–340.
9. Kunc, V. Three decades of activations: a comprehensive survey of 400 activation functions for neural networks / V. Kunc, J. Klema // Cornell University. – 2024. – 107 c.
10. Хашин, С. И. Сравнение активаторных функций нейросети / С. И. Хашин // Вестник Ивановского государственного университета. Серия: Естественные, общественные науки. – 2020. – Вып. 2. – С. 106–111.
11. Kingma, D. P. Adam: a method for stochastic optimization / D. P. Kingma, J. L. Ba // 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings. – 2015. – 15 p.
12. Ruder, S. An overview of gradient descent optimization algorithms / S. Ruder // Cornell University. – Dublin, 2016. – 14 p.
13. Kochenderfer, M. J. Algorithms for Optimization / M. J. Kochenderfer, T. A. Wheeler // The MIT Press. – Cambridge, Massachusetts, 2019. – P. 79–82.
14. Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems / A. Géron. – 2nd ed. – O'Reilly Media, Inc., 2019. – 856 p.
15. Singh, P. Learn TensorFlow 2.0: Implement Machine Learning and Deep Learning Models with Python / P. Singh, A. Manure // Apress. – Berkeley, CA, 2020. – 177 p. – DOI: 10.1007/978-1-4842-5558-2.
16. Рашка, С. Python и машинное обучение / С. Рашка, А. В. Логунова. – М. : ДМК Пресс, 2017. – 418 с.
17. Рамсундар, Б. TensorFlow для глубокого обучения / Б. Рамсундар, Р. Б. Заде ; пер. с англ. А. В. Логунова. – СПб. : БХВ-Петербург, 2019. – 256 с.
18. Шукла, Н. Машинное обучение и TensorFlow / Н. Шукла, К. Фритлас ; пер. с англ. А. А. Слинкина. – СПб. : Питер, 2019. – 336 с.
19. Поляк, Б. Т. Введение в оптимизацию / Б. Т. Поляк. – М. : Наука. Главная редакция физико-математической литературы, 1983. – 384 с.
20. Траск, Э. Грокаем глубокое обучение / Э. Траск ; пер. с англ. А. А. Слинкина. – СПб. : Питер, 2019. – 352 с.
21. Потапов, А. С. Технологии искусственного интеллекта : учебное пособие / А. С. Потапов. – СПб. : СПбГУ ИТМО, 2010. – 218 с.
22. Шолле, Ф. Глубокое обучение на Python / Ф. Шолле ; пер. с англ. А. Н. Киселева. – 2-е изд. – СПб. : Питер, 2023. – 576 с.
23. Рашид, Т. Создай свою нейронную сеть / Т. Рашид ; [пер. с англ. А. В. Логунова]. – СПб. : Альфа-книга, 2017. – 272 с.
24. Гудфеллоу, Я. Глубокое обучение / Я. Гудфеллоу, И. Бенджио, А. Курвилль ; [пер. с англ. А. А. Слинкина]. – М. : ДМК Пресс, 2018. – 652 с.
25. Современные численные методы оптимизации / А. В. Гасников, А. А. Дородницына, Ю. Г. Евтушенко [и др.] // Современные численные методы оптимизации : в 2 ч. / под ред. А. В. Гасникова. – М. : МЦНМО, 2021. – Ч. 1. – 272 с.
26. Матренин, П. В. Методы стохастической оптимизации / П. В. Матренин, М. Г. Гриф, В. Г. Секаев. – Новосибирск : Изд-во НГТУ, 2016. – 67 с.
27. Гасников, А. В. Основные конструкции над алгоритмами выпуклой оптимизации и их приложения к получению новых оценок для сильно выпуклых задач / А. В. Гасников, Д. И. Камзолов, М. А. Мендель // Труды МФТИ. – 2016. – Т. 8, № 4 (32). – С. 36–52.
28. Воронцова, Е. А. Выпуклая оптимизация : учебное пособие / Е. А. Воронцова. – М. : МФТИ, 2021. – 364 с.

29. Nocedal, J. Numerical Optimization / J. Nocedal, S. J. Wright. – 2nd ed. – New York : Springer Science+Business Media, LLC, 2006. – 664 p.
30. Bishop, C. M. Pattern Recognition and Machine Learning / C. M. Bishop. – New York : Springer Science+Business Media, LLC, 2006. – 738 p.

References

1. Dokumentaciya TesnorFlow. – URL: https://www.tensorflow.org/api_docs/python/tf (data obrashcheniya: 13.02.2026).
2. Dokumentaciya Keras. – URL: https://keras.io/api/layers/activations (data obrashcheniya: 13.02.2026).
3. Bouraya, S. A comparative analysis of activation functions in neural networks: unveiling categories / S. Bouraya, A. Belangour // Bulletin of Electrical Engineering and Informatics. – 2024. – Vol. 13. – 8 c.
4. Gustineli, M. A survey on recently proposed activation functions for Deep Learning / M. Gustineli // Cornell University. – 2022. – 7 c.
5. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning / C. E. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall // Cornell University. – 2018. – 20 c.
6. Dubey, S. R. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark / S. R. Dubey, S. K. Singh, B. B. Chaudhuri // Cornell University. – 2022. – 18 c.
7. Titiya, M. D. Analyzing the Effect of Different Activation Functions in Deep Learning on Accuracy and Execution time / M. D. Titiya, A. V. Bala, S. Degadwala // International Journal of intelligent systems and applications in engineering. – 2024. – 8 c.
8. Demeshchenko, M. V. Funkcii aktivacii / M. V. Demeshchenko, R. S. Markovec, T. A. Sugako, V. D. Vladymcev // Komp'yuternye sistemy i seti : sbornik statej 59-j nauch. konf. aspirantov, magistrantov i studentov, Minsk, 17–21 apr. 2023 g. / BGUIR. – Minsk, 2023. – S. 333–340.
9. Kunc, V. Three decades of activations: a comprehensive survey of 400 activation functions for neural networks / V. Kunc, J. Klema // Cornell University. – 2024. – 107 c.
10. Hashin, S. I. Sravnenie aktivatornyh funkcij nejroseti / S. I. Hashin // Vestnik Ivanovskogo gosudarstvennogo universiteta. Seriya: Estestvennye, obshchestvennye nauki. – 2020. – Vyp. 2. – S. 106–111.
11. Kingma, D. P. Adam: a method for stochastic optimization / D. P. Kingma, J. L. Ba // 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings. – 2015. – 15 p.
12. Ruder, S. An overview of gradient descent optimization algorithms / S. Ruder // Cornell University. – Dublin, 2016. – 14 p.
13. Kochenderfer, M. J. Algorithms for Optimization / M. J. Kochenderfer, T. A. Wheeler // The MIT Press. – Cambridge, Massachusetts, 2019. – P. 79–82.
14. Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems / A. Géron. – 2nd ed. – O'Reilly Media, Inc., 2019. – 856 p.
15. Singh, P. Learn TensorFlow 2.0: Implement Machine Learning and Deep Learning Models with Python / P. Singh, A. Manure // Apress. – Berkeley, CA, 2020. – 177 p. – DOI: 10.1007/978-1-4842-5558-2.
16. Rashka, S. Python i mashinnoe obuchenie / S. Rashka, A. V. Logunova. – M. : DMK Press, 2017. – 418 s.
17. Ramsundar, B. TensorFlow dlya glubokogo obucheniya / B. Ramsundar, R. B. Zade ; per. s angl. A. V. Logunova. – SPb. : BHV-Peterburg, 2019. – 256 s.
18. SHukla, N. Mashinnoe obuchenie i TensorFlow / N. SHukla, K. Fritlas ; per. s angl. A. A. Slinkina. – SPb. : Piter, 2019. – 336 s.
19. Polyak, B. T. Vvedenie v optimizaciyu / B. T. Polyak. – M. : Nauka. Glavnaya redakciya fiziko-matematicheskoj literatury, 1983. – 384 s.
20. Trask, E. Grokaem glubokoe obuchenie / E. Trask ; per. s angl. A. A. Slinkina. – SPb. : Piter, 2019. – 352 s.
21. Potapov, A. S. Tekhnologii iskusstvennogo intellekta : uchebnoe posobie / A. S. Potapov. – SPb. : SPbGU ITMO, 2010. – 218 s.
22. SHolle, F. Glubokoe obuchenie na Python / F. SHolle ; per. s angl. A. N. Kiseleva. – 2-e izd. – SPb. : Piter, 2023. – 576 s.
23. Rashid, T. Sozdaj svoyu nejronnuyu set' / T. Rashid ; [per. s angl. A. V. Logunova]. – SPb. : Al'fa-kniga, 2017. – 272 s.
24. Gudfellou, YA. Glubokoe obuchenie / YA. Gudfellou, I. Bendzhio, A. Kurvill' ; [per. s angl. A. A. Slinkina]. – M. : DMK Press, 2018. – 652 s.
25. Sovremennye chislennye metody optimizacii / A. V. Gasnikov, A. A. Dorodnicyna, YU. G. Evtushenko [i dr.] // Sovremennye chislennye metody optimizacii : v 2 ch. / pod red. A. V. Gasnikova. – M. : MCNMO, 2021. – CH. 1. – 272 s.
26. Matrenin, P. V. Metody stohasticheskoj optimizacii / P. V. Matrenin, M. G. Grif, V. G. Sekaev. – Novosibirsk : Izd-vo NGTU, 2016. – 67 s.
27. Gasnikov, A. V. Osnovnye konstrukcii nad algoritmami vypukloj optimizacii i ih prilozheniya k polucheniyu novyh ocenok dlya sil'no vypuklyh zadach / A. V. Gasnikov, D. I. Kamzolov, M. A. Mendel' // Trudy MFTI. – 2016. – T. 8, № 4 (32). – S. 36–52.
28. Voroncova, E. A. Vypuklaya optimizaciya : uchebnoe posobie / E. A. Voroncova. – M. : MFTI, 2021. – 364 s.
29. Nocedal, J. Numerical Optimization / J. Nocedal, S. J. Wright. – 2nd ed. – New York : Springer Science+Business Media, LLC, 2006. – 664 p.
30. Bishop, C. M. Pattern Recognition and Machine Learning / C. M. Bishop. – New York : Springer Science+Business Media, LLC, 2006. – 738 p.

Материал поступил 03.03.2026, одобрен 17.03.2026, принят к публикации 30.03.2026