Параллельное и распределенное программирование, грид-технологии, программирование на графических процессорах

https://doi.org/10.25559/SITITO.019.202304.913-923
УДК 004.32.24

Производительность параллельных вычислений при обучении моделей глубокого обучения

Т. А. Самойлова

ФГБОУ ВО «Смоленский государственный университет», г. Смоленск, Российская Федерация

Адрес: 214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4

tatsamoilova24@gmail.com

Оригинальная статья

Аннотация

Потребность в вычислительных ресурсах, необходимых для обучения новых моделей глубокого обучения, в настоящее время постоянно растет. Этот рост привел к появлению специализированных процессоров, называемых ускорителями. Такие ускорители, как графические процессоры (GPU) и тензорные процессоры (TPU) достигают лучшей производительности, чем центральный процессор (CPU), благодаря своим ресурсам параллельной реализации операций и высокой пропускной способности памяти. *Google Colaboratory* (Colab) – облачный сервис, позволяющий писать код на Python в Jupyter Notebook, обеспечивает свободный доступ к аппаратным платформам TPU и GPU, позволяя разрабатывать в браузере современные модели глубокого обучения. Основная цель данной статьи – сравнение производительности обучения моделей, работающих на ускорителях облачной платформы Colaboratory, предназначенных для обработки текстовых и графических данных. Представлено сравнение CPU и ускорителей GPU, TPU при обучении распознаванию изображений средствами моделей сверточной нейронной сети (CNN) и глубокой сети доверия (DBN). Выполнено сравнение производительности этих платформ в ходе обучения двунаправленных рекуррентных моделей долговременной краткосрочной памяти (BILSTM) и управляемого рекуррентного блока (BIGRU) с текстовыми данными на входе. Получены зависимости временных характеристик и метрик качества обучения моделей от их параметров, позволяющие количественно оценить производительность платформ. Для реализации параллелизма всех отобранных моделей выбрана Google-библиотека Tensorflow, поддерживающая реализацию параллельных вычислений на ускорителях GPU и TPU. Обучение распознаванию изображений выполнялось с использованием набора данных MNIST (*Modified National Institute of Standards and Technology* – модифицированная база данных Национального института стандартов и технологий), обучение классификации текстов – с набором IMDB (*Internet Movie Database* – интернет-база фильмов). Показано, что при этом время обучения на графических и текстовых данных сокращается в разы. Однако в случае текстов степень сокращения оказалась меньшей.

Ключевые слова: параллельные вычисления, GPU, TPU, Google Colab, Tensorflow, CNN, DBN, BILSTM, BIGRU

Конфликт интересов: автор заявляет об отсутствии конфликта интересов.

Для цитирования: Самойлова Т. А. Производительность параллельных вычислений при обучении моделей глубокого обучения // Современные информационные технологии и ИТ-образование. 2023. Т. 19, № 4. С. 913-923. https://doi.org/10.25559/SITITO.019.202304.913-923

Parallel and Distributed Programming, Grid Technologies, GPU Programming

Performance of Parallel Computing in Training Deep Learning Models

T. A. Samoilova

Smolensk State University, Smolensk, Russian Federation

Address: 4 Przhevalsky St., Smolensk 214000, Russian Federation

tatsamoilova24@gmail.com

Original article

Abstract

The demand for computing resources required to train new deep learning models is currently constantly growing. This growth has led to the introduction of specialized processors called accelerators. Accelerators such as graphics processing units (GPUs) and tensor processing units (TPUs) achieve better performance than a central processing unit (CPU) due to their parallel processing resources and high memory bandwidth. *Google Colaboratory* (Colab) is a cloud service that allows you to write Python code in Jupyter Notebook, provides free access to TPU and GPU hardware platforms, allowing you to develop modern deep learning models in the browser. The main goal of this article is to compare the training performance of models running on Colaboratory cloud platform accelerators designed for processing text and graphic data. A comparison of CPUs and GPU and TPU accelerators is presented when training image recognition using convolutional neural network (CNN) and deep belief network (DBN) models. The performance of these platforms is compared when training bidirectional long short-term memory (BILSTM) and guided recurrent unit (BIGRU) models with text input. Dependences of time characteristics and quality metrics of model training on their parameters were obtained, which make it possible to quantitatively assess the performance of the platforms. To implement the parallelism of all selected models, the Tensorflow Google library was selected, which supports parallel computing on GPU and TPU accelerators. Image recognition training was performed using the MNIST *(Modified National Institute of Standards and Technology)* dataset; text classification training was performed using the IMDB *(Internet Movie Database)* dataset. It is shown that in this case, the training time on graphic and text data is reduced significantly. However, in the case of texts, the degree of reduction was less.

Keywords: parallel computing, GPU, TPU, Google Colab, Tensorflow, CNN, DBN, BILSTM, BIGRU

Conflict of interests: The author declares no conflict of interests.

For citation: Samoilova T.A. Performance of Parallel Computing in Training Deep Learning Models. *Modern Information Technologies and IT-Education.* 2023;19(4):913-923. https://doi.org/10.25559/SITITO.019.202304.913-923

1 Введение

Глубокое обучение *(Deep Learning, DL)* в настоящее время считается главной технологией машинного обучения [1]. Благодаря своим вычислительным возможностям, основанным на искусственных нейронных сетях, оно широко применяется в различных областях, таких как здравоохранение, визуальное распознавание, анализ текста, кибербезопасность[1] [2-5]. Растет спрос на новые, более совершенные аппаратные и программные платформы для поддержки обучения и развертывания сложных моделей. Поскольку исследователи из научных кругов и промышленности пытаются предложить и внедрить такие системы, удовлетворяющие спрос[2], существует острая необходимость одновременно развивать систематический подход, как к выбору моделей, так и сравнительному анализу производительности их обучения на новых платформах. Сложность задачи вызвана динамическим характером самих моделей и различиями в обучающих данных [6]. Модели имеют разные нейросетевые структуры с несколькими уровнями, такие как сверточная нейронная сеть [7], глубокая сеть доверия [8], рекуррентные нейронные сети [9]. Чтобы их обучить на большом и нередко сложном наборе данных, требуется значительная вычислительная мощность. Её недостаточно у традиционных компьютеров с CPU, работающих последовательным образом. Для обучения модели DL необходимо огромное количество операций с плавающей запятой (математических вычислений). Графический процессор *(graphics processing unit, GPU)* [10] – революция в реализации алгоритмов DL. Для них GPU используют параллельную обработку, разделяя задачи на более мелкие подзадачи, которые распределяются между огромным количеством процессорных ядер. Поскольку GPU имеют тысячи ядер и могут выполнять тысячи операций умножения и сложения параллельно, их применяют для обучения моделей глубокого обучения, что позволяет GPU работать при этом в разы быстрее, чем CPU. В настоящее время лучший производитель GPU для DL – компания Nvidia представила собственную платформу под названием CUDA, позволяющую программистам иметь прямой доступ к набору виртуальных команд графического процессора и параллельным вычислительным элементам. Становится возможным настроить вычислительные ядра процессоров для задач DL, не нагружая при этом остальные ресурсы. Выполняя параллельные вычисления на значительном числе ядер, GPU тратит больше энергии на доступ к памяти. Чтобы решить эту проблему, Google разработал тензорный процессор *(Google Tensor Processing Unit, Google TPU)*[3] [11]. Тензорный процессор – это специализированная интегральная схема с матричным процессором, специально разработанная для DL, эффективно выполняет матричное сложение и умножение в нейронных сетях на высокой скорости и с очень низким энергопотреблением. Он загружает параметры модели из памяти в матрицу множителей и сумматоров. После этого загружает обучающие данные из памяти. Результат каждого выполнения умножения передается следующим умножителям, одновременно выполняющим суммирование. Таким образом, выходные данные представляют собой сумму результата всех умножений между данными и параметрами. Во всем процессе сложных вычислений и передачи данных, доступ к памяти не требуется. Вот почему TPU достигает высокой вычислительной производительности на алгоритмах DL.

2 Цель исследования

В работе выполнена оценка производительности обучения нейронных сетей, которая позволит исследователям и практикам обучать модели DL в больших масштабах и более быстрыми темпами. Рассмотрены нейронные сети для решения двух категорий проблем, связанных с параллельной реализацией пространственных и временных данных: обучение классификации изображений и классификации текстов. Для обработки изображений отобраны хорошо зарекомендовавшие себя в этой сфере сверточная нейронная сеть (CNN) и глубокая сеть доверия (DBN) [12-14]. Обработка текстовой информации выполняется современными рекуррентными сетями (RNN) – BILSTM и BIGRU [15, 16]. Распараллеливание алгоритмов обработки пространственных и временных данных существенно различается. У пространственных данных в алгоритмах обучения доминирующим шаблоном вычислений слоев сети является умножение матрицы на матрицу, тогда как у временных – умножение матрицы на вектор. Это означает, что RNN требуют меньше повторного использования данных. К тому же, вычисления RNN включают как внутривременные, так и межвременные зависимости, приводящие к худшему использованию оборудования GPU и TPU и низкой производительности в устаревших ускорителях. Однако вычислительный механизм, используемый в современных ускорителях, реализует как матрично-матричное, так и матрично-векторное умножение, позволяющее повышать производительность и при обучении RNN [17, 18]. В данной работе для создания моделей DL была выбрана библиотеки TensorFlow, разработанной Google и изначально поддерживающая параллельные вычисления ускорителей GPU и TPU. Эта библиотека имеет репутацию фреймворка,

[1] Алхажа О. М. А. Решение задачи машинного перевода посредством глубокого обучения // Информационные системы и технологии в моделировании и управлении: Сб. тр. VII Межд. НПК. Симферополь: Изд-во Типография «Ариал», 2023. С. 91-95. EDN: HMCSLG

[2] Основные рыночные решения искусственного интеллекта для реализации процесса обучения нейронных сетей / Д. Ю. Сохинов [и др.] // Интеллектуальные автоматизированные управляющие системы в биотехнологических процессах: сб. докл. всерос. НПК. М.: РОСБИОТЕХ; Университетская книга, 2023. С. 300-306. EDN: AOTZON

[3] Князьков В. С. Тензорные процессоры семейства Google TPU: архитектура и технические особенности // Фундаментальная и прикладная наука: состояние и тенденции развития : монография. Петрозаводск: Новая Наука, 2021. С. 134-155. EDN: WAFBSV

ориентированного на практические разработки. Доступ к ресурсам ускорителей выполнен средствами облачной платформы Google под названием Colab – "Google Colaboratory" [19]. На момент написания статьи Colab предоставляла следующие ресурсные мощности:

- GPU: Tesla T4 (cuda ядер 2560, Memory – 16 GB).
- TPU: TPU v3 (Total Memory – 32 ТБ).

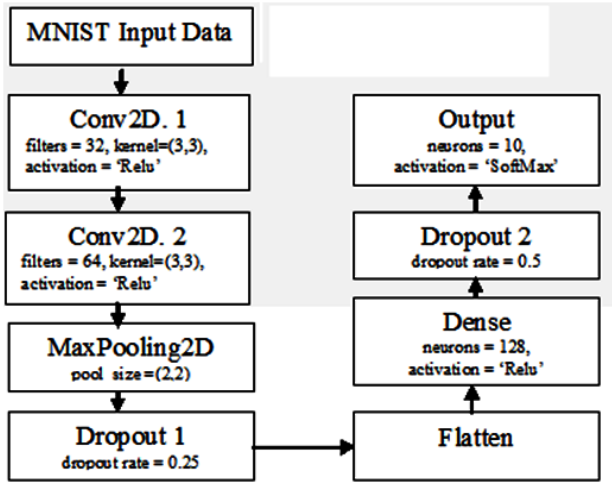
Целью статьи является сравнительный анализ производительности хорошо себя зарекомендовавших, современных моделей глубокого обучения на этих ускорителях.

3 Архитектуры нейронных сетей для экспериментов

Для экспериментов с глубоким обучением были отобраны четыре архитектуры нейронных сетей: CNN, DBN, BILSTM, BIGRU. Заданы параметры сетей, обеспечивающие максимальную точность и ускорение. Обучение сетей реализовано с помощью API-интерфейса TensorFlow[4].

3.1 Сверточная нейронная сеть

Сверточная нейронная сеть *(Convolutional Neural Network, CNN)* основана на широко используемом алгоритме классификации изображений [20, 21]. В основном она состоит из сверточных слоев, сокращающих размер обучающих входных изображений. Чтобы идентифицировать закономерности внутри изображения, CNN использует операции умножения матриц. После обучения сеть классифицирует изображения по определенным категориям. Обычно компьютер интерпретирует каждое входное изображение как массив пикселей, зависящий от разрешения изображения. Основным назначением CNN является выделение из исходного изображения малых частей, содержащих опорные (характерные) признаки *(features)*, такие как ребра, контуры, дуги или грани. Эксперимент проводился на данных MNIST, представляющей собой изображения рукописных цифр [22]. Они часто используются при оценке обучающих систем классификации. Набор содержит 60000 тренировочных и 10000 тестовых черно-белых изображений рукописных цифр от 0 до 9, размером 28×28 = 784 пикселей каждая. Каждый пиксель имеет значение от 0 до 255, описывающее интенсивность пикселя: 0 для белого и 255 – черного. Предлагаемая архитектура модели обучения представлена на рисунке 1. В ней два каскада сверточных слоев и один слой подвыборки. В процессе обучения каждое входное изображение (числовое значение пикселя) обрабатывается слоями свертки, сканирующими изображение, сокращая его размеры. При этом используем фильтры (32 и 64 соответственно) размера 3*3, применяя к ним поэлементно функцию активации. Каждый сверточный слой меняет размерность выходного изображения, что приводит к увеличению времени обучения такой сети. Чтобы его снизить, после сверточных слоев устанавливаем подвыборочный слой (*MaxPooling*, максимальной группировки), который уменьшает размерность выходного изображения. Он производит операцию подвыборки – процесс сжатия (уменьшения размеров) изображения путем сложения значений блоков его пикселей. В результате будет получено изображение меньшего размера по сравнению с оригинальным. Нельзя пропустить через нейронную сеть разом весь датасет, поэтому данные делим на партии или батчи *(batch)*. Размер батча, соответствующий подмножеству данных при выполнении градиентного спуска во время обучения, установлен как параметр batch_size=256. Для предотвращения переобучения используется слой регуляризации *Dropout*, выключающий нейроны с вероятностью 50%. Слой *Flatten* не имеет параметров обучения, но меняет форму тензора, выпрямляя двумерные матрицы в одномерные вектора. *Dense()* – полносвязный (плотный) слой, обрабатывает каждый элемент предыдущего слоя, реализуя матричное перемножение его выходных элементов со своими весами. Он позволяет выполнять обработку всех элементов предыдущего слоя и выдавать результат в форме одного вектора, который используется в дальнейшем процессе обучения. Выходной слой – Output подсчитывает оценки по классам, где каждое из 10 значений будет соответствовать оценке определенного класса среди 10 категорий изображений. Его функция активации *softmax* применяется для классификации объекта с вероятностными значениями от 0 до 1.



Р и с. 1. Архитектура CNN
F i g. 1. CNN architecture

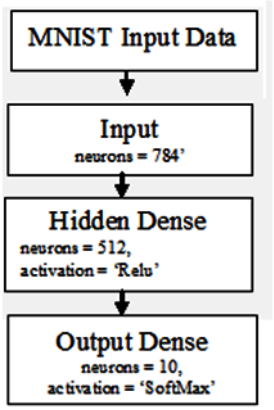
Источник: здесь и далее в статье все таблицы и рисунки составлены автором.
Source: Hereinafter in this article all tables and figures were made by the author.

3.2 Глубокая сеть доверия

Модель глубокой сети доверия *(deep belief network, DBN)* получила признание, как успешная реализация эффективной методики обучения, объединяющей более простые модели, известные как ограниченные машины Больцмана *(Restricted Boltzmann Machines, RBM)*. RBM

[4] Koul A., Ganju S., Kasam M. Practical deep learning for cloud, mobile, & edge: Real-world AI & computer-vision projects using Python, Keras, and TensorFlow. Sebastopol, CA: O'Reilly Media, 2019. 583 p.

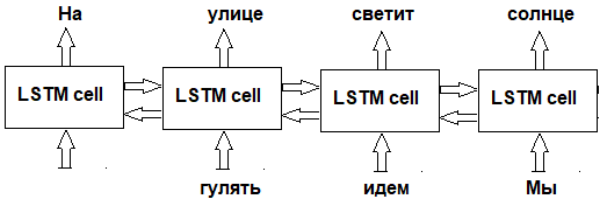
представляют собой мелкие нейронные сети, которые можно обучать с использованием вероятностного, неконтролируемого обучения [23]. Скрытый слой DBN включает в себя нескольких соединённых RBM. Последний уровень RBM обычно и представляет собой классификатор. Таким образом, DBN – многослойная стохастическая нейросеть, у которой веса нейронов скрытого слоя инициализируются случайным образом. В последние годы DBN широко распространены в сообществе глубокого обучения и нередко превосходят CNN [24], благодаря способности обрабатывать многомерные данные, хорошей масштабируемости и способности моделировать сложные нелинейные отношения в данных. В [25] предложили использовать модель DBN для классификации изображений. Эксперимент настоящей работы проведен на базе данных MNIST. Предлагаемая модель обучения представлена на рисунке 2. Ее архитектура состоит из одного видимого слоя с 784 нейронами, одного скрытого слоя с 500 нейронами и последнего выходного слоя с 10 нейронами, соответствующим категориям распознаваемых цифр. Настройка скрытого слоя реализуется с использованием функции активации ReLU, выходного слоя – функцией SoftMax. Веса соединений нейросети инициализируются случайными значениями. В отличие от этапа точной настройки, повторяющегося много раз для каждого слоя, пока сеть не сойдется, здесь этап обучения выполняется один раз. Входными данными каждого слоя являются только данные с предыдущих слоев RBM. В последнем слое выходные данные генерируются с использованием вероятностей, полученных из активаций предыдущих слоев, что и повышает скорость ее обучения. Выбранный для обучения размер параметра batch_size=256.



Р и с. 2. Архитектура DBN
F i g. 2. DBN architecture

3.3 Двунаправленная сеть долговременной краткосрочной памяти

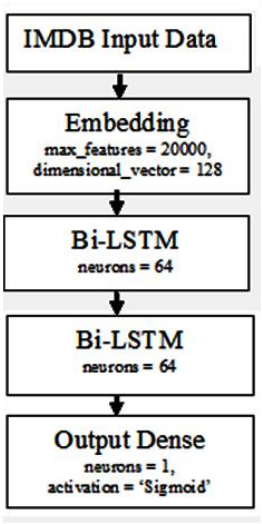
Долгая краткосрочная память *(long short-term memory, LSTM)* – особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям. Сети с длинной краткосрочной памятью *(long short-term memory, LSTM)* – нейронные сети, которые используются в задачах обработки естественного языка *(natural language processing – NLP)*, связанных с длинными последовательностями слов в тексте [26]. Благодаря рекуррентному сегменту, который означает, что выходные данные LSTM возвращаются в себя, эти сети могут использовать при прогнозировании последующей выборки прошлый контекст. Традиционно, LSTM представляли собой односторонние модели, называемые однонаправленными. В них последовательности токенов (слов), читаются только слева направо или только справа налево. В более поздних подходах к построению LSTM для повышения производительности предлагается делать их двунаправленными [27], когда обработка входных данных при обучении модели выполняется и слева направо и справа налево (рисунок 3). Эти BILSTM *(bidirectional LSTM)* используют прошлую и будущую информацию одновременно. В них, чтобы предсказать следующее слово в предложении, используется контекст вокруг слова, а не только слова идущие перед ним. Поэтому в основе эксперимента с последовательностями слов – двунаправленная рекуррентная нейронная сеть BILSTM.



Р и с. 3. Обработка слов текста в BILSTM
F i g. 3. Word Processing in BiLSTM

Набор данных для обучения – IMDB [28] – состоит из 50000 записей, каждая из которых включает два текстовых поля – отзыв (рецензия) и мнение. Поле отзыва содержит текст отзыва о продукте электронной коммерции, фильме, телевизионной программе, видеоигре и т.п., а второе поле содержит эмоциональную окраску отзыва *(negative, positive)*. Поле мнения имеет два значения, что делает этот набор пригодным для бинарной классификации, цель которой – классифицировать отзыв как положительный или отрицательный. Набор из 25 000 обзоров, помечен как обучающий набор, не включает в себя ни одного обзора, входящего в тестовый набор такого же размера. Обзоры предварительно обрабатываются, и каждый кодируется списком индексов слов (целых чисел). Слова индексируются по тому, как часто они встречаются в обучающем наборе, и сохраняются только наиболее часто встречающиеся слова. Например, целое число «3» кодирует третье по частоте слово в данных. Архитектура предлагаемой модели обучения представлена на рисунке 4. Она включает входной слой внедрения *(Embedding)*, два двунаправленных слоя BILSTM и один выходной полносвязный слой *(Dense)*. Слой внедрения преобразует категориальные данные (слова) высокой размерности в векторы фиксированного размера, понятные LSTM, что упрощает обучение сети. Входной параметр этого слоя – заданное количество наиболее часто встречающихся слов в исходном наборе (размер

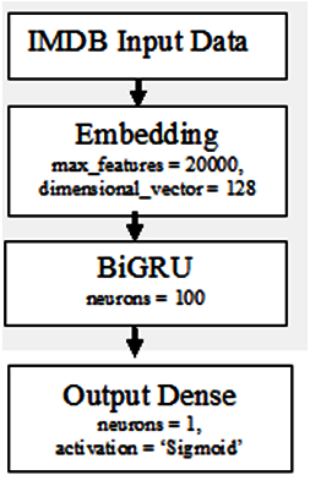
словаря) = 20000. Выходной параметр – размер плотного вектора для каждого слова *(dimensional vector)*. Каждый из двух следующих двунаправленных слоев копирует переданный ему входной слой и переворачивает его поля. При этом будут созданы две копии: одна будет соответствовать входной последовательности, как есть, а другая – перевернутой копии последовательности. По умолчанию выходные значения этого слоя будут объединены. Размер выходных векторов двунаправленных слоев (число нейронов) = 64. Выходной слой *(Dense)* содержит один нейрон *(0 – negative, 1 – positive)*. Параметр обучения batch_size = 128.



Р и с. 4. Архитектура BILSTM
F i g. 4. BILSTM architecture

3.4 Двунаправленный управляемый рекуррентный блок

Управляемый рекуррентный блок *(gated recurrent unit, GRU)* – также разновидность рекуррентной нейронной сети, младший брат более распространенной сети LSTM. GRU использует механизм шлюзования для контроля и управления потоком информации между его ячейками [29]. Был представлен в 2014 году для решения распространенной проблемы исчезающего градиента[5], с которой сталкиваются программисты при обучении LSTM. Как и его брат, GRU способен эффективно сохранять долгосрочные зависимости в последовательных текстовых данных, кроме того, он не обладает внутренней памятью, решая проблему «краткосрочной памяти», от которой страдают LSTM [30]. Учитывая наследие рекуррентных архитектур в моделировании последовательностей и прогнозировании, GRU находится на пути к тому, чтобы затмить LSTM, благодаря более высокой скорости, достигая при этом аналогичной точности и эффективности. При этом у GRU более простая структура и они требуют меньшей памяти. В настоящей работе анализ производительности был реализован на двунаправленной модели BIGRU, обрабатывающей текстовые последовательности. Она включает два GRU: один принимает входные данные в прямом направлении, а другой – в обратном. Сеть BIGRU имеет меньше параметров, чем BILSTM, меньшие вычислительные затраты на обучение и более высокую скорость обучения в задачах предсказания последовательностей [6]. Эксперимент проведен на базе данных IMDB, как и в случае BILSTM. Основанный на корпусе настроений, этот датасет широко применяется с целью проверки качества обучения BIGRU [4]. Архитектура модели представлена на рисунке 5. Ее структура включает слой внедрения *(Embedding)*, расширенный главным двунаправленным слоем BIGRU, преобразующим последовательности высокой размерности в векторы фиксированного размера. Он состоит из прямого и обратного подразделений GRU, включая только шлюз сброса и шлюз обновления, что снижает сложность сети и повышает эффективность вычислений. На выходе подключается полносвязный слой *(Dense)*, единственный нейрон которого завершает бинарную классификацию. Выбранный параметр обучения – размер батча – batch_size = 128.

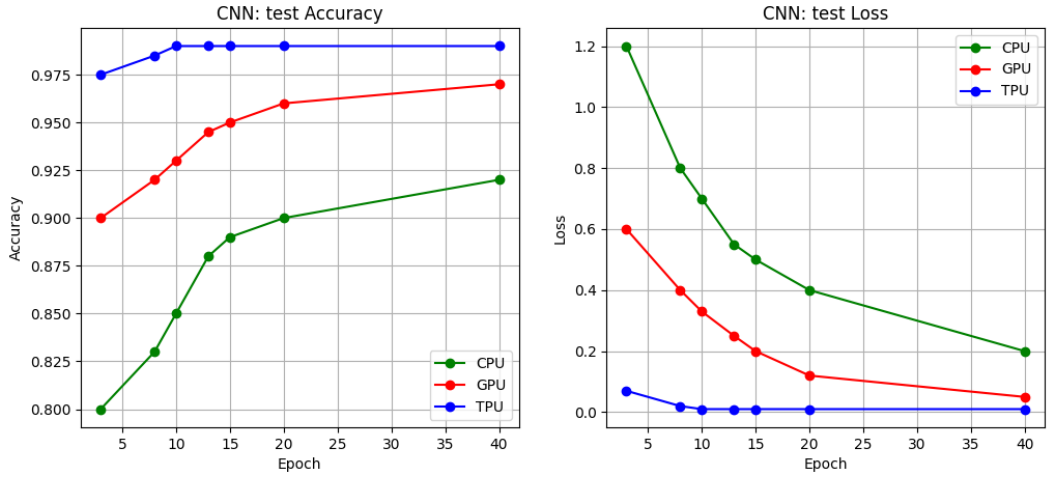


Р и с. 5. Архитектура BIGRU
F i g. 5. BIGRU architecture

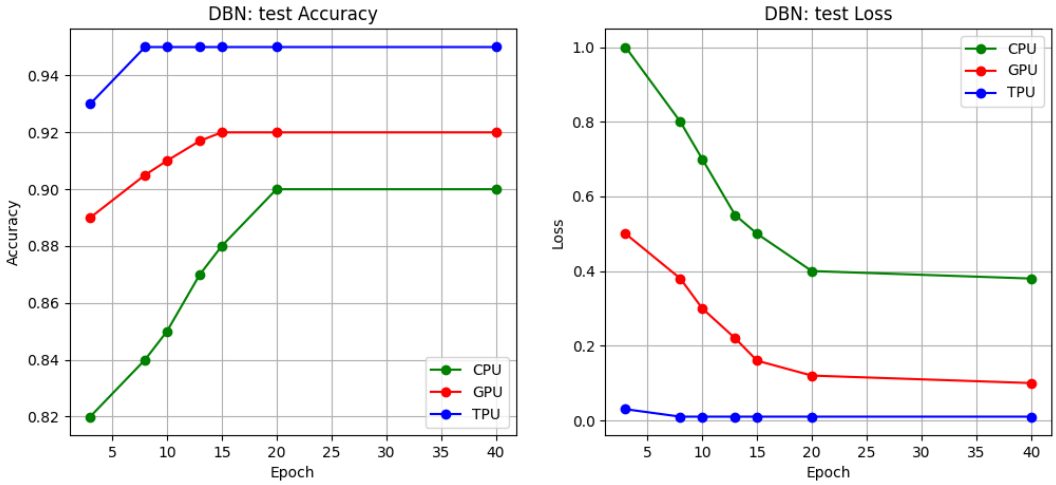
4 Результаты и анализ экспериментов

Обучение выбранных моделей происходит в два этапа: первый – тренировка модели на обучающих данных, второй – проверка модели, включающая оценку её производительности, на тестовых данных. Далее приводятся результаты сравнения производительности отобранных моделей для облачных платформ CPU, GPU и TPU. Оценка производительности выполняется на основании графиков зависимостей метрик Accuracy и Loss от числа эпох обучения на тестовой выборке. Accuracy – это доля правильных классификаций модели, равное отношению числа правильных прогнозов ко всем прогнозам, точность обучения. Значение функции потерь (ошибка) Loss соответствует разнице между истинным и прогнозируемым значением классов модели. На рисунке 6 представлены построенные графики оценок производительности обучения глубоких нейронных сетей CNN (a) и DBN (b). Данные для обучения обеих моделей – набор изображений рукописных цифр MNIST.

[5] Chen M. Vanishing Gradient Problem in training neural networks: Thesis. Australian National University, 2022. 97 p. https://doi.org/10.25911/BMDG-3N85



(a)



(b)

Р и с. 6. Оценка производительности CNN (a) и DBN (b) для CPU, CPU и TPU на данных MNIST
F i g 6. Performance evaluation of CNN(a) and DBN(b) for CPU, CPU and TPU based on MNIST data

Графики показывают, что лучшие результаты значений точности достигаются при обучении CNN на GPU и TPU (94% и 97%). Видно, что большая часть такой точности для TPU сгруппирована на ранних стадиях, соответствующих 7-12 эпохе. Обучение DBN на GPU и TPU дает чуть худшие результаты (92%, 95%), но процесс обучения происходит быстрее (5-10 эпохи). Кривые функции потерь обучения на каждом графике свидетельствуют о том, что для всех аппаратных платформ имеет место устойчивый спад потерь обучения, причем меньшие значения потерь имеет CNN, а более быстрый спад, но до бóльших значений, происходит у DBN. Таким образом, можем заметить разницу в качестве прогноза этих НС – сложная модель CNN дала более точный прогноз. Все кривые демонстрируют улучшение качества прогнозирования на протяжении всего процесса обучения, но после определенного числа эпох дальнейшего улучшения метрик качества для тестовой выборки добиться не удалось. Для оценки времени достижения максимального уровня точности выполнен расчет временных характеристик моделей для всех участвующих в эксперименте аппаратных платформ. В таблице 1 представлены временные параметры и соответствующие им метрики моделей CNN и DBN: время обучения, соответствующие ему максимальное значение точности, ошибка, число эпох и среднее время реализации одной эпохи.

Т а б л и ц а 1. **Временные характеристики обучения CNN, DBN на платформах CPU, GPU, TPU**
T a b l e 1. **Training Time Characteristics of CNN and DBN on CPU, GPU, and TPU Platforms**

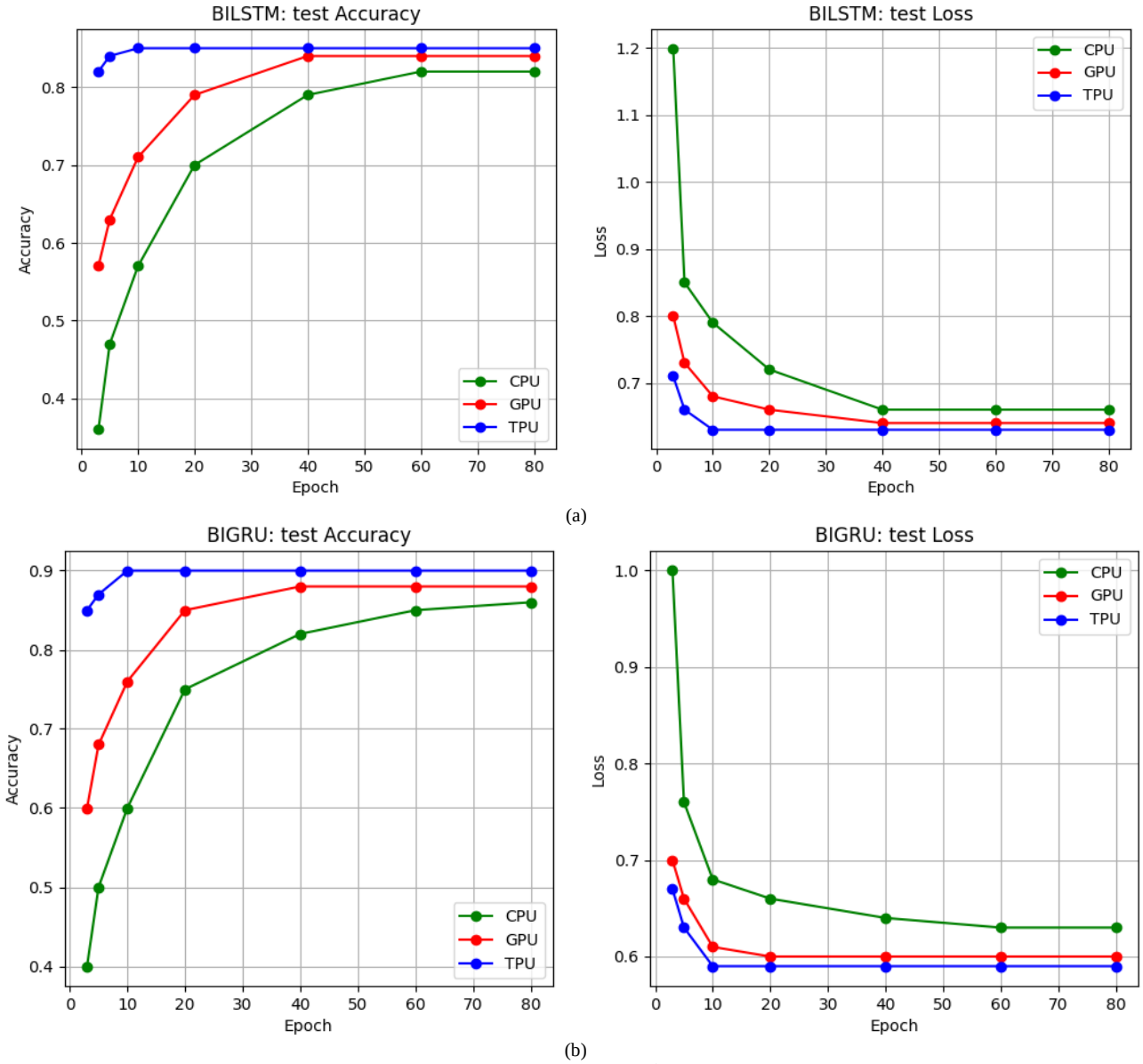
	CNN			DBN		
	CPU	GPU	TPU	CPU	GPU	TPU
Time, s	1404	41	25	960	31	16
Max Test Accuracy, %	0.9	0.94	0.97	0.88	0.92	0.95
Max Train Accuracy,%	0.98	0.99	0.99	0.96	0.99	0.98
Epoch	30	12	7	20	10	5
Avg (Time/epoch),s	46,8	3.4	3.5	48	3.1	3.1
Loss Test	0.35	0.3	0.1	0.4	0.36	0.15
Loss Train	0.01	0.01	0.01	0.01	0.01	0.01

Сравнение временных параметров для CPU, GPU и TPU показывает, что DBN обучается быстрее CNN на всех платформах. При этом время обучения CNN на графическом процессоре сокращается в 34 раза по сравнению с временем на центральном. Тензорный ускоряет обучение CNN в 56 раз. Сокращение времени обучения DBN – в 31 и 60 раз соответственно.

Зависимость оценок измерения производительности глубоких нейронных сетей BILSTM и BIGRU от числа эпох обучения на тестовой выборке представлены на рисунке 7. Данные для обучения моделей – текстовый набор IMDB. Оба графика показывают устойчивый рост точности и уменьшение ошибки с увеличением числа эпох на всех аппаратных платформах. Хотя обе модели работают очень хорошо, поведение BIGRU (b) свидетельствует о лучшей общей производительности параллельных вычислений по сравнению с BILSTM (a). Предельная точность модели BILSTM в случае тестовой выборки составляет 0.8, 0.82 и 0.84 соответственно для CPU, GPU и TPU. Минимальная ошибка – 0.65, 0.63 и 0.6. Для модели BIGRU эти значения составляют: точность – 0.83, 0.86 и 0.9, ошибка – 0.63, 0.6 и 0.6. Уже через 5 эпох предлагаемая BIGRU достигает на TPU точности около 90%, тогда как BILSTM обеспечивает даже на TPU точность 84%.

В таблице 2 приведены временные характеристики и соответствующие им метрики моделей BILSTM и BIGRU для рассматриваемых аппаратных платформ.

Приведенные временные параметры для CPU, GPU и TPU показывают, что время обучения BILSTM на графическом процессоре, по сравнению с CPU, сокращается в 2.7 раза. На тензорном процессоре – в 11 раз. Сокращение времени обучения BIGRU – соответственно в 4 и 12 раз. На диаграмме рисунка 8 приведены размеры отношений времени обучения на ускорителях GPU и TPU к времени обучения на CPU для рассмотренных нейронных сетей.

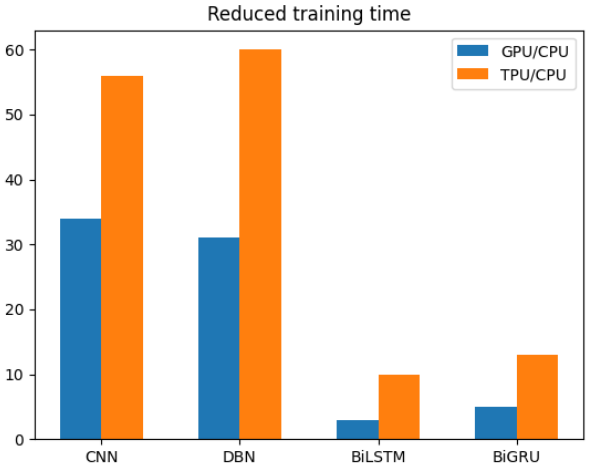


Р и с. 7. Оценка производительности BILSTM (a) и BIGRM (b) для CPU, CPU и TPU на данных IMDB
F i g. 7. Performance evaluation of BILSTM(a) and BIGRM(b) for CPU, CPU and TPU based on IMDB data

Т а б л и ц а 2. **Временные характеристики обучения BILSTM, BIGRU на платформах CPU, GPU, TPU**
T a b l e 2. **Training Time Characteristics of BiLSTM and BiGRU on CPU, GPU, and TPU Platforms**

	BILSTM			BIGRU		
	CPU	GPU	TPU	CPU	GPU	TPU
Time, s	2650	960	230	2205	525	180
Max Test Accuracy, %	0.8	0.82	0.84	0.83	0.86	0.9
Max Train Accuracy,%	0.97	0.99	0.99	0.97	0.99	0.99
Epoch	50	40	10	45	25	10
Avg (Time/epoch),s	53	24	23	49	21	18
Loss Test	0.65	0.63	0.6	0.63	0.6	0.6
Loss Train	0.03	0.01	0.01	0.02	0.01	0.01

Сравнение результатов экспериментов для графических и текстовых данных, представленных в таблицах 1, 2 и рисунке 8, свидетельствует о том, что сокращение времени обучения за счет параллельных вычислений на GPU и TPU гораздо больше для графических данных. Это говорит о том, что оба ускорителя менее гибки для реализации вычислений, не связанных с перемножением матриц. Однако обе рекуррентные сети обрабатывают на TPU текстовые данные более чем в 10 раз быстрее. Поэтому, основываясь на приведенных выше экспериментальных результатах, можно утверждать, что как для графических, так и для текстовых данных стратегия параллельных вычислений значительно повышает производительность рассмотренных моделей глубокого обучения.



Р и с. 8. Сокращение времени обучения на ускорителях
F i g. 8. Reduction of accelerator training time

5 Обсуждение и заключение

Обучение современных глубоких нейронных сетей не представляется возможным без использования аппаратных ускорителей для параллельных вычислений. В статье исследуется производительность ускорителей GPU, TPU в среде Google Colab через Jupyter Notebook для глубокого обучения моделей распознавания изображений и классификации текста. Для сравнения выбраны модели CNN, DBN, BILSTM, BIGRU. Многочисленные прогоны нейронных сетей были выполнены на данных стандартных датасетов MNIST и IMDB. В качестве программной платформы использовалась Python – библиотека TensorFlow. Результаты экспериментов по точности, ошибке и времени обучения сравнивались для CPU, GPU и TPU. Для ускорителей GPU и TPU сравнение показало ускорение обучения в 30 и 60 раз на графических данных и более чем в 4 и 10 раз на текстовых. Можно сделать вывод, что время обработки текстовых данных на графическом и тензорном процессорах умеренно лучше, чем на CPU, при этом точность классификации повышается незначительно. Обработка графических данных на этих ускорителях за счет матричного умножения происходит гораздо быстрее и сопровождается ростом точности. Таким образом, предположение о возможностях повышения производительности параллельной обработки для всех рассмотренных моделей DL подтвердилось.

Очевидно, что алгоритмы DL сейчас находятся на стадии быстрого развития. Однако проблема недостаточной производительности их обучения ограничивает развитие многих отраслей, поэтому способы их распараллеливания останутся в центре внимания будущих исследований.

References

1. Deng L., Yu D. Deep learning: methods and applications. *Foundations and Trends in Signal Processing*. 2014;7(3-4):197-387. https://doi.org/10.1561/2000000039
2. Li X., et al. Deep Learning Attention Mechanism in Medical Image Analysis: Basics and Beyonds. *International Journal of Network Dynamics and Intelligence.* 2023;2(1):93-116. https://doi.org/10.53941/ijndi0201006
3. Bharadiya J. A Comprehensive Survey of Deep Learning Techniques Natural Language Processing. European *Journal of Technology.* 2023;7(1):58-66. https://doi.org/10.47672/ejt.1473
4. Yin X., Liu C., Fang X. Sentiment analysis based on BiGRU information enhancement. *Journal of Physics: Conference Series.* 2021;1748(3):032054. https://doi.org/10.1088/1742-6596/1748/3/032054

5. Mijwil M.M., Salem I.E., Ismaeel M.M. The Significance of Machine Learning and Deep Learning Techniques in Cybersecurity: A Comprehensive Review. *Iraqi Journal For Computer Science and Mathematics*. 2023;4(1):87-101. https://doi.org/10.52866/ijcsm.2023.01.01.008
6. Shiri F.M., et al. A Comprehensive Overview and Comparative Analysis on Deep Learning Models. *Journal on Artificial Intelligence.* 2024;6:301-360. https://doi.org/10.32604/jai.2024.054314
7. Venkatesan R., Li B. Convolutional neural networks in visual computing: a concise guide. CRC Press; 2017. 186 p. https://doi.org/10.4324/9781315154282
8. Naskath J., Sivakamasundari G., Begum A.A.S. A study on different deep learning algorithms used in deep neural nets: MLP SOM and DBN. *Wireless Personal Communications*. 2023;128(4):2913-2936. https://doi.org/10.1007/s11277-022-10079-4
9. Mikolov T., et al. Recurrent neural network based language model. *Interspeech*. 2010;2(3):1045-1048. https://doi.org/10.21437/Interspeech.2010-343
10. Dally W.J., Keckler S.W., Kirk D.B. Evolution of the graphics processing unit (GPU). *IEEE Micro*. 2021;41(6):42-51. https://doi.org/10.1109/MM.2021.3113475
11. Künas C.A., Padoin E.L., Navaux P.O. A. Accelerating Deep Learning Model Training on Cloud Tensor Processing Unit. In: Proceedings of the 13th International Conference on Cloud Computing and Services Science – CLOSER. Vol. 1. SciTePress; 2023. p. 316-323. https://doi.org/10.5220/0012017300003488
12. Qin C., et al. Long short-term memory with activation on gradient. *Neural Networks*. 2023;164:135-145. https://doi.org/10.1016/j.neunet.2023.04.026
13. Patil G.G., Banyal R.K. Techniques of deep learning for image recognition. In: 2019 IEEE 5th International Conference for Convergence in Technology (I2CT). Bombay, India: IEEE Press; 2019. p. 1-5. https://doi.org/10.1109/I2CT45611.2019.9033628
14. Ravikumar A., et al. Effect of neural network structure in accelerating performance and accuracy of a convolutional neural network with GPU/TPU for image analytics. *PeerJ Computer Science*. 2022;8:e909. https://doi.org/10.7717/peerj-cs.909
15. Islam M.S., Ghani N.A. A Novel BiGRUBiLSTM Model for Multilevel Sentiment Analysis Using Deep Neural Network with BiGRU-BiLSTM. In: Ab. Nasir A.F., Ibrahim A.N., Ishak I., Mat Yahya N., Zakaria M.A., P. P. Abdul Majeed A. (eds.) Recent Trends in Mechatronics Towards Industry 4.0. *Lecture Notes in Electrical Engineering*. Vol. 730. Singapore: Springer; 2021. p. 403-414. https://doi.org/10.1007/978-981-33-4597-3_37
16. Khasanah I.N. Sentiment classification using fasttext embedding and deep learning model. *Procedia Computer Science*. 2021;189:343-350. https://doi.org/10.1016/j.procs.2021.05.103
17. Geiping J., Goldstein T. Cramming: training a language model on a single GPU in one day. In: Proceedings of the 40th International Conference on Machine Learning (ICML'23). Vol. 202. JMLR.org; 2023. Article number: 446. p. 11117-11143.
18. Siek M. Benchmarking CPU vs. GPU performance in building predictive LSTM deep learning models. *AIP Conference Proceedings.* 2023;2510(1):030017. https://doi.org/10.1063/5.0128638
19. Kimm H., Paik I., Kimm H. Performance comparision of TPU, GPU, CPU on Google colaboratory over distributed deep learning. In: 2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). Singapore, Singapore: IEEE Press; 2021. p. 312-319. https://doi.org/10.1109/MCSoC51149.2021.00053
20. Zhao X., Wang L., Zhang Y., et al. A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*. 2024;57:99. https://doi.org/10.1007/s10462-024-10721-6
21. Sharma S., Guleria K. Deep learning models for image classification: comparison and applications. In: 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE). Greater Noida, India: IEEE Press; 2022. p. 1733-1738. https://doi.org/10.1109/ICACITE53722.2022.9823516
22. Zhu W. Classification of MNIST handwritten digit database using neural network. In: Proceedings of the research school of computer science. Acton: Australian National University; 2018. Vol. 2601. Available at: https://users.cecs.anu.edu.au/~Tom.Gedeon/conf/ABCs2018/paper/ABCs2018_paper_117.pdf (accessed 06.10.2023).
23. Dewi C., Chen R.C., Hendry, Hung H.T. Comparative Analysis of Restricted Boltzmann Machine Models for Image Classification. In: Nguyen N., Jearanaitanakij K., Selamat A., Trawiński B., Chittayasothorn S. (eds.) Intelligent Information and Database Systems. ACIIDS 2020. *Lecture Notes in Computer Science*. Vol. 12034. Cham: Springer; 2020. p. 285-296. https://doi.org/10.1007/978-3-030-42058-1_24
24. Yang T., Silver D.L. The Disadvantage of CNN versus DBN Image Classification Under Adversarial Conditions. *Canadian AI 2021.* Canadian Artificial Intelligence Association (CAIAC); 2021. Article number: 2021S18. https://doi.org/10.21428/594757db.b65acd40
25. Liu G., Xiao L., Xiong C. Image classification with deep belief networks and improved gradient descent. In: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC). Guangzhou, China: IEEE Press; 2017. p. 375-380. https://doi.org/10.1109/CSE-EUC.2017.74
26. Hochreiter S., Schmidhuber J. Long short-term memory. *Neural computation.* 1997;9(8):1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

27. Kim J., Lee J.H. Multiple range-restricted bidirectional gated recurrent units with attention for relation classification. *arXiv:*1707.01265. 2017. https://doi.org/10.48550/arXiv.1707.01265
28. Tripathi S., et al. Analyzing sentiment using IMDb dataset. In: 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN). Bhimtal, India: IEEE Press; 2020. p. 30-33. https://doi.org/10.1109/CICN49253.2020.9242570
29. Cho K., et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics; 2013. p. 1724-1734. https://doi.org/10.3115/v1/D14-1179
30. Qin C., et al. Long short-term memory with activation on gradient. *Neural Networks*. 2023;164:135-145. https://doi.org/10.1016/j.neunet.2023.04.026

Поступила 06.10.2023; одобрена после рецензирования 04.11.2023; принята к публикации 29.11.2023.

Submitted 06.10.2023; approved after reviewing 04.11.2023; accepted for publication 29.11.2023.

Об авторе:

Самойлова Татьяна Аркадьевна, доцент кафедры прикладной математики и информатики физико-математического факультета, ФГБОУ ВО «Смоленский государственный университет» (214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4), кандидат технических наук, доцент, **ORCID: https://orcid.org/0000-0002-3712-327X,** tatsamoilova24@gmail.com

Автор прочитал и одобрил окончательный вариант рукописи.

About the author:

Tatiana A. Samoilova, Associate Professor of the Chair of Applied Mathematics and Informatics, Faculty of Physics and Mathematics, Smolensk State University, Smolensk (4 Przhevalsky St., Smolensk 214000, Russian Federation), Cand. Sci. (Eng.), Associate Professor, **ORCID: https://orcid.org/0000-0002-3712-327X,** tatsamoilova24@gmail.com

The author has read and approved the final manuscript.