Вестник Томского государственного архитектурно-строительного университета. 2023. Т. 25. № 5. С. 84–94.

ISSN 1607-1859 (для печатной версии)
ISSN 2310-0044 (для электронной версии)

Vestnik Tomskogo gosudarstvennogo arkhitekturno-stroitel'nogo universiteta – Journal of Construction and Architecture. 2023; 25 (5): 84–94.
Print ISSN 1607-1859
Online ISSN 2310-0044

НАУЧНАЯ СТАТЬЯ
УДК 721
DOI: 10.31675/1607-1859-2023-25-5-84-94 EDN: TFEVAN

ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНАЯ СЕТЬ КАК ОСНОВА ИНТЕЛЛЕКТУАЛЬНОЙ МОДЕЛИ ФОРМИРОВАНИЯ ИЗОБРАЖЕНИЙ АРХИТЕКТУРНЫХ ОБЪЕКТОВ ЗАДАННОГО СТИЛЯ ПО ИХ ТЕКСТОВОМУ ОПИСАНИЮ

Петр Андреевич Пылов[1], Анна Владимировна Дягилева[1], Евгения Александровна Николаева[1], Роман Вячеславович Майтак[1], Татьяна Анатольевна Шалыгина[2]

[1]Кузбасский государственный технический университет имени Т.Ф. Горбачева, г. Кемерово, Россия
[2]Томский государственный архитектурно-строительный университет, г. Томск, Россия

Аннотация. *Актуальность* тематики основывается на растущих темпах урбанизации и цифровизации современного общества: чтобы идти в ногу со временем, архитектура градостроительства должна отвечать не только новейшим эстетическим требованиям, но и критерию скорости разработки проектов будущих зданий. Очевидно, что сократить время реализации проектов новых зданий возможно на основе внедрения современных информационных технологий в процесс разработки архитектурного концепта.

Основной целью научной статьи является реализация модели машинного представления генерации изображений заданных пользователем архитектурных объектов выбранного стиля.

Одной из вариаций подобной информационно-интеллектуальной системы является авторская модель машинного обучения, основанная на генеративно-состязательных нейронных сетях, которые открывают возможность генерации изображений здания выбранного архитектурного стиля на основе текстового описания пользователя.

Вывод: рассматриваемая автоматизирующая система позволит существенно сократить временные, человеческие и денежные ресурсы, требуемые для разработки проекта будущего здания.

Ключевые слова: искусственный интеллект, прикладное машинное обучение, градостроительство, архитектура зданий

Для цитирования: Пылов П.А., Дягилева А.В., Николаева Е.А., Майтак Р.В., Шалыгина Т.А. Генеративно-состязательная сеть как основа интеллектуальной модели формирования изображений архитектурных объектов заданного стиля по их текстовому описанию // Вестник Томского государственного архитектурно-строительного университета. 2023. Т. 25. № 5. С. 84–94. DOI: 10.31675/1607-1859-2023-25-4-84-94. EDN: TFEVAN

ORIGINAL ARTICLE

GENERATIVE ADVERSARIAL NETWORK AS A BASIS FOR INTELLIGENT MODEL OF IMAGING ARCHITECTURAL OBJECTS BASED ON TEXTUAL DESCRIPTION

Petr A. Pylov[1], Anna V. Dyagileva[1], Evgenija A. Nikolaeva[1], Roman V. Maitak[1], Tat'jana A. Shalygina[2]
[1]*Gorbachev Kuzbass State Technical University, Kemerovo, Russia*
[2]*Tomsk State University of Architecture and Building, Tomsk, Russia*

Abstract. Due to the high technology integrated into a person's daily life (smart house), this topic is relevant. One of elements of generative adversarial network is robot vacuum cleaners of various surface. Difficulties caused by this technique largely depend on the environment in which it locates.

Purpose: The development of the convolutional neural network concept allowing real-time distinguishing between the building interior and exterior.

Practical implication: The proposed intelligent system can distinguish between the building interior and exterior, that will considerably improve the firmware performance of modern technology in both the domestic and industrial segments.

Keywords: artificial intelligence, applied machine learning, urban planning, building architecture

For citation: Pylov P.A., Dyagileva A.V., Nikolaeva E.A., Maitak R.V., Shalygina T.A. Generative adversarial network as a basis for intelligent model of imaging architectural objects based on textual description. Vestnik Tomskogo gosudarstvennogo arkhitekturno-stroitel'nogo universiteta – Journal of Construction and Architecture. 2023; 25 (5): 84–94. DOI: 10.31675/1607-1859-2023-25-4-84-94. EDN: TFEVAN

Урбанизация общества закономерно вызывает увеличение темпов строительства новых зданий: жилых комплексов, деловых районов и других элементов инфраструктуры, свойственных городской среде. Городские жители ежедневно сталкиваются с этими элементами инфраструктуры, но лишь немногие задумываются над количеством ресурсов, необходимых для разработки, например, одного жилого комплекса на выделенном участке земли.

Между тем разработка любой атомарной части городской архитектуры сопровождается изучением как физических (тип грунта, близость водоемов и т. д.), так и инфраструктурных особенностей местности, на которой планируется застройка [1]. Даже эти критерии предъявляют высокие требования к ресурсам застройщика.

Кроме того, в крупных городах необходимо учитывать требования единообразия стиля застройки и/или гармоничного сочетания соседствующих комплексов друг с другом [2].

Соблюдение всех требований, предъявляемых к застройке, закономерно повышает период времени, необходимый для разработки проекта здания застройщиком. Сфокусировав внимание на данной функциональной связи, можно сделать логичный вывод: если определение физических и технологических критериев действительно требует высокой ответственности и всецелого участия инженеров, то формирование графического представления архитектурного сти-

ля является той творческой задачей, автоматизировать которую можно современными интеллектуальными алгоритмами [3].

Перед тем как приступить к непосредственной реализации модели машинного обучения, необходимо детерминировать входной и ожидаемый выходной результат функционирования алгоритма прикладного искусственного интеллекта. Ожидаемым выходным результатом является изображение, которое характеризует заданный архитектурный стиль здания.

В свою очередь, базовым входным информационным потоком интеллектуальной системы будет являться информация о том, что желает получить пользователь на изображении. Входную информацию удобнее всего представить для пользователя в виде текстового поля, т. к. естественный язык является универсальным инструментом кодирования требований к системе, единственный потенциальный недостаток которого – большой объем текстовых описаний для детальной обрисовки желаемого результата.

После определения всех необходимых потоков данных для реализации решения перейдем к разработке программного модуля интеллектуальной системы. Следует заметить, что основой решения будут выступать генеративно-состязательные нейронные сети, поскольку они позволяют итеративно совершенствовать полученную базовую точность в процессе своего дальнейшего функционирования [4].

Логика работы генеративно-состязательных сетей в рамках рассматриваемой задачи может быть представлена в виде схемы (рис. 1).

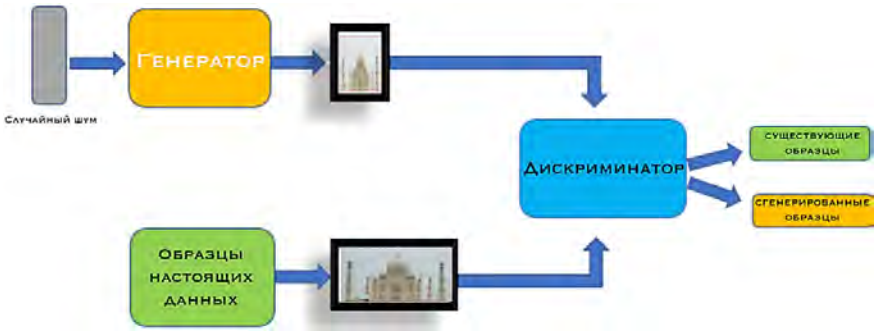

Рис. 1. Логика функционирования генеративно-состязательных сетей, на основе которых работает интеллектуальная система генерации изображений зданий архитектурного стиля. Создано с помощью программного продукта Adobe Photoshop 2022

Fig. 1. The logic of generative adversarial network functioning used for imaging the building architecture. The image is created in Adobe Photoshop 2022

Согласно логике функционирования генеративно-состязательной сети (рис. 1), интеллектуальная модель получает в качестве обучающей выборки данных известные изображения, которым присвоена метка целевого класса архитектурного стиля (образцы настоящих данных/существующие образцы). В процессе обучения модель машинного обучения агрегирует наиболее частные особенности и закономерности каждого архитектурного стиля [5].

После того, как модель прикладного искусственного интеллекта будет обучена, наступает процесс её прикладного применения. На основе получен-

ного текстового запроса визуализации конкретного архитектурного стиля здания (или сложного смешения архитектурных стилей в одном общем здании) интеллектуальная система старается «понять» смысл того, что передал пользователь, и реконструирует изображение на основе уже известных общих закономерностей и специфических особенностей архитектурных стилей [6].

В подавляющем большинстве случаев изображение будет именно генерироваться, а не восстанавливаться из обучающей выборки, т. к. один и тот же архитектурный стиль может быть изображен интеллектуальной системой в разных ракурсах – при соответствующей формулировке запроса.

Генератор, находящийся в составе интеллектуальной системы, обеспечивает подбор такого изображения, которое будет наиболее полно соответствовать описанию текста, полученного от пользователя. Программная реализация модуля генератора (рис. 1) интеллектуальной модели представлена на рис. 2.

```
# Модель генератора интеллектуальной системы
class Generator(torch.nn.Module):
    def __init__(self, input_dim, label_dim, num_filters, output_dim):
        super(Generator, self).__init__()
        # Скрытые слои нейронной сети
        self.hidden_layer1 = torch.nn.Sequential()
        self.hidden_layer2 = torch.nn.Sequential()
        self.hidden_layer = torch.nn.Sequential()
        for i in range(len(num_filters)):
            if i == 0:
                input_deconv = torch.nn.ConvTranspose2d(input_dim, int(num_filters[i]/2),/
                                                        kernel_size=4, stride=1, padding=0)
                self.hidden_layer1.add_module('input_deconv', input_deconv)
                torch.nn.init.normal_(input_deconv.weight, mean=0.0, std=0.02)
                torch.nn.init.constant_(input_deconv.bias, 0.0)
                self.hidden_layer1.add_module('input_bn', torch.nn.BatchNorm2d(int(num_filters[i]/2)))
                self.hidden_layer1.add_module('input_act', torch.nn.ReLU())
                label_deconv = torch.nn.ConvTranspose2d(label_dim, int(num_filters[i]/2), /
                                                        kernel_size=4, stride=1, padding=0)
                self.hidden_layer2.add_module('label_deconv', label_deconv)
                torch.nn.init.normal_(label_deconv.weight, mean=0.0, std=0.02)
                torch.nn.init.constant_(label_deconv.bias, 0.0)
                self.hidden_layer2.add_module('label_bn', torch.nn.BatchNorm2d(int(num_filters[i]/2)))
                self.hidden_layer2.add_module('label_act', torch.nn.ReLU())
            else:
                deconv = torch.nn.ConvTranspose2d(num_filters[i-1], num_filters[i], /
                                                  чkernel_size=4, stride=2, padding=1)
                deconv_name = 'deconv' + str(i + 1)
                self.hidden_layer.add_module(deconv_name, deconv)
                torch.nn.init.normal_(deconv.weight, mean=0.0, std=0.02)
                torch.nn.init.constant_(deconv.bias, 0.0)
                bn_name = 'bn' + str(i + 1)
                self.hidden_layer.add_module(bn_name, torch.nn.BatchNorm2d(num_filters[i]))
                act_name = 'act' + str(i + 1)
                self.hidden_layer.add_module(act_name, torch.nn.ReLU())

        # Выходной слой сети
        self.output_layer = torch.nn.Sequential()
        out = torch.nn.ConvTranspose2d(num_filters[i], output_dim, kernel_size=4, stride=2, padding=1)
        self.output_layer.add_module('out', out)
        torch.nn.init.normal_(out.weight, mean=0.0, std=0.02)
        torch.nn.init.constant_(out.bias, 0.0)
        self.output_layer.add_module('act', torch.nn.Tanh())
    def forward(self, z, c):
        h1 = self.hidden_layer1(z)
        h2 = self.hidden_layer2(c)
        x = torch.cat([h1, h2], 1)
        h = self.hidden_layer(x)
        out = self.output_layer(h)
        return out
```

Рис. 2. Программная реализация модуля генератора. Получено с помощью снимка экрана из среды разработки JupyterNotebook

Fig. 2. Software implementation of the generator module. Screenshot from the Jupyter Notebook application

Генеративно-состязательная сеть выделяется среди всего класса моделей машинного обучения наличием дискриминатора, главной задачей которого является сопоставление полученного от генератора изображения с текстом, переданным пользователем. Авторская реализация дискриминатора представлена на рис. 3.

```
# Модель дискриминатора интеллектулаьной системы
class Discriminator(torch.nn.Module):
    def __init__(self, input_dim, label_dim, num_filters, output_dim):
        super(Discriminator, self).__init__()
        self.hidden_layer1 = torch.nn.Sequential()
        self.hidden_layer2 = torch.nn.Sequential()
        self.hidden_layer = torch.nn.Sequential()
        for i in range(len(num_filters)):
            # Сверточные слои нейронной сети
            if i == 0:
                input_conv = torch.nn.Conv2d(input_dim, int(num_filters[i]/2), kernel_size=4, stride=2, padding=1)
                self.hidden_layer1.add_module('input_conv', input_conv)
                torch.nn.init.normal_(input_conv.weight, mean=0.0, std=0.02)
                torch.nn.init.constant_(input_conv.bias, 0.0)
                self.hidden_layer1.add_module('input_act', torch.nn.LeakyReLU(0.2))
                label_conv = torch.nn.Conv2d(label_dim, int(num_filters[i]/2), kernel_size=4, stride=2, padding=1)
                self.hidden_layer2.add_module('label_conv', label_conv)
                torch.nn.init.normal_(label_conv.weight, mean=0.0, std=0.02)
                torch.nn.init.constant_(label_conv.bias, 0.0)
                self.hidden_layer2.add_module('label_act', torch.nn.LeakyReLU(0.2))
            else:
                conv = torch.nn.Conv2d(num_filters[i-1], num_filters[i], kernel_size=4, stride=2, padding=1)
                conv_name = 'conv' + str(i + 1)
                self.hidden_layer.add_module(conv_name, conv)
                torch.nn.init.normal_(conv.weight, mean=0.0, std=0.02)
                torch.nn.init.constant_(conv.bias, 0.0)
                bn_name = 'bn' + str(i + 1)
                self.hidden_layer.add_module(bn_name, torch.nn.BatchNorm2d(num_filters[i]))
                act_name = 'act' + str(i + 1)
                self.hidden_layer.add_module(act_name, torch.nn.LeakyReLU(0.2))

        # Выходные слои нейронной сети
        self.output_layer = torch.nn.Sequential()
        # Сверточные слои нейронной сети
        out = torch.nn.Conv2d(num_filters[i], output_dim, kernel_size=4, stride=1, padding=0)
        self.output_layer.add_module('out', out)
        torch.nn.init.normal_(out.weight, mean=0.0, std=0.02)
        torch.nn.init.constant_(out.bias, 0.0)
        self.output_layer.add_module('act', torch.nn.Sigmoid())
    def forward(self, z, c):
        h1 = self.hidden_layer1(z)
        h2 = self.hidden_layer2(c)
        x = torch.cat([h1, h2], 1)
        h = self.hidden_layer(x)
        out = self.output_layer(h)
        return out
```

Рис. 3. Программная реализация модуля дискриминатора интеллектуальной системы. Получено с помощью снимка экрана из среды разработки JupyterNotebook

Fig. 3. Software implementation of the intelligent system discriminator module. Screenshot from the Jupyter Notebook application

Исходя из терминологии, можно отметить, что задачи генератора и дискриминатора являются обратными [7]. Более того, дискриминатор и генератор конкурируют друг с другом, что позволяет получить более высокие показатели точности в задаче генерации изображений результирующей моделью машинного обучения [8].

Оценку итоговой точности разработанной модели лучше всего обобщать на основании сгенерированных изображений, т. к. по ним можно сопоставить текстовое описание с полученным изображением (рис. 4).

Соотнести полноту генерации визуального архитектурного представления на основании текстового описания можно на основе передачи информации об уже известном существующем здании.

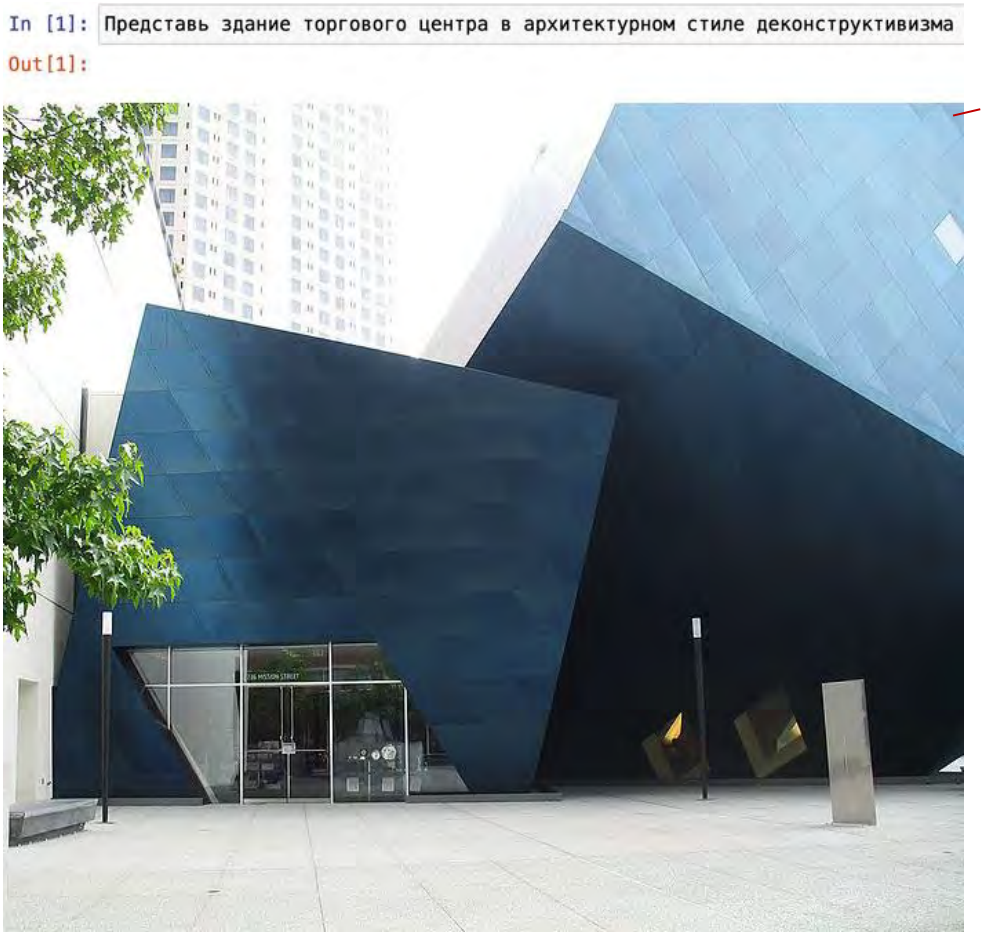

Рис. 4. Процесс генерации изображения по запросу пользователя. Получено с помощью снимка экрана из среды разработки и запуска проектов JupyterNotebook
Fig. 4. Imaging process by user's request. Screenshot from the Jupyter Notebook application

В качестве примера рассмотрим, насколько хорошо будет соотноситься генерация особых критериев об условно известном объекте – церкви (рис. 5), выдержанной в стиле архитектуры русского Возрождения (неорусский стиль).

Особое внимание следует обратить на тот факт, что первоначально у интеллектуальной модели с ключевым словом «церковь» ассоциировалась только фотография краснокаменного храма периода русского зодчества начала XX в. (рис. 6).

Как следует из соотнесения данных, приведенных на рис. 5 и 6, интеллектуальная модель сформировала достаточную для генерации обобщающую способность даже для наиболее малочисленных элементов набора данных.

Стоит отметить, что любая разработка интеллектуальной модели в первую очередь должна быть ориентирована на решение конкретной предметно-прикладной задачи. Представленная в рамках данной статьи интеллектуальная модель может быть использована для решения нескольких задач.

```
In [3]: Визуализируй церковь, выполненную из белого кирпича, которая будет
        выдержана в неорусском стиле или стиле Русского Возрождения

Out[3]:
```

Рис. 5. Процесс генерации изображения по запросу пользователя. Получено с помощью снимка экрана из среды разработки и запуска проектов JupyterNotebook
Fig. 5. Imaging process by user's request. Screenshot from the Jupyter Notebook application

1. Автоматизация создания эскизного проекта нового здания на основе заданного пользователем архитектурного стиля. Создание проекта здания часто требует значительных временных и человеческих затрат, но автоматизация процесса на начальной стадии вариантного проектирования позволит сократить время на выбор заказчиком облика объекта и, соответственно, сократить этот этап проектирования.

2. Моделирование будущей городской среды. Данный процесс зависит от многих критериев. Поскольку текстовое описание множества объектов архитектурной среды достаточно сложный процесс, то с помощью разработан-

ного интеллектуального решения, при соответствующей базе возможных архитектурных решений, открывается перспектива выполнить моделирование городских агломераций, имея при этом минимальное количество ресурсов – один персональный компьютер. Безусловно, подобные модели являются только общим примером и далеки от рабочих чертежей объекта, но общее графическое представление объекта в 3D-модели позволяет составить наглядное представление о нем.

Рис. 6. Первоначальная ассоциация интеллектуальной модели с термином «церковь». Графическая информация взята из обучающей выборки данных
Fig. 6. Initial association of the intelligent model with the term church. Graphical information is taken from training sample

3. Восстановление памятников архитектуры, у которых отсутствуют проектные чертежи. История человечества насчитывает множество примеров и случаев, когда памятники архитектуры были подвергнуты частичному или полному разрушению в силу различных причин: например, 15 апреля 2019 г. в соборе Парижской Богоматери (Нотр-Дам-де-Пари) возник пожар, который

смогли потушить только к утру [10]. В результате пожара пострадал шпиль и обрушилась крыша собора, при этом основные конструкции и произведения искусства уцелели, но, несмотря на относительно небольшие разрушения, восстанавливать кафедральный собор архиепархией Парижа было решено на основе материалов компьютерной игры [11], т. к. оригинальных чертежей не сохранилось. Учитывая тот факт, что собор в компьютерной игре являлся непрофильным атрибутом, стоит отметить, что применение разработанной интеллектуальной модели может помочь в подобных ситуациях значительно больше: на основе сгенерированного моделью макета здания можно будет получить трехмерный макет для использования в качестве отправной точки для восстановления памятников архитектуры.

Выводы

Разработанная автоматизирующая модель демонстрирует высокие показатели эффективности и прецизионности, что является весомым аргументом и основанием для прикладного внедрения программного решения в предметно-областные задачи генерации изображений архитектурных объектов выбранного стиля по их текстовому описанию. Данные изображения будут становиться все более точными с пополнением базы изображений подобных объектов.

Отдельно стоит отметить правильный выбор инструмента решения – генеративно-состязательную сеть. Генеративно-состязательные сети высоко зарекомендовали себя в задаче генерации изображений, поэтому при разработке концепта решения в первую очередь необходимо ориентироваться на общую постановку задачи и только во вторую – улучшать точность программной реализации визуального облика объекта. Конечно, на данном этапе полученные изображения далеки от реальных проектных решений и позволяют составить лишь общее представление об объекте, однако со временем данные разработки, возможно, заменят весь цикл архитектурных и инженерных работ при подготовке соответствующей проектно-сметной документации.

Однако при всех преимуществах авторской разработки было бы честным признать тот факт, что для максимально быстрого функционирования автоматизирующей модели исходный программный код желательно реализовать на низкоуровневом языке программирования (например, Сили C++), а не на высокоуровневым (в настоящем проекте использовался Python версии 3.9.2). При выполнении сложных запросов такой подход позволит сократить до 4 с времени на генерацию одного изображения архитектурного стиля здания [9, 10].

Список источников

1. *Campanario G.* The urban sketching handbook: architecture and cityscapes: tips and techniques for drawing on location. Quarry Books. 2014. 112 c. ISBN 9781592539611.
2. *Radford A., Srivastava A., Morcoç S.B.* The elements of modern architecture: understanding contemporary buildings. Thames & Hudson. 2014. 344 c. ISBN 978-0500023624.
3. *Bengio Y.* Practical Recommendations for gradient-based training of deep architectures // Arxiv : [сайт]. URL: https://doi.org/10.48550/arXiv.1206.5533 (дата обращения: 28.08.2023).
4. *Wolohan J.T.* Mastering large datasets with Python. Manning Press. 2020. 312 c. ISBN 9781617296239.

5. *Николенко С., Кадурин А., Архангельская Е.* Глубокое обучение. Погружение в мир нейронных сетей. Санкт-Петербург : Издательский дом «Питер», 2018. 482 с. ISBN 978-5-496-02536-2.
6. *Kopec D.* Classic Computer Science Problems in Python. Manning Shelter Island. 2019. 201 с. ISBN 9781617295980.
7. *Пылов П.А., Майтак Р.В., Дягилева А.В.* Разработка интеллектуальных систем для обработки сигналов с датчиков давления. Москва ; Вологда : Инфра-Инженерия, 2023. 172 с. ISBN 978-5-9729-1594-1.
8. *Abrahams S., Hafner D.* TensorFlow for machine learning intelligence. Bleeding Edge Press. 2019. 245 с.
9. *Аггарвал Ч.* Нейронные сети и глубокое обучение: перевод с английского. Санкт-Петербург : ООО «Диалектика», 2020. 752 с. ISBN 978-5-907203-01-3.
10. *Когалов Ю.* Криминалисты назвали причину пожара в Нотр-Даме // RGRU.Российская газета : [сайт]. URL: https://rg.ru/2019/04/17/kriminalisty-nazvali-prichinu-pozhara-v-notr-dame.html (дата обращения: 08.10.2023).
11. *Собор Парижской Богоматери* восстановят благодаря игре Assassin's Creed // Росбалт.RU : [сайт].URL: https://www.rosbalt.ru/world/2019/04/16/1776251.html (дата обращения: 08.10.2023).

REFERENCES

1. *Campanario G.* The urban sketching handbook: Architecture and cityscapes: Tips and techniques for drawing on location. Quarry Books. 2014. 112 p. ISBN 9781592539611.
2. *Radford A., Srivastava A., Morcoç S.B.* The elements of modern architecture: Understanding contemporary buildings. Thames & Hudson. 2014. 344 p. ISBN 978-0500023624.
3. *Bengio Y.* Practical recommendations for gradient-based training of deep architectures. https://doi.org/10.48550/arXiv.1206.5533 (accessed August 28, 2023).
4. *Wolohan J.T.* Mastering large datasets with Python. Manning Press. 2020. 312 p. ISBN 9781617296239.
5. *Nikolenko S., Kadurin A., Arkhangelskaya E.* Deep learning. Immersion in the world of neural networks. Saint-Petersburg: Piter, 2018. 482 p. ISBN 978-5-496-02536-2. (In Russian)
6. *Kopec D.* Classic computer science problems in Python. Manning Shelter Island. 2019. 201 p. ISBN 9781617295980.
7. *Pylov P.A., Maitak R.V., Dyagileva A.V.* Development of intelligent systems for processing signals from pressure sensors. Moscow, Vologda: Infra-Engineering, 2023. 172 p. ISBN 978-5-9729-1594-1. (In Russian)
8. *Abrahams S., Hafner D.* Tensor flow for machine learning intelligence. Bleeding Edge Press. 2019. 245 p.
9. *Aggarwal Ch.* Neural networks and deep learning. Saint-Petersburg: Dialektika, 2020. 752 p. ISBN 978-5-907203-01-3. (Russian translation)
10. *Kogalov Yu.* Forensic experts have named the cause of the Notre Dame fire. Available: https://rg.ru/2019/04/17/kriminalisty-nazvali-prichinu-pozhara-v-notr-dame.html (accessed October 8, 2023).
11. The Cathedral of Notre Dame de Paris will be restored thanks to the Assassin's Creed game. Available: www.rosbalt.ru/world/2019/04/16/1776251.html (accessed October 8, 2023).

Сведения об авторах

Пылов Петр Андреевич, аспирант, Кузбасский государственный технический университет имени Т.Ф. Горбачева, 650000, г. Кемерово, ул. Весенняя, 28, gedrosten@mail.ru

Дягилева Анна Владимировна, канд. техн. наук, доцент, Кузбасский государственный технический университет имени Т.Ф. Горбачева, 650000, г. Кемерово, ул. Весенняя, 28, dyagileva@mail.ru

Николаева Евгения Александровна, канд. физ.-мат. наук, доцент, зав. кафедрой, Кузбасский государственный технический университет имени Т.Ф. Горбачева, 650000, г. Кемерово, ул. Весенняя, 28, nikolaevaea@yandex.ru

Майтак Роман Вячеславович, магистрант, Кузбасский государственный технический университет имени Т.Ф. Горбачева, 650000, г. Кемерово, ул. Весенняя, 28, super-energy@mail.ru

Шалыгина Татьяна Анатольевна, канд. техн. наук, доцент, Томский государственный архитектурно-строительный университет, 634003, г. Томск, пл. Соляная, 2, shal53@mail.ru

Authors Details

Petr A. Pylov, Research Assistant, Gorbachev Kuzbass State Technical University, 28, Vesennyaya Str., 650000, Kemerovo, Russia, gedrosten@mail.ru

Anna V. Dyagileva, PhD, A/Professor, Gorbachev Kuzbass State Technical University, 28, Vesennyaya Str., 650000, Kemerovo, Russia, dyagileva@mail.ru

Evgenia A. Nikolaeva, PhD, A/Professor, Gorbachev Kuzbass State Technical University, 28, Vesennyaya Str., 650000, Kemerovo, Russia, nikolaevaea@yandex.ru

Roman V. Maitak, Graduate Student, Gorbachev Kuzbass State Technical University, 28, Vesennyaya Str., 650000, Kemerovo, Russia, super-energy@mail.ru

Tat'jana A. Shalygina, PhD, A/Professor, Tomsk State University of Architecture and Building, 2, Solyanaya Sq., 634003, Tomsk, Russia, shal53@mail.ru

Вклад авторов

Все авторы сделали эквивалентный вклад в подготовку публикации.
Авторы заявляют об отсутствии конфликта интересов.

Authors contributions

The authors contributed equally to this article.
The authors declare no conflicts of interests.

Статья поступила в редакцию 22.09.2023
Одобрена после рецензирования 06.10.2023
Принята к публикации 09.10.2023

Submitted for publication 22.09.2023
Approved after review 06.10.2023
Accepted for publication 09.10.2023