ТЕХНИЧЕСКИЕ НАУКИ

Научная статья
УДК 004.032.26:58
doi: 10.34822/1999-7604-2022-4-6-13

АНАЛИЗ СОСТОЯНИЯ РАСТЕНИЙ С ПРИМЕНЕНИЕМ ТЕХНОЛОГИЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Валентин Валерьевич Брыкин [1], ***Михаил Яковлевич Брагинский*** [2], ***Ирина Олеговна Тараканова*** [3], ***Дмитрий Викторович Тараканов*** [4]✉
[1, 2, 3, 4] *Сургутский государственный университет, Сургут, Россия*
[1] *valentin.brykin@mail.ru, https://orcid.org/0000-0002-6143-8992*
[2] *braginskiy_mya@surgu.ru, https://orcid.org/0000-0003-1332-463X*
[3] *tarakanova_io@surgu.ru, https://orcid.org/0000-0002-7707-5890*
[4] *sprtdv@mail.ru* ✉*, https://orcid.org/0000-0003-1851-1039*

Аннотация. Представлены этапы разработки с помощью языка разметки HTML, каскадных таблиц стилей и языка программирования JavaScript веб-приложения для мобильных устройств, содержащего в качестве основы обученную средствами библиотек машинного обучения Tensorflow и Keras модель искусственной нейронной сети для классификации с максимальной точностью состояния растений.

Все операции, связанные с обработкой изображений, выполнены в цветовом пространстве RGB.

Ключевые слова: сверточные нейронные сети, классификация, веб-приложение

Для цитирования: *Брыкин В. В., Брагинский М. Я., Тараканова И. О., Тараканов Д. В. Анализ состояния растений с применением технологий искусственного интеллекта // Вестник кибернетики. 2022. № 4 (48). С. 6–13. DOI 10.34822/1999-7604-2022-4-6-13.*

Original article

ANALYSIS OF PLANTS HEALTH USING ARTIFICIAL INTELLIGENCE TECHNOLOGIES

Valentin V. Brykin [1], ***Mikhail Ya. Braginsky*** [2], ***Irina O. Tarakanova*** [3], ***Dmitry V. Tarakanov*** [4]✉
[1, 2, 3, 4] *Surgut State University, Surgut, Russia*
[1] *valentin.brykin@mail.ru, https://orcid.org/0000-0002-6143-8992*
[2] *braginskiy_mya@surgu.ru, https://orcid.org/0000-0003-1332-463X*
[3] *tarakanova_io@surgu.ru, https://orcid.org/0000-0002-7707-5890*
[4] *sprtdv@mail.ru* ✉*, https://orcid.org/0000-0003-1851-1039*

Abstract. The article describes the stages of developing a web-application for mobile phones based on an artificial neural network model trained by machine learning libraries (Tensorflow and Keras) to classify plant health with maximum accuracy. The HTML markup language, cascading style sheets, and JavaScript programming language were all used during the development process.

All photo editing manipulations were performed using the RGB color model.

Keywords: convolutional neural networks, classification, web-application

For citation: *Brykin V. V., Braginsky M. Ya., Tarakanova I. O., Tarakanov D. V. Analysis of Plants Health Using Artificial Intelligence Technologies // Proceedings in Cybernetics. 2022. No. 4 (48). P. 6–13. DOI 10.34822/1999-7604-2022-4-6-13.*

ВВЕДЕНИЕ

В настоящее время находят широкое применение и активно разрабатываются автоматические системы дистанционного контроля состояния растений, позволяющие своевременно определить наличие заболеваний растений, водный статус, получать сведения о биомассе в сыром и сухом состояниях как в полевых условиях, так и в тепличных хозяйствах [1–6].

Распознавание состояния растений предполагает решение задачи классификации изображений, и в качестве классификатора предлагается использовать искусственную нейронную сеть (ИНС) как обучающуюся модель, работа которой практически не требует вмешательства пользователя.

При решении задачи оценки состояния растений используется многоклассовая классификация, т. е. изображение будет относиться к одному из 8 выбранных классов наиболее распространенных заболеваний растительных культур:

- bacterial spot (бактериальная пятнистость);
- fungle spot (грибковая пятнистость);
- healthy (здоровое);
- late blight (фитофтороз);
- leaf mold (листовая плесень);
- leaf scorch (ожог листьев);
- powdery mildew (мучнистая роса);
- yellow curl virus (вирус желтой курчавости).

На выходе классификатора формируется вектор вероятностей отношения входного изображения к каждому классу. Класс с наивысшим значением вероятности и будет являться результатом решения задачи классификации.

Представлены этапы создания и развертывания для общего пользования модели ИНС сверточной архитектуры, решающей задачу многоклассовой классификации состояния растений по цветным изображениям. Все операции, связанные с обработкой изображений, выполнены в цветовом пространстве RGB.

МАТЕРИАЛЫ И МЕТОДЫ

Материалом для исследований послужили цифровые фотографии листьев здоровых и больных растений, опубликованные в открытом доступе платформы Kaggle. Объем входных данных составляет примерно из 18 000 изображений 8 разных классов. Количество примеров в каждом классе должно быть одинаковым, чтобы избежать несбалансированности классов. Несбалансированность классов создает проблемы при решении задач классификации, поскольку построенные на таких данных модели имеют «перекос» в сторону мажоритарного класса [7, 8]. Выборка изображений делится на 3 компонента в следующем соотношении данных: обучающая – 70 %, валидационная (проверочная) и тестовая – по 15 %. Обучающая выборка используется непосредственно в процессе обучения и является основной, проверочная часть датасета – для оптимизации гиперпараметров сети и исключения переобучения, тестовая – для оценки качества работы сети.

В качестве среды разработки использовалась облачная платформа Google Colab, связанная с учетной записью разработчика на Google Диске, что позволяет существенно упростить исследования в области машинного обучения [8–10]. В качестве классификатора состояния растений используется модель ИНС сверточной архитектуры (CNN), решающая задачу многоклассовой классификации состояния растений по цветным изображениям.

При решении поставленной задачи была использована предобученная сверточная нейронная сеть MobileNet на наборе данных «ImageNet», что позволяет значительно уменьшить время обучения.

Одна из причин переобучения нейронной сети [8] – слишком большое количество в сети нейронов. В данной работе модель CNN MobileNet имеет очень большое число нейронов ввиду своей большой глубины. Таким образом, если наблюдается расхождение в точности выходных значений между обучающей и проверочной выборкой, то процесс обучения следует остановить и уменьшить число нейронов в модели. Но с уменьшением числа нейронов уменьшается и точность выходных значений, то есть ухудшится показатель качества работы ИНС, поэтому при переобучении необходимо сохранение числа нейронов [11].

Это помогает сделать широко распространенный алгоритм Dropout, часто называемый в русском языке методом исключения, целью которого является снижение специализации

каждого отдельного нейрона и его преобразование для решения более общей задачи [11], а именно: на каждой итерации изменения весовых коэффициентов часть нейронов нужно исключать с заданной вероятностью *p*.

Алгоритм Dropout используется к классификационной части нейронной сети и располагается перед последним, выходным слоем «Dense» (рис. 1).

У алгоритма Dropout используется один аргумент – вероятность *p* того, что нейрон не будет исключен из сети.

```
head_model = base_model.output
head_model = GlobalAveragePooling2D()(head_model)
head_model = Dropout(0.2)(head_model)
outputs = Dense(8, activation="softmax")(head_model)
mobilenet_model = Model(base_model.input, outputs, name='pretrained_mobilenet' )
for layer in mobilenet_model.layers:
    layer.trainable = True
```

Рис. 1. Применение алгоритма Dropout в Keras для модели на основе MobileNet
Примечание: составлено авторами.

Для исследования влияния алгоритма Dropout на качество работы ИНС проведено несколько обучений CNN-модели с разными значениями вероятности *p*. Результаты моделирования приведены в табл. 1.

Таблица 1

Результаты обучения ИНС с использованием алгоритма Dropout

Значение вероятности *p*	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
Точность (обучающая выборка), %	77	80	86	88	87	85	81	81	79
Точность (проверочная выборка), %	67	72	76	85	83	77	74	70	70

Примечание: составлено авторами.

Из табл. 1 видно, что лучшим значением вероятности является $p = 0{,}4$, так как при этом значении были продемонстрированы лучшие показатели в точности на обеих выборках, а также наилучший прогресс в точности – на проверочной выборке.

На рис. 2 отображен ход обучения сети MobileNet методом исключения с выбранной лучшей вероятностью $p = 0{,}4$.

Таким образом, метод вероятностного исключения нейронов Dropout помог избавиться от переобучения в рамках данной конкретной задачи.

На рис. 3 отражены зависимости качества обучения модели нейронной сети MobileNet (точность и ошибка для обучающей и валидационной выборок), улучшенная методами аугментации данных и Dropout.

```
Epoch 1/8
64/64 [==============================] - 671s 10s/step - loss: 1.5996 - accuracy: 0.4538 - val_loss: 11.1643 - val_accuracy: 0.2246 - lr: 0.0010
Epoch 2/8
64/64 [==============================] - 669s 10s/step - loss: 0.9971 - accuracy: 0.6742 - val_loss: 6.3521 - val_accuracy: 0.4600 - lr: 0.0010
Epoch 3/8
64/64 [==============================] - 525s 8s/step - loss: 0.8133 - accuracy: 0.7410 - val_loss: 2.3861 - val_accuracy: 0.6426 - lr: 0.0010
Epoch 4/8
64/64 [==============================] - 440s 7s/step - loss: 0.7048 - accuracy: 0.7696 - val_loss: 1.8320 - val_accuracy: 0.7048 - lr: 0.0010
Epoch 5/8
64/64 [==============================] - 366s 6s/step - loss: 0.6680 - accuracy: 0.7924 - val_loss: 0.7391 - val_accuracy: 0.8143 - lr: 0.0010
Epoch 6/8
64/64 [==============================] - 305s 5s/step - loss: 0.6342 - accuracy: 0.7990 - val_loss: 0.8367 - val_accuracy: 0.7984 - lr: 0.0010
Epoch 7/8
64/64 [==============================] - 252s 4s/step - loss: 0.5914 - accuracy: 0.8042 - val_loss: 0.7685 - val_accuracy: 0.8170 - lr: 0.0010
Epoch 8/8
64/64 [==============================] - 210s 3s/step - loss: 0.5744 - accuracy: 0.8772 - val_loss: 0.5322 - val_accuracy: 0.8506 - lr: 0.0010
```

Рис. 2. Обучение модели на основе MobileNet при $p = 0{,}4$
Примечание: составлено авторами.

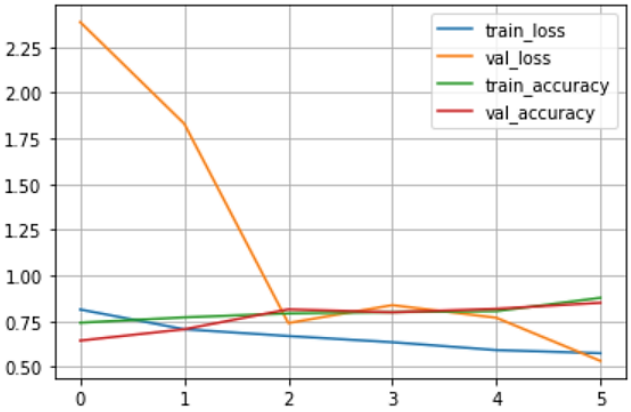

Рис. 3. Графическое представление процесса обучения
Примечание: составлено авторами.

Из рис. 3 видно, что значения точности на обучающей и особенно проверочной выборках повысились и находятся практически на одном уровне 88 и 85 % соответственно (зеленая и красная линии), что подтверждает действенность описанных ранее методов совершенствования модели.

Затем выполнена процедура тестирования обученной модели: случайным образом при помощи библиотек Keras для работы с изображениями выбирается изображение из тестовой выборки (не участвовавшее в обучении) (рис. 4).

```
from PIL import Image
np.random.seed(200)
idx = np.random.randint(30)
test_images_dir = os.path.join('/content/drive/My Drive/PLANT_DIS_REC/datasets/Dataset/test', 'late_blight')
test1 = Image.open(os.path.join(test_images_dir, os.listdir(test_images_dir)[idx]))
```

```
plt.imshow(test1)
plt.title(os.listdir(test_images_dir)[idx])
```

```
Text(0.5, 1.0, 'late_blight.2050.jpg')
```

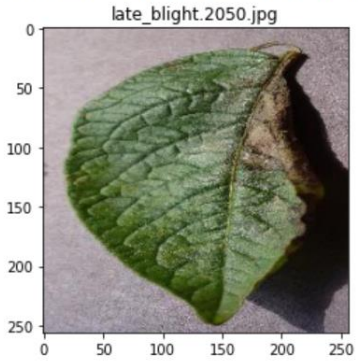

Рис. 4. Генерация случайного изображения из тестовой выборки
Примечание: составлено авторами.

Как видно из названия объекта, это растение с болезнью «фитофтороз» (класс «late_blight»).

Результатом является массив вероятностей отношения тестируемого изображения ко всем 8 классам (рис. 5). При этом видно, что с вероятностью 99,9 % изображение относится к классу с меткой «3». На рис. 6 с метками классов видно, что данной метке соответствует класс «late_blight», следовательно, классификация выполнена успешно.

При создании приложения необходимо выполнить развертывание модели на основе MobileNet для общего пользования и преобразовать модель «mobilenet_model» из формата «H5» в формат «JSON», совместимый с любой JavaScript средой. Файлы формата «JSON» позволяют осуществлять обмен данными между клиентской и серверной частями приложения.

```
test1 = test1.resize((224,224))
test1_scaled = np.expand_dims(np.asarray(test1), axis = 0) / 255
predictions = mobilenet_model.predict(test1_scaled)
print(predictions)
```

```
[[5.6686909e-05 2.8846331e-04 1.5034264e-05 9.9963975e-01 6.2135714e-09
  8.8132729e-10 3.3346109e-10 4.1249504e-10]]
```

Рис. 5. Результат классификации моделью на основе MobileNet
Примечание: составлено авторами.

```
classes_dict = train_set_from_dir.class_indices
classes_dict = { v:k for (k,v) in classes_dict.items() }
classes_dict[np.argmax(predictions)]
```

```
'late_blight'
```

Рис. 6. Успешная классификация
Примечание: составлено авторами.

Для запуска моделей машинного обучения в браузере с использованием JavaScript [12] и решения задачи конвертации используется библиотека «Tensorflow.js», имеющая следующие преимущества:

- наличие большого числа инструментов для визуализации происходящих процессов (графики, анимация и др.);
- наличие прямого доступа к сенсорам устройства (камера, GPS и пр.);
- отсутствие необходимости отправлять обрабатываемые данные на сервер, как следствие – безопасность данных пользователя;
- совместимость с Python-моделями.

Tensorflow.js запускается в браузере, поскольку это обеспечивает высокую производительность за счет использования библиотеки WebGL для выполнения различных математических операций с тензорами.

Далее необходимо выполнить преобразование модели в JSON-формат командой «tensorflowjs-converter». По окончанию выполнения команды на Google Диске создается архив с выходными данными, который нужно распаковать, а данные из него сохранить в ранее созданном каталоге «tensorflowjs-model» (рис. 7).

```
!tensorflowjs_converter --input_format keras models/mobilenet_model.h5 tensorflowjs-model/
```

```
!zip -r tensorflowjs-model.zip tensorflowjs-model
```

```
updating: tensorflowjs-model/ (stored 0%)
updating: tensorflowjs-model/group1-shard1of4.bin (deflated 7%)
updating: tensorflowjs-model/group1-shard2of4.bin (deflated 7%)
updating: tensorflowjs-model/group1-shard3of4.bin (deflated 7%)
updating: tensorflowjs-model/group1-shard4of4.bin (deflated 7%)
updating: tensorflowjs-model/model.json (deflated 94%)
```

Рис. 7. Конвертация Keras-модели в формат JSON и сохранение выходных данных в каталоге «tensorflowjs-model»
Примечание: составлено авторами.

Развертывание модели машинного обучения для общего пользования предполагает создание web-приложения – программы, позволяющей пользователю взаимодействовать с этой моделью через браузер с использованием компьютера, в частности сети и веб-сервера.

Пользовательский интерфейс web-приложения создается путем верстки на языке разметки HTML с применением языка формального описания CSS.

РЕЗУЛЬТАТЫ И ИХ ОБСУЖДЕНИЕ

Тестирование разработанного приложения осуществляли на локальном веб-сервере – эмуляторе хостинга с использованием XAMPP (Apache + MariaDB + PHP + Perl) – бесплатной кроссплатформенной сборки веб-сервера.

Произведена проверка работы функциональных характеристик системы: загрузки в браузер модели машинного обучения, вывода на экран загруженного изображения и результатов классификации. Для этого нужно нажать левой клавишей мыши на большую светлую область в центре экрана (кнопка со значком стрелки загрузки) и выбрать на устройстве изображение растения (рис. 8).

Сообщение «Model Loaded Successfully» оповещает об успешной загрузке конвертированной модели машинного обучения (рис. 8). Надпись на кнопке загрузки поменялась – после загрузки картинки предлагается заменить изображение.

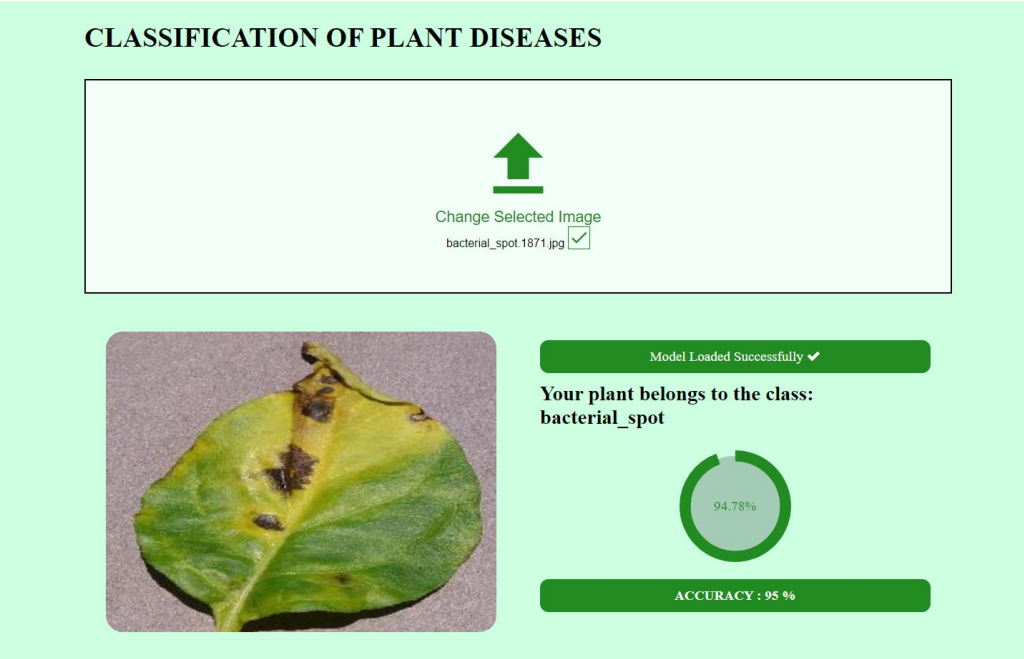

Рис. 8. Результат работы приложения
Примечание: составлено авторами.

Изображение на рис. 8 листа растения, пораженного бактериальной пятнистостью, было получено из тестовой выборки изображений, не участвовавших в обучении загруженной в браузер модели. Как видно из рисунка, картинка была отнесена нейронной сетью к классу «bacterial_spot» (это и есть бактериальная пятнистость) с вероятностью 94,78 %. Этот результат отражен на круглом индикаторе со шкалой, заполненной соответственно значению вероятности, а также на блоке под индикатором в виде округленного до ближайшего целого значения.

При тестировании приложения выявлены следующие особенности работы системы классификации:

- объект должен занимать максимальное пространство на изображении;
- объект должен быть равномерно освещен.

При невыполнении этих требований точность распознавания падает.

Для выполнения первой классификации требуется немного больше времени, чем для последующих (примерно 7–10 секунд против 3–5 на последующих изображениях). Это связано с затратами времени на инициализацию модели.

При попытке загрузки файла, не являющегося изображением (рис. 9), откроется всплывающее окно с предупреждением о некорректном формате выбранного файла.

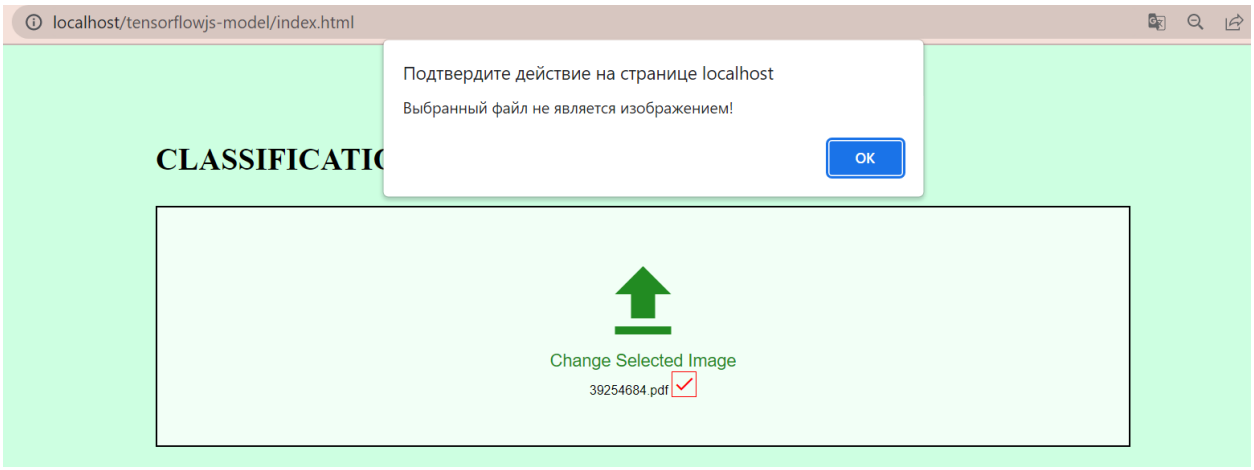

Рис. 9. Предупреждение при загрузке файла некорректного формата
Примечание: составлено авторами.

Таким образом, приложение успешно загрузило модель нейронной сети и корректно классифицировало предоставленное изображение.

ЗАКЛЮЧЕНИЕ

Результатом работы является развернутое для публичного использования и адаптированное под мобильные устройства веб-приложение, функционирующее на основе созданной модели сверточной нейронной сети, решающей задачу классификации состояния растений по анализу фотографий.

Использованы методы улучшения качества работы реализованной модели – аугментация входных данных и алгоритм Dropout, которые позволили повысить точность работы сети на 10 % на обучающей выборке и на 26 % – на проверочной. Кроме того, с их помощью была решена проблема переобучения модели. Для повышения качества классификации необходимо использовать предварительную обработку регистрируемых изображений: повышение контрастности, масштабирование изображения и т. д.

Улучшенная CNN-модель была развернута для общего пользования в виде веб-приложения.

Проведено тестирование приложения с использованием локального веб-сервера XAMPP для проверки корректности выполнения основных функций: запуска приложения, загрузки модели и файлов, генерации результатов классификации и вывода на экран данных, обработки исключений.

Список источников

1. Rahman C. R., Arko P. S., Ali M. E. et al. Identification and Recognition of Rice Diseases and Pests Using Convolutional Neural Networks // Biosystems Engineering. 2020. Vol. 194. P. 112–120. DOI 10.1016/j.biosystemseng.2020.03.020.
2. Янишевская Н. А., Болодурина И. П. Применение технологий компьютерного зрения для разработки модели распознавания поражений культурных растений // Вестн. Юж.-Урал. гос. ун-та.

References

1. Rahman C. R., Arko P. S., Ali M. E. et al. Identification and Recognition of Rice Diseases and Pests Using Convolutional Neural Networks // Biosystems Engineering. 2020. Vol. 194. P. 112–120. DOI 10.1016/j.biosystemseng.2020.03.020.
2. Yanishevskaya N. A., Bolodurina I. P. Application of Computer Vision Technologies for the Development of a Model for the Recognition of Lesions of Cultivated Plants // Bulletin of the South Ural State

Серия «Компьютерные технологии, управление, радиоэлектроника». 2021. Т. 21, № 3. С. 5–13.
3. Тараканов Д. В., Брагинский М. Я., Брыкин В. В. Повышение качества идентификации состояния растений искусственной нейронной сетью // Наука и образование в эпоху перемен: перспективы развития, новые парадигмы : материалы X Всерос. науч.-практич. конф., Ростов-на-Дону, 15 июля 2022 г. Ростов-на-Дону : Манускрипт, 2022. С. 51–60.
4. Брагинский М. Я., Тараканов Д. В. Фенотипирование растений адаптивной системой обработки изображений на базе сверточных нейронных сетей // Вестник кибернетики. 2021. № 2 (42). С. 6–16.
5. Rahman M. A., Islam M. M., Mahdee G. M. S., Kabir M. W. U. Improved Segmentation Approach for Plant Disease Detection // Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), May 03–05, 2019, Bangladesh. P. 1–5. DOI 10.1109/ICASERT.2019.8934895.
6. Чешкова А. Ф. Обзор современных методов обнаружения и идентификации болезней растений на основе анализа гиперспектральных изображений // Вавилов. журн. генетики и селекции. 2022. Т. 26, № 2. С. 202–213.
7. Сэмплинг в условиях несбалансированности классов. URL: https://loginom.ru/blog/imbalance-class (дата обращения: 02.11.2022).
8. Погружение в сверточные нейронные сети: передача обучения (transfer learning). URL: https://habr.com/ru/post/467967/ (дата обращения: 02.11.2022).
9. Keras Applications. URL: https://keras.io/api/applications/ (дата обращения: 02.11.2022).
10. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // ArXiv. 2015. 14 p. URL: https://arxiv.org/pdf/1409.1556.pdf (дата обращения: 02.11.2022).
11. Dropout – метод борьбы с переобучением нейронной сети. URL: https://proproprogs.ru/neural_network/dropout-metod-borby-s-pereobucheniem-neyronnoy-seti (дата обращения: 02.11.2022).
12. Представляем Tensorflow.js: машинное обучение в JavaScript. URL: https://datastart.ru/blog/read/predstavlyaem-tensorflowjs-mashinnoe-obuchenie-v-java script (дата обращения: 02.11.2022).

Информация об авторах

В. В. Брыкин – аспирант.

М. Я. Брагинский – кандидат технических наук, доцент.

И. О. Тараканова – аспирант.

Д. В. Тараканов – кандидат технических наук, доцент.

University. Series “Computer Technologies, Automatic Control, Radioelectronics”. 2021. Vol. 21, No. 3. P. 5–13. (In Russian).
3. Tarakanov D. V., Braginsky M. Ya., Brykin V. V. Improving the Quality of Identifying the State of Plants by an Artificial Neural Network // Science and Education in an Era of Change: Development Prospects, New Paradigms : Proceedings of the X All-Russian Research-to-Practice Conference, Rostov-on-Don, July 15, 2022. Rostov-on-Don : Manuskript, 2022. P. 51–60. (In Russian).
4. Braginsky M. Ya., Tarakanov D. V. Plant Phenotyping by an Adaptive Image Processing System Based on Convolutional Neural Networks // Proceedings in Cybernetics. 2021. No. 2 (42). P. 6–16. (In Russian).
5. Rahman M. A., Islam M. M., Mahdee G. M. S., Kabir M. W. U. Improved Segmentation Approach for Plant Disease Detection // Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), May 03–05, 2019, Bangladesh. P. 1–5. DOI 10.1109/ICASERT.2019.8934895.
6. Cheshkova A. F. A Review of Hyperspectral Image Analysis Techniques for Plant Disease Detection and Identification // Vavilov Journal of Genetics and Breeding. 2022. Vol. 26, No. 2. P. 202–213. (In Russian).
7. Sempling v usloviiakh nesbalansirovannosi klassov. URL: https://loginom.ru/blog/imbalance-class (accessed: 02.11.2022). (In Russian).
8. Pogruzhenie v svertochnye neironnye seti: peredacha obucheniia (transfer learning). URL: https://habr.com/ru/post/467967/ (accessed: 02.11.2022). (In Russian).
9. Keras Applications. URL: https://keras.io/api/applications/ (accessed: 02.11.2022).
10. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // ArXiv. 2015. 14 p. URL: https://arxiv.org/pdf/1409.1556.pdf (accessed: 02.11.2022).
11. Dropout – metod borby s pereobucheniem. URL: https://proproprogs.ru/neural_network/dropout-metod-borby-s-pereobucheniem-neyronnoy-seti (accessed: 02.11.2022). (In Russian).
12. Predstavliaem Tensorflow.js: mashinnoe obuchenie v JavaScript. URL: https://datastart.ru/blog/read/predstavlyaem-tensorflowjs-mashinnoe-obuchenie-v-java script (accessed: 02.11.2022). (In Russian).

Information about the authors

V. V. Brykin – Postgraduate.

M. Ya. Braginsky – Candidate of Sciences (Engineering), Associate Professor.

I. O. Tarakanova – Postgraduate.

D. V. Tarakanov – Candidate of Sciences (Engineering), Associate Professor.