

Классификация изображений с помощью нейронных сетей

Андриенко Иван Сергеевич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье была создана нейронная сеть для классификации изображений. Для создания нейросети использовались библиотеки tensorflow и keras. В результате была создана и протестирована нейросеть, классифицирующая изображения на типы пейзажа.

Ключевые слова: Python, нейронная сеть, машинное обучение, tensorflow, keras, google colab, датасет.

Classification of images using neural networks

Andrienko Ivan Sergeevich

Sholom-Aleichem Priamursky State University

Student

Abstract

In this article, a neural network was created to classify images. Tensorflow and keras libraries were used to create a neural network. As a result, a neural network was created and tested, classifying images into types of landscape.

Keywords: Python, neural network, machine learning, tensorflow, keras, google colab, dataset.

1 Введение

1.1 Актуальность

Сегодня искусственные нейронные сети часто используются для решения разных проблем там, где простые алгоритмы неэффективны или вовсе неприменимы. Например, распознавание спама или текста, обнаружение мошеннических действий по счетам, прогнозирование цены на вино и др. Решения на основе искусственных нейронных сетей с каждым днем становятся все более популярными, а их использование проще.

1.2 Обзор исследований

В своей работе О.Е. Первун, А.И. Хомутов изложили основные концепции сверточных нейронных сетей. Объяснены слои, необходимые для их построения, и подробно изложены способы наилучшей структуризации сети в большинстве задач анализа изображений [1]. Р.В. Семченко, П.А.Еровлев описали процесс создания нейронной сети в приложении google notebook с помощью библиотек keras и tensorflow [2]. В своей работе Е.С.

Локтев, Н.С. Бутенко, В.А. Смирнов, А.А. Андреева рассмотрели концепцию нейронных сетей, а также провели анализ библиотек языка Python для работы с нейронными сетями [3]. В своей работе А.Н. Цаунит изучил основные тенденции развития искусственных нейронных сетей [4]. А.Е. Беженарь, Ю.П. Беженарь теоретически обосновывают явления нейронных сетей, описывают авторскую разработку, распознающую рукописные цифры нейронной сети [5].

1.3 Цель исследования

Цель исследования – создать нейронную сеть для классификации изображений на виды пейзажа.

2 Материалы и методы

Процесс создания нейросети происходит в интерактивной облачной среде для работы с кодом Google Colab. Использовался язык программирования python и библиотеки tensorflow и keras,

3 Результаты и обсуждения

Перед началом работы необходимо найти готовый датасет с изображениями. В данной работе использовался датасет «Intel Image Classification» [6], содержащий в себе около 25 тысяч изображений размером 150x150, распределенных по 6 категориям: здания, лес, ледник, гора, море, улица.

Переходим в Google colab. Он предоставляет мощные процессоры для облачных вычислений, что позволяет не перегружать компьютер и делать все вычисления быстро. Для того чтобы добавить наборы данных в colab, необходимо скачать файл на Kaggle [6] с именем «kaggle.json», который содержит имя пользователя и ключ API. Затем добавляем файл «kaggle.json» в сессионное хранилище colab (рис. 1).

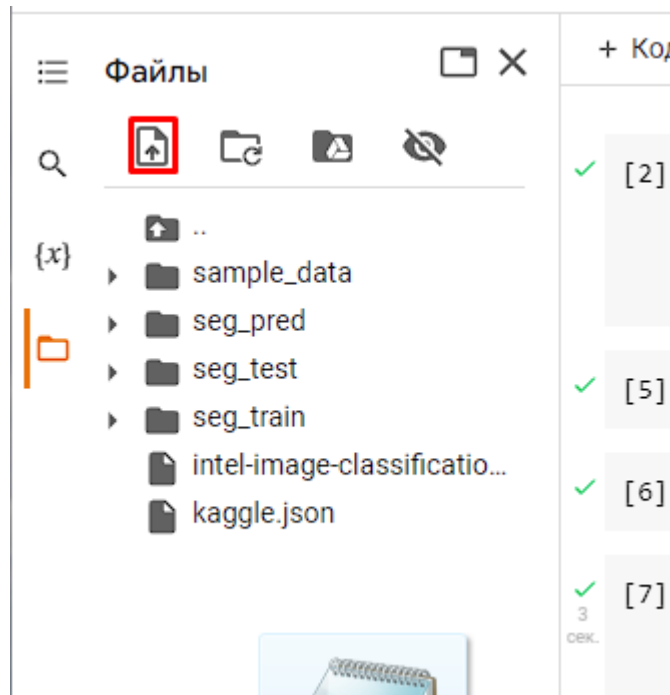


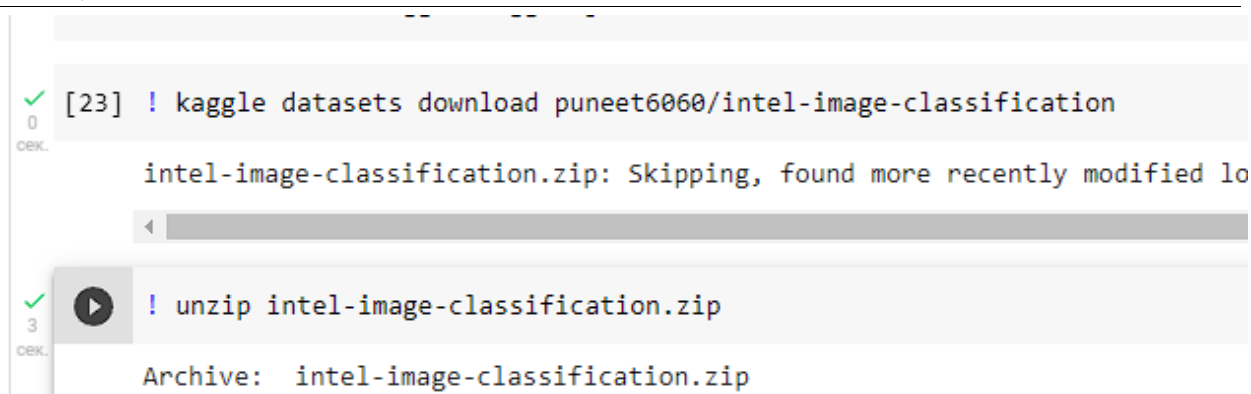
Рисунок 1 – Добавление файла «kaggle.json»

Так как набор данных находится на Kaggle, нужно загрузить его с помощью команд. Готовим блокнот к загрузке данных, вписываем команды в ячейку кода (рис. 2).



Рисунок 2 – Подготовка к загрузке данных

Далее загружаем набор данных с сайта. Для этого необходимо скопировать фрагмент ссылки, содержащий имя пользователя и название датасета, в данном случае это `puneet6060/intel-image-classification`, и вставить в команду загрузки. После загрузки датасета необходимо разархивировать его (рис. 3).



```
[23] ! kaggle datasets download puneet6060/intel-image-classification


intel-image-classification.zip: Skipping, found more recently modified lo

! unzip intel-image-classification.zip

Archive: intel-image-classification.zip
```

Рисунок 3 – Скачивание и разархивирование набора данных

Переходим к написанию кода. Для начала импортируем необходимые библиотеки. Библиотека PIL необходима для работы с изображениями, а numpy для массива. Остальные библиотеки используются для нейросети (рис. 4).



```
import os
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from random import randint

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, AvgPool2D, BatchNormalization, Dropout, Flatten, Dense
from tensorflow.keras.models import Sequential
```

Рисунок 4 – Импорт библиотек

При создании нейросети используются два набора: тестовый и проверочный. Проверочный набор необходим для оценки модели. Если проверять модель на наборе данных, которые использовались для обучения, это приведет к смещенной оценке. Следовательно, чтобы дать объективную оценку модели, используется разделенный подход к оценке алгоритма. С помощью ImageDataGenerator изменим коэффициент масштабирования изображения. Загрузим данные для тестирования в test_loader и данные для обучения в train_loader (рис. 5).

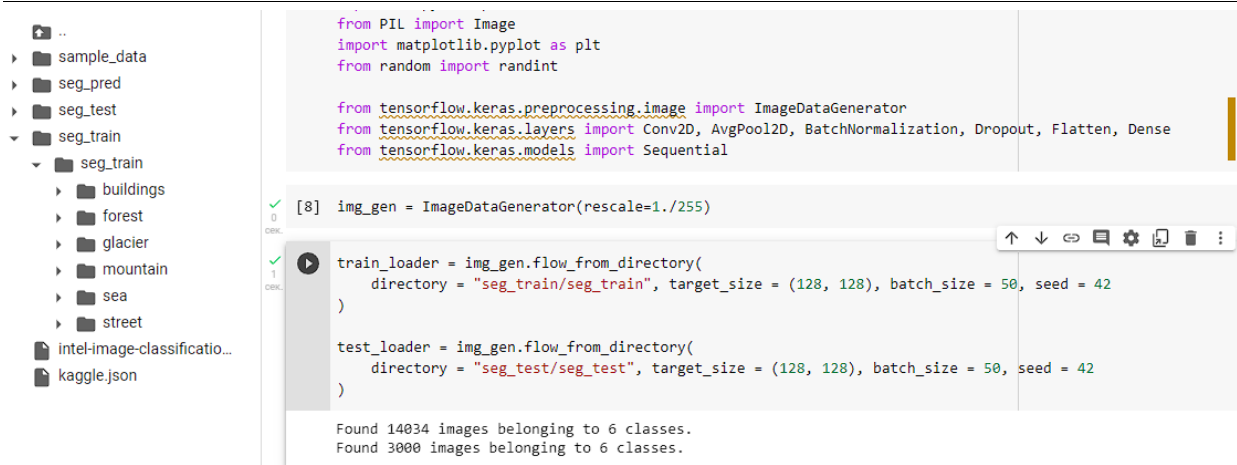


Рисунок 5 – Преобразование и загрузка данных

Далее создаем модель. Модель Sequential представляет собой линейный стек слоев. Создаем модель Sequential, с помощью метода .add() добавляем слои. Так как модель должна знать, какая размерность будет ей передана, первый слой модели Sequential будет получать информацию о размерности входных данных. Это происходит с помощью аргумента input_shape. Выполняем метод compile(). Conv2D создает сверточный слой с 32 нейронами и ядром свертки (3, 3). Слой AvgPool2D это операция максимальной подвыборки для пространственных данных. Его аргумент pool_size изменит масштаб у входного значения (рис. 6). Слой BatchNormalization нормализует свои входные данные. Dropout применяет исключение на вводе. Flatten Выравнивает входные данные, не влияет на размер партии. Последний слой Dense плотно связанный слой нейросети.

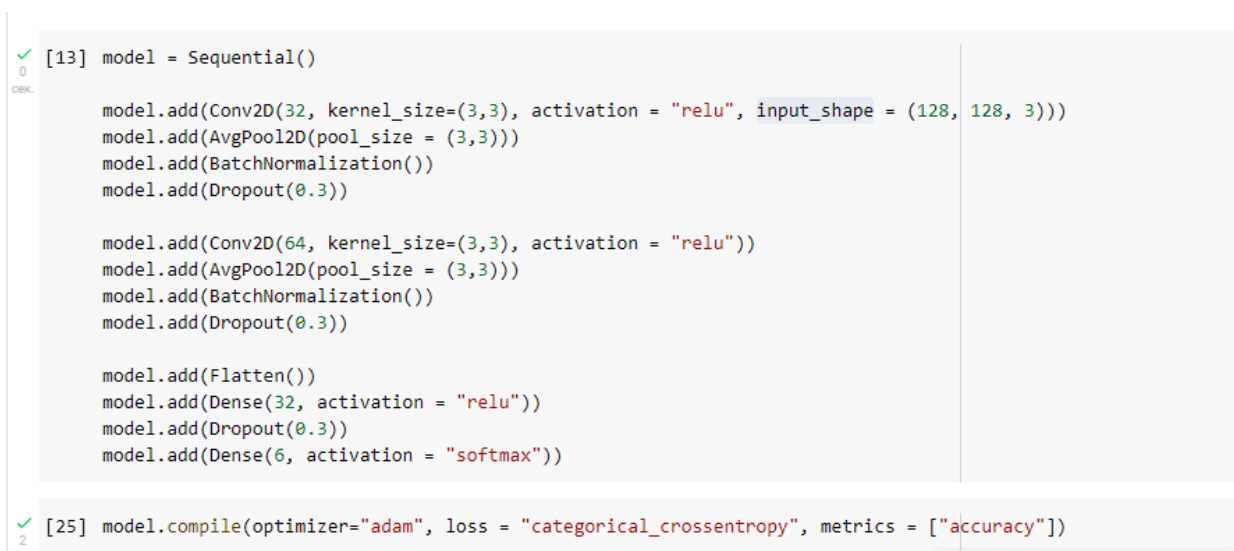


Рисунок 6 – Создание модели

Для визуального представления изобразим данную нейронную сеть, используя онлайн-сервис (рис. 7).

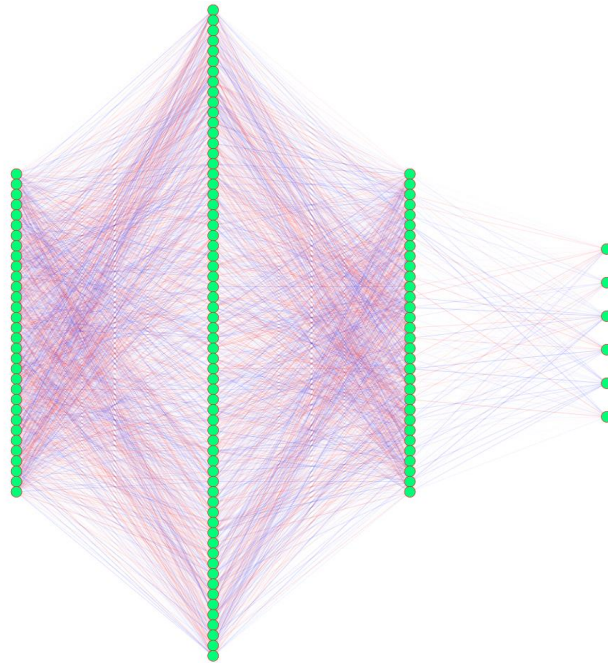


Рисунок 7 – Визуализация нейронной сети

Далее необходимо настроить модель для обучения. Метод `fit` обучает модель для определенного количества итераций на наборе данных. В данном случае их будет 10, так как этого достаточно для хорошей точности модели (рис. 8).

```

train_steps = len(train_loader)
test_steps = len(test_loader)

train_metrics = model.fit_generator(
    generator = train_loader,
    steps_per_epoch = train_steps,
    epochs = 10,
    validation_data = test_loader,
    validation_steps = test_steps
)

```

Epoch 1/10
 <ipython-input-28-cf07089c7235>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which
 train_metrics = model.fit_generator(
 281/281 [=====] - 125s 440ms/step - loss: 1.5363 - accuracy: 0.5126 - val_loss: 1.6896 - val_accuracy: 0.3250
 Epoch 2/10
 281/281 [=====] - 122s 434ms/step - loss: 0.9900 - accuracy: 0.6296 - val_loss: 1.4043 - val_accuracy: 0.5320
 Epoch 3/10
 281/281 [=====] - 123s 437ms/step - loss: 0.8149 - accuracy: 0.6968 - val_loss: 1.2514 - val_accuracy: 0.6430
 Epoch 4/10
 281/281 [=====] - 123s 436ms/step - loss: 0.7317 - accuracy: 0.7340 - val_loss: 1.2261 - val_accuracy: 0.6233
 Epoch 5/10
 281/281 [=====] - 125s 446ms/step - loss: 0.6925 - accuracy: 0.7460 - val_loss: 0.6899 - val_accuracy: 0.7827
 Epoch 6/10
 281/281 [=====] - 125s 446ms/step - loss: 0.6216 - accuracy: 0.7719 - val_loss: 0.6089 - val_accuracy: 0.7807
 Epoch 7/10
 281/281 [=====] - 125s 446ms/step - loss: 0.5880 - accuracy: 0.7817 - val_loss: 0.8374 - val_accuracy: 0.7333
 Epoch 8/10
 281/281 [=====] - 127s 452ms/step - loss: 0.5573 - accuracy: 0.7964 - val_loss: 0.6628 - val_accuracy: 0.7827
 Epoch 9/10
 281/281 [=====] - 126s 449ms/step - loss: 0.5281 - accuracy: 0.8003 - val_loss: 1.0416 - val_accuracy: 0.6950
 Epoch 10/10
 281/281 [=====] - 125s 446ms/step - loss: 0.5107 - accuracy: 0.8092 - val_loss: 0.7278 - val_accuracy: 0.7740

Рисунок 8 – Обучение модели

При обучении можно заметить поле `accuracy`. Это показывает точность модели на определенной итерации. Полученная модель показала точность более 80%, что является хорошим показателем.

После обучения модели тестируем её. С помощью цикла загрузим в модель 10 фото из набора для проверки (рис. 9).

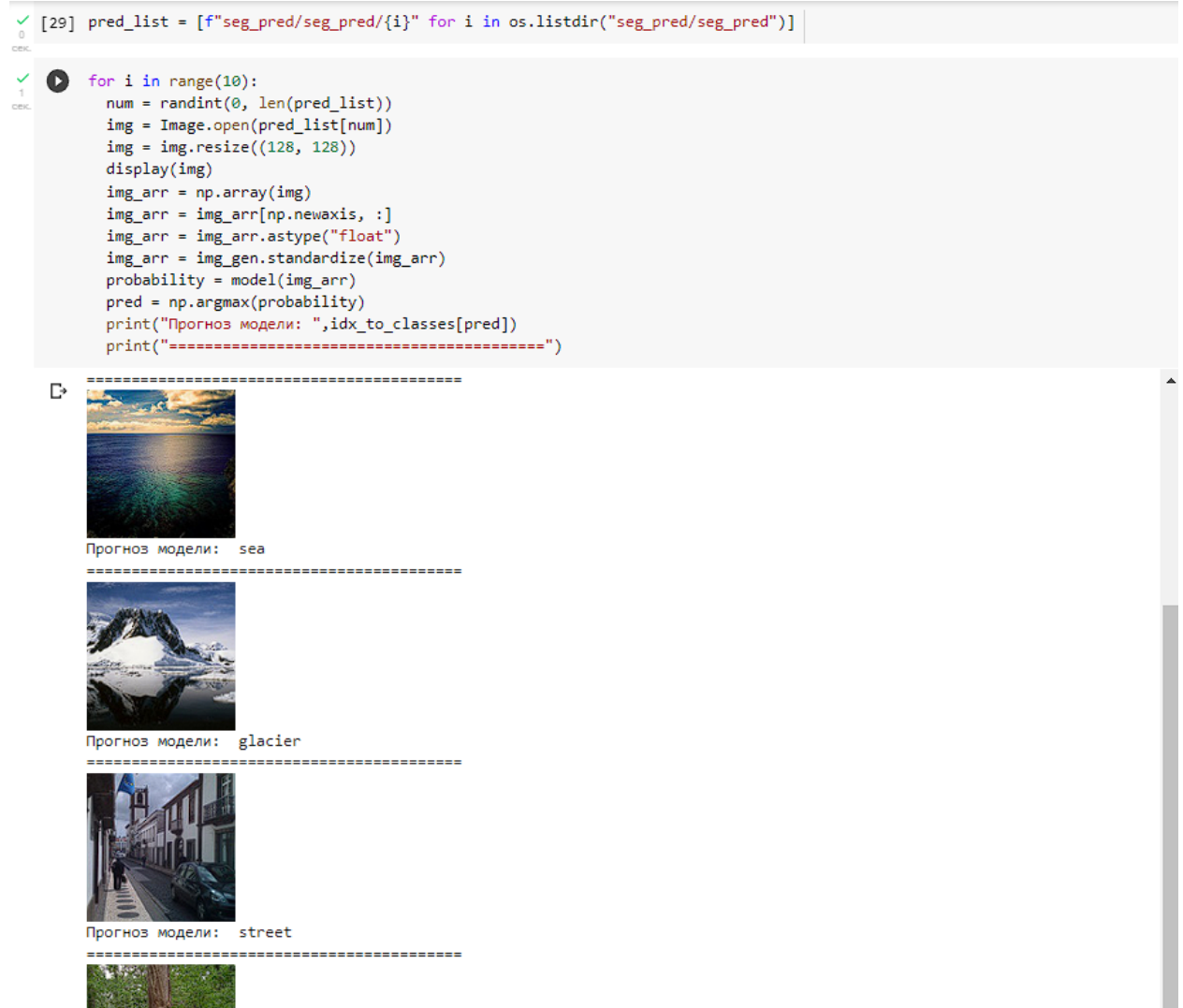


Рисунок 9 – Проверка модели

Создадим новую модель. Данная модель будет содержать в себе меньшее количество нейронов.

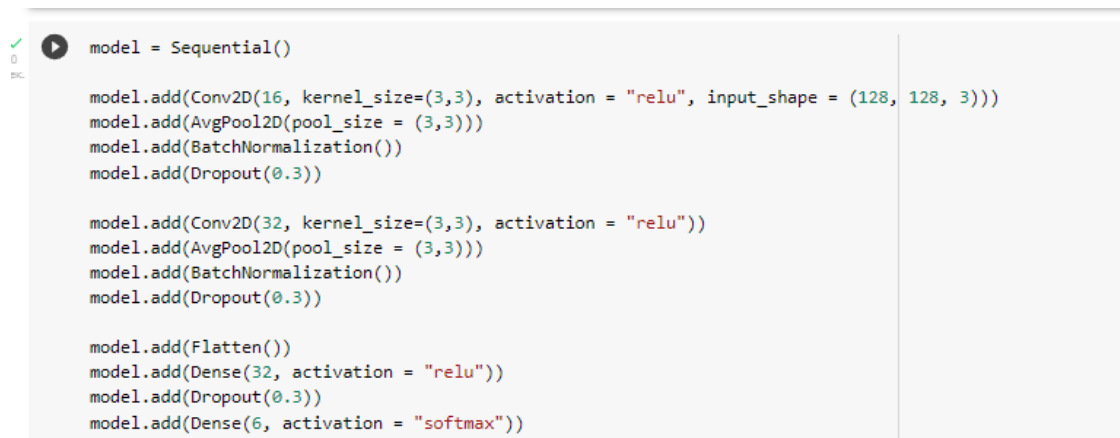


Рисунок 10 – Новая модель

В данном случае новая модель показала большую эффективность, её точность возросла с 0,8092 до 0,8102, а потери уменьшились с 0,5107 до 0,5065 (рис. 11).

```
Epoch 10/10  
281/281 [=====] - 134s 478ms/step - loss: 0.5065 - accuracy: 0.8102 - v
```

Рисунок 11 – Результаты новой модели

Выводы

В данной работе была создана нейронная сеть для классификации изображений. Описаны основные процессы для создания нейронных сетей. Созданная нейронная сеть была протестирована и показала свою точность более 80%.

Библиографический список

1. Первун О.Е., Хомутов А.И. Сверточные нейронные сети и реализация на языке программирования python // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2019. № 2 (24). С. 67-74.
2. Семченко Р.В., Еролев П.А. Программирование нейронных сетей в python с использованием библиотек keras и tensorflow // Постулат. 2020. № 7 (57). С. 6.
3. Локтев Е.С., Бутенко Н.С., Смирнов В.А., Андреева А.А. Анализ библиотек языка python для работы с нейронными сетями// Современные тенденции развития науки и производства. Сборник материалов IX Международной научно-практической конференции. 2018. С. 24-27.
4. Цаунит А. Н. Перспективы развития и применения нейронных сетей // Молодой ученый. 2021. № 23 (365). С. 114-117.
5. Беженарь А.Е., Беженарь Ю.П. Нейронная сеть, распознающая рукописные цифры на языке программирования python. // Молодой ученый. 2020. № 7 (297). С. 5-10
6. Kaggle: Your Home for Data Science URL:<https://www.kaggle.com/> (дата обращения: 14.01.2023)
7. Русскоязычная документация Keras. URL:<https://ru-keras.com/> (дата обращения: 14.01.2023)