УДК 004.8
DOI: 10.15827/0236-235X.133.145-150
Дата подачи статьи: 16.12.20
2021. Т. 34. № 1. С. 145–150

Программные среды для изучения основ нейронных сетей

П.Ю. Богданов [1], *ст. преподаватель, 45bogdanov@gmail.ru*
Е.В. Краева [1], *ассистент, kate.smitt.by@mail.ru*
С.А. Веревкин [1], *студент, vrjovkin@rambler.ru*
Е.Д. Пойманова [1], *к.т.н., доцент, e.d.poymanova@gmail.com*
Т.М. Татарникова [1], *д.т.н., доцент, директор Института информационных систем и геотехнологий, tm-tatarn@yandex.ru*

[1] *Российский государственный гидрометеорологический университет, г. Санкт-Петербург, 192007, Россия*

В статье рассмотрены способы и методы изучения и построения нейронных сетей. Показано, что изучение принципов функционирования нейронных сетей, их применение для решения тех или иных задач возможны только через практику.

Проведен анализ различных программных сред, которые могут быть использованы на лабораторных и практических занятиях по изучению и применению нейронных сетей. Выделен современный облачный сервис Google Colaboratory, рекомендуемый для обучения основам нейронных сетей благодаря наличию в нем предустановки библиотеки Tensorflow и библиотеки для работы на языке Python, бесплатного доступа к графическим процессорам, возможности написания и выполнения программного кода в браузере, а также отсутствию необходимости специальной настройки сервиса.

Рассматриваются примеры проектирования нейронных сетей в Colaboratory, в частности, решение задач распознавания и классификации изображений, прогнозирования. Показано, что для распознавания и классификации изображений может быть использована сверточная нейронная сеть, особенностью которой является получение карты признаков изображения с последующей сверткой. Приведены фрагменты программного кода для этапов подключения необходимых библиотек, загрузки датасетов, нормализации изображений, сборки нейронной сети и ее обучения.

Решение задачи прогнозирования рассмотрено на примере нейронной сети прямого распространения с алгоритмом обратного распространения ошибок в процессе обучения, суть которой в получении на выходном слое ожидаемого значения при подаче на входной слой соответствующих данных. Обратное распространение ошибок заключается в настройке весовых коэффициентов, дающих наибольшую корреляцию между входным набором данных и соответствующим ему результатом.

Ключевые слова: *нейронные сети, программные среды, нейронные сети для начинающих, библиотеки и языки программирования, задача классификации, задача прогнозирования.*

В последние годы методы глубинного обучения – нейронные сети – позволили достичь впечатляющих успехов в таких областях, как компьютерное зрение, обработка естественного языка, обработка аудио [1, 2]. Нейронные сети используются для решения сложных задач, которые требуют аналитических вычислений, подобных выполняемым человеческим мозгом. Бытует мнение, что нет таких задач, с которыми не может справиться нейронная сеть, только было бы достаточно примеров для ее обучения [3].

Распространенными задачами, для решения которых привлекаются нейронные сети, являются: классификация – разделение данных по значимым признакам, прогнозирование – предсказывание следующего шага, распознавание – анализ изображения (объекта) с дальнейшей классификацией [4, 5].

В высшей школе дисциплины, направленные на изучение нейронных сетей, активно внедряются в учебные планы технических направлений обучения, и не только в них. При этом, как правило, изучение нейронных сетей связывают с формированием следующих компетенций:

– получение знаний о возможностях и сферах применения современной теории искусственных нейронных сетей посредством математического моделирования распространенных задач с помощью нейронных сетей различных видов;

– умение пользоваться специализированными пакетами прикладных программ, программными средами и языками программирования для проектирования нейронных сетей;

– формирование навыков решения трудноформализуемых инженерных задач в нейросетевом логическом базисе.

Очевидно, что изучение принципов функционирования нейронных сетей и их применения для решения тех или иных задач возможно только через практику. В статье дан анализ различных программных сред, которые могут быть использованы на лабораторных и практических занятиях по изучению и применению нейронных сетей.

Обзор существующих программных сред для изучения нейронных сетей

Для программирования нейронных сетей в настоящее время наиболее часто используется язык Python благодаря множеству библиотек с набором встроенных математических функций, таких как произведение векторов, транспонирование и тому подобное. Например, используя библиотеку Numpy, можно разработать простую нейронную сеть, решающую задачу прогнозирования. Библиотека Keras применяется при программировании сетей прямого распространения и решения задач распознавания речи. Для нейронных сетей, работающих с изображениями, необходимо подключение другого модуля, например TensorFlow [6].

TensorFlow – библиотека для машинного обучения от компании Google с открытым исходным кодом. Применяется для построения и тренировки нейронной сети, решающей задачи нахождения и классификации образов. Библиотека построена на парадигме программирования потоков данных, позволяющей оптимизировать математические вычисления. Вычисления в TensorFlow выполняются при помощи графа потоков данных, узлы которого отображают операции, а ребра – потоки данных между узлами.

Помимо Python, для написания программного кода, реализующего нейронную сеть, используются языки R, C Sharp, C++, Haskell, Java, Go и Swift. По-прежнему применяются такие пакеты прикладных программ, как MatLab и Deductor. Однако их использование ограничено отсутствием выбора видов и архитектур нейронных сетей.

Также необходима среда для разработки. Для экспериментов с нейронными сетями принято использовать Jupyter Notebook.

Jupyter Notebook – инструмент для интерактивной разработки и представления программных проектов, своего рода блокнот, объединяющий программный код в едином документе. Если нет возможности установить этот инструмент по причине разных версий разрядности операционных систем и прочего, Google предлагает аналог – Colaboratory, не требующий установки на компьютер.

Google Colaboratory – современный облачный сервис, направленный на упрощение исследований в области машинного и глубокого обучения, так как все вычисления в Colaboratory выполняются на удаленных серверах.

В Colaboratory предустановлены TensorFlow и практически все необходимые для работы Python библиотеки. Какой-либо отсутствующий пакет можно установить командой pip или утилитой apt-get.

Помимо того, что в Colaboratory можно писать и выполнять код Python в браузере, не требуется настройка сервиса, имеется бесплатный доступ к графическим процессорам и документам других пользователей. Все это делает облачный сервис Colaboratory доступным решением для обучения студентов основам нейронных сетей.

Примеры решения некоторых задач в Colaboratory

Рассмотрим решение задачи распознавания объекта с последующей классификацией: работая с изображениями, нейронная сеть прежде всего должна определить, что изображено на картинке, а затем назначить метку класса (классифицировать распознанный объект).

Используем сверточную нейронную сеть, которая, как известно, направлена на работу с изображениями. В отличие от персептрона, рассматривающего все изображение сразу, сверточная нейронная сеть сканирует изображение по частям. Сверточные нейронные сети работают на основе фильтров, распознающих определенные характеристики изображения [7]. Фильтр представляет собой коллекцию кернелов – матрицу чисел, называемых весами, которая является результатом обучения и получения карты признаков «кернел-изображение». Фильтр перемещается вдоль изображения и определяет, присутствует ли искомая характеристика (кернел) на сканируемом участке изображения (рис. 1). Для получения ответа выполняется операция свертки, которая является суммой произведений элементов фильтра и матрицы входных сигналов (пикселей изображения).

Далее представлены фрагменты программного кода нейронной сети, реализованной в Colaboratory, которая умеет решать задачу классификации – различает изображения ко-

шек и изображения собак. Первый фрагмент – подключение необходимых библиотек:

```
   import numpy as np
   import tensorflow as tf
   import tensorflow_datasets as tfds
   from tensorflow.keras.preprocessing.image
import load_img, img_to_array
   from tensorflow.keras.layers import
Dense, GlobalAveragePooling2D, Dropout
   import matplotlib.pyplot as plt
   from google.colab import files
```

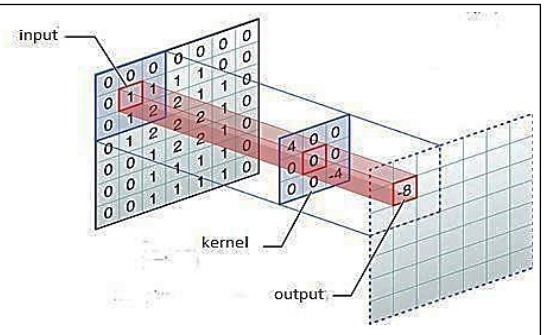

Рис. 1. Операция свертки

Fig. 1. Convolution operation

Поскольку нейронная сеть работает с изображениями, необходимо загрузить датасет для ее обучения. Представим пример загрузки Microsoft датасет с разными изображениями кошек и собак, объем датасета – 25 000 изображений, в описании датасета указано, что 1 700 картинок из 25 000 поврежденные, но TensorFlow умеет их фильтровать:

```
   train, _ = tfds.load('cats_vs_dogs',
split=['train[:100%]'], with_info=True,
as_supervised=True)
```

Поскольку все изображения для обучения нейронной сети должны иметь одинаковое разрешение, в переменной SIZE необходимо это указать, например 224×224. Затем проводится нормализация изображений – числовые значения пикселей переводятся в диапазон [0, 1] путем деления значения каждого пикселя на 255:

```
   import numpy as np
   import tensorflow as tf
   import tensorflow_datasets as tfds
   from tensorflow.keras.preprocessing.image
import load_img, img_to_array
   from tensorflow.keras.layers import
Dense, GlobalAveragePooling2D, Dropout
   import matplotlib.pyplot as plt
   from google.colab import files
```

Далее происходит сборка архитектуры нейронной сети: вызывается метод для последовательного объявления слоев – сверточного, пулинга, полносвязного (base_layers, GlobalAveragyPooling2D() и Dense (1) соответственно). Слой пулинга представляет собой нелинейное уплотнение карты признаков, при котором группа пикселей уплотняется до одного пикселя, проходя нелинейное преобразование. Полносвязный слой выполняет нелинейные преобразования извлеченных признаков и собственно реализует классификацию. Для решения проблемы переобучения используется метод Dropout(0.2), где 0.2 – доля нейронов, случайно выключаемых из процесса обучения.

Нейронная сеть, собранная из готовых функций:

```
   [ ] model = tf.keras.Sequential([
                    base_layers,
                    GlobalAveragePooling2D(),
                    Dropout(0.2),
                    Dense(1)
   ])
   model.compile(optimizer='adam',
loss=tf.keras.losses.BinaryCrossen-
tropy(from_logits=True), metrics=['accura-
cy'])
```

Следующий этап – обучение нейронной сети. В методе обучения указываются датасет и количество эпох обучения. Уже на второй эпохе точность классификации составила порядка 98,9 %:

```
[ ] model = tif(tran_batches, epochs = 2)
    Epoch = 1/2
    1454/1454[==============================]
- 847s 582ms/step - loss: 0.0436 - accuracy:
0.9844
    Epoch = 2/2
    1454/1454[==============================]
- 842s 579ms/step - loss: 0.0418 - accuracy:
0.9847
    <tensorflow.pyton.keras.callbacks.History
at 0x7fea19b85d30>
```

После обучения нейронная сеть будет выдавать числовые результаты, по значениям которых осуществляется классификация: если результат значительно больше 1, то высока уверенность в определении метки класса «собаки», если результат намного меньше 0, то изображению присваивается метка класса «кошки» (рис. 2).

Таким образом, жизненный цикл нейронной сети [8] включает следующие этапы:

– подготовка данных (подключение библиотек, датасетов, нормализация данных);

– создание нейронной сети (выбор архитектуры, размера входного вектора, параметров обучения);

– обучение нейронной сети на датасетах;

– эксплуатация (решение задач, для которых создавалась нейронная сеть).

В следующем примере рассматривается задача прогнозирования с помощью нейронной

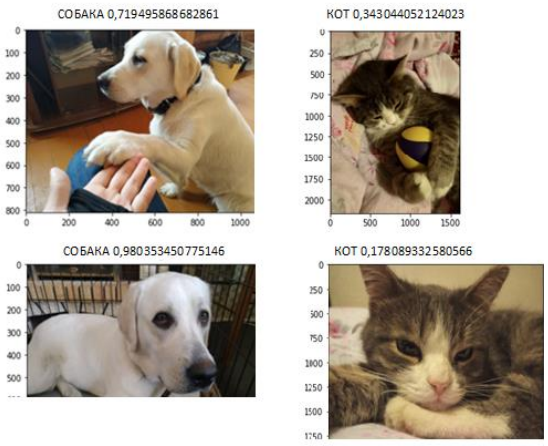

Рис. 2. Визуализация результатов работы нейронной сети

Fig. 2. Visualizing results of the neural network

сети прямого распространения (персептрона), на котором можно изучать работу алгоритма обратного распространения ошибок, реализованного на Python [9, 10].

Нейронная сеть, обучаемая через обратное распространение ошибок, запоминает весовые коэффициенты, соответствующие паре «входные данные–выходные данные»:

Входные данные	Выходные данные
0 0 1	0
1 1 1	1
1 0 1	1
0 1 1	0

Задача предсказания (прогноза), решаемая нейронной сетью, заключается в получении на выходном слое ожидаемого значения при подаче на входной слой соответствующих данных. В рассматриваемой задаче нет строгой математической функции, связывающей строку входных данных с выходными, поэтому и предлагается использовать нейронную сеть. Обученная нейронная сеть должна определить статистическое соответствие между входными и выходными данными – результат коррелирует с крайним левым столбцом входных данных. Обратное распространение ошибок заключается в подсчете подобной статистики при создании нейронной сети.

После подключения библиотеки Numpy для работы с массивами объявляется функция активации, например сигмоидальная (рис. 3). Кроме сигмоидальной функции, в нейронных сетях используются функции активации – линейная, нелинейная, ступенчатая, гиперболический тангенс и другие. Для задачи прогнозирования выбрана сигмоидальная функция.

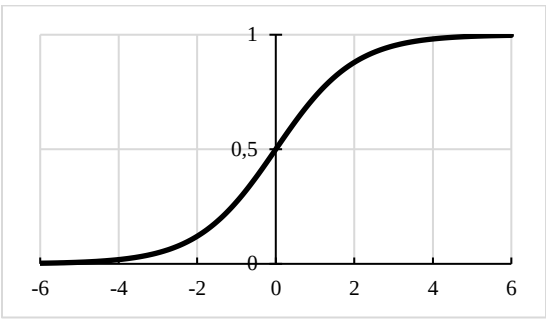

Рис. 3. Сигмоидальная функция активации нейронов

Fig. 3. The sigmoid function of neuronal activation

Далее происходит инициализация массива *X* входных данных в виде numpy-матрицы. Каждая строка – тренировочный пример, столбцы – нейроны входного слоя. Таким образом, получаются 3 нейрона входного слоя и 4 тренировочных примера.

Затем следуют выходные данные – матрица-столбец *Y*. Таким образом, выходной слой содержит 1 нейрон.

Следующий шаг – инициализация случайным образом матрицы весов нейронной сети syn0. После обучения сети в ней будут зафиксированы веса.

Тренировка нейронной сети заключается в многократных итерациях (в приведенном примере 10 000 раз), которые включают операции:

– определение первого слоя с входными данными;

– шаг предсказания (матричное перемножение X и syn0 и передача этих данных на вывод через сигмоиду);

– расчет ошибок.

Программный код нейронной сети с обратным распространением ошибок:

```
import numpy as np
def nonlin(x,deriv=False):
    if(deriv==True):
        return f(x)*(1-f(x))
    return 1/(1+np.exp(-x))

X = np.array([  [0,0,1],
                [0,1,1],
                [1,0,1],
                [1,1,1] ])
  y = np.array([[0,0,1,1]]).T
np.random.seed(1)
syn0 = 2*np.random.random((3,1)) - 1
for iter in xrange(10000):
    l0 = X
    l1 = nonlin(np.dot(l0,syn0))
    l1_error = y - l1
    l1_delta = l1_error * nonlin(l1,True)
# !!!
```

```
        syn0 += np.dot(l0.T,l1_delta) # !!!
    print ("Выходные данные после трени-
ровки:")
    print( l1 )
```

После обучения получены следующие результаты: [0.00966449] [0.00786506] [0.99358898] [0.99211957], из которых можно заключить, что нейронная сеть их верно предсказала. Чем больше итераций, тем качественнее обучение – точность результата растет. И, чем больше объем датасета, тем выше вероятность того, что нейронная сеть при новых входных данных верно предскажет результат.

Заключение

Из приведенных примеров решения задач предложены лабораторные работы для изучения дисциплин, связанных с изучением нейронных сетей: разработка сверточной нейронной сети в облачном сервисе Google Colaboratory, решающей задачу классификации, и разработка нейронной сети прямого распространения на языке Python, решающей задачу предсказания.

В дальнейшем планируется дополнить учебно-методический материал еще одной лабораторной работой по изучению нейронных сетей по распознаванию речи с применением библиотеки Keras.

Цели каждой из предлагаемых работ – ознакомление обучающихся с основами нейронных сетей и базовыми понятиями (нейрон, функция активации, функция потери и прочее) и изучение алгоритмов обучения нейронных сетей с применением их на практике.

Литература

1. Созыкин А.В. Обзор методов обучения глубоких нейронных сетей // Вестн. ЮУрГУ. Сер.: Вычислительная математика и информатика. 2017. Т. 6. № 3. С. 28–59. DOI: 10.14529/CMSE170303.

2. Тархов Д.А. Нейросетевые модели и алгоритмы. М.: Радиотехника, 2014. 349 с.

3. Каллан Р. Нейронные сети: Краткий справочник; [пер. с англ.]. М.: Вильямс, 2017. 288 с.

4. Пальчевский Е.В., Христодуло О.И. Разработка метода самообучения импульсной нейронной сети для защиты от DDoS-атак // Программные продукты и системы. 2019. Т. 32. № 3. С. 419–432. DOI: 10.15827/0236-235X.127.419-432.

5. Гелиг А.Х. Введение в математическую теорию обучаемых распознающих систем и нейронных сетей. СПб: Изд-во СПбГУ, 2014. 223 с.

6. Ertam F., Aydin G. Data classification with deep learning using Tensorflow. Intern. Conf. on Computer Science and Engineering (UBMK), 2017, pp. 755–758. DOI: 10.1109/UBMK.2017.8093521.

7. Zhao B., Lu H., Chen S., Liu J., Wu D. Convolutional neural networks for time series classification. Journal of Systems Engineering and Electronics, 2017, vol. 28, pp. 162–169. DOI: 10.21629/JSEE.2017.01.18.

8. Рашид Т. Создаем нейронную сеть; [пер. с англ.]. СПб: Альфа-книга, 2017. 272 с.

9. Татарникова Т.М. Анализ данных. СПб: Изд-во СПбГЭУ, 2018. 82 с.

10. Татарникова Т.М., Сидоренко А.Ю., Степанов С.Ю., Петров Я.А. Реализация метода для защиты пространственных данных ГИС на основе нейронной сети // Естественные и технические науки. 2020. № 1. С. 134–136.

Software & Systems
DOI: 10.15827/0236-235X.133.145-150

Received 16.12.20
2021, vol. 34, no. 1, pp. 145–150

Software environments for studying the basics of neural networks

P.Yu. Bogdanov [1], *Senior Lecturer, 45bogdanov@gmail.ru*
E.V. Kraeva [1], *Assistant, kate.smitt.by@mail.ru*
S.A. Verevkin [1], *Student, vrjovkin@rambler.ru*
E.D. Poymanova [1], *Ph.D. (Engineering), Associate Professor, e.d.poymanova@gmail.com*
T.M. Tatarnikova [1], *Dr.Sc. (Engineering), Associate Professor, tm-tatarn@yandex.ru*

[1] *Russian State Hydrometeorological University, St. Petersburg, 192007, Russian Federation*

Abstract. The paper describes the ways and methods of studying and constructing neural networks.

It is shown that the study of the functioning guidelines of neural networks, their application for solving certain problems is possible only through practice.

There is the analysis of various software environments that can be used in the laboratory and practical classes for the study and application of neural networks in the paper. Highlighted the modern cloud service Google Colaboratory, which is recommended for teaching the basics of neural networks due to the presence of a pre-installation of the Tensorflow library and a library for working in Python, free access to graphics processors, the ability to write and execute program code in a browser, and no need for special configuration of the service.

Examples of designing neural networks in the Colaboratory are considered. In particular, solving recognition problems and image classification, predictive modeling. The authors show that a convolutional neural network can be used for image recognition and classification, a feature of which is obtaining the image features a map with subsequent convolution. There are chunks of code for the connecting phases the necessary libraries, loading data sets, normalizing images, assembling a neural network, and its training, in the paper.

The solving of the forecasting problem is considered on the example of a feed-forward neural network with an algorithm for backpropagation of errors in the learning process, the essence of which is to obtain the expected value at the output layer when the corresponding data is fed to the input layer. Backpropagation of errors consists of adjusting the weights that give the greatest correlation between the input dataset and its corresponding result.

Keywords: neural networks, software environments, learning how neural networks work for beginners, libraries and programming languages, classification problem, forecasting problem.

References

1. Sozykin A.V. An overview of methods for deep learning in neural networks. *Bull. of SUSU, Ser.: Computational Mathematics and Software Engineering*, 2017, vol. 6, no. 3, pp. 28–59 (in Russ.).
2. Tarkhov D.A. *Neural Network Models and Algorithms*. Moscow, 2017, 349 p. (in Russ.).
3. Callan R. *The Essence of Neural Networks*. 1998, 248 p. (Russ. ed.: Moscow, 2017, 288 p.).
4. Palchevsky E.V., Khristodulo O.I. Development of a self-learning method for a pulsed neural network for protection against DDoS attacks. *Software and Systems*, 2019, vol. 32, no. 3, pp. 419–432 (in Russ.). DOI: 10.15827 / 0236-235X.127.419-432.
5. Gelig A.H. *Introduction to the Mathematical Theory of Learning Recognition Systems and Neural Networks*. St. Petersburg, 2014, 223 p. (in Russ).
6. Ertam F., Aydin G. Data classification with deep learning using Tensorflow. *Intern. Conf. on Computer Science and Engineering (UBMK)*, 2017, pp. 755–758. DOI: 10.1109/UBMK.2017.8093521.
7. Zhao B., Lu H., Chen S., Liu J., Wu D. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 2017, vol. 28, pp. 162–169. DOI: 10.21629/JSEE.2017.01.18.
8. Rashid T. *Make Your Own Neural Network*. 2016, 224 p. (Russ. ed.: St. Petersburg, 2017, 272 p.).
9. Tatarnikova T.M. *Data Analysis*. St. Petersburg, 2018, 82 p. (in Russ.).
10. Tatarnikova T.M., Sidorenko A.Yu., Stepanov S.Yu., Petrov Y.A. Implementation of a method for protecting spatial GIS data based on a neural network. *Natural and Technical Sciences*, 2020, no. 1, pp. 134–136 (in Russ.).

Для цитирования

Богданов П.Ю., Краева Е.В., Веревкин С.А., Пойманова Е.Д., Татарникова Т.М. Программные среды для изучения основ нейронных сетей // Программные продукты и системы. 2021. Т. 34. № 1. С. 145–150. DOI: 10.15827/0236-235X.133.145-150.

For citation

Bogdanov P.Yu., Kraeva E.V., Verevkin S.A., Poymanova E.D., Tatarnikova T.M. Software environments for studying the basics of neural networks. *Software & Systems*, 2021, vol. 34, no. 1, pp. 145–150 (in Russ.). DOI: 10.15827/0236-235X.133.145-150.