МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Системы искусственного интеллекта**

**Отчет по выполнению лабораторной работы №6**

Выполнил

студент группы ИВТАПбд-31

Гаязов Т. Р.

Проверил:

преподаватель

Хайруллин И. Д.

Ульяновск

2024

**Цель работы**

1. С использованием библиотеки Keras загрузить обучающую выборку, создать модель сверточной сети, обучить ее на обучающей выборке, сохранить модель в файл.
2. Написать дополнительно программу, которая загружает модель из файла, и предоставляет возможность загрузить файл или данные любым иным способом, чтобы проверить точность классификатора.

**Ход работы**

**Первый файл**

Сначала импортиурем библиотеку Keras, которая используется для создания и обучения нейронных сетей. Из библиотеки Keras загружается набор данных MNIST, который используется для обучения модели классификации изображений. Импортируется класс Sequential из модуля keras.models. Sequential - это тип модели Keras, который позволяет создавать последовательные нейронные сети. Импортируются различные типы слоев, которые будут использоваться при создании модели. Среди них сверточные слои Conv2D, слои пулинга MaxPooling2D, слой Flatten для преобразования данных в одномерный вектор и полносвязные слои Dense.

1. Создается пустая модель Sequential.
2. Добавляется сверточный слой Conv2D с 32 фильтрами и функцией активации ReLU.
3. Добавляется второй сверточный слой Conv2D с 32 фильтрами и функцией активации ReLU.
4. Добавляется слой пулинга MaxPooling2D для уменьшения размерности данных.
5. Добавляется третий сверточный слой Conv2D с 64 фильтрами и функцией активации ReLU.
6. Добавляется четвертый сверточный слой Conv2D с 64 фильтрами и функцией активации ReLU.
7. Добавляется слой пулинга MaxPooling2D.
8. Добавляется слой Flatten для преобразования данных в одномерный вектор.
9. Добавляется полносвязный слой Dense с 512 нейронами и функцией активации ReLU.
10. Добавляется выходной слой Dense с количеством нейронов, соответствующим количеству классов (10 для MNIST), и функцией активации softmax.

Листинг 1. Создание модели

|  |
| --- |
| model = Sequential([  # relu нелинейность  Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),  MaxPooling2D((2, 2)),  Conv2D(64, (3, 3), activation='relu'),  MaxPooling2D((2, 2)),  Flatten(),  Dense(64, activation='relu'),# Полносвязные слои  Dense(10, activation='softmax') ]) |

* model.compile(...): Компилируется модель с выбранным алгоритмом оптимизации, функцией потерь и метриками для оценки производительности модели.
* model.fit(...): Обучается модель на обучающих данных, используя выбранный оптимизатор, функцию потерь и размер пакета. Эпохи итерируются для тренировки модели.
* model.save("cifar10\_cnn\_model.h5"): Сохраняет обученную модель в файле с именем "cifar10\_cnn\_model.h5", чтобы можно было использовать её позже без необходимости повторного обучения.

**Второй файл**

Таким же образом импортируем библиотеку Keras, и загружаем функцию load\_model, которая позволяет загрузить сохраненную модель из файла.

Листинг 2. Функция предсказания

|  |
| --- |
| def predict\_image(model, image\_path):  # Загрузка и подготовка изображения  img = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)  if img is None:  print(f"Error: Unable to load image at {image\_path}")  return None  img = cv2.resize(img, (28, 28))  img = img.reshape(1, 28, 28, 1).astype('float32') / 255   # Предсказание  prediction = model.predict(img)  return np.argmax(prediction) |

* img = keras.preprocessing.image.load\_img(image\_path, target\_size=(32, 32)): Загружается изображение из указанного пути и изменяется размер до 32x32 пикселей.
* img\_array = keras.preprocessing.image.img\_to\_array(img): Преобразуется объект PIL изображения в массив numpy.
* img\_array = img\_array / 255.0: Нормализуется массив изображения, чтобы значения пикселей находились в диапазоне от 0 до 1.
* img\_array = img\_array.reshape((1,) + img\_array.shape): Добавляется дополнительное измерение для пакета (batch dimension), так как модель ожидает
* входные данные в формате (batch\_size, width, height, channels).
* prediction = model.predict(img\_array): Выполняется предсказание класса для изображения с помощью загруженной модели.
* class\_index = prediction.argmax(): Выбирается индекс класса с наибольшей вероятностью предсказания.
* return class\_index: Возвращается индекс предсказанного класса.

**Тестирование**

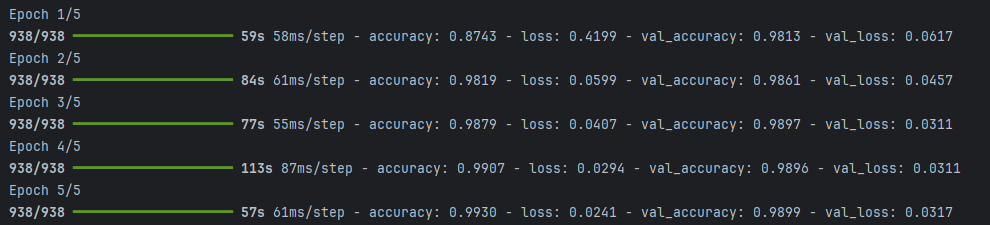


Рис.1. Работа первого файла

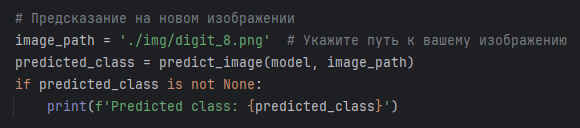


Рис.2.Входные значения для второго файла

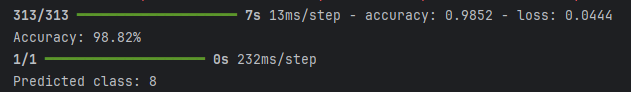


Рис.3.Работа второго файла

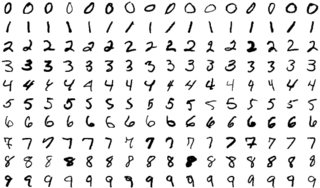


Рис.4. Dataset MNIST для проверки работы кода

**Заключение**

Выполнение данной лабораторной работы позволяет понять процесс обучения нейронных сетей с использованием библиотеки Keras, а также научиться сохранять и загружать обученные модели для их дальнейшего использования.

**Список литературы**

1. <https://habrahabr.ru/post/309508/>
2. <https://habrahabr.ru/company/wunderfund/blog/314242/>
3. <https://habrahabr.ru/company/wunderfund/blog/314872/>
4. <http://datareview.info/article/eto-nuzhno-znat-klyuchevyie-rekomendatsii-po-glubokomu-obucheniyu-chast-1/>

**Приложение A**

|  |
| --- |
| import numpy as np import tensorflow as tf from tensorflow.keras.datasets import mnist from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense from tensorflow.keras.utils import to\_categorical  # Загрузка данных MNIST (x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()  # Подготовка данных x\_train = x\_train.reshape((x\_train.shape[0], 28, 28, 1)).astype('float32') / 255 x\_test = x\_test.reshape((x\_test.shape[0], 28, 28, 1)).astype('float32') / 255 y\_train = to\_categorical(y\_train, 10) y\_test = to\_categorical(y\_test, 10)  model = Sequential([  # relu нелинейность  Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),  MaxPooling2D((2, 2)),  Conv2D(64, (3, 3), activation='relu'),  MaxPooling2D((2, 2)),  Flatten(),  Dense(64, activation='relu'),# Полносвязные слои  Dense(10, activation='softmax') ])  # Компиляция модели model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  model.fit(x\_train, y\_train, epochs=5, batch\_size=64, validation\_data=(x\_test, y\_test))  model.save('mnist\_cnn\_model.h5') |

**Приложение B**

|  |
| --- |
| import cv2 import numpy as np from tensorflow.keras.models import load\_model from tensorflow.keras.datasets import mnist from tensorflow.keras.utils import to\_categorical   # Функция для предсказания класса изображения def predict\_image(model, image\_path):  # Загрузка и подготовка изображения  img = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)  if img is None:  print(f"Error: Unable to load image at {image\_path}")  return None  img = cv2.resize(img, (28, 28))  img = img.reshape(1, 28, 28, 1).astype('float32') / 255   # Предсказание  prediction = model.predict(img)  return np.argmax(prediction)   # Основная программа def main():  # Загрузка данных MNIST  (x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()   # Подготовка данных  x\_test = x\_test.reshape((x\_test.shape[0], 28, 28, 1)).astype('float32') / 255  y\_test = to\_categorical(y\_test, 10)   # Загрузка модели  model = load\_model('mnist\_cnn\_model.h5')   # Оценка модели  loss, accuracy = model.evaluate(x\_test, y\_test)  print(f'Accuracy: {accuracy \* 100:.2f}%')   # Предсказание на новом изображении  image\_path = './img/digit\_8.png' # Укажите путь к вашему изображению  predicted\_class = predict\_image(model, image\_path)  if predicted\_class is not None:  print(f'Predicted class: {predicted\_class}')   if \_\_name\_\_ == "\_\_main\_\_":  main() |