А. Н. Друзик

СОЗДАНИЕ СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ ДЛЯ РАСПОЗНАВАНИЯ РУКОПИСНЫХ ЦИФР

Цель этой статьи — показать создание простой сверточной нейронной сети для распознавания рукописных чисел. Основными инструментами в этой статье будут Tensorflow, Keras и набор данных MNIST.

Введение

Нейронные сети в частности и машинное обучение в целом демонстрируют потрясающие результаты в тех областях науки и техники, в которых от них никто не ожидал этого еще лет 10 назад. Уже на текущий момент модели машинного обучения превзошли человека в задачах классификации, распознавания, предсказания.

I. Архитектура простейшей сверточной нейронной сети

Самый простой вариант создания нейронной сети для распознавания рукописных чисел от 0 до 9, рассматриваемый в этой статье, использует всего два слоя – входной и выходной. Поскольку размер исходного изображения составляет 28×28 пикселей, размер входного слоя составляет 28×28=784 нейрона [1]. Каждый из нейронов связан с одним из пикселей изображения. На выходе получается слой с 10 нейронами, по одному на цифру. (см.рис.1.)

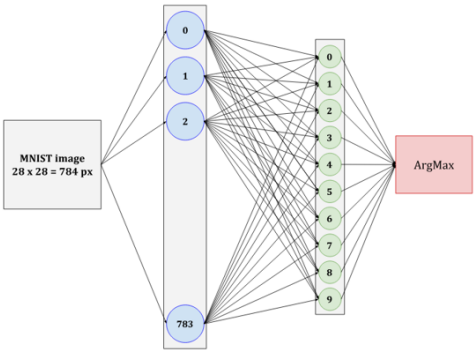

Рис. 1 – Архитектура простейшей сверточной нейронной сети

II. Загрузка набора данных MNIST с выборками и его разделение

Загрузка набора данных MNIST с выборками и разделение его по обучающим и тестовым осуществляется с помощью следующих команд. (см.рис.2.)

```
mnist = tf.keras.datasets.mnist
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Рис. 2 – 20 случайных элементов из набора данных MNIST

Далее показаны примеры 20 случайных элементов из набора данных MNIST. (см.рис.3.)

Рис. 3 – Загрузка набора данных MNIST с выборки

На рис.4 показано преобразование значений от 255 до 1 для последующей работы с библиотекой TensorFlow.

```
X_train = tf.keras.utils.normalize(X_train, axis=1)
X_test = tf.keras.utils.normalize(X_test, axis=1)
```

Рис. 4 – Преобразование значений от 255 до 1

III. Создание сверточной нейронной сети

Далее рассмотрим создание самой сверточной нейронной сети. Последовательный слой – служит для описания модели нейронной сети, имеющей входной, скрытый и выходной слой (см. рис.5.). Слой Flatten используется для преобразования многомерного массива в плоское входное измерение. На первом этапе мы создаем входной слой, равный 128 нейронам [3]. Первый слой имеет 128 «нейронов» и для активации использует специальную математическую функцию tf.nn.relu. Этот уровень помогает сети изучать закономерности в данных. Второй параметр — функция активации. Как правило, для сверточных нейронных сетей используется функция выпрямленной линейной активации или сокращенно ReLU — это кусочно-линейная функция, которая выводит входные данные напрямую, если они положительны, в противном случае — ноль. Она стала функцией активации по умолчанию для многих типов нейронных сетей, поскольку модель, которая ее использует, легче обучать и часто обеспечивает более высокую производительность. На втором этапе мы создаем выход-

ной слой, равный количеству распознанных рукописных цифр, в нашем случае это 10. Этот слой помогает сети делать прогнозы. На этом этапе используется функция активации softmax: это похоже на систему голосования для 10 нейронов второго слоя. Он берет числа, поступающие от этих нейронов, и превращает их в вероятности. Представьте, что у вас есть 10 чисел, отражающих уверенность сети в различных вариантах. Эти числа могут быть такими: [2,0, 3,0, 1,0, 0,1, 2,5, 1,8, 0,5, 1,2, 0,7, 2,2]. Когда мы используем softmax, эти цифры становятся более понятными. Он сжимает их так, что в сумме они дают 1. Эти новые цифры говорят о вероятности. Например, сеть наиболее уверена (38,3 процентов), во втором варианте (3,0 процентов) не очень уверена.

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(units=128, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(units=10, activation=tf.nn.softmax))
```

Рис. 5 – Создание модели нейронной сети

На рис.6 отображен процесс компиляции и обучения нейронной сети.

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=3)
```

Рис. 6 – Процесс компиляции и обучения нашей нейронной сети

На следующем этапе нам нужно оценить производительность модели глубокого обучения на тестовом наборе данных с использованием Keras. Хорошая модель должна иметь низкие потери и высокую точность на всех наборах, а также избегать переобучения или недостаточного подбора (см. рис.7).

```
val_loss, val_acc = model.evaluate(X_test, y_test)
print(val_loss)
print(val_acc)
model.save('handwritten_digits.model')
```

Рис. 7 – Оценка эффективности модели

IV. Тестирование модели

Затем мы читаем файлы как изображения, используя cv2, и извлекаем первый канал (предполагая, что это изображение в оттенках серого), используя [:,:,0]. Он инвертирует значения пикселей изображения с помощью np.invert, так что фон становится черным, а цифры — белыми. Он печатает сообщение «Это число, вероятно, равно », где заменяется индексом наибольшей вероятности с использованием np.argmax. Например, если прогноз равен [0,1, 0,2, 0,3, 0,4, 0, 0, 0, 0, 0, 0], он напечатает «Число, вероятно, равно 3». Затем мы отображаем изображение с помощью plt.imshow с бинарной картой цветов и plt.show (см.рис. 8, рис.9).

```
image_number = 1
while os.path.isfile(f"digits/digit{image_number}.png".format(image_number)):
    try:
        img = cv2.imread(f"digits/digit{image_number}.png".format(image_number))[:,:,0]
        img = np.invert(np.array([img]))
        prediction = model.predict(img)
        print("The number is probably a {}".format(np.argmax(prediction)))
        plt.imshow(img[0], cmap=plt.cm.binary)
        plt.show()
        image_number += 1
    except:
        print("Error reading image! Proceeding with next image...")
        image_number += 1
```

Рис. 8 – Загрузка пользовательских изображений и их прогнозирование

На рис.9 отображен процесс компиляции и обучения нейронной сети.

```
image_number = 1
while os.path.isfile(f"digits/digit{image_number}.png".format(image_number)):
    try:
        img = cv2.imread(f"digits/digit{image_number}.png".format(image_number))[:,:,0]
        img = np.invert(np.array([img]))
        prediction = model.predict(img)
        print("The number is probably a {}".format(np.argmax(prediction)))
        plt.imshow(img[0], cmap=plt.cm.binary)
        plt.show()
        image_number += 1
    except:
        print("Error reading image! Proceeding with next image...")
        image_number += 1
```

Рис. 9 – Отображение случайной картинки с числом 7 и ее предсказание

Итоговый результат с предсказанными картинками чисел, лежащими в папке распознавания, показан на рис.10. В нашем случае количество распознанных файлов равно 5.

```
1/1 [==============================] - 0s 76ms/step
The number is probably a 7
1/1 [==============================] - 0s 17ms/step
The number is probably a 2
1/1 [==============================] - 0s 19ms/step
The number is probably a 9
1/1 [==============================] - 0s 16ms/step
The number is probably a 8
1/1 [==============================] - 0s 17ms/step
The number is probably a 5
```

Рис. 10 – Отображение всех прогнозов изображений в папке с 5 изображениями от 0 до 9

V. Выводы

Рассмотрены шаги для создания простейшей сверточной сети для распознавания рукописных чесел на базе TensorFlow, Keras и набор данных MNIST.

1. Ярмолик, В. Н. Физически неклонируемые функции / В. Н. Яролик, Ю. Г. Вашинко // Информатика. – 2011. – №2. – С. 20-30.

Друзик Алексей Николаевич, магистрант кафедры интеллектуальных информационных технологий БГУИР, alexeydruzik@gmail.com.

Научный руководитель: Гулякина Наталья Анатольевна, кандидат физико-математических наук, доцент, доцент кафедры ИИТ, guliakina@bsuir.by.