1. Распознавание рукописных цифр

1.1 Подготовка к эксперименту

Время выполнения эксперимента: 120 минут

Цель

После завершения данного эксперимента вы сможете:

• • Ознакомьтесь с методами программирования нейронных сетей и конволюционных нейронных сетей.

• • Освоение методов цифрового распознавания с использованием нейронных сетей и конволюционных нейронных сетей

• • Овладение методами сохранения и восстановления нейронных сетей

• • Ознакомьтесь с основами обработки изображений с помощью библиотеки opencv.

Предварительная среда

Для проведения эксперимента необходимо подготовить среду работы с Python версии 3.6 и выше. Экспериментальная среда должна включать следующие элементы:

• • Anaconda 。

• • opencv。

• • Данный набор данных MNIST

1.2 Цель эксперимента

Распознавание рукописных цифр является хорошо изученной задачей классификации в области машинного обучения; для её решения могут использоваться как ручные нейронные сети, так и конволюционные нейронные сети.

Цель данного эксперимента — обучение модели на данных из набора MNIST с целью классификации рукописных цифр с использованием двух различных моделей. В ходе эксперимента также изучаются процессы создания, настройки и сохранения данных для работы с нейронными сетями.

1.3 Классификация и распознавание рукописных цифр

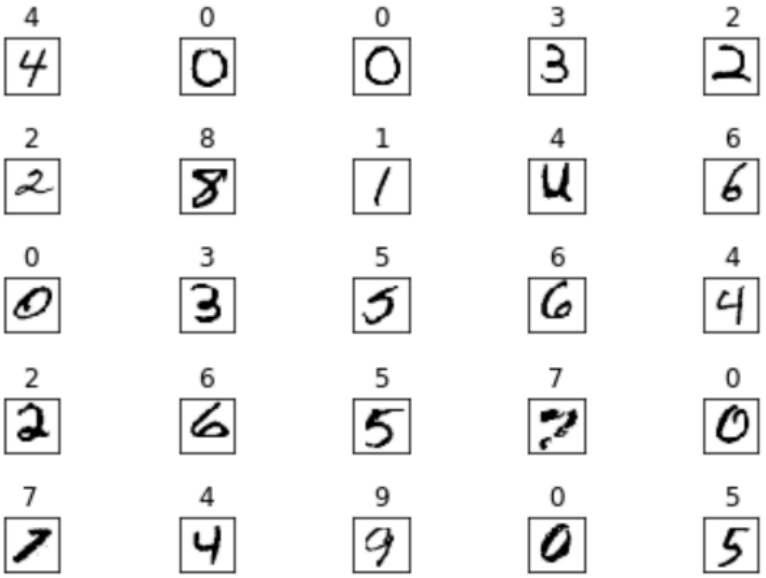
Набор данных MNIST для рукописных цифр является классическим примером задачи многоклассовой классификации в области машинного обучения. Он был разработан

Национальным институтом стандартов и технологий (NIST) в США. Обучающий набор включает 250 рукописных цифр, написанных различными людьми: половина из них — старшеклассники, а другая половина — сотрудники Бюро переписи населения США. Тестовый набор также состоит из рукописных цифр, распределенных в том же соотношении.

Набор данных MNIST доступен по адресу http://yann.lecun.com/exdb/mnist/ и включает в себя четыре файла.

- Изображения для обучения: train-images-idx3-ubyte.gz (размер файла: 9,9 МБ; после распаковки — 47 МБ; содержит 60 000 образцов).
- Метки для обучающей группы: train-labels-idx1-ubyte.gz (размер файла: 29 КБ; после распаковки — 60 КБ; содержит 60 000 меток).
- Изображения для тестового набора: t10k-images-idx3-ubyte.gz (размер файла — 1,6 МБ; после распаковки размер файла составляет 7,8 МБ; в наборе содержится 10 000 образцов).
- Названия тестовых наборов: t10k-labels-idx1-ubyte.gz (размер файла — 5 КБ; после распаковки объем файла увеличивается до 10 КБ; в файле содержится 10 000 меток).

На изображениях представлены рукописные цифры от 0 до 9; разрешение изображений составляет 28×28 пикселей. Изображения и соответствующие метки показаны на рисунке.



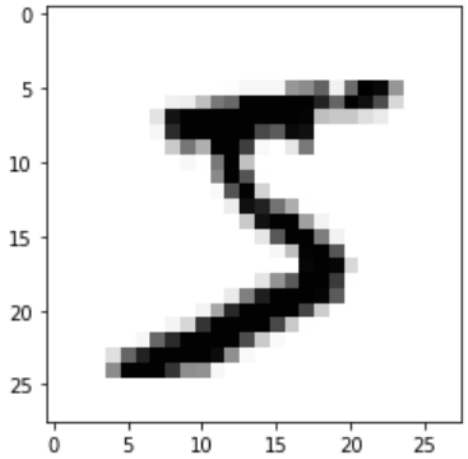
1.4 Импорт наборов данных и визуализация

Компания TensorFlow предоставляет возможность онлайн-загрузки наборов данных с помощью метода read_dataSets(). Однако из-за того, что данные из официальных источников, используемые по умолчанию, находятся за рубежом, процесс их загрузки часто приводит к прерыванию выполнения программы (тайм-ауту). Поэтому в данном эксперименте данные были заранее загружены на локальный компьютер.

```
def load_mnist(path):
# Директория, в которой находится файл mnist.py. Обратите внимание на наличие слеша (slash).
f = np.load(path)
x_train, y_train = f['x_train'], f['y_train']
x_test, y_test = f['x_test'], f['y_test']
f.close()
return (x_train, y_train), (x_test, y_test)

def mnist_parse_file(fname):
fopen = gzip.open if os.path.splitext(fname)[1] == '.gz' else open
with fopen(fname, 'rb') as fd:
    return mnist.parse_idx(fd)

train_images = mnist_parse_file(".\\mnist_dataset\\train-images-idx3-ubyte.gz")
train_labels = mnist_parse_file(".\\mnist_dataset\\train-labels-idx1-ubyte.gz")
test_images = mnist_parse_file(".\\mnist_dataset\\t10k-images-idx3-ubyte.gz")
test_labels = mnist_parse_file(".\\mnist_dataset\\t10k-labels-idx1-ubyte.gz")
```

Изображения можно отображать с помощью библиотеки matplotlib.pyplot или opencv.

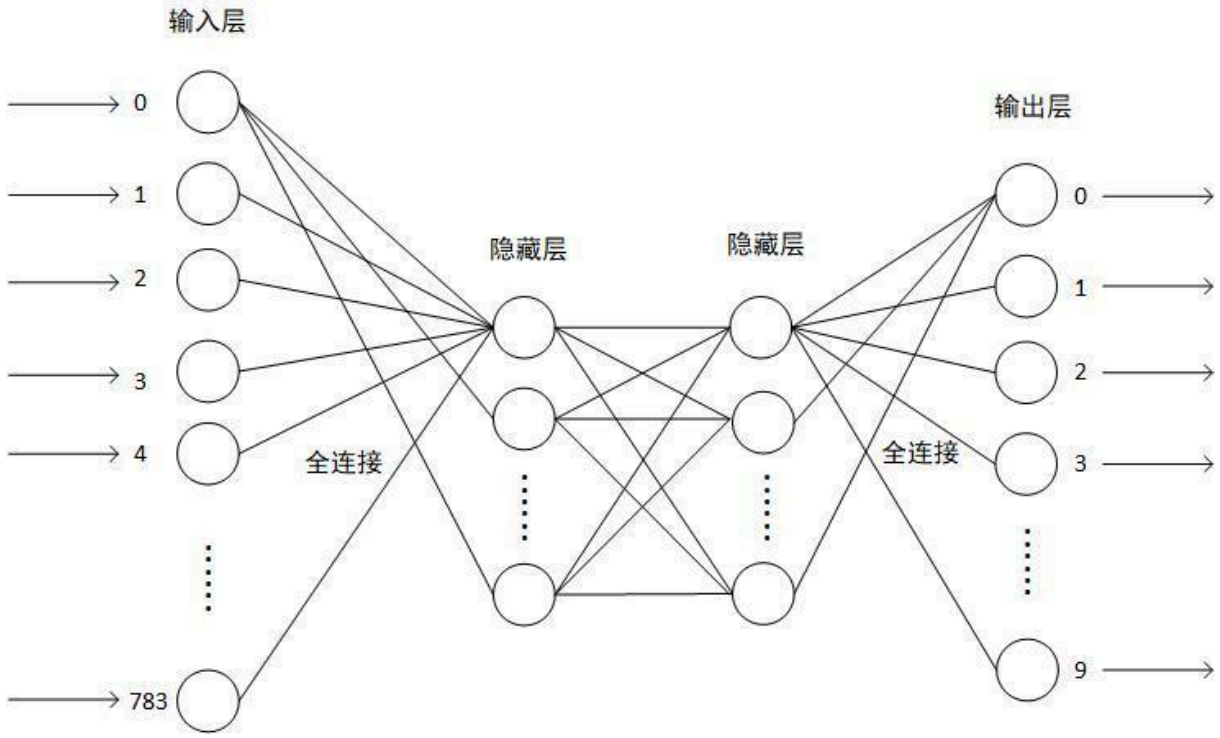
```
plt.imshow(train_images[0],cmap='Greys')
```



Для одновременного отображения нескольких изображений и соответствующих меток можно использовать функцию подграфов (subplots) библиотеки matplotlib; каждый подграф отображает одно изображение. Полный код см. в файле mnist_all_models.ipynb (функция showpics).

1.5 Модель 1: Нейронная сеть

Сети нейронных сетей могут использоваться для распознавания рукописных цифр. Каждый изображение цифры имеет размер 28×28 пикселей; таким образом, данные представляют собой информацию в 784 измерения. В выходном слое используются 10 категорий (от 0 до 9).



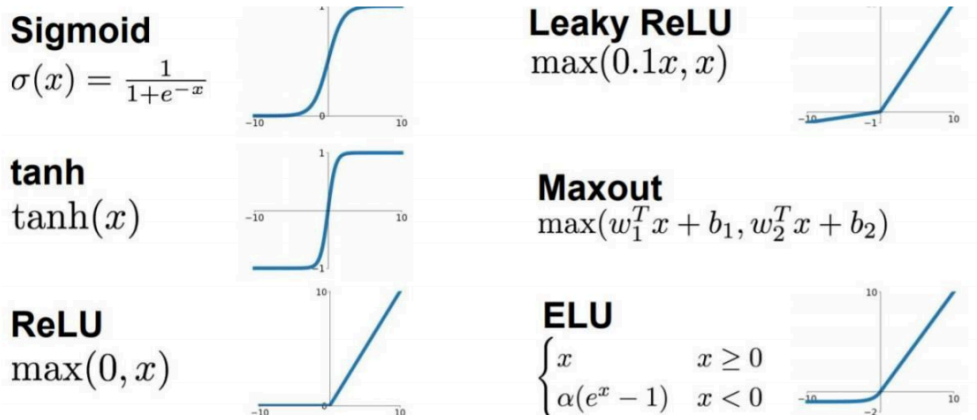
Инициализация модели последовательности

```
model = models.Sequential()

# 1 скрытый слой, 128 нейронов
model.add(layers.Dense(128,input_shape=[784]))
Второй скрытый слой содержит 40 нейронов; для активации используется функция relu.
model.add(layers.Dense(40, activation='relu'))
# Выходной слой, соответствующий диапазону от 0 до 9
model.add(layers.Dense(10, activation='softmax'))
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()

x_train = train_images.reshape(-1, 784)
x_test = test_images.reshape(-1, 784)
x_train, x_test = x_train / 255.0, x_test / 255.0

model.fit(x_train,train_labels,epochs=20,validation_data=(x_test,test_labels))
model.save("mnist_ann.keras")
```

```
Model: "sequential_26"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_36 (Dense)             (None, 128)               100480
_________________________________________________________________
dense_37 (Dense)             (None, 40)                5160
_________________________________________________________________
dense_38 (Dense)             (None, 10)                410
=================================================================
Total params: 106,050
Trainable params: 106,050
Non-trainable params: 0
```

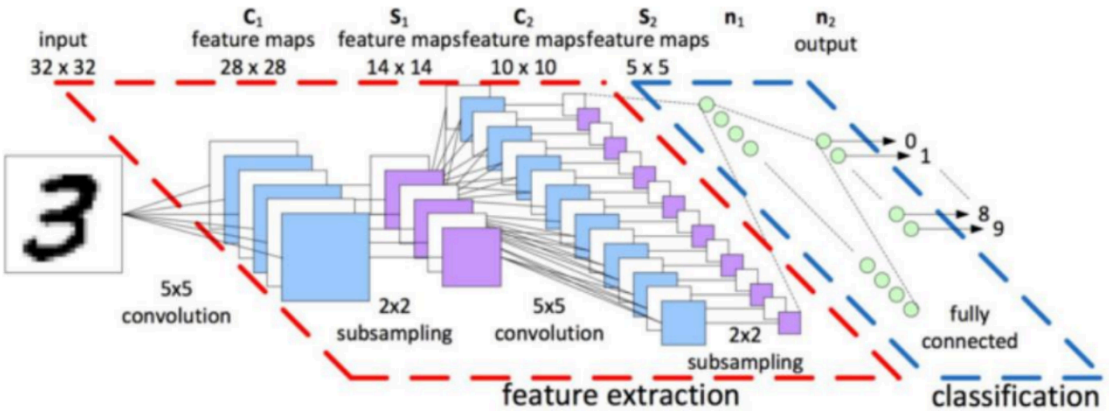
```
Epoch 17/20
1875/1875 [==============================] - 2s 809us/step - loss: 0.0433 - accuracy: 0.9857 - val_loss: 0.1399 - val_accuracy: 0.9663
Epoch 18/20
1875/1875 [==============================] - 1s 767us/step - loss: 0.0342 - accuracy: 0.9885 - val_loss: 0.1846 - val_accuracy: 0.9617
Epoch 19/20
1875/1875 [==============================] - 1s 755us/step - loss: 0.0384 - accuracy: 0.9877 - val_loss: 0.1432 - val_accuracy: 0.9688
Epoch 20/20
1875/1875 [==============================] - 1s 768us/step - loss: 0.0330 - accuracy: 0.9889 - val_loss: 0.1382 - val_accuracy: 0.9720
INFO:tensorflow:Assets written to: mnist_ann.model\assets
```

В нейронных сетях изменяются такие параметры, как структура сети (количество слоёв и количество узлов на каждом слое), функция активации и количество итераций. Как правило, увеличение числа скрытых слоёв и узлов повышает точность распознавания. На данном наборе данных точность на тестовом наборе достигает 97,2%. Для реализации нелинейного моделирования нейронных сетей и решения задач, не поддающихся линейному разделению, обычно используются функции активации, вводящие нелинейные факторы. Все функции активации являются нелинейными, наиболее распространёнными из них являются Sigmoid, tanh, ReLU и другие.



1.6 Модель 2: Конволюционная нейронная сеть

Для обработки изображений существует специальная структура нейронных сетей — конволюционные нейронные сети. Это особая нейронная сеть, предназначенная для обработки изображений. В отличие от традиционных нейронных сетей, в конволюционные нейронные сети включены слои «конволюции» и «пульсации», которые имитируют способ обработки зрительной информации человеческим мозгом, что значительно повышает точность распознавания изображений. С помощью Keras можно легко построить конволюционную нейронную сеть:



Конволюционные нейронные сети в основном включают в себя слои конволюции, слои пуллинга и слои dropout. Следует обратиться к статье на сайте https://www.cnblogs.com/lxl616/p/11247954.html для получения подробного объяснения функций и методов работы этих слоев. Способ создания конволюционной нейронной сети следующий:

```
Инициализация модели последовательности
model = models.Sequential()

# Добавьте первый слой конволюции с использованием 32 конволюционных ядер размером 3×3; для активации выберите функцию relu. Размер входного слоя (input_shape) должен быть равен 28×28 (размер каждого изображения из набора данных MNIST). Значение 1 указывает на количество цветов в изображении; поскольку MNIST представляет собой серый цветовой набор, выберите значение 1.
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28,1)))
# Добавить слой максимального пуллинга с окном пуллинга размером 2×2
model.add(layers.MaxPooling2D((2, 2)))
Добавьте слой конволюции с 16 конволюционными ядрами размером 2×2; все последующие параметры входных данных (input_shape) будут автоматически определяться.
model.add(layers.Conv2D(16,(2,2), activation='relu'))
# Добавить слой максимального пуллинга с окном пуллинга размером 2×2
model.add(layers.MaxPooling2D((2,2)))
Добавьте ещё один слой конволюции с 8 конволюционными ядрами размером 3×3.
model.add(layers.Conv2D(8,(3,3), activation='relu'))
# Разложение матрицы после применения операции конволюции; это первый слой полностью связанных нейронов.
model.add(layers.Flatten())
# Добавление ещё одного слоя с полным соединением; 64 нейрона; функция активации типа ReLU.
model.add(layers.Dense(64, activation='relu'))
# Выходный слой, соответствующий диапазону от 0 до 9
model.add(layers.Dense(10))

model.summary()
```

```
Model: "sequential_27"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 12, 12, 16)        2064
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 6, 6, 16)          0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 4, 4, 8)           1160
_________________________________________________________________
flatten_2 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_39 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_40 (Dense)             (None, 10)                650
=================================================================
Total params: 12,450
Trainable params: 12,450
Non-trainable params: 0
```

```
Epoch 8/10
1875/1875 [==============================] - 14s 7ms/step - loss: 0.0365 - accuracy: 0.9888 - val_loss: 0.0449 - val_accuracy: 0.9862
Epoch 9/10
1875/1875 [==============================] - 14s 7ms/step - loss: 0.0354 - accuracy: 0.9888 - val_loss: 0.0555 - val_accuracy: 0.9851
Epoch 10/10
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0316 - accuracy: 0.9893 - val_loss: 0.0690 - val_accuracy: 0.9807
```

Точность на тестовом наборе данных составляет 98,07%.

1.7 Использование модели нейронной сети

Процесс обучения нейронной сети представляет собой поиск оптимальных параметров сети; после завершения обучения её можно использовать для классификации изображений.

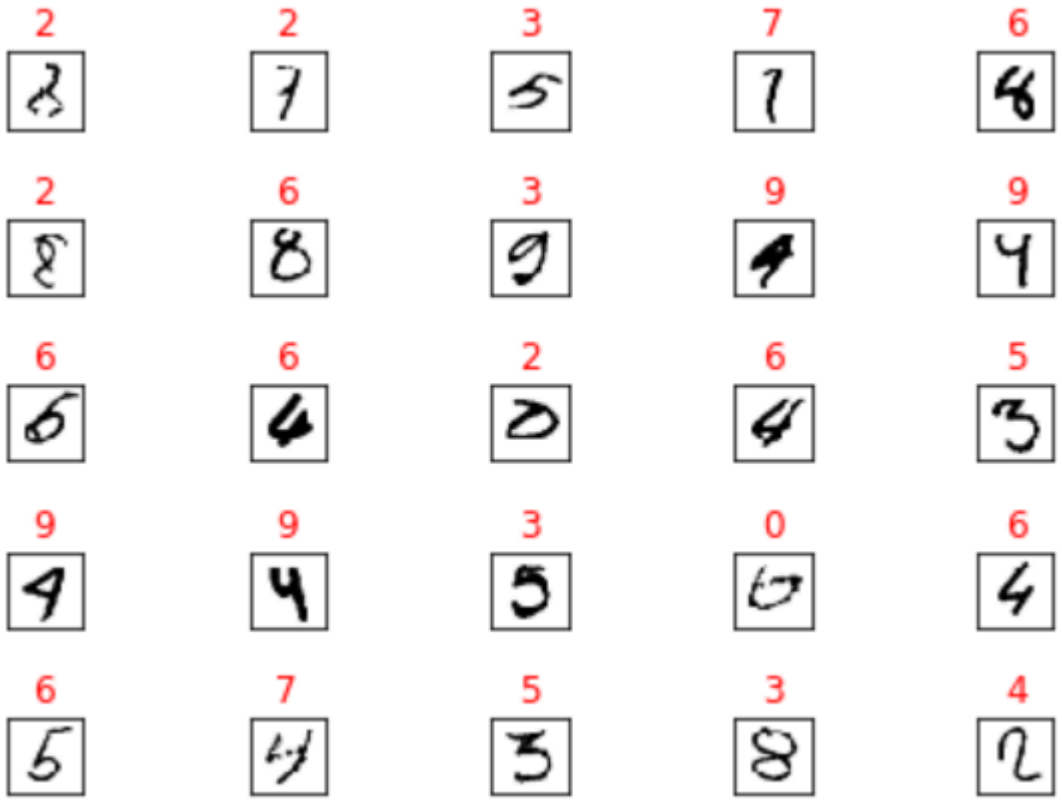
```
pred_labels = model.predict(test_images.reshape(10000, 28, 28, 1))
pred_labels = np.argmax(pred_labels, axis=1)
```

Метод `predict` в модели возвращает вероятности для 10 категорий; числом, соответствующим наибольшей вероятности, является предсказанный результат. Сравнивая предсказанные значения с этикетами, предоставленными на тестовом наборе данных, можно убедиться, что точность модели составляет примерно 98,07%.

```
wrong_set=np.nonzero(pred_labels-test_labels)
print(1-len(wrong_set[0])/len(pred_labels))
print(wrong_set[0])
```

С помощью визуализирующих функций можно отобразить изображения и метки, в которых произошли ошибки при распознавании.

```
showpics(test_images, pred_labels,test_labels, list(wrong_set[0]))
```



Обучение модели обычно занимает довольно много времени. После завершения процесса обучения модель может быть сохранена в виде файла, и при необходимости её структура и параметры можно использовать непосредственно.

```
# Чтение модели из файловой системы
model = tf.keras.models.load_model("mnist_cnn.keras")
model.summary()
```

Помимо существующих наборов данных, можно также вручную записывать цифры и использовать модель для их предсказания. Для этого откройте встроенную программу рисования Windows, укажите размер полотна на 56×56 (изображение размером 28×28 слишком маленькое для удобства ручного ввода). Затем используйте мышь или рисовальный карандаш, чтобы записать цифры на полотне, после чего сохраните результат в формате jpg.

С помощью библиотеки opencv можно считывать изображения и преобразовывать их в формат 28×28 пикселей.

```
import cv2

img=cv2.imread("./test_images/8.jpg",cv2.IMREAD_GRAYSCALE)

resize=cv2.resize(img, (28,28), interpolation=cv2.INTER_CUBIC)
```

Затем можно использовать обученный модель для прогнозирования.

```
possibles=model.predict(resize.reshape(1, 28, 28, 1))
label = np.argmax(possibles)
print(possibles,label)
```

1.8 Задание для эксперимента

Набор данных EMNIST — это набор данных с рукописными буквами и цифрами, имеющий тот же формат, что и MNIST. https://www.nist.gov/itl/products-and-services/emnist-dataset

Используйте знания, полученные в ходе экспериментов, чтобы распознавать рукописные буквы и определить точность работы модели.