Лабораторная работа №8. Сверточные нейронные сети для классификации визуальных образов.

В рамках данной лабораторной работы предполагается построение многоклассового классификатора с помощью сверточной нейронной сети, определяющий рукописные цифры. Набор данных MNIST можно импортировать из модуля Keras.

Задания:

Набор данных MNIST (сокращение от «Modified National Institute of Standards and Technology») — объёмная база данных образцов рукописного написания цифр. База данных является стандартом, предложенным Национальным институтом стандартов и технологий США с целью калибрации и сопоставления методов распознавания изображений с помощью машинного обучения в первую очередь на основе нейронных сетей. Данные состоят из заранее подготовленных примеров изображений, на основе которых проводится обучение и тестирование систем. База данных была создана после переработки оригинального набора чёрно-белых образцов размером 20x20 пикселей NIST. Создатели базы данных NIST, в свою очередь, использовали набор образцов из Бюро переписи населения США, к которому были добавлены ещё тестовые образцы, написанные студентами американских университетов. Образцы из набора NIST были нормализированы, прошли сглаживание и приведены к серому полутоновому изображению размером 28x28 пикселей. Набор данных MNIST содержит 60000 изображений для обучения и 10000 изображений для тестирования. Половина образцов для обучения и тестирования были взяты из набора NIST для обучения, а другая половина — из набора NIST для тестирования.

1. Установите все необходимые библиотеки и импортируйте их в проект.

```
import numpy as np
import random
import matplotlib.pyplot as plt

from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import np_utils

from sklearn.metrics import confusion_matrix ,
classification_report, accuracy_score
```

2. Выполните считывание и просмотр набора данных

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
```

```
print(y_test.shape)
```

Выполните вывод части обучающих изображений и соответствующих им меток.

```
image_number = random.randint(0, len(X_train) - 20)
plt.figure(figsize=(10,10))
for i in range(image_number, image_number + 20):
    plt.subplot(5,5,i-image_number + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(X_train[i], cmap=plt.cm.binary)
    plt.xlabel(y_train[i])
```

3. Подготовьте данные

```
batch_size, img_rows, img_cols = 64, 28, 28 # Размер
изображения
X_train = X_train.reshape(X_train.shape[0], img_rows, img_cols,
1)
X_test = X_test.reshape(X_test.shape[0], img_rows, img_cols, 1)
input_shape = (img_rows, img_cols, 1)
```

Выполните нормализацию данных

```
X_train = X_train.astype("float32")
X_test = X_test.astype("float32")
X_train /= 255
X_test /= 255
```

Преобразуйте метки в категории

```
Y_train = np_utils.to_categorical(y_train, 10)
Y_test = np_utils.to_categorical(y_test, 10)
```

4. Задайте модель полносвязной нейронной сети с использованием фреймворка Keras

Создайте последовательную модель

```
model = Sequential()
```

Добавьте уровни сети

```
model.add(Conv2D(75, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(100, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

Выполните просмотр модели

```
model.summary()
```

5. Компиляция модели

Настройте процесс обучения. Задайте оптимизатор и функцию потерь, которые должны использоваться моделью, а также метрику для мониторинга во время обучения

```
model.compile(loss="categorical_crossentropy",
optimizer="adam", metrics=["accuracy"])
```

6. Обучите модель

Выполните обучение модели на 10 эпохах, 20 процентов обучающей выборки будет использовано для оценки модели, во время обучения, размер пакета 200. Можете изменить количество эпох обучения на более высокое значение, однако это займет значительно больше времени.

```
history = model.fit(X_train,
                    Y_train,
                    batch_size=200,
                    epochs=10,
                    validation_split=0.2, verbose=2)
```

Сохраните модели

```
model.save('Model_MNIST_CNN.hdf5')
```

7. Оцените модель

После постройте предсказания и визуализируйте их. Постройте график для визуализации изменения оценки модели на разных эпохах. Когда наступает переобучение?

```
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
```

```
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Оцените качество обучения сети на тестовых данных

```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Test score: %f" % scores[0])
print("Test accuracy: %f" % scores[1])
```