Лабораторная работа №4

Целью данной лабораторной работы является разработка нейронной сети для решения задачи классификации или регрессии в зависимости от набора данных в рамках варианта. Лабораторная работа предполагает разработку на языке программирования Python с использованием библиотеки Keras.

Варианты заданий:

1) Распознавание цифр на изображении
   (MNIST digits classification dataset)
2) Распознавание 10 предметов на уменьшенных изображениях
   (CIFAR10 small images classification dataset)
3) Определение эмоционального окраса рецензии фильма
   (IMDB movie review sentiment classification dataset)
4) Классификация ленты новостей
   (Reuters newswire classification dataset)
5) Определение цены недвижимости
   (Boston Housing price regression dataset)

Все наборы данных доступны по ссылке : https://keras.io/api/datasets/

При разработке нейронной сети следует соблюсти наличие необходимых составляющих исходя из следующих вариантов:

1) Нейросеть должна состоять из трёх полносвязных слоёв, обязательное использование Dropout, в качестве оптимизатора использовать Adam;
2) Нейросеть должна состоять из четырех полносвязных слоёв, обязательное использование GaussianDropout, в качестве оптимизатора использовать SGD;
3) Нейросеть должна состоять из пяти полносвязных слоёв, обязательное использование ActivityRegularization, в качестве оптимизатора использовать RMSprop.

Выбор количества нейронов на всех внутренних слоях, функций активации и других параметров должен быть обусловлен оптимальностью работы модели.

Формула вычисления варианта:

$N_{в1} = N_{сп} mod\ 5\ + 1$

$N_{в2} = (N_{сп}\ +\ 2)\ mod\ 3\ + 1$

где $N_{сп}$ - номер студента в списке.

Для защиты лабораторной работы следует обосновать выбор значений дополнительных параметров и продемонстрировать работу обученной нейронной сети. Обоснование должно включать в себя демонстрацию качества работы сети на валидационном наборе данных в процессе обучения. Параметры выбираются те, на которых валидация даёт наилучший результат.

Пример разработки и запуска нейронной сети с использованием среды google colab:

```
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import *
from tensorflow.keras import utils
from tensorflow.keras.preprocessing import image
from google.colab import files
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
```

Загрузка данных, объявление списка с названиями классов, нормализация данных, преобразование классов в метки категорий:

```
classes = ['футболка', 'брюки', 'свитер', 'платье', 'пальто', 'туфли', 'рубашка', 'кроссовки', 'сумка', 'ботинки']
(x_train,y_train),(x_test,y_test)=fashion_mnist.load_data()
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train=x_train/255
x_test=x_test/255
y_train = utils.to_categorical(y_train, 10)
y_test = utils.to_categorical(y_test, 10)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
```

Создание последовательной модели, добавление уровней сети, компиляция модели:

```
from tensorflow.keras import activations

model=Sequential()
model.add(Dense(1600,input_dim=784,activation='relu'))
model.add(Dense(10,activation='softmax'))
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1600)              1256000
_________________________________________________________________
dense_1 (Dense)              (None, 10)                16010
=================================================================
Total params: 1,272,010
Trainable params: 1,272,010
Non-trainable params: 0
_________________________________________________________________
```

Обучение сети:

```
model.fit(x=x_train,y=y_train,batch_size=50,epochs=15,validation_split=0.2)
```

```
Epoch 1/15
960/960 [==============================] - 2s 3ms/step - loss: 0.3405 - accuracy: 0.8754 - val_loss: 0.3500 - val_accuracy: 0.8709
Epoch 2/15
960/960 [==============================] - 2s 3ms/step - loss: 0.3124 - accuracy: 0.8851 - val_loss: 0.3528 - val_accuracy: 0.8715
Epoch 3/15
960/960 [==============================] - 2s 3ms/step - loss: 0.2912 - accuracy: 0.8924 - val_loss: 0.3476 - val_accuracy: 0.8752
Epoch 4/15
960/960 [==============================] - 2s 3ms/step - loss: 0.2772 - accuracy: 0.8970 - val_loss: 0.3217 - val_accuracy: 0.8832
Epoch 5/15
960/960 [==============================] - 2s 3ms/step - loss: 0.2596 - accuracy: 0.9023 - val_loss: 0.3171 - val_accuracy: 0.8884
Epoch 6/15
960/960 [==============================] - 2s 3ms/step - loss: 0.2445 - accuracy: 0.9092 - val_loss: 0.3173 - val_accuracy: 0.8871
Epoch 7/15
960/960 [==============================] - 2s 3ms/step - loss: 0.2354 - accuracy: 0.9116 - val_loss: 0.3731 - val_accuracy: 0.8785
```

Оценка доли верных ответов на тестовых данных:

```
L=len(y_test)
correct=0
YP=model.predict(x_test)
for i in range(L):
  y1=np.argmax(y_test[i])
  ypred=np.argmax(YP[i])
  if ypred==y1:
    correct+=1
print(correct,' ',L)
print(correct/L*100)
```

```
8941   10000
89.41
```

Запуск распознавания:

```
prediction = model.predict(x)
```

Результаты распознавания:

```
[ ]  prediction
```

```
array([[0.0000000e+00, 8.6703852e-31, 4.3886688e-01, 3.0877629e-15,
        0.0000000e+00, 2.0248261e-23, 1.6152668e-25, 0.0000000e+00,
        5.6112343e-01, 9.6952454e-06]], dtype=float32)
```

```
prediction = np.argmax(prediction)
print("Номер класса:", prediction)
print("Название класса:", classes[prediction])
```