Минобрнауки России

Федеральное государственное бюджетное образовательное

учреждение высшего образования

**НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ИМ. Р.Е. АЛЕКСЕЕВА**

ИНСТИТУТ РАДИОЭЛЕКТРОНИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

|  |  |  |
| --- | --- | --- |
|  |  |  |

Курс “Аппаратное и программное обеспечение роботизированных систем”

**Отчет по лабораторной работе №3**

Выполнил:

Смирнов А. А.

Проверил:

Гай В.Е.

Нижний Новгород 2021

Задание:

Используя модуль Keras, написать алгоритм работы нейронной сети для распознавания изображений. Используемая модель нейронной сети: ResNet50. Набор данных: MNIST.

Ход работы:

Для начала мы подключаем все модели, которые будут необходимы для реализации нейронной сети:

import tensorflow as tf

import numpy as np

Затем загружаем данные:

(x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.mnist.load\_data()

Приведем загруженные данные к виду, воспринимаемому нейронной сетью. утраиваем каждый черно-белый пиксель, чтобы сымитировать цветное изображение, так как ResNet50 работает с цветными изображениями, а также нормализуем их. Набор ответов также приводим к виду, воспринимаемому нейронной сетью:

# увеличиваем колитество каналов до 3

x\_train = np.expand\_dims(x\_train, axis=-1)

x\_train = np.repeat(x\_train, 3, axis=-1)

# нормализуем

x\_train = x\_train.astype('float32') / 255

# изменяем размер изображений

x\_train = tf.image.resize(x\_train, [32,32]) # if we want to resize

# обрабатываем подписи

y\_train = tf.keras.utils.to\_categorical(y\_train , num\_classes=10)

print(x\_train.shape, y\_train.shape)

Создаем экземпляр модели сети ResNet50 и выводим информацию о слоях:

#определяем вход

input = tf.keras.Input(shape=(32,32,3))

#создаём сеть resnet50

resnet = tf.keras.applications.ResNet50(weights='imagenet',

include\_top = False,

input\_tensor = input)

resnet.trainable = False

resnet.summary()

Создаем составную сеть и выводим информацию о слоях:

# Создание модели составной сети

model = tf.keras.models.Sequential()

# Добавляем сверточные слои

model.add(resnet)

# Преобразуем двумерный массив MobileNet в одномерный

model.add(tf.keras.layers.Flatten())

# Полносвязный слой

model.add(tf.keras.layers.Dense(256, activation='relu'))

# Слой регуляризации (для предотвращения переобучения)

model.add(tf.keras.layers.Dropout(0.5))

# Кол-во классов

model.add(tf.keras.layers.Dense(10, activation='sigmoid'))

model.summary()

Компилируем модель:

#компиляция составной сети

# Создание модели составной сети

model = tf.keras.models.Sequential()

# Добавляем сверточные слои

model.add(resnet)

# Преобразуем двумерный массив MobileNet в одномерный

model.add(tf.keras.layers.Flatten())

# Полносвязный слой

model.add(tf.keras.layers.Dense(256, activation='relu'))

# Слой регуляризации (для предотвращения переобучения)

model.add(tf.keras.layers.Dropout(0.5))

# Кол-во классов

model.add(tf.keras.layers.Dense(10, activation='sigmoid'))

model.summary()

Обучаем составную сеть:

#обучение

model.fit(x\_train, y\_train, batch\_size=64, epochs=10, verbose = 1)

Обобщаем данные в процентах:

#проверка точности

model.evaluate(x\_test,y\_test, batch\_size=10, verbose = 1, )