ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

старший преподаватель		Т.А. Суетина
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

ОБУЧЕНИЕ НЕЙРОСЕТИ – СВЕРТОЧНАЯ НЕЙРОННАЯ СЕТЬ

по курсу: ИНТЕГРИРОВАННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

РАБОТУ ВЫПОЛНИЛ(А)

СТУДЕНТ ГР. №	4128		В. А. Воробьев
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

1 Цель работы

Получить теоретические знания по основным блокам свёрточных нейронных сетей: свёрточные слои, слои подвыборки (пулинга), слои активации и полносвязные слои. Реализовать на конкретном примере блоки сверточных сетей и провести оценку качества.

2 Формулировка задания

1. Загрузить подготовленные данные.
2. Разделить данные на обучающие и проверочные в соответствии 80/20.
3. Построить свёрточную нейронную сеть со свёрточным слоем, слоем подвыборки (пулинга), слоем активации и полносвязным слоем.
4. Провести оценку качества предсказания по коэффициенту сходства.

3 Результаты выполнения работы

Был выбран набор данных, представляющий собой набор изображений фруктов. Размер набора данных – 3000 изображений. Поставленная задача – определить название фрукта.

Для этого была написана программа классификации изображений фруктов с использованием сверточной нейронной сети на языке программирования Python. В программе были использованы библиотеки TensorFlow, Keras, NumPy, Matplotlib, а также ImageDataGenerator для загрузки и предобработки изображений. Полный листинг программы представлен в приложении А.

В программе выбранный набор данных был разделён на обучающею выборку и тестовую в соотношении 80/20.

Разработанная модель сверточной нейронной сети включает в себя следующие слои:

- Входной слой с размером 100x100 пикселей и 3 каналами (RGB).
- Три сверточных слоя (Conv2D) с различным количеством фильтров (32, 64, 128) и функцией активации для выделения характерных признаков изображений.
- Три слоя подвыборки (MaxPooling2D) с размером ядра (2x2) для уменьшения размерности признаковых карт и повышения устойчивости модели.
- Слой Flatten, который преобразовывал тензор в одномерный вектор.
- Полносвязный слой (Dense) с 128 нейронами и функцией активации.
- Выходной слой (Dense) с числом нейронов, равным количеству классов, и функцией активации Softmax для предсказания вероятностей принадлежности к классам.

В процессе обучения модели использовались следующие параметры:

- Количество эпох: 10
- Размер пакета данных: 32
- Метрика оценки: точность (accuracy)

На рисунке 1 представлен график, визуализирующий результаты обучения модели. График показывает, как изменяется точность на обучающем и валидационном (тестовом) наборах данных.

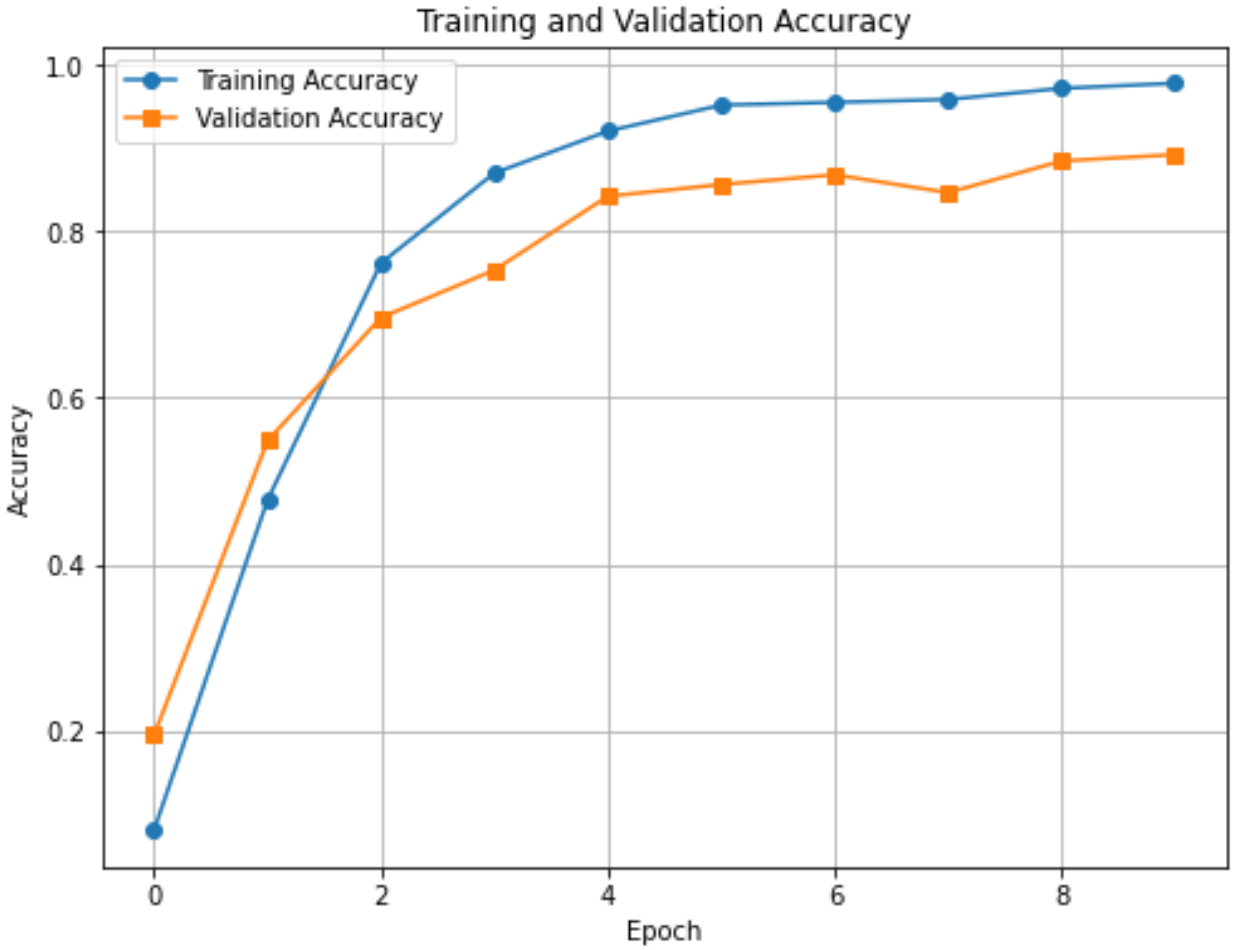

Рисунок 1 – Визуализация результатов обучения

Из графика можно увидеть, что точность при обучении и валидации выше 88%, что свидетельствует о том, что построенная модель хорошо справляется с классификацией фруктов.

На рисунке 2 представлены эпохи (от 1 до 10) и значения точности модели (для обучающей и тестовой выборок), функции потерь (для обучающей и тестовой выборок), время выполнения одного шага обучения для каждой из них.

```
Found 24128 images belonging to 176 classes.
Found 5944 images belonging to 176 classes.
Epoch 1/10
93/93 [==============================] - 118s 1s/step - loss: 4.5143 - accuracy: 0.0813 - val_loss: 3.3828 - val_accuracy: 0.
1952
Epoch 2/10
93/93 [==============================] - 117s 1s/step - loss: 2.0556 - accuracy: 0.4775 - val_loss: 1.7954 - val_accuracy: 0.
5487
Epoch 3/10
93/93 [==============================] - 109s 1s/step - loss: 0.8560 - accuracy: 0.7621 - val_loss: 1.0955 - val_accuracy: 0.
6966
Epoch 4/10
93/93 [==============================] - 102s 1s/step - loss: 0.4504 - accuracy: 0.8703 - val_loss: 0.8821 - val_accuracy: 0.
7537
Epoch 5/10
93/93 [==============================] - 105s 1s/step - loss: 0.2820 - accuracy: 0.9207 - val_loss: 0.6330 - val_accuracy: 0.
8424

Epoch 6/10
93/93 [==============================] - 94s 1s/step - loss: 0.1814 - accuracy: 0.9519 - val_loss: 0.6281 - val_accuracy: 0.8
565
Epoch 7/10
93/93 [==============================] - 94s 1s/step - loss: 0.1600 - accuracy: 0.9553 - val_loss: 0.5335 - val_accuracy: 0.8
683
Epoch 8/10
93/93 [==============================] - 92s 992ms/step - loss: 0.1436 - accuracy: 0.9587 - val_loss: 0.6418 - val_accuracy:
0.8468
Epoch 9/10
93/93 [==============================] - 90s 973ms/step - loss: 0.0997 - accuracy: 0.9721 - val_loss: 0.5071 - val_accuracy:
0.8847
Epoch 10/10
93/93 [==============================] - 99s 1s/step - loss: 0.0699 - accuracy: 0.9782 - val_loss: 0.4916 - val_accuracy: 0.8
925
93/93 [==============================] - 23s 249ms/step - loss: 0.4880 - accuracy: 0.8871
Accuracy: 88.71%
```

Рисунок 2 – Процесс обучения модели

На рисунках 3-12 показан процесс визуализации активаций слоев. На каждом уровне обработки выбранного случайно изображения из тестовой выборки можно увидеть, какие признаки выделяются нейросетью на каждом этапе обработки. На рисунках 4, 6, 8 карты активации слоёв Conv2D демонстрируют какие особенности изображения учитывает модель:

- На первых слоях (conv1) выделяются простые признаки, такие как границы и углы.
- На средних слоях (conv2) сеть начинает выделять более детальные элементы объектов.
- На глубоких слоях (conv3) нейросеть комбинирует выявленные признаки в сложные формы, характерные для конкретных классов.

После каждого слоя подвыборки (MaxPooling2D) размер карты активаций уменьшается, сохраняя наиболее значимые признаки, что делает их более сглаженными и обобщенными.

На последнем этапе выходной слой Softmax преобразует выходные значения нейронов в вероятности принадлежности к различным классам, выбирая наиболее вероятный вариант.

Рисунок 3 – Обработка изображения фрукта Melon Piel de Sapo l нейросетью (1)

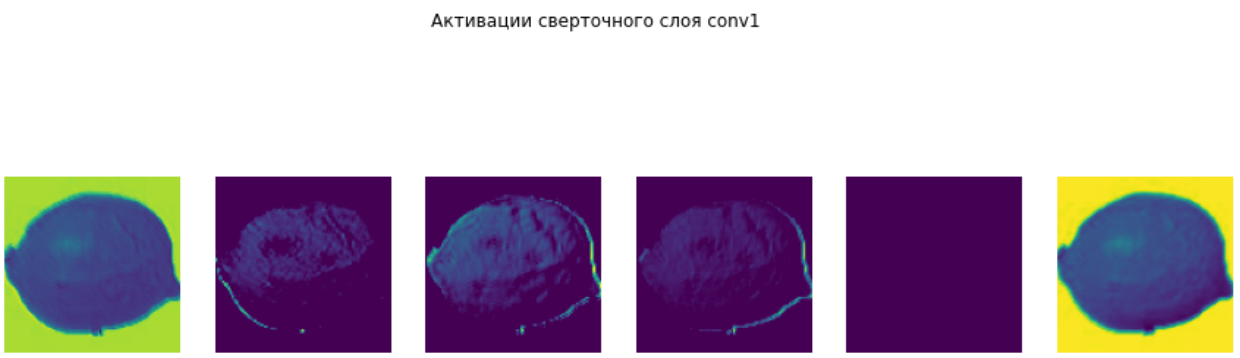

Рисунок 4 – Обработка изображения фрукта Melon Piel de Sapo l нейросетью (2)

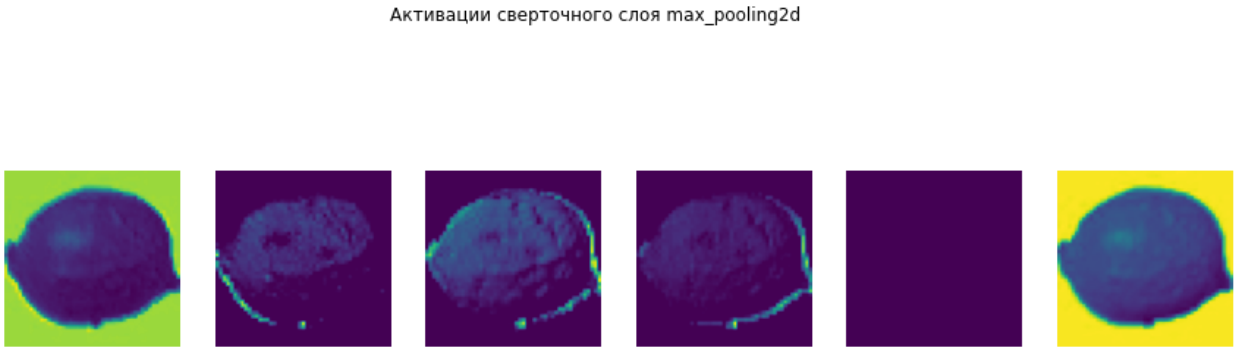

Рисунок 5 – Обработка изображения фрукта Melon Piel de Sapo l нейросетью (3)

Рисунок 6 – Обработка изображения фрукта Melon Piel de Sapo l нейросетью (4)

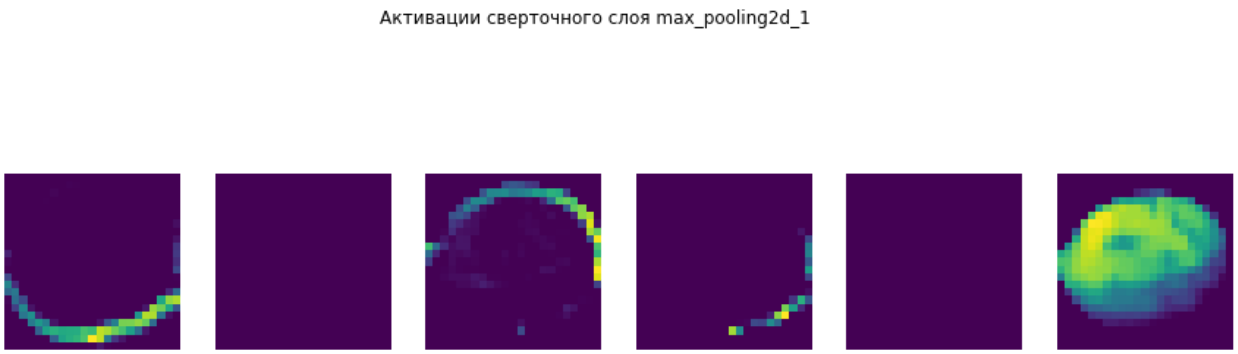

Рисунок 7 – Обработка изображения фрукта Melon Piel de Sapo l нейросетью (5)

Рисунок 8 – Обработка изображения фрукта Melon Piel de Sapo l нейросетью (6)

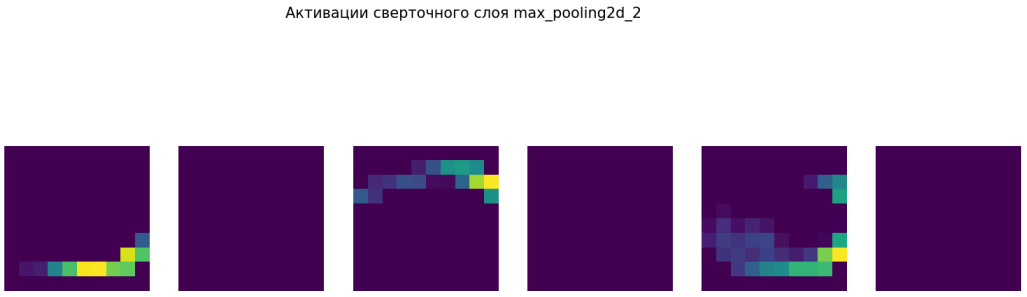

Рисунок 9 – Обработка изображения фрукта Melon Piel de Sapo l нейросетью (7)

Рисунок 10 – Обработка изображения фрукта Melon Piel de Sapo l нейросетью (8)

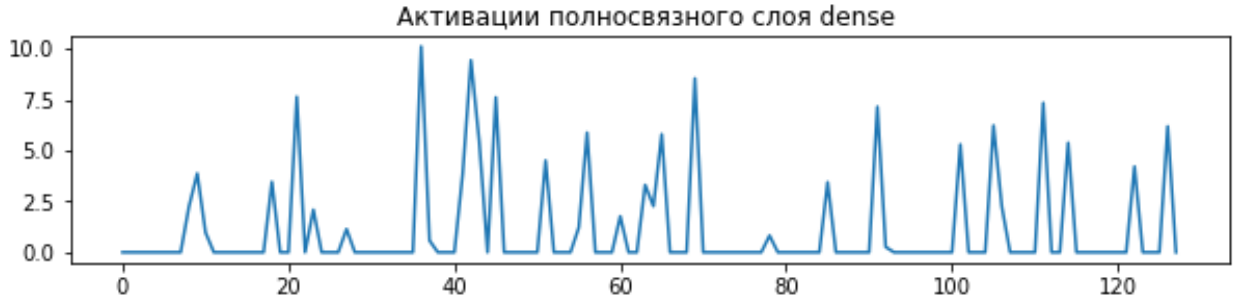

Рисунок 11 – Обработка изображения фрукта Melon Piel de Sapo l нейросетью (9)

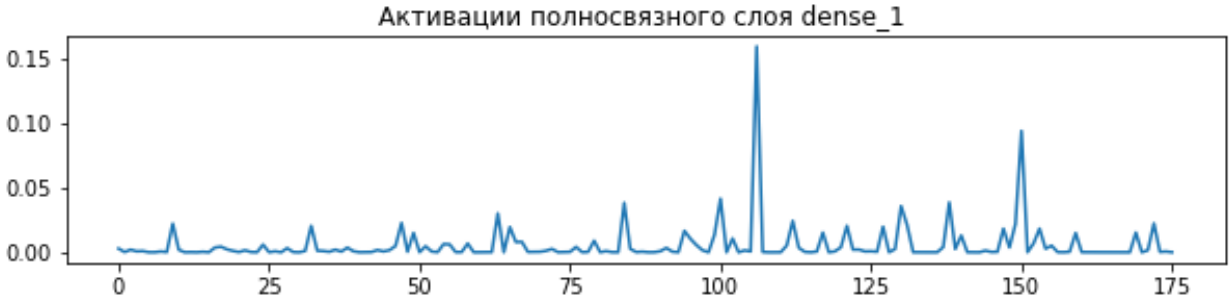

Рисунок 12 – Обработка изображения фрукта Melon Piel de Sapo l нейросетью (10)

На рисунках 13-32 представлены процессы обработки изображений нейронной сетью для двух других фруктов: Strawberry Wegde 1 и Pepper Orange.

Рисунок 13 – Обработка изображения фрукта Strawberry Wegde 1 нейронной сетью (1)

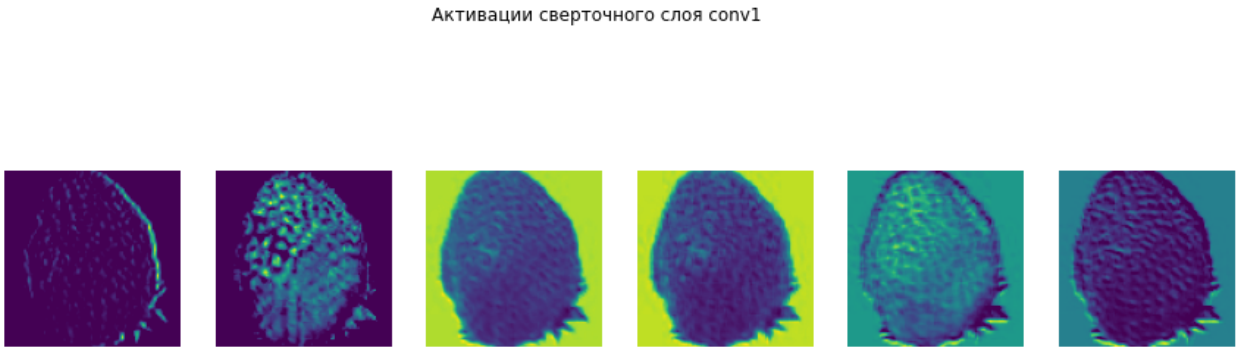

Рисунок 14 – Обработка изображения фрукта Strawberry Wegde 1 нейронной сетью (2)

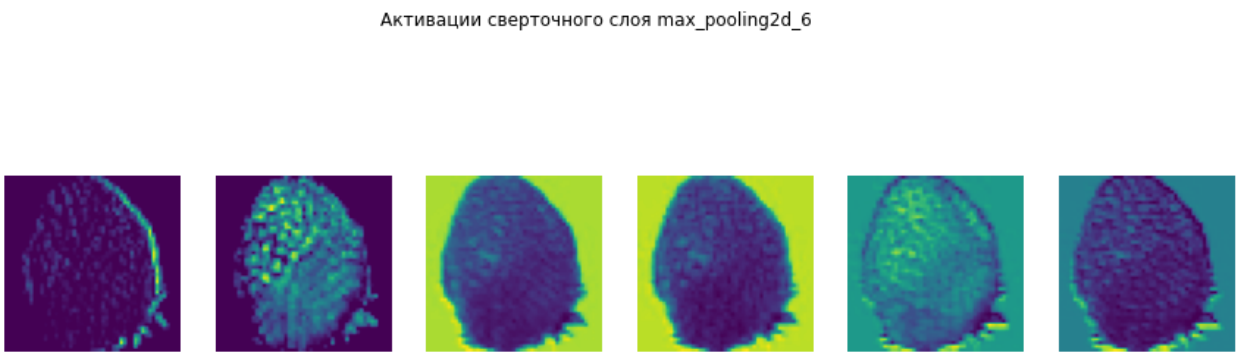

Рисунок 15 – Обработка изображения фрукта Strawberry Wegde 1 нейронной сетью (3)

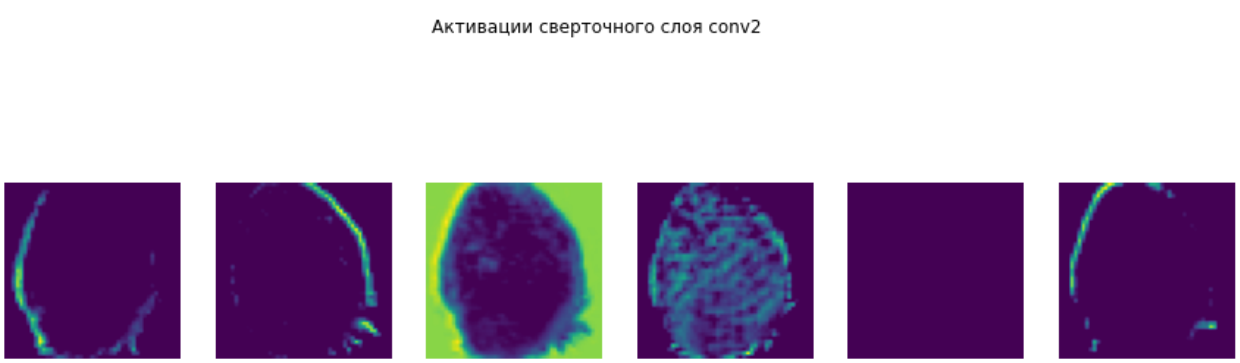

Рисунок 16 – Обработка изображения фрукта Strawberry Wegde 1 нейронной сетью (4)

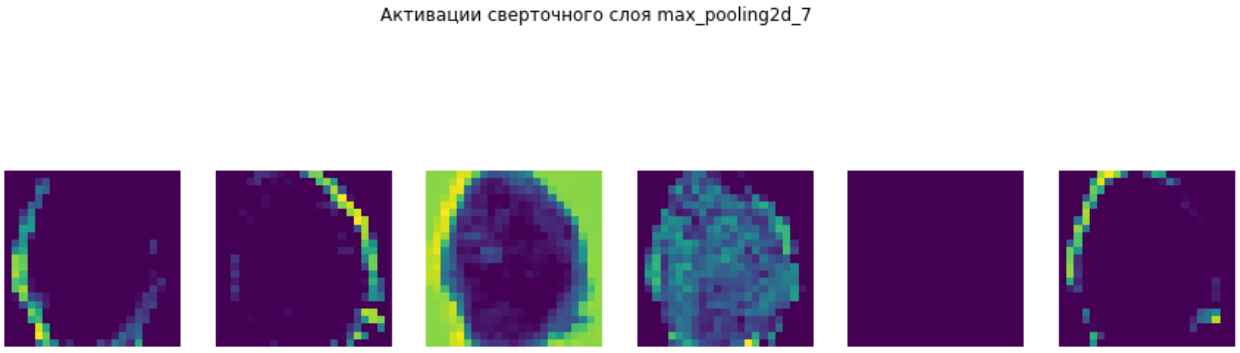

Рисунок 17 – Обработка изображения фрукта Strawberry Wegde 1 нейронной сетью (5)

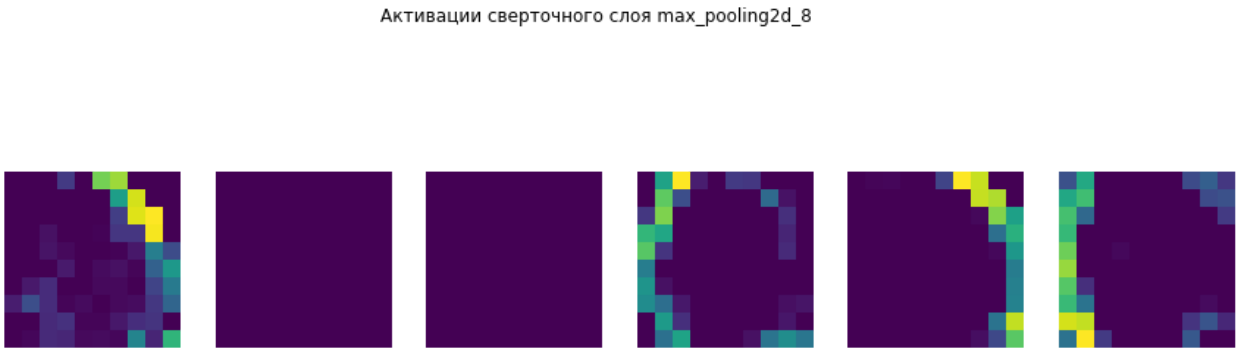

Рисунок 18 – Обработка изображения фрукта Strawberry Wegde 1 нейронной сетью (6)

Рисунок 20 – Обработка изображения фрукта Strawberry Wegde 1 нейронной сетью (8)

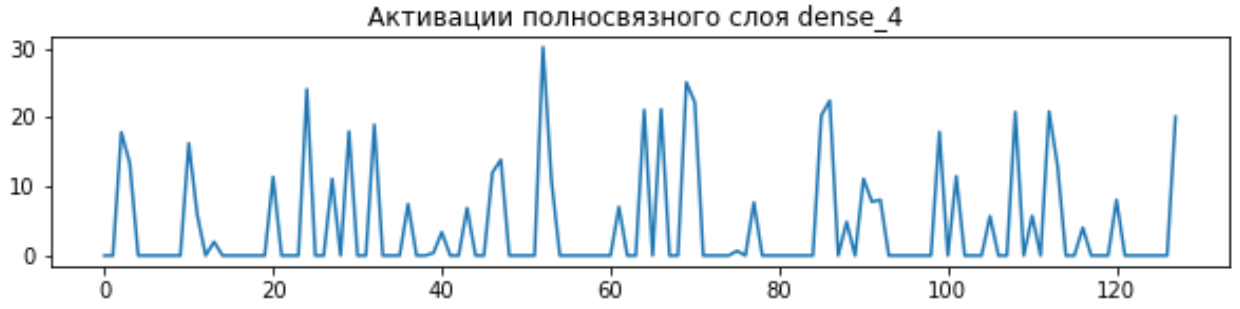

Рисунок 21 – Обработка изображения фрукта Strawberry Wegde 1 нейронной сетью (9)

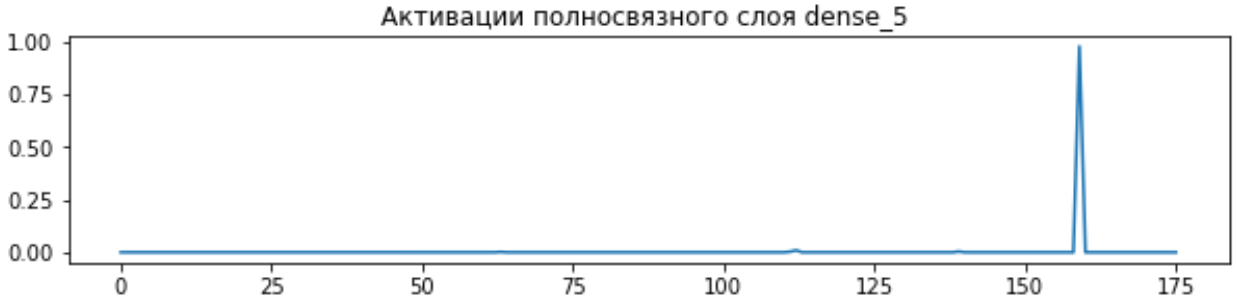

Рисунок 22 – Обработка изображения фрукта Strawberry Wegde 1 нейронной сетью (10)

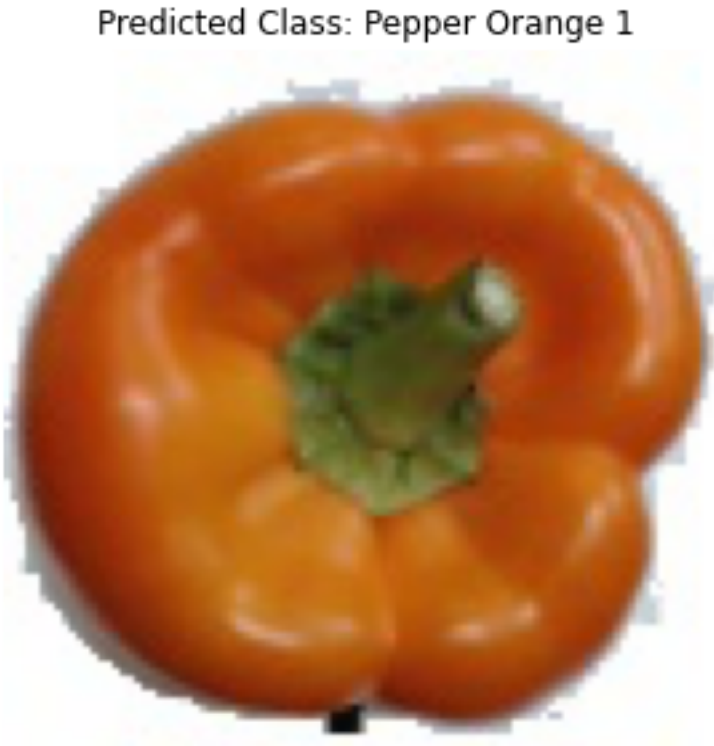

Рисунок 23 – Обработка изображения фрукта Pepper Orange 1 нейронной сетью (1)

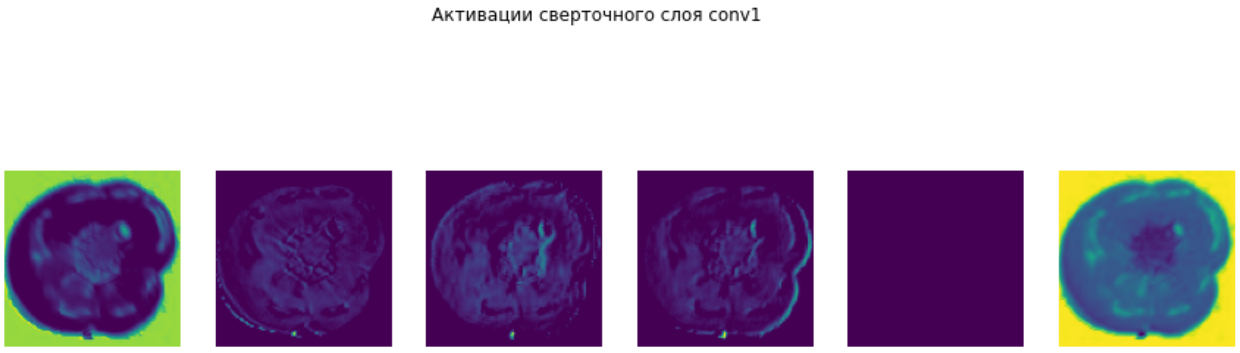

Рисунок 24 – Обработка изображения фрукта Pepper Orange нейронной сетью (2)

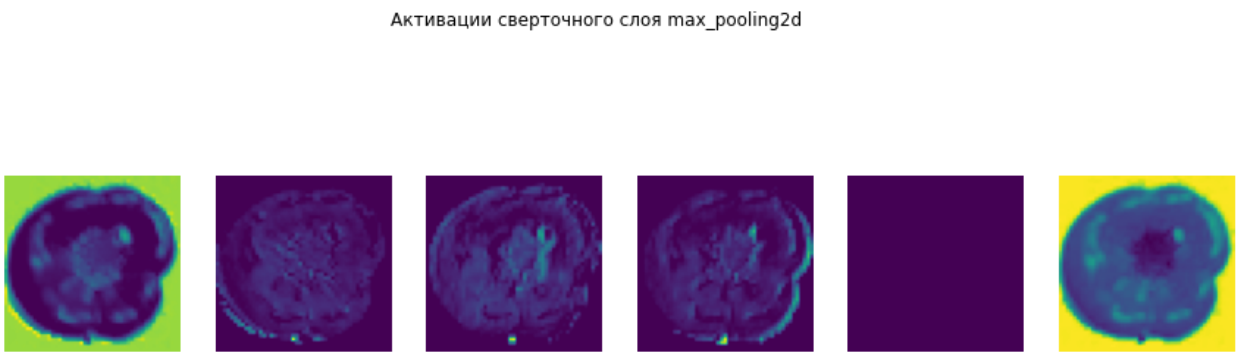

Рисунок 25 – Обработка изображения фрукта Pepper Orange нейронной сетью (3)

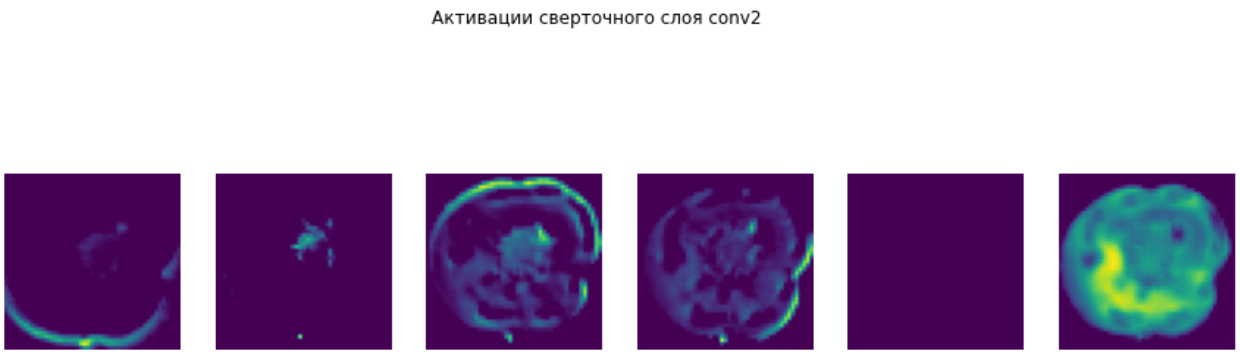

Рисунок 26 – Обработка изображения фрукта Pepper Orange нейронной сетью (4)

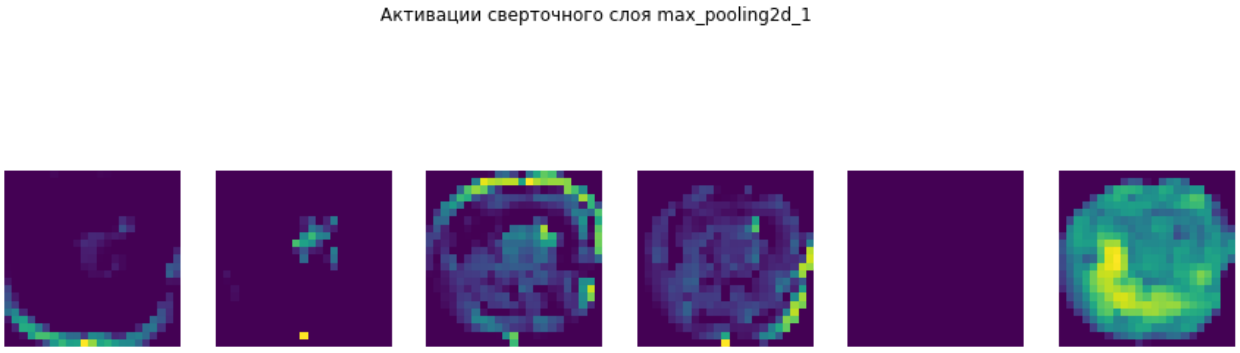

Рисунок 27 – Обработка изображения фрукта Pepper Orange нейронной сетью (5)

Рисунок 28 – Обработка изображения фрукта Pepper Orange нейронной сетью (6)

Рисунок 29 – Обработка изображения фрукта Pepper Orange нейронной сетью (7)

Рисунок 30 – Обработка изображения фрукта Pepper Orange нейронной сетью (8)

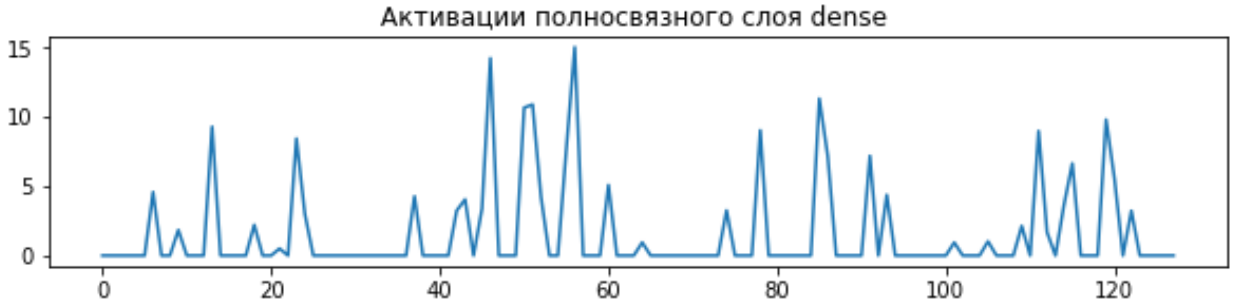

Рисунок 31 – Обработка изображения фрукта Pepper Orange нейронной сетью (9)

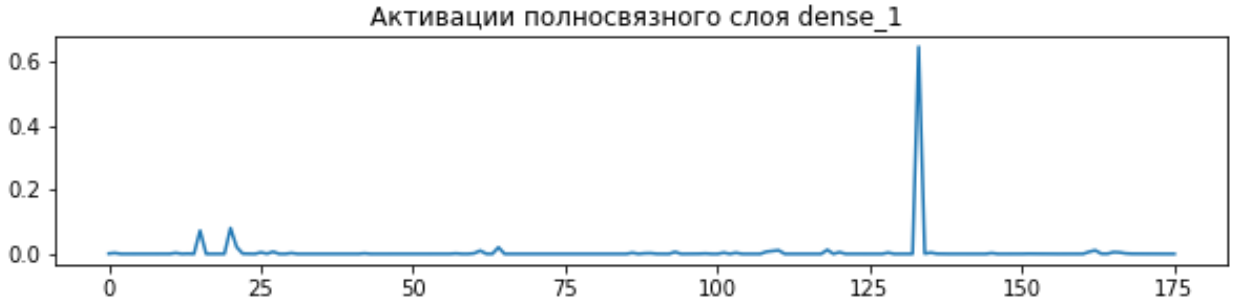

Рисунок 32 – Обработка изображения фрукта Pepper Orange нейронной сетью (10)

Выводы

В ходе выполнения лабораторной работы были получены теоретические знания о основных блоках свёрточных нейронных сетей, включая свёрточные слои, слои подвыборки (пулинга), слои активации и полносвязные слои. На основе изученного материала была реализована модель классификации изображений фруктов с использованием свёрточной нейросети с архитектурой VGG19. Реализация модели выполнена с использованием языка программирования Python и библиотек ImageDataGenerator, TensorFlow, Keras, NumPy и Matplotlib.

Проведенная оценка качества показала точность модели 88,84% на тестовой выборке. Также была выполнена визуализация активаций слоев, позволяющая проанализировать, какие признаки выделяются сетью на разных этапах обработки изображения.

ПРИЛОЖЕНИЕ А

```
import os
import keras
import tensorflow as tf
from keras import layers
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np
from keras.models import Model
from keras.preprocessing import image
import random

# Определение пути к датасету
DATASET_PATH = "fruits-360_100x100/fruits-360/Test"
IMG_SIZE = 100  # Размер изображений
BATCH_SIZE = 32  # Размер партии данных

# Создание генератора изображений с уменьшенным количеством загружаемых данных
datagen = ImageDataGenerator(validation_split=0.2, rescale=1./255)

# Ограничение количества изображений для обучения и валидации
MAX_SAMPLES = 500  # Максимальное количество изображений

# Загрузка обучающего набора данных
train_generator = datagen.flow_from_directory(
    DATASET_PATH,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='sparse',
    subset='training',  # Указываем, что это обучающая выборка
    shuffle=True  # Перемешивание данных
)

# Загрузка валидационного набора данных
validation_generator = datagen.flow_from_directory(
    DATASET_PATH,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='sparse',
    subset='validation',  # Указываем, что это валидационная выборка
    shuffle=True  # Перемешивание данных
)

# Получаем количество классов
num_classes = len(train_generator.class_indices)

# Построение архитектуры сверточной нейронной сети
model = keras.Sequential([
    layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3)),  # Входной слой
    layers.Conv2D(32, (3, 3), activation='relu', name='conv1'),  # Первый
сверточный слой
    layers.MaxPooling2D((2, 2)),  # Первый слой подвыборки
    layers.Conv2D(64, (3, 3), activation='relu', name='conv2'),  # Второй
сверточный слой
    layers.MaxPooling2D((2, 2)),  # Второй слой подвыборки
    layers.Conv2D(128, (3, 3), activation='relu', name='conv3'),  # Третий
сверточный слой
    layers.MaxPooling2D((2, 2)),  # Третий слой подвыборки
```

```
    layers.Flatten(),  # Преобразование тензора в вектор
    layers.Dense(128, activation='relu'),  # Полносвязный слой
    layers.Dense(num_classes, activation='softmax')  # Выходной слой с функцией
активации softmax
])

# Компиляция модели
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Обучение модели
history = model.fit(train_generator, steps_per_epoch=MAX_SAMPLES // BATCH_SIZE,
epochs=10, validation_data=validation_generator, validation_steps=MAX_SAMPLES
// BATCH_SIZE)

# Оценка модели на валидационной выборке
loss, accuracy = model.evaluate(validation_generator, steps=MAX_SAMPLES //
BATCH_SIZE)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Визуализация результатов обучения
plt.figure(figsize=(8, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy', marker='o')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy',
marker='s')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.grid(True)
plt.show()


# Функция для визуализации активаций слоев с выводом исходного изображения
def visualize_layer_outputs(model, generator, class_labels):
    # Получаем случайный пакет изображений и меток из генератора
    img_batch, label_batch = next(generator)  # Используем next() для
извлечения одного пакета
    img = img_batch[0]  # Берем первое изображение из текущего пакета
    label = label_batch[0]  # Берем метку для этого изображения

    # Преобразуем изображение в формат, который подходит для подачи в модель
    img_array = np.expand_dims(img, axis=0)  # Для подачи в модель

    # Прогоняем изображение через модель, чтобы получить предсказание
    prediction = model.predict(img_array)
    predicted_class_index = np.argmax(prediction, axis=1)  # Индекс
предсказанного класса
    predicted_class = class_labels[predicted_class_index[0]]  # Название
предсказанного класса

    # Отображаем исходное изображение с предсказанием
    plt.figure(figsize=(5, 5))
    plt.imshow(img)
    plt.title(f'Predicted Class: {predicted_class}')
    plt.axis('off')
    plt.show()
```

```
    # Создаем модель для извлечения выходов всех слоев
    layer_outputs = [layer.output for layer in model.layers]  # Список выходов
всех слоев
    activation_model = Model(inputs=model.inputs, outputs=layer_outputs)  #
Используем model.inputs

    # Получаем активации для изображения
    activations = activation_model.predict(img_array)

    # Визуализируем активации по порядку
    for i, (layer_name, activation) in enumerate(zip([layer.name for layer in
model.layers], activations)):
        # Для сверточных слоев
        if len(activation.shape) == 4:
            n_filters = activation.shape[-1]
            fig, axes = plt.subplots(1, min(n_filters, 6),
                                     figsize=(15, 5))  # 6 - максимальное
количество фильтров для отображения
            for j, ax in enumerate(axes):
                ax.imshow(activation[0, :, :, j], cmap='viridis')
                ax.axis('off')
            plt.suptitle(f'Активации сверточного слоя {layer_name} ')
            plt.show()

        # Для полносвязных слоев
        elif len(activation.shape) == 2:
            plt.figure(figsize=(10, 2))
            plt.plot(activation[0])
            plt.title(f'Активации полносвязного слоя {layer_name} ')
            plt.show()


# Пример вызова
# Получаем метки классов из генератора
class_labels = list(train_generator.class_indices.keys())  # Получаем метки
классов
while True:
    visualize_layer_outputs(model, train_generator, class_labels)

    user_input = input("Введите 1, чтобы повторить, или 0 для выхода: ")
    if user_input == '0':
        print("Завершение программы.")
        break
```