|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Технологии и инструментарий машинного обучения»

**Практическое занятие № 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИМБО-02-22, Ким Кирилл Сергеевич* | (подпись) | |
| Преподаватель | *Трушин Степан Михайлович, преподаватель* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_202\_\_г. | |  | |

Москва 2025 г.

СОДЕРЖАНИЕ

[ПРАКТИЧЕСКАЯ РАБОТА 3 3](#_Toc195188865)

[Задание 3](#_Toc195188866)

[Введение 4](#_Toc195188867)

[Шаги выполнения 5](#_Toc195188868)

[Этап 1. Общий пример (1 пара) 5](#_Toc195188869)

[Этап 2. Индивидуальные задания (2 пара) 9](#_Toc195188870)

[Этап 3. Анализ и защита 17](#_Toc195188871)

[Результат работы: 19](#_Toc195188872)

[Вывод: 20](#_Toc195188873)

[Список использованных источников и литературы: 20](#_Toc195188874)

ПРАКТИЧЕСКАЯ РАБОТА 3

Задание

**Часть 1: Общий пример (1 пара)**

1. Загрузка и исследование данных:

* Используя Загрузите датасет CIFAR-10 или аналогичный из библиотеки keras.datasets или torchvision.datasets.
* Выполните предварительную обработку данных: нормализация изображений и преобразование классов в категориальный вид.

1. Создание сверточной нейронной сети:

* Архитектура сети:
* Первый сверточный слой с фильтрами (3x3), активация ReLU.
* Пулинг-слой (2x2).
* Добавьте 2-3 сверточных слоя с последующими пулингами.
* Полносвязный слой с Softmax для классификации.
* Инициализация модели с использованием TensorFlow или PyTorch.

1. Обучение сети:

* Оптимизатор: Adam.
* Функция потерь: кросс-энтропия.
* Обучите сеть на 10-20 эпохах и сохраните результаты.

1. Визуализация:

* Графики изменения потерь и точности на обучающей и тестовой выборках.
* Визуализация фильтров первого сверточного слоя.

**Часть 2: Индивидуальные задания (2 пара)**

1. Параметры сети:

* Каждому студенту предоставляется уникальный набор параметров:
* Количество фильтров (32, 64, 128).
* Размеры фильтров (3x3, 5x5).
* Функции активации (ReLU, Sigmoid).
* Постройте сверточную сеть с предложенными параметрами и обучите её на заданном датасете.

1. Анализ моделей:

* Сравните результаты обучения при изменении параметров сети (фильтры, размер свёрток).
* Проанализируйте, как параметры влияют на переобучение и точность на тестовой выборке.

**Часть 3: Анализ и защита**

1. Защитите свою модель, представив основные результаты:

* Итоговые метрики (точность, F1-мера).
* Влияние архитектурных параметров на обучение и переобучение.

1. Используйте инструменты интерпретации (например, Grad-CAM) для визуализации важных областей изображений, на которые обращала внимание сеть.

Введение

**Цель**

Научиться проектировать сверточные нейронные сети, понимать их архитектуру и обучать их на задачах классификации изображений. Освоить использование свёрток, пулинга и других ключевых операций.

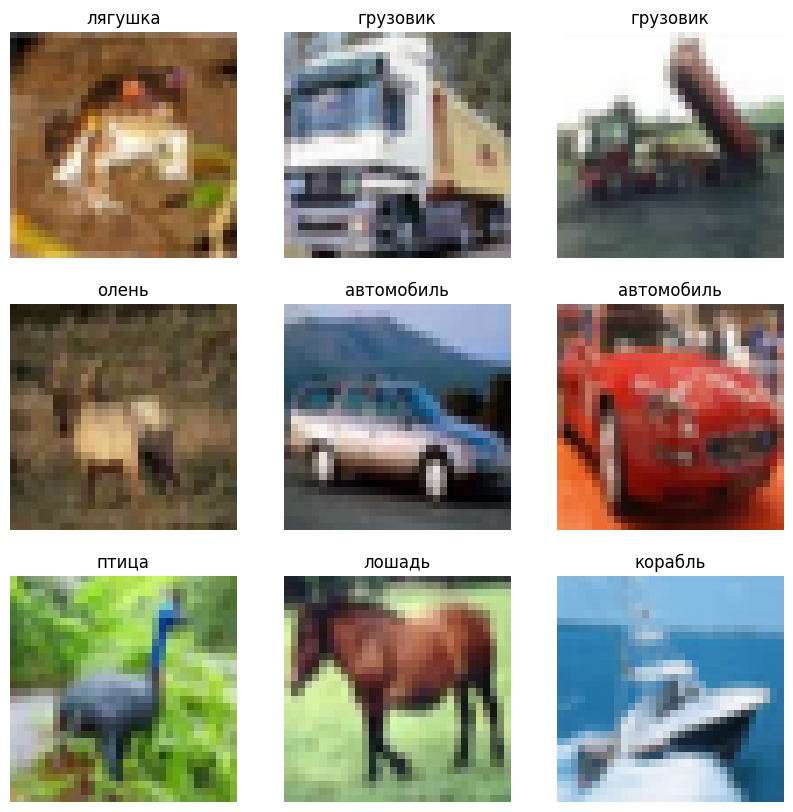
Шаги выполнения

Этап 1. Общий пример (1 пара)

1. Загрузка и предварительная обработка данных.

Листинг 1 – Создание нейронной сети

|  |
| --- |
| import tensorflow as tf  from tensorflow.keras.datasets import cifar10  from tensorflow.keras.utils import to\_categorical  import matplotlib.pyplot as plt  import numpy as np  # Загрузка данных CIFAR-10  (x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()  # Нормализация изображений (приведение к диапазону [0, 1])  x\_train = x\_train.astype('float32') / 255.0  x\_test = x\_test.astype('float32') / 255.0  # Преобразование меток в one-hot encoding  y\_train = to\_categorical(y\_train, 10)  y\_test = to\_categorical(y\_test, 10)  # Вывод информации о данных  print("Форма обучающих данных:", x\_train.shape)  print("Форма тестовых данных:", x\_test.shape)  print("Форма обучающих меток:", y\_train.shape)  print("Форма тестовых меток:", y\_test.shape)  # Визуализация нескольких изображений  class\_names = ['самолет', 'автомобиль', 'птица', 'кошка', 'олень',  'собака', 'лягушка', 'лошадь', 'корабль', 'грузовик']  plt.figure(figsize=(10, 10))  for i in range(9):  plt.subplot(3, 3, i+1)  plt.imshow(x\_train[i])  plt.title(class\_names[np.argmax(y\_train[i])])  plt.axis('off')  plt.show() |



**Рисунок 1 – Визуализация изображений**

1. Создание сверточной нейронной сети.

Листинг 2 – Реализация CNN

|  |
| --- |
| from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense  # Создание модели  model = Sequential([  # Первый сверточный блок  Conv2D(32, (3, 3), activation='relu', padding='same', input\_shape=(32, 32, 3)),  MaxPooling2D((2, 2)),  # Второй сверточный блок  Conv2D(64, (3, 3), activation='relu', padding='same'),  MaxPooling2D((2, 2)),  # Третий сверточный блок  Conv2D(128, (3, 3), activation='relu', padding='same'),  MaxPooling2D((2, 2)),  # Полносвязные слои  Flatten(),  Dense(128, activation='relu'),  Dense(10, activation='softmax')  ])  # Вывод архитектуры модели  model.summary() |

1. Обучение сети

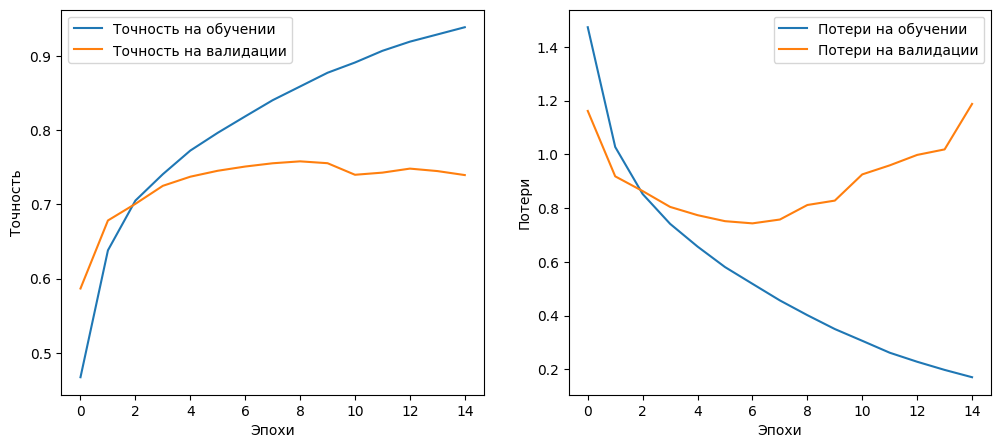
Листинг 3 – Обучение модели

|  |
| --- |
| # Компиляция модели  model.compile(optimizer='adam',  loss='categorical\_crossentropy',  metrics=['accuracy'])  # Обучение модели  history = model.fit(x\_train, y\_train,  epochs=15,  batch\_size=64,  validation\_data=(x\_test, y\_test))  # Сохранение модели  model.save('cifar10\_cnn\_model.h5') |

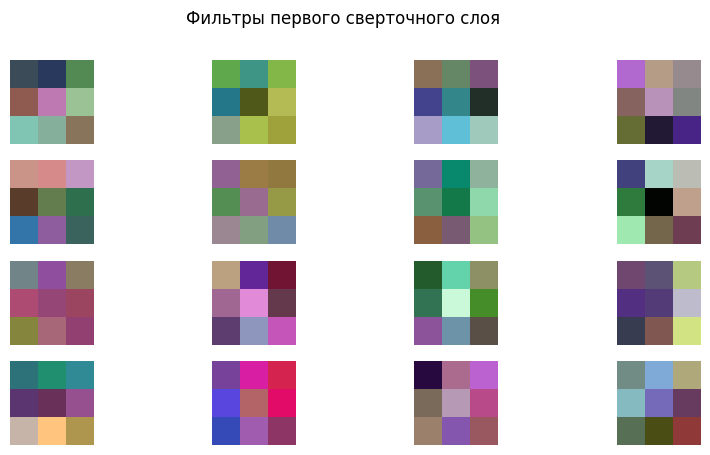
1. Визуализация

Листинг 4 – Графики изменения ошибки

|  |
| --- |
| # Графики точности и потерь  plt.figure(figsize=(12, 5))  # График точности  plt.subplot(1, 2, 1)  plt.plot(history.history['accuracy'], label='Точность на обучении')  plt.plot(history.history['val\_accuracy'], label='Точность на валидации')  plt.xlabel('Эпохи')  plt.ylabel('Точность')  plt.legend()  # График потерь  plt.subplot(1, 2, 2)  plt.plot(history.history['loss'], label='Потери на обучении')  plt.plot(history.history['val\_loss'], label='Потери на валидации')  plt.xlabel('Эпохи')  plt.ylabel('Потери')  plt.legend()  plt.show()  # Визуализация фильтров первого слоя  first\_layer\_weights = model.layers[0].get\_weights()[0]  # Нормализация весов для визуализации  filters = (first\_layer\_weights - first\_layer\_weights.min()) / (first\_layer\_weights.max() - first\_layer\_weights.min())  plt.figure(figsize=(10, 5))  for i in range(16): # Покажем первые 16 фильтров  plt.subplot(4, 4, i+1)  plt.imshow(filters[:, :, :, i])  plt.axis('off')  plt.suptitle('Фильтры первого сверточного слоя')  plt.show() |



**Рисунок 2 – Графики точности и потерь**



**Рисунок 3 – Визуализация фильтров первого слоя**

Этап 2. Индивидуальные задания (2 пара)

1. Загрузка изображений в Dataframe

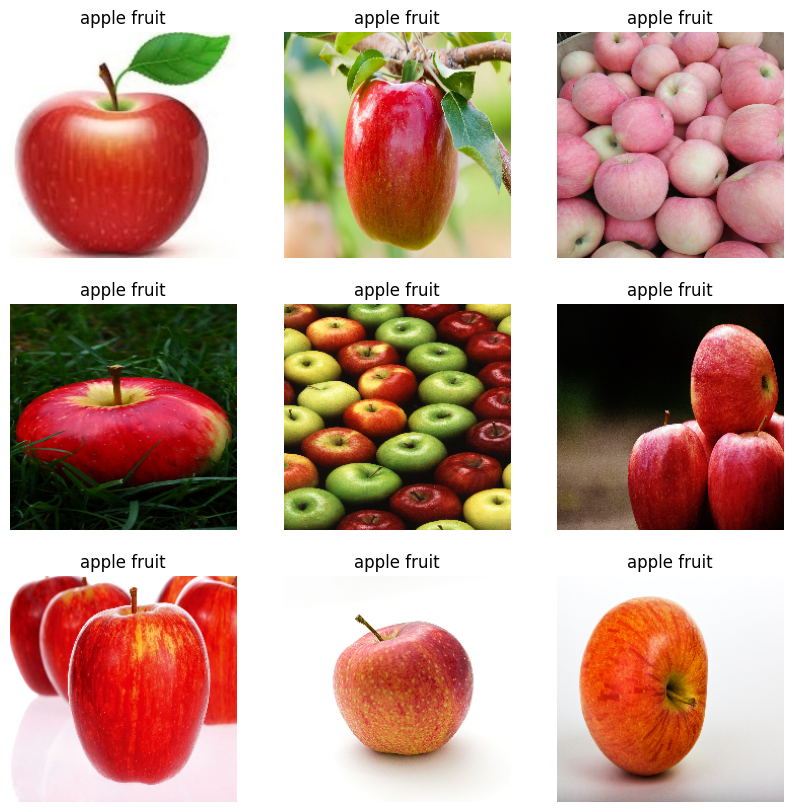
Листинг 5 – Загрузка изображений

|  |
| --- |
| import opendatasets as od  od.download('https://www.kaggle.com/datasets/shreyapmaher/fruits-dataset-images/data')  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  import warnings  warnings.filterwarnings("ignore")  import tensorflow as tf  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Dense, Flatten, Dropout, MaxPooling2D, Conv2D  from keras.utils import to\_categorical  from tensorflow.keras.optimizers import Adam  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import classification\_report, confusion\_matrix  from sklearn.preprocessing import LabelEncoder  from sklearn.utils import shuffle  from PIL import Image  from tensorflow.keras.preprocessing.image import ImageDataGenerator  from tensorflow.keras.callbacks import EarlyStopping  import os  import cv2  X\_train = []  y\_train = []  labels = ["apple fruit", "banana fruit", "cherry fruit", "chickoo fruit", "grapes fruit",  "kiwi fruit", "mango fruit", "orange fruit", "strawberry fruit"]  image\_size = 224  for label in labels:  folderPath = os.path.join("/content/fruits-dataset-images/images", label)    if not os.path.exists(folderPath):  print(f"Ошибка: {folderPath} не найден")  continue    for path in os.listdir(folderPath):  image\_path = os.path.join(folderPath, path)    if not os.path.exists(image\_path):  print(f"Ошибка: {image\_path} не найден")  continue    image = cv2.imread(image\_path)    if image is None:  print(f"Ошибка: {image\_path} не может быть загружен")  continue    # Конвертируем BGR в RGB и изменяем размер  image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  image = cv2.resize(image, (image\_size, image\_size))  X\_train.append(image)  y\_train.append(label)  X\_test = []  y\_test = []  labels = ["apple fruit", "banana fruit", "cherry fruit", "chickoo fruit", "grapes fruit",  "kiwi fruit", "mango fruit", "orange fruit", "strawberry fruit"]  image\_size = 224  for label in labels :  folderPath = os.path.join("/content/fruits-dataset-images/images", label)  if not os.path.exists(folderPath):  print(f"Ошибка: {folderPath} не найден")  continue    for path in os.listdir(folderPath):  image\_path = os.path.join(folderPath, path)    if not os.path.exists(image\_path):  print(f"Ошибка: {image\_path} не найден")  continue    image = cv2.imread(image\_path)    if image is None:  print(f"Ошибка: {image\_path} не может быть загружен")  continue    # Конвертируем BGR в RGB и изменяем размер  image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  image = cv2.resize(image, (image\_size, image\_size))  X\_test.append(image)  y\_test.append(label) |

1. Преобразование данных и визуализация

Листинг 6 – Визуализация изображений

|  |
| --- |
| # Преобразование данных  X\_train = np.array(X\_train) / 255.0  X\_test = np.array(X\_test) / 255.0  # Кодирование меток  label\_encoder = LabelEncoder()  y\_train\_encoded = label\_encoder.fit\_transform(y\_train)  y\_test\_encoded = label\_encoder.transform(y\_test)  y\_train\_categorical = to\_categorical(y\_train\_encoded)  y\_test\_categorical = to\_categorical(y\_test\_encoded)  # Визуализация  plt.figure(figsize=(10, 10))  for i in range(9):  plt.subplot(3, 3, i+1)  plt.imshow(X\_train[i])  plt.title(labels[np.argmax(y\_train\_categorical[i])])  plt.axis('off')  plt.show() |



**Рисунок 4 – Визуализация данных**

1. Построение сверточной сети с такими параметрами, которые я выбрал

* Количество фильтров: 32;
* Размеры фильтров: 3x3;
* Функция активации: Relu.

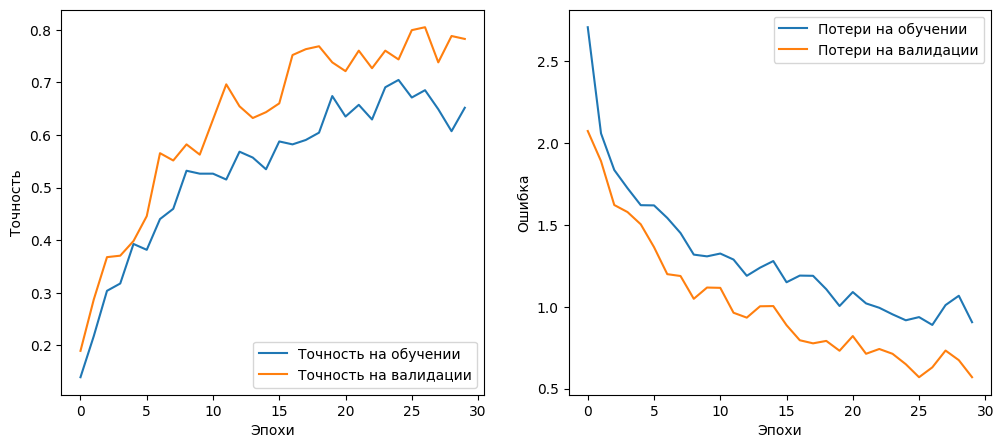
Листинг 7 – Построение сверточной нейронной сети

|  |
| --- |
| FILTERS = 32  KERNEL\_SIZE = (3, 3)  ACTIVATION = 'relu'  IMAGE\_SIZE = 224  # Создание модели  model = Sequential([  Conv2D(FILTERS, KERNEL\_SIZE, activation=ACTIVATION,  padding='same', input\_shape=(IMAGE\_SIZE, IMAGE\_SIZE, 3)),  MaxPooling2D((2, 2)),  Conv2D(FILTERS\*2, KERNEL\_SIZE, activation=ACTIVATION, padding='same'),  MaxPooling2D((2, 2)),  Conv2D(FILTERS\*4, KERNEL\_SIZE, activation=ACTIVATION, padding='same'),  MaxPooling2D((2, 2)),  Flatten(),  Dense(128, activation=ACTIVATION),  Dropout(0.5),  Dense(9, activation='softmax')  ])  model.compile(optimizer=Adam(learning\_rate=0.001),  loss='categorical\_crossentropy',  metrics=['accuracy'])  # Аугментация данных  datagen = ImageDataGenerator(  rotation\_range=20,  width\_shift\_range=0.2,  height\_shift\_range=0.2,  horizontal\_flip=True,  zoom\_range=0.2  )  # Обучение с ранней остановкой  early\_stopping = EarlyStopping(monitor='val\_loss', patience=5, restore\_best\_weights=True)  history = model.fit(  datagen.flow(X\_train, y\_train\_categorical, batch\_size=32),  epochs=30,  validation\_data=(X\_test, y\_test\_categorical),  callbacks=[early\_stopping]  ) |

1. Строим графики ошибки и точности

Листинг 8 – Визуализация

|  |
| --- |
| # Визуализация обучения  plt.figure(figsize=(12, 5))  plt.subplot(1, 2, 1)  plt.plot(history.history['accuracy'], label='Точность на обучении')  plt.plot(history.history['val\_accuracy'], label='Точность на валидации')  plt.ylabel('Точность')  plt.xlabel('Эпохи')  plt.legend()  plt.subplot(1, 2, 2)  plt.plot(history.history['loss'], label='Потери на обучении')  plt.plot(history.history['val\_loss'], label='Потери на валидации')  plt.ylabel('Ошибка')  plt.xlabel('Эпохи')  plt.legend()  plt.show()  # Оценка модели  test\_loss, test\_acc = model.evaluate(X\_test, y\_test\_categorical, verbose=0)  print(f'Точность: {test\_acc:.4f}')  print(f'Потеря: {test\_loss:.4f}') |

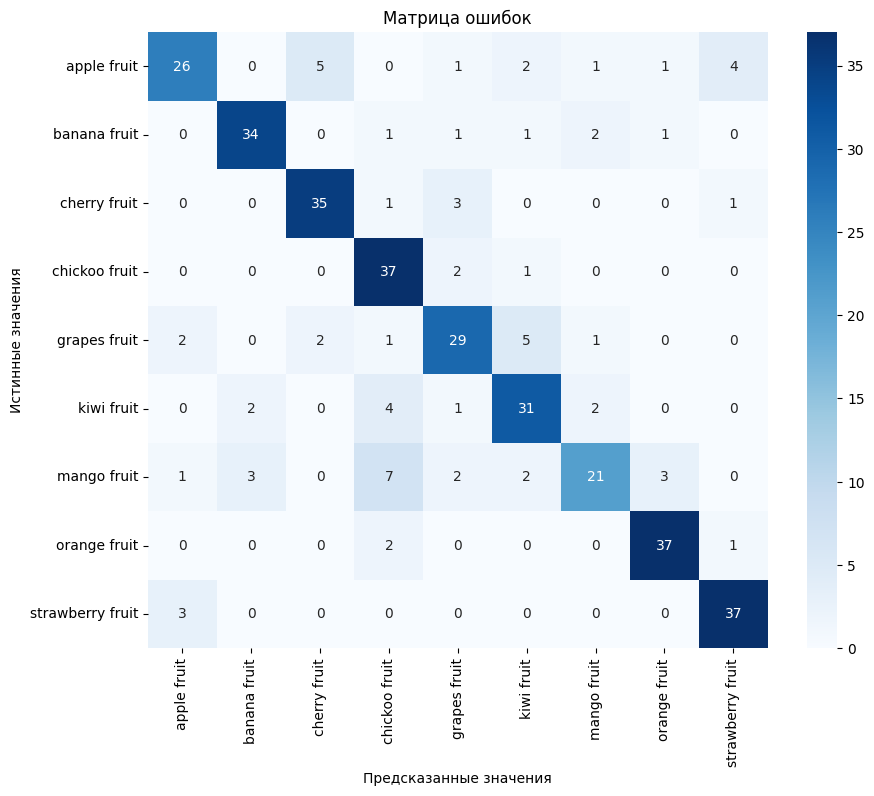


**Рисунок 5 – Графики ошибки и точности**

1. Матрица ошибок

Листинг 9 – Отчет классификации и матрица ошибок

|  |
| --- |
| y\_pred = model.predict(X\_test)  y\_pred\_classes = np.argmax(y\_pred, axis=1)  y\_true = np.argmax(y\_test\_categorical, axis=1)  print("Отчет о классификации:")  print(classification\_report(y\_true, y\_pred\_classes, target\_names=labels))  # Матрица ошибок  plt.figure(figsize=(10, 8))  cm = confusion\_matrix(y\_true, y\_pred\_classes)  sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',  xticklabels=labels, yticklabels=labels)  plt.title('Матрица ошибок')  plt.xlabel('Предсказанные значения')  plt.ylabel('Истинные значения')  plt.show() |



**Рисунок 6 – Матрица ошибок**

Этап 3. Анализ и защита

1. Анализ результатов (итоговые метрики F1-мера)

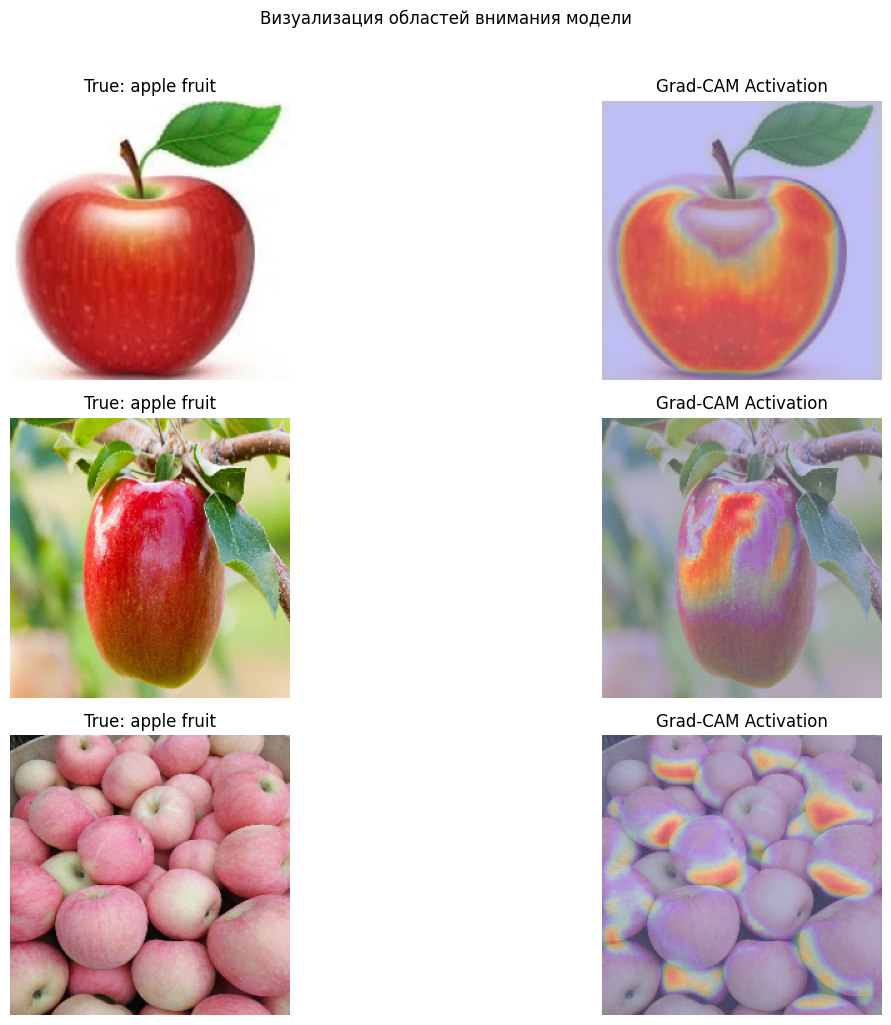
Листинг 10 – Вычисление метрик

|  |
| --- |
| # Вычисление F1-меры  from sklearn.metrics import f1\_score  f1 = f1\_score(y\_true, y\_pred\_classes, average='weighted')  print(f'F1-Score: {f1:.4f}')  # Анализ переобучения  train\_acc = history.history['accuracy'][-1]  val\_acc = history.history['val\_accuracy'][-1]  overfitting = train\_acc - val\_acc  print(f'\nТочность обучения: {train\_acc:.4f}')  print(f'Точность валидации: {val\_acc:.4f}')  print(f'Переобучение: {overfitting:.4f}') |

1. Визуализация с помощью Grad-CAM

Листинг 11 – Визуализация

|  |
| --- |
| from tf\_keras\_vis.gradcam import Gradcam  from tf\_keras\_vis.utils.model\_modifiers import ReplaceToLinear  from tf\_keras\_vis.utils.scores import CategoricalScore  # Подготовка модели для Grad-CAM  def model\_modifier(cloned\_model):  cloned\_model.layers[-1].activation = tf.keras.activations.linear  return cloned\_model  # Выбор трех примеров для визуализации  sample\_images = X\_test[:3]  sample\_labels = y\_true[:3]  # Создание Grad-CAM  gradcam = Gradcam(model, model\_modifier, clone=True)  def score\_function(output):  return [output[i, label] for i, label in enumerate(sample\_labels)]  # Генерация тепловых карт  cam = gradcam(score\_function, sample\_images, penultimate\_layer=-5) # Последний сверточный слой  # Визуализация  plt.figure(figsize=(15, 10))  for i in range(3):  # Оригинальное изображение  plt.subplot(3, 2, 2\*i+1)  plt.imshow(sample\_images[i])  plt.title(f"True: {labels[sample\_labels[i]]}")  plt.axis('off')  # Grad-CAM  plt.subplot(3, 2, 2\*i+2)  heatmap = np.uint8(255 \* cam[i])  plt.imshow(heatmap, cmap='jet', alpha=0.5)  plt.imshow(sample\_images[i], alpha=0.5)  plt.title("Grad-CAM Activation")  plt.axis('off')  plt.suptitle('Визуализация областей внимания модели', y=1.02)  plt.tight\_layout()  plt.show() |



**Рисунок 7 – Визуализация областей**

Результат работы:

Данную работу можете увидеть в блокноте Google Colab:

<https://colab.research.google.com/drive/16G98AZwaepLVH3NB7ykzCjTb6PnGIck7?usp=sharing>

Вывод:

Можно сделать вывод, мы реализовали CNN-модель с параметрами (32 фильтра, ядро 3x3, ReLU) демонстрирует хорошее качество классификации и может быть использована в реальных системах сортировки фруктов. Дальнейшее улучшение точности целесообразно проводить через применение методов трансферного обучения и более сложных архитектур.

Список использованных источников и литературы:

1. Ростовцев В.С. Искусственные нейронные сети,   
   Издательство "Лань", 2019. — 216 с. — URL: https://e.lanbook.com/book/122180
2. Араки М. Манга: Машинное обучение,   
   Издательство "ДМК Пресс", 2020. — 214 с. — URL: <https://e.lanbook.com/book/179473>
3. Платонов, А. В. Машинное обучение : учебное пособие для вузов / А. В. Платонов. — Москва : Издательство Юрайт, 2022. — 85 с. — (Высшее образование). — ISBN 978-5-534-15561-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/508804