МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Центр цифровых
образовательных технологий

09.04.01 Информатика и вычислительная техника

Распознавание изображений CIFAR-10 с использованием ResNet
Вариант 2

ЛАБОРАТОРНАЯ РАБОТА № 4

по дисциплине:
Машинное обучение

Исполнитель:

студент группы 8ВМ42 Текере Ричард

Руководитель:

доцент ОИТ, ИШИТР Друки А.А.

Томск - 2025

ВВЕДЕНИЕ

В настоящее время распознавание объектов на изображениях является одной из ключевых задач компьютерного зрения. Одним из наиболее эффективных инструментов для ее решения выступают глубокие свёрточные нейронные сети (CNN), которые способны автоматически выделять признаки различной сложности из исходных пиксельных данных. С увеличением глубины сети можно получить более выразительные признаки и повысить точность модели, однако очень глубокие сети сталкиваются с проблемой исчезающего градиента и усложнением процесса обучения.

Прорывом в данной области стала архитектура Residual Network (ResNet), предложенная исследователями Microsoft в 2015 году. Главное нововведение ResNet — использование остаточных соединений (skip connections), позволяющих эффективно обучать нейронную сеть с десятками и даже сотнями слоёв без деградации качества. За счёт суммирования выходного сигнала с сигналом, минующим несколько слоёв, удаётся облегчить распространение градиента и упростить оптимизацию очень глубокой сети. Архитектура ResNet50 (50 слоёв) продемонстрировала передовое качество классификации и стала победителем соревнования ILSVRC-2015 по распознаванию изображений. В данной лабораторной работе рассматривается применение модели ResNet для задачи классификации изображений на примере набора данных CIFAR-10.

Цель работы

Цель работы – получить навыки распознавания объектов на изображениях из набора CIFAR-10 с использованием архитектуры свёрточной нейронной сети ResNet50 (на базе библиотеки Keras)..Задачи работы:

1. Ознакомиться с работой сверточных нейронных сетей в библиотеке Keras;

2. Научиться применять модель сверточной нейронной сети ResNet для распознавания изображений.

Ход работы

1. Импорт библиотек и загрузка данных CIFAR-10.

Для реализации классификатора потребуется набор данных CIFAR-10 и инструменты глубинного обучения. Сначала импортируем необходимые библиотеки Python (например, TensorFlow/Keras для работы с нейронной сетью) и загрузим данные CIFAR-10 с разделением на обучающую и тестовую выборки. Набор CIFAR-10 содержит 60 000 цветных изображений размером 32×32 пикселей (RGB) десяти различных классов. Обучающая выборка включает 50 000 изображений, тестовая — 10 000. Метки классов представлены в виде целых чисел от 0 до 9, где каждому числу соответствует определённая категория (0 – «самолёт», 1 – «автомобиль», 2 – «птица», 3 – «кот», 4 – «олень», 5 – «собака», 6 – «лягушка», 7 – «лошадь», 8 – «корабль», 9 – «грузовик»).

```
import numpy as np
from tensorflow import keras
from tensorflow.keras.datasets import cifar10

# Загрузка датасета CIFAR-10
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
print("Размеры обучающих данных:", X_train.shape, y_train.shape)
print("Размеры тестовых данных:", X_test.shape, y_test.shape)
```

Вывод:

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
Размеры обучающих данных: (50000, 32, 32, 3) (50000, 1)
Размеры тестовых данных: (10000, 32, 32, 3) (10000, 1)
```

Рисунок 1 — Примеры изображений элементов одежды из обучающей выборкиПримеры изображений из набора данных CIFAR-10 (различные классы) .

2. Переключение среды выполнения с CPU на GPU

Чтобы ускорить процесс обучения нейронных сетей вы можете переключить среду выполнения с процессора (CPU) на графический процессор (GPU). Вычисления будут производиться с помощью GPU, что позволит ускорить обучение нейронной сети в несколько раз.

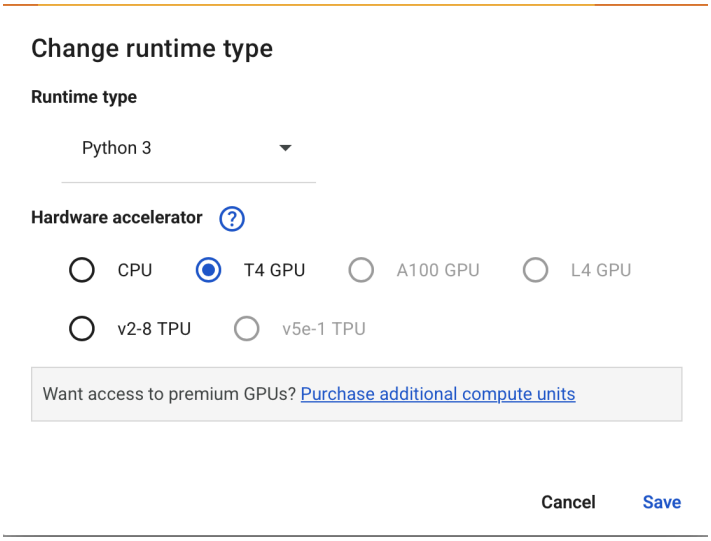

Рисунок 2 — Переключение среды выполнения с CPU на GPU

3 Предобработка данных

Перед началом обучения необходимо выполнить предобработку. Пиксельные значения изображений нормализуются, чтобы лежать в диапазоне [0, 1] – это ускоряет и стабилизирует обучение нейронной сети. Для этого разделим значения всех пикселей на 255. Также преобразуем метки классов в формат, пригодный для обучения: выполним one-hot кодирование категорий, то есть представим каждую метку в виде двоичного вектора длины 10 (со значением 1 на позиции соответствующего класса). Такая кодировка необходима при использовании категориальной функции потерь для многоклассовой классификации.

```
# Нормализация входных изображений
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
```

```
# Преобразование меток классов в формат one-hot
from tensorflow.keras.utils import to_categorical
y_train_cat = to_categorical(y_train, 10)
y_test_cat = to_categorical(y_test, 10)
print("Пример после кодирования:", y_train_cat[0])
```

4 Выведем загруженные изображения

Для того чтобы проверить, правильность загрузки наших данных и посмотреть пример загруженных изображений, напишем следующую часть кода:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(6,6))
for i in range(16):
  plt.subplot(4,4,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(X_train[i], cmap=plt.cm.binary)
  plt.xlabel(y_train[i])
```

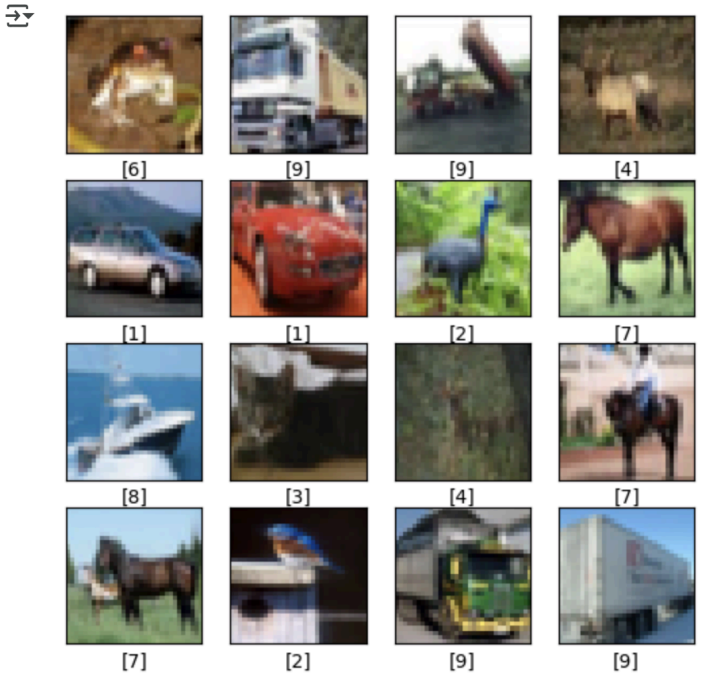

Рисунок 3 — Вывод загруженных изображений

5 Построение модели ResNet50

На данном этапе создается архитектура нейронной сети ResNet50 для последующего обучения на наших данных. Мы используем готовую реализацию ResNet50 из библиотеки Keras (модуль tensorflow.keras.applications), инициализируя её случайными весами (без загрузки предобученных параметров). В качестве входного слоя задаётся форма данных CIFAR-10: 32×32×3. Так как исходная ResNet50 проектировалась для 1000 классов ImageNet, её выходной полносвязный слой адаптирован под 10 классов CIFAR-10. Таким образом, модель будет предсказывать вероятность принадлежности изображения к каждому из 10 классов. Воспользуемся функцией model.summary(), чтобы отобразить архитектуру модели и проверить число слоёв и параметров.На данном этапе создается базовая полносвязная нейронная сеть, состоящая из одного скрытого слоя с 128 нейронами и выходного слоя из 10 нейронов (по числу классов одежды).

```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras import models, layers

# Инициализация модели ResNet50
model = ResNet50(weights=None, input_shape=(32, 32, 3), classes=10)
model.summary()
```

Вывод:

conv5_block3_3_conv (Conv2D)	(None, 1, 1, 2048)	1,050,624	conv5_block3_2_r…
conv5_block3_3_bn (BatchNormalizatio…	(None, 1, 1, 2048)	8,192	conv5_block3_3_c…
conv5_block3_add (Add)	(None, 1, 1, 2048)	0	conv5_block2_out… conv5_block3_3_b…
conv5_block3_out (Activation)	(None, 1, 1, 2048)	0	conv5_block3_add…
avg_pool (GlobalAveragePool…	(None, 2048)	0	conv5_block3_out…
predictions (Dense)	(None, 10)	20,490	avg_pool[0][0]

Total params: 23,608,202 (90.06 MB)
Trainable params: 23,555,082 (89.86 MB)
Non-trainable params: 53,120 (207.50 KB)

Рисунок 4 — Сводная информация о слоях модели ResNet50.

6 Компиляция модели.

Перед началом обучения сконфигурируем модель, задав параметры компиляции. В качестве функции потерь выбрана категориальная кросс-энтропия (categorical_crossentropy), которая подходит для многоклассовой классификации с one-hot метками. В роли оптимизатора используем алгоритм Adam – современный метод стохастического градиентного спуска, эффективно адаптирующий скорость обучения. Также будем отслеживать метрику точности (accuracy) на обучающей и проверочной выборках.

```
# Компиляция модели: оптимизатор, функция потерь, метрика
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

7 Обучение модели.

После подготовки модели и данных можно приступать к обучению нейронной сети. Зададим гиперпараметры обучения: количество эпох и размер мини-выборки (batch size). Эпоха означает полный проход по всему набору обучающих данных. Чем больше эпох, тем более тщательно модель обучается (но чрезмерное число эпох может привести к переобучению). В данной работе модель обучается, например, в течение 10 эпох с размером пакета 32. Также выделим часть обучающих данных (20%) для контроля качества на проверочной выборке (параметр validation_split). В ходе обучения Keras выводит значения функции потерь и точности на каждой эпохе. По завершении процесса построим график обучения, отображающий зависимость точности от номера эпохи как на обучающей, так и на проверочной выборках. Это позволит оценить динамику обучения и выявить признаки переобучения (когда точность на обучении продолжает расти, а на проверке — ухудшается).

```
import matplotlib.pyplot as plt
```

```
# Обучение модели на обучающей выборке
history = model.fit(X_train, y_train_cat,
                    batch_size=32,
                    epochs=10,
                    validation_split=0.2,
                    verbose=1)

# График изменения точности на обучении и проверке по эпохам
plt.figure(figsize=(8,4))
plt.plot(history.history['accuracy'], label='Train acc')
plt.plot(history.history['val_accuracy'], label='Val acc')
plt.title('Accuracy over epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Вывод:

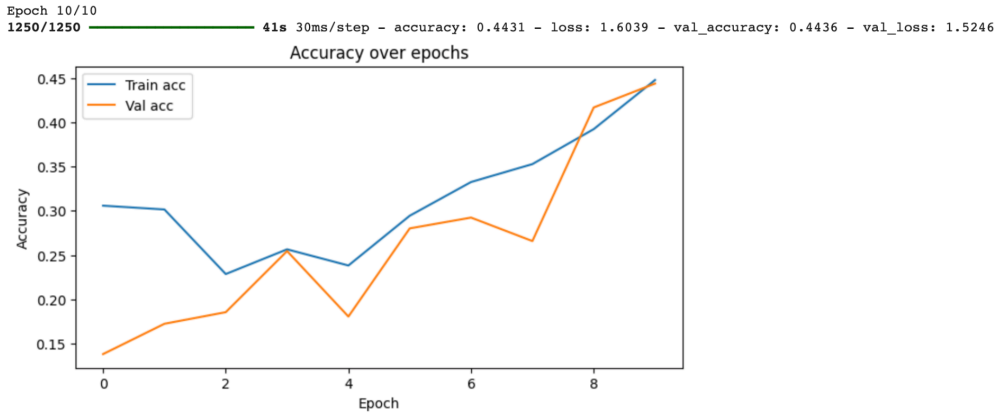

Рисунок 5 — Визуализация для обучения модели ResNet50.

8 Визуализация предсказаний.

Наконец, проиллюстрируем качество работы модели на конкретных примерах из тестового набора. Для нескольких случайных изображений выполним предсказание класса с помощью обученной модели и сравним его с истинной меткой. Визуализируем эти изображения, отобразив под каждым название предсказанного и реального класса. Это наглядно демонстрирует, какие объекты распознаются верно, а где модель ошибается.

```
# Предсказание классов на тестовых изображениях
predictions = model.predict(X_test)
y_pred = np.argmax(predictions, axis=1)
y_true = y_test.flatten()

# Словарь индексов классов в названия
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

# Выбор 5 случайных изображений из тестовой выборки
indices = np.random.choice(len(X_test), 5, replace=False)
plt.figure(figsize=(10, 3))
for i, idx in enumerate(indices):
    plt.subplot(1, 5, i+1)
    plt.imshow(X_test[idx])
    plt.title(f"Pred: {class_names[y_pred[idx]]}\nTrue:
{class_names[y_true[idx]]}")
    plt.axis('off')
plt.show()
```

Рисунок 6 — Примеры результатов работы классификатора ResNet50.

Заключение

В ходе лабораторной работы были приобретены навыки построения, обучения и тестирования нейронных сетей для решения задачи классификации изображений. Использование библиотеки Keras позволило реализовать модель с высокой точностью и провести полноценный анализ результатов работы сети.В ходе лабораторной работы была реализована и обучена модель глубокого свёрточного нейронного сети ResNet50 для распознавания изображений из набора CIFAR-10.

Приложение А

[Google _Colab_Link]

```
# Импорт библиотек и загрузка данных
import numpy as np
from tensorflow import keras
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.applications import ResNet50
import matplotlib.pyplot as plt

# Загрузка датасета CIFAR-10
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
print("Размеры обучающих данных:", X_train.shape, y_train.shape)
print("Размеры тестовых данных:", X_test.shape, y_test.shape)

# Предобработка данных: нормализация пикселей и one-hot кодирование меток
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
y_train_cat = to_categorical(y_train, 10)
y_test_cat = to_categorical(y_test, 10)

# Построение модели ResNet50
model = ResNet50(weights=None, input_shape=(32, 32, 3), classes=10)
model.summary()

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

# Обучение модели
history = model.fit(X_train, y_train_cat,
                    batch_size=32,
                    epochs=10,
                    validation_split=0.2,
                    verbose=1)

# Оценка модели на тестовой выборке
test_loss, test_accuracy = model.evaluate(X_test, y_test_cat, verbose=0)
print("Точность на тестовой выборке:", round(test_accuracy, 4))

# Визуализация нескольких предсказаний на тестовых данных
predictions = model.predict(X_test)
y_pred = np.argmax(predictions, axis=1)
y_true = y_test.flatten()
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

# Вывод нескольких изображений с их предсказанными и истинными метками
```

```
import random
indices = random.sample(range(len(X_test)), 5)
plt.figure(figsize=(10, 3))
for i, idx in enumerate(indices):
    plt.subplot(1, 5, i+1)
    plt.imshow(X_test[idx])
    plt.title(f"Pred: {class_names[y_pred[idx]]}\nTrue: {class_names[y_true[idx]]}")
    plt.axis('off')
plt.show()
```