МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Факультет прикладной математики и информатики

Кафедра ТПИ

Дисциплина: «Основы теории машинного обучения»

Лабораторная работа №4, модуль «Нейронные сети глубокого обучения»

**ПЕРЕНОСИМОЕ ОБУЧЕНИЕ**

Факультет: ФПМИ

Группа: ПМИМ-31

Выполнили: Монгуш Н. С., Тарулин М. А., Филипенко Ю. Д.

Проверил: Попов А. А.

Дата выполнения:

Отметка о защите:

1. ЦЕЛЬ РАБОТЫ

Изучить основные понятия и методы переносимого обучения. Приобретение навыков перепрофилирования готовых моделей и их тонкой настройки.

1. ЗАДАНИЕ
2. Создать экземпляр базовой модели и загрузить в нее предварительно обученные веса.
3. Закрепить слои в базовой модели.
4. Создать новую модель поверх выходных данных одного (или нескольких) слоев базовой модели.
5. Обучить новую модель на новом наборе данных.
6. ИСХОДНЫЕ ДАННЫЕ

В качестве исходных данных был выбран датасет CIFAR10, который имеет 10 классов изображений. Изображения имеют размеры 32x32 пикселя, 3 цветовых канала, размер батча – 4.



Рис. 1 – Изображения из датасета CIFAR10 с указанием класса изображения.

1. ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

В качестве базовой модели была выбрана предобученная модель «RasNet50», которая используются для решения задач компьютерного зрения. Базовые слои в данной модели были закреплены.

Поверх базовых слоев была создана новая модель с 5 новыми слоями, которые изображены на рис. 2. Модель содержит следующие слои: UpSampling2D для увеличения размера входных изображений до размеров, подходящих для ResNet50; базовый слой ResNet50, предварительно обученный на ImageNet для эффективного извлечения признаков; GlobalAveragePooling2D, который преобразует матрицу признаков в вектор, упрощая обработку различных размеров входных данных; два Dense слоя, первый для классификации признаков в 10 категорий с активацией softmax и второй с функцией Dropout для предотвращения переобучения путем случайного исключения нейронов в процессе обучения, повышая тем самым обобщающую способность модели. Эта комбинация слоев позволяет использовать глубокие свойства обучения ResNet для задач классификации изображений с улучшенной стабильностью и точностью.

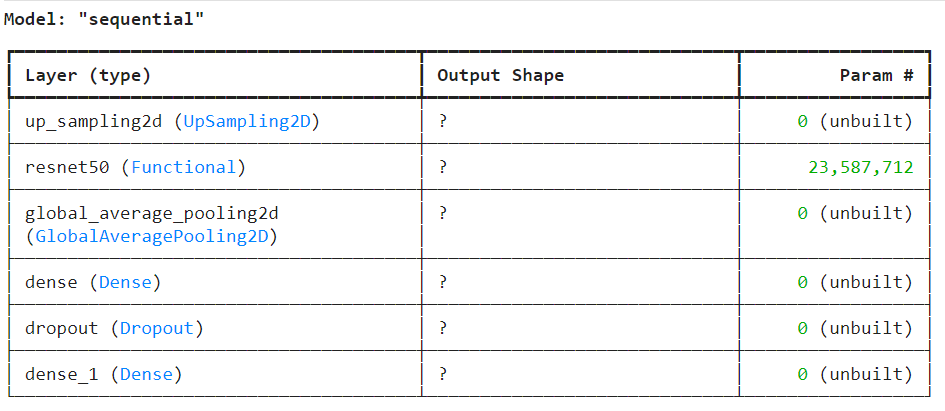


Рис. 2 – Дополнительные слои для модели ResNet50.

Далее модель обучалась на датасете CIFAR10 в течение 3 эпох, на рис. 3 изображены графики потерь и точности предсказания на обучающей и проверочной выборке соответственно.

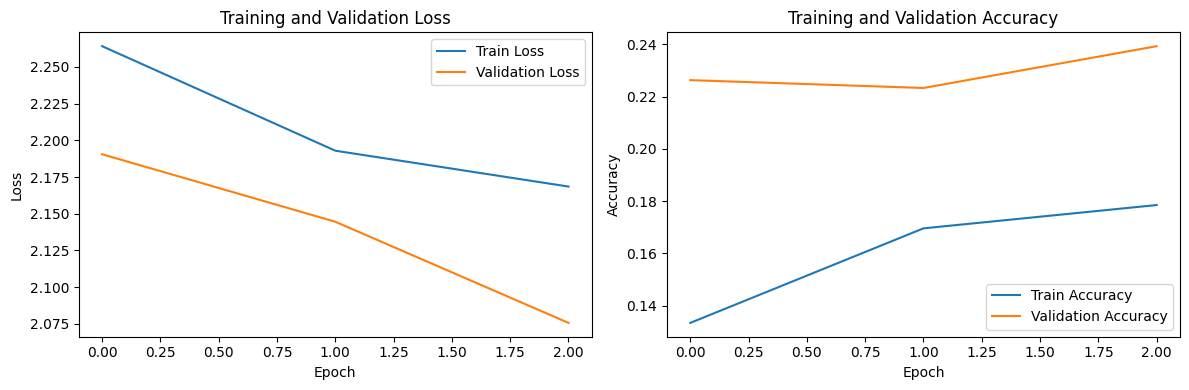


Рис. 3 – 1) График зависимости потерь на обучающей и проверочной выборке от эпохи обучения; 2) график зависимости точности предсказания на обучающей и проверочной выборке от эпохи обучения.

Результат предсказания разработанной нейронной сети представлен на рис. 4.

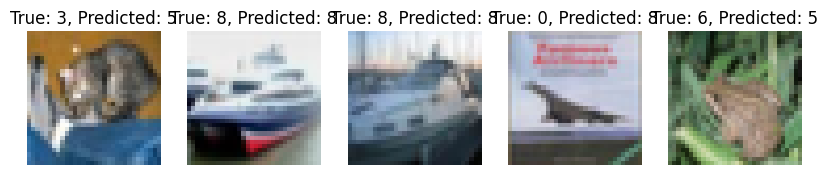


Рис. 4. Результат предсказания нейронной сети.

Из 5 изображений разработанная нейронная сеть правильно классифицировала изображения 2 раза.

1. ВЫВОД

В результате выполнения работы был создан экземпляр базовой модели «RasNet50» с предварительно загруженными весами и закрепленными слоями. Поверх этой модели было создано 5 слоев, которые специализировались под датасет CIFAR10. Данный подход ускоряет разработку архитектуры и время обучения нейронной сети.

Графики потерь и точности предсказания показывают рост в течение обучения, что говорит о правильной архитектуре нейронной сети. Результаты предсказания показывают низкую эффективность, что связано с малым количеством обучающих эпох.

1. КОД ПРОГРАММЫ

import tensorflow as tf

from tensorflow.keras.applications import ResNet50

from tensorflow.keras.datasets import cifar10

from tensorflow.keras.utils import to\_categorical

import matplotlib.pyplot as plt

import numpy as np

# Создание экземпляра базовой модели

base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

# Проверка модели

base\_model.summary()

base\_model.trainable = False

from tensorflow.keras import layers, models

# Добавление новых слоев

model = models.Sequential([

layers.UpSampling2D((7,7)), # Увеличение размера изображений с 32x32 до 224x224

base\_model,

layers.GlobalAveragePooling2D(),

layers.Dense(1024, activation='relu'),

layers.Dropout(0.5),

layers.Dense(10, activation='softmax')

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Проверка полной модели

model.summary()

(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()

x\_train = x\_train.astype('float32') / 255.0

x\_test = x\_test.astype('float32') / 255.0

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

plt.figure(figsize=(10, 2))

for i in range(5):

ax = plt.subplot(1, 5, i + 1)

plt.imshow(x\_train[i])

plt.title(f"Class: {y\_train[i].argmax()}")

plt.axis('off')

plt.show()

# Обучение модели

history = model.fit(x\_train, y\_train, epochs=3, validation\_data=(x\_test, y\_test), batch\_size=32)

# Проверка результатов обучения

model.evaluate(x\_test, y\_test)

plt.figure(figsize=(12, 4))

# График потерь

plt.subplot(1, 2, 1)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.title('Training and Validation Loss')

plt.legend()

# График точности

plt.subplot(1, 2, 2)

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.title('Training and Validation Accuracy')

plt.legend()

plt.tight\_layout()

plt.show()

# Предсказание классов для тестовых изображений

predictions = model.predict(x\_test)

# Преобразование вероятностей в индексы классов

predicted\_classes = np.argmax(predictions, axis=1)

# Функция для отображения изображений с их предсказанными классами

def plot\_images(images, labels, predictions, num\_images=5):

plt.figure(figsize=(10, 8))

for i in range(num\_images):

plt.subplot(1, num\_images, i + 1)

plt.imshow(images[i])

plt.title(f"True: {labels[i]}, Predicted: {predictions[i]}")

plt.axis('off')

plt.show()

# Визуализация предсказаний для нескольких изображений

plot\_images(x\_test, np.argmax(y\_test, axis=1), predicted\_classes)