Министерство цифрового развития

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт

по лабораторной работе № 5 «Нейронные сети для обработки изображений»

Выполнил:

студент группы ИП-312

Дорогин Н.С.

Работу проверил: старший преподаватель кафедры ПМиК Дементьева К.И.

Новосибирск 2025 г.

Введение (задание)

Цель: освоить на практике принципы построения, обучения и оценки нейронных сетей для решения базовых задач компьютерного зрения.

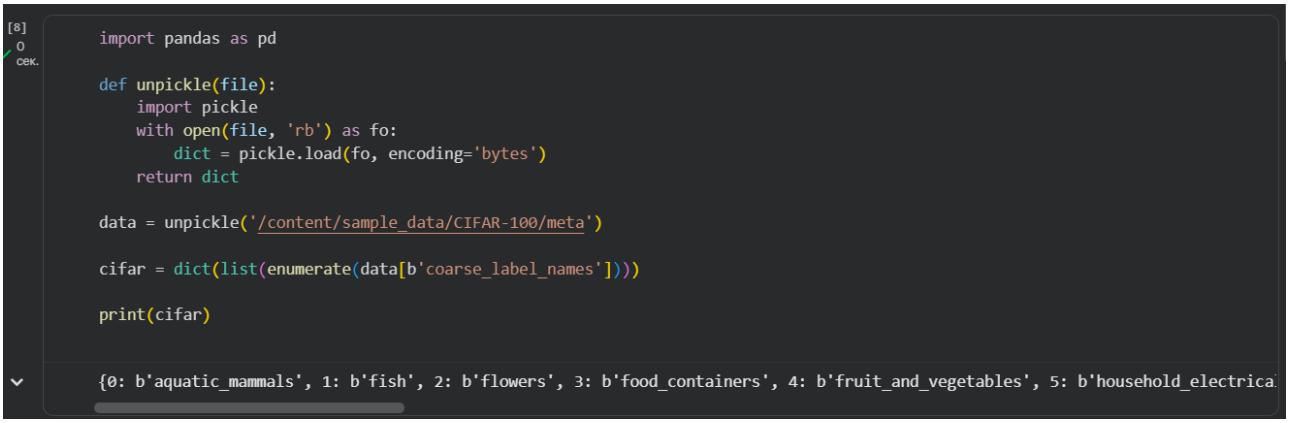
Основная часть:

1. Распределение вариантов

Формула: (Номер студента в списке) mod 6 + 1

В моём случае: 8 mod 6 + 1 = 3

Вариант 3: CIFAR-100 - Расширенная версия CIFAR-10 (100 классов)



```
[8]
0
сек.
```

```python
import pandas as pd

def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict

data = unpickle('/content/sample_data/CIFAR-100/meta')

cifar = dict(list(enumerate(data[b'coarse_label_names'])))

print(cifar)
```

```
{0: b'aquatic_mammals', 1: b'fish', 2: b'flowers', 3: b'food_containers', 4: b'fruit_and_vegetables', 5: b'household_electrica
```

2. Создание архитектуры нейронной сети

Вариант архитектуры выбирается по формуле
(Номер студента в списке) mod 8 + 1

В моём случае: 8 mod 8 + 1 = 1

Вариант	Архитектура CNN	Регуляризация	Оптимизатор
1	2 сверточных слоя + 2 полносвязных	Dropout	Adam

```
#-------------------------------------------------------------------------------
# МОДЕЛЬ
#-------------------------------------------------------------------------------
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Conv2D(
        filters=600,
        kernel_size=(3,3),
        activation='relu',
        input_shape=(32,32,3)
    ),
    layers.Conv2D(
        filters=400,
        kernel_size=(3,3),
        activation='relu',
    ),
    layers.Flatten(),
    layers.Dense(200, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(100, activation='relu'),
    layers.Dense(20, activation='softmax'),
])
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
#-------------------------------------------------------------------------------
```

3. Анализ модели

```
#-------------------------------------------------------------------------------
# ДАННЫЕ
#-------------------------------------------------------------------------------
import pandas as pd

def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict

data = unpickle('/content/sample_data/CIFAR-100/meta')

cifar = dict(list(enumerate(data[b'coarse_label_names'])))

data_pre_path = '/content/sample_data/CIFAR-100/'
data_train_path = data_pre_path + 'train'
data_test_path = data_pre_path + 'test'

data_train_dict = unpickle(data_train_path)
data_test_dict = unpickle(data_test_path)
```

```
import numpy as np
# Данные для обучения
data_train = data_train_dict[b'data']
label_train = np.array(data_train_dict[b'coarse_labels'])

# Тестовые данные
data_test = data_test_dict[b'data']
label_test = np.array(data_test_dict[b'coarse_labels'])

data_train = data_train.reshape(-1, 3, 32, 32).transpose(0, 2, 3, 1) / 255.0
data_test = data_test.reshape(-1, 3, 32, 32).transpose(0, 2, 3, 1) / 255.0
```

```python
#-----------------------------------------------------------------------------------------
# ОБУЧЕНИЕ
#-----------------------------------------------------------------------------------------
history = model.fit(data_train, label_train, epochs=10, batch_size=600, validation_split=0.3, verbose=1)
print(model.summary())
```

```
Epoch 1/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 171s 1s/step - accuracy: 0.0842 - loss: 3.6862 - val_accuracy: 0.1890 - val_loss: 2.6492
Epoch 2/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 44s 745ms/step - accuracy: 0.1990 - loss: 2.5960 - val_accuracy: 0.2880 - val_loss: 2.3360
Epoch 3/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 44s 745ms/step - accuracy: 0.2687 - loss: 2.3830 - val_accuracy: 0.3151 - val_loss: 2.2194
Epoch 4/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 44s 747ms/step - accuracy: 0.3088 - loss: 2.2350 - val_accuracy: 0.3571 - val_loss: 2.0883
Epoch 5/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 44s 749ms/step - accuracy: 0.3450 - loss: 2.1093 - val_accuracy: 0.3681 - val_loss: 2.0518
Epoch 6/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 44s 748ms/step - accuracy: 0.3666 - loss: 2.0402 - val_accuracy: 0.3889 - val_loss: 1.9924
Epoch 7/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 44s 749ms/step - accuracy: 0.3928 - loss: 1.9541 - val_accuracy: 0.4003 - val_loss: 1.9595
Epoch 8/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 44s 748ms/step - accuracy: 0.4176 - loss: 1.8744 - val_accuracy: 0.4039 - val_loss: 1.9460
Epoch 9/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 44s 748ms/step - accuracy: 0.4432 - loss: 1.7815 - val_accuracy: 0.4138 - val_loss: 1.9182
Epoch 10/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 44s 748ms/step - accuracy: 0.4709 - loss: 1.6824 - val_accuracy: 0.4089 - val_loss: 1.9517
```

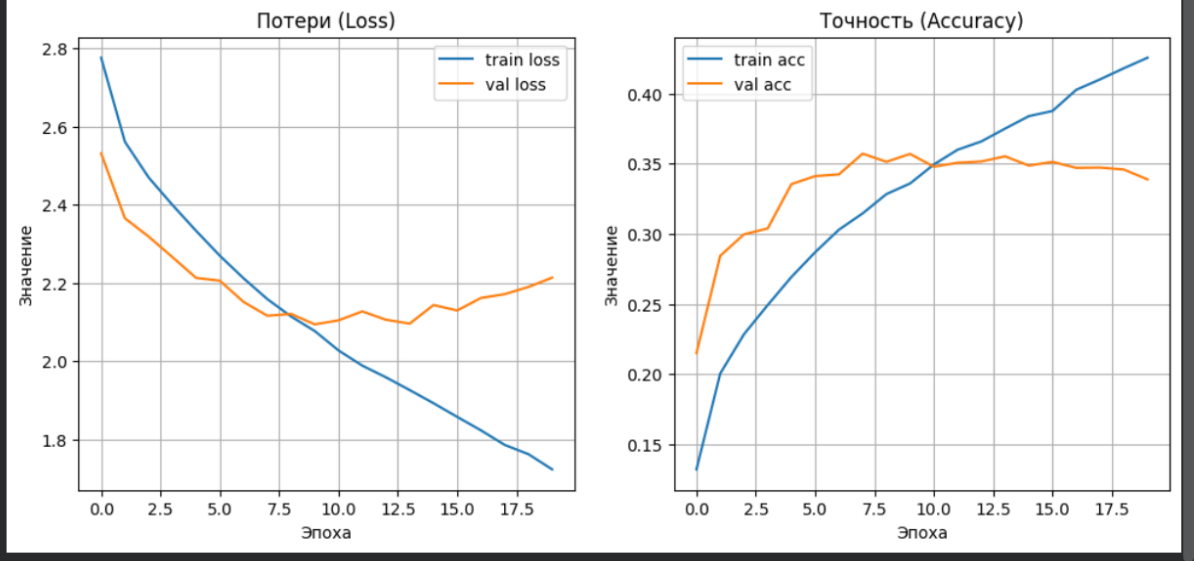
```
Model: "sequential_1"
```

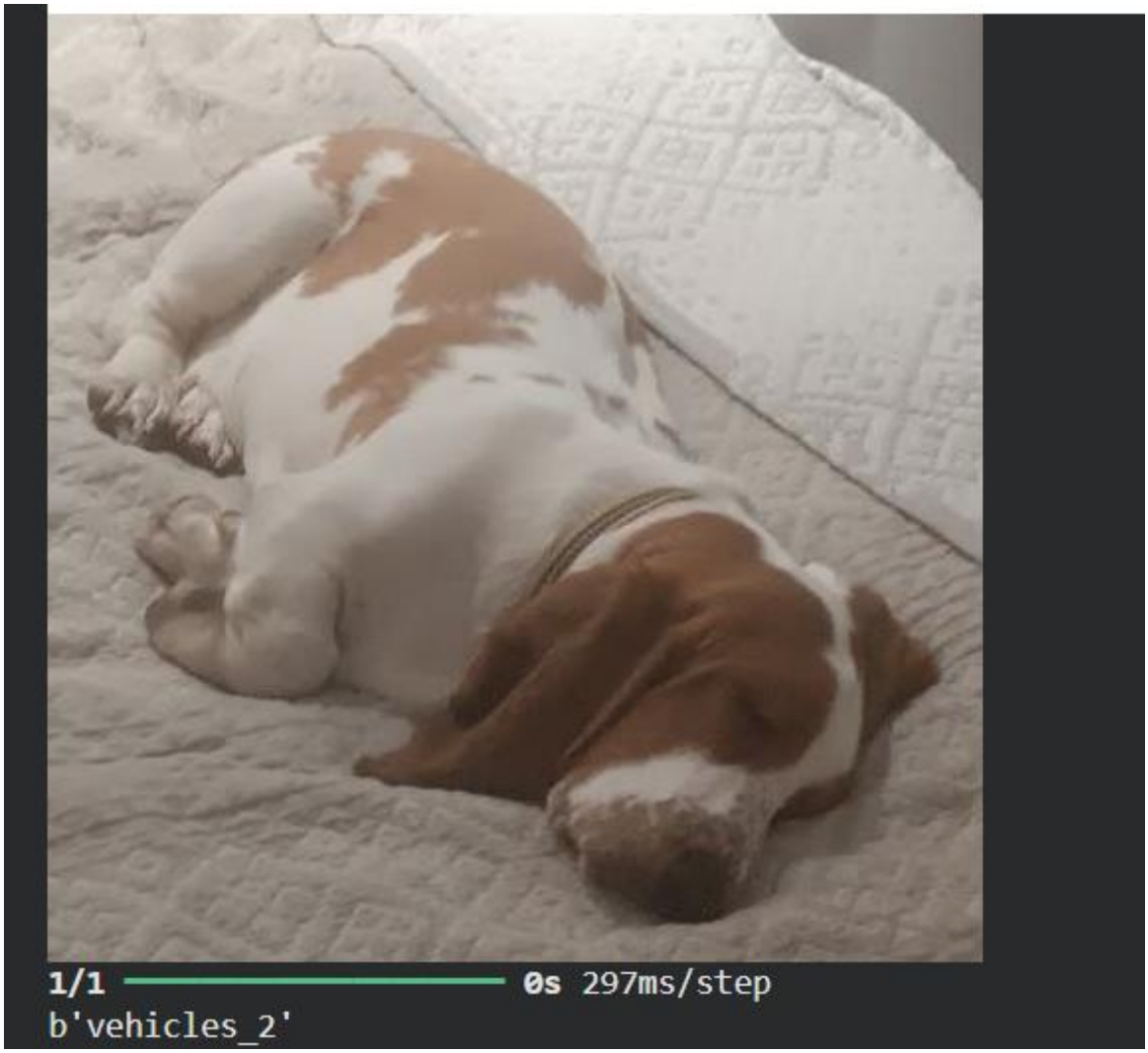
Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 30, 30, 600)	16,800
conv2d_3 (Conv2D)	(None, 28, 28, 400)	2,160,400
flatten_1 (Flatten)	(None, 313600)	0
dense_3 (Dense)	(None, 200)	62,720,200
dropout_1 (Dropout)	(None, 200)	0
dense_4 (Dense)	(None, 100)	20,100
dense_5 (Dense)	(None, 20)	2,020

```
Total params: 194,758,562 (742.94 MB)
Trainable params: 64,919,520 (247.65 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 129,839,042 (495.30 MB)
None
313/313 ━━━━━━━━━━━━━━━━━━━━ 7s 21ms/step
```

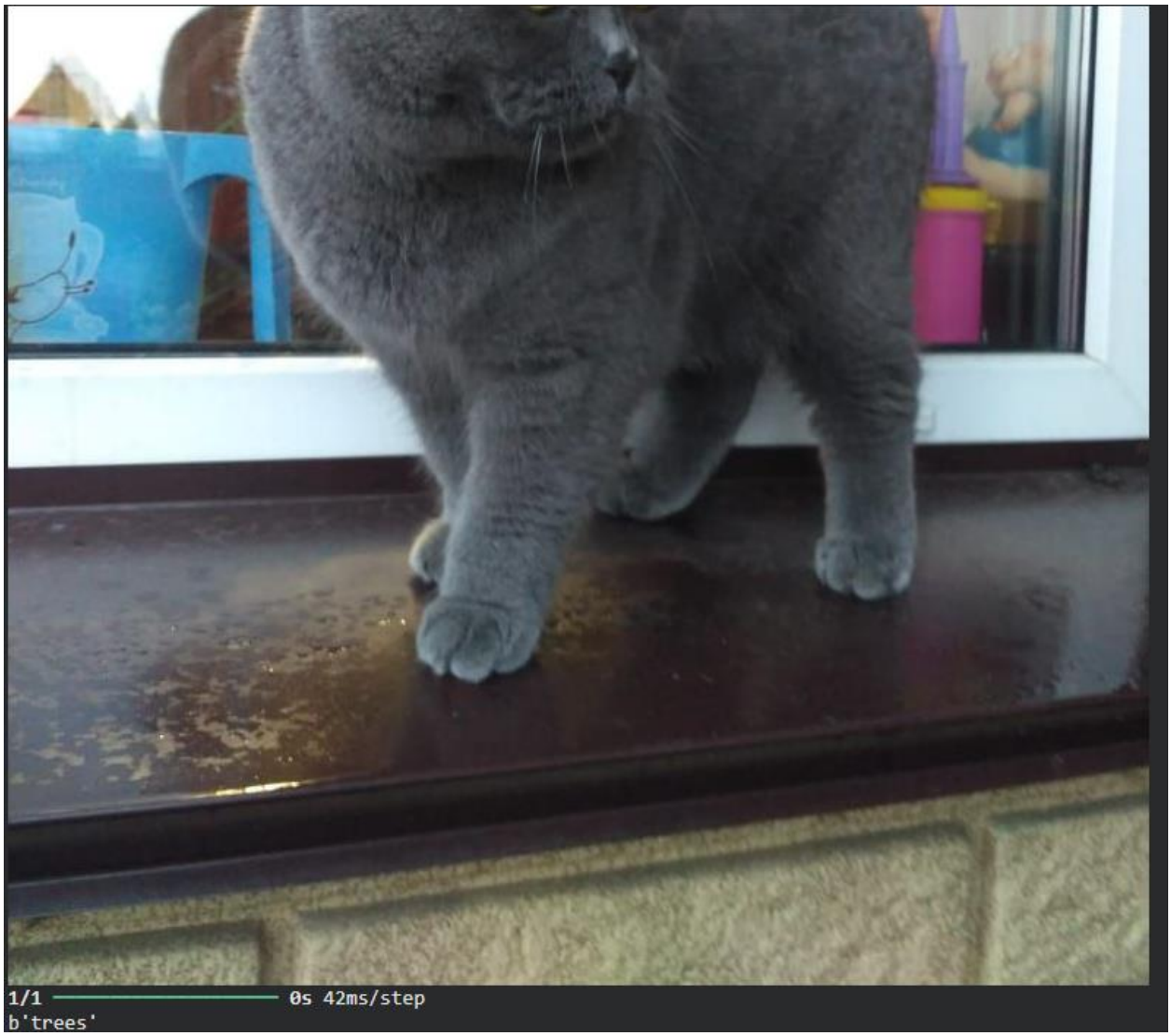
Результаты:

```
Точность на тестовой выборке: 0.3422
Потери на тестовой выборке: 2.2034
```

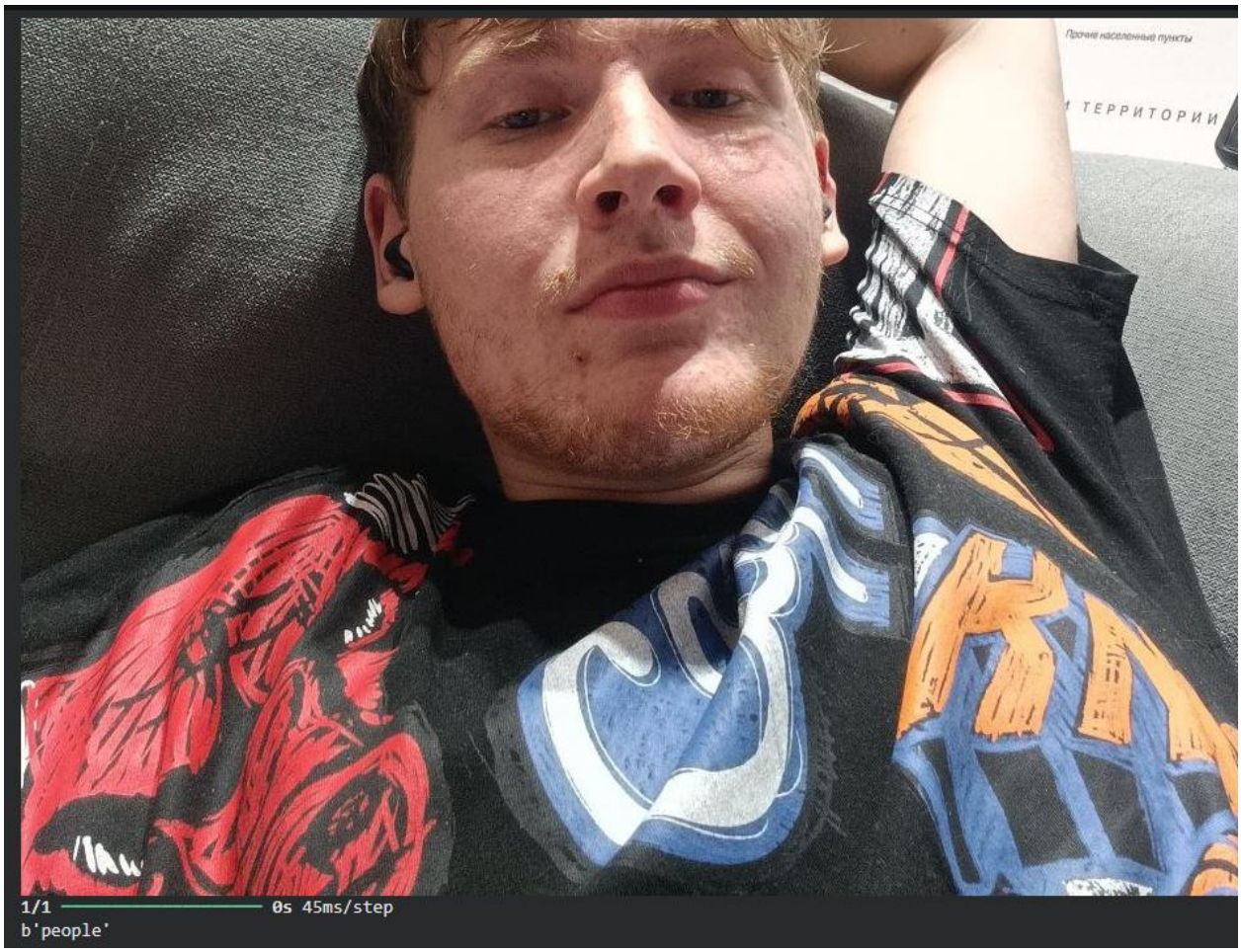




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 297ms/step
b'vehicles_2'
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
b'trees'
```



5. Отчёт

Ссылка на GoogleCollab:

https://colab.research.google.com/drive/1InYZSFYOxjGShodyJXLZvcIH-hd156SS#scrollTo=_5