Лабораторная работа №2

Выполнил студент группы ББМО-01-23
Буланов Андрей Алексеевич

```
import kagglehub

path = kagglehub.dataset_download("meowmeowmeowmeowmeow/gtsrb-german-traffic-sign")
print(path)
```

```
Warning: Looks like you're using an outdated `kagglehub` version (installed: 0.3.5), please consider upgrading to the latest version (0.3.6).
Downloading from https://www.kaggle.com/api/v1/datasets/download/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign?dataset_version_number=1...

100%|██████████| 612M/612M [00:29<00:00, 21.7MB/s]

Extracting files...



/root/.cache/kagglehub/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign/versions/1
```

```
!mv /root/.cache/kagglehub/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign/versions/ /content/gtsrb
```

```
import torch
device = torch.device("cuda")
```

```
import tensorflow as tf
import numpy as np
import os
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

# Загрузка и подготовка набора данных
def load_gtsrb_data(data_dir):
    images = []
    labels = []
    for label in range(43):
        class_dir = os.path.join(data_dir, str(label))
```

```
        if not os.path.exists(class_dir):
            print(f"Папка {class_dir} не найдена. Пропускаем...")
            continue
        for file in os.listdir(class_dir):
            img_path = os.path.join(class_dir, file)
            img = tf.keras.preprocessing.image.load_img(img_path,
target_size=(32, 32))
            img_array = tf.keras.preprocessing.image.img_to_array(img)
/ 255.0
            images.append(img_array)
            labels.append(label)
    return np.array(images), np.array(labels)

data_dir = "/content/gtsrb/1/train"

# Загрузка данных
images, labels = load_gtsrb_data(data_dir)

# Разделение на обучающую, валидационную и тестовую выборки
X_train, X_temp, y_train, y_temp = train_test_split(images, labels,
test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42)

# One-hot encoding меток классов
y_train = to_categorical(y_train, num_classes=43)
y_val = to_categorical(y_val, num_classes=43)
y_test = to_categorical(y_test, num_classes=43)

print(f"Размеры: X_train={X_train.shape}, X_val={X_val.shape},
X_test={X_test.shape}")
```

```
Размеры: X_train=(27446, 32, 32, 3), X_val=(5881, 32, 32, 3),
X_test=(5882, 32, 32, 3)
```

```
# Для уменьшения времени работы проведем обучение лишь на ограниченной
выборке из 5000 штук
small_X_train = X_train[:5000]
small_y_train = y_train[:5000]

from tensorflow.keras.applications import VGG16
from tensorflow.keras import Model, layers, optimizers

# Загрузка VGG16 с предобученными весами
base_model_vgg = VGG16(weights="imagenet", include_top=False,
input_shape=(32, 32, 3))
x = layers.Flatten()(base_model_vgg.output)
x = layers.Dense(128, activation="relu")(x)
output = layers.Dense(43, activation="softmax")(x)

model_vgg = Model(inputs=base_model_vgg.input, outputs=output)
```

```python
# Компиляция модели
model_vgg.compile(
    optimizer=optimizers.Adam(learning_rate=0.001),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
)

# Обучение
history_vgg = model_vgg.fit(
    small_X_train, small_y_train,
    validation_data=(X_val, y_val),
    epochs=5,
    batch_size=32
)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
Epoch 1/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 26s 101ms/step - accuracy: 0.0593 - loss:
3.5780 - val_accuracy: 0.1058 - val_loss: 3.2752
Epoch 2/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 6s 41ms/step - accuracy: 0.1266 - loss:
3.0922 - val_accuracy: 0.2168 - val_loss: 2.6006
Epoch 3/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 10s 42ms/step - accuracy: 0.2085 - loss:
2.6143 - val_accuracy: 0.2683 - val_loss: 2.3068
Epoch 4/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 7s 42ms/step - accuracy: 0.2915 - loss:
2.1795 - val_accuracy: 0.3855 - val_loss: 1.9001
Epoch 5/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 7s 43ms/step - accuracy: 0.4143 - loss:
1.7127 - val_accuracy: 0.5094 - val_loss: 1.4087
```

```python
from tensorflow.keras.applications import ResNet50

# Загрузка ResNet50 с предобученными весами
base_model_resnet = ResNet50(weights="imagenet", include_top=False,
input_shape=(32, 32, 3))
x = layers.Flatten()(base_model_resnet.output)
x = layers.Dense(128, activation="relu")(x)
output = layers.Dense(43, activation="softmax")(x)

model_resnet = Model(inputs=base_model_resnet.input, outputs=output)

# Компиляция модели
model_resnet.compile(
    optimizer=optimizers.Adam(learning_rate=0.001),
    loss="categorical_crossentropy",
```

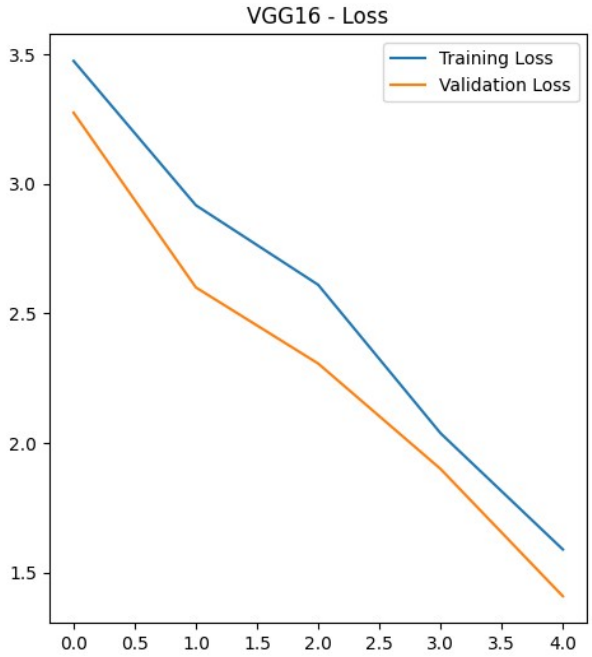
```
    metrics=["accuracy"]
)

# Обучение
history_resnet = model_resnet.fit(
    small_X_train, small_y_train,
    validation_data=(X_val, y_val),
    epochs=5,
    batch_size=32
)
```

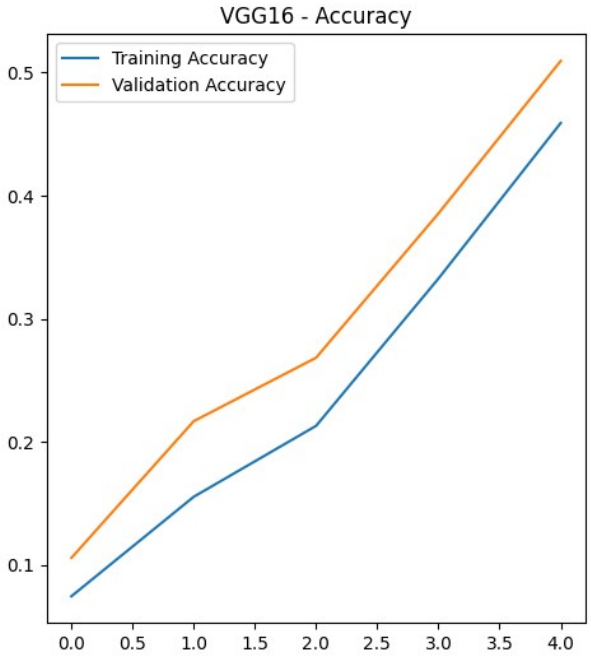
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/
resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 5s 0us/step
Epoch 1/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 90s 215ms/step - accuracy: 0.2654 - loss:
2.9799 - val_accuracy: 0.0536 - val_loss: 10.2140
Epoch 2/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 30s 40ms/step - accuracy: 0.6461 - loss:
1.2155 - val_accuracy: 0.0377 - val_loss: 5.2024
Epoch 3/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 6s 37ms/step - accuracy: 0.7771 - loss:
0.7459 - val_accuracy: 0.1585 - val_loss: 3.1703
Epoch 4/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 10s 37ms/step - accuracy: 0.8293 - loss:
0.6434 - val_accuracy: 0.3685 - val_loss: 2.9753
Epoch 5/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 7s 45ms/step - accuracy: 0.8887 - loss:
0.3878 - val_accuracy: 0.4052 - val_loss: 3.4946
```

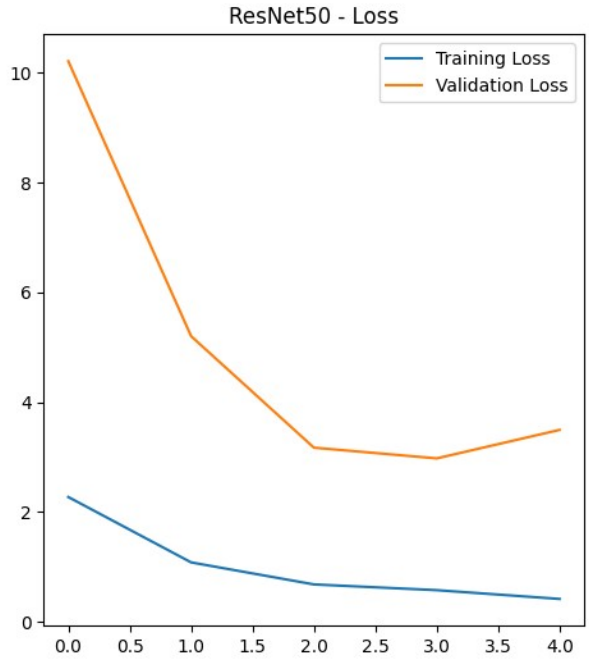
```
# Оценка моделей
vgg_eval = model_vgg.evaluate(X_test, y_test, verbose=0)
resnet_eval = model_resnet.evaluate(X_test, y_test, verbose=0)

print(f"VGG16 - Тестовая точность: {vgg_eval[1]:.2f}, Тестовая потеря:
{vgg_eval[0]:.2f}")
print(f"ResNet50 - Тестовая точность: {resnet_eval[1]:.2f}, Тестовая
потеря: {resnet_eval[0]:.2f}")

# Построение графиков функции потерь и точности
def plot_history(history, title):
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.plot(history.history["loss"], label="Training Loss")
    plt.plot(history.history["val_loss"], label="Validation Loss")
    plt.title(f"{title} - Loss")
    plt.legend()

    plt.subplot(1, 2, 2)
```

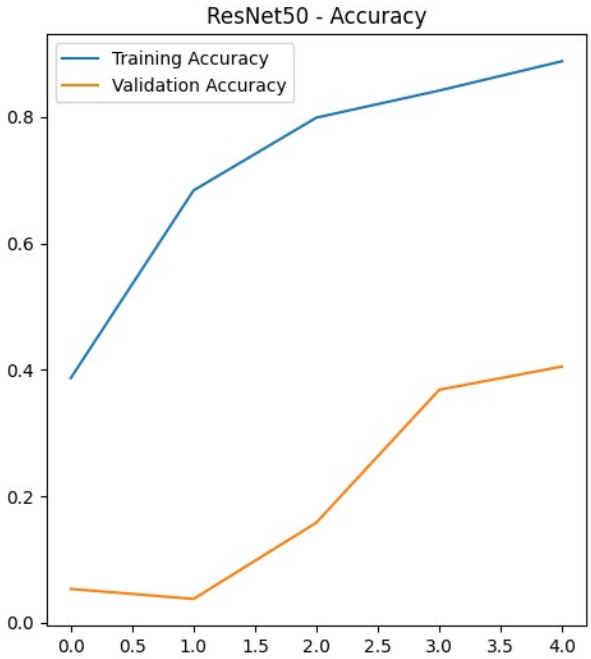
```
    plt.plot(history.history["accuracy"], label="Training Accuracy")
    plt.plot(history.history["val_accuracy"], label="Validation
Accuracy")
    plt.title(f"{title} - Accuracy")
    plt.legend()
    plt.show()

plot_history(history_vgg, "VGG16")
plot_history(history_resnet, "ResNet50")
```

```
VGG16 - Тестовая точность: 0.51, Тестовая потеря: 1.42
ResNet50 - Тестовая точность: 0.40, Тестовая потеря: 3.57
```



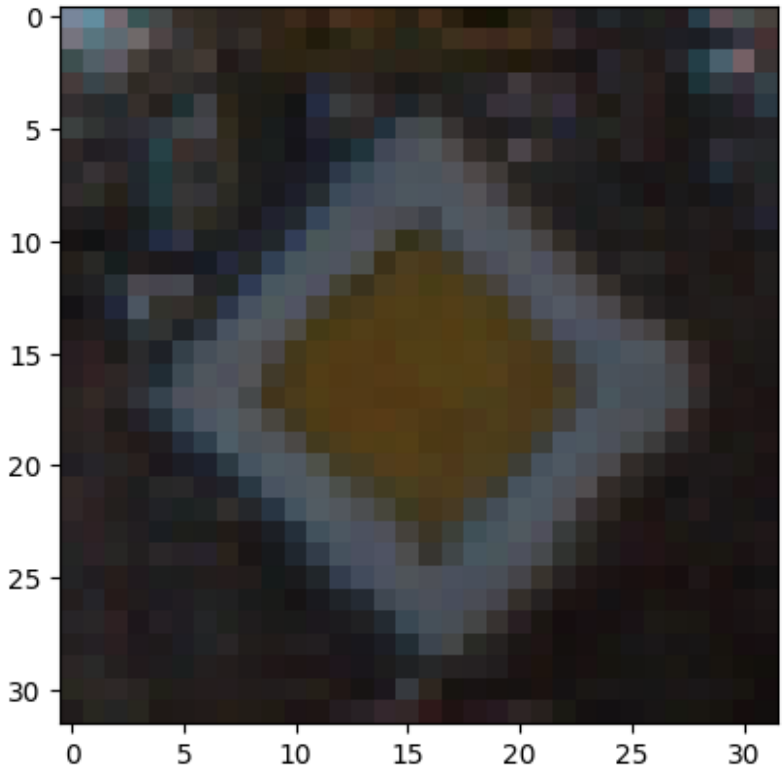







```
def fgsm_attack(model, image, label, epsilon=0.01):
    image = tf.convert_to_tensor([image], dtype=tf.float32)
    label = tf.convert_to_tensor([label], dtype=tf.float32)

    with tf.GradientTape() as tape:
        tape.watch(image)
        prediction = model(image)
        loss = tf.keras.losses.categorical_crossentropy(label,
prediction)

    gradient = tape.gradient(loss, image)
    perturbation = epsilon * tf.sign(gradient)
    adversarial_image = tf.clip_by_value(image + perturbation, 0, 1)
    return adversarial_image[0]

# Пример атаки на одно изображение
adv_image = fgsm_attack(model_vgg, X_test[0], y_test[0])
plt.imshow(adv_image.numpy())
```

```
<matplotlib.image.AxesImage at 0x7a1c29db5960>
```




```
# Проверка эффективности FGSM-атак на ResNet50
for i in range(10):  # Пример из 10 изображений
    adv_image = fgsm_attack(model_vgg, X_test[i], y_test[i])
    pred_resnet =
np.argmax(model_resnet.predict(tf.convert_to_tensor([adv_image])))
    true_label = np.argmax(y_test[i])
    print(f"Истинный класс: {true_label}, Предсказание ResNet50:
{pred_resnet}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 5s 5s/step
Истинный класс: 12, Предсказание ResNet50: 32
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
Истинный класс: 10, Предсказание ResNet50: 13
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
Истинный класс: 10, Предсказание ResNet50: 10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
Истинный класс: 25, Предсказание ResNet50: 26
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
Истинный класс: 3, Предсказание ResNet50: 13
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
Истинный класс: 28, Предсказание ResNet50: 24
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
Истинный класс: 33, Предсказание ResNet50: 33
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
Истинный класс: 1, Предсказание ResNet50: 4
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
Истинный класс: 30, Предсказание ResNet50: 13
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
Истинный класс: 11, Предсказание ResNet50: 11
```