Лабораторная работа № 3

Обучение нейросетевых моделей классификации изображений

Боровских Вадим, 932003

Построить нейросетевые модели – бинарный классификатор

Ввод [19]:
```
import os
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import RMSprop
from tensorflow.python.keras.layers import Activation, Dropout, Flatten, Dense
```

Набор данных изображений больных листьев перца и здоровых листьев котрофеля.

Ввод [28]:
```
# Гиперпараметры
IMG_HEIGHT = IMG_WIDTH = 150
IMG_CHANNEL = 3
EPOCHS = 30
BATCH_SIZE = 16
TRAIN_SAMPLES = 2475
VAL_SAMPLES = 1695
TEST_SAMPLES = 2045
```

Ввод [29]:
```
# Каталог с данными для обучения
train_dir = 'train'
# Каталог с данными для проверки
validation_dir = 'val'
# Каталог с данными для тестирования
test_dir = 'test'
```

Ввод [30]:
```
# Генераторы изображений
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

test_and_val_datagen = ImageDataGenerator(rescale=1./255)
```

Ввод [31]:
```
# Обучающая, тестовая и валидационная выборки
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='binary')

validation_generator = test_and_val_datagen.flow_from_directory(
    validation_dir,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='binary')

test_generator = test_and_val_datagen.flow_from_directory(
    test_dir,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='binary')
```

```
Found 2475 images belonging to 2 classes.
Found 1695 images belonging to 2 classes.
Found 2045 images belonging to 2 classes.
```

Ввод [32]:
```
# Модель
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(IMG_HE
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Dropout(0.2),  # Dropout
    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Dropout(0.2),  # Dropout
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Архитектура нейросети:

Слои свертки:

- Первый сверточный слой с 32 фильтрами, функцией активации ReLU и размером ядра 3x3.
- Слой максимальной пулинга (MaxPooling) с размером окна 2x2.
- Слой Dropout с коэффициентом 0.2.
- Второй сверточный слой с 64 фильтрами, функцией активации ReLU и размером ядра 3x3.
- Слой максимальной пулинга (MaxPooling) с размером окна 2x2.
- Третий сверточный слой с 128 фильтрами, функцией активации ReLU и размером ядра 3x3.
- Слой максимальной пулинга (MaxPooling) с размером окна 2x2.
- Четвертый сверточный слой с 128 фильтрами, функцией активации ReLU и размером ядра 3x3.
- Слой максимальной пулинга (MaxPooling) с размером окна 2x2.
- Слой Dropout с коэффициентом 0.2.

Полносвязные слои:

- Слой вытягивания (Flatten) для преобразования выходных данных сверточных слоев в одномерный вектор.
- Полносвязный слой с 512 нейронами и функцией активации ReLU.
- Выходной полносвязный слой с 1 нейроном и функцией активации sigmoid (для бинарной классификации).

Ввод [33]:
```
# Компиляция
model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(lr=1e-4),
              metrics=['acc'])
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the l
egacy optimizer, e.g.,tf.keras.optimizers.legacy.RMSprop.
```

Ввод [34]:
```
# Обучение
history = model.fit(
    train_generator,
    steps_per_epoch=TRAIN_SAMPLES // BATCH_SIZE,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=VAL_SAMPLES // BATCH_SIZE)
```

```
Epoch 1/30
154/154 [==============================] - 143s 908ms/step - loss: 0.6843 - acc: 0.6287 - v
al_loss: 0.6670 - val_acc: 0.5619
Epoch 2/30
154/154 [==============================] - 139s 902ms/step - loss: 0.3994 - acc: 0.8573 - v
al_loss: 0.2345 - val_acc: 0.9238
Epoch 3/30
154/154 [==============================] - 138s 897ms/step - loss: 0.2657 - acc: 0.9170 - v
al_loss: 0.1891 - val_acc: 0.9804
Epoch 4/30
154/154 [==============================] - 139s 901ms/step - loss: 0.1785 - acc: 0.9516 - v
al_loss: 0.1439 - val_acc: 0.9690
Epoch 5/30
154/154 [==============================] - 139s 899ms/step - loss: 0.1629 - acc: 0.9658 - v
al_loss: 0.0404 - val_acc: 0.9940
Epoch 6/30
154/154 [==============================] - 139s 901ms/step - loss: 0.0930 - acc: 0.9764 - v
al_loss: 0.0356 - val_acc: 0.9946
Epoch 7/30
154/154 [==============================] - 138s 898ms/step - loss: 0.0813 - acc: 0.9780 - v
al_loss: 0.0759 - val_acc: 0.9827
Epoch 8/30
154/154 [==============================] - 140s 905ms/step - loss: 0.0968 - acc: 0.9740 - v
al_loss: 0.0680 - val_acc: 0.9786
Epoch 9/30
154/154 [==============================] - 138s 897ms/step - loss: 0.0613 - acc: 0.9760 - v
al_loss: 0.0458 - val_acc: 0.9899
Epoch 10/30
154/154 [==============================] - 139s 900ms/step - loss: 0.0672 - acc: 0.9870 - v
al_loss: 0.0328 - val_acc: 0.9952
Epoch 11/30
154/154 [==============================] - 139s 903ms/step - loss: 0.0802 - acc: 0.9784 - v
al_loss: 0.0156 - val_acc: 0.9964
Epoch 12/30
154/154 [==============================] - 141s 917ms/step - loss: 0.0610 - acc: 0.9862 - v
al_loss: 0.0160 - val_acc: 0.9964
Epoch 13/30
154/154 [==============================] - 177s 1s/step - loss: 0.0481 - acc: 0.9854 - val_
loss: 0.0132 - val_acc: 0.9964
Epoch 14/30
154/154 [==============================] - 180s 1s/step - loss: 0.0563 - acc: 0.9829 - val_
loss: 0.0338 - val_acc: 0.9946
Epoch 15/30
154/154 [==============================] - 164s 1s/step - loss: 0.0484 - acc: 0.9850 - val_
loss: 0.0290 - val_acc: 0.9917
Epoch 16/30
154/154 [==============================] - 145s 940ms/step - loss: 0.0493 - acc: 0.9878 - v
al_loss: 0.0193 - val_acc: 0.9935
Epoch 17/30
154/154 [==============================] - 141s 915ms/step - loss: 0.0472 - acc: 0.9915 - v
al_loss: 0.0417 - val_acc: 0.9875
Epoch 18/30
154/154 [==============================] - 142s 921ms/step - loss: 0.0465 - acc: 0.9854 - v
al_loss: 0.0245 - val_acc: 0.9923
Epoch 19/30
154/154 [==============================] - 141s 913ms/step - loss: 0.0618 - acc: 0.9890 - v
al_loss: 0.0107 - val_acc: 0.9958
Epoch 20/30
154/154 [==============================] - 141s 916ms/step - loss: 0.0410 - acc: 0.9878 - v
al_loss: 0.0407 - val_acc: 0.9905
Epoch 21/30
154/154 [==============================] - 141s 913ms/step - loss: 0.0427 - acc: 0.9894 - v
al_loss: 0.0076 - val_acc: 0.9964
Epoch 22/30
154/154 [==============================] - 141s 917ms/step - loss: 0.0481 - acc: 0.9866 - v
al_loss: 0.0157 - val_acc: 0.9940
Epoch 23/30
154/154 [==============================] - 141s 914ms/step - loss: 0.0413 - acc: 0.9882 - v
al_loss: 0.0014 - val_acc: 1.0000
Epoch 24/30
```

```
154/154 [==============================] - 141s 917ms/step - loss: 0.0512 - acc: 0.9845 - v
al_loss: 0.0041 - val_acc: 0.9988
Epoch 25/30
154/154 [==============================] - 141s 914ms/step - loss: 0.0366 - acc: 0.9902 - v
al_loss: 0.0288 - val_acc: 0.9893
Epoch 26/30
154/154 [==============================] - 141s 913ms/step - loss: 0.0399 - acc: 0.9870 - v
al_loss: 0.0254 - val_acc: 0.9887
Epoch 27/30
154/154 [==============================] - 141s 916ms/step - loss: 0.0394 - acc: 0.9874 - v
al_loss: 0.0099 - val_acc: 0.9952
Epoch 28/30
154/154 [==============================] - 141s 916ms/step - loss: 0.0549 - acc: 0.9874 - v
al_loss: 0.0108 - val_acc: 0.9982
Epoch 29/30
154/154 [==============================] - 141s 916ms/step - loss: 0.0435 - acc: 0.9878 - v
al_loss: 0.0213 - val_acc: 0.9982
Epoch 30/30
154/154 [==============================] - 141s 915ms/step - loss: 0.0361 - acc: 0.9894 - v
al_loss: 0.1178 - val_acc: 0.9702
```

Ввод [35]:

```
# Оценка модели на тестировании
score = model.evaluate(test_generator, steps=TEST_SAMPLES // BATCH_SIZE)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
127/127 [==============================] - 36s 280ms/step - loss: 0.1076 - acc: 0.9719
Test loss: 0.10762838274240494
Test accuracy: 0.9719488024711609
```

Результаты обучения:

- Потери (loss): 0.1076
- Точность (accuracy): 0.9719

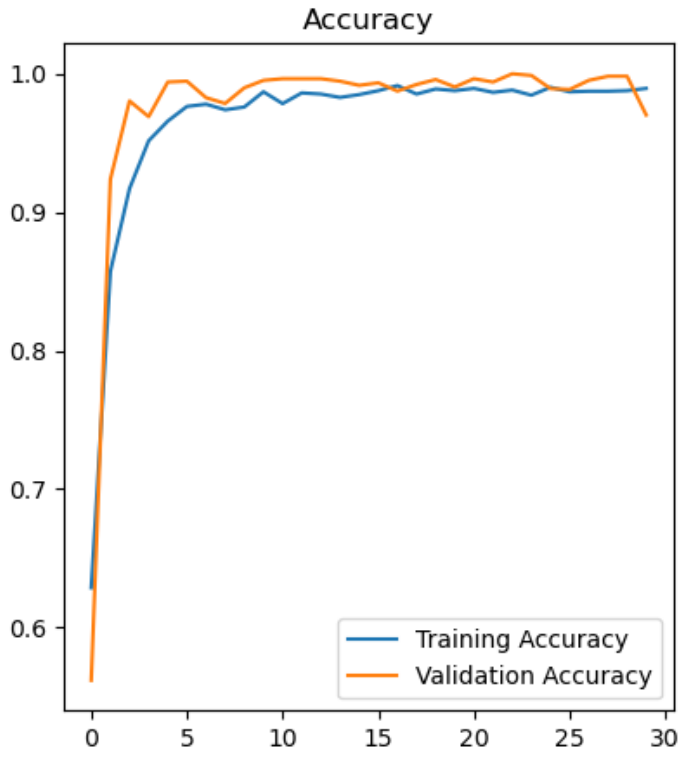
Значение потерь (loss) составляет 0.1076, что свидетельствует о том, что модель хорошо обобщает данные и имеет небольшие ошибки в предсказаниях.

Точность (accuracy) модели на валидационной выборке составляет 0.9719, что означает, что модель правильно классифицирует объекты с точностью более 97%, что является высоким показателем.

Выводы:Данные результаты свидетельствуют о том, что модель успешно обучена и продемонстрировала высокую точность как на валидационной, так и на тестовой выборках.

Ввод [37]:
```
# Создаем график точности
plt.figure(figsize=(10,5))
plt.subplot(1, 2, 1)
plt.plot(history.history['acc'], label='Training Accuracy')
plt.plot(history.history['val_acc'], label='Validation Accuracy')
plt.title('Accuracy')
plt.legend()
```

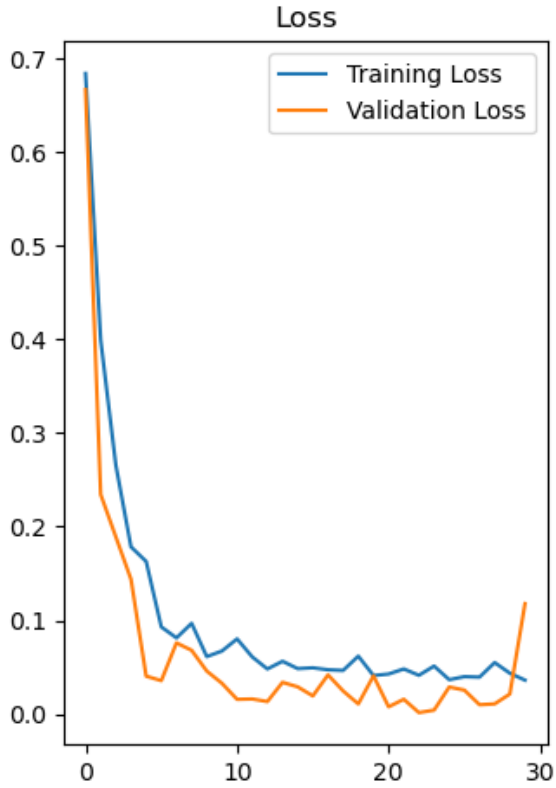
Out[37]:
```
<matplotlib.legend.Legend at 0x24a810a6e30>
```



Вывод: На протяжении всего процесса обучения, точность (accuracy) как на обучающей, так и на валидационной выборке улучшается, что свидетельствует о том, что модель успешно учится и демонстрирует хорошие предсказательные способности. Хотя в конце она незначительно падает, результат остается высоким ~0.9-1.

Ввод [38]:
```
# Создаем график потерь
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```



По мере увеличения количества эпох обучения, значения функции потерь (loss) как на обучающей выборке, так и на валидационной выборке уменьшаются с увеличением точности (accuracy). Это указывает на то, что модель успешно обучается и постепенно улучшает свои предсказательные способности.

Ввод [47]:
```
# Сохранение модели
model.save('model.h5')
```

Ввод [48]:
```
# Загрузка модели
from tensorflow.keras.models import load_model
model = load_model('model.h5')

# Обучающая, тестовая и валидационная выборки
train_generator = train_datagen.flow_from_directory(
    'new_train_dir', # Новый каталог с данными для обучения
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='binary')

validation_generator = test_and_val_datagen.flow_from_directory(
    'new_validation_dir', # Новый каталог с данными для проверки
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='binary')

test_generator = test_and_val_datagen.flow_from_directory(
    'new_test_dir', # Новый каталог с данными для тестирования
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='binary')

NEW_TRAIN_SAMPLES = 2475
NEW_VAL_SAMPLES = 2255
NEW_TEST_SAMPLES = 2255
```

```
Found 2475 images belonging to 2 classes.
Found 2255 images belonging to 2 classes.
Found 2255 images belonging to 2 classes.
```

Ввод [49]:
```
# Повторное обучение
history = model.fit(
    train_generator,
    steps_per_epoch=NEW_TRAIN_SAMPLES // BATCH_SIZE, # Новое количество обучающих образцов
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=NEW_VAL_SAMPLES // BATCH_SIZE) # Новое количество образцов для валидаци
```

```
Epoch 1/30
154/154 [==============================] - 230s 1s/step - loss: 0.0378 - acc: 0.9874 - val_
loss: 0.0110 - val_acc: 0.9951
Epoch 2/30
154/154 [==============================] - 162s 1s/step - loss: 0.0281 - acc: 0.9911 - val_
loss: 0.6909 - val_acc: 0.7982
Epoch 3/30
154/154 [==============================] - 162s 1s/step - loss: 0.0371 - acc: 0.9878 - val_
loss: 0.0348 - val_acc: 0.9893
Epoch 4/30
154/154 [==============================] - 161s 1s/step - loss: 0.0470 - acc: 0.9870 - val_
loss: 0.0522 - val_acc: 0.9857
Epoch 5/30
154/154 [==============================] - 164s 1s/step - loss: 0.0329 - acc: 0.9931 - val_
loss: 0.0070 - val_acc: 0.9969
Epoch 6/30
154/154 [==============================] - 154s 1s/step - loss: 0.0388 - acc: 0.9919 - val_
loss: 0.0018 - val_acc: 0.9996
Epoch 7/30
154/154 [==============================] - 145s 942ms/step - loss: 0.0415 - acc: 0.9870 - v
al_loss: 0.0055 - val_acc: 0.9982
Epoch 8/30
154/154 [==============================] - 145s 938ms/step - loss: 0.0313 - acc: 0.9927 - v
al_loss: 0.0070 - val_acc: 0.9973
Epoch 9/30
154/154 [==============================] - 145s 944ms/step - loss: 0.0264 - acc: 0.9923 - v
al_loss: 0.0033 - val_acc: 0.9996
Epoch 10/30
154/154 [==============================] - 145s 938ms/step - loss: 0.0258 - acc: 0.9919 - v
al_loss: 0.0040 - val_acc: 0.9987
Epoch 11/30
154/154 [==============================] - 145s 940ms/step - loss: 0.0293 - acc: 0.9911 - v
al_loss: 0.3273 - val_acc: 0.8509
Epoch 12/30
154/154 [==============================] - 144s 937ms/step - loss: 0.0254 - acc: 0.9911 - v
al_loss: 0.0109 - val_acc: 0.9960
Epoch 13/30
154/154 [==============================] - 145s 940ms/step - loss: 0.0257 - acc: 0.9959 - v
al_loss: 0.0453 - val_acc: 0.9982
Epoch 14/30
154/154 [==============================] - 144s 936ms/step - loss: 0.0496 - acc: 0.9927 - v
al_loss: 0.0148 - val_acc: 0.9946
Epoch 15/30
154/154 [==============================] - 145s 943ms/step - loss: 0.0321 - acc: 0.9931 - v
al_loss: 0.0087 - val_acc: 0.9978
Epoch 16/30
154/154 [==============================] - 144s 938ms/step - loss: 0.0261 - acc: 0.9902 - v
al_loss: 0.0017 - val_acc: 0.9991
Epoch 17/30
154/154 [==============================] - 145s 941ms/step - loss: 0.0362 - acc: 0.9931 - v
al_loss: 0.0170 - val_acc: 0.9960
Epoch 18/30
154/154 [==============================] - 146s 948ms/step - loss: 0.0271 - acc: 0.9915 - v
al_loss: 0.0059 - val_acc: 0.9978
Epoch 19/30
154/154 [==============================] - 145s 941ms/step - loss: 0.0267 - acc: 0.9935 - v
al_loss: 0.0217 - val_acc: 0.9937
Epoch 20/30
154/154 [==============================] - 145s 941ms/step - loss: 0.0188 - acc: 0.9927 - v
al_loss: 0.0386 - val_acc: 0.9929
Epoch 21/30
154/154 [==============================] - 145s 940ms/step - loss: 0.0372 - acc: 0.9923 - v
al_loss: 0.0061 - val_acc: 0.9978
Epoch 22/30
154/154 [==============================] - 145s 940ms/step - loss: 0.0179 - acc: 0.9955 - v
al_loss: 3.9138e-04 - val_acc: 1.0000
Epoch 23/30
154/154 [==============================] - 145s 941ms/step - loss: 0.0280 - acc: 0.9927 - v
al_loss: 0.0074 - val_acc: 0.9973
Epoch 24/30
```

```
154/154 [==============================] - 145s 943ms/step - loss: 0.0309 - acc: 0.9931 - v
al_loss: 0.0031 - val_acc: 0.9982
Epoch 25/30
154/154 [==============================] - 147s 954ms/step - loss: 0.0235 - acc: 0.9919 - v
al_loss: 0.0039 - val_acc: 0.9991
Epoch 26/30
154/154 [==============================] - 146s 944ms/step - loss: 0.0388 - acc: 0.9923 - v
al_loss: 0.0074 - val_acc: 0.9991
Epoch 27/30
154/154 [==============================] - 145s 944ms/step - loss: 0.0438 - acc: 0.9911 - v
al_loss: 0.0035 - val_acc: 0.9987
Epoch 28/30
154/154 [==============================] - 148s 957ms/step - loss: 0.0172 - acc: 0.9935 - v
al_loss: 0.0014 - val_acc: 0.9991
Epoch 29/30
154/154 [==============================] - 146s 951ms/step - loss: 0.0297 - acc: 0.9902 - v
al_loss: 0.0097 - val_acc: 0.9964
Epoch 30/30
154/154 [==============================] - 146s 949ms/step - loss: 0.0410 - acc: 0.9894 - v
al_loss: 5.2215e-04 - val_acc: 1.0000
```

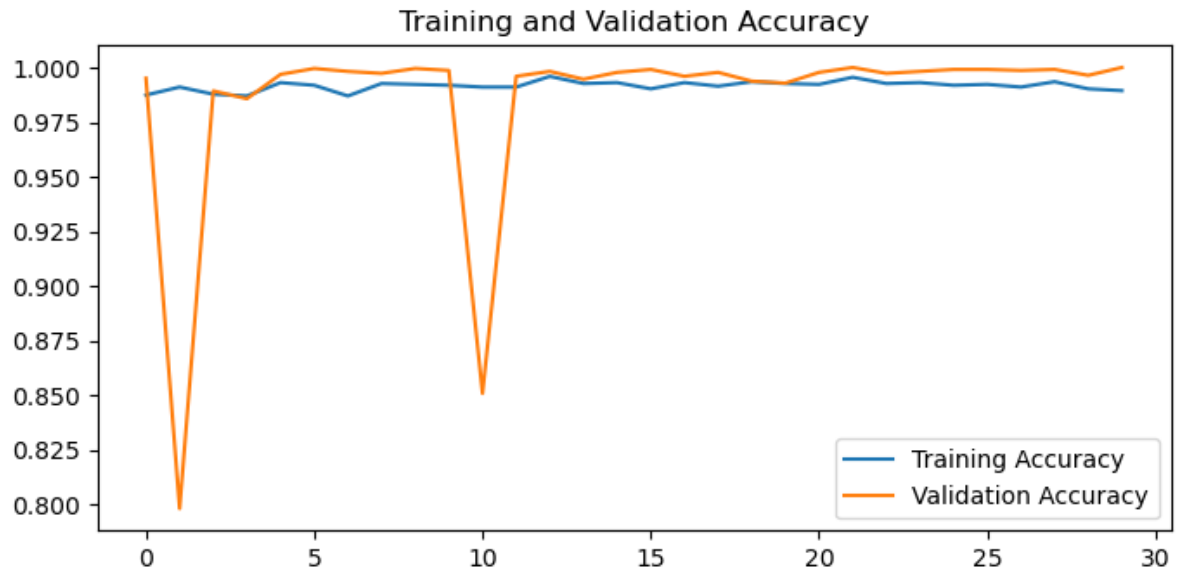
Ввод [50]:
```
# Оценка модели на новом тестовом наборе
score = model.evaluate(test_generator, steps=NEW_TEST_SAMPLES // BATCH_SIZE) # Новое количес
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
140/140 [==============================] - 44s 316ms/step - loss: 5.2238e-04 - acc: 1.0000
Test loss: 0.000522375397849828
Test accuracy: 1.0
```

Вывод: После повторного обучения модель продемонстрировала не только сохранение своей предыдущей высокой точности, но и показала улучшение в своей способности к обобщению на новые данные.

Ввод [51]:
```
# Создаем график точности
plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(history.history['acc'], label='Training Accuracy')
plt.plot(history.history['val_acc'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
```

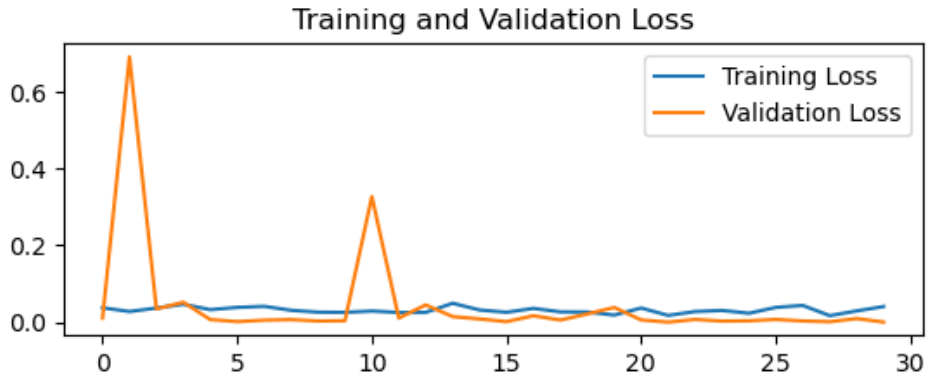
Out[51]:
```
Text(0.5, 1.0, 'Training and Validation Accuracy')
```



Выводы:

- Точность (accuracy) модели на новом тестовом наборе составляет 1.0, что означает, что модель классифицирует объекты с точностью 100%. Это является отличным показателем и свидетельствует о

Ввод [52]:
```
# Создаем график потерь
plt.subplot(2, 1, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Выводы:

- Значение потерь (loss) на новом тестовом наборе составляет 0.00052, что является очень низким значением. Это свидетельствует о том, что модель продолжает хорошо обобщать данные и имеет минимальные ошибки в предсказаниях на новых данных.

Ввод [53]:
```
# Сохранение модели после повторного обучения
model.save('model_retrained.h5')
```

Вывод:

В данной работе была построена модель бинарного классификатора, а также было проведено повторное обучение на новом наборе. Результаты обучения примерно одинаковы, поэтому было решено сохранить обе модели