Пожидаев Антон Николаевич, ИТУ (маг. 1 к.)
Лабораторная работа №2 Вариант - 3

Задание:
Решить задачу классификации исходного изображения с помощью глубокой сверточной нейронной сети (домашние животные). Оценить точность полученной модели. Не использовать переобученную нейронную сеть.

Датасет был взять здесь.

Код программы (dataPets.py)

```
import os, shutil

# Путь к исходным данным
original_dataset_dir = 'D:\\pyProj\\lab2\\dataset\\pets'

# Каталог для сохранения небольшого набора данных
base_dir = 'D:\\pyProj\\lab2\\dataset\\pets_small'
os.mkdir(base_dir)

# Каталоги для обучающего, проверочного и контрольного поднаборов
train_dir = os.path.join(base_dir, 'train')
os.mkdir(train_dir)
validation_dir = os.path.join(base_dir, 'validation')
os.mkdir(validation_dir)
test_dir = os.path.join(base_dir, 'test')
os.mkdir(test_dir)

# Каталог для обучающих изображений
train_cats_dir = os.path.join(train_dir, 'catsPets')
os.mkdir(train_cats_dir)

train_dogs_dir = os.path.join(train_dir, 'dogsPets')
os.mkdir(train_dogs_dir)

# Каталог для проверочных изображения
validation_cats_dir = os.path.join(validation_dir, 'catsPets')
os.mkdir(validation_cats_dir)

validation_dogs_dir = os.path.join(validation_dir, 'dogsPets')
os.mkdir(validation_dogs_dir)

# Каталог для контрольных изображений
test_cats_dir = os.path.join(test_dir, 'catsPets')
os.mkdir(test_cats_dir)

test_dogs_dir = os.path.join(test_dir, 'dogsPets')
os.mkdir(test_dogs_dir)
```

```
fnames = ['cat{}.jpg'.format(i) for i in range(1000)]
for fname in fnames:
src = os.path.join(original_dataset_dir, fname)
dst = os.path.join(train_cats_dir, fname)
shutil.copyfile(src, dst)

fnames = ['cat{}.jpg'.format(i) for i in range(1000, 1500)]
for fname in fnames:
src = os.path.join(original_dataset_dir, fname)
dst = os.path.join(validation_cats_dir, fname)
shutil.copyfile(src, dst)

fnames = ['cat{}.jpg'.format(i) for i in range(1500, 2000)]
for fname in fnames:
src = os.path.join(original_dataset_dir, fname)
dst = os.path.join(test_cats_dir, fname)
shutil.copyfile(src, dst)

fnames = ['dog{}.jpg'.format(i) for i in range(1000)]
for fname in fnames:
src = os.path.join(original_dataset_dir, fname)
dst = os.path.join(train_dogs_dir, fname)
shutil.copyfile(src, dst)

fnames = ['dog{}.jpg'.format(i) for i in range(1000, 1500)]
for fname in fnames:
src = os.path.join(original_dataset_dir, fname)
dst = os.path.join(validation_dogs_dir, fname)
shutil.copyfile(src, dst)

fnames = ['dog{}.jpg'.format(i) for i in range(1500, 2000)]
for fname in fnames:
src = os.path.join(original_dataset_dir, fname)
dst = os.path.join(test_dogs_dir, fname)
shutil.copyfile(src, dst)

print('total training cat images:',
len(os.listdir(train_cats_dir)))
print('total training dog images:',
len(os.listdir(train_dogs_dir)))
print('total validation cat images:',
len(os.listdir(validation_cats_dir)))
print('total validation dog images:',
len(os.listdir(validation_dogs_dir)))
print('total test cat images:', len(os.listdir(test_cats_dir)))
print('total test dog images:', len(os.listdir(test_dogs_dir)))
```

Код программы (main.py)

```
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from keras import layers
from keras import models
from dataPets import train_dir, validation_dir
from keras import optimizers

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
             optimizer=optimizers.RMSprop(lr=1e-4),
             metrics=['acc'])

train_datagen = ImageDataGenerator(rescale=1. / 255)
test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator =
   train_datagen.flow_from_directory( train_dir,
   target_size=(150, 150),
   batch_size=20,
   class_mode='binary')

validation_generator =
   test_datagen.flow_from_directory( validation_dir,
   target_size=(150, 150),
   batch_size=20,
   class_mode='binary')

for data_batch, labels_batch in train_generator:
   print('data batch shape:', data_batch.shape)
   print('labels batch shape:', labels_batch.shape)
```

```
    break

history = 
    model.fit( train_gene
    rator,
    steps_per_epoch=100,
    epochs=30,
    validation_data=validation_generator,
    validation_steps=50)

model.save('pets.h5')

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```

Код программы (loadModel.py)

```
import tensorflow as tf

load_model = tf.keras.models.load_model('pets.h5')
load_model.summary()
```

Результаты:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)
                                                         0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 128)       147584

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 512)               3211776

 dense_1 (Dense)             (None, 1)                 513


=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

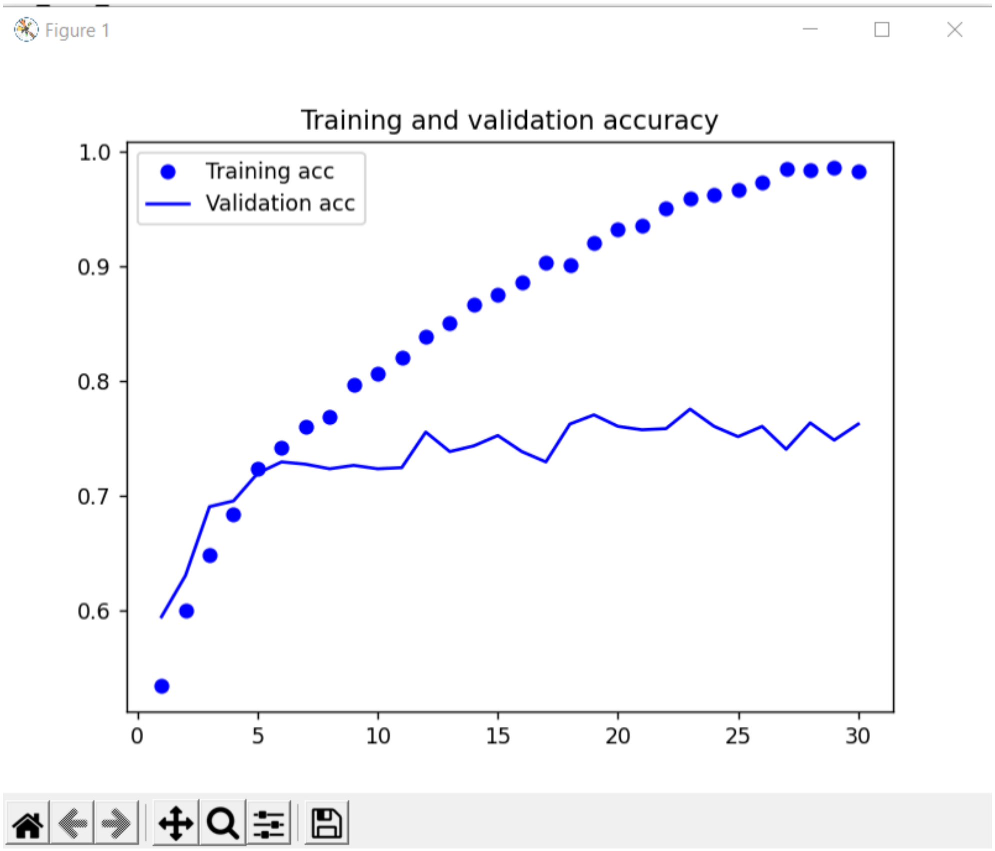

Рисунок 1 - Точность на этапах обучения и проверки

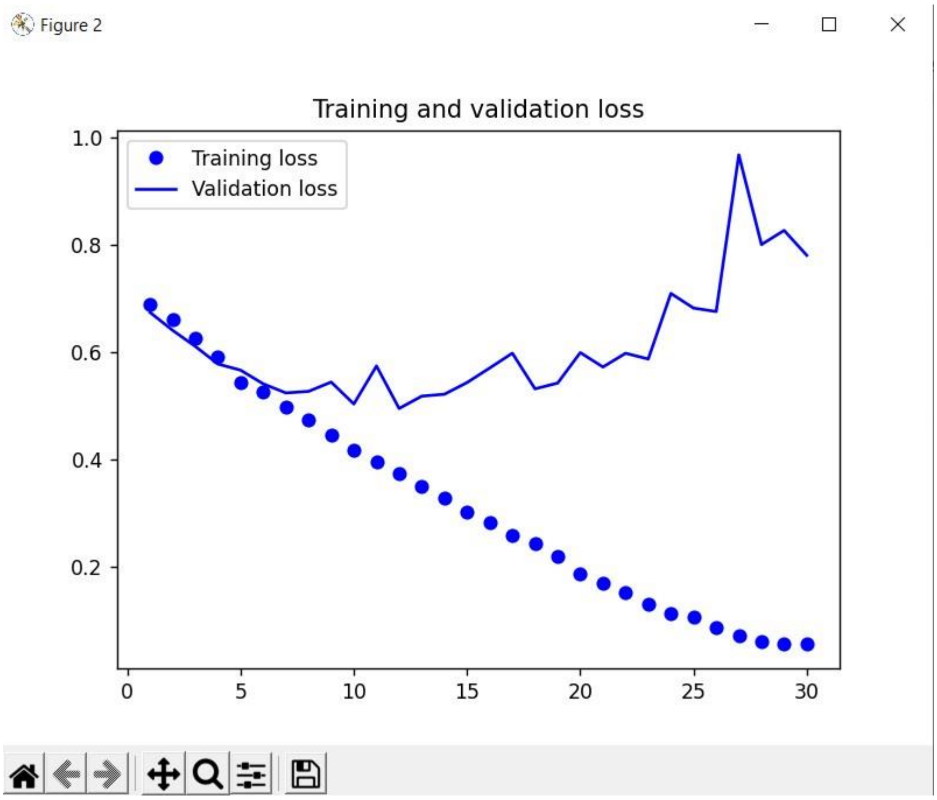

Рисунок 2 - Потери на этапах обучения и проверки

Вывод:

Точность на обучающих данных линейно растет и приближается к 100%, а точность на проверочных данных останавливается на отметке 70%. Потери на этапе проверки достигает минимума после 15 эпох, а потери на этапе обучения продолжают линейно уменьшаться.