Лабораторная работа №5. Применение сверточных нейронных сетей (бинарная классификация)

Данные: Набор данных DogsVsCats, который состоит из изображений различной размерности, содержащих фотографии собак и кошек. Обучающая выборка включает в себя 25 тыс. изображений (12,5 тыс. кошек: cat.0.jpg, …, cat.12499.jpg и 12,5 тыс. собак: dog.0.jpg, …, dog.12499.jpg), а контрольная выборка содержит 12,5 тыс. неразмеченных изображений. Скачать данные, а также проверить качество классификатора на тестовой выборке можно на сайте Kaggle -> https://www.kaggle.com/c/dogs-vs-cats/data

Задание 1.

Загрузите данные. Разделите исходный набор данных на обучающую, валидационную и контрольную выборки.

In [23]:

```
import os
from pandas import DataFrame
from tensorflow import keras
from sklearn.model_selection import train_test_split
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

data_folder = '../data'
path_to_train = data_folder + '/dogs-vs-cats/train'
path_to_test = data_folder + '/dogs-vs-cats/test1'
h, w, c = 128, 128, 3
batch_size = 256
epochs = 3
```

In [24]:

```
def data_frame(path):
    filenames = os.listdir(path)

    return DataFrame({
        'filename': filenames,
        'class': [filename.split('.')[0] for filename in filenames]
    })
```

In [25]:

```
train_df = data_frame(path_to_train)
train_df.head()
```

Out[25]:

	filename	class
0	dog.8186.jpg	dog
1	dog.10408.jpg	dog
2	dog.10087.jpg	dog
3	cat.6925.jpg	cat
4	cat.4682.jpg	cat

In [26]:

```
train_df, dev_df = train_test_split(train_df)
```

In [27]:

```
train_data_generator = ImageDataGenerator()
train_generator = train_data_generator.flow_from_dataframe(
```

```
    train_df,
    path_to_train,
    class_mode='binary',
    target_size=(h, w),
    batch_size=batch_size
)

dev_data_generator = ImageDataGenerator()
dev_generator = dev_data_generator.flow_from_dataframe(
    dev_df,
    path_to_train,
    class_mode='binary',
    target_size=(h, w),
    batch_size=batch_size
)
```

```
Found 18750 validated image filenames belonging to 2 classes.
Found 6250 validated image filenames belonging to 2 classes.
```

Задание 2.

Реализуйте глубокую нейронную сеть с как минимум тремя сверточными слоями. Какое качество классификации получено?

In [28]:

```
def model_factory():
    model = keras.models.Sequential([
        keras.layers.Conv2D(6, (5, 5), activation='relu', input_shape=(h, w, c)),
        keras.layers.MaxPooling2D(),

        keras.layers.Conv2D(16, (5, 5), activation='relu'),
        keras.layers.MaxPooling2D(),

        keras.layers.Conv2D(32, (5, 5), activation='relu'),
        keras.layers.MaxPooling2D(),

        keras.layers.Flatten(),
        keras.layers.Dense(120, activation='relu'),

        keras.layers.Dense(84, activation='relu'),

        keras.layers.Dense(1, activation='sigmoid')
    ])

    return model
```

In [29]:

```
def train(model):
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    model.fit(
        train_generator,
        epochs=epochs,
        validation_data=dev_generator,
        validation_steps=dev_df.shape[0] // batch_size,
        steps_per_epoch=train_df.shape[0] // batch_size
    )
```

In [30]:

```
train(model_factory())
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
```

```
Train for 73 steps, validate for 24 steps
Epoch 1/3
73/73 [==============================] - 91s 1s/step - loss: 3.7510 - accuracy: 0.5261 - val_loss:
0.6887 - val_accuracy: 0.5280
Epoch 2/3
73/73 [==============================] - 91s 1s/step - loss: 0.6782 - accuracy: 0.5568 - val_loss:
0.6906 - val_accuracy: 0.5594
Epoch 3/3
73/73 [==============================] - 87s 1s/step - loss: 0.6610 - accuracy: 0.5858 - val_loss:
0.6884 - val_accuracy: 0.5511
```

Задание 3.

Примените дополнение данных (data augmentation). Как это повлияло на качество классификатора?

In [31]:

```
train_data_generator = ImageDataGenerator(
    rotation_range=15,
    shear_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1
)
train_generator = train_data_generator.flow_from_dataframe(
    train_df,
    path_to_train,
    class_mode='binary',
    target_size=(h, w),
    batch_size=batch_size
)
```

```
Found 18750 validated image filenames belonging to 2 classes.
```

In [32]:

```
train(model_factory())
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 73 steps, validate for 24 steps
Epoch 1/3
73/73 [==============================] - 132s 2s/step - loss: 2.0473 - accuracy: 0.5273 -
val_loss: 0.6815 - val_accuracy: 0.5584
Epoch 2/3
73/73 [==============================] - 125s 2s/step - loss: 0.6643 - accuracy: 0.5914 -
val_loss: 0.6534 - val_accuracy: 0.6050
Epoch 3/3
73/73 [==============================] - 126s 2s/step - loss: 0.6383 - accuracy: 0.6358 -
val_loss: 0.6073 - val_accuracy: 0.6686
```

Задание 4.

Поэкспериментируйте с готовыми нейронными сетями (например, AlexNet, VGG16, Inception и т.п.), применив передаточное обучение. Как это повлияло на качество классификатора? Какой максимальный результат удалось получить на сайте Kaggle? Почему?

In [33]:

```
def pre_trained_model_factory():
    pre_trained_model = keras.applications.VGG16(input_shape=(h, w, c),
                                                 include_top=False,
                                                 weights='imagenet')
```

```
    pre_trained_model.trainable = False

    model = keras.models.Sequential([
        pre_trained_model,
        keras.layers.GlobalAveragePooling2D(),
        keras.layers.Dense(512, activation='relu'),
        keras.layers.Dense(1, activation='sigmoid')
    ])

    return model
```

In [34]:

```
model = pre_trained_model_factory()
train(model)
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 73 steps, validate for 24 steps
Epoch 1/3
73/73 [==============================] - 486s 7s/step - loss: 1.1058 - accuracy: 0.8529 -
val_loss: 0.2997 - val_accuracy: 0.9250
Epoch 2/3
73/73 [==============================] - 490s 7s/step - loss: 0.2791 - accuracy: 0.9049 -
val_loss: 0.2177 - val_accuracy: 0.9297
Epoch 3/3
73/73 [==============================] - 496s 7s/step - loss: 0.2137 - accuracy: 0.9178 -
val_loss: 0.1915 - val_accuracy: 0.9292
```

In [35]:

```
test_filenames = os.listdir(path_to_test)
test_df = DataFrame({
    'filename': test_filenames
})
```

In [36]:

```
test_data_generator = ImageDataGenerator()
test_generator = test_data_generator.flow_from_dataframe(
    test_df,
    path_to_test,
    y_col=None,
    class_mode=None,
    batch_size=batch_size,
    target_size=(h, w),
    shuffle=False
)
```

```
Found 12500 validated image filenames.
```

In [39]:

```
predict = model.predict(test_generator)
threshold = 0.5
test_df['class'] = np.where(predict > threshold, 1, 0)
```

In [40]:

```
submission_df = test_df.copy()
submission_df['id'] = submission_df['filename'].str.split('.').str[0]
submission_df['label'] = submission_df['class']
submission_df.drop(['filename', 'class'], axis=1, inplace=True)
submission_df.to_csv(data_folder + '/submission_vgg.csv', index=False)
```

Submission and Description	Private Score	Public Score	Use for Final Score
submission_vgg.csv 3 minutes ago by Valera Yalovchuk VGG pre trained	2.25471	2.25471	☐