In [1]:

```python
# Пятая нейросеть. CNN для цветных изображений разного размера. Улучшение качества обучения

# Импорт всего важного
from __future__ import absolute_import, division, print_function, unicode_literals
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.models import load_model
import tensorflow_datasets as tfds
import math
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```python
# Создание переменных путей

_URL = 'https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip'
zip_dir = tf.keras.utils.get_file('cats_and_dogs_filterted.zip', origin=_URL, extract=True)

zip_dir_base = os.path.dirname(zip_dir)

base_dir = os.path.join(os.path.dirname(zip_dir), 'cats_and_dogs_filtered')
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')

train_cats_dir = os.path.join(train_dir, 'cats')
train_dogs_dir = os.path.join(train_dir, 'dogs')
validation_cats_dir = os.path.join(validation_dir, 'cats')
validation_dogs_dir = os.path.join(validation_dir, 'dogs')

num_cats_tr = len(os.listdir(train_cats_dir))
num_dogs_tr = len(os.listdir(train_dogs_dir))
num_cats_val = len(os.listdir(validation_cats_dir))
num_dogs_val = len(os.listdir(validation_dogs_dir))
total_train = num_cats_tr + num_dogs_tr
total_val = num_cats_val + num_dogs_val
```

In [3]:

```python
# Функция, которая возвращает изображения со случайными преобразованиями
# Нужна для расширения тестовых данных

def plotImages(images_arr):
    fig, axes = plt.subplots(1, 5, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip(images_arr, axes):
        ax.imshow(img)
    plt.tight_layout()
    plt.show()
```

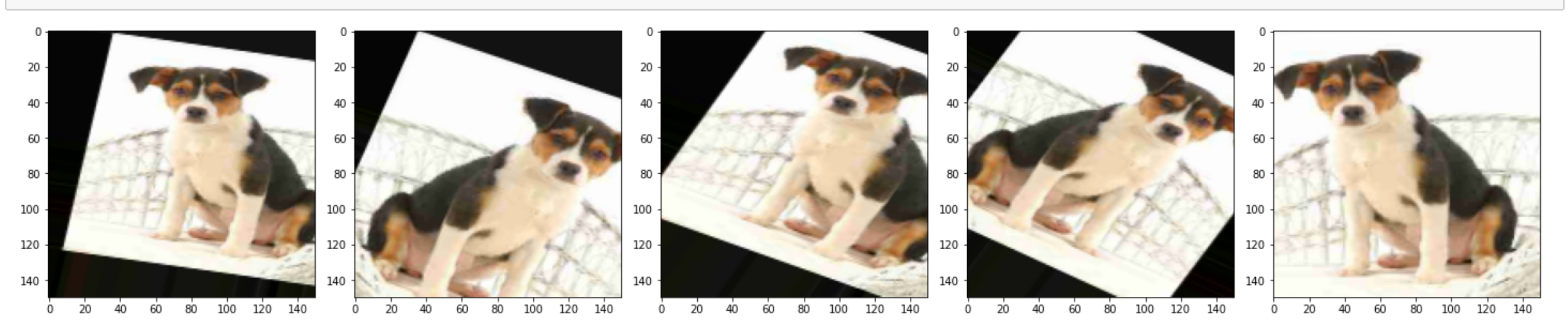
In [4]:

```python
BATCH_SIZE = 100 # Количество тренировочных изображений для обработки перед обновлением параметров модели
IMG_SHAPE = 150 # Размерность 150x150 к которой будет преведено входное изображение

# Приведение с помощью ImageDataGenerator изображений к тензорам с плавающей запятой
# А также нормализация изображений (переход от формата [0; 255] к формату [0; 1])
train_image_generator = 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')
validation_image_generator = ImageDataGenerator(rescale=1./255)

# Загрузка данных
# Сначала тренировочные данные
train_data_gen = train_image_generator.flow_from_directory(batch_size=BATCH_SIZE,
                                                           directory=train_dir,
                                                           shuffle=True,
                                                           target_size=(IMG_SHAPE,IMG_SHAPE),
                                                           class_mode='binary')

# Затем валидационные данные
val_data_gen = validation_image_generator.flow_from_directory(batch_size=BATCH_SIZE,
                                                              directory=validation_dir,
                                                              shuffle=False,
                                                              target_size=(IMG_SHAPE,IMG_SHAPE),
                                                              class_mode='binary')
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

In [5]:

```python
# Преобразования в действии

augmented_images = [train_data_gen[0][0][0] for i in range(5)]
plotImages(augmented_images)
```



In [6]:

```python
# Создание модели
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(IMG_SHAPE, IMG_SHAPE, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.5), # Dropout-слой, отключающий половину случайных нейронов
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])

# Компиляция модели
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Представление модели
model.summary()
```

```
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 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
dropout (Dropout)            (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 512)               3211776
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1026
=================================================================
Total params: 3,453,634
Trainable params: 3,453,634
Non-trainable params: 0
_________________________________________________________________
```

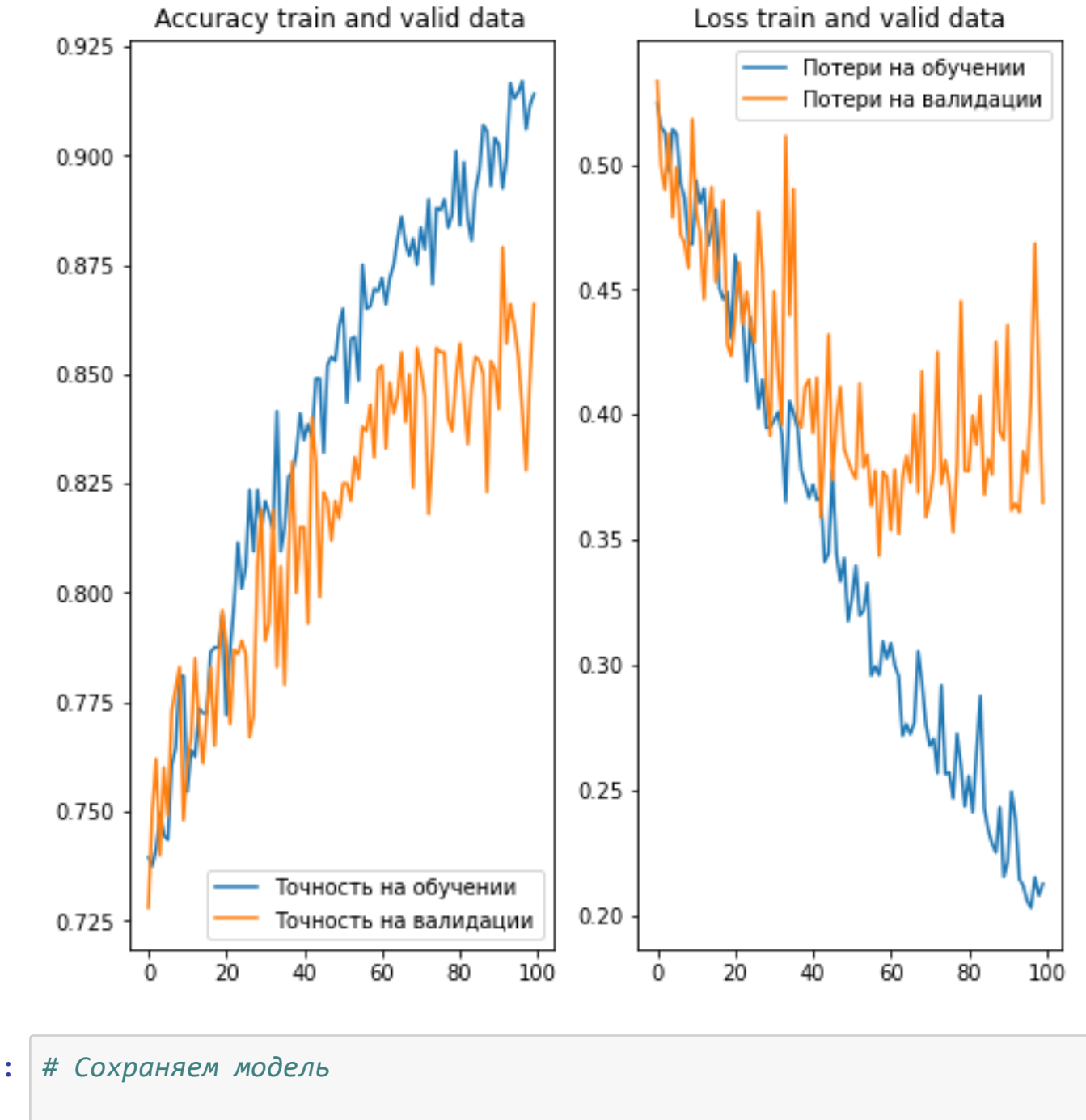
In [7]:

```python
# Тренировка моделей
EPOCHS = 100
history = model.fit_generator( # Используется fit_generator вместо обычного fit
    train_data_gen,
    steps_per_epoch=int(np.ceil(total_train / float(BATCH_SIZE))),
    epochs=EPOCHS,
    validation_data=val_data_gen,
    validation_steps=int(np.ceil(total_val / float(BATCH_SIZE)))
)
```

```
Epoch 1/100
20/20 [==============================] - 16s 782ms/step - loss: 0.5245 - accuracy: 0.7395 - val_loss: 0.5333 - val_accuracy: 0.
7280
Epoch 2/100
20/20 [==============================] - 16s 781ms/step - loss: 0.5151 - accuracy: 0.7375 - val_loss: 0.4993 - val_accuracy: 0.
7500
Epoch 3/100
20/20 [==============================] - 16s 784ms/step - loss: 0.5127 - accuracy: 0.7415 - val_loss: 0.4898 - val_accuracy: 0.
7620
Epoch 4/100
20/20 [==============================] - 16s 786ms/step - loss: 0.4971 - accuracy: 0.7495 - val_loss: 0.5125 - val_accuracy: 0.
7400
Epoch 5/100
20/20 [==============================] - 16s 778ms/step - loss: 0.5142 - accuracy: 0.7445 - val_loss: 0.4788 - val_accuracy: 0.
7600
Epoch 6/100
20/20 [==============================] - 16s 780ms/step - loss: 0.5120 - accuracy: 0.7435 - val_loss: 0.4990 - val_accuracy: 0.
7490
Epoch 7/100
20/20 [==============================] - 16s 783ms/step - loss: 0.4925 - accuracy: 0.7605 - val_loss: 0.4721 - val_accuracy: 0.
7730
Epoch 8/100
20/20 [==============================] - 16s 780ms/step - loss: 0.4868 - accuracy: 0.7645 - val_loss: 0.4686 - val_accuracy: 0.
7780
Epoch 9/100
20/20 [==============================] - 16s 789ms/step - loss: 0.4686 - accuracy: 0.7810 - val_loss: 0.4586 - val_accuracy: 0.
7830
Epoch 10/100
20/20 [==============================] - 16s 789ms/step - loss: 0.4680 - accuracy: 0.7810 - val_loss: 0.5180 - val_accuracy: 0.
7480
Epoch 11/100
20/20 [==============================] - 16s 781ms/step - loss: 0.4935 - accuracy: 0.7545 - val_loss: 0.4811 - val_accuracy: 0.
7610
Epoch 12/100
20/20 [==============================] - 16s 781ms/step - loss: 0.4846 - accuracy: 0.7640 - val_loss: 0.4729 - val_accuracy: 0.
7670
Epoch 13/100
20/20 [==============================] - 16s 776ms/step - loss: 0.4904 - accuracy: 0.7625 - val_loss: 0.4461 - val_accuracy: 0.
7850
Epoch 14/100
20/20 [==============================] - 16s 780ms/step - loss: 0.4674 - accuracy: 0.7735 - val_loss: 0.4771 - val_accuracy: 0.
7700
Epoch 15/100
20/20 [==============================] - 16s 779ms/step - loss: 0.4738 - accuracy: 0.7725 - val_loss: 0.4910 - val_accuracy: 0.
7610
Epoch 16/100
20/20 [==============================] - 16s 777ms/step - loss: 0.4821 - accuracy: 0.7725 - val_loss: 0.4529 - val_accuracy: 0.
7720
Epoch 17/100
20/20 [==============================] - 16s 783ms/step - loss: 0.4504 - accuracy: 0.7865 - val_loss: 0.4637 - val_accuracy: 0.
7830
Epoch 18/100
20/20 [==============================] - 16s 781ms/step - loss: 0.4460 - accuracy: 0.7875 - val_loss: 0.4857 - val_accuracy: 0.
7650
Epoch 19/100
20/20 [==============================] - 16s 778ms/step - loss: 0.4490 - accuracy: 0.7875 - val_loss: 0.4281 - val_accuracy: 0.
7840
Epoch 20/100
20/20 [==============================] - 16s 778ms/step - loss: 0.4305 - accuracy: 0.7950 - val_loss: 0.4234 - val_accuracy: 0.
7960
Epoch 21/100
20/20 [==============================] - 16s 780ms/step - loss: 0.4639 - accuracy: 0.7720 - val_loss: 0.4389 - val_accuracy: 0.
7880
Epoch 22/100
20/20 [==============================] - 16s 784ms/step - loss: 0.4535 - accuracy: 0.7870 - val_loss: 0.4608 - val_accuracy: 0.
7700
Epoch 23/100
20/20 [==============================] - 16s 781ms/step - loss: 0.4374 - accuracy: 0.7975 - val_loss: 0.4361 - val_accuracy: 0.
7870
Epoch 24/100
20/20 [==============================] - 16s 778ms/step - loss: 0.4131 - accuracy: 0.8115 - val_loss: 0.4491 - val_accuracy: 0.
7860
Epoch 25/100
20/20 [==============================] - 16s 779ms/step - loss: 0.4387 - accuracy: 0.8010 - val_loss: 0.4355 - val_accuracy: 0.
7890
Epoch 26/100
20/20 [==============================] - 16s 776ms/step - loss: 0.4202 - accuracy: 0.8060 - val_loss: 0.4288 - val_accuracy: 0.
7860
Epoch 27/100
20/20 [==============================] - 15s 773ms/step - loss: 0.4024 - accuracy: 0.8235 - val_loss: 0.4811 - val_accuracy: 0.
7670
Epoch 28/100
20/20 [==============================] - 16s 780ms/step - loss: 0.4139 - accuracy: 0.8095 - val_loss: 0.4576 - val_accuracy: 0.
7720
Epoch 29/100
20/20 [==============================] - 16s 786ms/step - loss: 0.3947 - accuracy: 0.8235 - val_loss: 0.4116 - val_accuracy: 0.
8060
Epoch 30/100
20/20 [==============================] - 16s 777ms/step - loss: 0.3950 - accuracy: 0.8145 - val_loss: 0.3916 - val_accuracy: 0.
8190
Epoch 31/100
20/20 [==============================] - 16s 776ms/step - loss: 0.3975 - accuracy: 0.8210 - val_loss: 0.4493 - val_accuracy: 0.
7890
Epoch 32/100
20/20 [==============================] - 16s 777ms/step - loss: 0.4009 - accuracy: 0.8180 - val_loss: 0.4172 - val_accuracy: 0.
7930
Epoch 33/100
20/20 [==============================] - 16s 780ms/step - loss: 0.3920 - accuracy: 0.8145 - val_loss: 0.3959 - val_accuracy: 0.
8190
Epoch 34/100
20/20 [==============================] - 16s 782ms/step - loss: 0.3651 - accuracy: 0.8415 - val_loss: 0.5114 - val_accuracy: 0.
7830
Epoch 35/100
20/20 [==============================] - 16s 777ms/step - loss: 0.4054 - accuracy: 0.8095 - val_loss: 0.4396 - val_accuracy: 0.
8060
Epoch 36/100
20/20 [==============================] - 16s 784ms/step - loss: 0.4006 - accuracy: 0.8150 - val_loss: 0.4901 - val_accuracy: 0.
7790
Epoch 37/100
20/20 [==============================] - 16s 782ms/step - loss: 0.3949 - accuracy: 0.8265 - val_loss: 0.3985 - val_accuracy: 0.
8080
Epoch 38/100
20/20 [==============================] - 16s 787ms/step - loss: 0.3775 - accuracy: 0.8275 - val_loss: 0.3947 - val_accuracy: 0.
8300
Epoch 39/100
20/20 [==============================] - 16s 783ms/step - loss: 0.3721 - accuracy: 0.8320 - val_loss: 0.4112 - val_accuracy: 0.
8000
Epoch 40/100
20/20 [==============================] - 16s 782ms/step - loss: 0.3667 - accuracy: 0.8410 - val_loss: 0.4140 - val_accuracy: 0.
8150
Epoch 41/100
20/20 [==============================] - 16s 782ms/step - loss: 0.3720 - accuracy: 0.8350 - val_loss: 0.3926 - val_accuracy: 0.
8150
Epoch 42/100
20/20 [==============================] - 16s 780ms/step - loss: 0.3658 - accuracy: 0.8385 - val_loss: 0.4148 - val_accuracy: 0.
7930
Epoch 43/100
20/20 [==============================] - 16s 784ms/step - loss: 0.3671 - accuracy: 0.8355 - val_loss: 0.3590 - val_accuracy: 0.
8400
Epoch 44/100
20/20 [==============================] - 16s 780ms/step - loss: 0.3412 - accuracy: 0.8490 - val_loss: 0.3906 - val_accuracy: 0.
8300
Epoch 45/100
20/20 [==============================] - 16s 777ms/step - loss: 0.3446 - accuracy: 0.8490 - val_loss: 0.4319 - val_accuracy: 0.
7990
Epoch 46/100
20/20 [==============================] - 16s 780ms/step - loss: 0.3779 - accuracy: 0.8320 - val_loss: 0.3738 - val_accuracy: 0.
8230
Epoch 47/100
20/20 [==============================] - 16s 784ms/step - loss: 0.3445 - accuracy: 0.8520 - val_loss: 0.3982 - val_accuracy: 0.
8210
Epoch 48/100
20/20 [==============================] - 16s 792ms/step - loss: 0.3333 - accuracy: 0.8540 - val_loss: 0.4111 - val_accuracy: 0.
8120
Epoch 49/100
20/20 [==============================] - 16s 783ms/step - loss: 0.3427 - accuracy: 0.8530 - val_loss: 0.3863 - val_accuracy: 0.
8210
Epoch 50/100
20/20 [==============================] - 16s 778ms/step - loss: 0.3174 - accuracy: 0.8610 - val_loss: 0.3818 - val_accuracy: 0.
8170
Epoch 51/100
20/20 [==============================] - 16s 779ms/step - loss: 0.3265 - accuracy: 0.8650 - val_loss: 0.3774 - val_accuracy: 0.
8250
Epoch 52/100
20/20 [==============================] - 16s 778ms/step - loss: 0.3396 - accuracy: 0.8435 - val_loss: 0.3743 - val_accuracy: 0.
8250
Epoch 53/100
20/20 [==============================] - 16s 785ms/step - loss: 0.3196 - accuracy: 0.8580 - val_loss: 0.4124 - val_accuracy: 0.
8210
Epoch 54/100
20/20 [==============================] - 16s 778ms/step - loss: 0.3217 - accuracy: 0.8585 - val_loss: 0.3787 - val_accuracy: 0.
8310
Epoch 55/100
20/20 [==============================] - 16s 779ms/step - loss: 0.3326 - accuracy: 0.8485 - val_loss: 0.3840 - val_accuracy: 0.
8260
Epoch 56/100
20/20 [==============================] - 15s 774ms/step - loss: 0.2957 - accuracy: 0.8750 - val_loss: 0.3635 - val_accuracy: 0.
8380
Epoch 57/100
20/20 [==============================] - 15s 774ms/step - loss: 0.2994 - accuracy: 0.8650 - val_loss: 0.3775 - val_accuracy: 0.
8370
Epoch 58/100
20/20 [==============================] - 16s 783ms/step - loss: 0.2959 - accuracy: 0.8655 - val_loss: 0.3436 - val_accuracy: 0.
8430
Epoch 59/100
20/20 [==============================] - 16s 775ms/step - loss: 0.3093 - accuracy: 0.8695 - val_loss: 0.3772 - val_accuracy: 0.
8310
Epoch 60/100
20/20 [==============================] - 15s 774ms/step - loss: 0.3025 - accuracy: 0.8690 - val_loss: 0.3750 - val_accuracy: 0.
8510
Epoch 61/100
20/20 [==============================] - 16s 776ms/step - loss: 0.3086 - accuracy: 0.8720 - val_loss: 0.3538 - val_accuracy: 0.
8520
Epoch 62/100
20/20 [==============================] - 15s 774ms/step - loss: 0.2998 - accuracy: 0.8660 - val_loss: 0.3779 - val_accuracy: 0.
8330
Epoch 63/100
20/20 [==============================] - 16s 786ms/step - loss: 0.2954 - accuracy: 0.8720 - val_loss: 0.3523 - val_accuracy: 0.
8480
Epoch 64/100
20/20 [==============================] - 15s 775ms/step - loss: 0.2717 - accuracy: 0.8750 - val_loss: 0.3746 - val_accuracy: 0.
8410
Epoch 65/100
20/20 [==============================] - 15s 774ms/step - loss: 0.2763 - accuracy: 0.8810 - val_loss: 0.3835 - val_accuracy: 0.
8450
Epoch 66/100
20/20 [==============================] - 16s 778ms/step - loss: 0.2723 - accuracy: 0.8860 - val_loss: 0.3729 - val_accuracy: 0.
8550
Epoch 67/100
20/20 [==============================] - 16s 787ms/step - loss: 0.2768 - accuracy: 0.8800 - val_loss: 0.4000 - val_accuracy: 0.
8390
Epoch 68/100
20/20 [==============================] - 16s 780ms/step - loss: 0.3054 - accuracy: 0.8770 - val_loss: 0.3687 - val_accuracy: 0.
8500
Epoch 69/100
20/20 [==============================] - 16s 776ms/step - loss: 0.2931 - accuracy: 0.8810 - val_loss: 0.4173 - val_accuracy: 0.
8240
Epoch 70/100
20/20 [==============================] - 16s 777ms/step - loss: 0.2762 - accuracy: 0.8750 - val_loss: 0.3590 - val_accuracy: 0.
8560
Epoch 71/100
20/20 [==============================] - 16s 775ms/step - loss: 0.2675 - accuracy: 0.8835 - val_loss: 0.3660 - val_accuracy: 0.
8510
Epoch 72/100
20/20 [==============================] - 16s 776ms/step - loss: 0.2704 - accuracy: 0.8785 - val_loss: 0.3783 - val_accuracy: 0.
8450
Epoch 73/100
20/20 [==============================] - 16s 781ms/step - loss: 0.2567 - accuracy: 0.8900 - val_loss: 0.4251 - val_accuracy: 0.
8180
Epoch 74/100
20/20 [==============================] - 15s 772ms/step - loss: 0.2918 - accuracy: 0.8705 - val_loss: 0.3722 - val_accuracy: 0.
8330
Epoch 75/100
20/20 [==============================] - 16s 776ms/step - loss: 0.2564 - accuracy: 0.8880 - val_loss: 0.3818 - val_accuracy: 0.
8560
Epoch 76/100
20/20 [==============================] - 16s 777ms/step - loss: 0.2569 - accuracy: 0.8875 - val_loss: 0.3717 - val_accuracy: 0.
8550
Epoch 77/100
20/20 [==============================] - 15s 774ms/step - loss: 0.2466 - accuracy: 0.8900 - val_loss: 0.3530 - val_accuracy: 0.
8550
Epoch 78/100
20/20 [==============================] - 16s 781ms/step - loss: 0.2725 - accuracy: 0.8835 - val_loss: 0.3804 - val_accuracy: 0.
8400
Epoch 79/100
20/20 [==============================] - 15s 775ms/step - loss: 0.2596 - accuracy: 0.8865 - val_loss: 0.4453 - val_accuracy: 0.
8370
Epoch 80/100
20/20 [==============================] - 15s 774ms/step - loss: 0.2434 - accuracy: 0.9010 - val_loss: 0.3775 - val_accuracy: 0.
8490
Epoch 81/100
20/20 [==============================] - 15s 773ms/step - loss: 0.2555 - accuracy: 0.8840 - val_loss: 0.3774 - val_accuracy: 0.
8570
Epoch 82/100
20/20 [==============================] - 15s 774ms/step - loss: 0.2411 - accuracy: 0.8985 - val_loss: 0.3996 - val_accuracy: 0.
8450
Epoch 83/100
20/20 [==============================] - 15s 774ms/step - loss: 0.2651 - accuracy: 0.8855 - val_loss: 0.3881 - val_accuracy: 0.
8340
Epoch 84/100
20/20 [==============================] - 15s 771ms/step - loss: 0.2876 - accuracy: 0.8805 - val_loss: 0.4076 - val_accuracy: 0.
8470
Epoch 85/100
20/20 [==============================] - 15s 771ms/step - loss: 0.2425 - accuracy: 0.8920 - val_loss: 0.3680 - val_accuracy: 0.
8540
Epoch 86/100
20/20 [==============================] - 16s 787ms/step - loss: 0.2340 - accuracy: 0.8965 - val_loss: 0.3825 - val_accuracy: 0.
8530
Epoch 87/100
20/20 [==============================] - 16s 776ms/step - loss: 0.2285 - accuracy: 0.9070 - val_loss: 0.3759 - val_accuracy: 0.
8500
Epoch 88/100
20/20 [==============================] - 16s 780ms/step - loss: 0.2250 - accuracy: 0.9055 - val_loss: 0.4290 - val_accuracy: 0.
8230
Epoch 89/100
20/20 [==============================] - 15s 771ms/step - loss: 0.2430 - accuracy: 0.8930 - val_loss: 0.3933 - val_accuracy: 0.
8530
Epoch 90/100
20/20 [==============================] - 15s 774ms/step - loss: 0.2152 - accuracy: 0.9040 - val_loss: 0.3899 - val_accuracy: 0.
8510
Epoch 91/100
20/20 [==============================] - 16s 775ms/step - loss: 0.2214 - accuracy: 0.9025 - val_loss: 0.4358 - val_accuracy: 0.
8420
Epoch 92/100
20/20 [==============================] - 16s 785ms/step - loss: 0.2493 - accuracy: 0.8925 - val_loss: 0.3617 - val_accuracy: 0.
8790
Epoch 93/100
20/20 [==============================] - 16s 778ms/step - loss: 0.2384 - accuracy: 0.8995 - val_loss: 0.3645 - val_accuracy: 0.
8570
Epoch 94/100
20/20 [==============================] - 15s 774ms/step - loss: 0.2143 - accuracy: 0.9165 - val_loss: 0.3610 - val_accuracy: 0.
8660
Epoch 95/100
20/20 [==============================] - 15s 773ms/step - loss: 0.2116 - accuracy: 0.9130 - val_loss: 0.3853 - val_accuracy: 0.
8610
Epoch 96/100
20/20 [==============================] - 16s 780ms/step - loss: 0.2056 - accuracy: 0.9145 - val_loss: 0.3768 - val_accuracy: 0.
8540
Epoch 97/100
20/20 [==============================] - 16s 784ms/step - loss: 0.2029 - accuracy: 0.9170 - val_loss: 0.4079 - val_accuracy: 0.
8420
Epoch 98/100
20/20 [==============================] - 16s 782ms/step - loss: 0.2151 - accuracy: 0.9060 - val_loss: 0.4684 - val_accuracy: 0.
8280
Epoch 99/100
20/20 [==============================] - 15s 774ms/step - loss: 0.2078 - accuracy: 0.9115 - val_loss: 0.4150 - val_accuracy: 0.
8490
Epoch 100/100
20/20 [==============================] - 16s 776ms/step - loss: 0.2123 - accuracy: 0.9140 - val_loss: 0.3649 - val_accuracy: 0.
8660
```

In [8]:

```python
# Визуализация результатов тренировки
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(EPOCHS)

plt.figure(figsize=(8,8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Точность на обучении')
plt.plot(epochs_range, val_acc, label='Точность на валидации')
plt.legend(loc='lower right')
plt.title('Accuracy train and valid data')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Потери на обучении')
plt.plot(epochs_range, val_loss, label='Потери на валидации')
plt.legend(loc='upper right')
plt.title('Loss train and valid data')
plt.show()
```



In [8]:

```python
# Сохраняем модель

model.save('Fifth_network_CNN_model.h5')
```