In [1]:
```
# Четвёртая нейросеть. CNN для цветных изображений разного размера

# 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
import tqdm
import tqdm.auto
tqdm.tqdm = tqdm.auto.tqdm
```

In [2]:
```
# Создание переменных путей

base_dir = '.\cats_and_dogs_filtered' # Тут хранится разархивированный архив
# Ссылка на сам архив: https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip
# Архив содержит 2000 цветных изображений для тренировки и 1000 изображений для валидации
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')

print(train_dir, train_cats_dir, train_dogs_dir, validation_cats_dir, validation_dogs_dir)
```

```
.\cats_and_dogs_filtered\train .\cats_and_dogs_filtered\train\cats .\cats_and_dogs_filtered\train\dogs .\cats_and_dogs_filtered
\validation\cats .\cats_and_dogs_filtered\validation\dogs
```

In [3]:
```
# Подсчёт количества изображений в наборах

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

print('Кошек в тестовом наборе данных: ', num_cats_tr)
print('Собак в тестовом наборе данных: ', num_dogs_tr)
print('Кошек в валидационном наборе данных: ', num_cats_val)
print('Собак в валидационном наборе данных: ', num_dogs_val, '\n')
print('Всего изображений в тренировочном наборе данных: ', total_train)
print('Всего изображений в валидационном наборе данных: ', total_val)
```

```
Кошек в тестовом наборе данных:  1000
Собак в тестовом наборе данных:  1000
Кошек в валидационном наборе данных:  500
Собак в валидационном наборе данных:  500

Всего изображений в тренировочном наборе данных:  2000
Всего изображений в валидационном наборе данных:  1000
```

In [4]:
```
BATCH_SIZE = 100 # Количество тренировочных изображений для обработки перед обновлением параметров модели
IMG_SHAPE = 150 # Размерность 150х150 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)
validation_image_generator = ImageDataGenerator(rescale=1./255)

# Метод flow_from_directory загрузит изображения с диска, нормализует данные и изменит размер изображений
# 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')
```

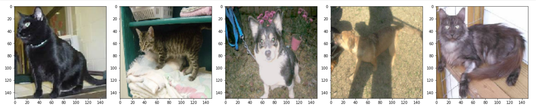
```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

In [5]:
```
# Визуализация изображений

# Функция next возвращает блок изображений из набора данных.
# Один блок представляет собой кортеж из (множество изображений, множество меток).
sample_training_images, _ = next(train_data_gen)

# Данная функция отрисует изображения в сетке размером 1х5
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()

plotImages(sample_training_images[:5]) # Отрисовываем изображения 0-4
```



In [6]:
```
# 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.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
_________________________________________________________________
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 [8]:
```
# Тренировка моделей
EPOCHS = 15 #Количество прогонов в тренировке
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 1/15
20/20 [==============================] - 123s 6s/step - loss: 0.6923 - accuracy: 0.5210 - val_loss: 0.6894 - val_accuracy: 0.50
00
Epoch 2/15
20/20 [==============================] - 114s 6s/step - loss: 0.6862 - accuracy: 0.5380 - val_loss: 0.6805 - val_accuracy: 0.57
20
Epoch 3/15
20/20 [==============================] - 112s 6s/step - loss: 0.6581 - accuracy: 0.6015 - val_loss: 0.6396 - val_accuracy: 0.62
90
Epoch 4/15
20/20 [==============================] - 117s 6s/step - loss: 0.6295 - accuracy: 0.6460 - val_loss: 0.6288 - val_accuracy: 0.63
30
Epoch 5/15
20/20 [==============================] - 118s 6s/step - loss: 0.5995 - accuracy: 0.6700 - val_loss: 0.6137 - val_accuracy: 0.65
80
Epoch 6/15
20/20 [==============================] - 130s 6s/step - loss: 0.5530 - accuracy: 0.7055 - val_loss: 0.5819 - val_accuracy: 0.69
70
Epoch 7/15
20/20 [==============================] - 134s 7s/step - loss: 0.5224 - accuracy: 0.7335 - val_loss: 0.5795 - val_accuracy: 0.68
90
Epoch 8/15
20/20 [==============================] - 132s 7s/step - loss: 0.4790 - accuracy: 0.7700 - val_loss: 0.5652 - val_accuracy: 0.70
30
Epoch 9/15
20/20 [==============================] - 134s 7s/step - loss: 0.4496 - accuracy: 0.7880 - val_loss: 0.5818 - val_accuracy: 0.68
80
Epoch 10/15
20/20 [==============================] - 143s 7s/step - loss: 0.4012 - accuracy: 0.8190 - val_loss: 0.5791 - val_accuracy: 0.72
20
Epoch 11/15
20/20 [==============================] - 139s 7s/step - loss: 0.3578 - accuracy: 0.8410 - val_loss: 0.6170 - val_accuracy: 0.71
20
Epoch 12/15
20/20 [==============================] - 141s 7s/step - loss: 0.3277 - accuracy: 0.8560 - val_loss: 0.7458 - val_accuracy: 0.68
90
Epoch 13/15
20/20 [==============================] - 143s 7s/step - loss: 0.3149 - accuracy: 0.8620 - val_loss: 0.6242 - val_accuracy: 0.72
80
Epoch 14/15
20/20 [==============================] - 146s 7s/step - loss: 0.2331 - accuracy: 0.9110 - val_loss: 0.6764 - val_accuracy: 0.73
10
Epoch 15/15
20/20 [==============================] - 149s 7s/step - loss: 0.1484 - accuracy: 0.9405 - val_loss: 0.6955 - val_accuracy: 0.74
70
```

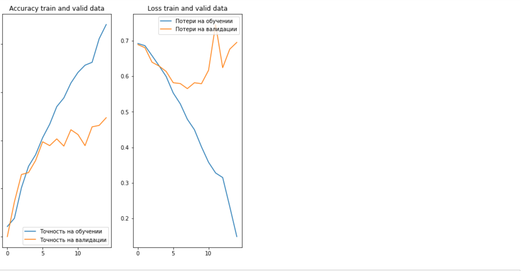
In [12]:
```
# Визуализация результатов тренировки
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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.savefig('./foo.png')
plt.show()
```



In [13]:
```
# Расхождение на тренировочных и валидационных данных говорит о переобучении сети и недостатке исходных данных
```