In [1]:
```
# Импорт всех библиотек
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 numpy
import numpy as np
import matplotlib.pyplot as plt
import glob
import shutil
```

In [2]:
```
# Загрузка и распаковка данных
_URL = "https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz"
zip_file = tf.keras.utils.get_file(origin=_URL, fname="flower_photos.tgz", extract=True)
base_dir = os.path.join(os.path.dirname(zip_file), 'flower_photos')

# Создание списка имен классов
classes = ['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
print(zip_file, base_dir)
```

In [3]:
```
# Вывести количество цветов в каждой папке

for cl in classes:
    img_path = os.path.join(base_dir, cl)
    images = glob.glob(img_path + '/*.jpg')
    print("{}: {} изображений".format(cl, len(images)))
    train, val = images[:round(len(images)*0.8)], images[round(len(images)*0.8):]

    for t in train:
        if not os.path.exists(os.path.join(base_dir, 'train', cl)):
            os.makedirs(os.path.join(base_dir, 'train', cl))
        shutil.move(t, os.path.join(base_dir, 'train', cl))

    for v in val:
        if not os.path.exists(os.path.join(base_dir, 'val', cl)):
            os.makedirs(os.path.join(base_dir, 'val', cl))
        shutil.move(v, os.path.join(base_dir, 'val', cl))

# Создание директорий
train_dir = os.path.join(base_dir, 'train')
val_dir = os.path.join(base_dir, 'val')
```

```
daisy: 633 изображений
dandelion: 898 изображений
roses: 641 изображений
sunflowers: 699 изображений
tulips: 799 изображений
```

In [4]:
```
# Функция, которая будет выводить изображения (с случайным преобразованием). 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 [5]:
```
BATCH_SIZE = 100 # Количество тренировочных изображений для обработки перед обновлением параметров модели
IMG_SHAPE = 150 # Квадратное 150x150 к которой будет приведено каждое изображение

train_image_generator = ImageDataGenerator(rescale=1./255, rotation_range=45, zoom_range=0.2,
                                           horizontal_flip=True, fill_mode='nearest')
validation_image_generator = ImageDataGenerator(rescale=1./255)

# Загрузка данных
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')

# Валидационные данные
val_data_gen = validation_image_generator.flow_from_directory(batch_size=BATCH_SIZE,
                                                              directory=val_dir,
                                                              shuffle=False,
                                                              target_size=(IMG_SHAPE,IMG_SHAPE),
                                                              class_mode='binary')

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

```
Found 2939 images belonging to 5 classes.
Found 735 images belonging to 5 classes.
```


In [6]:
```
# Создание модели
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(IMG_SHAPE, IMG_SHAPE, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, 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, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 17, 17, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 18496)             0
_________________________________________________________________
dense (Dense)                (None, 512)               9470464
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 2565
=================================================================
Total params: 9,496,613
Trainable params: 9,496,613
Non-trainable params: 0
_________________________________________________________________
```

In [7]:
```
# Тренировка модели
EPOCHS = 50
history = model.fit_generator( # Используется fit_generator вместо обычного fit
    train_data_gen,
    steps_per_epoch=int(np.ceil(2900 / float(BATCH_SIZE))),
    epochs=EPOCHS,
    validation_data=val_data_gen,
    validation_steps=int(np.ceil(700 / float(BATCH_SIZE)))
)
```

```
WARNING:tensorflow:From <ipython-input-7-585e7b5d7575>:8: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/50
29/29 [==============================] - 22s 775ms/step - loss: 1.6875 - accuracy: 0.2910 - val_loss: 1.2683 - val_accuracy: 0.4773
Epoch 2/50
29/29 [==============================] - 23s 793ms/step - loss: 1.1978 - accuracy: 0.4988 - val_loss: 1.1872 - val_accuracy: 0.5186
Epoch 3/50
29/29 [==============================] - 23s 792ms/step - loss: 1.0707 - accuracy: 0.5567 - val_loss: 1.2199 - val_accuracy: 0.5200
Epoch 4/50
29/29 [==============================] - 23s 785ms/step - loss: 1.0098 - accuracy: 0.6021 - val_loss: 1.0210 - val_accuracy: 0.5788
Epoch 5/50
29/29 [==============================] - 23s 781ms/step - loss: 0.9626 - accuracy: 0.6187 - val_loss: 0.9240 - val_accuracy: 0.6243
Epoch 6/50
29/29 [==============================] - 22s 771ms/step - loss: 0.8882 - accuracy: 0.6554 - val_loss: 0.9782 - val_accuracy: 0.6371
Epoch 7/50
29/29 [==============================] - 23s 780ms/step - loss: 0.8699 - accuracy: 0.6614 - val_loss: 0.8785 - val_accuracy: 0.6557
Epoch 8/50
29/29 [==============================] - 22s 774ms/step - loss: 0.8345 - accuracy: 0.6702 - val_loss: 0.9172 - val_accuracy: 0.6314
Epoch 9/50
29/29 [==============================] - 22s 764ms/step - loss: 0.7896 - accuracy: 0.6917 - val_loss: 0.8467 - val_accuracy: 0.6829
Epoch 10/50
29/29 [==============================] - 23s 782ms/step - loss: 0.7825 - accuracy: 0.6882 - val_loss: 0.8723 - val_accuracy: 0.6643
Epoch 11/50
29/29 [==============================] - 23s 778ms/step - loss: 0.7589 - accuracy: 0.6981 - val_loss: 0.8189 - val_accuracy: 0.6843
Epoch 12/50
29/29 [==============================] - 23s 781ms/step - loss: 0.7157 - accuracy: 0.7185 - val_loss: 0.7717 - val_accuracy: 0.7000
Epoch 13/50
29/29 [==============================] - 22s 769ms/step - loss: 0.7194 - accuracy: 0.7168 - val_loss: 0.7563 - val_accuracy: 0.7114
Epoch 14/50
29/29 [==============================] - 23s 778ms/step - loss: 0.6787 - accuracy: 0.7333 - val_loss: 0.7619 - val_accuracy: 0.7029
Epoch 15/50
29/29 [==============================] - 23s 783ms/step - loss: 0.6696 - accuracy: 0.7397 - val_loss: 0.7603 - val_accuracy: 0.7257
Epoch 16/50
29/29 [==============================] - 23s 778ms/step - loss: 0.6618 - accuracy: 0.7393 - val_loss: 0.7223 - val_accuracy: 0.7257
Epoch 17/50
29/29 [==============================] - 22s 765ms/step - loss: 0.6358 - accuracy: 0.7545 - val_loss: 0.7203 - val_accuracy: 0.7271
Epoch 18/50
29/29 [==============================] - 22s 773ms/step - loss: 0.6219 - accuracy: 0.7577 - val_loss: 0.7089 - val_accuracy: 0.7371
Epoch 19/50
29/29 [==============================] - 23s 772ms/step - loss: 0.6006 - accuracy: 0.7723 - val_loss: 0.6862 - val_accuracy: 0.7357
Epoch 20/50
29/29 [==============================] - 22s 768ms/step - loss: 0.5671 - accuracy: 0.7813 - val_loss: 0.7231 - val_accuracy: 0.7143
Epoch 21/50
29/29 [==============================] - 22s 761ms/step - loss: 0.5888 - accuracy: 0.7795 - val_loss: 0.7268 - val_accuracy: 0.7286
Epoch 22/50
29/29 [==============================] - 23s 765ms/step - loss: 0.5439 - accuracy: 0.7887 - val_loss: 0.6931 - val_accuracy: 0.7543
Epoch 23/50
29/29 [==============================] - 23s 778ms/step - loss: 0.5390 - accuracy: 0.7898 - val_loss: 0.6796 - val_accuracy: 0.7586
Epoch 24/50
29/29 [==============================] - 23s 790ms/step - loss: 0.5250 - accuracy: 0.7975 - val_loss: 0.6785 - val_accuracy: 0.7500
Epoch 25/50
29/29 [==============================] - 23s 789ms/step - loss: 0.5158 - accuracy: 0.8011 - val_loss: 0.8227 - val_accuracy: 0.6900
Epoch 26/50
29/29 [==============================] - 23s 805ms/step - loss: 0.5234 - accuracy: 0.7990 - val_loss: 0.6576 - val_accuracy: 0.7614
Epoch 27/50
29/29 [==============================] - 23s 786ms/step - loss: 0.5072 - accuracy: 0.8102 - val_loss: 0.7278 - val_accuracy: 0.7314
Epoch 28/50
29/29 [==============================] - 23s 803ms/step - loss: 0.4761 - accuracy: 0.8219 - val_loss: 0.7381 - val_accuracy: 0.7443
Epoch 29/50
29/29 [==============================] - 23s 805ms/step - loss: 0.4568 - accuracy: 0.8303 - val_loss: 0.7253 - val_accuracy: 0.7529
Epoch 30/50
29/29 [==============================] - 23s 800ms/step - loss: 0.4449 - accuracy: 0.8330 - val_loss: 0.7525 - val_accuracy: 0.7371
Epoch 31/50
29/29 [==============================] - 24s 816ms/step - loss: 0.4335 - accuracy: 0.8353 - val_loss: 0.7036 - val_accuracy: 0.7557
Epoch 32/50
29/29 [==============================] - 24s 818ms/step - loss: 0.4091 - accuracy: 0.8490 - val_loss: 0.7244 - val_accuracy: 0.7486
Epoch 33/50
29/29 [==============================] - 24s 818ms/step - loss: 0.4108 - accuracy: 0.8483 - val_loss: 0.8159 - val_accuracy: 0.7157
Epoch 34/50
29/29 [==============================] - 24s 819ms/step - loss: 0.4067 - accuracy: 0.8578 - val_loss: 0.7830 - val_accuracy: 0.7486
Epoch 35/50
29/29 [==============================] - 24s 826ms/step - loss: 0.4123 - accuracy: 0.8526 - val_loss: 0.7240 - val_accuracy: 0.7629
Epoch 36/50
29/29 [==============================] - 24s 827ms/step - loss: 0.3859 - accuracy: 0.8571 - val_loss: 0.6963 - val_accuracy: 0.7500
Epoch 37/50
29/29 [==============================] - 24s 826ms/step - loss: 0.3870 - accuracy: 0.8586 - val_loss: 0.7853 - val_accuracy: 0.7429
Epoch 38/50
29/29 [==============================] - 25s 848ms/step - loss: 0.3903 - accuracy: 0.8451 - val_loss: 0.7055 - val_accuracy: 0.7629
Epoch 39/50
29/29 [==============================] - 24s 826ms/step - loss: 0.3624 - accuracy: 0.8667 - val_loss: 0.7783 - val_accuracy: 0.7486
Epoch 40/50
29/29 [==============================] - 24s 828ms/step - loss: 0.3611 - accuracy: 0.8575 - val_loss: 0.9013 - val_accuracy: 0.7129
Epoch 41/50
29/29 [==============================] - 24s 818ms/step - loss: 0.3670 - accuracy: 0.8646 - val_loss: 0.8023 - val_accuracy: 0.7414
Epoch 42/50
29/29 [==============================] - 24s 834ms/step - loss: 0.3296 - accuracy: 0.8695 - val_loss: 0.7128 - val_accuracy: 0.7671
Epoch 43/50
29/29 [==============================] - 24s 834ms/step - loss: 0.3244 - accuracy: 0.8741 - val_loss: 0.8090 - val_accuracy: 0.7300
Epoch 44/50
29/29 [==============================] - 24s 830ms/step - loss: 0.3312 - accuracy: 0.8751 - val_loss: 0.8088 - val_accuracy: 0.7243
Epoch 45/50
29/29 [==============================] - 24s 831ms/step - loss: 0.3022 - accuracy: 0.8959 - val_loss: 0.9023 - val_accuracy: 0.7214
Epoch 46/50
29/29 [==============================] - 24s 830ms/step - loss: 0.3297 - accuracy: 0.8741 - val_loss: 0.8356 - val_accuracy: 0.7300
Epoch 47/50
29/29 [==============================] - 24s 822ms/step - loss: 0.3063 - accuracy: 0.8885 - val_loss: 0.7924 - val_accuracy: 0.7500
Epoch 48/50
29/29 [==============================] - 24s 833ms/step - loss: 0.2937 - accuracy: 0.8970 - val_loss: 0.8410 - val_accuracy: 0.7429
Epoch 49/50
29/29 [==============================] - 24s 837ms/step - loss: 0.2885 - accuracy: 0.8977 - val_loss: 0.7918 - val_accuracy: 0.7586
Epoch 50/50
29/29 [==============================] - 24s 835ms/step - loss: 0.2642 - accuracy: 0.9051 - val_loss: 0.8605 - val_accuracy: 0.7586
```



```
Epoch 51/50
29/29 [==============================] - 24s 835ms/step - loss: 0.2611 - accuracy: 0.9017 - val_loss: 0.8681 - val_accuracy: 0.7571
Epoch 52/50
29/29 [==============================] - 24s 832ms/step - loss: 0.2619 - accuracy: 0.9023 - val_loss: 0.8225 - val_accuracy: 0.7743
Epoch 53/50
29/29 [==============================] - 24s 836ms/step - loss: 0.2668 - accuracy: 0.9016 - val_loss: 0.8411 - val_accuracy: 0.7786
Epoch 54/50
29/29 [==============================] - 24s 827ms/step - loss: 0.2382 - accuracy: 0.9097 - val_loss: 0.9231 - val_accuracy: 0.7600
Epoch 55/50
29/29 [==============================] - 24s 835ms/step - loss: 0.2082 - accuracy: 0.9256 - val_loss: 0.8348 - val_accuracy: 0.7786
Epoch 56/50
29/29 [==============================] - 24s 836ms/step - loss: 0.2316 - accuracy: 0.9188 - val_loss: 0.8932 - val_accuracy: 0.7614
Epoch 57/50
29/29 [==============================] - 24s 836ms/step - loss: 0.2168 - accuracy: 0.9249 - val_loss: 0.9192 - val_accuracy: 0.7586
Epoch 58/50
29/29 [==============================] - 24s 836ms/step - loss: 0.2054 - accuracy: 0.9266 - val_loss: 0.9715 - val_accuracy: 0.7500
Epoch 59/50
29/29 [==============================] - 24s 837ms/step - loss: 0.2076 - accuracy: 0.9228 - val_loss: 1.0385 - val_accuracy: 0.7529
Epoch 60/50
29/29 [==============================] - 24s 842ms/step - loss: 0.2206 - accuracy: 0.9159 - val_loss: 0.9451 - val_accuracy: 0.7614
Epoch 61/50
29/29 [==============================] - 24s 843ms/step - loss: 0.2040 - accuracy: 0.9288 - val_loss: 0.9189 - val_accuracy: 0.7671
Epoch 62/50
29/29 [==============================] - 24s 841ms/step - loss: 0.1798 - accuracy: 0.9365 - val_loss: 1.0837 - val_accuracy: 0.7529
Epoch 63/50
29/29 [==============================] - 24s 839ms/step - loss: 0.1733 - accuracy: 0.9340 - val_loss: 0.9587 - val_accuracy: 0.7629
Epoch 64/50
29/29 [==============================] - 24s 838ms/step - loss: 0.1913 - accuracy: 0.9270 - val_loss: 0.9363 - val_accuracy: 0.7614
Epoch 65/50
29/29 [==============================] - 24s 831ms/step - loss: 0.1687 - accuracy: 0.9411 - val_loss: 1.1045 - val_accuracy: 0.7500
Epoch 66/50
29/29 [==============================] - 24s 828ms/step - loss: 0.1914 - accuracy: 0.9319 - val_loss: 0.9351 - val_accuracy: 0.7500
Epoch 67/50
29/29 [==============================] - 24s 834ms/step - loss: 0.1868 - accuracy: 0.9354 - val_loss: 0.9981 - val_accuracy: 0.7614
Epoch 68/50
29/29 [==============================] - 24s 827ms/step - loss: 0.1547 - accuracy: 0.9458 - val_loss: 0.9634 - val_accuracy: 0.7600
Epoch 69/50
29/29 [==============================] - 24s 836ms/step - loss: 0.1555 - accuracy: 0.9439 - val_loss: 1.0190 - val_accuracy: 0.7486
Epoch 70/50
29/29 [==============================] - 24s 836ms/step - loss: 0.1717 - accuracy: 0.9383 - val_loss: 1.0340 - val_accuracy: 0.7414
Epoch 71/50
29/29 [==============================] - 24s 840ms/step - loss: 0.1514 - accuracy: 0.9448 - val_loss: 1.0559 - val_accuracy: 0.7529
Epoch 72/50
29/29 [==============================] - 24s 841ms/step - loss: 0.1239 - accuracy: 0.9559 - val_loss: 1.0394 - val_accuracy: 0.7657
Epoch 73/50
29/29 [==============================] - 24s 837ms/step - loss: 0.1391 - accuracy: 0.9533 - val_loss: 1.0874 - val_accuracy: 0.7629
Epoch 74/50
29/29 [==============================] - 24s 839ms/step - loss: 0.1247 - accuracy: 0.9552 - val_loss: 1.1760 - val_accuracy: 0.7443
Epoch 75/50
29/29 [==============================] - 24s 829ms/step - loss: 0.1398 - accuracy: 0.9517 - val_loss: 1.0588 - val_accuracy: 0.7557
Epoch 76/50
29/29 [==============================] - 24s 832ms/step - loss: 0.1175 - accuracy: 0.9612 - val_loss: 1.0965 - val_accuracy: 0.7729
Epoch 77/50
29/29 [==============================] - 24s 832ms/step - loss: 0.1314 - accuracy: 0.9528 - val_loss: 1.2626 - val_accuracy: 0.7529
Epoch 78/50
29/29 [==============================] - 24s 822ms/step - loss: 0.1284 - accuracy: 0.9517 - val_loss: 1.2416 - val_accuracy: 0.7529
Epoch 79/50
29/29 [==============================] - 24s 824ms/step - loss: 0.1491 - accuracy: 0.9489 - val_loss: 1.3302 - val_accuracy: 0.7357
Epoch 80/50
29/29 [==============================] - 24s 826ms/step - loss: 0.1570 - accuracy: 0.9439 - val_loss: 1.1115 - val_accuracy: 0.7486
```

In [8]:
```
# 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 [9]:
```
# Сохранение модели
model.save('birth_network_CNN_model.h5')
```