https://paperswithcode.com/dataset/oxford-102-flower

Oxford 102 Flower

oxford 102 flower это датасет составленный для задач классификации, состоящий из 102 категорий цветов, в основном встречающихся на территории Великобритании. Каждый класс состоит от 40 до 258 картинок класса.

Картинки имеют достаточно большое разрешение, различные ракурсы, цветовые палитры и освещение. Плюсом к этому есть категории которые имеют сильно отличающиеся друг от друга картинки и несколько похожих категорий.

Изначально датасет был сформирован для задачи классификации по большому количеству классов, которые достаточно похожи.

Визуализации

colouriso.jpg shapeiso.jpg

Решаемые задачи

Согласно *paperswithcode.com* датасет используется для следующих задач:

- Классификация изображений
- Генерация картинки по тексту
- Генерации картинок
- Инкрементальное обучение

Датасет представлен:

- 102 категории цветов (RGB картинки) + лейблы
- Маски сегментации

Однако маски нам не нужны, так что будем использовать версию датасета без них.

Статья

Big Transfer (BiT): General Visual Representation Learning (2019)

В статье предложено решение проблемы эффективности тренировки нейронных сетей на собственных семплах и упрощение гиперпараметров при тренировке посредством операции **Big Transfer** : Суть метода в тренировке нейронной сети на большом, хорошо изученном и модерированном датасете. Авторы, применяя сильно эскалированный претрейн и перенос параметров достигают хорошей точности на большом количестве датасетов.

На картинке видим как перенос перформанса с BiT-L, SOTA, ResNet-50 на конкретные задачи влияет на точность.

Screenshot 2022-12-26 at 19.46.29.png

Сама методика состоит из нескольких частей:

- Upstream Pre-train - тренировка на очень большом датасете изначальной модели
- Transfer to downstream tasks - выбираем только важные гиперпараметры и тренируем модель на данных для таски с большим количеством препроцессинга - обрез картинок, перевороты.

Импорт и загрузка датасета

Импорты и загрузка

In [167…

```
import tensorflow as tf
import tensorflow_hub as hub

import time

from PIL import Image, ImageStat
import requests
from io import BytesIO

import matplotlib.pyplot as plt
import numpy as np

import glob, os
import pathlib

import plotly as xplt
import plotly.express as px
import plotly.graph_objects as go
import datetime
```

Определим лейблы для переноса

In [98]:

```
tf_flowers_labels = ['dandelion', 'daisy', 'tulips', 'sunflowers', 'roses']
```

Возьмем готовые веса модели ResNet50

In [3]:

```
model_url = "https://tfhub.dev/google/bit/m-r50x1/1"
module = hub.KerasLayer(model_url)
```

```
Metal device set to: Apple M1 Pro
```

```
2022-12-26 14:01:16.990700: I tensorflow/core/common_runtime/pluggable_devi
ce/pluggable_device_factory.cc:305] Could not identify NUMA node of platfor
m GPU ID 0, defaulting to 0. Your kernel may not have been built with NUMA
support.
2022-12-26 14:01:16.990917: I tensorflow/core/common_runtime/pluggable_devi
ce/pluggable_device_factory.cc:271] Created TensorFlow device (/job:localho
st/replica:0/task:0/device:GPU:0 with 0 MB memory) -> physical PluggableDev
ice (device: 0, name: METAL, pci bus id: <undefined>)
```

Скачаем и разархивируем датасет

In [4]:
```
data_dir = tf.keras.utils.get_file(origin='https://storage.googleapis.com/do
                                   fname='flower_photos', untar=True)
data_dir = pathlib.Path(data_dir)
```

Хелпер-функции для датасет инфо

In [176…
```
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from statistics import mean
import cv2

def plotdistribhw(imgs):
    df = {'width':[],'height':[]}
    for img in imgs:
        image = Image.open(img)
        df['width'].append(image.size[0])
        df['height'].append(image.size[1])
    fig = make_subplots(rows=1, cols=2)
    fig.add_trace(go.Histogram(x = df['width'], name = 'Распределение по шир
    fig.add_trace(go.Histogram(x = df['height'], name = 'Распределение по вы
    fig.update_layout(title_text='Размерности картинок')
    fig.show()

def plotdistribcolor(imgs):
    df = {'R':[],'G':[],'B':[]}
    for img in imgs:
        image = Image.open(img)
        stat = ImageStat.Stat(image)
        df['R'].append(stat.mean[0])
        df['G'].append(stat.mean[1])
        df['B'].append(stat.mean[2])
    fig = go.Figure()
    fig.add_trace(go.Histogram(x = df['R'], name = 'Red Channel'))
    fig.add_trace(go.Histogram(x = df['G'], name = 'Green Channel'))
    fig.add_trace(go.Histogram(x = df['B'], name = 'Blue Channel'))
    fig.update_layout(title_text='Распределение по цветовым каналам')
    fig.show()

def datasetinfo(data_dir, tf_flowers_labels):
    imgs = glob.glob(f'{data_dir}/**/*.jpg', recursive=True)

    fig = make_subplots(rows=1, cols=5)
    for i, img in enumerate(imgs[55:60]):
```

```
        x = Image.open(img)
        fig.add_trace(go.Image(z=x), 1, i+1)
    fig.update_layout(
    height=300,
    title_text='Picture examples from dataset'
    )
    fig.show()

    print(f'Количество изображений: {len(imgs)}\nКоличество классов (урезанн

    plotdistribhw(imgs)

    plotdistribcolor(imgs)
```

Информация о датасете

In [71]:
```
print(data_dir)
```

```
/Users/ivanskvortsov/.keras/datasets/flower_photos
```

In [177…
```
datasetinfo(data_dir, tf_flowers_labels)
```

```
Количество изображений: 3670
Количество классов (урезанная версия): 5
```

BiT-M R50x1

Feature extraction модель построенная на архитектуре ResNet50-v2, тренированная на большом количестве классов ImageNet-21k - датасет с 14 миллионами картинок и 21.843 классами. Выводом являются 2048-размерные фича векторы, модель используется для файн-тюнинга на новой задаче.

Ссылка: ResNet

Готовые леммы для ResNet50

In [8]:
```
!wget https://storage.googleapis.com/bit_models/ilsvrc2012_wordnet_lemmas.tx

imagenet_int_to_str = {}

with open('ilsvrc2012_wordnet_lemmas.txt', 'r') as f:
  for i in range(1000):
    row = f.readline()
    row = row.rstrip()
    imagenet_int_to_str.update({i: row})
```

```
--2022-12-26 14:01:48--  https://storage.googleapis.com/bit_models/ilsvrc20
12_wordnet_lemmas.txt
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.205.12
8, 64.233.165.128, 173.194.73.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.205.12
8|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 21675 (21K) [text/plain]
Saving to: ‘ilsvrc2012_wordnet_lemmas.txt.3’

ilsvrc2012_wordnet_ 100%[===================>]  21,17K  --.-KB/s    in 0,02
s

2022-12-26 14:01:48 (1,04 MB/s) - ‘ilsvrc2012_wordnet_lemmas.txt.3’ saved
[21675/21675]
```

Дообучение

Функции-хелперы для загрузки датасета и небольшой предобработки

In [9]:
```python
IMG_HEIGHT = 224
IMG_WIDTH = 224

CLASS_NAMES = tf_flowers_labels  # from plotting helper functions above
NUM_CLASSES = len(CLASS_NAMES)
num_examples = len(list(data_dir.glob('*/*.jpg')))

def get_label(file_path):
  # convert the path to a list of path components
  parts = tf.strings.split(file_path, os.path.sep)
  # The second to last is the class-directory

  return tf.where(parts[-2] == CLASS_NAMES)[0][0]

def decode_img(img):
  # convert the compressed string to a 3D uint8 tensor
  img = tf.image.decode_jpeg(img, channels=3)
  return img

def process_path(file_path):
  label = get_label(file_path)
  # load the raw data from the file as a string
  img = tf.io.read_file(file_path)
  img = decode_img(img)
  features = {'image': img, 'label': label}
  return features

list_ds = tf.data.Dataset.list_files(str(data_dir/'*/*'))
ds = list_ds.map(process_path, num_parallel_calls=tf.data.experimental.AUTOT
```

Непосредственно препроцессинг

In [10]:
```python
def preprocess_image(image):
```

```
  image = np.array(image)
  # reshape into shape [batch_size, height, width, num_channels]
  img_reshaped = tf.reshape(image, [1, image.shape[0], image.shape[1], image
  # Use `convert_image_dtype` to convert to floats in the [0,1] range.
  image = tf.image.convert_image_dtype(img_reshaped, tf.float32)
  return image

def load_image_from_url(url):
  """Returns an image with shape [1, height, width, num_channels]."""
  response = requests.get(url)
  image = Image.open(BytesIO(response.content))
  image = preprocess_image(image)
  return image
```

Хелпер функции

In [11]:

```
# Show the MAX_PREDS highest scoring labels:
MAX_PREDS = 5
# Do not show labels with lower score than this:
MIN_SCORE = 0.8

def show_preds(logits, image, correct_flowers_label=None, tf_flowers_logits=

  if len(logits.shape) > 1:
    logits = tf.reshape(logits, [-1])

  fig, axes = plt.subplots(1, 2, figsize=(7, 4), squeeze=False)

  ax1, ax2 = axes[0]

  ax1.axis('off')
  ax1.imshow(image)
  if correct_flowers_label is not None:
    ax1.set_title(tf_flowers_labels[correct_flowers_label])
  classes = []
  scores = []
  logits_max = np.max(logits)
  softmax_denominator = np.sum(np.exp(logits - logits_max))
  for index, j in enumerate(np.argsort(logits)[-MAX_PREDS::][::-1]):
    score = 1.0/(1.0 + np.exp(-logits[j]))
    if score < MIN_SCORE: break
    if not tf_flowers_logits:
      # predicting in imagenet label space
      classes.append(imagenet_int_to_str[j])
    else:
      # predicting in tf_flowers label space
      classes.append(tf_flowers_labels[j])
    scores.append(np.exp(logits[j] - logits_max)/softmax_denominator*100)

  ax2.barh(np.arange(len(scores)) + 0.1, scores)
  ax2.set_xlim(0, 100)
  ax2.set_yticks(np.arange(len(scores)))
  ax2.yaxis.set_ticks_position('right')
  ax2.set_yticklabels(classes, rotation=0, fontsize=14)
  ax2.invert_xaxis()
```

```
  ax2.invert_yaxis()
  ax2.set_xlabel('Prediction probabilities', fontsize=11)
```

In [12]:
```
print(data_dir)
```

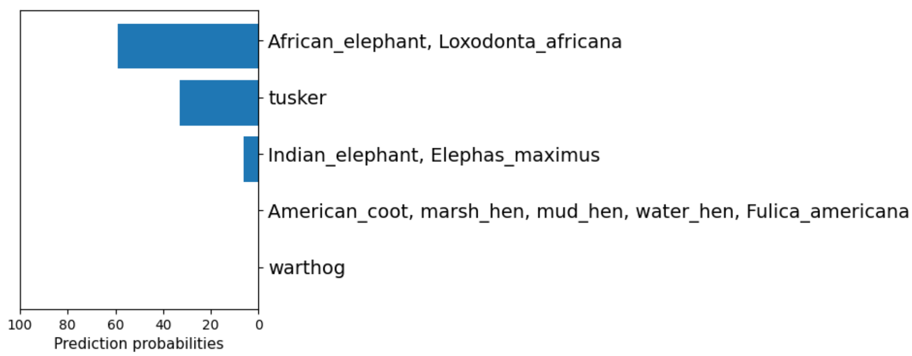
```
/Users/ivanskvortsov/.keras/datasets/flower_photos
```

Showcase

Как работает модель на любой картинке из интернета

In [13]:
```
model_url = "https://tfhub.dev/google/bit/m-r50x1/ilsvrc2012_classification/
imagenet_module = hub.KerasLayer(model_url)
```

In [16]:
```
# Load image (image provided is CC0 licensed)
img_url = "http://images6.fanpop.com/image/photos/34700000/Grey-Elephant-col
image = load_image_from_url(img_url)

# Run model on image
logits = imagenet_module(image)

# Show image and predictions
show_preds(logits, image[0])
```



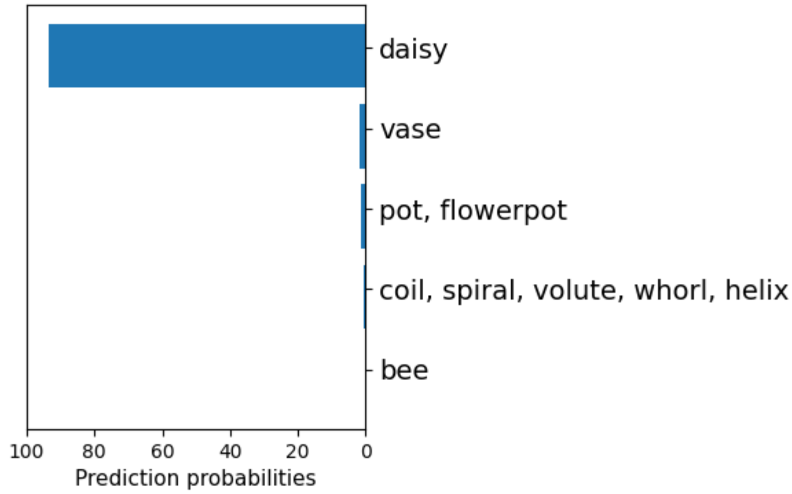
Как работает модель на данных из датасета. Модель иногда понимает что видит цветок, однако понимания какой именно - нет, много путает с другими объектами.

In [17]:
```
train_split = 0.9
num_train = int(train_split * num_examples)
ds_train = ds.take(num_train)
ds_test = ds.skip(num_train)

DATASET_NUM_TRAIN_EXAMPLES = num_examples
```

In [18]:
```
for features in ds_train.take(1):
  image = features['image']
  image = preprocess_image(image)

  # Run model on image
  logits = imagenet_module(image)
```

```
# Show image and predictions
show_preds(logits, image[0], correct_flowers_label=features['label'].numpy
```

```
2022-12-26 14:02:57.453063: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enab
led.
```



Перенос (дообучение) на нашем датасете

Гиперпараметры и прочее

In [19]:
```
# Add new head to the BiT model

class MyBiTModel(tf.keras.Model):
  """BiT with a new head."""

  def __init__(self, num_classes, module):
    super().__init__()

    self.num_classes = num_classes
    self.head = tf.keras.layers.Dense(num_classes, kernel_initializer='zeros
    self.bit_model = module

  def call(self, images):
    # No need to cut head off since we are using feature extractor model
    bit_embedding = self.bit_model(images)
    return self.head(bit_embedding)

model = MyBiTModel(num_classes=NUM_CLASSES, module=module)
```

In [20]:
```
IMAGE_SIZE = "=\u003C96x96 px" #@param ["=<96x96 px","> 96 x 96 px"]
DATASET_SIZE = "\u003C20k examples" #@param ["<20k examples", "20k-500k exam

if IMAGE_SIZE == "=<96x96 px":
  RESIZE_TO = 160
  CROP_TO = 128
else:
  RESIZE_TO = 512
```

```
  CROP_TO = 480

if DATASET_SIZE == "<20k examples":
  SCHEDULE_LENGTH = 500
  SCHEDULE_BOUNDARIES = [200, 300, 400]
elif DATASET_SIZE == "20k-500k examples":
  SCHEDULE_LENGTH = 10000
  SCHEDULE_BOUNDARIES = [3000, 6000, 9000]
else:
  SCHEDULE_LENGTH = 20000
  SCHEDULE_BOUNDARIES = [6000, 12000, 18000]
```

In [21]:

```
# Preprocessing helper functions

# Create data pipelines for training and testing:
BATCH_SIZE = 512
SCHEDULE_LENGTH = SCHEDULE_LENGTH * 512 / BATCH_SIZE

STEPS_PER_EPOCH = 10

def cast_to_tuple(features):
  return (features['image'], features['label'])

def preprocess_train(features):
  # Apply random crops and horizontal flips for all tasks
  # except those for which cropping or flipping destroys the label semantics
  # (e.g. predict orientation of an object)
  features['image'] = tf.image.random_flip_left_right(features['image'])
  features['image'] = tf.image.resize(features['image'], [RESIZE_TO, RESIZE_
  features['image'] = tf.image.random_crop(features['image'], [CROP_TO, CROP
  features['image'] = tf.cast(features['image'], tf.float32) / 255.0
  return features

def preprocess_test(features):
  features['image'] = tf.image.resize(features['image'], [RESIZE_TO, RESIZE_
  features['image'] = tf.cast(features['image'], tf.float32) / 255.0
  return features

pipeline_train = (ds_train
                  .shuffle(10000)
                  .repeat(int(SCHEDULE_LENGTH * BATCH_SIZE / DATASET_NUM_TRA
                  .map(preprocess_train, num_parallel_calls=8)
                  .batch(BATCH_SIZE)
                  .map(cast_to_tuple)  # for keras model.fit
                  .prefetch(2))

pipeline_test = (ds_test.map(preprocess_test, num_parallel_calls=1)
                  .map(cast_to_tuple)  # for keras model.fit
                  .batch(BATCH_SIZE)
                  .prefetch(2))
```

In [22]:

```
# Define optimiser and loss

lr = 0.003 * BATCH_SIZE / 512
```

```
# Decay learning rate by a factor of 10 at SCHEDULE_BOUNDARIES.
lr_schedule = tf.keras.optimizers.schedules.PiecewiseConstantDecay(boundarie
                                                                   values=[l
optimizer = tf.keras.optimizers.SGD(learning_rate=lr_schedule, momentum=0.9)

loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

Обучение

In [27]:
```
%load_ext tensorboard
```

In [32]:
```
# model.compile(optimizer=optimizer,
#               loss=loss_fn,
#               metrics=['accuracy'])

# log_dir = "./logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S"
# tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir)#, h

# # histogram_freq=1 - на маке полностью сломано (рассчет гистограммы каждую

# # Fine-tune model
# history = model.fit(
#     pipeline_train,
#     batch_size=BATCH_SIZE,
#     steps_per_epoch=STEPS_PER_EPOCH,
#     epochs= 10,
#     validation_data=pipeline_test,
#     callbacks=[tensorboard_callback]
# )
```

```
Epoch 1/10
2022-12-26 14:40:56.799844: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enab
led.
10/10 [==============================] - ETA: 0s - loss: 0.0626 - accuracy:
0.9812
2022-12-26 14:41:36.350812: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enab
led.
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_test_f
unction.<locals>.test_function at 0x2e6ce8ee0> triggered tf.function retrac
ing. Tracing is expensive and the excessive number of tracings could be due
to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with
different shapes, (3) passing Python objects instead of tensors. For (1), p
lease define your @tf.function outside of the loop. For (2), @tf.function h
as reduce_retracing=True option that can avoid unnecessary retracing. For
(3), please refer to https://www.tensorflow.org/guide/function#controlling_
retracing and https://www.tensorflow.org/api_docs/python/tf/function for  m
ore details.
```

```
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_test_f
unction.<locals>.test_function at 0x2e6ce8ee0> triggered tf.function retrac
ing. Tracing is expensive and the excessive number of tracings could be due
to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with
different shapes, (3) passing Python objects instead of tensors. For (1), p
lease define your @tf.function outside of the loop. For (2), @tf.function h
as reduce_retracing=True option that can avoid unnecessary retracing. For
(3), please refer to https://www.tensorflow.org/guide/function#controlling_
retracing and https://www.tensorflow.org/api_docs/python/tf/function for  m
ore details.
10/10 [==============================] - 59s 5s/step - loss: 0.0626 - accur
acy: 0.9812 - val_loss: 0.0389 - val_accuracy: 0.9891
Epoch 2/10
10/10 [==============================] - 25s 3s/step - loss: 0.0599 - accur
acy: 0.9826 - val_loss: 0.0699 - val_accuracy: 0.9782
Epoch 3/10
10/10 [==============================] - 26s 3s/step - loss: 0.0542 - accur
acy: 0.9822 - val_loss: 0.0503 - val_accuracy: 0.9891
Epoch 4/10
10/10 [==============================] - 24s 2s/step - loss: 0.0512 - accur
acy: 0.9863 - val_loss: 0.0231 - val_accuracy: 0.9973
Epoch 5/10
10/10 [==============================] - 26s 3s/step - loss: 0.0441 - accur
acy: 0.9852 - val_loss: 0.0296 - val_accuracy: 0.9864
Epoch 6/10
10/10 [==============================] - 23s 2s/step - loss: 0.0424 - accur
acy: 0.9863 - val_loss: 0.0431 - val_accuracy: 0.9837
Epoch 7/10
10/10 [==============================] - 24s 2s/step - loss: 0.0371 - accur
acy: 0.9873 - val_loss: 0.0158 - val_accuracy: 1.0000
Epoch 8/10
10/10 [==============================] - 26s 3s/step - loss: 0.0369 - accur
acy: 0.9904 - val_loss: 0.0278 - val_accuracy: 0.9946
Epoch 9/10
10/10 [==============================] - 23s 2s/step - loss: 0.0391 - accur
acy: 0.9875 - val_loss: 0.0458 - val_accuracy: 0.9864
Epoch 10/10
10/10 [==============================] - 22s 2s/step - loss: 0.0317 - accur
acy: 0.9906 - val_loss: 0.0469 - val_accuracy: 0.9864
```

In [33]:

```
%tensorboard --logdir logs/fit
```

In [36]:

```python
# #Save fine-tuned model as SavedModel
# export_module_dir = './tmp/my_saved_bit_model/'
# tf.saved_model.save(model, export_module_dir)
```

```
INFO:tensorflow:Assets written to: ./tmp/my_saved_bit_model/assets
```


```
INFO:tensorflow:Assets written to: ./tmp/my_saved_bit_model/assets
```

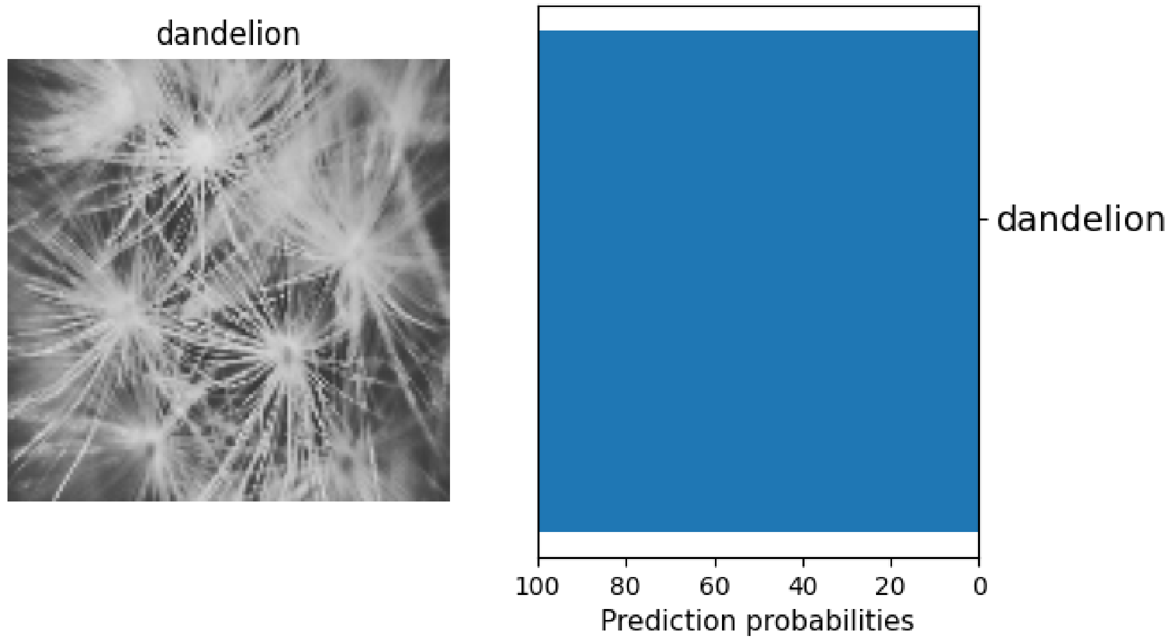
Загрузка из сохраненного стейта и showcase

In [37]:
```
saved_module = hub.KerasLayer(export_module_dir, trainable=True)
```

```
2022-12-26 14:49:02.825135: W tensorflow/core/common_runtime/graph_construc
tor.cc:805] Node 're_lu_48/PartitionedCall' has 1 outputs but the _output_s
hapes attribute specifies shapes for 2 outputs. Output shapes may be inaccu
rate.
2022-12-26 14:49:02.825908: W tensorflow/core/common_runtime/graph_construc
tor.cc:805] Node 'global_average_pooling2d/PartitionedCall' has 1 outputs b
ut the _output_shapes attribute specifies shapes for 4 outputs. Output shap
es may be inaccurate.
```

In [38]:
```
for features in ds_train.take(1):
  image = features['image']
  image = preprocess_image(image)
  image = tf.image.resize(image, [CROP_TO, CROP_TO])

  # Run model on image
  logits = saved_module(image)

  # Show image and predictions
  show_preds(logits, image[0], correct_flowers_label=features['label'].numpy
```

```
2022-12-26 14:49:07.590225: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enab
led.
```



Preprocessing ESRGAN

Enhanced Super Resolution GAN

Модель для обработки изображений в супер-разрешение.

Хелпер-функции

In [180…

```
def preprocess_image(image_path):
  """ Loads image from path and preprocesses to make it model ready
      Args:
        image_path: Path to the image file
  """
  hr_image = tf.image.decode_image(tf.io.read_file(image_path))
  # If PNG, remove the alpha channel. The model only supports
  # images with 3 color channels.
  if hr_image.shape[-1] == 4:
    hr_image = hr_image[...,:-1]
  hr_size = (tf.convert_to_tensor(hr_image.shape[:-1]) // 4) * 4
  hr_image = tf.image.crop_to_bounding_box(hr_image, 0, 0, hr_size[0], hr_si
  hr_image = tf.cast(hr_image, tf.float32)
  return tf.expand_dims(hr_image, 0)

def save_image(image, filename):
  """
    Saves unscaled Tensor Images.
    Args:
      image: 3D image tensor. [height, width, channels]
      filename: Name of the file to save.
  """
  if not isinstance(image, Image.Image):
    image = tf.clip_by_value(image, 0, 255)
    image = Image.fromarray(tf.cast(image, tf.uint8).numpy())
  image.save("%s.jpg" % filename)
  print("Saved as %s.jpg" % filename)


def plot_image(image, title=""):
  """
    Plots images from image tensors.
    Args:
      image: 3D image tensor. [height, width, channels].
      title: Title to display in the plot.
  """
  image = np.asarray(image)
  image = tf.clip_by_value(image, 0, 255)
  image = Image.fromarray(tf.cast(image, tf.uint8).numpy())
  plt.imshow(image)
  plt.axis("off")
  plt.title(title)
```

Применение ESRGAN

In [181…

```
ESR_SAVED_MODEL_PATH = "https://tfhub.dev/captain-pool/esrgan-tf2/1"
IMAGE_PATH="image.jpg"
```

Загрузим картинку из датасета и используем готовые веса модели ESRGAN из tfhub

In [182…

```
hr_image = preprocess_image(IMAGE_PATH)
plot_image(tf.squeeze(hr_image), title="Original Image")
save_image(tf.squeeze(hr_image), filename="Original Image")
```

```
Saved as Original Image.jpg
```

Original Image

In [183…

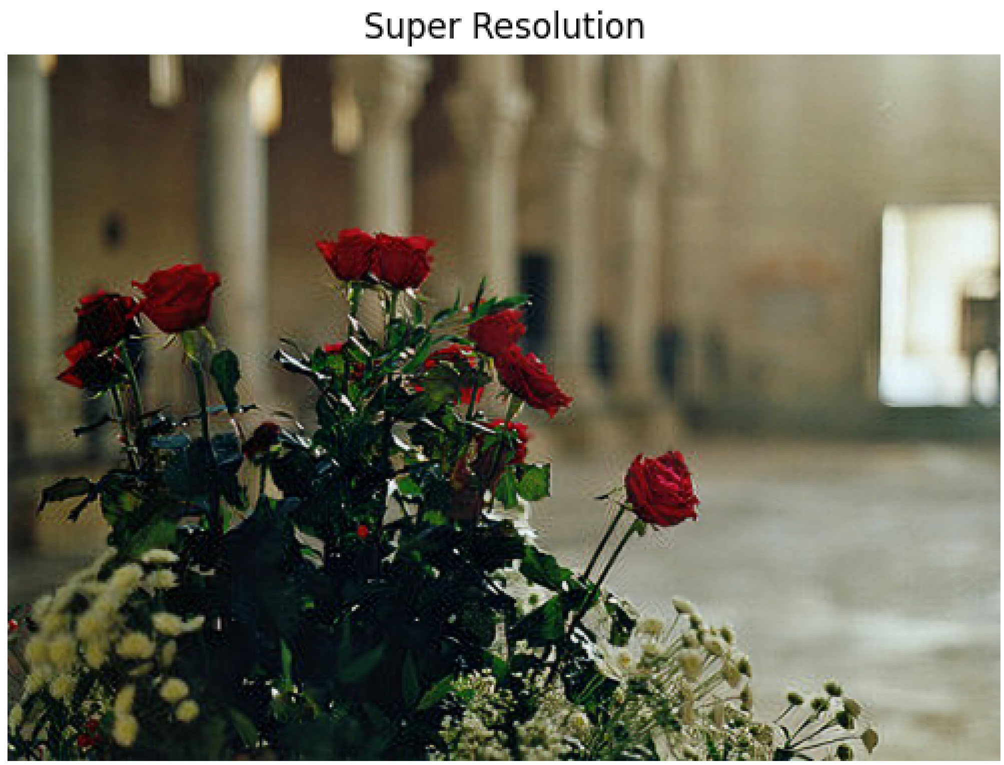
```
model = hub.load(ESR_SAVED_MODEL_PATH)
```

Пропустим это изображение через ESRGAN, время потраченное на одну картинку достаточно мало чтобы обработать все изображения в датасете.

In [185…

```
start = time.time()
fake_image = model(hr_image)
fake_image = tf.squeeze(fake_image)
print("Time Taken: %f" % (time.time() - start))
plot_image(tf.squeeze(fake_image), title="Super Resolution")
save_image(tf.squeeze(fake_image), filename="Super Resolution")
```

```
Time Taken: 0.193167
Saved as Super Resolution.jpg
```



Выводы

В курсовой был изучен и применен датасет oxford102. На датасете была дообучена модель ResNet50, для улучшения изображений была применена модель ESRGAN, метрики обучения выведены в TensorBoard. Построены графики статистики датасета в Plotly.

Для принта в пдф

In [186…

```
!export PATH=/Library/TeX/texbin:$PATH
```

In [ ]: