Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**Домашнее задание №1, 2 по дисциплине**

«Проектирование интеллектуальных систем»

**ИСПОЛНИТЕЛЬ:**

Куликов А. Б.

Группа ИУ5-22М

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Москва 2020

# Цель работы

Научиться использовать предобученные модели распознавания и преобразовывать датасеты в TFRecord.

### Предобработка изображений для использования в нейросети

|  |  |
| --- | --- |
|  | import tensorflow as tf  AUTOTUNE = tf.data.experimental.AUTOTUNE |

## Загрузка датасета

|  |  |
| --- | --- |
|  | import pathlib  data\_root\_orig = tf.keras.utils.get\_file(origin='https://storage.googleapis.com/download.tensorflow.org/example\_images/flower\_photos.tgz',  fname='flower\_photos', untar=True)  data\_root = pathlib.Path(data\_root\_orig)  print(data\_root) |

Downloading data from <https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz>

228818944/228813984 [==============================] - 12s 0us/step

/home/kbuilder/.keras/datasets/flower\_photos

**for item in data\_root.iterdir():**

**print(item)**

/home/kbuilder/.keras/datasets/flower\_photos/sunflowers

/home/kbuilder/.keras/datasets/flower\_photos/daisy

/home/kbuilder/.keras/datasets/flower\_photos/LICENSE.txt

/home/kbuilder/.keras/datasets/flower\_photos/roses

/home/kbuilder/.keras/datasets/flower\_photos/tulips

/home/kbuilder/.keras/datasets/flower\_photos/dandelion

**import random**

**all\_image\_paths = list(data\_root.glob('\*/\*'))**

**all\_image\_paths = [str(path) for path in all\_image\_paths]**

**random.shuffle(all\_image\_paths)**

**image\_count = len(all\_image\_paths)**

**image\_count**

3670

**all\_image\_paths[:10]**

['/home/kbuilder/.keras/datasets/flower\_photos/tulips/16732302779\_8aa56f255d\_n.jpg',

'/home/kbuilder/.keras/datasets/flower\_photos/sunflowers/21728822928\_9f6817325a\_n.jpg',

'/home/kbuilder/.keras/datasets/flower\_photos/daisy/8085329197\_41d53a21e2\_n.jpg',

'/home/kbuilder/.keras/datasets/flower\_photos/roses/14573732424\_1bb91e2e42\_n.jpg',

'/home/kbuilder/.keras/datasets/flower\_photos/dandelion/9726260379\_4e8ee66875\_m.jpg',

'/home/kbuilder/.keras/datasets/flower\_photos/sunflowers/19710076021\_f5bb162540.jpg',

'/home/kbuilder/.keras/datasets/flower\_photos/roses/15821959372\_518b9dcf57\_n.jpg',

'/home/kbuilder/.keras/datasets/flower\_photos/sunflowers/9699724719\_a8439cc0fd\_n.jpg',

'/home/kbuilder/.keras/datasets/flower\_photos/roses/6936225976\_a91b60d8c2\_m.jpg',

'/home/kbuilder/.keras/datasets/flower\_photos/sunflowers/1880606744\_23e3dc4f6b\_n.jpg']

### Изучение данных

**import os**

**attributions = (data\_root/"LICENSE.txt").open(encoding='utf-8').readlines()[4:]**

**attributions = [line.split(' CC-BY') for line in attributions]**

**attributions = dict(attributions)**

**import IPython.display as display**

**def caption\_image(image\_path):**

**image\_rel = pathlib.Path(image\_path).relative\_to(data\_root)**

**return "Image (CC BY 2.0) " + ' - '.join(attributions[str(image\_rel)].split(' - ')[:-1])**

**for n in range(3):**

**image\_path = random.choice(all\_image\_paths)**

**display.display(display.Image(image\_path))**

**print(caption\_image(image\_path))**

**print()**



### Определение меток для каждого изображения

**label\_names = sorted(item.name for item in data\_root.glob('\*/') if item.is\_dir())**

**label\_names**

['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']

**label\_to\_index = dict((name, index) for index, name in enumerate(label\_names))**

**label\_to\_index**

{'daisy': 0, 'dandelion': 1, 'roses': 2, 'sunflowers': 3, 'tulips': 4}

**all\_image\_labels = [label\_to\_index[pathlib.Path(path).parent.name]**

**for path in all\_image\_paths]**

**print("First 10 labels indices: ", all\_image\_labels[:10])**

First 10 labels indices: [4, 3, 0, 2, 1, 3, 2, 3, 2, 3]

**label\_to\_index = dict((name, index) for index,name in enumerate(label\_names))**

**label\_to\_index**

{'daisy': 0, 'dandelion': 1, 'roses': 2, 'sunflowers': 3, 'tulips': 4}

**all\_image\_labels = [label\_to\_index[pathlib.Path(path).parent.name]for path in all\_image\_paths]**

**print("First 10 labels indices: ", all\_image\_labels[:10])**

Первые 10 индексов изображений: [2, 1, 1, 0, 2, 3, 1, 4, 0, 2]

### Загрузка и форматирование изображений

|  |
| --- |
| **img\_path = all\_image\_paths[0]**  **img\_path**  '/home/kbuilder/.keras/datasets/flower\_photos/tulips/16732302779\_8aa56f255d\_n.jpg'  **img\_raw = tf.io.read\_file(img\_path)**  **print(repr(img\_raw)[:100]+"...")**  <tf.Tensor: shape=(), dtype=string, numpy=b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00\x00\x01\x00...  Преобразуем в тензор изображения:  **img\_tensor = tf.image.decode\_image(img\_raw)**  **print(img\_tensor.shape)**  **print(img\_tensor.dtype)**  (212, 320, 3)  <dtype: 'uint8'>  Изменим размер для нашей модели:  **img\_final = tf.image.resize(img\_tensor, [192, 192])**  **img\_final = img\_final/255.0**  **print(img\_final.shape)**  **print(img\_final.numpy().min())**  **print(img\_final.numpy().max())**  (192, 192, 3)  0.0  1.0  Обернём их в простые функции для будущего использования.  **def preprocess\_image(image):**  **image = tf.image.decode\_jpeg(image, channels=3)**  **image = tf.image.resize(image, [192, 192])**  **image /= 255.0 # normalize to [0,1] range**  **return image**  **def load\_and\_preprocess\_image(path):**  **image = tf.io.read\_file(path)**  **return preprocess\_image(image)**  **import matplotlib.pyplot as plt**  **image\_path = all\_image\_paths[0]**  **label = all\_image\_labels[0]**  **plt.imshow(load\_and\_preprocess\_image(img\_path))**  **plt.grid(False)**  **plt.xlabel(caption\_image(img\_path))**  **plt.title(label\_names[label].title())**  **print()** |

### Сборка tf.data.Dataset

**path\_ds = tf.data.Dataset.from\_tensor\_slices(all\_image\_paths)**

**print(path\_ds)**

<TensorSliceDataset shapes: (), types: tf.string>

**image\_ds = path\_ds.map(load\_and\_preprocess\_image, num\_parallel\_calls=AUTOTUNE)**

**import matplotlib.pyplot as plt**

**plt.figure(figsize=(8,8))**

**for n, image in enumerate(image\_ds.take(4)):**

**plt.subplot(2,2,n+1)**

**plt.imshow(image)**

**plt.grid(False)**

**plt.xticks([])**

**plt.yticks([])**

**plt.xlabel(caption\_image(all\_image\_paths[n]))**

**plt.show()**

**Датасет пар**

### label\_ds = tf.data.Dataset.from\_tensor\_slices(tf.cast(all\_image\_labels, tf.int64))

### for label in label\_ds.take(10):

### print(label\_names[label.numpy()])

tulips

sunflowers

daisy

roses

dandelion

sunflowers

roses

sunflowers

roses

sunflowers

Поскольку датасеты следуют в одном и том же порядке, мы можем просто собрать их вместе при помощи функции zip в набор данных пары (изображение, метка):

**image\_label\_ds = tf.data.Dataset.zip((image\_ds, label\_ds))**

`Shapes` и `types` нового датасета это кортежи размерностей и типов описывающие каждое поле:

**print(image\_label\_ds)**

<ZipDataset shapes: ((192, 192, 3), ()), types: (tf.float32, tf.int64)>

**ds = tf.data.Dataset.from\_tensor\_slices((all\_image\_paths, all\_image\_labels))**

**# Кортежи распаковываются в позиционные аргументы отображаемой функции**

**def load\_and\_preprocess\_from\_path\_label(path, label):**

**return load\_and\_preprocess\_image(path), label**

**image\_label\_ds = ds.map(load\_and\_preprocess\_from\_path\_label)**

**image\_label\_ds**

<MapDataset shapes: ((192, 192, 3), ()), types: (tf.float32, tf.int32)>

## Обучение существующей модели нейронной сети

## BATCH\_SIZE = 32

## # Установка размера буфера перемешивания, равного набору данных, гарантирует

## # полное перемешивание данных.

## ds = image\_label\_ds.shuffle(buffer\_size=image\_count)

## ds = ds.repeat()

## ds = ds.batch(BATCH\_SIZE)

## # `prefetch` позволяет датасету извлекать пакеты в фоновом режиме, во время обучения модели.

## ds = ds.prefetch(buffer\_size=AUTOTUNE)

## ds

<PrefetchDataset shapes: ((None, 192, 192, 3), (None,)), types: (tf.float32, tf.int32)>

### Передача набора данных в модель

**mobile\_net = tf.keras.applications.MobileNetV2(input\_shape=(192, 192, 3), include\_top=False)**

**mobile\_net.trainable=False**

Downloading data from <https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_192_no_top.h5>

9412608/9406464 [==============================] - 1s 0us/step

**def change\_range(image,label):**

**return 2\*image-1, label**

**keras\_ds = ds.map(change\_range)**

**image\_batch, label\_batch = next(iter(keras\_ds))**

**feature\_map\_batch = mobile\_net(image\_batch)**

**print(feature\_map\_batch.shape)**

(32, 6, 6, 1280)

**model = tf.keras.Sequential([**

**mobile\_net,**

**tf.keras.layers.GlobalAveragePooling2D(),**

**tf.keras.layers.Dense(len(label\_names), activation = 'softmax')])**

**logit\_batch = model(image\_batch).numpy()**

**print("min logit:", logit\_batch.min())**

**print("max logit:", logit\_batch.max())**

**print()**

**print("Shape:", logit\_batch.shape)**

min logit: 0.010125087

max logit: 0.83786047

Shape: (32, 5)

Скомпилируем модель чтобы описать процесс обучения:

**model.compile(optimizer=tf.keras.optimizers.Adam(),**

**loss='sparse\_categorical\_crossentropy',**

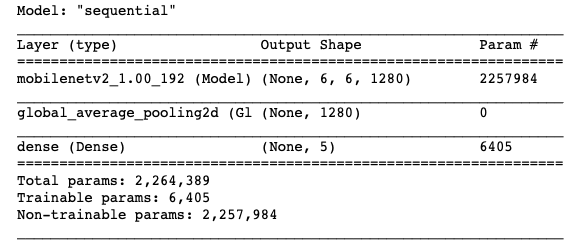
**metrics=["accuracy"])**

Есть две переменные для обучения - Dense weights и bias:

**len(model.trainable\_variables)**

2

**model.summary()**



Мы готовы обучать модель.

**steps\_per\_epoch=tf.math.ceil(len(all\_image\_paths)/BATCH\_SIZE).numpy()**

**steps\_per\_epoch**

115.0

**model.fit(ds, epochs=1, steps\_per\_epoch=3)**

3/3 [==============================] - 0s 31ms/step - loss: 1.8074 - accuracy: 0.3229

<tensorflow.python.keras.callbacks.History at 0x7f67a4bd3a90>

### Файл TFRecord

Сперва построим файл TFRecord из необработанных данных изображений:

**image\_ds = tf.data.Dataset.from\_tensor\_slices(all\_image\_paths).map(tf.io.read\_file)**

**tfrec = tf.data.experimental.TFRecordWriter('images.tfrec')**

**tfrec.write(image\_ds)**

Затем построим датасет, который прочитывает файл TFRecord и обрабатывает изображения с использованием функции `preprocess\_image`:

**image\_ds = tf.data.TFRecordDataset('images.tfrec').map(preprocess\_image)**

**keras\_ds = ds.map(change\_range)**

Объединим этот датасет с датасетом меток, чтобы получить пару из `(изображение, метка)`:

### ds = tf.data.Dataset.zip((image\_ds, label\_ds))

### ds = ds.apply(

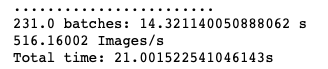
### tf.data.experimental.shuffle\_and\_repeat(buffer\_size=image\_count))

### ds=ds.batch(BATCH\_SIZE).prefetch(AUTOTUNE)

### ds

<PrefetchDataset shapes: ((None, 192, 192, 3), (None,)), types: (tf.float32, tf.int64)>

**timeit(ds)**



# Список литературы

1. MobileNet: меньше, быстрее, точнее – <https://habr.com/ru/post/352804/>
2. Transfer Learning Using Pretrained ConvNets – <https://www.tensorflow.org/alpha/tutorials/images/transfer_learning>
3. Natural Images – <https://www.kaggle.com/prasunroy/natural-images>
4. Load images with tf.data – <https://www.tensorflow.org/alpha/tutorials/load_data/images>