Обучить CHC с помощью Transfer Learning на датасете Food-101
Использовать тонкую настройку существующей предобученной модели и методы аугментации данных.

```
%tensorflow_version 2.x
```

```
Colab only includes TensorFlow 2.x; %tensorflow_version has no effect.
```

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

import tensorflow as tf
import tensorflow_datasets as tfds
from tensorflow.keras import layers


INP_SIZE = 160
NUM_EPOCHS = 5
BATCH_SIZE = 64
```

Загрузка датасета food101


```
.load(
    name = "food101",
    as_supervised=True,
    with_info=True,
    split=["train[:75%]", "train[75%:]"],
)
```

```
Downloading and preparing dataset 4.65 GiB (download: 4.65 GiB, generated: Unknown s
Dataset food101 downloaded and prepared to ~/tensorflow_datasets/food101/2.0.0. Subs
```

```
len(train_ds)
```

```
56812
```

```
num_classes = ds_info.features['label'].num_classes
print(num_classes)
```

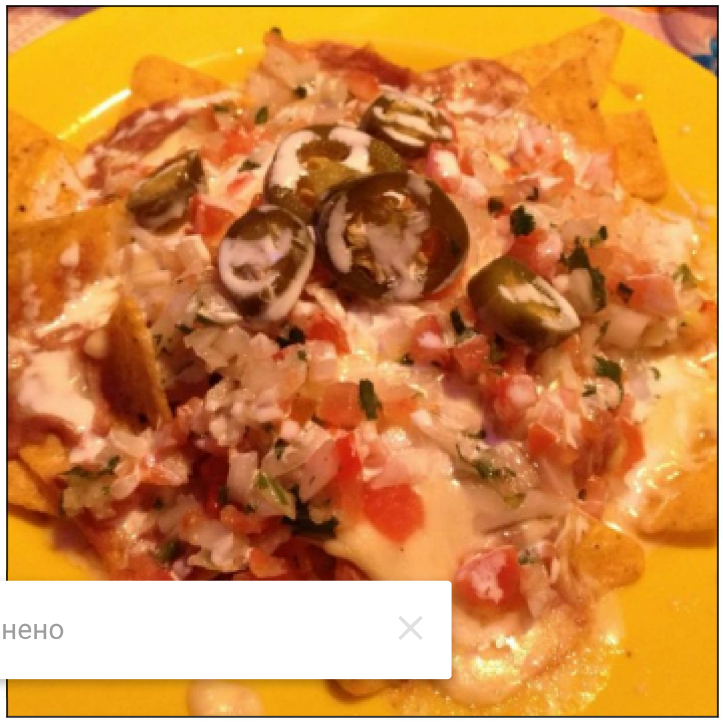
```
101
```

```
result_image = []
for i in iter(test_ds.take(30)):
  result_image.append(i)
```

```
result_image[2][0].shape
```

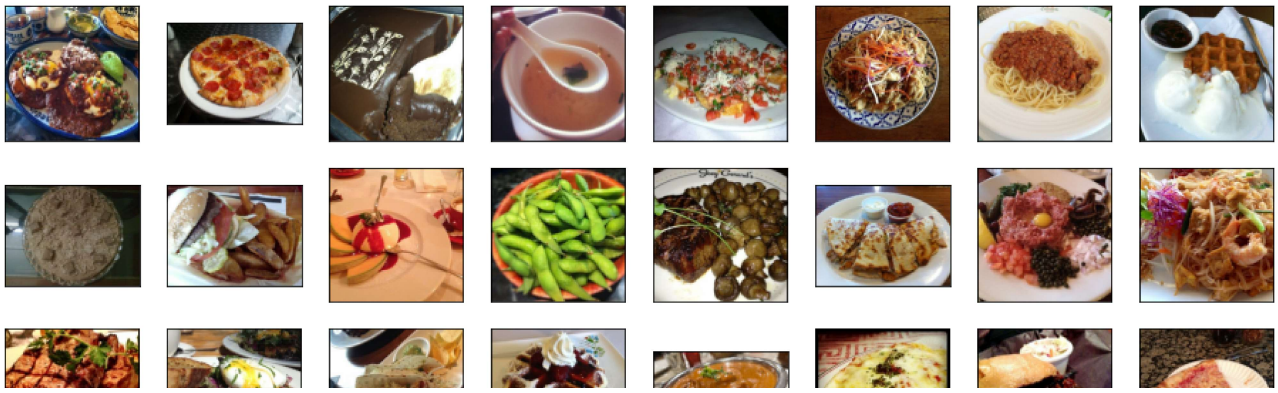
```
TensorShape([512, 512, 3])
```

```
image = result_image[3][0]
fig = plt.figure(figsize=(60, 30))
ax = fig.add_subplot(4, 8, 1)
ax.imshow(image / 255)
plt.xticks([]), plt.yticks([])
plt.show()
```




```
some_samples = [x[0] for x in iter(train_ds.take(32))]

fig = plt.figure(figsize=(16, 8))
for j in range(len(some_samples)):
    ax = fig.add_subplot(4, 8, j+1)
    ax.imshow(some_samples[j])
    plt.xticks([]), plt.yticks([])
plt.show()
```



```
# data_augmentation = tf.keras.Sequential([
#   layers.RandomFlip("horizontal_and_vertical"),
#   layers.RandomRotation(0.2),
# ])
```

```
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.RandomFlip('horizontal_and_vertical'),
  tf.keras.layers.RandomRotation(0.2),
])


some_samples_2 = [data_augmentation(x[0]) for x in iter(train_ds.take(32))]

fig = plt.figure(figsize=(16, 8))
for j in range(len(some_samples_2)):

    plt.xticks([]), plt.yticks([])
plt.show()
```

Сохранено

```
def prepare_aug(img, label):
      img = tf.cast(img,·tf.float32)/127.·-·1.
      img = data_augmentation(img)
      return tf.image.resize(img, (INP_SIZE, INP_SIZE)), label
def prepare(img, label):
      img = tf.cast(img, tf.float32)/127. - 1.
      return tf.image.resize(img, (INP_SIZE, INP_SIZE)), label

train_ds = train_ds.shuffle(buffer_size=1000)
train_ds_aug = train_ds.map(prepare_aug)
train_ds = train_ds.map(prepare)
# train = [train_ds_aug, train_ds]
train = train_ds.concatenate(train_ds_aug)
train = train.batch(BATCH_SIZE, drop_remainder=True)


test_ds = test_ds.shuffle(buffer_size=1000)
test_ds = test_ds.map(prepare)
test_ds = test_ds.batch(128, drop_remainder=True)
```

Подготовка модели CNN

```
EXP_NAME = 'transfer'
base_model = tf.keras.applications.MobileNetV2(
                                                ZE, 3),
    )
# base_model.trainable = True # Fine-tuning весов предобученной модели
base_model.trainable = False # Заморозка весов предобученной модели
```

Сохранено

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/m
9412608/9406464 [==============================] - 0s 0us/step
9420800/9406464 [==============================] - 0s 0us/step
```

```
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    # tf.keras.layers.Dense(512, activation='relu'),
    # tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(101, activation='softmax'),
])


LEARNING_RATE = 0.001
optimizer = tf.keras.optimizers.Adam(lr=LEARNING_RATE)

model.compile(optimizer=optimizer,
```

```
                loss='sparse_categorical_crossentropy',
                metrics=['accuracy'])

tensorboard_callback = tf.keras.callbacks.TensorBoard(
    log_dir='logs/'+EXP_NAME,
    write_graph=False, update_freq=100, profile_batch=0)
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning:
  super(Adam, self).__init__(name, **kwargs)
```

summary

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_160 (Funct  (None, 5, 5, 1280)       2257984
 ional)

 global_average_pooling2d (G  (None, 1280)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 101)               129381

=================================================================
Trainable params: 129,381
Non-trainable params: 2,257,984
_________________________________________________________________
```

Сохранено

Обучение модели

```
%%time

history = model.fit(
    train,
    epochs=NUM_EPOCHS,
    validation_data=test_ds,
    callbacks=[tensorboard_callback])
```

```
Epoch 1/5
1775/1775 [==============================] - 3912s 2s/step - loss: 3.5112 - accuracy
Epoch 2/5
1775/1775 [==============================] - 4092s 2s/step - loss: 3.1494 - accuracy
Epoch 3/5
1775/1775 [==============================] - 4162s 2s/step - loss: 3.0487 - accuracy
Epoch 4/5
1775/1775 [==============================] - 4168s 2s/step - loss: 2.9903 - accuracy
Epoch 5/5
1775/1775 [==============================] - 4039s 2s/step - loss: 2.9473 - accuracy
```

```
CPU times: user 7h 57min 28s, sys: 28min 8s, total: 8h 25min 37s
Wall time: 5h 39min 50s
```

Оценка качества модели

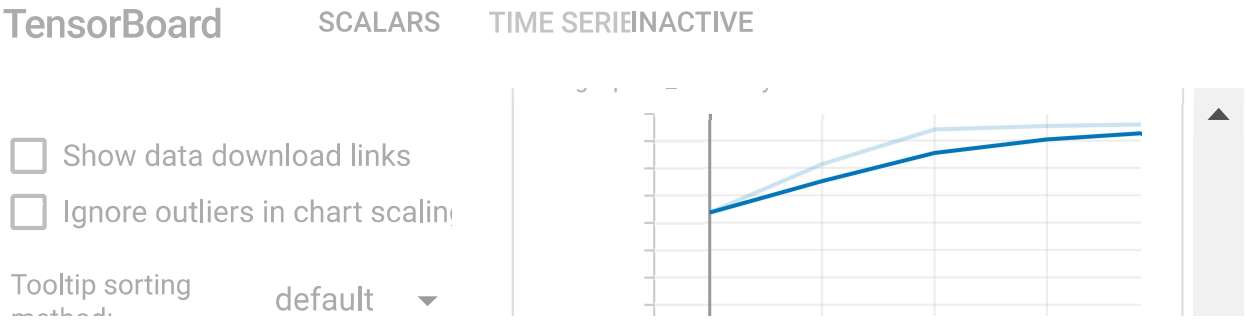
```
%%time

model.evaluate(test_ds)
```

```
147/147 [==============================] - 325s 2s/step - loss: 2.0024 - accuracy: 0
CPU times: user 9min 40s, sys: 27.9 s, total: 10min 8s
Wall time: 5min 25s
[2.0024359226226807, 0.512117326259613]
```

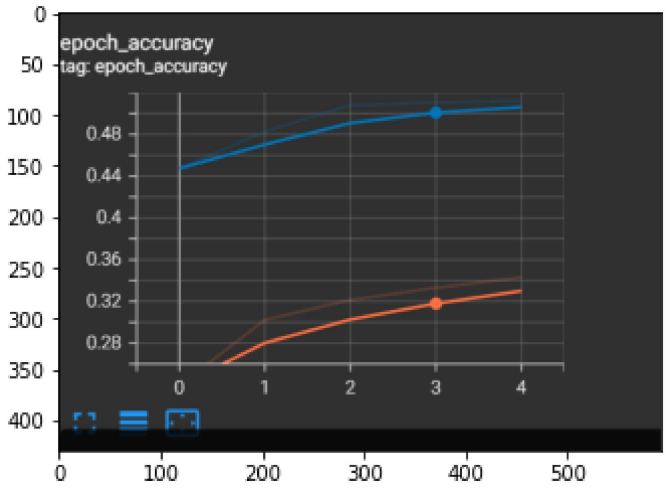
```
%load_ext tensorboard
%tensorboard --logdir logs
```

Сохранено




```
img_1 = imageio.imread('img_1.png', pilmode="RGB")
plt.imshow(img_1)
```

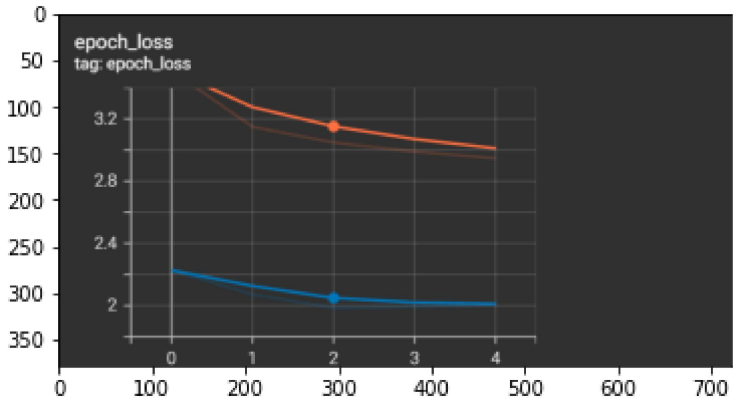
```
<matplotlib.image.AxesImage at 0x7f06ab7b8990>
```



Сохранено

```
pilmode="RGB")
plt.imshow(img_2)
```

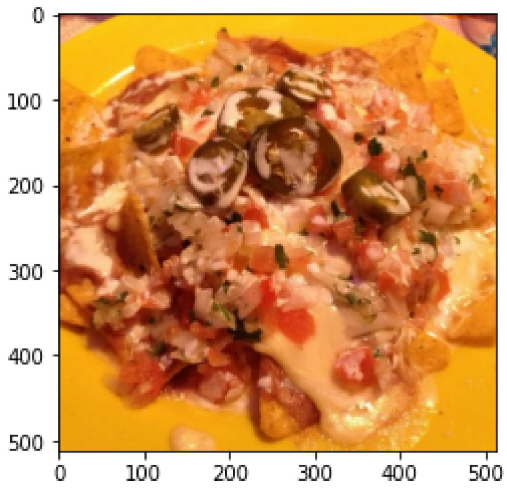
```
<matplotlib.image.AxesImage at 0x7f06ac8c1710>
```




```
import imageio
from skimage.transform import resize

def prepare_image(img):
    img = tf.cast(img, tf.float32)/127. - 1.
    img = resize(img, (INP_SIZE, INP_SIZE), order=3, mode='reflect', anti_aliasing=True)
    return img.astype(np.float32)
```

```
# img = imageio.imread('image.png', pilmode="RGB")
img = result_image[3][0]
plt.imshow(img)
img = prepare_image(img)
```




```
pred = model(img[None, ...], training=False)

pred_label = int(pred.numpy()[0][0] > 0.5)
print('Prediction: {}'.format(ds_info.features['label'].int2str(pred_label)))
```

```
Prediction: apple_pie
```

Сохранено

растет, ошибка падает, эпох если поставить больше

точность вырастет

✓ 0 сек. выполнено в 15:48