Курс «Глубокое обучение в компьютерном зрении»

Урок 3. Продвинутые архитектуры свёрточных нейросетей

Практическое задание 3

Обучить CHC с помощью Transfer Learning на датасете Food-101
Использовать тонкую настройку существующей предобученной модели и методы аугментации данных.
Библиотеки: [Python, Tensorflow]

Выполнил ***Соковнин ИЛ***

Food-101

https://www.tensorflow.org/datasets/catalog/food101

Описание :

Этот набор данных состоит из 101 категории продуктов питания и 101 000 изображений. Для каждого класса предоставляется 250 проверенных вручную тестовых изображений, а также 750 обучающих изображений. Учебные изображения намеренно не очищались и поэтому все еще содержат некоторое количество шума. В основном это проявляется в виде интенсивных цветов и иногда неправильных этикеток. Все изображения были масштабированы таким образом, чтобы максимальная длина стороны составляла 512 пикселей.

- **Домашняя страница :** https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/

```
%tensorflow_version 2.x
```

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

import tensorflow as tf
import tensorflow_datasets as tfds

from skimage.transform import resize
from tensorflow.keras import layers
```

```
gpu_devices = tf.config.experimental.list_physical_devices('GPU')
for device in gpu_devices:
    tf.config.experimental.set_memory_growth(device, True)
```

Загрузка датасета Food-101


```
tfds.disable_progress_bar()
(train_ds, test_ds), ds_info = tfds.load(
    'food101',
    as_supervised=True,
    with_info=True,
    split=['train[:90%]', 'train[90%:]'],  # ?
)
```

```
Downloading and preparing dataset food101/2.0.0 (download: 4.65 GiB, generated: Unknown size, total: 4.65 GiB) to /root/tensorf
Shuffling and writing examples to /root/tensorflow_datasets/food101/2.0.0.incomplete4D5QJZ/food101-train.tfrecord
Shuffling and writing examples to /root/tensorflow_datasets/food101/2.0.0.incomplete4D5QJZ/food101-validation.tfrecord
Dataset food101 downloaded and prepared to /root/tensorflow_datasets/food101/2.0.0. Subsequent calls will reuse this data.
CPU times: user 2min 4s, sys: 44.9 s, total: 2min 49s
Wall time: 8min 7s
```

```
ds_info.description
```

```
'This dataset consists of 101 food categories, with 101'000 images. For each class, 250 manually reviewed test images are provi
ded as well as 750 training images. On purpose, the training images were not cleaned, and thus still contain some amount of noi
se. This comes mostly in the form of intense colors and sometimes wrong labels. All images were rescaled to have a maximum side
length of 512 pixels.'
```

```
# Get class names
classes = ds_info.features["label"].names
classes[:10]
```

```
['apple_pie',
 'baby_back_ribs',
 'baklava',
 'beef_carpaccio',
 'beef_tartare',
 'beet_salad',
 'beignets',
 'bibimbap',
 'bread_pudding',
 'breakfast_burrito']
```

```
number_classes = len(classes)
number_classes
```

```
101
```

```
print(type(train_ds))
```

```
<class 'tensorflow.python.data.ops.dataset_ops.PrefetchDataset'>
```

Визуализация датасета Food0101

```
%%time

some_samples = []
some_labels =[]

for x in iter(test_ds.take(32)):
    some_samples.append(x[0])
    some_labels.append(x[1])

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])
    ax.title.set_text(classes[some_labels[j]])
    plt.xticks([]), plt.yticks([])
plt.show()
```

Создание пайплайна данных

```
# Так как мы будем использовать готовую архитектуру MobileNet с предобученными весами,
# они ожидают конкретные картинки, в конкретном диапазоне [-1, 1].

INP_SIZE = 224
NUM_EPOCHS = 5
BATCH_SIZE = 32

def prepare(img, label):
    # Делаем диапазоне [-1, 1]. Такие картинки нельзя визуализировать.
    # Создаём квадратные картинки с входным размером 224x224.
    img = tf.cast(img, tf.float32)/127. - 1.  # [0, 2] - 1 = [-1, 1]
    return tf.image.resize(img, (INP_SIZE, INP_SIZE)), label

train_ds = train_ds.shuffle(buffer_size=1000)
train_ds = train_ds.map(prepare)
train_ds = train_ds.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(BATCH_SIZE, drop_remainder=True)  # размер batcha=128, drop_remainder - ?
```

```
    CPU times: user 1.9 s, sys: 105 ms, total: 2 s
    Wall time: 2.25 s
```

```
train_ds, test_ds
```

```
    (<BatchDataset element_spec=(TensorSpec(shape=(32, 224, 224, 3), dtype=tf.float32, name=None), TensorSpec(shape=(32,),
    dtype=tf.int64, name=None))>,
     <BatchDataset element_spec=(TensorSpec(shape=(32, 224, 224, 3), dtype=tf.float32, name=None), TensorSpec(shape=(32,),
    dtype=tf.int64, name=None))>)
```

Transfer Learning с дообучением весов предобученной модели (Fine-tuning)

Fine-Tuning - тонкая настройка

Подготовка модели CNN

```
# Регуляризация - наложение ограничений на обучаемые параметры NN (решение проблемы переобучения).

# Вводим для модели L2-регуляризацию (L2 также называется WEIGHT_DECAY)
# т.е. 2-ю норму весов модели стремим к 0
WEIGHT_DECAY = 0.001   # Коэффициент lambda при 2-м слагаемом нашего лосса

wd = tf.keras.regularizers.l2(WEIGHT_DECAY)  # L2(WEIGHT_DECAY) регуляризация
```

```
# см. 3_1_Advanced_CNN.ipynb


EXP_NAME = 'transfer'
base_model = tf.keras.applications.MobileNetV2(
    input_shape=(INP_SIZE, INP_SIZE, 3),
    include_top=False,   # Верни модель без последнего слоя (top-слоя).
    # weights='imagenet' # Веса base_model будут предобучены на датасете Imagenet.
)

for layer in base_model.layers[:-5]:
  base_model.trainable = True # Fine-tuning весов слоёв предобученной модели

for layer in base_model.layers[-5:]:
    layer.kernel_regularizer=wd  # применяем L2(WEIGHT_DECAY)-регуляризацию

data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation((-0.2, 0.2)),
])

model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(INP_SIZE, INP_SIZE, 3)),
    data_augmentation,
    base_model,  # На выходе модели тензор с пространственными разметрами [7x7x1280 (1280 каналов)]
```

```
    # Добавляем собственные слои
    tf.keras.layers.GlobalAveragePooling2D(),  # Сокращаем все пространственные измерения и получаем вектор (по каналам)
    tf.keras.layers.Dense(256, activation='relu', kernel_regularizer=wd),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(101, activation='softmax')
])
```

Model Summary

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 224, 224, 3)       0

 mobilenetv2_1.00_224 (Funct  (None, 7, 7, 1280)       2257984
 ional)

 global_average_pooling2d (G  (None, 1280)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 256)               327936

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 101)               25957

=================================================================
Total params: 2,611,877
Trainable params: 2,577,765
Non-trainable params: 34,112
_________________________________________________________________
```

Подготовка к обучению

```
LEARNING_RATE = 0.0001
optimizer = tf.keras.optimizers.Adam(lr=LEARNING_RATE)

model.compile(optimizer=optimizer,
              loss='sparse_categorical_crossentropy',  # многоклассовая классификация
              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: The `lr` argument is deprecated, use `learn
  super(Adam, self).__init__(name, **kwargs)
```

Обучение модели

Сохранение...

```
%%time

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

```
Epoch 1/5
2130/2130 [==============================] - 475s 214ms/step - loss: 3.4858 - accuracy: 0.2958 - val_loss: 2.4899 - val_accuracy
Epoch 2/5
2130/2130 [==============================] - 456s 213ms/step - loss: 2.3752 - accuracy: 0.5078 - val_loss: 2.1219 - val_accuracy
Epoch 3/5
2130/2130 [==============================] - 456s 214ms/step - loss: 2.0160 - accuracy: 0.5784 - val_loss: 1.8927 - val_accuracy
```

```
Epoch 4/5
2130/2130 [==============================] - 460s 216ms/step - loss: 1.7984 - accuracy: 0.6201 - val_loss: 1.6964 - val_accuracy
Epoch 5/5
2130/2130 [==============================] - 458s 214ms/step - loss: 1.6325 - accuracy: 0.6521 - val_loss: 1.6614 - val_accuracy
CPU times: user 57min 4s, sys: 1min 33s, total: 58min 38s
Wall time: 39min 41s
```

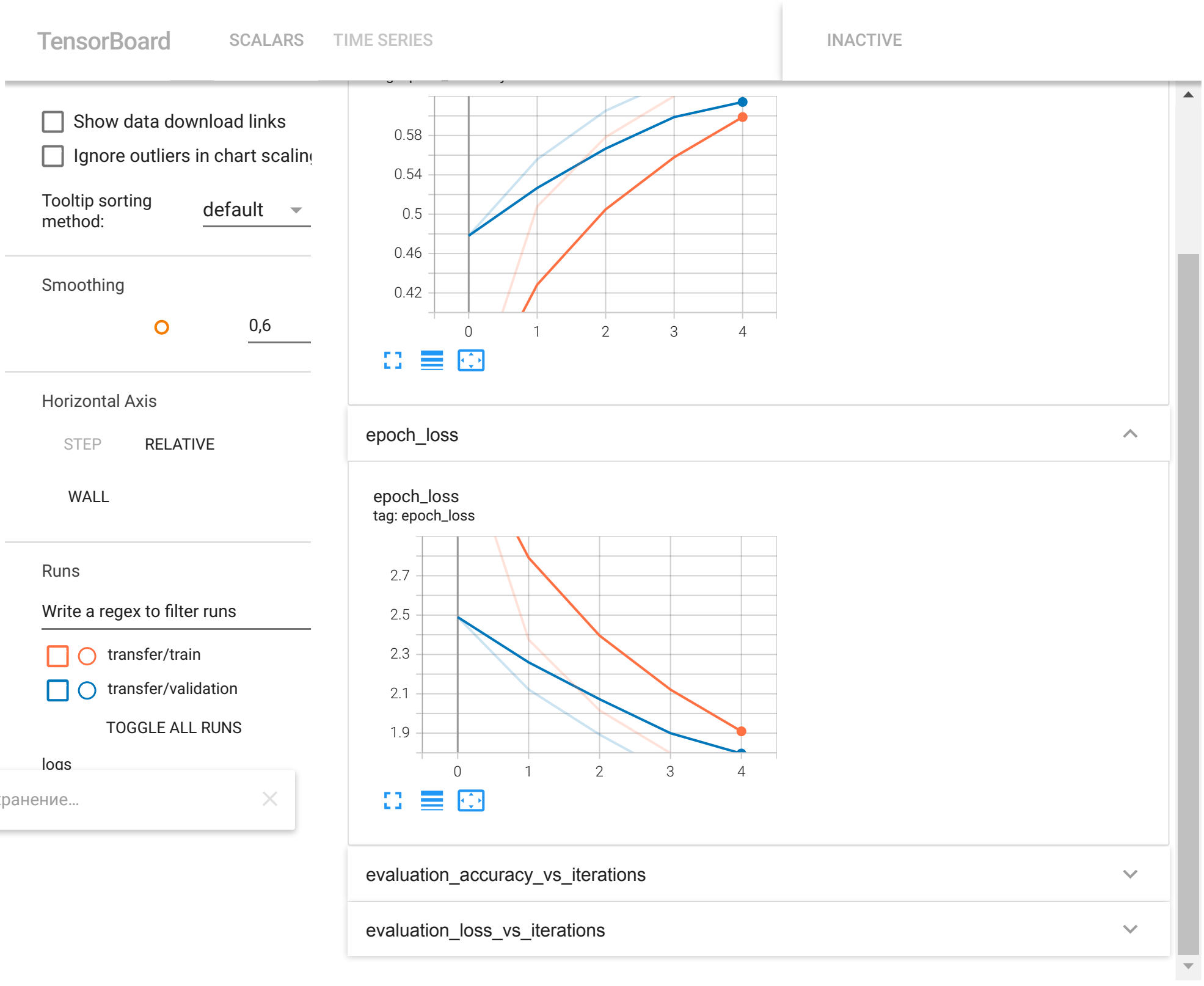
Оценка качества модели

```
%%time

result = model.evaluate(test_ds)
result
```

```
236/236 [==============================] - 25s 100ms/step - loss: 1.6602 - accuracy: 0.6344
CPU times: user 38.8 s, sys: 1.73 s, total: 40.6 s
Wall time: 41 s
```

TensorBoard

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

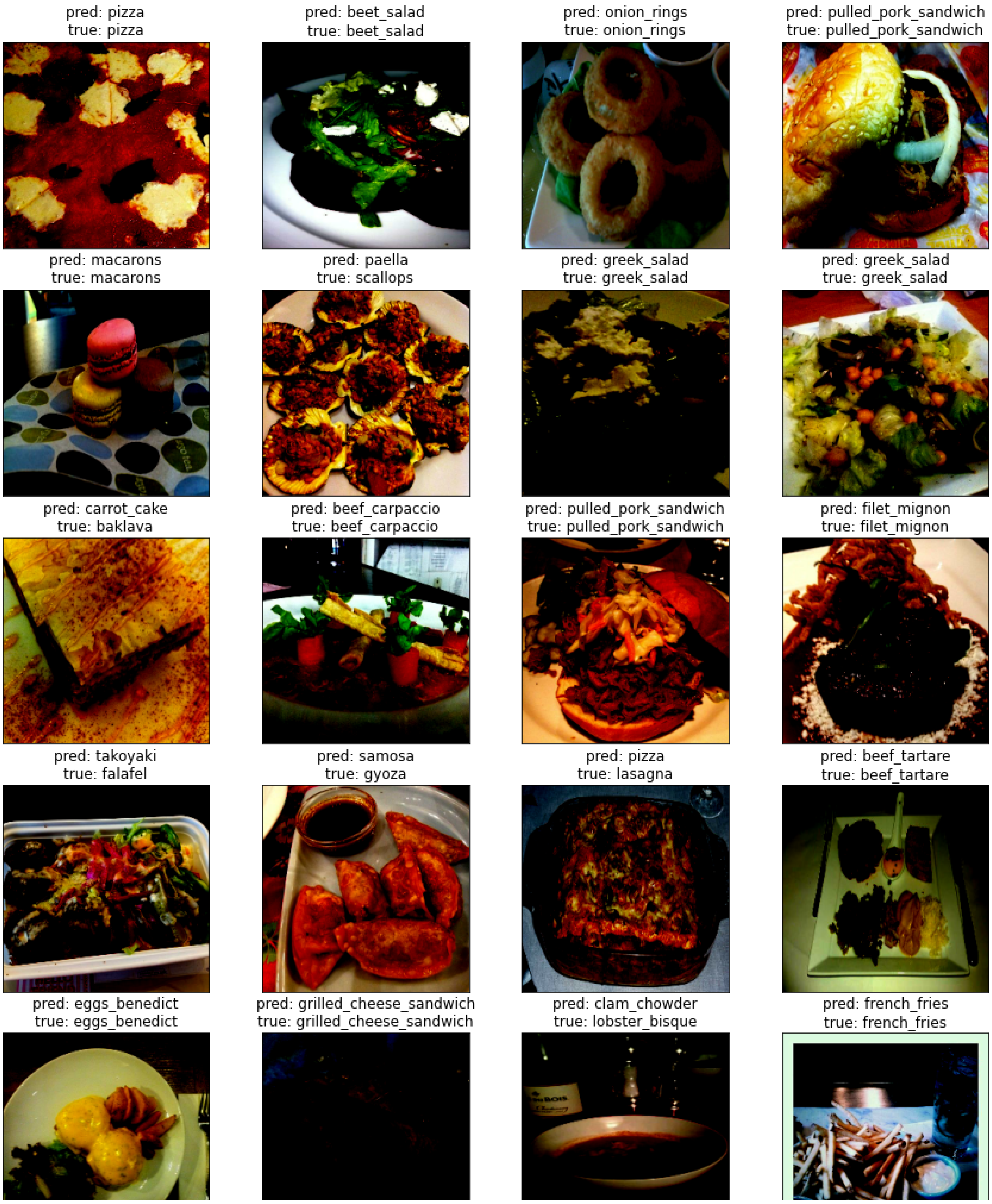


Запуск предсказания на изображении

```
some_samples = []
some_labels =[]
```

```
for x in iter(test_ds.take(32)):
    some_samples.append(x[0])
    some_labels.append(x[1])



predictions = model.predict(some_samples[0])


prediction_labels = tf.argmax(predictions, axis=1, output_type=tf.int32)


fig = plt.figure(figsize=(15, 30))
for j in range(len(some_samples[0])):
    ax = fig.add_subplot(8, 4, j+1)
    ax.imshow(some_samples[0][j])
    ax.title.set_text(f'pred: {classes[prediction_labels[j]]}\n' + \
                      f'true: {classes[some_labels[0][j]]}')
    plt.xticks([]), plt.yticks([])
plt.show()
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for i
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for i
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for i
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for i
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for i
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for i
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for i
```



Сохранение...