HW3 Мельчук А.Б.

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

In [0]:

```
%tensorflow_version 2.x
```

In [0]:

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

import tensorflow as tf
import tensorflow_datasets as tfds
AUTOTUNE = tf.data.experimental.AUTOTUNE
```

Загрузка датасета Food 101

In [3]:

```
tfds.disable_progress_bar()
ds, ds_info = tfds.load(
    'food101',
    as_supervised=True,
    with_info=True,
)
```

Downloading and preparing dataset food101/2.0.0 (download: 4.65 GiB, generated: Unknown size, total: 4.65 GiB) to /root/tensorflow_datasets/food101/2.0.0...

```
/usr/local/lib/python3.6/dist-packages/urllib3/connectionpool.py:847: InsecureRequestWarning: Unverified HTTPS request is being made. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings (https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings)
  InsecureRequestWarning)
```

```
Shuffling and writing examples to /root/tensorflow_datasets/food101/2.0.0.incompleteKBXSI5/food101-train.tfrecord
Shuffling and writing examples to /root/tensorflow_datasets/food101/2.0.0.incompleteKBXSI5/food101-validation.tfrecord
```

Dataset food101 downloaded and prepared to /root/tensorflow_datasets/food101/2.0.0. Subsequent calls will reuse this data.

In [4]:

```
train_ds = ds['train']
test_ds = ds['validation']
num_train_examples= ds_info.splits['train'].num_examples
num_train_examples
```

Out[4]:

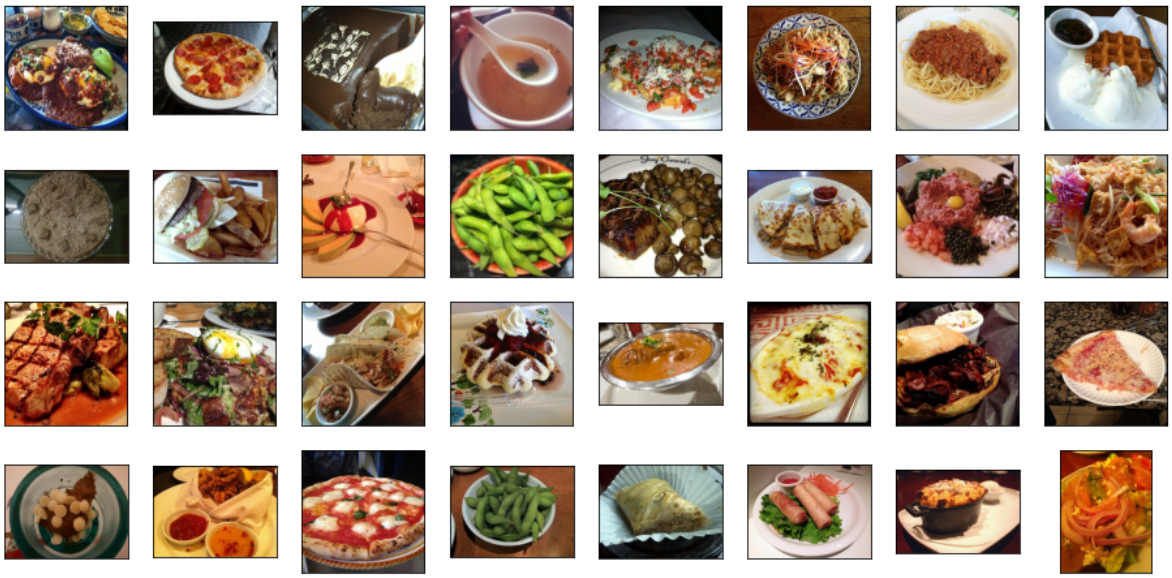
```
75750
```

Визуализация датасета Food 101

In [5]:

```
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()
```



Создание пайплайна данных

In [0]:

```
INP_SIZE = 224
NUM_EPOCHS = 10
BATCH_SIZE = 64
```

In [0]:

```
def convert(image, label):
  image = tf.image.convert_image_dtype(image, tf.float32) # Cast and normalize the image to
  image = tf.image.resize_with_crop_or_pad(image, INP_SIZE, INP_SIZE)
  return image, label

def augment(image,label):
  image = tf.image.convert_image_dtype(image, tf.float32) # Cast and normalize the image to
  image = tf.image.resize_with_crop_or_pad(image, INP_SIZE+50, INP_SIZE+50) # Add 50 pixels
  image = tf.image.random_crop(image, size=[INP_SIZE, INP_SIZE, 3]) # Random crop back to 2
  image = tf.image.random_brightness(image, max_delta=0.5) # Random brightness

  return image,label
```

In [0]:

```
augmented_train_batches = (
    train_ds
    .shuffle(buffer_size=1000)
    # The augmentation is added here.
    .map(augment, num_parallel_calls=AUTOTUNE)
    .batch(BATCH_SIZE)
    .prefetch(AUTOTUNE)
)
```

In [0]:

```
validation_batches = (
    test_ds
    .shuffle(buffer_size=1000)
    .map(convert, num_parallel_calls=AUTOTUNE)
    .batch(2*BATCH_SIZE)
)
```

Подготовка модели CNN

In [0]:

```
EXP_NAME = 'transfer_NASNetMobile_augmented'
base_model = tf.keras.applications.NASNetMobile(
    input_shape=(INP_SIZE, INP_SIZE, 3),
    include_top=False,
    weights='imagenet',
)
base_model.trainable = True # Fine-tuning весов предобученной модели

WEIGHT_DECAY = 0.001

wd = tf.keras.regularizers.l2(WEIGHT_DECAY)

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

Подготовка к обучению

In [0]:

```
LEARNING_RATE = 0.0001
optimizer = tf.keras.optimizers.RMSprop(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)
```

Model Summary

In [12]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
NASNet (Model)               (None, 7, 7, 1056)        4269716
_________________________________________________________________
global_average_pooling2d (Gl (None, 1056)              0
_________________________________________________________________
dense (Dense)                (None, 512)               541184
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 101)               51813
=================================================================
Total params: 4,862,713
Trainable params: 4,825,975
Non-trainable params: 36,738
_________________________________________________________________
```

Обучение модели

In [22]:

```
%%time

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

```
Epoch 1/10
1184/1184 [==============================] - 1295s 1s/step - loss: 3.7688 -
accuracy: 0.2777 - val_loss: 3.2564 - val_accuracy: 0.3573
Epoch 2/10
1184/1184 [==============================] - 1291s 1s/step - loss: 2.5088 -
accuracy: 0.5190 - val_loss: 2.8246 - val_accuracy: 0.4345
Epoch 3/10
1184/1184 [==============================] - 1288s 1s/step - loss: 2.0881 -
accuracy: 0.5990 - val_loss: 2.7509 - val_accuracy: 0.4652
Epoch 4/10
1184/1184 [==============================] - 1293s 1s/step - loss: 1.8071 -
accuracy: 0.6519 - val_loss: 2.4459 - val_accuracy: 0.5382
Epoch 5/10
1184/1184 [==============================] - 1293s 1s/step - loss: 1.5987 -
accuracy: 0.6867 - val_loss: 2.1818 - val_accuracy: 0.6034
Epoch 6/10
1184/1184 [==============================] - 1294s 1s/step - loss: 1.4230 -
accuracy: 0.7168 - val_loss: 2.3059 - val_accuracy: 0.6090
Epoch 7/10
1184/1184 [==============================] - 1298s 1s/step - loss: 1.2739 -
accuracy: 0.7435 - val_loss: 2.2372 - val_accuracy: 0.6340
Epoch 8/10
1184/1184 [==============================] - 1289s 1s/step - loss: 1.1543 -
accuracy: 0.7635 - val_loss: 2.6469 - val_accuracy: 0.6148
Epoch 9/10
1184/1184 [==============================] - 1290s 1s/step - loss: 1.0379 -
accuracy: 0.7857 - val_loss: 2.5113 - val_accuracy: 0.6383
Epoch 10/10
1184/1184 [==============================] - 1298s 1s/step - loss: 0.9373 -
accuracy: 0.8042 - val_loss: 2.4287 - val_accuracy: 0.6713
CPU times: user 3h 1min 5s, sys: 1h 7min 24s, total: 4h 8min 30s
Wall time: 3h 36min 51s
```

Оценка качества модели

In [24]:

```
%%time

model.evaluate(validation_batches)
```

```
198/198 [==============================] - 65s 330ms/step - loss: 2.4287 - a
ccuracy: 0.6713
CPU times: user 1min 47s, sys: 9.72 s, total: 1min 56s
Wall time: 1min 7s
```

Out[24]:

```
[2.428652286529541, 0.671326756477356]
```

TensorBoard

Используем обученную на *Imagenet* модель *InceptionResNetV2* в качестве базовой.

In [25]:

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

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
Reusing TensorBoard on port 6006 (pid 537), started 3:50:36 ago. (Use '!kill
537' to kill it.)
```

```
<IPython.core.display.Javascript object>
```

Transfer Learning

In [27]:

```
EXP_NAME = 'transfer_InceptionResNetV2_augmented'
base_model_InceptionResNetV2 = tf.keras.applications.InceptionResNetV2(
    input_shape=(INP_SIZE, INP_SIZE, 3),
    include_top=False,
    weights='imagenet',
)
base_model_InceptionResNetV2.trainable = True

WEIGHT_DECAY = 0.001

wd = tf.keras.regularizers.l2(WEIGHT_DECAY)

model = tf.keras.Sequential([
    base_model_InceptionResNetV2,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(512, activation='relu', kernel_regularizer=wd),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(101, activation='softmax'),
])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applic
ations/inception_resnet_v2/inception_resnet_v2_weights_tf_dim_ordering_tf_ke
rnels_notop.h5 (https://storage.googleapis.com/tensorflow/keras-application
s/inception_resnet_v2/inception_resnet_v2_weights_tf_dim_ordering_tf_kernels
_notop.h5)
219062272/219055592 [==============================] - 4s 0us/step
```

In [0]:

```
LEARNING_RATE = 0.0001
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)
```

In [29]:

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Model)  (None, 5, 5, 1536)        54336736
_________________________________________________________________
global_average_pooling2d_2 ( (None, 1536)              0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               786944
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 101)               51813
=================================================================
Total params: 55,175,493
Trainable params: 55,114,949
Non-trainable params: 60,544
_________________________________________________________________
```

In [30]:

```
%%time

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

```
Epoch 1/10
1184/1184 [==============================] - 1717s 1s/step - loss: 3.2362 -
accuracy: 0.4108 - val_loss: 2.1274 - val_accuracy: 0.6237
Epoch 2/10
1184/1184 [==============================] - 1720s 1s/step - loss: 2.1235 -
accuracy: 0.6204 - val_loss: 1.8534 - val_accuracy: 0.6712
Epoch 3/10
1184/1184 [==============================] - 1719s 1s/step - loss: 1.7013 -
accuracy: 0.6954 - val_loss: 1.7045 - val_accuracy: 0.6870
Epoch 4/10
1184/1184 [==============================] - 1720s 1s/step - loss: 1.4039 -
accuracy: 0.7472 - val_loss: 1.5622 - val_accuracy: 0.7046
Epoch 5/10
1184/1184 [==============================] - 1721s 1s/step - loss: 1.1725 -
accuracy: 0.7874 - val_loss: 1.5442 - val_accuracy: 0.7044
Epoch 6/10
1184/1184 [==============================] - 1722s 1s/step - loss: 0.9930 -
accuracy: 0.8170 - val_loss: 1.4681 - val_accuracy: 0.7091
Epoch 7/10
1184/1184 [==============================] - 1723s 1s/step - loss: 0.8431 -
accuracy: 0.8419 - val_loss: 1.5185 - val_accuracy: 0.6913
Epoch 8/10
1184/1184 [==============================] - 1722s 1s/step - loss: 0.7229 -
accuracy: 0.8609 - val_loss: 1.4685 - val_accuracy: 0.6966
Epoch 9/10
1184/1184 [==============================] - 1720s 1s/step - loss: 0.6301 -
accuracy: 0.8775 - val_loss: 1.4994 - val_accuracy: 0.6993
Epoch 10/10
1184/1184 [==============================] - 1723s 1s/step - loss: 0.5457 -
accuracy: 0.8927 - val_loss: 1.4601 - val_accuracy: 0.7019
CPU times: user 3h 18min 20s, sys: 1h 31min 29s, total: 4h 49min 49s
Wall time: 4h 48min 38s
```

In [31]:

```
%%time

model.evaluate(validation_batches)
```

```
198/198 [==============================] - 154s 780ms/step - loss: 1.4601 -
accuracy: 0.7019
CPU times: user 1min 52s, sys: 13.4 s, total: 2min 5s
Wall time: 2min 37s
```

Out[31]:

```
[1.4600964784622192, 0.7019405961036682]
```

Модель переобучилась, но показала результат лучше чем у ***MobileNet.***