ccn_food

April 17, 2020

1 Introduction

1.0.1 Goal

Our goal was to develop a food recognition model, which would allow us to identify a food type based on an image.

1.0.2 Approach

We opted by implementing Convolutional Neural Networks (CNN), for it's relative ease of implementation, accuracy and because we think it is the one that best suited our problem.

We followed several ideas we found online, but tweaking them to our needs and available resources.

1.0.3 First idea

We first opted to base our approach on the work developed by Patrick Rodriguez as it seemed to be a very complete development of our idea.

However, we soon realized several problems with this approach:

1. **Old libraries**: The original notebook was built using old versions of tensorflow and keras, making it dificult to run the notebook on our machines.
2. **Large RAM usage**: The original author also opted to load the entire dataset into memory, which required more than 80GB of RAM
3. **Large dataset:** As discussed with the professor, we realized that using a very large dataset (with 100 classes and 1000 images for each class), would make training and building the model very difficult, given the resources we have available.

1.0.4 Data Description

To implement this algorithm we used one of the popular pick when it comes to food recognition, which is Food-101, which is divided in four major folders: * Images (contais all the +94000 images) * Meta (Declaration of 101 classes and labels) * Train (Contains the classes divided for training) * Test (Contains the classes divided for testing)

Since training this would be very costly computer and time wise, and we were bounded by restraints on both of those, we chose to create our subset called **Food-5** and **Food-10**, which had five and ten classes respectively.

The majority of training was made for the dataset Food-5, but we realized that the dataset had some similar classes and sometimes the algorithm couldnt identify the food with a good accuracy, so we ran a few more test for dataset Food-10, which took us about 12 hours.

2 Prepare environment

2.0.1 Install dependencies

[0]:
```
!pip install tensorflow==2.1.0
```

```
Requirement already satisfied: tensorflow==2.1.0 in
/usr/local/lib/python3.6/dist-packages (2.1.0)
Requirement already satisfied: gast==0.2.2 in /usr/local/lib/python3.6/dist-
packages (from tensorflow==2.1.0) (0.2.2)
Requirement already satisfied: numpy<2.0,>=1.16.0 in
/usr/local/lib/python3.6/dist-packages (from tensorflow==2.1.0) (1.18.2)
Requirement already satisfied: google-pasta>=0.1.6 in
/usr/local/lib/python3.6/dist-packages (from tensorflow==2.1.0) (0.2.0)
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.6/dist-
packages (from tensorflow==2.1.0) (0.9.0)
Requirement already satisfied: astor>=0.6.0 in /usr/local/lib/python3.6/dist-
packages (from tensorflow==2.1.0) (0.8.1)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.6/dist-packages (from tensorflow==2.1.0) (1.1.0)
Requirement already satisfied: tensorflow-estimator<2.2.0,>=2.1.0rc0 in
/usr/local/lib/python3.6/dist-packages (from tensorflow==2.1.0) (2.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.6/dist-
packages (from tensorflow==2.1.0) (1.12.0)
Requirement already satisfied: scipy==1.4.1; python_version >= "3" in
/usr/local/lib/python3.6/dist-packages (from tensorflow==2.1.0) (1.4.1)
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.6/dist-
packages (from tensorflow==2.1.0) (1.12.1)
Requirement already satisfied: keras-applications>=1.0.8 in
/usr/local/lib/python3.6/dist-packages (from tensorflow==2.1.0) (1.0.8)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.6/dist-packages (from tensorflow==2.1.0) (3.2.0)
Requirement already satisfied: protobuf>=3.8.0 in /usr/local/lib/python3.6/dist-
packages (from tensorflow==2.1.0) (3.10.0)
Requirement already satisfied: keras-preprocessing>=1.1.0 in
/usr/local/lib/python3.6/dist-packages (from tensorflow==2.1.0) (1.1.0)
Requirement already satisfied: tensorboard<2.2.0,>=2.1.0 in
/usr/local/lib/python3.6/dist-packages (from tensorflow==2.1.0) (2.1.1)
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.6/dist-
packages (from tensorflow==2.1.0) (1.28.1)
```

```
Requirement already satisfied: wheel>=0.26; python_version >= "3" in
/usr/local/lib/python3.6/dist-packages (from tensorflow==2.1.0) (0.34.2)
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages
(from keras-applications>=1.0.8->tensorflow==2.1.0) (2.10.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-
packages (from protobuf>=3.8.0->tensorflow==2.1.0) (46.1.3)
Requirement already satisfied: werkzeug>=0.11.15 in
/usr/local/lib/python3.6/dist-packages (from
tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (1.0.1)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-
packages (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (3.2.1)
Requirement already satisfied: google-auth<2,>=1.6.3 in
/usr/local/lib/python3.6/dist-packages (from
tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (1.7.2)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.6/dist-packages (from
tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (2.21.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.6/dist-packages (from
tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (0.4.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.6/dist-packages (from google-
auth<2,>=1.6.3->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (0.2.8)
Requirement already satisfied: cachetools<3.2,>=2.0.0 in
/usr/local/lib/python3.6/dist-packages (from google-
auth<2,>=1.6.3->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (3.1.1)
Requirement already satisfied: rsa<4.1,>=3.1.4 in /usr/local/lib/python3.6/dist-
packages (from google-
auth<2,>=1.6.3->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (4.0)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in
/usr/local/lib/python3.6/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (1.24.3)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in
/usr/local/lib/python3.6/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.6/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (2020.4.5.1)
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-
packages (from
requests<3,>=2.21.0->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.6/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (1.3.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.6/dist-packages (from pyasn1-modules>=0.2.1->google-
auth<2,>=1.6.3->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.6/dist-
```

```
packages (from requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (3.1.0)
```

[0]:
```
!pip install keras==2.2.4
```

```
Requirement already satisfied: keras==2.2.4 in /usr/local/lib/python3.6/dist-
packages (2.2.4)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.6/dist-
packages (from keras==2.2.4) (1.12.0)
Requirement already satisfied: keras-preprocessing>=1.0.5 in
/usr/local/lib/python3.6/dist-packages (from keras==2.2.4) (1.1.0)
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages
(from keras==2.2.4) (2.10.0)
Requirement already satisfied: keras-applications>=1.0.6 in
/usr/local/lib/python3.6/dist-packages (from keras==2.2.4) (1.0.8)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.6/dist-
packages (from keras==2.2.4) (1.18.2)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.6/dist-packages
(from keras==2.2.4) (3.13)
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.6/dist-
packages (from keras==2.2.4) (1.4.1)
```

2.0.2 Load Google Drive

[0]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

2.0.3 Imports

[0]:
```
import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, GlobalAveragePooling2D,␣
 ↪Flatten, Dense, InputLayer, BatchNormalization, Dropout
from tensorflow.keras.regularizers import l2
```

```
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import ModelCheckpoint

import matplotlib.pyplot as plt
import numpy as np
import os
import shutil
import collections
import json

# from predict import predict, predict_remote_image

print('imports success')
```

```
imports success
```

[0]:
```
print(tf.__version__)
print(tf.keras.__version__)
```

```
2.1.0
2.2.4-tf
```

3 Load dataset

3.0.1 Download dataset

The dataset we built from the original one can be downloaded here

3.0.2 Split images between *train* and *test*

[0]:
```
print('Spliting data between test and train dirs')

# Only split files if haven't already
if not os.path.isdir('/content/drive/My Drive/Colab Notebooks/food-5/test') and␣
↪not os.path.isdir('/content/drive/My Drive/Colab Notebooks/food-5/train'):

    def copytree(src, dst, symlinks = False, ignore = None):
        if not os.path.exists(dst):
            os.makedirs(dst)
            shutil.copystat(src, dst)
        lst = os.listdir(src)
        if ignore:
            excl = ignore(src, lst)
            lst = [x for x in lst if x not in excl]
        for item in lst:
            s = os.path.join(src, item)
            d = os.path.join(dst, item)
```

```
        if symlinks and os.path.islink(s):
            if os.path.lexists(d):
                os.remove(d)
            os.symlink(os.readlink(s), d)
            try:
                st = os.lstat(s)
                mode = stat.S_IMODE(st.st_mode)
                os.lchmod(d, mode)
            except:
                pass # lchmod not available
        elif os.path.isdir(s):
            copytree(s, d, symlinks, ignore)
        else:
            shutil.copy2(s, d)

def generate_dir_file_map(path):
    dir_files = collections.defaultdict(list)
    with open(path, 'r') as txt:
        files = [l.strip() for l in txt.readlines()]
        for f in files:
            dir_name, id = f.split('/')
            dir_files[dir_name].append(id + '.jpg')
    return dir_files

train_dir_files = generate_dir_file_map('/content/drive/My Drive/Colab␣
↪Notebooks/food-5/meta/train.txt')
test_dir_files = generate_dir_file_map('/content/drive/My Drive/Colab␣
↪Notebooks/food-5/meta/test.txt')


def ignore_train(d, filenames):
    print(d)
    subdir = d.split('/')[-1]
    to_ignore = train_dir_files[subdir]
    return to_ignore

def ignore_test(d, filenames):
    print(d)
    subdir = d.split('/')[-1]
    to_ignore = test_dir_files[subdir]
    return to_ignore

copytree('/content/drive/My Drive/Colab Notebooks/food-5/images', '/content/
↪drive/My Drive/Colab Notebooks/food-5/test', ignore=ignore_train)
copytree('/content/drive/My Drive/Colab Notebooks/food-5/images', '/content/
↪drive/My Drive/Colab Notebooks/food-5/train', ignore=ignore_test)
```

```
else:
    print('Train/Test files already copied into separate folders.')

print('done')
```

```
Spliting data between test and train dirs
Train/Test files already copied into separate folders.
done
```

3.0.3 Map class numbers to labels

[0]:
```
class_to_ix = {}
ix_to_class = {}
with open('/content/drive/My Drive/Colab Notebooks/food-5/meta/classes.txt',␣
↪'r') as txt:
    classes = [l.strip() for l in txt.readlines()]
    class_to_ix = dict(zip(classes, range(len(classes))))
    ix_to_class = dict(zip(range(len(classes)), classes))
    class_to_ix = {v: k for k, v in ix_to_class.items()}
sorted_class_to_ix = collections.OrderedDict(sorted(class_to_ix.items()))
```

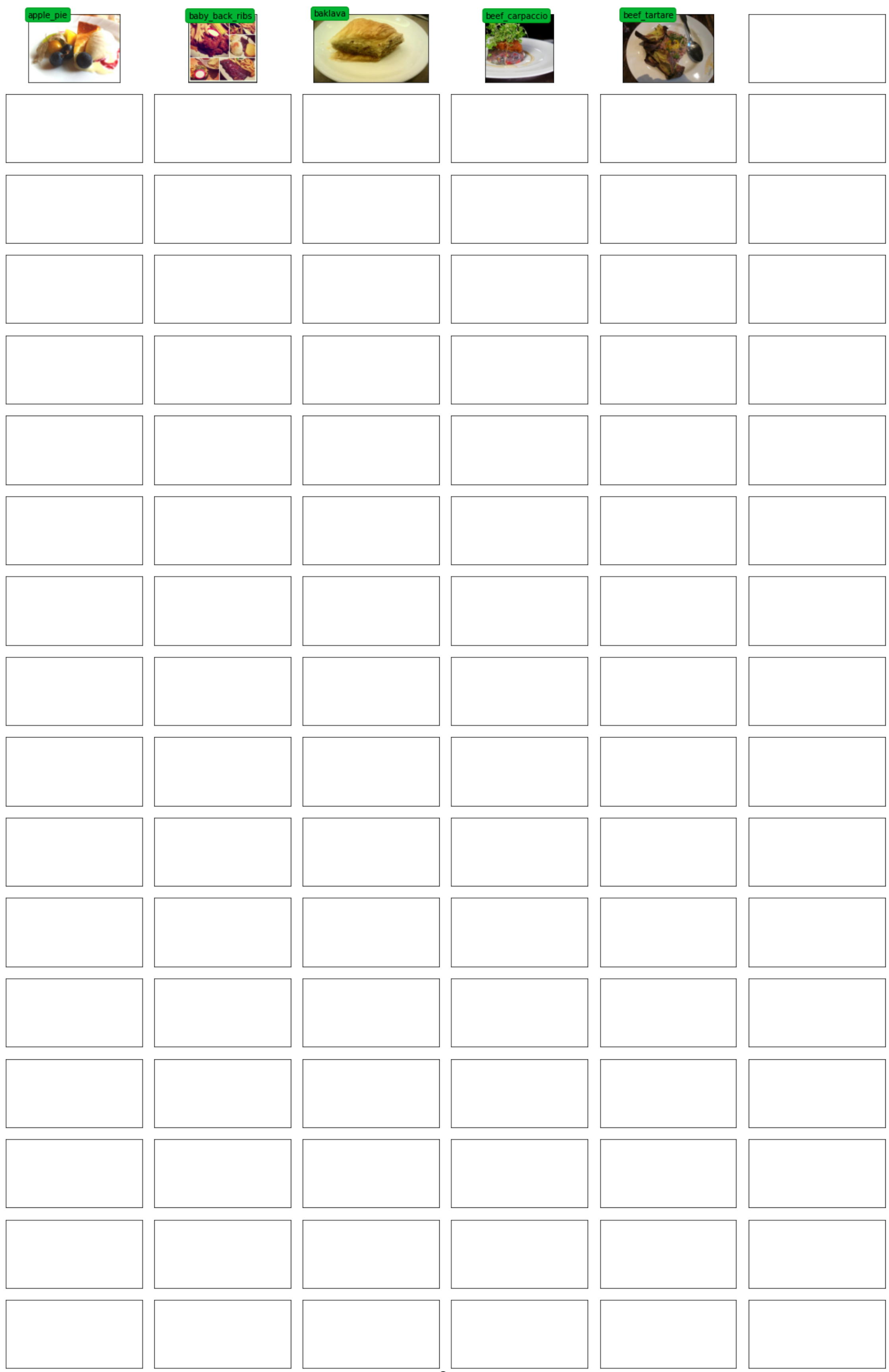
3.0.4 Visualize images from dataset

This allows us to view random images from the dataset

[0]:
```
# View random images from dataset

root_dir = '/content/drive/My Drive/Colab Notebooks/food-5/images/'
rows = 17
cols = 6
fig, ax = plt.subplots(rows, cols, frameon=False, figsize=(15, 25))
fig.suptitle('Random Image from Each Food Class', fontsize=20)
sorted_food_dirs = sorted(os.listdir(root_dir))
for i in range(rows):
    for j in range(cols):
        try:
            food_dir = sorted_food_dirs[i*cols + j]
        except:
            break
        all_files = os.listdir(os.path.join(root_dir, food_dir))
        rand_img = np.random.choice(all_files)
        img = plt.imread(os.path.join(root_dir, food_dir, rand_img))
        ax[i][j].imshow(img)
        ec = (0, .6, .1)
        fc = (0, .7, .2)
        ax[i][j].text(0, -20, food_dir, size=10, rotation=0,
                ha="left", va="top",
```

```
                bbox=dict(boxstyle="round", ec=ec, fc=fc))
plt.setp(ax, xticks=[], yticks=[])
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```

Random Image from Each Food Class



3.0.5 Import dataset into generator

We loaded the images for training using a ImageDataGenerator, which also allowed us to make the necessary preprocessing to the images.

In this case, we start by **rescaling** them by 1/255, reshape them into 244X244 pixels (by 3-RGB).

This method also allowed us to save on memory allocation, as addressed in the Introduction, as it allowed us to load the images in batches.

[0]:
```
base_dir = '/content/drive/My Drive/Colab Notebooks/food-5/'

train_dir = os.path.join(base_dir, 'train')
test_dir = os.path.join(base_dir, 'test')

# each class will be inside here

# all images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1.0/255.)
test_datagen = ImageDataGenerator(rescale=1.0/255.)

# flow train images in batches of 20 using train_datagen
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    batch_size=128,
                                                    class_mode='categorical',
                                                    shuffle=False,
                                                    target_size=(244, 244))

# flow validation images in batches of 20 using train_datagen
test_generator = test_datagen.flow_from_directory(test_dir,
                                                  batch_size=128,
                                                  class_mode='categorical',
                                                  shuffle=False,
                                                  target_size=(244, 244))
```

```
Found 3754 images belonging to 5 classes.
Found 1249 images belonging to 5 classes.
```

4 Training - 1st attempt

The following was our first attempt.

After trying to follow different, more complex approaches, we decided to start with a simpler model, built from scratch.

We have reached an accuracy of more than 0.99. However, following the metrics we established above (under **Evaluation**), we only got a score of arround 65%.

4.0.1 Build model

We built this model on top of a **Sequential** model, with **3 conventional blocks**, followed by a **flatten layer**, the **hidden dense layer**, and the ouput layer.

[0]:
```
# build a sequential model
model = Sequential()
model.add(InputLayer(input_shape=(244, 244, 3)))

# 1st conv block
model.add(Conv2D(25, (5, 5), activation='relu', strides=(1, 1), padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), padding='same'))
# 2nd conv block
model.add(Conv2D(50, (5, 5), activation='relu', strides=(2, 2), padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), padding='same'))
model.add(BatchNormalization())
# 3rd conv block
model.add(Conv2D(70, (3, 3), activation='relu', strides=(2, 2), padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), padding='valid'))
model.add(BatchNormalization())
# ANN block
model.add(Flatten())
model.add(Dense(units=100, activation='relu'))
model.add(Dense(units=100, activation='relu'))
model.add(Dropout(0.25))
# output layer
class_num = 5
model.add(Dense(units=class_num, activation='softmax'))

# compile model
model.compile(loss='categorical_crossentropy', optimizer="adam",␣
 ↪metrics=['accuracy'])

# evaluate the model
# scores = model.evaluate(X, Y, verbose=0)
# print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

# model summary
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 244, 244, 25)      1900
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 244, 244, 25)      15650
_________________________________________________________________
```

```
max_pooling2d (MaxPooling2D) (None, 122, 122, 25)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 50)        31300
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 31, 31, 50)        62550
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 50)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 16, 16, 50)        200
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 70)          31570
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 70)          44170
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 70)          0
_________________________________________________________________
batch_normalization_1 (Batch (None, 2, 2, 70)          280
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 1, 1, 70)          44170
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 1, 1, 70)          44170
_________________________________________________________________
global_average_pooling2d (Gl (None, 70)                0
_________________________________________________________________
flatten (Flatten)            (None, 70)                0
_________________________________________________________________
dense (Dense)                (None, 100)               7100
_________________________________________________________________
dense_1 (Dense)              (None, 100)               10100
_________________________________________________________________
dropout (Dropout)            (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 505
=================================================================
Total params: 293,665
Trainable params: 293,425
Non-trainable params: 240
_________________________________________________________________
None
```

4.0.2 Train

[ ]:
```
# checkpointer = ModelCheckpoint(filepath='model.2.hdf5', verbose=1,␣
↪save_best_only=True)

# fit on data for 30 epochs
```

```
#model.fit(train_generator, epochs=30, steps_per_epoch=60, batch_size=64,␣
 ↪validation_data=test_generator)
history = model.fit(train_generator, epochs=30, validation_data=test_generator)
```

[ ]:
```
Train for 59 steps, validate for 20 steps
Epoch 1/30
58/59 [============================>.] - ETA: 3s - loss: 2.0298 - accuracy: 0.
 ↪2024
Epoch 00001: val_loss improved from inf to 1.75221, saving model to model.2.hdf5
59/59 [==============================] - 259s 4s/step - loss: 2.0308 - accuracy:␣
 ↪0.1990 - val_loss: 1.7522 - val_accuracy: 0.2242
Epoch 2/30
58/59 [============================>.] - ETA: 3s - loss: 1.6268 - accuracy: 0.
 ↪2813
Epoch 00002: val_loss improved from 1.75221 to 1.56682, saving model to model.2.
 ↪hdf5
59/59 [==============================] - 261s 4s/step - loss: 1.6266 - accuracy:␣
 ↪0.2805 - val_loss: 1.5668 - val_accuracy: 0.2562
Epoch 3/30
58/59 [============================>.] - ETA: 3s - loss: 1.4998 - accuracy: 0.
 ↪3352
Epoch 00003: val_loss improved from 1.56682 to 1.46527, saving model to model.2.
 ↪hdf5
59/59 [==============================] - 262s 4s/step - loss: 1.5069 - accuracy:␣
 ↪0.3308 - val_loss: 1.4653 - val_accuracy: 0.3531
Epoch 4/30
58/59 [============================>.] - ETA: 3s - loss: 1.4167 - accuracy: 0.
 ↪3678
Epoch 00004: val_loss improved from 1.46527 to 1.42504, saving model to model.2.
 ↪hdf5
59/59 [==============================] - 259s 4s/step - loss: 1.4139 - accuracy:␣
 ↪0.3711 - val_loss: 1.4250 - val_accuracy: 0.3763
Epoch 5/30
58/59 [============================>.] - ETA: 3s - loss: 1.3461 - accuracy: 0.
 ↪4228
Epoch 00005: val_loss improved from 1.42504 to 1.39837, saving model to model.2.
 ↪hdf5
59/59 [==============================] - 261s 4s/step - loss: 1.3508 - accuracy:␣
 ↪0.4196 - val_loss: 1.3984 - val_accuracy: 0.4115
Epoch 6/30
58/59 [============================>.] - ETA: 3s - loss: 1.1381 - accuracy: 0.
 ↪5149
Epoch 00006: val_loss did not improve from 1.39837
59/59 [==============================] - 261s 4s/step - loss: 1.1339 - accuracy:␣
 ↪0.5176 - val_loss: 1.4636 - val_accuracy: 0.3955
Epoch 7/30
```

```
58/59 [============================>.] - ETA: 3s - loss: 1.0531 - accuracy: 0.
↪5657
Epoch 00007: val_loss did not improve from 1.39837
59/59 [==============================] - 258s 4s/step - loss: 1.0571 - accuracy:␣
↪0.5639 - val_loss: 1.5197 - val_accuracy: 0.3971
Epoch 8/30
58/59 [============================>.] - ETA: 3s - loss: 0.9623 - accuracy: 0.
↪6003
Epoch 00008: val_loss did not improve from 1.39837
59/59 [==============================] - 258s 4s/step - loss: 0.9557 - accuracy:␣
↪0.6036 - val_loss: 1.4727 - val_accuracy: 0.4532
Epoch 9/30
58/59 [============================>.] - ETA: 3s - loss: 0.8501 - accuracy: 0.
↪6550
Epoch 00009: val_loss did not improve from 1.39837
59/59 [==============================] - 256s 4s/step - loss: 0.8443 - accuracy:␣
↪0.6572 - val_loss: 1.6695 - val_accuracy: 0.3643
Epoch 10/30
58/59 [============================>.] - ETA: 3s - loss: 0.6629 - accuracy: 0.
↪7393
Epoch 00010: val_loss did not improve from 1.39837
59/59 [==============================] - 257s 4s/step - loss: 0.6567 - accuracy:␣
↪0.7427 - val_loss: 1.9386 - val_accuracy: 0.3915
Epoch 11/30
58/59 [============================>.] - ETA: 3s - loss: 0.5492 - accuracy: 0.
↪7740
Epoch 00011: val_loss did not improve from 1.39837
59/59 [==============================] - 256s 4s/step - loss: 0.5448 - accuracy:␣
↪0.7762 - val_loss: 1.8384 - val_accuracy: 0.4019
Epoch 12/30
58/59 [============================>.] - ETA: 3s - loss: 0.4436 - accuracy: 0.
↪8152
Epoch 00012: val_loss did not improve from 1.39837
59/59 [==============================] - 258s 4s/step - loss: 0.4458 - accuracy:␣
↪0.8157 - val_loss: 1.8748 - val_accuracy: 0.4059
Epoch 13/30
58/59 [============================>.] - ETA: 3s - loss: 0.4472 - accuracy: 0.
↪8225
Epoch 00013: val_loss did not improve from 1.39837
59/59 [==============================] - 258s 4s/step - loss: 0.4425 - accuracy:␣
↪0.8247 - val_loss: 3.3506 - val_accuracy: 0.2938
Epoch 14/30
58/59 [============================>.] - ETA: 3s - loss: 0.4385 - accuracy: 0.
↪8301
Epoch 00014: val_loss did not improve from 1.39837
```

```
59/59 [==============================] - 259s 4s/step - loss: 0.4324 - accuracy: 0.8324 - val_loss: 1.8021 - val_accuracy: 0.4692
Epoch 15/30
58/59 [============================>.] - ETA: 3s - loss: 0.2428 - accuracy: 0.9106
Epoch 00015: val_loss did not improve from 1.39837
59/59 [==============================] - 255s 4s/step - loss: 0.2460 - accuracy: 0.9105 - val_loss: 1.8323 - val_accuracy: 0.4764
Epoch 16/30
58/59 [============================>.] - ETA: 3s - loss: 0.2189 - accuracy: 0.9187
Epoch 00016: val_loss did not improve from 1.39837
59/59 [==============================] - 259s 4s/step - loss: 0.2158 - accuracy: 0.9198 - val_loss: 2.8318 - val_accuracy: 0.4011
Epoch 17/30
58/59 [============================>.] - ETA: 3s - loss: 0.2328 - accuracy: 0.9152
Epoch 00017: val_loss did not improve from 1.39837
59/59 [==============================] - 258s 4s/step - loss: 0.2294 - accuracy: 0.9166 - val_loss: 2.2904 - val_accuracy: 0.4275
Epoch 18/30
58/59 [============================>.] - ETA: 3s - loss: 0.1554 - accuracy: 0.9405
Epoch 00018: val_loss did not improve from 1.39837
59/59 [==============================] - 262s 4s/step - loss: 0.1538 - accuracy: 0.9411 - val_loss: 2.2376 - val_accuracy: 0.4556
Epoch 19/30
58/59 [============================>.] - ETA: 3s - loss: 0.1121 - accuracy: 0.9612
Epoch 00019: val_loss did not improve from 1.39837
59/59 [==============================] - 260s 4s/step - loss: 0.1108 - accuracy: 0.9616 - val_loss: 2.0762 - val_accuracy: 0.4596
Epoch 20/30
58/59 [============================>.] - ETA: 3s - loss: 0.0869 - accuracy: 0.9688
Epoch 00020: val_loss did not improve from 1.39837
59/59 [==============================] - 259s 4s/step - loss: 0.0855 - accuracy: 0.9694 - val_loss: 2.2811 - val_accuracy: 0.4644
Epoch 21/30
58/59 [============================>.] - ETA: 3s - loss: 0.0546 - accuracy: 0.9846
Epoch 00021: val_loss did not improve from 1.39837
59/59 [==============================] - 261s 4s/step - loss: 0.0538 - accuracy: 0.9848 - val_loss: 3.1850 - val_accuracy: 0.4219
Epoch 22/30
```

```
58/59 [============================>.] - ETA: 3s - loss: 0.0800 - accuracy: 0.
↪9751
Epoch 00022: val_loss did not improve from 1.39837
59/59 [==============================] - 259s 4s/step - loss: 0.0810 - accuracy:␣
↪0.9750 - val_loss: 3.0775 - val_accuracy: 0.3995
Epoch 23/30
58/59 [============================>.] - ETA: 3s - loss: 0.0863 - accuracy: 0.
↪9724
Epoch 00023: val_loss did not improve from 1.39837
59/59 [==============================] - 259s 4s/step - loss: 0.0856 - accuracy:␣
↪0.9726 - val_loss: 2.8091 - val_accuracy: 0.4331
Epoch 24/30
58/59 [============================>.] - ETA: 3s - loss: 0.0685 - accuracy: 0.
↪9794
Epoch 00024: val_loss did not improve from 1.39837
59/59 [==============================] - 258s 4s/step - loss: 0.0707 - accuracy:␣
↪0.9784 - val_loss: 2.6153 - val_accuracy: 0.4444
Epoch 25/30
58/59 [============================>.] - ETA: 3s - loss: 0.0817 - accuracy: 0.
↪9737
Epoch 00025: val_loss did not improve from 1.39837
59/59 [==============================] - 261s 4s/step - loss: 0.0819 - accuracy:␣
↪0.9739 - val_loss: 2.9071 - val_accuracy: 0.4283
Epoch 26/30
58/59 [============================>.] - ETA: 3s - loss: 0.0509 - accuracy: 0.
↪9840
Epoch 00026: val_loss did not improve from 1.39837
59/59 [==============================] - 259s 4s/step - loss: 0.0507 - accuracy:␣
↪0.9840 - val_loss: 2.8888 - val_accuracy: 0.4380
Epoch 27/30
58/59 [============================>.] - ETA: 3s - loss: 0.0511 - accuracy: 0.
↪9821
Epoch 00027: val_loss did not improve from 1.39837
59/59 [==============================] - 261s 4s/step - loss: 0.0505 - accuracy:␣
↪0.9824 - val_loss: 2.9906 - val_accuracy: 0.4315
Epoch 28/30
58/59 [============================>.] - ETA: 3s - loss: 0.0237 - accuracy: 0.
↪9932
Epoch 00028: val_loss did not improve from 1.39837
59/59 [==============================] - 260s 4s/step - loss: 0.0234 - accuracy:␣
↪0.9933 - val_loss: 3.2120 - val_accuracy: 0.4219
Epoch 29/30
58/59 [============================>.] - ETA: 3s - loss: 0.0144 - accuracy: 0.
↪9978
Epoch 00029: val_loss did not improve from 1.39837
```

```
59/59 [==============================] - 257s 4s/step - loss: 0.0144 - accuracy:␣
↪0.9979 - val_loss: 2.9914 - val_accuracy: 0.4428
Epoch 30/30
58/59 [============================>.] - ETA: 3s - loss: 0.0093 - accuracy: 0.
↪9986
Epoch 00030: val_loss did not improve from 1.39837
59/59 [==============================] - 258s 4s/step - loss: 0.0092 - accuracy:␣
↪0.9987 - val_loss: 3.0997 - val_accuracy: 0.4516
```

On this first attempt, training took a long time, about 4/5 hours...

As mentioned before, we managed to have a good accuracy, but a bad validation accuracy, which severely influenced our tests results.

RESULTS: Accuracy of 64.8% (as measured in evaluation); 4/5 hours to train

5 Trainning - 2nd attempt with transfer learning from MobileNetV2

For this second attempt, we tried to implement transfer learning.

After research, we decided to try with a relatively simple model, MobileNetV2, as our dataset is also small.

5.0.1 Reload *train* and *test* generators

For this implementation, we had to reload the images into the generator, as we followed a different classification mode.

[0]:
```
base_dir = '/content/drive/My Drive/Colab Notebooks/food-5/'

train_dir = os.path.join(base_dir, 'train')
test_dir = os.path.join(base_dir, 'test')

# each class will be inside here

# all images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1.0/255.)
test_datagen = ImageDataGenerator(rescale=1.0/255.)

# flow train images in batches of 20 using train_datagen
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    batch_size=128,
                                                    class_mode='sparse', #␣
↪change to 'categorical for 1st attempt
                                                    shuffle=False,
                                                    target_size=(244, 244))

# flow validation images in batches of 20 using train_datagen
```

```
test_generator = test_datagen.flow_from_directory(test_dir,
                                                  batch_size=128,
                                                  class_mode='sparse', # change␣
 ↪to 'categorical for 1st attempt
                                                  shuffle=False,
                                                  target_size=(244, 244))
```

5.0.2 Build model

We started build the model on top of MobileNetV2, adding just a **Flatten** and a **Dense** layer.

As mentioned before, we followed a different classification mode: **sparse**.

[0]:
```
from tensorflow.keras.preprocessing.image import load_img, img_to_array,␣
 ↪ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import GlobalAveragePooling2D, Dropout
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
import numpy as np
import argparse

shape = (244, 244, 3)
num_classes = 5

pretrained_model = tf.keras.applications.MobileNetV2(input_shape=shape,␣
 ↪include_top=False)
pretrained_model.trainable = False

model = tf.keras.Sequential([
    pretrained_model,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])

model.compile(
    optimizer='adam',
    loss = 'sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()
```

5.0.3 Train

[0]:
```
# model.fit(train_generator, validation_data=test_generator,
#                           epochs=25,
#                           steps_per_epoch=train_generator.samples//32,
#                           validation_steps=test_generator.samples//32)

history = model.fit(train_generator, epochs=30, validation_data=test_generator)
```

RESULTS: We reached an accuracy of arround 30%, but the training was the fastest of all the approaches, taking only about 30 minutes

6 Training - 3rd attempt from InceptionV3

After succeding with tranfer learning with a simpler model, we tried to move to a more comlex one, having chosen **InceptionV3**, because of the good documentation we found online with regards to similar problems.

This was also the pretrained model used by the author on the notebook we consulted.

We have also tried using **VGG16**, but **InceptionV3** proved to have better accuracy.

With more time, we would like to try loading different base models, and compare performance and results between all of them.

6.0.1 Import pretrained model

Import the model, excluding the last layer.

[0]:
```
from tensorflow.keras.applications import InceptionV3

# import pretrained model
pretrained_model = InceptionV3(weights = 'imagenet',include_top = False)

pretrained_model.summary()
```

```
Downloading data from https://github.com/fchollet/deep-learning-models/releases/
download/v0.5/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 7s 0us/step
Model: "inception_v3"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_2 (InputLayer)            [(None, None, None,  0
________________________________________________________________________________
__________________
conv2d (Conv2D)                 (None, None, None, 3 864         input_2[0][0]
```

```
--------------------------------------------------------------------------------
------------------
batch_normalization_5 (BatchNor (None, None, None, 3 96          conv2d[0][0]
--------------------------------------------------------------------------------
------------------
activation (Activation)         (None, None, None, 3 0
batch_normalization_5[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_1 (Conv2D)               (None, None, None, 3 9216
activation[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_6 (BatchNor (None, None, None, 3 96          conv2d_1[0][0]
--------------------------------------------------------------------------------
------------------
activation_1 (Activation)       (None, None, None, 3 0
batch_normalization_6[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_2 (Conv2D)               (None, None, None, 6 18432
activation_1[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_7 (BatchNor (None, None, None, 6 192         conv2d_2[0][0]
--------------------------------------------------------------------------------
------------------
activation_2 (Activation)       (None, None, None, 6 0
batch_normalization_7[0][0]
--------------------------------------------------------------------------------
------------------
max_pooling2d (MaxPooling2D)    (None, None, None, 6 0
activation_2[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_3 (Conv2D)               (None, None, None, 8 5120
max_pooling2d[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_8 (BatchNor (None, None, None, 8 240         conv2d_3[0][0]
--------------------------------------------------------------------------------
------------------
activation_3 (Activation)       (None, None, None, 8 0
batch_normalization_8[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_4 (Conv2D)               (None, None, None, 1 138240
activation_3[0][0]
```

```
________________________________________________________________________________
__________________
batch_normalization_9 (BatchNor (None, None, None, 1 576         conv2d_4[0][0]
________________________________________________________________________________
__________________
activation_4 (Activation)       (None, None, None, 1 0
batch_normalization_9[0][0]
________________________________________________________________________________
__________________
max_pooling2d_1 (MaxPooling2D)  (None, None, None, 1 0
activation_4[0][0]
________________________________________________________________________________
__________________
conv2d_8 (Conv2D)               (None, None, None, 6 12288
max_pooling2d_1[0][0]
________________________________________________________________________________
__________________
batch_normalization_13 (BatchNo (None, None, None, 6 192         conv2d_8[0][0]
________________________________________________________________________________
__________________
activation_8 (Activation)       (None, None, None, 6 0
batch_normalization_13[0][0]
________________________________________________________________________________
__________________
conv2d_6 (Conv2D)               (None, None, None, 4 9216
max_pooling2d_1[0][0]
________________________________________________________________________________
__________________
conv2d_9 (Conv2D)               (None, None, None, 9 55296
activation_8[0][0]
________________________________________________________________________________
__________________
batch_normalization_11 (BatchNo (None, None, None, 4 144         conv2d_6[0][0]
________________________________________________________________________________
__________________
batch_normalization_14 (BatchNo (None, None, None, 9 288         conv2d_9[0][0]
________________________________________________________________________________
__________________
activation_6 (Activation)       (None, None, None, 4 0
batch_normalization_11[0][0]
________________________________________________________________________________
__________________
activation_9 (Activation)       (None, None, None, 9 0
batch_normalization_14[0][0]
________________________________________________________________________________
__________________
average_pooling2d (AveragePooli (None, None, None, 1 0
max_pooling2d_1[0][0]
```

```
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, None, None, 6 12288
max_pooling2d_1[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, None, None, 6 76800
activation_6[0][0]
__________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, None, None, 9 82944
activation_9[0][0]
__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, None, None, 3 6144
average_pooling2d[0][0]
__________________________________________________________________________________________________
batch_normalization_10 (BatchNo (None, None, None, 6 192         conv2d_5[0][0]
__________________________________________________________________________________________________
batch_normalization_12 (BatchNo (None, None, None, 6 192         conv2d_7[0][0]
__________________________________________________________________________________________________
batch_normalization_15 (BatchNo (None, None, None, 9 288         conv2d_10[0][0]
__________________________________________________________________________________________________
batch_normalization_16 (BatchNo (None, None, None, 3 96          conv2d_11[0][0]
__________________________________________________________________________________________________
activation_5 (Activation)       (None, None, None, 6 0
batch_normalization_10[0][0]
__________________________________________________________________________________________________
activation_7 (Activation)       (None, None, None, 6 0
batch_normalization_12[0][0]
__________________________________________________________________________________________________
activation_10 (Activation)      (None, None, None, 9 0
batch_normalization_15[0][0]
__________________________________________________________________________________________________
activation_11 (Activation)      (None, None, None, 3 0
batch_normalization_16[0][0]
__________________________________________________________________________________________________
mixed0 (Concatenate)            (None, None, None, 2 0
activation_5[0][0]
```

```
activation_7[0][0]
activation_10[0][0]
activation_11[0][0]
__________________________________________________________________________________
__________________
conv2d_15 (Conv2D)              (None, None, None, 6 16384       mixed0[0][0]
__________________________________________________________________________________
__________________
batch_normalization_20 (BatchNo (None, None, None, 6 192         conv2d_15[0][0]
__________________________________________________________________________________
__________________
activation_15 (Activation)      (None, None, None, 6 0
batch_normalization_20[0][0]
__________________________________________________________________________________
__________________
conv2d_13 (Conv2D)              (None, None, None, 4 12288       mixed0[0][0]
__________________________________________________________________________________
__________________
conv2d_16 (Conv2D)              (None, None, None, 9 55296
activation_15[0][0]
__________________________________________________________________________________
__________________
batch_normalization_18 (BatchNo (None, None, None, 4 144         conv2d_13[0][0]
__________________________________________________________________________________
__________________
batch_normalization_21 (BatchNo (None, None, None, 9 288         conv2d_16[0][0]
__________________________________________________________________________________
__________________
activation_13 (Activation)      (None, None, None, 4 0
batch_normalization_18[0][0]
__________________________________________________________________________________
__________________
activation_16 (Activation)      (None, None, None, 9 0
batch_normalization_21[0][0]
__________________________________________________________________________________
__________________
average_pooling2d_1 (AveragePoo (None, None, None, 2 0           mixed0[0][0]
__________________________________________________________________________________
__________________
conv2d_12 (Conv2D)              (None, None, None, 6 16384       mixed0[0][0]
__________________________________________________________________________________
__________________
conv2d_14 (Conv2D)              (None, None, None, 6 76800
activation_13[0][0]
__________________________________________________________________________________
__________________
conv2d_17 (Conv2D)              (None, None, None, 9 82944
activation_16[0][0]
```

```
__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, None, None, 6 16384
average_pooling2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_17 (BatchNo (None, None, None, 6 192         conv2d_12[0][0]
__________________________________________________________________________________________________
batch_normalization_19 (BatchNo (None, None, None, 6 192         conv2d_14[0][0]
__________________________________________________________________________________________________
batch_normalization_22 (BatchNo (None, None, None, 9 288         conv2d_17[0][0]
__________________________________________________________________________________________________
batch_normalization_23 (BatchNo (None, None, None, 6 192         conv2d_18[0][0]
__________________________________________________________________________________________________
activation_12 (Activation)      (None, None, None, 6 0
batch_normalization_17[0][0]
__________________________________________________________________________________________________
activation_14 (Activation)      (None, None, None, 6 0
batch_normalization_19[0][0]
__________________________________________________________________________________________________
activation_17 (Activation)      (None, None, None, 9 0
batch_normalization_22[0][0]
__________________________________________________________________________________________________
activation_18 (Activation)      (None, None, None, 6 0
batch_normalization_23[0][0]
__________________________________________________________________________________________________
mixed1 (Concatenate)            (None, None, None, 2 0
activation_12[0][0]
activation_14[0][0]
activation_17[0][0]
activation_18[0][0]
__________________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, None, None, 6 18432       mixed1[0][0]
__________________________________________________________________________________________________
batch_normalization_27 (BatchNo (None, None, None, 6 192         conv2d_22[0][0]
__________________________________________________________________________________________________
activation_22 (Activation)      (None, None, None, 6 0
```

```
batch_normalization_27[0][0]
__________________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, None, None, 4 13824       mixed1[0][0]
__________________________________________________________________________________________________
conv2d_23 (Conv2D)              (None, None, None, 9 55296
activation_22[0][0]
__________________________________________________________________________________________________
batch_normalization_25 (BatchNo (None, None, None, 4 144         conv2d_20[0][0]
__________________________________________________________________________________________________
batch_normalization_28 (BatchNo (None, None, None, 9 288         conv2d_23[0][0]
__________________________________________________________________________________________________
activation_20 (Activation)      (None, None, None, 4 0
batch_normalization_25[0][0]
__________________________________________________________________________________________________
activation_23 (Activation)      (None, None, None, 9 0
batch_normalization_28[0][0]
__________________________________________________________________________________________________
average_pooling2d_2 (AveragePoo (None, None, None, 2 0           mixed1[0][0]
__________________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, None, None, 6 18432       mixed1[0][0]
__________________________________________________________________________________________________
conv2d_21 (Conv2D)              (None, None, None, 6 76800
activation_20[0][0]
__________________________________________________________________________________________________
conv2d_24 (Conv2D)              (None, None, None, 9 82944
activation_23[0][0]
__________________________________________________________________________________________________
conv2d_25 (Conv2D)              (None, None, None, 6 18432
average_pooling2d_2[0][0]
__________________________________________________________________________________________________
batch_normalization_24 (BatchNo (None, None, None, 6 192         conv2d_19[0][0]
__________________________________________________________________________________________________
batch_normalization_26 (BatchNo (None, None, None, 6 192         conv2d_21[0][0]
__________________________________________________________________________________________________
```

```
batch_normalization_29 (BatchNo (None, None, None, 9 288         conv2d_24[0][0]
________________________________________________________________________________
__________________
batch_normalization_30 (BatchNo (None, None, None, 6 192         conv2d_25[0][0]
________________________________________________________________________________
__________________
activation_19 (Activation)      (None, None, None, 6 0
batch_normalization_24[0][0]
________________________________________________________________________________
__________________
activation_21 (Activation)      (None, None, None, 6 0
batch_normalization_26[0][0]
________________________________________________________________________________
__________________
activation_24 (Activation)      (None, None, None, 9 0
batch_normalization_29[0][0]
________________________________________________________________________________
__________________
activation_25 (Activation)      (None, None, None, 6 0
batch_normalization_30[0][0]
________________________________________________________________________________
__________________
mixed2 (Concatenate)            (None, None, None, 2 0
activation_19[0][0]
activation_21[0][0]
activation_24[0][0]
activation_25[0][0]
________________________________________________________________________________
__________________
conv2d_27 (Conv2D)              (None, None, None, 6 18432       mixed2[0][0]
________________________________________________________________________________
__________________
batch_normalization_32 (BatchNo (None, None, None, 6 192         conv2d_27[0][0]
________________________________________________________________________________
__________________
activation_27 (Activation)      (None, None, None, 6 0
batch_normalization_32[0][0]
________________________________________________________________________________
__________________
conv2d_28 (Conv2D)              (None, None, None, 9 55296
activation_27[0][0]
________________________________________________________________________________
__________________
batch_normalization_33 (BatchNo (None, None, None, 9 288         conv2d_28[0][0]
________________________________________________________________________________
__________________
activation_28 (Activation)      (None, None, None, 9 0
batch_normalization_33[0][0]
```

```
--------------------------------------------------------------------------------
------------------
conv2d_26 (Conv2D)             (None, None, None, 3 995328      mixed2[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_29 (Conv2D)             (None, None, None, 9 82944
activation_28[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_31 (BatchNo (None, None, None, 3 1152        conv2d_26[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_34 (BatchNo (None, None, None, 9 288         conv2d_29[0][0]
--------------------------------------------------------------------------------
------------------
activation_26 (Activation)      (None, None, None, 3 0
batch_normalization_31[0][0]
--------------------------------------------------------------------------------
------------------
activation_29 (Activation)      (None, None, None, 9 0
batch_normalization_34[0][0]
--------------------------------------------------------------------------------
------------------
max_pooling2d_2 (MaxPooling2D)  (None, None, None, 2 0           mixed2[0][0]
--------------------------------------------------------------------------------
------------------
mixed3 (Concatenate)            (None, None, None, 7 0
activation_26[0][0]
activation_29[0][0]
max_pooling2d_2[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_34 (Conv2D)              (None, None, None, 1 98304       mixed3[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_39 (BatchNo (None, None, None, 1 384         conv2d_34[0][0]
--------------------------------------------------------------------------------
------------------
activation_34 (Activation)      (None, None, None, 1 0
batch_normalization_39[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_35 (Conv2D)              (None, None, None, 1 114688
activation_34[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_40 (BatchNo (None, None, None, 1 384         conv2d_35[0][0]
--------------------------------------------------------------------------------
```

```
------------------
activation_35 (Activation)      (None, None, None, 1 0
batch_normalization_40[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_31 (Conv2D)              (None, None, None, 1 98304       mixed3[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_36 (Conv2D)              (None, None, None, 1 114688
activation_35[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_36 (BatchNo (None, None, None, 1 384         conv2d_31[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_41 (BatchNo (None, None, None, 1 384         conv2d_36[0][0]
--------------------------------------------------------------------------------
------------------
activation_31 (Activation)      (None, None, None, 1 0
batch_normalization_36[0][0]
--------------------------------------------------------------------------------
------------------
activation_36 (Activation)      (None, None, None, 1 0
batch_normalization_41[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_32 (Conv2D)              (None, None, None, 1 114688
activation_31[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_37 (Conv2D)              (None, None, None, 1 114688
activation_36[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_37 (BatchNo (None, None, None, 1 384         conv2d_32[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_42 (BatchNo (None, None, None, 1 384         conv2d_37[0][0]
--------------------------------------------------------------------------------
------------------
activation_32 (Activation)      (None, None, None, 1 0
batch_normalization_37[0][0]
--------------------------------------------------------------------------------
------------------
activation_37 (Activation)      (None, None, None, 1 0
batch_normalization_42[0][0]
--------------------------------------------------------------------------------
------------------
```

```
average_pooling2d_3 (AveragePoo (None, None, None, 7 0           mixed3[0][0]
__________________________________________________________________________________________________
conv2d_30 (Conv2D)              (None, None, None, 1 147456      mixed3[0][0]
__________________________________________________________________________________________________
conv2d_33 (Conv2D)              (None, None, None, 1 172032
activation_32[0][0]
__________________________________________________________________________________________________
conv2d_38 (Conv2D)              (None, None, None, 1 172032
activation_37[0][0]
__________________________________________________________________________________________________
conv2d_39 (Conv2D)              (None, None, None, 1 147456
average_pooling2d_3[0][0]
__________________________________________________________________________________________________
batch_normalization_35 (BatchNo (None, None, None, 1 576         conv2d_30[0][0]
__________________________________________________________________________________________________
batch_normalization_38 (BatchNo (None, None, None, 1 576         conv2d_33[0][0]
__________________________________________________________________________________________________
batch_normalization_43 (BatchNo (None, None, None, 1 576         conv2d_38[0][0]
__________________________________________________________________________________________________
batch_normalization_44 (BatchNo (None, None, None, 1 576         conv2d_39[0][0]
__________________________________________________________________________________________________
activation_30 (Activation)      (None, None, None, 1 0
batch_normalization_35[0][0]
__________________________________________________________________________________________________
activation_33 (Activation)      (None, None, None, 1 0
batch_normalization_38[0][0]
__________________________________________________________________________________________________
activation_38 (Activation)      (None, None, None, 1 0
batch_normalization_43[0][0]
__________________________________________________________________________________________________
activation_39 (Activation)      (None, None, None, 1 0
batch_normalization_44[0][0]
__________________________________________________________________________________________________
mixed4 (Concatenate)            (None, None, None, 7 0
activation_30[0][0]
```

```
activation_33[0][0]
activation_38[0][0]
activation_39[0][0]
__________________________________________________________________________________________________
conv2d_44 (Conv2D)              (None, None, None, 1 122880      mixed4[0][0]
__________________________________________________________________________________________________
batch_normalization_49 (BatchNo (None, None, None, 1 480         conv2d_44[0][0]
__________________________________________________________________________________________________
activation_44 (Activation)      (None, None, None, 1 0
batch_normalization_49[0][0]
__________________________________________________________________________________________________
conv2d_45 (Conv2D)              (None, None, None, 1 179200
activation_44[0][0]
__________________________________________________________________________________________________
batch_normalization_50 (BatchNo (None, None, None, 1 480         conv2d_45[0][0]
__________________________________________________________________________________________________
activation_45 (Activation)      (None, None, None, 1 0
batch_normalization_50[0][0]
__________________________________________________________________________________________________
conv2d_41 (Conv2D)              (None, None, None, 1 122880      mixed4[0][0]
__________________________________________________________________________________________________
conv2d_46 (Conv2D)              (None, None, None, 1 179200
activation_45[0][0]
__________________________________________________________________________________________________
batch_normalization_46 (BatchNo (None, None, None, 1 480         conv2d_41[0][0]
__________________________________________________________________________________________________
batch_normalization_51 (BatchNo (None, None, None, 1 480         conv2d_46[0][0]
__________________________________________________________________________________________________
activation_41 (Activation)      (None, None, None, 1 0
batch_normalization_46[0][0]
__________________________________________________________________________________________________
activation_46 (Activation)      (None, None, None, 1 0
batch_normalization_51[0][0]
__________________________________________________________________________________________________
conv2d_42 (Conv2D)              (None, None, None, 1 179200
```

```
activation_41[0][0]
__________________________________________________________________________________________________
conv2d_47 (Conv2D)              (None, None, None, 1 179200
activation_46[0][0]
__________________________________________________________________________________________________
batch_normalization_47 (BatchNo (None, None, None, 1 480         conv2d_42[0][0]
__________________________________________________________________________________________________
batch_normalization_52 (BatchNo (None, None, None, 1 480         conv2d_47[0][0]
__________________________________________________________________________________________________
activation_42 (Activation)      (None, None, None, 1 0
batch_normalization_47[0][0]
__________________________________________________________________________________________________
activation_47 (Activation)      (None, None, None, 1 0
batch_normalization_52[0][0]
__________________________________________________________________________________________________
average_pooling2d_4 (AveragePoo (None, None, None, 7 0           mixed4[0][0]
__________________________________________________________________________________________________
conv2d_40 (Conv2D)              (None, None, None, 1 147456      mixed4[0][0]
__________________________________________________________________________________________________
conv2d_43 (Conv2D)              (None, None, None, 1 215040
activation_42[0][0]
__________________________________________________________________________________________________
conv2d_48 (Conv2D)              (None, None, None, 1 215040
activation_47[0][0]
__________________________________________________________________________________________________
conv2d_49 (Conv2D)              (None, None, None, 1 147456
average_pooling2d_4[0][0]
__________________________________________________________________________________________________
batch_normalization_45 (BatchNo (None, None, None, 1 576         conv2d_40[0][0]
__________________________________________________________________________________________________
batch_normalization_48 (BatchNo (None, None, None, 1 576         conv2d_43[0][0]
__________________________________________________________________________________________________
batch_normalization_53 (BatchNo (None, None, None, 1 576         conv2d_48[0][0]
__________________________________________________________________________________________________
```

```
batch_normalization_54 (BatchNo (None, None, None, 1 576         conv2d_49[0][0]
--------------------------------------------------------------------------------
------------------
activation_40 (Activation)      (None, None, None, 1 0
batch_normalization_45[0][0]
--------------------------------------------------------------------------------
------------------
activation_43 (Activation)      (None, None, None, 1 0
batch_normalization_48[0][0]
--------------------------------------------------------------------------------
------------------
activation_48 (Activation)      (None, None, None, 1 0
batch_normalization_53[0][0]
--------------------------------------------------------------------------------
------------------
activation_49 (Activation)      (None, None, None, 1 0
batch_normalization_54[0][0]
--------------------------------------------------------------------------------
------------------
mixed5 (Concatenate)            (None, None, None, 7 0
activation_40[0][0]
activation_43[0][0]
activation_48[0][0]
activation_49[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_54 (Conv2D)              (None, None, None, 1 122880      mixed5[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_59 (BatchNo (None, None, None, 1 480         conv2d_54[0][0]
--------------------------------------------------------------------------------
------------------
activation_54 (Activation)      (None, None, None, 1 0
batch_normalization_59[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_55 (Conv2D)              (None, None, None, 1 179200
activation_54[0][0]
--------------------------------------------------------------------------------
------------------
batch_normalization_60 (BatchNo (None, None, None, 1 480         conv2d_55[0][0]
--------------------------------------------------------------------------------
------------------
activation_55 (Activation)      (None, None, None, 1 0
batch_normalization_60[0][0]
--------------------------------------------------------------------------------
------------------
conv2d_51 (Conv2D)              (None, None, None, 1 122880      mixed5[0][0]
```

```
----------------------------------------------------------------------------------
------------------
conv2d_56 (Conv2D)              (None, None, None, 1 179200
activation_55[0][0]
----------------------------------------------------------------------------------
------------------
batch_normalization_56 (BatchNo (None, None, None, 1 480         conv2d_51[0][0]
----------------------------------------------------------------------------------
------------------
batch_normalization_61 (BatchNo (None, None, None, 1 480         conv2d_56[0][0]
----------------------------------------------------------------------------------
------------------
activation_51 (Activation)      (None, None, None, 1 0
batch_normalization_56[0][0]
----------------------------------------------------------------------------------
------------------
activation_56 (Activation)      (None, None, None, 1 0
batch_normalization_61[0][0]
----------------------------------------------------------------------------------
------------------
conv2d_52 (Conv2D)              (None, None, None, 1 179200
activation_51[0][0]
----------------------------------------------------------------------------------
------------------
conv2d_57 (Conv2D)              (None, None, None, 1 179200
activation_56[0][0]
----------------------------------------------------------------------------------
------------------
batch_normalization_57 (BatchNo (None, None, None, 1 480         conv2d_52[0][0]
----------------------------------------------------------------------------------
------------------
batch_normalization_62 (BatchNo (None, None, None, 1 480         conv2d_57[0][0]
----------------------------------------------------------------------------------
------------------
activation_52 (Activation)      (None, None, None, 1 0
batch_normalization_57[0][0]
----------------------------------------------------------------------------------
------------------
activation_57 (Activation)      (None, None, None, 1 0
batch_normalization_62[0][0]
----------------------------------------------------------------------------------
------------------
average_pooling2d_5 (AveragePoo (None, None, None, 7 0           mixed5[0][0]
----------------------------------------------------------------------------------
------------------
conv2d_50 (Conv2D)              (None, None, None, 1 147456      mixed5[0][0]
----------------------------------------------------------------------------------
------------------
```

```
conv2d_53 (Conv2D)             (None, None, None, 1 215040
activation_52[0][0]
__________________________________________________________________________________
__________________
conv2d_58 (Conv2D)             (None, None, None, 1 215040
activation_57[0][0]
__________________________________________________________________________________
__________________
conv2d_59 (Conv2D)             (None, None, None, 1 147456
average_pooling2d_5[0][0]
__________________________________________________________________________________
__________________
batch_normalization_55 (BatchNo (None, None, None, 1 576         conv2d_50[0][0]
__________________________________________________________________________________
__________________
batch_normalization_58 (BatchNo (None, None, None, 1 576         conv2d_53[0][0]
__________________________________________________________________________________
__________________
batch_normalization_63 (BatchNo (None, None, None, 1 576         conv2d_58[0][0]
__________________________________________________________________________________
__________________
batch_normalization_64 (BatchNo (None, None, None, 1 576         conv2d_59[0][0]
__________________________________________________________________________________
__________________
activation_50 (Activation)      (None, None, None, 1 0
batch_normalization_55[0][0]
__________________________________________________________________________________
__________________
activation_53 (Activation)      (None, None, None, 1 0
batch_normalization_58[0][0]
__________________________________________________________________________________
__________________
activation_58 (Activation)      (None, None, None, 1 0
batch_normalization_63[0][0]
__________________________________________________________________________________
__________________
activation_59 (Activation)      (None, None, None, 1 0
batch_normalization_64[0][0]
__________________________________________________________________________________
__________________
mixed6 (Concatenate)            (None, None, None, 7 0
activation_50[0][0]
activation_53[0][0]
activation_58[0][0]
activation_59[0][0]
__________________________________________________________________________________
__________________
conv2d_64 (Conv2D)             (None, None, None, 1 147456      mixed6[0][0]
```

```
__________________________________________________________________________________________________
batch_normalization_69 (BatchNo (None, None, None, 1 576         conv2d_64[0][0]
__________________________________________________________________________________________________
activation_64 (Activation)      (None, None, None, 1 0
batch_normalization_69[0][0]
__________________________________________________________________________________________________
conv2d_65 (Conv2D)              (None, None, None, 1 258048
activation_64[0][0]
__________________________________________________________________________________________________
batch_normalization_70 (BatchNo (None, None, None, 1 576         conv2d_65[0][0]
__________________________________________________________________________________________________
activation_65 (Activation)      (None, None, None, 1 0
batch_normalization_70[0][0]
__________________________________________________________________________________________________
conv2d_61 (Conv2D)              (None, None, None, 1 147456      mixed6[0][0]
__________________________________________________________________________________________________
conv2d_66 (Conv2D)              (None, None, None, 1 258048
activation_65[0][0]
__________________________________________________________________________________________________
batch_normalization_66 (BatchNo (None, None, None, 1 576         conv2d_61[0][0]
__________________________________________________________________________________________________
batch_normalization_71 (BatchNo (None, None, None, 1 576         conv2d_66[0][0]
__________________________________________________________________________________________________
activation_61 (Activation)      (None, None, None, 1 0
batch_normalization_66[0][0]
__________________________________________________________________________________________________
activation_66 (Activation)      (None, None, None, 1 0
batch_normalization_71[0][0]
__________________________________________________________________________________________________
conv2d_62 (Conv2D)              (None, None, None, 1 258048
activation_61[0][0]
__________________________________________________________________________________________________
conv2d_67 (Conv2D)              (None, None, None, 1 258048
activation_66[0][0]
__________________________________________________________________________________________________
```

```
__________________
batch_normalization_67 (BatchNo (None, None, None, 1 576         conv2d_62[0][0]
________________________________________________________________________________
__________________
batch_normalization_72 (BatchNo (None, None, None, 1 576         conv2d_67[0][0]
________________________________________________________________________________
__________________
activation_62 (Activation)      (None, None, None, 1 0
batch_normalization_67[0][0]
________________________________________________________________________________
__________________
activation_67 (Activation)      (None, None, None, 1 0
batch_normalization_72[0][0]
________________________________________________________________________________
__________________
average_pooling2d_6 (AveragePoo (None, None, None, 7 0           mixed6[0][0]
________________________________________________________________________________
__________________
conv2d_60 (Conv2D)              (None, None, None, 1 147456      mixed6[0][0]
________________________________________________________________________________
__________________
conv2d_63 (Conv2D)              (None, None, None, 1 258048
activation_62[0][0]
________________________________________________________________________________
__________________
conv2d_68 (Conv2D)              (None, None, None, 1 258048
activation_67[0][0]
________________________________________________________________________________
__________________
conv2d_69 (Conv2D)              (None, None, None, 1 147456
average_pooling2d_6[0][0]
________________________________________________________________________________
__________________
batch_normalization_65 (BatchNo (None, None, None, 1 576         conv2d_60[0][0]
________________________________________________________________________________
__________________
batch_normalization_68 (BatchNo (None, None, None, 1 576         conv2d_63[0][0]
________________________________________________________________________________
__________________
batch_normalization_73 (BatchNo (None, None, None, 1 576         conv2d_68[0][0]
________________________________________________________________________________
__________________
batch_normalization_74 (BatchNo (None, None, None, 1 576         conv2d_69[0][0]
________________________________________________________________________________
__________________
activation_60 (Activation)      (None, None, None, 1 0
batch_normalization_65[0][0]
________________________________________________________________________________
```

```
------------------
activation_63 (Activation)      (None, None, None, 1 0
batch_normalization_68[0][0]
________________________________________________________________________________
------------------
activation_68 (Activation)      (None, None, None, 1 0
batch_normalization_73[0][0]
________________________________________________________________________________
------------------
activation_69 (Activation)      (None, None, None, 1 0
batch_normalization_74[0][0]
________________________________________________________________________________
------------------
mixed7 (Concatenate)            (None, None, None, 7 0
activation_60[0][0]
activation_63[0][0]
activation_68[0][0]
activation_69[0][0]
________________________________________________________________________________
------------------
conv2d_72 (Conv2D)              (None, None, None, 1 147456      mixed7[0][0]
________________________________________________________________________________
------------------
batch_normalization_77 (BatchNo (None, None, None, 1 576         conv2d_72[0][0]
________________________________________________________________________________
------------------
activation_72 (Activation)      (None, None, None, 1 0
batch_normalization_77[0][0]
________________________________________________________________________________
------------------
conv2d_73 (Conv2D)              (None, None, None, 1 258048
activation_72[0][0]
________________________________________________________________________________
------------------
batch_normalization_78 (BatchNo (None, None, None, 1 576         conv2d_73[0][0]
________________________________________________________________________________
------------------
activation_73 (Activation)      (None, None, None, 1 0
batch_normalization_78[0][0]
________________________________________________________________________________
------------------
conv2d_70 (Conv2D)              (None, None, None, 1 147456      mixed7[0][0]
________________________________________________________________________________
------------------
conv2d_74 (Conv2D)              (None, None, None, 1 258048
activation_73[0][0]
________________________________________________________________________________
------------------
```

```
batch_normalization_75 (BatchNo (None, None, None, 1 576         conv2d_70[0][0]
________________________________________________________________________________
__________________
batch_normalization_79 (BatchNo (None, None, None, 1 576         conv2d_74[0][0]
________________________________________________________________________________
__________________
activation_70 (Activation)      (None, None, None, 1 0
batch_normalization_75[0][0]
________________________________________________________________________________
__________________
activation_74 (Activation)      (None, None, None, 1 0
batch_normalization_79[0][0]
________________________________________________________________________________
__________________
conv2d_71 (Conv2D)              (None, None, None, 3 552960
activation_70[0][0]
________________________________________________________________________________
__________________
conv2d_75 (Conv2D)              (None, None, None, 1 331776
activation_74[0][0]
________________________________________________________________________________
__________________
batch_normalization_76 (BatchNo (None, None, None, 3 960         conv2d_71[0][0]
________________________________________________________________________________
__________________
batch_normalization_80 (BatchNo (None, None, None, 1 576         conv2d_75[0][0]
________________________________________________________________________________
__________________
activation_71 (Activation)      (None, None, None, 3 0
batch_normalization_76[0][0]
________________________________________________________________________________
__________________
activation_75 (Activation)      (None, None, None, 1 0
batch_normalization_80[0][0]
________________________________________________________________________________
__________________
max_pooling2d_3 (MaxPooling2D)  (None, None, None, 7 0           mixed7[0][0]
________________________________________________________________________________
__________________
mixed8 (Concatenate)            (None, None, None, 1 0
activation_71[0][0]
activation_75[0][0]
max_pooling2d_3[0][0]
________________________________________________________________________________
__________________
conv2d_80 (Conv2D)              (None, None, None, 4 573440      mixed8[0][0]
________________________________________________________________________________
__________________
```

```
batch_normalization_85 (BatchNo (None, None, None, 4 1344        conv2d_80[0][0]
________________________________________________________________________________
__________________
activation_80 (Activation)      (None, None, None, 4 0
batch_normalization_85[0][0]
________________________________________________________________________________
__________________
conv2d_77 (Conv2D)              (None, None, None, 3 491520      mixed8[0][0]
________________________________________________________________________________
__________________
conv2d_81 (Conv2D)              (None, None, None, 3 1548288
activation_80[0][0]
________________________________________________________________________________
__________________
batch_normalization_82 (BatchNo (None, None, None, 3 1152        conv2d_77[0][0]
________________________________________________________________________________
__________________
batch_normalization_86 (BatchNo (None, None, None, 3 1152        conv2d_81[0][0]
________________________________________________________________________________
__________________
activation_77 (Activation)      (None, None, None, 3 0
batch_normalization_82[0][0]
________________________________________________________________________________
__________________
activation_81 (Activation)      (None, None, None, 3 0
batch_normalization_86[0][0]
________________________________________________________________________________
__________________
conv2d_78 (Conv2D)              (None, None, None, 3 442368
activation_77[0][0]
________________________________________________________________________________
__________________
conv2d_79 (Conv2D)              (None, None, None, 3 442368
activation_77[0][0]
________________________________________________________________________________
__________________
conv2d_82 (Conv2D)              (None, None, None, 3 442368
activation_81[0][0]
________________________________________________________________________________
__________________
conv2d_83 (Conv2D)              (None, None, None, 3 442368
activation_81[0][0]
________________________________________________________________________________
__________________
average_pooling2d_7 (AveragePoo (None, None, None, 1 0           mixed8[0][0]
________________________________________________________________________________
__________________
conv2d_76 (Conv2D)              (None, None, None, 3 409600      mixed8[0][0]
```

```
__________________________________________________________________________________________________
batch_normalization_83 (BatchNo (None, None, None, 3 1152        conv2d_78[0][0]
__________________________________________________________________________________________________
batch_normalization_84 (BatchNo (None, None, None, 3 1152        conv2d_79[0][0]
__________________________________________________________________________________________________
batch_normalization_87 (BatchNo (None, None, None, 3 1152        conv2d_82[0][0]
__________________________________________________________________________________________________
batch_normalization_88 (BatchNo (None, None, None, 3 1152        conv2d_83[0][0]
__________________________________________________________________________________________________
conv2d_84 (Conv2D)              (None, None, None, 1 245760
average_pooling2d_7[0][0]
__________________________________________________________________________________________________
batch_normalization_81 (BatchNo (None, None, None, 3 960         conv2d_76[0][0]
__________________________________________________________________________________________________
activation_78 (Activation)      (None, None, None, 3 0
batch_normalization_83[0][0]
__________________________________________________________________________________________________
activation_79 (Activation)      (None, None, None, 3 0
batch_normalization_84[0][0]
__________________________________________________________________________________________________
activation_82 (Activation)      (None, None, None, 3 0
batch_normalization_87[0][0]
__________________________________________________________________________________________________
activation_83 (Activation)      (None, None, None, 3 0
batch_normalization_88[0][0]
__________________________________________________________________________________________________
batch_normalization_89 (BatchNo (None, None, None, 1 576         conv2d_84[0][0]
__________________________________________________________________________________________________
activation_76 (Activation)      (None, None, None, 3 0
batch_normalization_81[0][0]
__________________________________________________________________________________________________
mixed9_0 (Concatenate)          (None, None, None, 7 0
activation_78[0][0]
activation_79[0][0]
________________________________________________________________________________
```

```
------------------
concatenate (Concatenate)       (None, None, None, 7 0
activation_82[0][0]
activation_83[0][0]
________________________________________________________________________________
------------------
activation_84 (Activation)      (None, None, None, 1 0
batch_normalization_89[0][0]
________________________________________________________________________________
------------------
mixed9 (Concatenate)            (None, None, None, 2 0
activation_76[0][0]
                                                                 mixed9_0[0][0]
concatenate[0][0]
activation_84[0][0]
________________________________________________________________________________
------------------
conv2d_89 (Conv2D)              (None, None, None, 4 917504      mixed9[0][0]
________________________________________________________________________________
------------------
batch_normalization_94 (BatchNo (None, None, None, 4 1344        conv2d_89[0][0]
________________________________________________________________________________
------------------
activation_89 (Activation)      (None, None, None, 4 0
batch_normalization_94[0][0]
________________________________________________________________________________
------------------
conv2d_86 (Conv2D)              (None, None, None, 3 786432      mixed9[0][0]
________________________________________________________________________________
------------------
conv2d_90 (Conv2D)              (None, None, None, 3 1548288
activation_89[0][0]
________________________________________________________________________________
------------------
batch_normalization_91 (BatchNo (None, None, None, 3 1152        conv2d_86[0][0]
________________________________________________________________________________
------------------
batch_normalization_95 (BatchNo (None, None, None, 3 1152        conv2d_90[0][0]
________________________________________________________________________________
------------------
activation_86 (Activation)      (None, None, None, 3 0
batch_normalization_91[0][0]
________________________________________________________________________________
------------------
activation_90 (Activation)      (None, None, None, 3 0
batch_normalization_95[0][0]
________________________________________________________________________________
------------------
```

```
conv2d_87 (Conv2D)              (None, None, None, 3 442368
activation_86[0][0]
__________________________________________________________________________________________________
conv2d_88 (Conv2D)              (None, None, None, 3 442368
activation_86[0][0]
__________________________________________________________________________________________________
conv2d_91 (Conv2D)              (None, None, None, 3 442368
activation_90[0][0]
__________________________________________________________________________________________________
conv2d_92 (Conv2D)              (None, None, None, 3 442368
activation_90[0][0]
__________________________________________________________________________________________________
average_pooling2d_8 (AveragePoo (None, None, None, 2 0           mixed9[0][0]
__________________________________________________________________________________________________
conv2d_85 (Conv2D)              (None, None, None, 3 655360      mixed9[0][0]
__________________________________________________________________________________________________
batch_normalization_92 (BatchNo (None, None, None, 3 1152        conv2d_87[0][0]
__________________________________________________________________________________________________
batch_normalization_93 (BatchNo (None, None, None, 3 1152        conv2d_88[0][0]
__________________________________________________________________________________________________
batch_normalization_96 (BatchNo (None, None, None, 3 1152        conv2d_91[0][0]
__________________________________________________________________________________________________
batch_normalization_97 (BatchNo (None, None, None, 3 1152        conv2d_92[0][0]
__________________________________________________________________________________________________
conv2d_93 (Conv2D)              (None, None, None, 1 393216
average_pooling2d_8[0][0]
__________________________________________________________________________________________________
batch_normalization_90 (BatchNo (None, None, None, 3 960         conv2d_85[0][0]
__________________________________________________________________________________________________
activation_87 (Activation)      (None, None, None, 3 0
batch_normalization_92[0][0]
__________________________________________________________________________________________________
activation_88 (Activation)      (None, None, None, 3 0
batch_normalization_93[0][0]
__________________________________________________________________________________
```

```
------------------
activation_91 (Activation)      (None, None, None, 3 0
batch_normalization_96[0][0]
________________________________________________________________________________
------------------
activation_92 (Activation)      (None, None, None, 3 0
batch_normalization_97[0][0]
________________________________________________________________________________
------------------
batch_normalization_98 (BatchNo (None, None, None, 1 576         conv2d_93[0][0]
________________________________________________________________________________
------------------
activation_85 (Activation)      (None, None, None, 3 0
batch_normalization_90[0][0]
________________________________________________________________________________
------------------
mixed9_1 (Concatenate)          (None, None, None, 7 0
activation_87[0][0]
activation_88[0][0]
________________________________________________________________________________
------------------
concatenate_1 (Concatenate)     (None, None, None, 7 0
activation_91[0][0]
activation_92[0][0]
________________________________________________________________________________
------------------
activation_93 (Activation)      (None, None, None, 1 0
batch_normalization_98[0][0]
________________________________________________________________________________
------------------
mixed10 (Concatenate)           (None, None, None, 2 0
activation_85[0][0]
                                                                 mixed9_1[0][0]
concatenate_1[0][0]
activation_93[0][0]
================================================================================
==================
Total params: 21,802,784
Trainable params: 21,768,352
Non-trainable params: 34,432
________________________________________________________________________________
------------------
```

6.0.2 Extract features

With this model, as oppose to the 2nd approach, we loaded the features from our training and testing set using the pretrained model, previously imported.

This step took almost 1 hour.

```
[0]: # extract train and val features
print('Extracting train features...', end='')
pretrained_features_train = pretrained_model.predict(train_generator)
print('Done')
print('Extracting test features...', end='')
pretrained_features_test = pretrained_model.predict(test_generator)
print('Done')
```

```
Extracting train features...Done
Extracting test features...Done
```

Extract the labels

```
[0]: from tensorflow.keras.utils import to_categorical

# OHE target column
train_target = to_categorical(train_generator.labels)
test_target = to_categorical(test_generator.labels)
```

6.0.3 Build model

On top of the pretrained model, we built a simple **Sequential** model, with a **Flatten** and a **Dense** layer.

```
[0]: model = Sequential()
model.add(Flatten(input_shape=(6, 6, 2048)))
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.5))
model.add(BatchNormalization())

class_num = 5
model.add(Dense(class_num, activation='softmax'))

# compile the model
model.compile(optimizer='adam', metrics=['accuracy'],␣
 ↪loss='categorical_crossentropy')

model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_8 (Flatten)          (None, 73728)             0
_________________________________________________________________
dense_15 (Dense)             (None, 100)               7372900
_________________________________________________________________
```

```
dropout_7 (Dropout)          (None, 100)              0
_________________________________________________________________
batch_normalization_101 (Bat (None, 100)              400
_________________________________________________________________
dense_16 (Dense)             (None, 5)                505
=================================================================
Total params: 7,373,805
Trainable params: 7,373,605
Non-trainable params: 200
_________________________________________________________________
```

6.0.4 Train

For the training, we fed the features (previously processed by the pretrained model) into the new model.

This was really fast, because our model was very simple, and the pretrained model had already done most of the processing.

[0]:

```
# train model using features generated from VGG16 model
history = model.fit(pretrained_features_train, train_target, epochs=50,␣
↪batch_size=128, validation_data=(pretrained_features_test, test_target))
```

```
Train on 3754 samples, validate on 1249 samples
Epoch 1/50
3754/3754 [==============================] - 5s 1ms/sample - loss: 0.9666 -
accuracy: 0.6393 - val_loss: 0.7803 - val_accuracy: 0.7598
Epoch 2/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.6109 -
accuracy: 0.7869 - val_loss: 0.5645 - val_accuracy: 0.8135
Epoch 3/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.4697 -
accuracy: 0.8420 - val_loss: 0.4549 - val_accuracy: 0.8271
Epoch 4/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.3565 -
accuracy: 0.8918 - val_loss: 0.5003 - val_accuracy: 0.8151
Epoch 5/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.2894 -
accuracy: 0.9193 - val_loss: 0.4385 - val_accuracy: 0.8359
Epoch 6/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.2167 -
accuracy: 0.9451 - val_loss: 0.4513 - val_accuracy: 0.8335
Epoch 7/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.1676 -
accuracy: 0.9632 - val_loss: 0.4616 - val_accuracy: 0.8327
Epoch 8/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.1322 -
accuracy: 0.9752 - val_loss: 0.4895 - val_accuracy: 0.8279
Epoch 9/50
```

```
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.1075 -
accuracy: 0.9827 - val_loss: 0.4398 - val_accuracy: 0.8407
Epoch 10/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0930 -
accuracy: 0.9830 - val_loss: 0.4558 - val_accuracy: 0.8359
Epoch 11/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0780 -
accuracy: 0.9864 - val_loss: 0.4462 - val_accuracy: 0.8431
Epoch 12/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0617 -
accuracy: 0.9907 - val_loss: 0.4499 - val_accuracy: 0.8495
Epoch 13/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0591 -
accuracy: 0.9901 - val_loss: 0.4654 - val_accuracy: 0.8191
Epoch 14/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0565 -
accuracy: 0.9909 - val_loss: 0.4789 - val_accuracy: 0.8431
Epoch 15/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0514 -
accuracy: 0.9899 - val_loss: 0.4588 - val_accuracy: 0.8463
Epoch 16/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0426 -
accuracy: 0.9928 - val_loss: 0.5033 - val_accuracy: 0.8431
Epoch 17/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0358 -
accuracy: 0.9952 - val_loss: 0.4880 - val_accuracy: 0.8439
Epoch 18/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0349 -
accuracy: 0.9952 - val_loss: 0.4835 - val_accuracy: 0.8383
Epoch 19/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0324 -
accuracy: 0.9941 - val_loss: 0.4840 - val_accuracy: 0.8487
Epoch 20/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0279 -
accuracy: 0.9976 - val_loss: 0.5150 - val_accuracy: 0.8415
Epoch 21/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0295 -
accuracy: 0.9947 - val_loss: 0.5686 - val_accuracy: 0.8351
Epoch 22/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0307 -
accuracy: 0.9947 - val_loss: 0.5852 - val_accuracy: 0.8263
Epoch 23/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0264 -
accuracy: 0.9965 - val_loss: 0.5451 - val_accuracy: 0.8439
Epoch 24/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0265 -
accuracy: 0.9952 - val_loss: 0.5309 - val_accuracy: 0.8455
Epoch 25/50
```

```
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0235 -
accuracy: 0.9955 - val_loss: 0.5355 - val_accuracy: 0.8431
Epoch 26/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0221 -
accuracy: 0.9968 - val_loss: 0.5162 - val_accuracy: 0.8431
Epoch 27/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0199 -
accuracy: 0.9984 - val_loss: 0.5650 - val_accuracy: 0.8319
Epoch 28/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0229 -
accuracy: 0.9960 - val_loss: 0.5747 - val_accuracy: 0.8399
Epoch 29/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0245 -
accuracy: 0.9963 - val_loss: 0.5579 - val_accuracy: 0.8455
Epoch 30/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0275 -
accuracy: 0.9944 - val_loss: 0.6037 - val_accuracy: 0.8335
Epoch 31/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0332 -
accuracy: 0.9928 - val_loss: 0.5943 - val_accuracy: 0.8367
Epoch 32/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0350 -
accuracy: 0.9925 - val_loss: 0.6643 - val_accuracy: 0.8110
Epoch 33/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0384 -
accuracy: 0.9904 - val_loss: 0.5620 - val_accuracy: 0.8319
Epoch 34/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0415 -
accuracy: 0.9899 - val_loss: 0.5372 - val_accuracy: 0.8327
Epoch 35/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0347 -
accuracy: 0.9936 - val_loss: 0.5607 - val_accuracy: 0.8527
Epoch 36/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0307 -
accuracy: 0.9939 - val_loss: 0.5528 - val_accuracy: 0.8439
Epoch 37/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0427 -
accuracy: 0.9893 - val_loss: 0.6695 - val_accuracy: 0.8215
Epoch 38/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0494 -
accuracy: 0.9861 - val_loss: 0.5651 - val_accuracy: 0.8367
Epoch 39/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0373 -
accuracy: 0.9923 - val_loss: 0.6244 - val_accuracy: 0.8367
Epoch 40/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0405 -
accuracy: 0.9901 - val_loss: 0.6138 - val_accuracy: 0.8407
Epoch 41/50
```

```
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0446 -
accuracy: 0.9888 - val_loss: 0.6414 - val_accuracy: 0.8311
Epoch 42/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0474 -
accuracy: 0.9907 - val_loss: 0.5497 - val_accuracy: 0.8431
Epoch 43/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0375 -
accuracy: 0.9901 - val_loss: 0.5839 - val_accuracy: 0.8359
Epoch 44/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0397 -
accuracy: 0.9899 - val_loss: 0.5713 - val_accuracy: 0.8399
Epoch 45/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0420 -
accuracy: 0.9883 - val_loss: 0.8006 - val_accuracy: 0.8062
Epoch 46/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0452 -
accuracy: 0.9864 - val_loss: 0.5970 - val_accuracy: 0.8271
Epoch 47/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0533 -
accuracy: 0.9864 - val_loss: 0.6006 - val_accuracy: 0.8223
Epoch 48/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0665 -
accuracy: 0.9798 - val_loss: 0.6167 - val_accuracy: 0.8423
Epoch 49/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0640 -
accuracy: 0.9822 - val_loss: 0.6072 - val_accuracy: 0.8391
Epoch 50/50
3754/3754 [==============================] - 4s 1ms/sample - loss: 0.0464 -
accuracy: 0.9872 - val_loss: 0.6093 - val_accuracy: 0.8471
```

6.0.5 Evaluate the model

[0]:
```
#Evaluate the model on the test data
score  = model.evaluate(pretrained_features_test, test_target)

#Accuracy on test data
print('Accuracy on the Test Images: ', score[1])
```

```
1249/1249 [==============================] - 1s 583us/sample - loss: 0.6093 -
accuracy: 0.8471
Accuracy on the Test Images:  0.84707767
```

We got a very good accuracy, as the results (shown ahead) will confirm.

RESULTS: Accuracy of 80%, taking about 1 hour.

7 Save model

[0]:
```
# serialize model to JSON
model_json = model.to_json()
with open("model3.json", "w") as json_file:
    json_file.write(model_json)

# serialize weights to HDF5
model.save_weights("model3.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

[0]:
```
# serialize ix_to_class
with open("ix_to_class.json", "w") as json_file:
  json_file.write(json.dumps(ix_to_class))
```

8 Load model

[0]:
```
from tensorflow.keras.models import model_from_json

# load json and create model
json_file = open('/content/drive/My Drive/Colab Notebooks/model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("/content/drive/My Drive/Colab Notebooks/model.h5")
print("Loaded model from disk")

model = loaded_model;
```

```
    ---------------------------------------------------------------------------

    FileNotFoundError                         Traceback (most recent call last)

    <ipython-input-2-c492095b1b11> in <module>()
          2
          3 # load json and create model
    ----> 4 json_file = open('/content/drive/My Drive/Colab Notebooks/model.json',␣
↪'r')
          5 loaded_model_json = json_file.read()
          6 json_file.close()
```

```
FileNotFoundError: [Errno 2] No such file or directory: '/content/drive/My␣
↪Drive/Colab Notebooks/model.json'
```

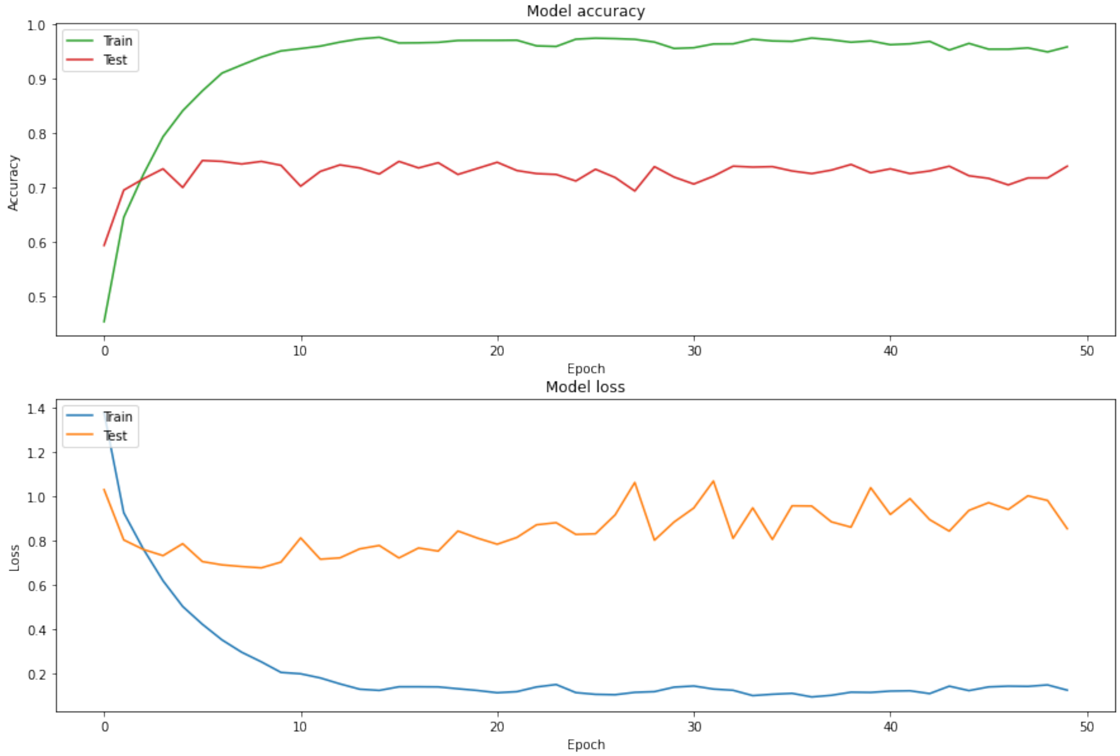
```
[0]: # read ix_to_class
with open("/content/drive/My Drive/Colab Notebooks/ix_to_class.json") as␣
 ↪json_file:
  loaded_ix_to_class = json_file.read()

ix_to_class = json.loads(loaded_ix_to_class)

print(ix_to_class)
```

```
{'0': 'apple_pie', '1': 'baby_back_ribs', '2': 'baklava', '3': 'beef_carpaccio',
'4': 'beef_tartare'}
```

9 Evaluation

9.0.1 Plot history

```
[0]: def plot_hist(history):
    f,ax = plt.subplots(2,1,figsize=(15,10))
    ax[0].plot(history.history['accuracy'],c='C2')
    ax[0].plot(history.history['val_accuracy'],c='C3')
    ax[0].set_title('Model accuracy')
    ax[0].set_ylabel('Accuracy')
    ax[0].set_xlabel('Epoch')
    ax[0].legend(['Train', 'Test'], loc='upper left')

    # summarize history for loss
    ax[1].plot(history.history['loss'],c='C0')
    ax[1].plot(history.history['val_loss'],c='C1')
    ax[1].set_title('Model loss')
    ax[1].set_ylabel('Loss')
    ax[1].set_xlabel('Epoch')
    ax[1].legend(['Train', 'Test'], loc='upper left')

plot_hist(history)
```



9.0.2 Accuracy Score

[0]:
```
base_dir = '/content/drive/My Drive/Colab Notebooks/food-5/'

test_dir = os.path.join(base_dir, 'test')

# each class will be inside here

# all images will be rescaled by 1./255
val_datagen = ImageDataGenerator(rescale=1.0/255.)

# flow validation images in batches of 20 using train_datagen
val_generator = val_datagen.flow_from_directory(test_dir,
                                                batch_size=128,
                                                class_mode='categorical',
                                                shuffle=False,
                                                target_size=(244, 244))
```

```
Found 1249 images belonging to 5 classes.
```

[0]:
```
from sklearn.metrics import accuracy_score
```

```
x_test, y_test = val_generator.next()
y_pred_conf = model.predict(x_test) #return probabilities of each class
y_pred = np.argmax(y_pred_conf,axis=1)
y_label = np.argmax(y_test,axis=1)

print('Accuracy score: {:.1f}%'.format(accuracy_score(y_pred,y_label)*100))
```

For the **3rd attemp** (with transfer learning), we need to apply another layer of processing.

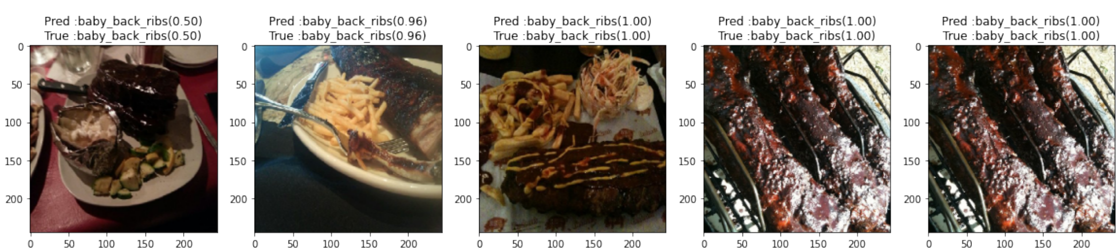
First, we extract the features using the pretrained model, and then we predict with our model.

```
[0]: from sklearn.metrics import accuracy_score

x_test, y_test = val_generator.next()
features = pretrained_model.predict(x_test)
y_pred_conf = model.predict(features) #return probabilities of each class
y_pred = np.argmax(y_pred_conf,axis=1)
y_label = np.argmax(y_test,axis=1)

print('Accuracy score: {:.1f}%'.format(accuracy_score(y_pred,y_label)*100))
```

```
Accuracy score: 89.1%
```

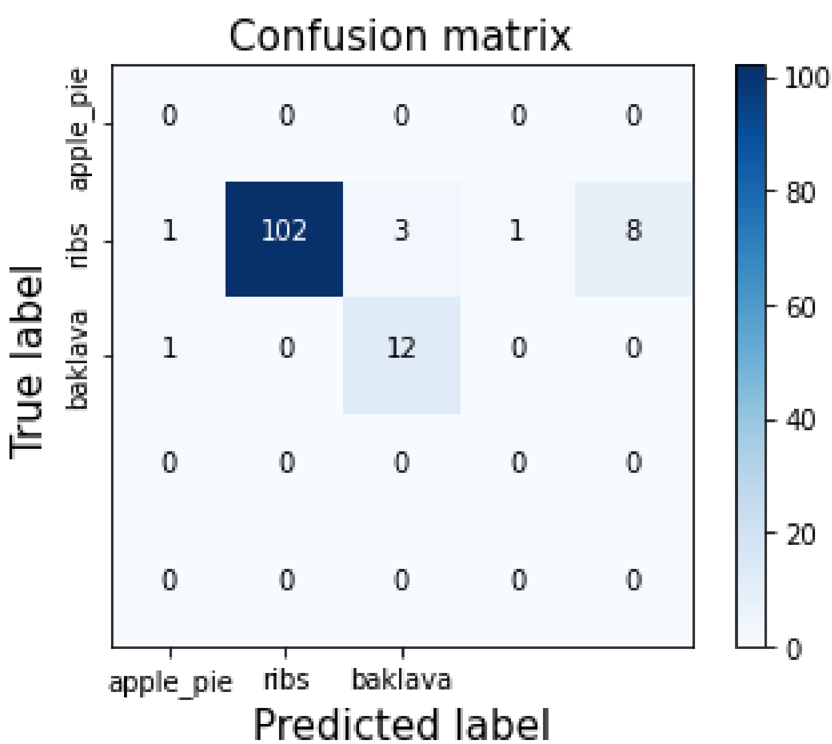
9.0.3 Check 5 random images

```
[0]: ind = np.random.randint(1,len(x_test),5)
f, ax=plt.subplots(1,5,figsize=(20,10))
for i,j in enumerate(ind):
    ax[i].imshow(x_test[j])
    ax[i].set_title("Pred :{}({:.2f})\nTrue :{}({:.2f})".format
                            (ix_to_class[str(y_pred[j])],np.max(y_pred_conf[j]),
                            ␣
 ↪ix_to_class[str(y_label[j])],y_pred_conf[j][(y_label[j])],fontweight="bold",␣
 ↪size=20))
```



9.0.4 Confusion matrix

[0]:
```
from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(cm):
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title('Confusion matrix',fontsize=15)
    plt.colorbar()
    classes = ['apple_pie','ribs','baklava']
    plt.xticks([0,1,2], classes, fontsize=10)
    plt.yticks([0,1,2], classes,␣
 ↪fontsize=10,rotation=90,verticalalignment="center")
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            plt.text(j, i, format(cm[i, j], 'd'), horizontalalignment="center",␣
 ↪color="white" if cm[i, j] > np.max(cm)/2. else "black")
    plt.xlabel('Predicted label',fontsize=15)
    plt.ylabel('True label',fontsize=15)

plot_confusion_matrix(confusion_matrix(y_label,y_pred))
```

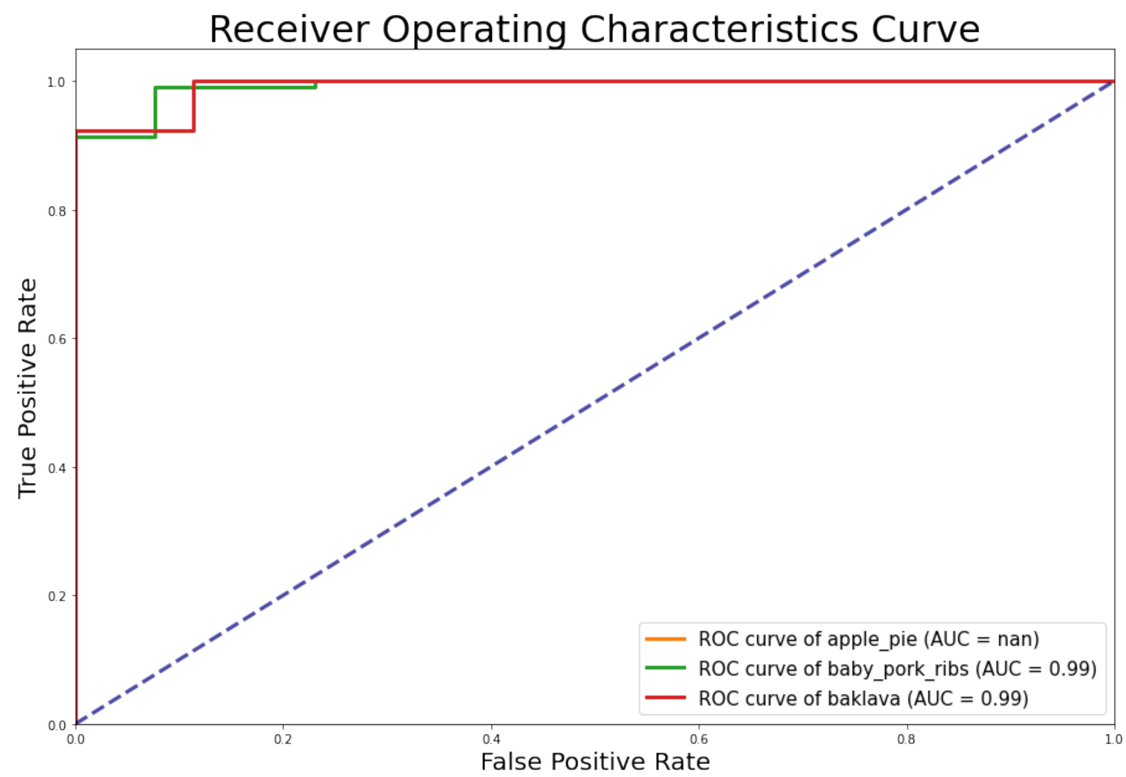


9.0.5 Receiver Operating Characterisics (ROC) Curve

[0]:
```
from sklearn.metrics import roc_curve, auc

fpr = dict() # false positive rate
tpr = dict() # true positive rate
roc_auc = dict() # area under roc curve
for i in range(3):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_pred_conf[:, i]) # roc_curve␣
 ↪function apply to binary class only
    roc_auc[i] = auc(fpr[i], tpr[i])  # using the trapezoidal rule to get area␣
 ↪under curve
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_ranking.py:808:
UndefinedMetricWarning: No positive samples in y_true, true positive value
should be meaningless
  UndefinedMetricWarning)
```

[0]:
```
def plot_roc(fpr,tpr,roc_auc):
    plt.figure(figsize=(15,10))
    plt.plot(fpr[0], tpr[0], color='C1', lw=3, label='ROC curve of apple_pie␣
 ↪(AUC = %0.2f)' % roc_auc[0])
    plt.plot(fpr[1], tpr[1], color='C2', lw=3, label='ROC curve of␣
 ↪baby_pork_ribs (AUC = %0.2f)' % roc_auc[1])
    plt.plot(fpr[2], tpr[2], color='C3', lw=3, label='ROC curve of baklava (AUC␣
 ↪= %0.2f)' % roc_auc[2])
    plt.plot([0, 1], [0, 1], color='navy', lw=3, linestyle='--',alpha=0.7)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate',fontsize=20)
    plt.ylabel('True Positive Rate',fontsize=20)
    plt.title('Receiver Operating Characteristics Curve',fontsize=30)
    plt.legend(loc="lower right",fontsize=15)
    plt.show()

plot_roc(fpr,tpr,roc_auc)
```



9.0.6 Inspect the predictions with wrong labels

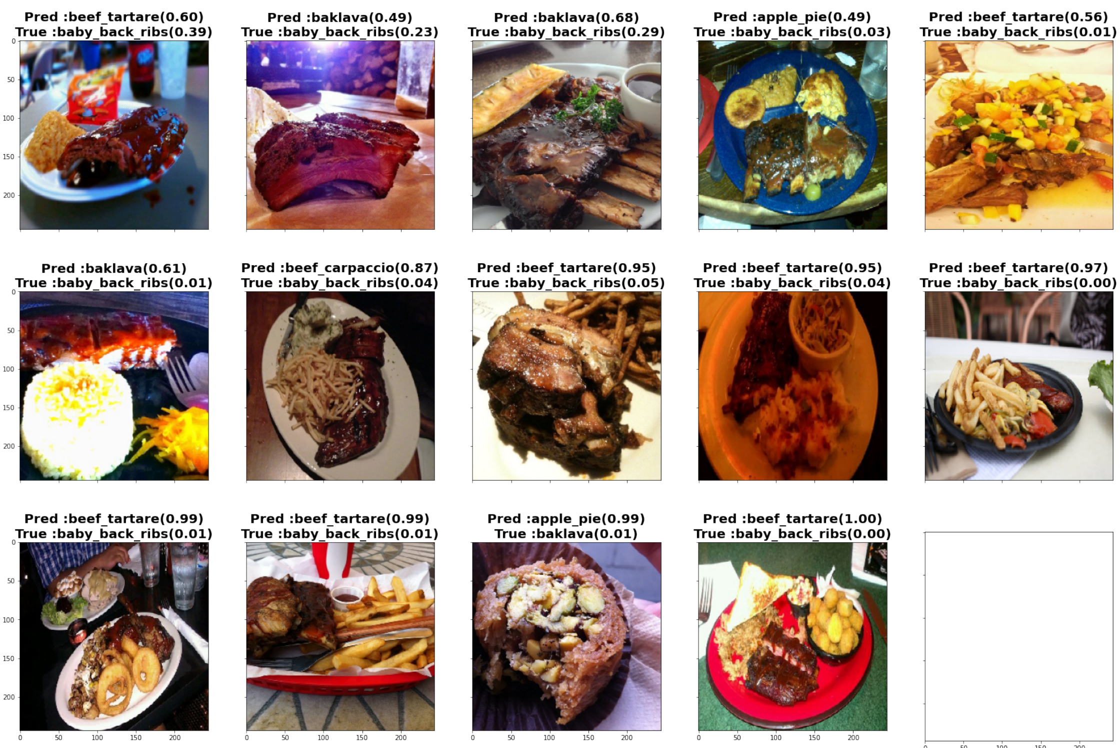
[0]:
```
# find the wrong-est label (largest confidence wrong label)
def show_wrongest_label(x_test,y_test,y_pred_conf):
    y_pred = np.argmax(y_pred_conf,axis=1) # convert predictions to labels
    y_label = np.argmax(y_test,axis=1) # convert answer to labels

    errors = (y_pred - y_label != 0) # find booleans of wrong predictions
    y_pred_errors = y_pred_conf[errors] #the probabilities of the wrong Y_pred␣
 ↪[0.5,0.2,0.3]

    y_pred_classes_errors = y_pred[errors] # the wrong pred label [2]
    y_pred_errors_prob = np.max(y_pred_errors,axis = 1) # Probabilities of the␣
 ↪wrong predicted numbers [0.5]

    y_true_classes_errors = y_label[errors] # the true label [0]
    y_true_errors_prob = np.diagonal(np.take(y_pred_errors,␣
 ↪y_true_classes_errors, axis=1)) # Predicted prob of the true values in the␣
 ↪error set[0.2]

    img_errors = x_test[errors] # image of each errors
```

```
    # Difference between the probability of the predicted label and the true␣
↪label
    delta_pred_true_errors = y_pred_errors_prob - y_true_errors_prob
    # Get index of delta prob errors in ascending order
    sorted_delta_errors = np.argsort(delta_pred_true_errors)
    # The index of top 15 errors
    most_important_errors = sorted_delta_errors[-15:]


    def display_errors(errors_index,img_errors,pred_errors, obs_errors):
        n = 0
        nrows = 3
        ncols = 5
        fig, ax = plt.subplots(nrows,ncols,sharex=True,sharey=True)
        fig.set_figheight(20)
        fig.set_figwidth(30)
        for row in range(nrows):
            for col in range(ncols):
                error = errors_index[n]
                ax[row,col].imshow((img_errors[error]))
                ax[row,col].set_title("Pred :{}({:.2f})\nTrue :{}({:.2f})".format
                                      ␣
↪(ix_to_class[str(pred_errors[error])],y_pred_errors_prob[error],
                                       ␣
↪ix_to_class[str(obs_errors[error])],y_true_errors_prob[error]),
                                       fontweight="bold", size=20)
                n += 1

    display_errors(most_important_errors, img_errors, y_pred_classes_errors,␣
↪y_true_classes_errors)
```

[0]:
```
show_wrongest_label(x_test,y_test,y_pred_conf)
```

```
        ---------------------------------------------------------------------------

        IndexError                                Traceback (most recent call last)

        <ipython-input-103-1886f2901dc0> in <module>()
    ----> 1 show_wrongest_label(x_test,y_test,y_pred_conf)


        <ipython-input-102-4606eeb59b69> in show_wrongest_label(x_test, y_test,␣
↪y_pred_conf)
         39                 n += 1
         40
```

```
---> 41     display_errors(most_important_errors, img_errors,␣
↪y_pred_classes_errors, y_true_classes_errors)


     <ipython-input-102-4606eeb59b69> in display_errors(errors_index,␣
↪img_errors, pred_errors, obs_errors)
     31         for row in range(nrows):
     32             for col in range(ncols):
---> 33                 error = errors_index[n]
     34                 ax[row,col].imshow((img_errors[error]))
     35                 ax[row,col].set_title("Pred :{}({:.2f})\nTrue :{}({:.
↪2f})".format


     IndexError: index 14 is out of bounds for axis 0 with size 14
```

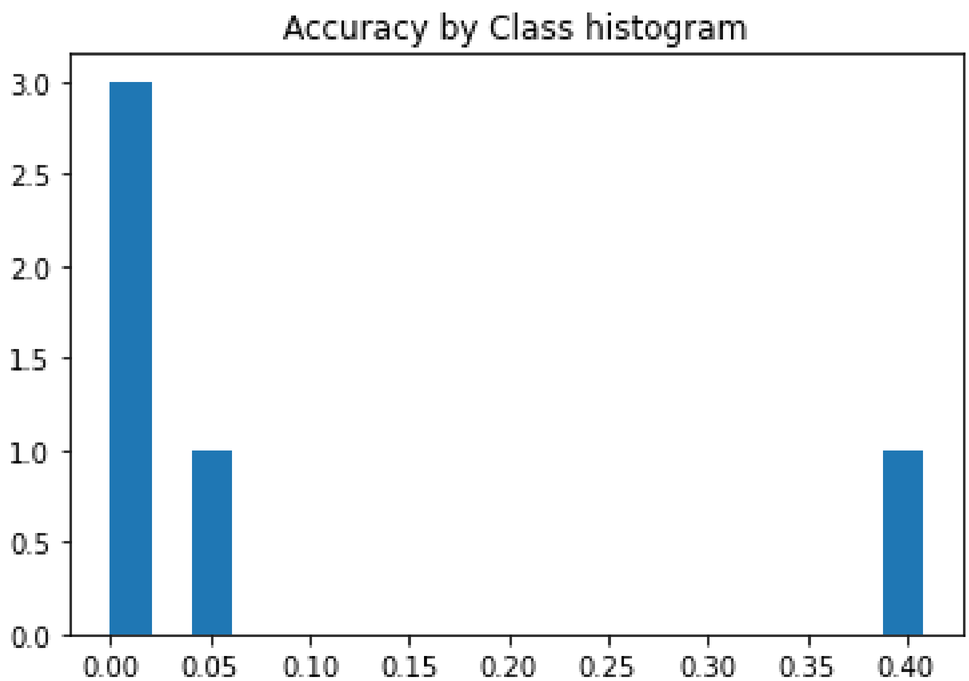


9.0.7 Accuracy by class

[0]:
```python
corrects = collections.defaultdict(int)
incorrects = collections.defaultdict(int)
for (pred, actual_v) in zip(y_pred, y_test):
    actual = np.where(actual_v == 1)[0][0]
```

```
    if pred == actual:
        corrects[actual] += 1
    else:
        incorrects[actual] += 1

class_accuracies = {}
for ix in range(5):
    class_accuracies[ix] = corrects[ix]/250

plt.hist(list(class_accuracies.values()), bins=20)
plt.title('Accuracy by Class histogram')
```

[0]: Text(0.5, 1.0, 'Accuracy by Class histogram')



[0]:
```
sorted_class_accuracies = sorted(class_accuracies.items(), key=lambda x: -x[1])
[(ix_to_class[str(c[0])], c[1]) for c in sorted_class_accuracies]
```

[0]:
```
[('baby_back_ribs', 0.408),
 ('baklava', 0.048),
 ('apple_pie', 0.0),
 ('beef_carpaccio', 0.0),
 ('beef_tartare', 0.0)]
```

10 Predicions

[0]:
```
### Predict funcs

from tensorflow.keras.models import load_model
from tensorflow.keras.applications.inception_v3 import preprocess_input,␣
↪decode_predictions
from skimage.transform import resize

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as img
import collections

import urllib.request

def center_crop(x, center_crop_size, **kwargs):
    centerw, centerh = x.shape[0]//2, x.shape[1]//2
    halfw, halfh = center_crop_size[0]//2, center_crop_size[1]//2
    return x[centerw-halfw:centerw+halfw+1,centerh-halfh:centerh+halfh+1, :]

def predict_10_crop(img, ix, top_n=5, plot=False, preprocess=True, debug=False):
    print('Img shape:', np.array(img).shape)
    flipped_X = np.fliplr(img)
    size = 150
    crops = [
        img[:size,:size, :], # Upper Left
        img[:size, img.shape[1]-size:, :], # Upper Right
        img[img.shape[0]-size:, :size, :], # Lower Left
        img[img.shape[0]-size:, img.shape[1]-size:, :], # Lower Right
        center_crop(img, (size, size)),

        flipped_X[:size,:size, :],
        flipped_X[:size, flipped_X.shape[1]-size:, :],
        flipped_X[flipped_X.shape[0]-size:, :size, :],
        flipped_X[flipped_X.shape[0]-size:, flipped_X.shape[1]-size:, :],
        center_crop(flipped_X, (size, size))
    ]
    if preprocess:
        crops = [preprocess_input(x.astype('float32')) for x in crops]

    if plot:
        fig, ax = plt.subplots(2, 5, figsize=(10, 4))
        ax[0][0].imshow(crops[0])
        ax[0][1].imshow(crops[1])
        ax[0][2].imshow(crops[2])
        ax[0][3].imshow(crops[3])
```

```
        ax[0][4].imshow(crops[4])
        ax[1][0].imshow(crops[5])
        ax[1][1].imshow(crops[6])
        ax[1][2].imshow(crops[7])
        ax[1][3].imshow(crops[8])
        ax[1][4].imshow(crops[9])

    print('Crop. shape:', np.array(crops).shape)
    y_pred = model.predict(np.array(crops))
    preds = np.argmax(y_pred, axis=1)
    top_n_preds= np.argpartition(y_pred, -top_n)[:,-top_n:]
    if debug:
        print('Top-1 Predicted:', preds)
        print('Top-5 Predicted:', top_n_preds)
        print('True Label:', y_test[ix])
    return preds, top_n_preds

def predict(img, model, top_n=5, debug=False):
    resized_img = np.resize(img, (1, 244, 244, 3))
    y_pred = model.predict(np.array(resized_img))
    preds = np.argmax(y_pred, axis=1)
    top_n_preds= np.argpartition(y_pred, -top_n)[:,-top_n:]
    if debug:
        print('Top-1 Predicted:', preds)
        print('Top-5 Predicted:', top_n_preds)
    return preds, top_n_preds

def predict_remote_image(url, model, ix_to_class, debug=False):
    with urllib.request.urlopen(url) as f:
        pic = plt.imread(f, format='jpg')
        preds = predict(np.array(pic), model, debug=debug)[0]
        best_pred = collections.Counter(preds).most_common(1)[0][0]
        print(ix_to_class[best_pred])
        # plt.imshow(pic)

def predict_image(path, model, ix_to_class, debug=False):
    pic = img.imread(path)
    preds = predict(np.array(pic), model, 0, debug=debug)[0]
    best_pred = collections.Counter(preds).most_common(1)[0][0]
    print(ix_to_class[best_pred])
```

The following functions need to be run only when using the **3rd model**, because of transfer learning.

[0]:
```
from tensorflow.keras.models import load_model
from tensorflow.keras.applications.inception_v3 import preprocess_input,␣
 ↪decode_predictions
```

```python
from skimage.transform import resize

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as img
import collections

import urllib.request

def predict(img, model, ix_to_class=None):
  resized_img = np.resize(img, (1, 244, 244, 3))
  feats = pretrained_model.predict(np.array(resized_img))
  preds = model.predict(feats)
  if (ix_to_class != None):
    for pred in preds:
      print(pred)
  preds = np.argmax(preds, axis=1)
  best_pred = collections.Counter(preds).most_common(1)[0][0]
  print(ix_to_class[str(best_pred)])
  return preds

def predict_remote_image(url, model, ix_to_class=None):
    with urllib.request.urlopen(url) as f:
        pic = plt.imread(f, format='jpg')
        #plt.imshow(pic)
        preds = predict(np.array(pic), model, ix_to_class=ix_to_class)
        return preds
```

[0]:
```python
# baklava
predict_remote_image(url='https://www.fifteenspatulas.com/wp-content/uploads/
 ↪2012/03/Baklava-Fifteen-Spatulas-11.jpg', model=model, ix_to_class=ix_to_class)
# apple pie
predict_remote_image(url='https://images-gmi-pmc.edge-generalmills.com/
 ↪75593ed5-420b-4782-8eae-56bdfbc2586b.jpg', model=model,␣
 ↪ix_to_class=ix_to_class)
# beef tartare
predict_remote_image(url='https://mission-food.com/wp-content/uploads/2020/02/
 ↪Steak-Tartare-16.jpg', model=model, ix_to_class=ix_to_class)
```

```
[0.0000000e+00 1.3306978e-34 1.0000000e+00 1.9543608e-11 8.4210285e-33]
baklava
[0.0000000e+00 1.0000000e+00 0.0000000e+00 0.0000000e+00 6.4783588e-34]
baby_back_ribs
[0.0000000e+00 1.0000000e+00 1.0167557e-30 4.1130151e-33 6.0166949e-27]
baby_back_ribs
```

[0]:
```
array([1])
```

[0]:
```
predict_remote_image(url='https://images-gmi-pmc.edge-generalmills.com/
↪75593ed5-420b-4782-8eae-56bdfbc2586b.jpg', model=model,␣
↪ix_to_class=ix_to_class)
```

```
[0.0000000e+00 1.0000000e+00 0.0000000e+00 0.0000000e+00 6.4783588e-34]
baby_back_ribs
```

[0]:
```
array([1])
```

11 Conclusions

In conclusion, the results we got were overall very satisfying, since this problem is very complex and demands a lot of computing power and resources.
We had the best accuracy with the last model: **InceptionV3**, which is based on an already existing model but retrained for our purpose.
Our dataset was also very small and it clearly showed in the data not being as accurate as it could. Bigger datasets would have improved our accuracy but as mentioned before, would be too expensive to train.
In a future stage, we would like to continue working on this and training the model with the full dataset to get a more complex model, capable of understanding a lot of common dishes.