Sejal Sanas

Food Classification and Calories Estimation

Develop a model that can accurately recognize food items from images and estimate their calorie content, enabling users to track their dietary intake and make informed food choices.

Dataset :- https://www.kaggle.com/dansbecker/food-101 (https://www.kaggle.com/dansbecker/food-101)

In [3]:
```
import tensorflow as tf
import matplotlib.image as img
%matplotlib inline
import numpy as np
from collections import defaultdict
import collections
from shutil import copy
from shutil import copytree, rmtree
import tensorflow.keras.backend as K
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
import os
import random
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras import regularizers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.regularizers import l2
from tensorflow import keras
from tensorflow.keras import models
import cv2
```

In [4]:
```
# Check if GPU is enabled
print(tf.__version__)
print(tf.test.gpu_device_name())
```

```
2.13.0
/device:GPU:0

2023-09-08 03:01:14.701430: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355 (https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355)
2023-09-08 03:01:14.809153: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355 (https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355)
2023-09-08 03:01:14.809385: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355 (https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355)
2023-09-08 03:01:14.883917: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355 (https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355)
2023-09-08 03:01:14.884143: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355 (https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355)
2023-09-08 03:01:14.884308: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355 (https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-pci#L344-L355)
2023-09-08 03:01:14.884442: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1639] Created device /device:GPU:0 with 2791 MB memory:  -> device: 0, name: NVIDIA GeForce GTX 1650, pci bus id: 0000:01:00.0, compute capability: 7.5
```

In [4]:
```
# Helper function to download data and extract
def get_data_extract():
  if "food-101" in os.listdir():
    print("Dataset already exists")
  else:
    print("Downloading the data...")
    !wget http://data.vision.ee.ethz.ch/cvl/food-101.tar.gz
    print("Dataset downloaded!")
    print("Extracting data..")
    !tar xzvf food-101.tar.gz
    print("Extraction done!")
```

- **Commented the below cell as the Food-101 dataset is available from Kaggle Datasets and need not be downloaded..**

In [5]:
```
# Download data and extract it to folder

get_data_extract()
```

```
Dataset already exists
```

Understand dataset structure and files

In [6]:
```
# Check the extracted dataset folder
!ls food-101/
```

```
README.txt  images  license_agreement.txt  meta
```

In [7]:
```
os.listdir('food-101/images')
```

Out[7]:
```
['macarons',
 'french_toast',
 'lobster_bisque',
 'prime_rib',
 'pork_chop',
 'guacamole',
 'baby_back_ribs',
 'mussels',
 'beef_carpaccio',
 'poutine',
 'hot_and_sour_soup',
 'seaweed_salad',
 'foie_gras',
 'dumplings',
 'peking_duck',
 'takoyaki',
 'bibimbap',
 'falafel',
 'pulled_pork_sandwich',
 'lobster_roll_sandwich',
 'carrot_cake',
 'beet_salad',
 'panna_cotta',
 'donuts',
 'red_velvet_cake',
 'grilled_cheese_sandwich',
 'cannoli',
 'spring_rolls',
 'shrimp_and_grits',
 'clam_chowder',
 'omelette',
 'fried_calamari',
 'caprese_salad',
 'oysters',
 'scallops',
 'ramen',
 'grilled_salmon',
 'croque_madame',
 'filet_mignon',
 'hamburger',
 'spaghetti_carbonara',
 'miso_soup',
 'bread_pudding',
 'lasagna',
 'crab_cakes',
 'cheesecake',
 'spaghetti_bolognese',
 'cup_cakes',
 'creme_brulee',
 'waffles',
 'fish_and_chips',
 'paella',
 'macaroni_and_cheese',
 'chocolate_mousse',
 'ravioli',
 'chicken_curry',
 'caesar_salad',
 'nachos',
 'tiramisu',
 'frozen_yogurt',
 'ice_cream',
 'risotto',
 'club_sandwich',
 'strawberry_shortcake',
 'steak',
 'churros',
 'garlic_bread',
 'baklava',
 'bruschetta',
 'hummus',
 'chicken_wings',
 'greek_salad',
 'tuna_tartare',
 'chocolate_cake',
 'gyoza',
 'eggs_benedict',
 'deviled_eggs',
 'samosa',
 'sushi',
 'breakfast_burrito',
 'ceviche',
 'beef_tartare',
 'apple_pie',
 '.DS_Store',
```

```
 'huevos_rancheros',
 'beignets',
 'pizza',
 'edamame',
 'french_onion_soup',
 'hot_dog',
 'tacos',
 'chicken_quesadilla',
 'pho',
 'gnocchi',
 'pancakes',
 'fried_rice',
 'cheese_plate',
 'onion_rings',
 'escargots',
 'sashimi',
 'pad_thai',
 'french_fries']
```

In [8]:
```
os.listdir('food-101/meta')
```

Out[8]:
```
['test.txt',
 'train.json',
 'labels.txt',
 'test.json',
 'train.txt',
 'classes.txt']
```

In [9]:
```
!head food-101/meta/train.txt
```

```
apple_pie/1005649
apple_pie/1014775
apple_pie/1026328
apple_pie/1028787
apple_pie/1043283
apple_pie/1050519
apple_pie/1057749
apple_pie/1057810
apple_pie/1072416
apple_pie/1074856
```

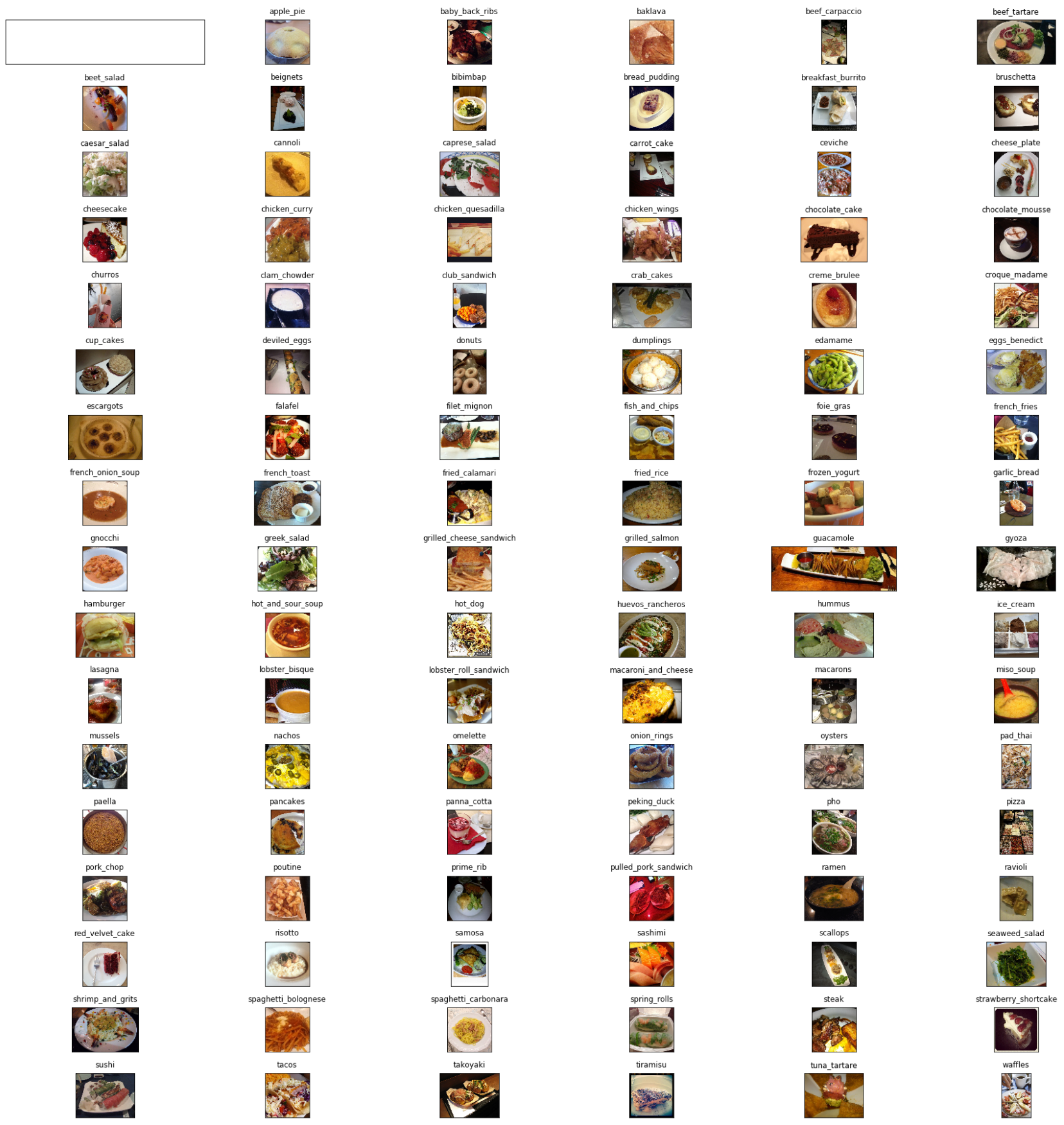
In [10]:
```
!head food-101/meta/classes.txt
```

```
apple_pie
baby_back_ribs
baklava
beef_carpaccio
beef_tartare
beet_salad
beignets
bibimbap
bread_pudding
breakfast_burrito
```

Visualize random image from each of the 101 classes

In [11]:
```
# Visualize the data, showing one image per class from 101 classes
rows = 17
cols = 6
fig, ax = plt.subplots(rows, cols, figsize=(25,25))
fig.suptitle("Showing one random image from each class", y=1.05, fontsize=24) # Adding  y=1.05, fontsi
data_dir = "food-101/images/"
foods_sorted = sorted(os.listdir(data_dir))
food_id = 0
for i in range(rows):
  for j in range(cols):
    try:
      food_selected = foods_sorted[food_id]
      food_id += 1
    except:
      break
    if food_selected == '.DS_Store':
        continue
    food_selected_images = os.listdir(os.path.join(data_dir,food_selected)) # returns the list of all
    food_selected_random = np.random.choice(food_selected_images) # picks one food item from the list
    img = plt.imread(os.path.join(data_dir,food_selected, food_selected_random))
    ax[i][j].imshow(img)
    ax[i][j].set_title(food_selected, pad = 10)

plt.setp(ax, xticks=[],yticks=[])
plt.tight_layout()
# https://matplotlib.org/users/tight_layout_guide.html
```



Split the image data into train and test using train.txt and test.txt

In [12]:
```
# Helper method to split dataset into train and test folders
def prepare_data(filepath, src,dest):
  classes_images = defaultdict(list)
  with open(filepath, 'r') as txt:
      paths = [read.strip() for read in txt.readlines()]
      for p in paths:
        food = p.split('/')
        classes_images[food[0]].append(food[1] + '.jpg')

  for food in classes_images.keys():
    print("\nCopying images into ",food)
    if not os.path.exists(os.path.join(dest,food)):
      os.makedirs(os.path.join(dest,food))
    for i in classes_images[food]:
      copy(os.path.join(src,food,i), os.path.join(dest,food,i))
  print("Copying Done!")
```

In [13]:
```
# Prepare train dataset by copying images from food-101/images to food-101/train using the file train.t
print("Creating train data...")
prepare_data('/food-101/food-101/meta/train.txt', '/food-101/food-101/images', 'train')
```

```
/
Creating train data...

Copying images into  apple_pie

Copying images into  baby_back_ribs

Copying images into  baklava

Copying images into  beef_carpaccio

Copying images into  beef_tartare

Copying images into  beet_salad

Copying images into  beignets

Copying images into  bibimbap

Copying images into  bread_pudding

Copying images into  breakfast_burrito

Copying images into  bruschetta

Copying images into  caesar_salad

Copying images into  cannoli

Copying images into  caprese_salad

Copying images into  carrot_cake

Copying images into  ceviche

Copying images into  cheesecake

Copying images into  cheese_plate

Copying images into  chicken_curry

Copying images into  chicken_quesadilla

Copying images into  chicken_wings

Copying images into  chocolate_cake

Copying images into  chocolate_mousse

Copying images into  churros

Copying images into  clam_chowder

Copying images into  club_sandwich

Copying images into  crab_cakes

Copying images into  creme_brulee

Copying images into  croque_madame

Copying images into  cup_cakes

Copying images into  deviled_eggs

Copying images into  donuts

Copying images into  dumplings

Copying images into  edamame

Copying images into  eggs_benedict

Copying images into  escargots

Copying images into  falafel

Copying images into  filet_mignon

Copying images into  fish_and_chips

Copying images into  foie_gras

Copying images into  french_fries
```

```
Copying images into  french_onion_soup
Copying images into  french_toast
Copying images into  fried_calamari
Copying images into  fried_rice
Copying images into  frozen_yogurt
Copying images into  garlic_bread
Copying images into  gnocchi
Copying images into  greek_salad
Copying images into  grilled_cheese_sandwich
Copying images into  grilled_salmon
Copying images into  guacamole
Copying images into  gyoza
Copying images into  hamburger
Copying images into  hot_and_sour_soup
Copying images into  hot_dog
Copying images into  huevos_rancheros
Copying images into  hummus
Copying images into  ice_cream
Copying images into  lasagna
Copying images into  lobster_bisque
Copying images into  lobster_roll_sandwich
Copying images into  macaroni_and_cheese
Copying images into  macarons
Copying images into  miso_soup
Copying images into  mussels
Copying images into  nachos
Copying images into  omelette
Copying images into  onion_rings
Copying images into  oysters
Copying images into  pad_thai
Copying images into  paella
Copying images into  pancakes
Copying images into  panna_cotta
Copying images into  peking_duck
Copying images into  pho
Copying images into  pizza
Copying images into  pork_chop
Copying images into  poutine
Copying images into  prime_rib
Copying images into  pulled_pork_sandwich
Copying images into  ramen
Copying images into  ravioli
```

```
Copying images into  red_velvet_cake

Copying images into  risotto

Copying images into  samosa

Copying images into  sashimi

Copying images into  scallops

Copying images into  seaweed_salad

Copying images into  shrimp_and_grits

Copying images into  spaghetti_bolognese

Copying images into  spaghetti_carbonara

Copying images into  spring_rolls

Copying images into  steak

Copying images into  strawberry_shortcake

Copying images into  sushi

Copying images into  tacos

Copying images into  takoyaki

Copying images into  tiramisu

Copying images into  tuna_tartare

Copying images into  waffles
Copying Done!
```

In [14]:
```
# Prepare test data by copying images from food-101/images to food-101/test using the file test.txt
print("Creating test data...")
prepare_data('food-101/food-101/meta/test.txt', 'food-101/food-101/images', 'test')
```

```
Creating test data...

Copying images into  apple_pie

Copying images into  baby_back_ribs

Copying images into  baklava

Copying images into  beef_carpaccio

Copying images into  beef_tartare

Copying images into  beet_salad

Copying images into  beignets

Copying images into  bibimbap

Copying images into  bread_pudding
```

In [15]:
```
# Check how many files are in the train folder
print("Total number of samples in train folder")
!find train -type d -or -type f -printf '.' | wc -c
```

```
Total number of samples in train folder
75750
```

In [16]:
```
# Check how many files are in the test folder
print("Total number of samples in test folder")
!find test -type d -or -type f -printf '.' | wc -c
```

```
Total number of samples in test folder
25250
```

In [17]:
```
os.chdir('/')
```

Create a subset of data with few classes(3) - train_mini and test_mini for experimenting

In [18]:
```
# List of all 101 types of foods(sorted alphabetically)
del foods_sorted[0] # remove .DS_Store from the list
```

In [19]:
```
foods_sorted
```

Out[19]:
```
['apple_pie',
 'baby_back_ribs',
 'baklava',
 'beef_carpaccio',
 'beef_tartare',
 'beet_salad',
 'beignets',
 'bibimbap',
 'bread_pudding',
 'breakfast_burrito',
 'bruschetta',
 'caesar_salad',
 'cannoli',
 'caprese_salad',
 'carrot_cake',
 'ceviche',
 'cheese_plate',
 'cheesecake',
 'chicken_curry',
 'chicken_quesadilla',
 'chicken_wings',
 'chocolate_cake',
 'chocolate_mousse',
 'churros',
 'clam_chowder',
 'club_sandwich',
 'crab_cakes',
 'creme_brulee',
 'croque_madame',
 'cup_cakes',
 'deviled_eggs',
 'donuts',
 'dumplings',
 'edamame',
 'eggs_benedict',
 'escargots',
 'falafel',
 'filet_mignon',
 'fish_and_chips',
 'foie_gras',
 'french_fries',
 'french_onion_soup',
 'french_toast',
 'fried_calamari',
 'fried_rice',
 'frozen_yogurt',
 'garlic_bread',
 'gnocchi',
 'greek_salad',
 'grilled_cheese_sandwich',
 'grilled_salmon',
 'guacamole',
 'gyoza',
 'hamburger',
 'hot_and_sour_soup',
 'hot_dog',
 'huevos_rancheros',
 'hummus',
 'ice_cream',
 'lasagna',
 'lobster_bisque',
 'lobster_roll_sandwich',
 'macaroni_and_cheese',
 'macarons',
 'miso_soup',
 'mussels',
 'nachos',
 'omelette',
 'onion_rings',
 'oysters',
 'pad_thai',
 'paella',
 'pancakes',
 'panna_cotta',
 'peking_duck',
 'pho',
 'pizza',
 'pork_chop',
 'poutine',
 'prime_rib',
 'pulled_pork_sandwich',
 'ramen',
 'ravioli',
 'red_velvet_cake',
```

```
'risotto',
'samosa',
'sashimi',
'scallops',
'seaweed_salad',
'shrimp_and_grits',
'spaghetti_bolognese',
'spaghetti_carbonara',
'spring_rolls',
'steak',
'strawberry_shortcake',
'sushi',
'tacos',
'takoyaki',
'tiramisu',
'tuna_tartare',
'waffles']
```

In [20]:
```
# Helper method to create train_mini and test_mini data samples
def dataset_mini(food_list, src, dest):
  if os.path.exists(dest):
    rmtree(dest) # removing dataset_mini(if it already exists) folders so that we will have only the c
  os.makedirs(dest)
  for food_item in food_list :
    print("Copying images into",food_item)
    copytree(os.path.join(src,food_item), os.path.join(dest,food_item))
```

In [21]:
```
# picking 3 food items and generating separate data folders for the same
food_list = ['apple_pie','pizza','omelette']
src_train = 'train'
dest_train = 'train_mini/'
src_test = 'test'
dest_test = 'test_mini/'
```

In [22]:
```
print("Creating train data folder with new classes")
dataset_mini(food_list, src_train, dest_train)
```

```
Creating train data folder with new classes
Copying images into apple_pie
Copying images into pizza
Copying images into omelette
```

In [23]:
```
print("Total number of samples in train folder")

!find /kaggle/working/train_mini -type d -or -type f -printf '.' | wc -c
```

```
Total number of samples in train folder
2250
```

In [24]:
```
print("Creating test data folder with new classes")
dataset_mini(food_list, src_test, dest_test)
```

```
Creating test data folder with new classes
Copying images into apple_pie
Copying images into pizza
Copying images into omelette
```

In [25]:
```
print("Total number of samples in test folder")
!find /kaggle/working/test_mini -type d -or -type f -printf '.' | wc -c
```

```
Total number of samples in test folder
750
```

Fine tune ResNet50 Pretrained model using Food 101 dataset

In [26]:
```
from tensorflow.keras.applications.resnet50 import ResNet50

K.clear_session()
n_classes = 3
img_width, img_height = 224, 224
train_data_dir = 'train_mini'
validation_data_dir = 'test_mini'
nb_train_samples = 2250 #75750
nb_validation_samples = 750 #25250
batch_size = 16

train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')


resnet50 = ResNet50(weights='imagenet', include_top=False)
x = resnet50.output
x = GlobalAveragePooling2D()(x)
x = Dense(128,activation='relu')(x)
x = Dropout(0.2)(x)

predictions = Dense(3,kernel_regularizer=regularizers.l2(0.005), activation='softmax')(x)

model = Model(inputs=resnet50.input, outputs=predictions)
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy', metrics=['accur
checkpointer = ModelCheckpoint(filepath='best_model_3class.hdf5', verbose=1, save_best_only=True)
csv_logger = CSVLogger('history_3class.log')

history = model.fit_generator(train_generator,
                    steps_per_epoch = nb_train_samples // batch_size,
                    validation_data=validation_generator,
                    validation_steps=nb_validation_samples // batch_size,
                    epochs=30,
                    verbose=1,
                    callbacks=[csv_logger, checkpointer])

model.save('model_trained_3class.hdf5')
```

```
Found 2250 images belonging to 3 classes.
Found 750 images belonging to 3 classes.
/opt/conda/lib/python3.6/site-packages/keras_applications/resnet50.py:265: UserWarning: The output sh
ape of `ResNet50(include_top=False)` has been changed since Keras 2.2.0.
  warnings.warn('The output shape of `ResNet50(include_top=False)` '
```

```
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.2/resnet5
0_weights_tf_dim_ordering_tf_kernels_notop.h5 (https://github.com/fchollet/deep-learning-models/relea
ses/download/v0.2/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5)
94658560/94653016 [==============================] - 1s 0us/step
Epoch 1/30
139/140 [============================>.] - ETA: 0s - loss: 1.1083 - acc: 0.4540
Epoch 00001: val_loss improved from inf to 2.34971, saving model to /kaggle/working/best_model_3clas
s.hdf5
140/140 [==============================] - 58s 414ms/step - loss: 1.1083 - acc: 0.4544 - val_loss: 2.
3497 - val_acc: 0.3302
Epoch 2/30
139/140 [============================>.] - ETA: 0s - loss: 0.7917 - acc: 0.6757
Epoch 00002: val_loss improved from 2.34971 to 2.34536, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 40s 283ms/step - loss: 0.7924 - acc: 0.6749 - val_loss: 2.
3454 - val_acc: 0.3302
Epoch 3/30
139/140 [============================>.] - ETA: 0s - loss: 0.6342 - acc: 0.7631
Epoch 00003: val_loss improved from 2.34536 to 1.83978, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 279ms/step - loss: 0.6346 - acc: 0.7630 - val_loss: 1.
8398 - val_acc: 0.2799
Epoch 4/30
139/140 [============================>.] - ETA: 0s - loss: 0.5400 - acc: 0.8117
Epoch 00004: val_loss improved from 1.83978 to 1.80936, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 278ms/step - loss: 0.5394 - acc: 0.8113 - val_loss: 1.
8094 - val_acc: 0.3274
Epoch 5/30
139/140 [============================>.] - ETA: 0s - loss: 0.4759 - acc: 0.8254
Epoch 00005: val_loss improved from 1.80936 to 1.65712, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 279ms/step - loss: 0.4758 - acc: 0.8257 - val_loss: 1.
6571 - val_acc: 0.4049
Epoch 6/30
139/140 [============================>.] - ETA: 0s - loss: 0.4308 - acc: 0.8542
Epoch 00006: val_loss improved from 1.65712 to 0.86345, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 40s 282ms/step - loss: 0.4302 - acc: 0.8548 - val_loss: 0.
8635 - val_acc: 0.6522
Epoch 7/30
139/140 [============================>.] - ETA: 0s - loss: 0.4069 - acc: 0.8621
Epoch 00007: val_loss improved from 0.86345 to 0.44602, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 278ms/step - loss: 0.4061 - acc: 0.8622 - val_loss: 0.
4460 - val_acc: 0.8234
Epoch 8/30
139/140 [============================>.] - ETA: 0s - loss: 0.3646 - acc: 0.8745
Epoch 00008: val_loss improved from 0.44602 to 0.30098, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 278ms/step - loss: 0.3660 - acc: 0.8736 - val_loss: 0.
3010 - val_acc: 0.8954
Epoch 9/30
139/140 [============================>.] - ETA: 0s - loss: 0.3456 - acc: 0.8845
Epoch 00009: val_loss improved from 0.30098 to 0.27120, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 278ms/step - loss: 0.3458 - acc: 0.8849 - val_loss: 0.
2712 - val_acc: 0.9144
Epoch 10/30
139/140 [============================>.] - ETA: 0s - loss: 0.3072 - acc: 0.9005
Epoch 00010: val_loss improved from 0.27120 to 0.25908, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 277ms/step - loss: 0.3089 - acc: 0.8995 - val_loss: 0.
2591 - val_acc: 0.9171
Epoch 11/30
139/140 [============================>.] - ETA: 0s - loss: 0.2933 - acc: 0.9068
Epoch 00011: val_loss improved from 0.25908 to 0.24678, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 280ms/step - loss: 0.2932 - acc: 0.9071 - val_loss: 0.
2468 - val_acc: 0.9226
Epoch 12/30
139/140 [============================>.] - ETA: 0s - loss: 0.2747 - acc: 0.9141
Epoch 00012: val_loss improved from 0.24678 to 0.24273, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 276ms/step - loss: 0.2740 - acc: 0.9143 - val_loss: 0.
2427 - val_acc: 0.9212
Epoch 13/30
139/140 [============================>.] - ETA: 0s - loss: 0.2647 - acc: 0.9138
Epoch 00013: val_loss improved from 0.24273 to 0.23711, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 276ms/step - loss: 0.2652 - acc: 0.9136 - val_loss: 0.
2371 - val_acc: 0.9239
Epoch 14/30
139/140 [============================>.] - ETA: 0s - loss: 0.2438 - acc: 0.9309
```

```
Epoch 00014: val_loss improved from 0.23711 to 0.22959, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 277ms/step - loss: 0.2456 - acc: 0.9305 - val_loss: 0.
2296 - val_acc: 0.9266
Epoch 15/30
139/140 [============================>.] - ETA: 0s - loss: 0.2309 - acc: 0.9291
Epoch 00015: val_loss improved from 0.22959 to 0.22657, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 280ms/step - loss: 0.2299 - acc: 0.9296 - val_loss: 0.
2266 - val_acc: 0.9253
Epoch 16/30
139/140 [============================>.] - ETA: 0s - loss: 0.2226 - acc: 0.9309
Epoch 00016: val_loss did not improve from 0.22657
140/140 [==============================] - 38s 269ms/step - loss: 0.2223 - acc: 0.9310 - val_loss: 0.
2271 - val_acc: 0.9239
Epoch 17/30
139/140 [============================>.] - ETA: 0s - loss: 0.1975 - acc: 0.9402
Epoch 00017: val_loss improved from 0.22657 to 0.22351, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 38s 273ms/step - loss: 0.1978 - acc: 0.9402 - val_loss: 0.
2235 - val_acc: 0.9293
Epoch 18/30
139/140 [============================>.] - ETA: 0s - loss: 0.2024 - acc: 0.9395
Epoch 00018: val_loss improved from 0.22351 to 0.22288, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 38s 271ms/step - loss: 0.2015 - acc: 0.9399 - val_loss: 0.
2229 - val_acc: 0.9266
Epoch 19/30
139/140 [============================>.] - ETA: 0s - loss: 0.1977 - acc: 0.9379
Epoch 00019: val_loss improved from 0.22288 to 0.21527, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 38s 273ms/step - loss: 0.1978 - acc: 0.9375 - val_loss: 0.
2153 - val_acc: 0.9307
Epoch 20/30
139/140 [============================>.] - ETA: 0s - loss: 0.1832 - acc: 0.9451
Epoch 00020: val_loss improved from 0.21527 to 0.20799, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 39s 275ms/step - loss: 0.1835 - acc: 0.9451 - val_loss: 0.
2080 - val_acc: 0.9321
Epoch 21/30
139/140 [============================>.] - ETA: 0s - loss: 0.1737 - acc: 0.9519
Epoch 00021: val_loss did not improve from 0.20799
140/140 [==============================] - 37s 266ms/step - loss: 0.1730 - acc: 0.9522 - val_loss: 0.
2188 - val_acc: 0.9293
Epoch 22/30
139/140 [============================>.] - ETA: 0s - loss: 0.1531 - acc: 0.9593
Epoch 00022: val_loss did not improve from 0.20799
140/140 [==============================] - 38s 270ms/step - loss: 0.1542 - acc: 0.9582 - val_loss: 0.
2108 - val_acc: 0.9307
Epoch 23/30
139/140 [============================>.] - ETA: 0s - loss: 0.1546 - acc: 0.9606
Epoch 00023: val_loss did not improve from 0.20799
140/140 [==============================] - 37s 265ms/step - loss: 0.1547 - acc: 0.9604 - val_loss: 0.
2116 - val_acc: 0.9307
Epoch 24/30
139/140 [============================>.] - ETA: 0s - loss: 0.1432 - acc: 0.9663
Epoch 00024: val_loss did not improve from 0.20799
140/140 [==============================] - 37s 267ms/step - loss: 0.1427 - acc: 0.9665 - val_loss: 0.
2116 - val_acc: 0.9334
Epoch 25/30
139/140 [============================>.] - ETA: 0s - loss: 0.1312 - acc: 0.9640
Epoch 00025: val_loss did not improve from 0.20799
140/140 [==============================] - 38s 272ms/step - loss: 0.1321 - acc: 0.9638 - val_loss: 0.
2125 - val_acc: 0.9321
Epoch 26/30
139/140 [============================>.] - ETA: 0s - loss: 0.1358 - acc: 0.9658
Epoch 00026: val_loss did not improve from 0.20799
140/140 [==============================] - 37s 267ms/step - loss: 0.1368 - acc: 0.9652 - val_loss: 0.
2177 - val_acc: 0.9239
Epoch 27/30
139/140 [============================>.] - ETA: 0s - loss: 0.1371 - acc: 0.9645
Epoch 00027: val_loss did not improve from 0.20799
140/140 [==============================] - 38s 268ms/step - loss: 0.1376 - acc: 0.9643 - val_loss: 0.
2144 - val_acc: 0.9280
Epoch 28/30
139/140 [============================>.] - ETA: 0s - loss: 0.1274 - acc: 0.9692
Epoch 00028: val_loss did not improve from 0.20799
140/140 [==============================] - 38s 268ms/step - loss: 0.1283 - acc: 0.9685 - val_loss: 0.
2173 - val_acc: 0.9239
Epoch 29/30
139/140 [============================>.] - ETA: 0s - loss: 0.1269 - acc: 0.9705
Epoch 00029: val_loss improved from 0.20799 to 0.20611, saving model to /kaggle/working/best_model_3c
lass.hdf5
140/140 [==============================] - 38s 271ms/step - loss: 0.1267 - acc: 0.9703 - val_loss: 0.
2061 - val_acc: 0.9375
```

```
Epoch 30/30
139/140 [============================>.] - ETA: 0s - loss: 0.1220 - acc: 0.9712
Epoch 00030: val_loss did not improve from 0.20611
140/140 [==============================] - 38s 272ms/step - loss: 0.1219 - acc: 0.9714 - val_loss: 0.
2169 - val_acc: 0.9239
```

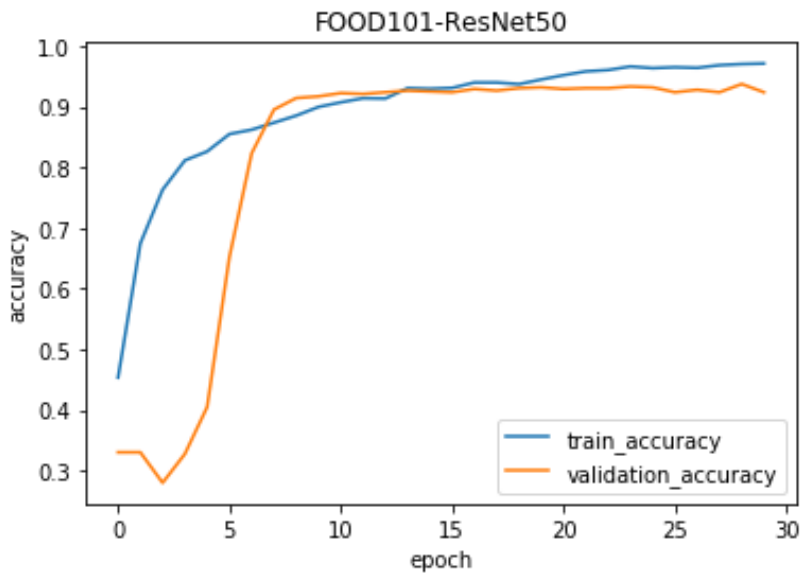
In [27]:
```
class_map_3 = train_generator.class_indices
class_map_3
```

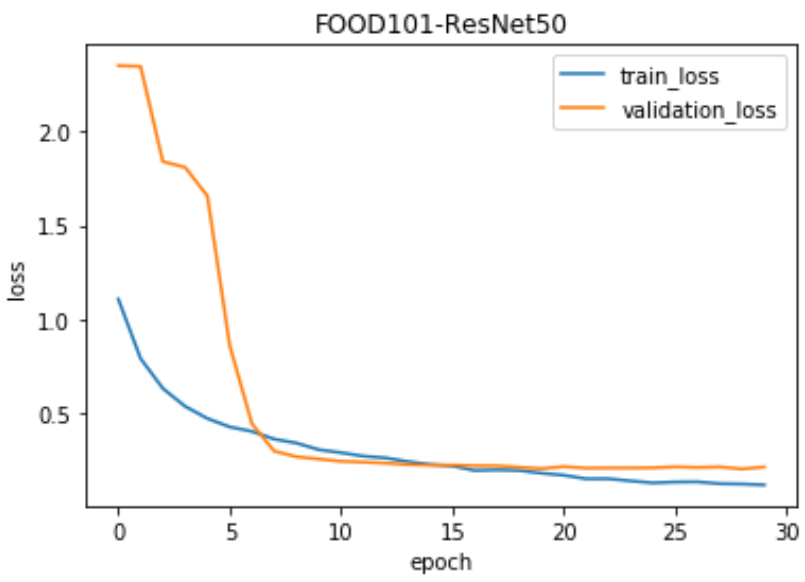
Out[27]:
```
{'apple_pie': 0, 'omelette': 1, 'pizza': 2}
```

Visualize the accuracy and loss plots

In [28]:
```
def plot_accuracy(history,title):
    plt.title(title)
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train_accuracy', 'validation_accuracy'], loc='best')
    plt.show()
def plot_loss(history,title):
    plt.title(title)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train_loss', 'validation_loss'], loc='best')
    plt.show()
```

In [29]:
```
plot_accuracy(history,'FOOD101-ResNet50')
plot_loss(history,'FOOD101-ResNet50')
```





Predicting classes for new images from internet using the best trained model

In [30]:
```
%%time
# Loading the best saved model to make predictions
K.clear_session()
model_best = load_model('/kaggle/working/best_model_3class.hdf5',compile = False)
```

```
CPU times: user 6.64 s, sys: 194 ms, total: 6.84 s
Wall time: 6.79 s
```

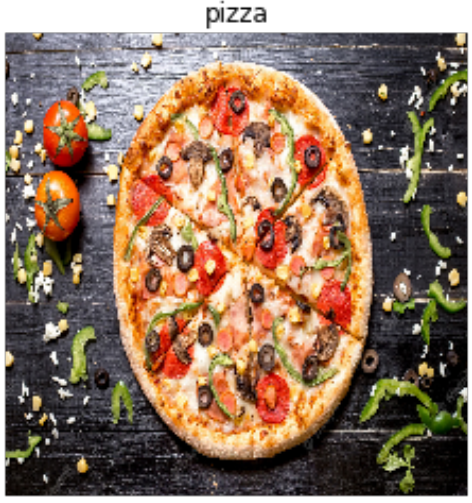
In [31]:
```
def predict_class(model, images, show = True):
  for img in images:
    img = image.load_img(img, target_size=(224, 224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img /= 255.

    pred = model.predict(img)
    index = np.argmax(pred)
    food_list.sort()
    pred_value = food_list[index]
    if show:
        plt.imshow(img[0])
        plt.axis('off')
        plt.title(pred_value)
        plt.show()
```

In [35]:
```
# Make a list of downloaded images and test the trained model
images = []
images.append('applepie.jpg')
images.append('pizza.jpg')
images.append('omelette.jpg')
predict_class(model_best, images, True)
```

apple_pie

pizza



omelette

Fine tune ResNet50 model with 11 classes of data

In [36]:
```
# Helper function to select n random food classes
def pick_n_random_classes(n):
  food_list = []
  random_food_indices = random.sample(range(len(foods_sorted)),n) # We are picking n random food class
  for i in random_food_indices:
    food_list.append(foods_sorted[i])
  food_list.sort()
  return food_list
```

In [37]:
```
# Lets try with more classes than just 3. Also, this time lets randomly pick the food classes
n = 11
food_list = pick_n_random_classes(n)
food_list = ['apple_pie', 'beef_carpaccio', 'bibimbap', 'cup_cakes', 'foie_gras', 'french_fries', 'gar
print("These are the randomly picked food classes we will be training the model on...\n", food_list)
```

```
These are the randomly picked food classes we will be training the model on...
 ['apple_pie', 'beef_carpaccio', 'bibimbap', 'cup_cakes', 'foie_gras', 'french_fries', 'garlic_bread'
, 'pizza', 'spring_rolls', 'spaghetti_carbonara', 'strawberry_shortcake']
```

In [38]:
```
# Create the new data subset of n classes
print("Creating training data folder with new classes...")
dataset_mini(food_list, src_train, dest_train)
```

```
Creating training data folder with new classes...
Copying images into apple_pie
Copying images into beef_carpaccio
Copying images into bibimbap
Copying images into cup_cakes
Copying images into foie_gras
Copying images into french_fries
Copying images into garlic_bread
Copying images into pizza
Copying images into spring_rolls
Copying images into spaghetti_carbonara
Copying images into strawberry_shortcake
```

In [39]:
```
print("Total number of samples in train folder")
!find train_mini/ -type d -or -type f -printf '.' | wc -c
```

```
Total number of samples in train folder
8250
```

In [40]:
```
print("Creating test data folder with new classes")
dataset_mini(food_list, src_test, dest_test)
```

```
Creating test data folder with new classes
Copying images into apple_pie
Copying images into beef_carpaccio
Copying images into bibimbap
Copying images into cup_cakes
Copying images into foie_gras
Copying images into french_fries
Copying images into garlic_bread
Copying images into pizza
Copying images into spring_rolls
Copying images into spaghetti_carbonara
Copying images into strawberry_shortcake
```

In [41]:
```
print("Total number of samples in test folder")
!find test_mini/ -type d -or -type f -printf '.' | wc -c
```

```
Total number of samples in test folder
2750
```

In [42]:
```
# Let's use a pretrained Inceptionv3 model on subset of data with 11 food classes
K.clear_session()

n_classes = n
img_width, img_height = 224, 224
train_data_dir = 'train_mini'
validation_data_dir = 'test_mini'
nb_train_samples = 8250 #75750
nb_validation_samples = 2750 #25250
batch_size = 16

train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')


resnet50 = ResNet50(weights='imagenet', include_top=False)
x = resnet50.output
x = GlobalAveragePooling2D()(x)
x = Dense(128,activation='relu')(x)
x = Dropout(0.2)(x)

predictions = Dense(n,kernel_regularizer=regularizers.l2(0.005), activation='softmax')(x)

model = Model(inputs=resnet50.input, outputs=predictions)
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy', metrics=['accur
checkpointer = ModelCheckpoint(filepath='best_model_11class.hdf5', verbose=1, save_best_only=True)
csv_logger = CSVLogger('history_11class.log')

history_11class = model.fit_generator(train_generator,
                    steps_per_epoch = nb_train_samples // batch_size,
                    validation_data=validation_generator,
                    validation_steps=nb_validation_samples // batch_size,
                    epochs=30,
                    verbose=1,
                    callbacks=[csv_logger, checkpointer])

model.save('model_trained_11class.hdf5')
```

```
Found 8250 images belonging to 11 classes.
Found 2750 images belonging to 11 classes.
/opt/conda/lib/python3.6/site-packages/keras_applications/resnet50.py:265: UserWarning: The output sh
ape of `ResNet50(include_top=False)` has been changed since Keras 2.2.0.
  warnings.warn('The output shape of `ResNet50(include_top=False)` '
```

```
Epoch 1/30
514/515 [============================>.] - ETA: 0s - loss: 2.0506 - acc: 0.3578
Epoch 00001: val_loss improved from inf to 2.63834, saving model to /kaggle/working/best_model_11clas
s.hdf5
515/515 [==============================] - 163s 316ms/step - loss: 2.0494 - acc: 0.3581 - val_loss: 2
.6383 - val_acc: 0.1758
Epoch 2/30
514/515 [============================>.] - ETA: 0s - loss: 1.1769 - acc: 0.6723
Epoch 00002: val_loss improved from 2.63834 to 0.78432, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 148s 287ms/step - loss: 1.1759 - acc: 0.6725 - val_loss: 0
.7843 - val_acc: 0.7855
Epoch 3/30
514/515 [============================>.] - ETA: 0s - loss: 0.9007 - acc: 0.7499
Epoch 00003: val_loss improved from 0.78432 to 0.57513, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 148s 287ms/step - loss: 0.9000 - acc: 0.7501 - val_loss: 0
.5751 - val_acc: 0.8465
Epoch 4/30
514/515 [============================>.] - ETA: 0s - loss: 0.7720 - acc: 0.7880
Epoch 00004: val_loss improved from 0.57513 to 0.53192, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 148s 287ms/step - loss: 0.7715 - acc: 0.7880 - val_loss: 0
.5319 - val_acc: 0.8571
Epoch 5/30
514/515 [============================>.] - ETA: 0s - loss: 0.6954 - acc: 0.8086
Epoch 00005: val_loss improved from 0.53192 to 0.47222, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 148s 287ms/step - loss: 0.6949 - acc: 0.8088 - val_loss: 0
.4722 - val_acc: 0.8761
Epoch 6/30
514/515 [============================>.] - ETA: 0s - loss: 0.6268 - acc: 0.8333
Epoch 00006: val_loss improved from 0.47222 to 0.44181, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 147s 286ms/step - loss: 0.6268 - acc: 0.8333 - val_loss: 0
.4418 - val_acc: 0.8852
Epoch 7/30
514/515 [============================>.] - ETA: 0s - loss: 0.5830 - acc: 0.8498
Epoch 00007: val_loss improved from 0.44181 to 0.42070, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 149s 289ms/step - loss: 0.5829 - acc: 0.8499 - val_loss: 0
.4207 - val_acc: 0.8907
Epoch 8/30
514/515 [============================>.] - ETA: 0s - loss: 0.5408 - acc: 0.8591
Epoch 00008: val_loss improved from 0.42070 to 0.41421, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 147s 286ms/step - loss: 0.5403 - acc: 0.8592 - val_loss: 0
.4142 - val_acc: 0.8955
Epoch 9/30
514/515 [============================>.] - ETA: 0s - loss: 0.5019 - acc: 0.8699
Epoch 00009: val_loss improved from 0.41421 to 0.40790, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 147s 285ms/step - loss: 0.5016 - acc: 0.8701 - val_loss: 0
.4079 - val_acc: 0.8984
Epoch 10/30
514/515 [============================>.] - ETA: 0s - loss: 0.4652 - acc: 0.8858
Epoch 00010: val_loss improved from 0.40790 to 0.39773, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 147s 285ms/step - loss: 0.4653 - acc: 0.8858 - val_loss: 0
.3977 - val_acc: 0.9002
Epoch 11/30
514/515 [============================>.] - ETA: 0s - loss: 0.4437 - acc: 0.8917
Epoch 00011: val_loss improved from 0.39773 to 0.37855, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 148s 287ms/step - loss: 0.4436 - acc: 0.8916 - val_loss: 0
.3785 - val_acc: 0.9050
Epoch 12/30
514/515 [============================>.] - ETA: 0s - loss: 0.4209 - acc: 0.8974
Epoch 00012: val_loss did not improve from 0.37855
515/515 [==============================] - 147s 286ms/step - loss: 0.4220 - acc: 0.8971 - val_loss: 0
.3810 - val_acc: 0.9079
Epoch 13/30
514/515 [============================>.] - ETA: 0s - loss: 0.3847 - acc: 0.9103
Epoch 00013: val_loss improved from 0.37855 to 0.36627, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 148s 286ms/step - loss: 0.3850 - acc: 0.9102 - val_loss: 0
.3663 - val_acc: 0.9097
Epoch 14/30
514/515 [============================>.] - ETA: 0s - loss: 0.3682 - acc: 0.9175
Epoch 00014: val_loss improved from 0.36627 to 0.36368, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 149s 289ms/step - loss: 0.3680 - acc: 0.9176 - val_loss: 0
.3637 - val_acc: 0.9108
Epoch 15/30
```

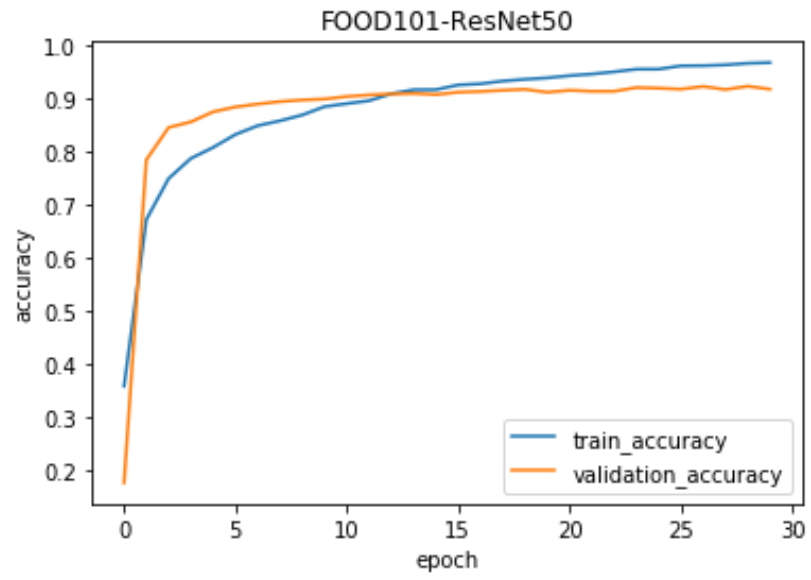
```
514/515 [============================>.] - ETA: 0s - loss: 0.3643 - acc: 0.9177
Epoch 00015: val_loss did not improve from 0.36368
515/515 [==============================] - 146s 283ms/step - loss: 0.3638 - acc: 0.9179 - val_loss: 0
.3684 - val_acc: 0.9086
Epoch 16/30
514/515 [============================>.] - ETA: 0s - loss: 0.3384 - acc: 0.9265
Epoch 00016: val_loss did not improve from 0.36368
515/515 [==============================] - 145s 282ms/step - loss: 0.3383 - acc: 0.9265 - val_loss: 0
.3655 - val_acc: 0.9130
Epoch 17/30
514/515 [============================>.] - ETA: 0s - loss: 0.3221 - acc: 0.9290
Epoch 00017: val_loss improved from 0.36368 to 0.35765, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 146s 284ms/step - loss: 0.3219 - acc: 0.9290 - val_loss: 0
.3577 - val_acc: 0.9145
Epoch 18/30
514/515 [============================>.] - ETA: 0s - loss: 0.3106 - acc: 0.9339
Epoch 00018: val_loss improved from 0.35765 to 0.35532, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 146s 283ms/step - loss: 0.3103 - acc: 0.9341 - val_loss: 0
.3553 - val_acc: 0.9167
Epoch 19/30
514/515 [============================>.] - ETA: 0s - loss: 0.2943 - acc: 0.9374
Epoch 00019: val_loss improved from 0.35532 to 0.34781, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 146s 283ms/step - loss: 0.2941 - acc: 0.9375 - val_loss: 0
.3478 - val_acc: 0.9185
Epoch 20/30
514/515 [============================>.] - ETA: 0s - loss: 0.2866 - acc: 0.9400
Epoch 00020: val_loss did not improve from 0.34781
515/515 [==============================] - 145s 282ms/step - loss: 0.2865 - acc: 0.9400 - val_loss: 0
.3514 - val_acc: 0.9130
Epoch 21/30
514/515 [============================>.] - ETA: 0s - loss: 0.2649 - acc: 0.9444
Epoch 00021: val_loss did not improve from 0.34781
515/515 [==============================] - 145s 281ms/step - loss: 0.2655 - acc: 0.9443 - val_loss: 0
.3564 - val_acc: 0.9167
Epoch 22/30
514/515 [============================>.] - ETA: 0s - loss: 0.2611 - acc: 0.9476
Epoch 00022: val_loss did not improve from 0.34781
515/515 [==============================] - 147s 286ms/step - loss: 0.2616 - acc: 0.9473 - val_loss: 0
.3526 - val_acc: 0.9148
Epoch 23/30
514/515 [============================>.] - ETA: 0s - loss: 0.2487 - acc: 0.9515
Epoch 00023: val_loss did not improve from 0.34781
515/515 [==============================] - 145s 282ms/step - loss: 0.2488 - acc: 0.9515 - val_loss: 0
.3517 - val_acc: 0.9148
Epoch 24/30
514/515 [============================>.] - ETA: 0s - loss: 0.2323 - acc: 0.9566
Epoch 00024: val_loss improved from 0.34781 to 0.34566, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 147s 286ms/step - loss: 0.2325 - acc: 0.9566 - val_loss: 0
.3457 - val_acc: 0.9218
Epoch 25/30
514/515 [============================>.] - ETA: 0s - loss: 0.2314 - acc: 0.9570
Epoch 00025: val_loss improved from 0.34566 to 0.34565, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 147s 286ms/step - loss: 0.2320 - acc: 0.9567 - val_loss: 0
.3456 - val_acc: 0.9207
Epoch 26/30
514/515 [============================>.] - ETA: 0s - loss: 0.2168 - acc: 0.9629
Epoch 00026: val_loss improved from 0.34565 to 0.34340, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 147s 285ms/step - loss: 0.2171 - acc: 0.9629 - val_loss: 0
.3434 - val_acc: 0.9189
Epoch 27/30
514/515 [============================>.] - ETA: 0s - loss: 0.2128 - acc: 0.9630
Epoch 00027: val_loss improved from 0.34340 to 0.33705, saving model to /kaggle/working/best_model_11
class.hdf5
515/515 [==============================] - 146s 284ms/step - loss: 0.2126 - acc: 0.9631 - val_loss: 0
.3370 - val_acc: 0.9240
Epoch 28/30
514/515 [============================>.] - ETA: 0s - loss: 0.2067 - acc: 0.9647
Epoch 00028: val_loss did not improve from 0.33705
515/515 [==============================] - 147s 285ms/step - loss: 0.2068 - acc: 0.9646 - val_loss: 0
.3509 - val_acc: 0.9181
Epoch 29/30
514/515 [============================>.] - ETA: 0s - loss: 0.1957 - acc: 0.9675
Epoch 00029: val_loss did not improve from 0.33705
515/515 [==============================] - 146s 284ms/step - loss: 0.1956 - acc: 0.9676 - val_loss: 0
.3393 - val_acc: 0.9243
Epoch 30/30
514/515 [============================>.] - ETA: 0s - loss: 0.1916 - acc: 0.9690
Epoch 00030: val_loss did not improve from 0.33705
515/515 [==============================] - 146s 283ms/step - loss: 0.1918 - acc: 0.9689 - val_loss: 0
```

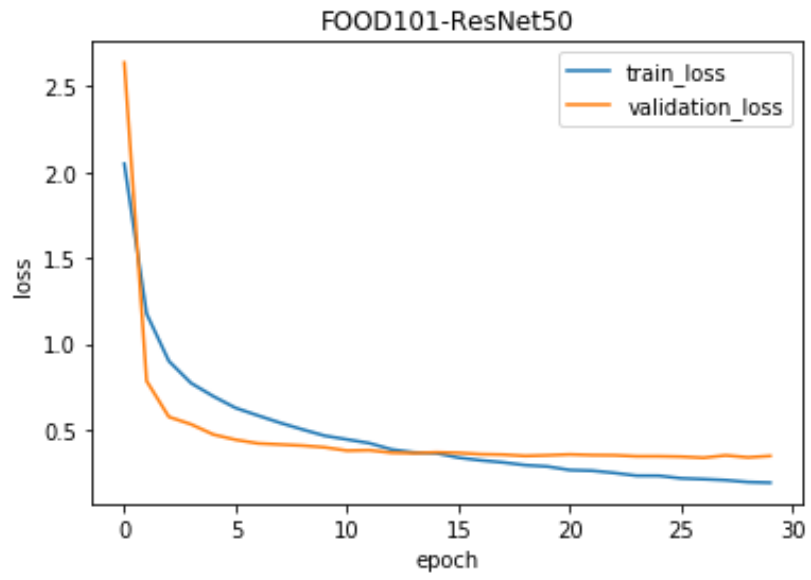
```
.3475 - val_acc: 0.9189
```

In [43]:
```
class_map_11 = train_generator.class_indices
class_map_11
```

Out[43]:
```
{'apple_pie': 0,
 'beef_carpaccio': 1,
 'bibimbap': 2,
 'cup_cakes': 3,
 'foie_gras': 4,
 'french_fries': 5,
 'garlic_bread': 6,
 'pizza': 7,
 'spaghetti_carbonara': 8,
 'spring_rolls': 9,
 'strawberry_shortcake': 10}
```

In [44]:
```
plot_accuracy(history_11class,'FOOD101-ResNet50')
plot_loss(history_11class,'FOOD101-ResNet50')
```





In [45]:
```
%%time
# Loading the best saved model to make predictions
K.clear_session()
model_best = load_model('/kaggle/working/best_model_11class.hdf5',compile = False)
```

```
CPU times: user 6.88 s, sys: 165 ms, total: 7.05 s
Wall time: 7.05 s
```

In [47]:
```
# Make a list of downloaded images and test the trained model
images = []
images.append('cupcakes.jpg')
# images.append('pizza.jpg')
images.append('springrolls.jpg')
images.append('garlicbread.jpg')
predict_class(model_best, images, True)
```

cup_cakes

spring_rolls

garlic_bread

In [48]:
```
os.chdir("AbdulQadeer/Dataset/")
```

In [49]:
```
tf.keras.utils.plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```