project

June 12, 2020

[2]:
```
!pip install tf-nightly
import os
import random
import time
import tensorflow as tf
from glob import glob

from tensorflow.keras import layers
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.resnet import ResNet50
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Flatten, MaxPooling2D,␣
 ↪Conv2D, BatchNormalization, GlobalAveragePooling2D
from keras.callbacks import ModelCheckpoint, EarlyStopping

from sklearn.model_selection import train_test_split
from sklearn.datasets import load_files
from keras.utils import np_utils
from sklearn.utils import shuffle
from sklearn.metrics import log_loss

import numpy as np
import pandas
import matplotlib.pyplot as plt
import cv2

import seaborn as sns
```

```
Collecting tf-nightly
  Downloading https://files.pythonhosted.org/packages/64/d1/8592937ebedcc5
9a248688a5ad5654ba1d339e61737f12cf4b6c1dfe7537/tf_nightly-2.3.0.dev20200612-cp36
-cp36m-manylinux2010_x86_64.whl (350.0MB)
     |                | 350.0MB 50kB/s
Requirement already satisfied: h5py<2.11.0,>=2.10.0 in
```

```
/usr/local/lib/python3.6/dist-packages (from tf-nightly) (2.10.0)
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.6/dist-
packages (from tf-nightly) (1.12.1)
Requirement already satisfied: gast==0.3.3 in /usr/local/lib/python3.6/dist-
packages (from tf-nightly) (0.3.3)
Requirement already satisfied: numpy<2.0,>=1.16.0 in
/usr/local/lib/python3.6/dist-packages (from tf-nightly) (1.18.5)
Requirement already satisfied: astunparse==1.6.3 in
/usr/local/lib/python3.6/dist-packages (from tf-nightly) (1.6.3)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.6/dist-packages (from tf-nightly) (3.2.1)
Collecting tb-nightly<2.4.0a0,>=2.3.0a0
  Downloading https://files.pythonhosted.org/packages/51/01/04bbab951759ef
4006c89b004272e7e82341457325d50695b6baf349f48c/tb_nightly-2.3.0a20200612-py3-non
e-any.whl (3.0MB)
     |                  | 3.0MB 9.9MB/s
Requirement already satisfied: absl-py>=0.7.0 in
/usr/local/lib/python3.6/dist-packages (from tf-nightly) (0.9.0)
Requirement already satisfied: keras-preprocessing<1.2,>=1.1.1 in
/usr/local/lib/python3.6/dist-packages (from tf-nightly) (1.1.2)
Requirement already satisfied: scipy==1.4.1 in /usr/local/lib/python3.6/dist-
packages (from tf-nightly) (1.4.1)
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.6/dist-
packages (from tf-nightly) (1.29.0)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.6/dist-
packages (from tf-nightly) (0.34.2)
Requirement already satisfied: google-pasta>=0.1.8 in
/usr/local/lib/python3.6/dist-packages (from tf-nightly) (0.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.6/dist-
packages (from tf-nightly) (1.12.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.6/dist-packages (from tf-nightly) (1.1.0)
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.6/dist-
packages (from tf-nightly) (3.10.0)
Collecting tf-estimator-nightly
  Downloading https://files.pythonhosted.org/packages/62/73/c1d64128cb545c
25aa45e543365838852f1fa9d4dad0f3c26862c43ecf70/tf_estimator_nightly-2.3.0.dev202
0061201-py2.py3-none-any.whl (459kB)
     |                  | 460kB 50.8MB/s
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.6/dist-packages (from tb-nightly<2.4.0a0,>=2.3.0a0->tf-
nightly) (0.4.1)
Requirement already satisfied: setuptools>=41.0.0 in
/usr/local/lib/python3.6/dist-packages (from tb-nightly<2.4.0a0,>=2.3.0a0->tf-
nightly) (47.1.1)
Requirement already satisfied: werkzeug>=0.11.15 in
/usr/local/lib/python3.6/dist-packages (from tb-nightly<2.4.0a0,>=2.3.0a0->tf-
nightly) (1.0.1)
```

```
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.6/dist-packages (from tb-nightly<2.4.0a0,>=2.3.0a0->tf-
nightly) (1.6.0.post3)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.6/dist-packages (from tb-nightly<2.4.0a0,>=2.3.0a0->tf-
nightly) (2.23.0)
Requirement already satisfied: google-auth<2,>=1.6.3 in
/usr/local/lib/python3.6/dist-packages (from tb-nightly<2.4.0a0,>=2.3.0a0->tf-
nightly) (1.7.2)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-
packages (from tb-nightly<2.4.0a0,>=2.3.0a0->tf-nightly) (3.2.2)
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->tb-nightly<2.4.0a0,>=2.3.0a0->tf-nightly) (1.3.0)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.6/dist-packages (from requests<3,>=2.21.0->tb-
nightly<2.4.0a0,>=2.3.0a0->tf-nightly) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-
packages (from requests<3,>=2.21.0->tb-nightly<2.4.0a0,>=2.3.0a0->tf-nightly)
(2.9)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.6/dist-packages (from requests<3,>=2.21.0->tb-
nightly<2.4.0a0,>=2.3.0a0->tf-nightly) (2020.4.5.1)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.6/dist-packages (from requests<3,>=2.21.0->tb-
nightly<2.4.0a0,>=2.3.0a0->tf-nightly) (1.24.3)
Requirement already satisfied: cachetools<3.2,>=2.0.0 in
/usr/local/lib/python3.6/dist-packages (from google-auth<2,>=1.6.3->tb-
nightly<2.4.0a0,>=2.3.0a0->tf-nightly) (3.1.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.6/dist-packages (from google-auth<2,>=1.6.3->tb-
nightly<2.4.0a0,>=2.3.0a0->tf-nightly) (0.2.8)
Requirement already satisfied: rsa<4.1,>=3.1.4 in /usr/local/lib/python3.6/dist-
packages (from google-auth<2,>=1.6.3->tb-nightly<2.4.0a0,>=2.3.0a0->tf-nightly)
(4.0)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in
/usr/local/lib/python3.6/dist-packages (from markdown>=2.6.8->tb-
nightly<2.4.0a0,>=2.3.0a0->tf-nightly) (1.6.0)
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->tb-
nightly<2.4.0a0,>=2.3.0a0->tf-nightly) (3.1.0)
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->tb-nightly<2.4.0a0,>=2.3.0a0->tf-nightly) (0.4.8)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-
packages (from importlib-metadata; python_version < "3.8"->markdown>=2.6.8->tb-
nightly<2.4.0a0,>=2.3.0a0->tf-nightly) (3.1.0)
Installing collected packages: tb-nightly, tf-estimator-nightly, tf-nightly
```

```
Successfully installed tb-nightly-2.3.0a20200612 tf-estimator-
nightly-2.3.0.dev2020061201 tf-nightly-2.3.0.dev20200612

Using TensorFlow backend.
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

[3]:
```
print(tf.__version__)
```

```
2.3.0-dev20200612
```

[4]:
```
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
```

Import and analyze the dataset

[5]:
```
!pip install kaggle
os.environ['KAGGLE_CONFIG_DIR'] = "/content/gdrive/My Drive/Kaggle"
os.environ['KAGGLE_USERNAME'] = "zhiyuanliu"
os.environ['KAGGLE_KEY'] = "8ef7c299850b6140aa71bd6860a1299d"
import kaggle

kaggle.api.authenticate()
kaggle.api.competition_download_files('state-farm-distracted-driver-detection',␣
 ↪path = 'data/DistractedDriver')
!unzip -q data/DistractedDriver/state-farm-distracted-driver-detection.zip ␣
 ↪#unzips kaggle data into content/data
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.6/dist-packages
(1.5.6)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in
/usr/local/lib/python3.6/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.6/dist-
packages (from kaggle) (4.0.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/dist-
packages (from kaggle) (1.12.0)
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-
packages (from kaggle) (2.23.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-
packages (from kaggle) (2.8.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages
(from kaggle) (4.41.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages
(from kaggle) (2020.4.5.1)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.6/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-
packages (from requests->kaggle) (2.9)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.6/dist-packages (from requests->kaggle) (3.0.4)
```

[6]:
```
img_list = pandas.read_csv('driver_imgs_list.csv')
img_list.head(10)
```

[6]:

	subject	classname	img
0	p002	c0	img_44733.jpg
1	p002	c0	img_72999.jpg
2	p002	c0	img_25094.jpg
3	p002	c0	img_69092.jpg
4	p002	c0	img_92629.jpg
5	p002	c0	img_3370.jpg
6	p002	c0	img_67639.jpg
7	p002	c0	img_58560.jpg
8	p002	c0	img_35779.jpg
9	p002	c0	img_10012.jpg

[7]:
```
unique_drivers = img_list.groupby('subject')
unique_drivers = unique_drivers.groups.keys()
print(unique_drivers)
print(len(unique_drivers))
```

```
dict_keys(['p002', 'p012', 'p014', 'p015', 'p016', 'p021', 'p022', 'p024',
'p026', 'p035', 'p039', 'p041', 'p042', 'p045', 'p047', 'p049', 'p050', 'p051',
'p052', 'p056', 'p061', 'p064', 'p066', 'p072', 'p075', 'p081'])
26
```

[8]:
```
img_rows = 64
img_cols = 64
color_type = 1

train_images = []
train_labels = []
for cat in range(10):
```

```
    images_paths = glob('imgs/train/c' + str(cat) + '/*.jpg')
    for image_path in images_paths:
        img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (img_rows, img_cols))
        train_images.append(img)
        train_labels.append(cat)
train_labels = np_utils.to_categorical(train_labels, 10)
x_train, x_test, y_train, y_test = train_test_split(train_images, train_labels,␣
 ↪test_size=0.2, random_state=42)
x_train = np.array(x_train, dtype=np.uint8).
 ↪reshape(-1,img_rows,img_cols,color_type)
x_test = np.array(x_test, dtype=np.uint8).
 ↪reshape(-1,img_rows,img_cols,color_type)
print('train shape:', x_train.shape)
```

```
train shape: (17939, 64, 64, 1)
```

[0]:

[9]:
```
test_sample_size = 500

def load_test(size=200000, img_rows=64, img_cols=64):
    images_paths = sorted(glob('imgs/test/*.jpg'))
    X_test = []
    X_test_id = []
    total = 0
    files_size = len(images_paths)
    for image_path in images_paths:
        if total >= size or total >= files_size:
            break
        file_base = os.path.basename(image_path)
        img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (img_rows, img_cols))
        X_test.append(img)
        X_test_id.append(file_base)
        total += 1
    return X_test, X_test_id

test_files, test_targets = load_test(test_sample_size, img_rows, img_cols)
test_files = np.array(test_files, dtype=np.uint8)
test_files = test_files.reshape(-1,img_rows,img_cols,color_type)

print('Test shape:', test_files.shape)
print(test_files.shape[0], 'Test samples')
```

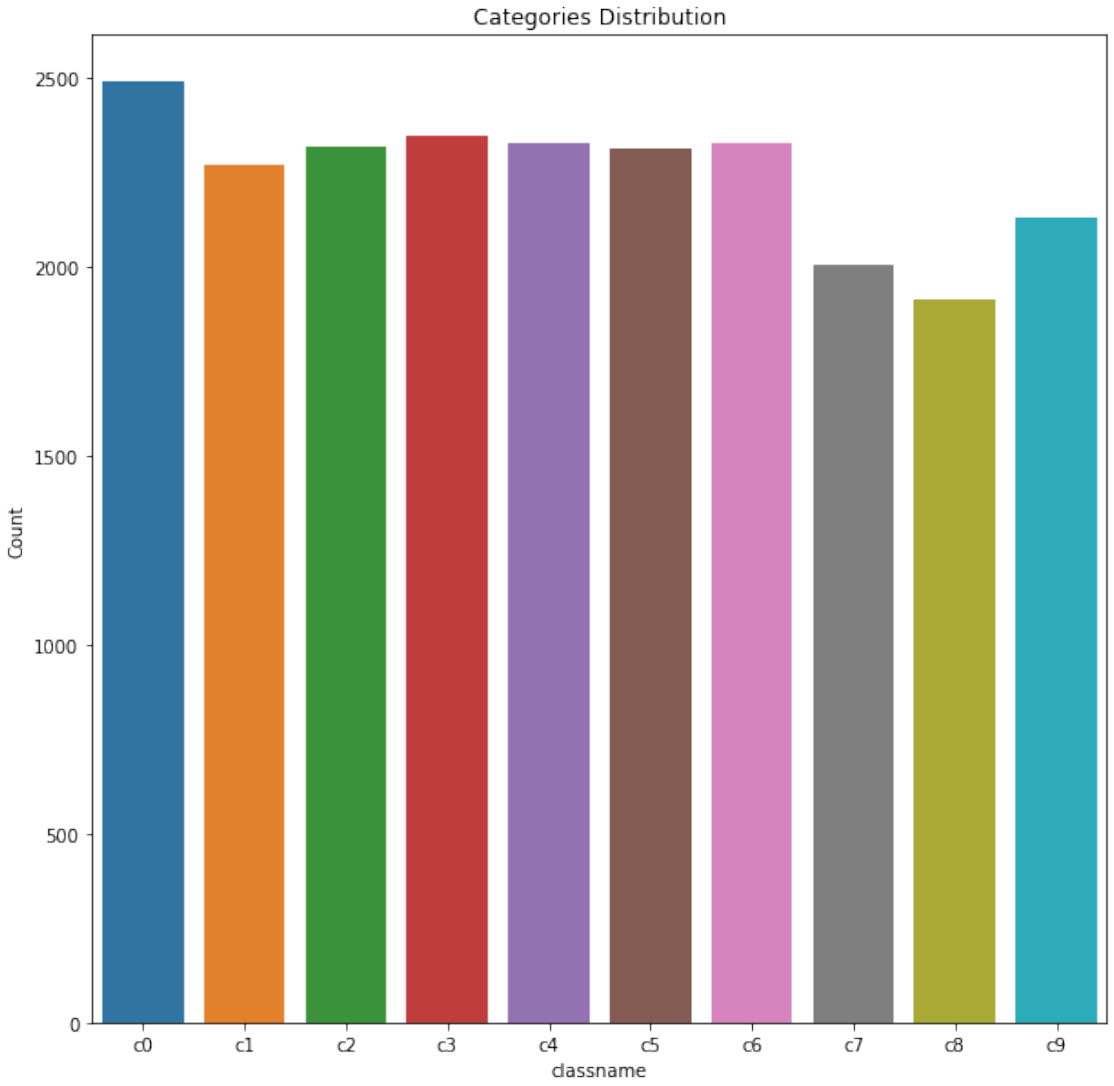
```
Test shape: (500, 64, 64, 1)
500 Test samples
```

[10]:
```
names = [item[17:19] for item in sorted(glob('imgs/train/*/'))]
test_files_size = len(np.array(glob('imgs/test/*.jpg')))
x_train_size = len(x_train)
categories_size = len(names)
x_test_size = len(x_test)
print('There are %s total images.\n' % (test_files_size + x_train_size +␣
 ↪x_test_size))
print('There are %d training images.' % x_train_size)
print('There are %d total training categories.' % categories_size)
print('There are %d validation images.' % x_test_size)
print('There are %d test images.'% test_files_size)
```

```
There are 102150 total images.

There are 17939 training images.
There are 10 total training categories.
There are 4485 validation images.
There are 79726 test images.
```

[11]:
```
# Plot figure size
plt.figure(figsize = (10,10))
# Count the number of images per category
sns.countplot(x = 'classname', data = img_list)
# Change the Axis names
plt.ylabel('Count')
plt.title('Categories Distribution')
# Show plot
plt.show()
```



[12]:
```
drivers_id = pandas.DataFrame((img_list['subject'].value_counts()).
 ↪reset_index())
drivers_id.columns = ['driver_id', 'Counts']
drivers_id
```

[12]:

	driver_id	Counts
0	p021	1237
1	p022	1233
2	p024	1226
3	p026	1196
4	p016	1078
5	p066	1034
6	p049	1011

```
7       p051     920
8       p014     876
9       p015     875
10      p035     848
11      p047     835
12      p081     823
13      p012     823
14      p064     820
15      p075     814
16      p061     809
17      p056     794
18      p050     790
19      p052     740
20      p002     725
21      p045     724
22      p039     651
23      p041     605
24      p042     591
25      p072     346
```

[0]:
```
activity_map = {'c0': 'Safe driving',
                'c1': 'Texting - right',
                'c2': 'Talking on the phone - right',
                'c3': 'Texting - left',
                'c4': 'Talking on the phone - left',
                'c5': 'Operating the radio',
                'c6': 'Drinking',
                'c7': 'Reaching behind',
                'c8': 'Hair and makeup',
                'c9': 'Talking to passenger'}
```

[0]:
```
batch_size = 40
epoch = 10

#callbacks = [checkpointer, es]
```

[0]:
```
CNNmodel = Sequential()

CNNmodel.add(Conv2D(32,(3,3),activation='relu',input_shape=(img_rows, img_cols,␣
↪color_type)))
CNNmodel.add(BatchNormalization())
CNNmodel.add(Conv2D(32,(3,3),activation='relu',padding='same'))
CNNmodel.add(BatchNormalization(axis = 3))
CNNmodel.add(MaxPooling2D(pool_size=(2,2),padding='same'))
CNNmodel.add(Dropout(0.3))

CNNmodel.add(Conv2D(64,(3,3),activation='relu',padding='same'))
```

```
CNNmodel.add(BatchNormalization())
CNNmodel.add(Conv2D(64,(3,3),activation='relu',padding='same'))
CNNmodel.add(BatchNormalization(axis = 3))
CNNmodel.add(MaxPooling2D(pool_size=(2,2),padding='same'))
CNNmodel.add(Dropout(0.3))

CNNmodel.add(Conv2D(128,(3,3),activation='relu',padding='same'))
CNNmodel.add(BatchNormalization())
CNNmodel.add(Conv2D(128,(3,3),activation='relu',padding='same'))
CNNmodel.add(BatchNormalization(axis = 3))
CNNmodel.add(MaxPooling2D(pool_size=(2,2),padding='same'))
CNNmodel.add(Dropout(0.5))

CNNmodel.add(Flatten())
CNNmodel.add(Dense(512,activation='relu'))
CNNmodel.add(BatchNormalization())
CNNmodel.add(Dropout(0.5))
CNNmodel.add(Dense(128,activation='relu'))
CNNmodel.add(Dropout(0.25))
CNNmodel.add(Dense(10,activation='softmax'))
```

[21]:
```
CNNmodel.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 62, 62, 32)        320
_________________________________________________________________
batch_normalization_7 (Batch (None, 62, 62, 32)        128
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 62, 62, 32)        9248
_________________________________________________________________
batch_normalization_8 (Batch (None, 62, 62, 32)        128
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 31, 31, 64)        18496
_________________________________________________________________
batch_normalization_9 (Batch (None, 31, 31, 64)        256
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 31, 31, 64)        36928
_________________________________________________________________
batch_normalization_10 (Batc (None, 31, 31, 64)        256
_________________________________________________________________
```

```
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 16, 16, 128)       73856
_________________________________________________________________
batch_normalization_11 (Batc (None, 16, 16, 128)       512
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 16, 16, 128)       147584
_________________________________________________________________
batch_normalization_12 (Batc (None, 16, 16, 128)       512
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
dropout_7 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               4194816
_________________________________________________________________
batch_normalization_13 (Batc (None, 512)               2048
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_9 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                1290
=================================================================
Total params: 4,552,042
Trainable params: 4,550,122
Non-trainable params: 1,920
_________________________________________________________________
```

[0]:
```
CNNmodel.compile(optimizer='rmsprop', loss='categorical_crossentropy',␣
↪metrics=['accuracy'])
```

[23]:
```
test_CNN = CNNmodel.fit(x_train, y_train,
          validation_data=(x_test, y_test),
          epochs=epoch, batch_size=batch_size, verbose=1)
```

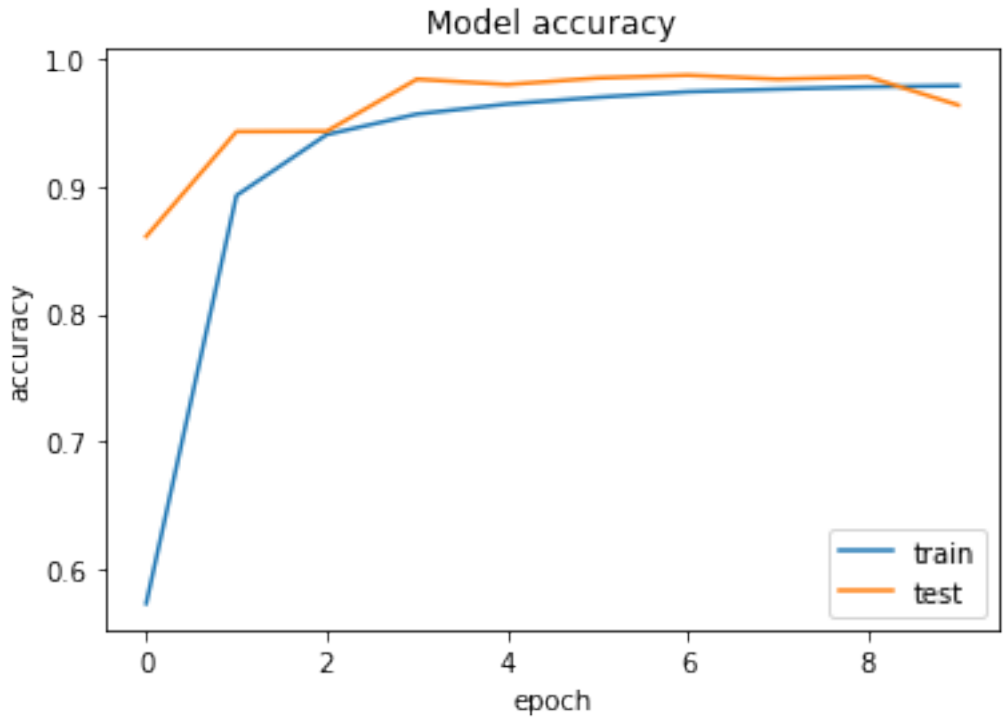
```
Epoch 1/10
449/449 [==============================] - 7s 16ms/step - loss: 1.2921 -
accuracy: 0.5734 - val_loss: 0.4882 - val_accuracy: 0.8606
Epoch 2/10
449/449 [==============================] - 7s 15ms/step - loss: 0.3369 -
accuracy: 0.8927 - val_loss: 0.2187 - val_accuracy: 0.9427
```

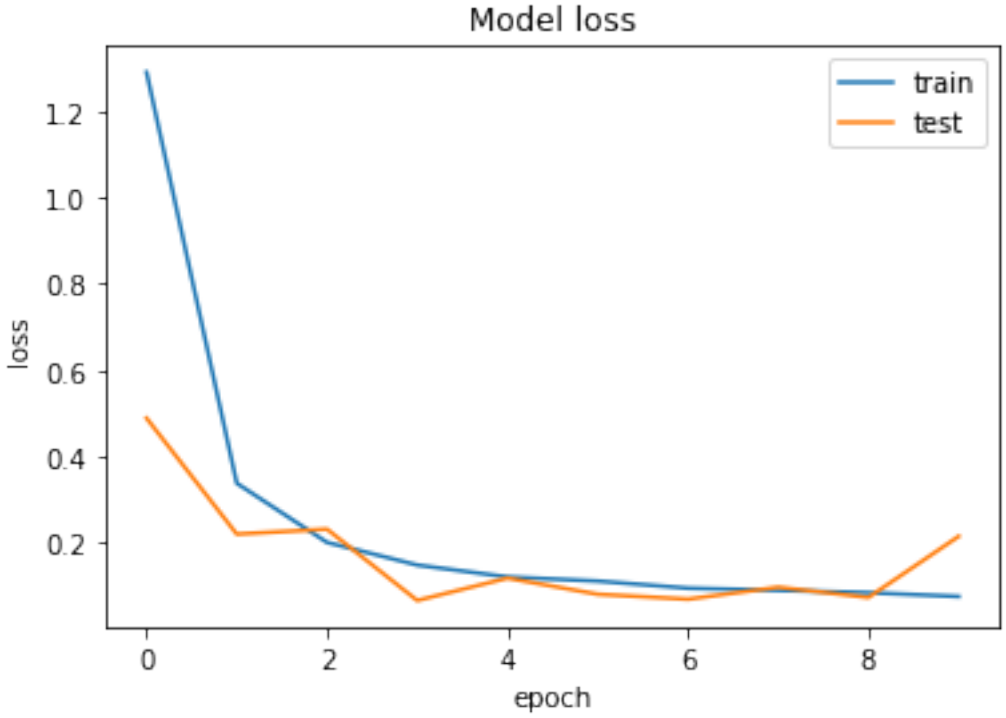
```
Epoch 3/10
449/449 [==============================] - 7s 15ms/step - loss: 0.1994 -
accuracy: 0.9404 - val_loss: 0.2302 - val_accuracy: 0.9429
Epoch 4/10
449/449 [==============================] - 7s 15ms/step - loss: 0.1468 -
accuracy: 0.9564 - val_loss: 0.0645 - val_accuracy: 0.9837
Epoch 5/10
449/449 [==============================] - 7s 15ms/step - loss: 0.1192 -
accuracy: 0.9643 - val_loss: 0.1164 - val_accuracy: 0.9795
Epoch 6/10
449/449 [==============================] - 7s 15ms/step - loss: 0.1096 -
accuracy: 0.9695 - val_loss: 0.0790 - val_accuracy: 0.9846
Epoch 7/10
449/449 [==============================] - 7s 15ms/step - loss: 0.0931 -
accuracy: 0.9739 - val_loss: 0.0676 - val_accuracy: 0.9868
Epoch 8/10
449/449 [==============================] - 7s 15ms/step - loss: 0.0881 -
accuracy: 0.9759 - val_loss: 0.0953 - val_accuracy: 0.9837
Epoch 9/10
449/449 [==============================] - 7s 15ms/step - loss: 0.0826 -
accuracy: 0.9777 - val_loss: 0.0716 - val_accuracy: 0.9855
Epoch 10/10
449/449 [==============================] - 7s 15ms/step - loss: 0.0738 -
accuracy: 0.9786 - val_loss: 0.2136 - val_accuracy: 0.9634
```

[0]:
```python
def plot_model_result(result):
    plt.plot(result.history['accuracy'])
    plt.plot(result.history['val_accuracy'])
    plt.title('Model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'])
    plt.show()

    # Summarize history for loss
    plt.plot(result.history['loss'])
    plt.plot(result.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'])
    plt.show()
```

[27]:
```python
plot_model_result(test_CNN)
```


Model accuracy
1.0
0.9
0.8
0.7
0.6
accuracy
0
2
4
6
8
epoch
train
test


Model loss
1.2
1.0
0.8
0.6
0.4
0.2
loss
0
2
4
6
8
epoch
train
test

[0]: