project_code

July 20, 2020

0.1 # Image classification with Machine Learning

0.2 University of Milan

0.2.1 DataScience and Economics - Machine Learning Module

Authors : Andrea Ierardi, Emanuele Morales, Gregorio Luigi Saporito

How to load the dataset:

```
from google.colab import drive
drive.mount('/content/gdrive')


import os
os.environ['KAGGLE_CONFIG_DIR'] = "/content/gdrive/My Drive/Kaggle"


%cd /content/gdrive/My Drive/Kaggle


!kaggle datasets download --force -d moltean/fruits

!unzip fruits.zip
```

1 Image classification with Neural Networks

1.1 1. The dataset

1.1.1 1.1 Libraries

[ ]:
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import os
from tqdm import tqdm
import random
import pandas as pd

from plotnine import *
from sklearn.decomposition import PCA
```

```
from sklearn.datasets import load_files
from keras.preprocessing.image import array_to_img, img_to_array, load_img
from sklearn import preprocessing

from keras.utils import np_utils
from sklearn.utils import shuffle
import numpy as np

import matplotlib.pyplot as plt


from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D
from keras.layers import Activation, Dense, Flatten, Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint
from keras import backend as K

from keras.applications import MobileNetV2
```

```
Using TensorFlow backend.
```

2 1.1 Data Loading

[ ]:
```
DATADIR = "fruits-360/Training"
DATADIR_test = "fruits-360/Test"

TYPES = ["Apple", "Banana", "Plum", "Pepper", "Cherry", "Grape", "Tomato",␣
↪"Potato", "Pear", "Peach"]
fruits = {}
def load_dataset(dire):
    fruits = {}
    images_as_array = []
    labels =[]
    for category in tqdm(os.listdir(dire)):
        for typ in TYPES:
            if(category.split()[0] == typ):
                fruits[category]= typ
                path = os.path.join(dire,category)
                class_num =TYPES.index(fruits[category])

                class_name = fruits[category]
                for img in tqdm(os.listdir(path)):
                    file = os.path.join(path,img)
```

```
                    images_as_array.
↪append(img_to_array(load_img(file,target_size=(32, 32))))
                    labels.append(class_num)
    images_as_array =  np.array(images_as_array)
    labels = np.array(labels)
    return images_as_array, labels
```

2.0.1 Split in test and training sets

[ ]:
```
train = load_dataset(DATADIR)
test = load_dataset(DATADIR_test)
```

[ ]:
```
x_train, y_train= train
```

[ ]:
```
x_test, y_test = test
```

2.0.2 Train and test shape

[ ]:
```
print('Train shape:')
print('X: ',x_train.shape)
print('y: ',y_train.shape)

print('Test shape')
print('X: ',x_test.shape)
print('y: ',y_test.shape)
```

```
Train shape:
X:  (32607, 32, 32, 3)
y:  (32607,)
Test shape
X:  (10906, 32, 32, 3)
y:  (10906,)
```

3 1.2 Pre-processing

3.0.1 Pre-process the labels and the images

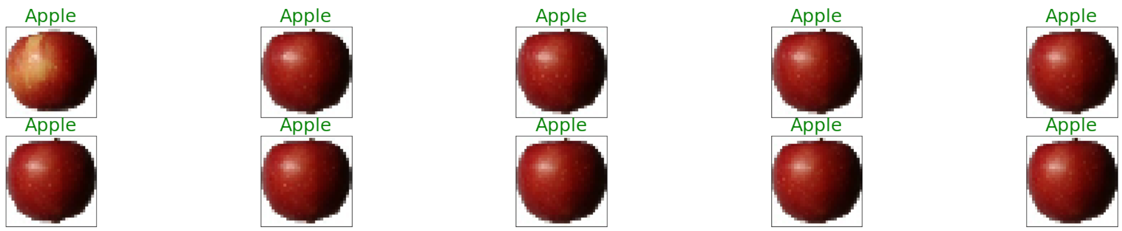
[ ]:
```
x_train = x_train.astype('float32')/255
x_test = x_test.astype('float32')/255

no_of_classes = len(np.unique(y_train))
y_train = np_utils.to_categorical(y_train,no_of_classes)
y_test = np_utils.to_categorical(y_test,no_of_classes)
```

[ ]:
```
print(y_train[0:10])
print("Number of classes: ",no_of_classes)
```

```
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
Number of classes:  10
```

3.0.2 Visualisation of the first 10 images

[ ]:
```
fig = plt.figure(figsize =(30,5))
for i in range(10):
    ax = fig.add_subplot(2,5,i+1,xticks=[],yticks=[])
    ax.imshow(np.squeeze(x_train[i]))
    ax.set_title("{}".format(TYPES[np.
↪argmax(y_train[i])]),color=("green"),fontdict= {'fontsize': '25'})
```

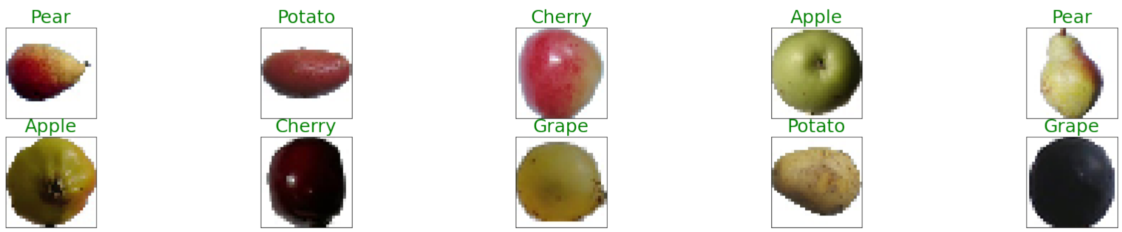


3.0.3 Suffle of the data

[ ]:
```
x_train,y_train = shuffle(x_train, y_train)
x_test,y_test = shuffle(x_test, y_test)
```

3.0.4 Visualisation of the first 10 images shuffled

[ ]:
```
fig = plt.figure(figsize =(30,5))
for i in range(10):
    ax = fig.add_subplot(2,5,i+1,xticks=[],yticks=[])
    ax.imshow(np.squeeze(x_train[i]))
    ax.set_title("{}".format(TYPES[np.
↪argmax(y_train[i])]),color=("green"),fontdict= {'fontsize': '25'})
```



3.0.5 Split in validation and test set

```
[ ]: # Using 80-20 rule
     split = len(x_test)*80//100

     print('Test len before split: ',len(x_test))
     print('Validation split len:', split)
```

```
Test len before split:  10906
Validation split len: 8724
```

```
[ ]: # Now, we have to divide the validation set into test and validation set
     x_test,x_valid = x_test[split:],x_test[:split]
     y_test,y_valid = y_test[split:],y_test[:split]
     print('Train X : ',x_train.shape)
     print('Train y :',y_train.shape)

     print('1st training image shape ',x_train[0].shape)

     print('Validation X : ',x_valid.shape)
     print('Validation y :',y_valid.shape)
     print('Test X : ',x_test.shape)
     print('Test y : ',y_test.shape)
```

```
Train X :  (32607, 32, 32, 3)
Train y : (32607, 10)
1st training image shape  (32, 32, 3)
Validation X :  (8724, 32, 32, 3)
Validation y : (8724, 10)
Test X :  (2182, 32, 32, 3)
Test y :  (2182, 10)
```

3.0.6 Definition of zero-one loss function

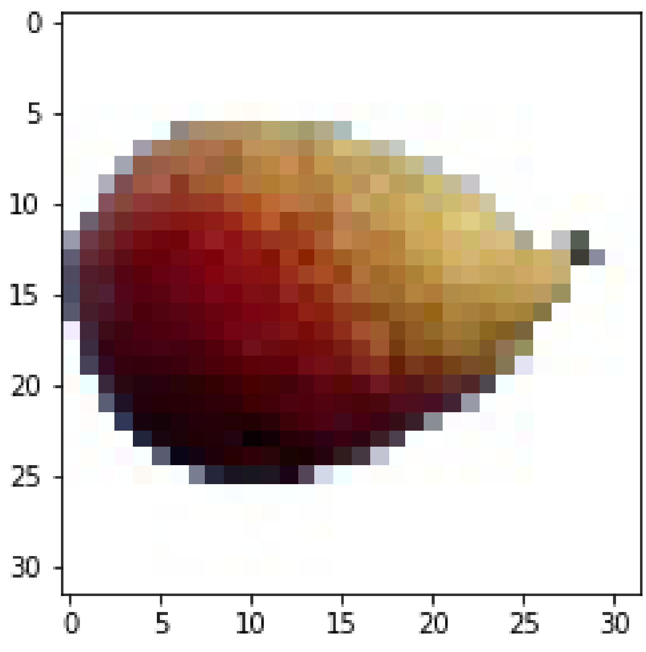
```
[ ]: def zero_one(prediz,test):
         y_hat = []
         y_t = []
         for i in range(len(prediz)):
```

```
        y_hat.append(np.argmax(prediz[i]))
        y_t.append(np.argmax(test[i]))


    loss = []
    for i in range(len(prediz)):
        if(y_hat[i] == y_t[i]):
            loss.append(0)
        else:
            loss.append(1)


    return np.mean(loss)
```

4 1.3 PCA and feed-forward NN

[ ]:
```
plt.imshow(x_train[0])
```

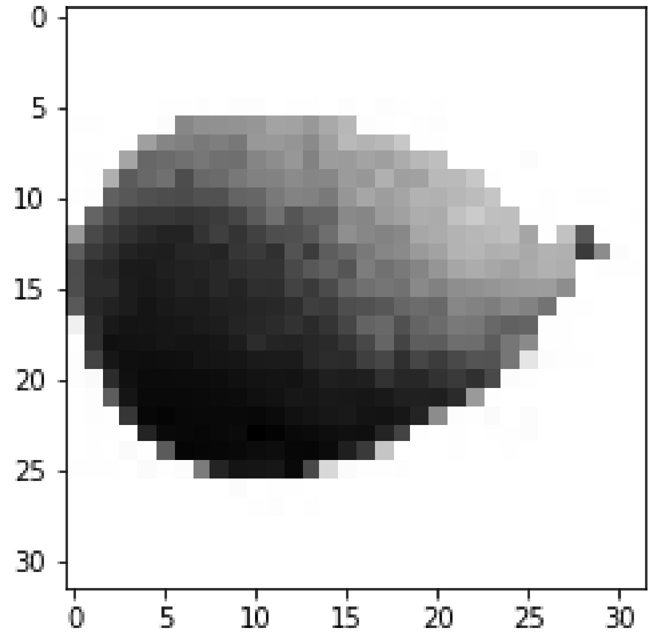
[ ]: <matplotlib.image.AxesImage at 0x7f72992944a8>



[ ]:
```
x_train[0].shape
type(x_train[1])
rgb_weights = [0.2989, 0.5870, 0.1140]
image_test = x_train[0]
image_grey = np.dot(image_test[...,:3], rgb_weights)
```

```
plt.imshow(image_grey, cmap=plt.get_cmap("gray"))
```

[ ]: <matplotlib.image.AxesImage at 0x7f729920a198>



[ ]:
```
# transform np.ndarray from rgb to grey
x_train_grey = np.ndarray(shape=(x_train.shape[0], 32, 32))
for i in range(x_train.shape[0]):
    image_convert = x_train[i]
    x_train_grey[i] = np.dot(image_convert[...,:3], rgb_weights)

x_valid_grey = np.ndarray(shape=(x_valid.shape[0], 32, 32))
for i in range(x_valid.shape[0]):
    image_convert = x_valid[i]
    x_valid_grey[i] = np.dot(image_convert[...,:3], rgb_weights)

x_test_grey = np.ndarray(shape=(x_test.shape[0], 32, 32))
for i in range(x_test.shape[0]):
    image_convert = x_test[i]
    x_test_grey[i] = np.dot(image_convert[...,:3], rgb_weights)
```

[ ]:
```
# flatten 32x32 images by concatenating them into a vector, each column of the
↪matrix will be an image
x_train_flat = np.ndarray(shape=(1024, x_train_grey.shape[0]))
for i in range(x_train_grey.shape[0]):
    x_train_flat[:,i] = np.concatenate(x_train_grey[i])
```

```
x_valid_flat = np.ndarray(shape=(1024, x_valid_grey.shape[0]))
for i in range(x_valid_grey.shape[0]):
    x_valid_flat[:,i] = np.concatenate(x_valid_grey[i])

x_test_flat = np.ndarray(shape=(1024, x_test_grey.shape[0]))
for i in range(x_test_grey.shape[0]):
    x_test_flat[:,i] = np.concatenate(x_test_grey[i])
```

[ ]:
```
standard_scaler = preprocessing.StandardScaler()
x_train_flat_T = standard_scaler.fit_transform(x_train_flat.T)
x_valid_flat_T = standard_scaler.transform(x_valid_flat.T)
x_test_flat_T = standard_scaler.transform(x_test_flat.T)
```

[ ]:
```
x_train_flat_T.shape
```

[ ]:
```
(32607, 1024)
```

[ ]:
```
x_train_flat = x_train_flat_T.T
x_valid_flat = x_valid_flat_T.T
x_test_flat = x_test_flat_T.T
```

[ ]:
```
x_train_flat.shape
```

[ ]:
```
(1024, 32607)
```

[ ]:
```
a = np.cov(x_train_flat)
b = np.linalg.eig(a)
b[0].shape
```

[ ]:
```
(1024,)
```

[ ]:
```
b
```

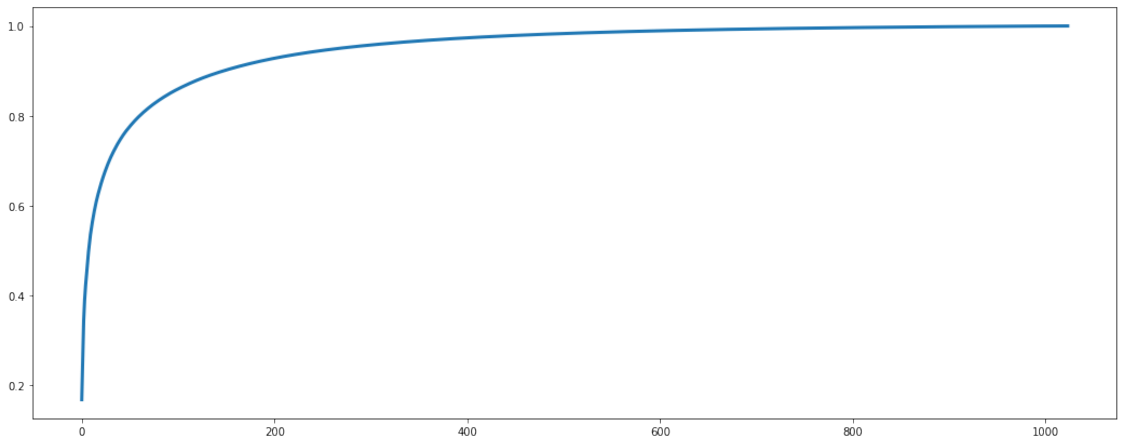
[ ]:
```
(array([384.41591328,  86.57192039,  60.39849467, …,   0.        ,
          0.        ,   0.        ]),
 array([[-2.77148730e-03,  1.99671963e-02, -7.00245270e-03, …,
          0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
        [ 1.06770166e-04,  5.26867240e-03, -3.60037909e-03, …,
          0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
        [-1.31497722e-04,  4.57673250e-03, -2.40477865e-03, …,
          0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
        …,
        [ 9.24994377e-04,  3.61739014e-03,  2.95348978e-03, …,
          0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
        [ 1.13335613e-03,  5.13559532e-03,  3.96259018e-03, …,
          0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
```

```
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, …,
  0.00000000e+00,  0.00000000e+00,  1.00000000e+00]]))
```

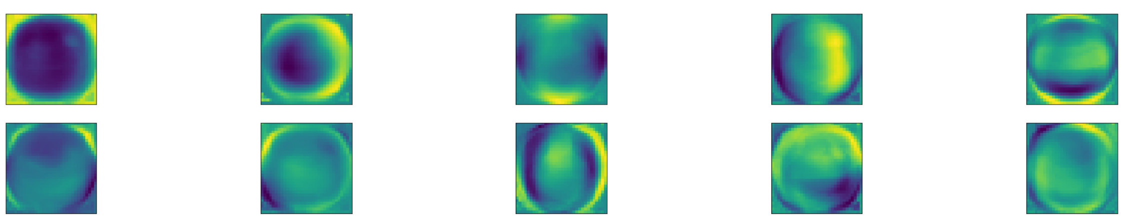
4.0.1 PCA explained variance ratio and "Eigenfruits"

[ ]:
```
pca = PCA().fit(x_train_flat)
plt.figure(figsize=(18, 7))
plt.plot(pca.explained_variance_ratio_.cumsum(), lw=3)
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7f72991f9b38>]
```



[ ]:
```
# try to plot some of the eigenvectors, the so called "eigenfruits"
fig = plt.figure(figsize =(30,5))
for i in range(10):
    ax = fig.add_subplot(2,5,i+1,xticks=[],yticks=[])
    ax.imshow(np.squeeze(b[1][:,i].reshape(32,32)))
```



4.0.2 Reduce image noise with PCA

[ ]:
```
x_train_flat.shape, x_valid_flat.shape, x_test_flat.shape
```

[ ]:
```
((1024, 32607), (1024, 8724), (1024, 2182))
```

```
[ ]: def PCA_iter(x_all,start, end, step):
         lis =[]
         for i in range(start, end, step):
                     print("\n\n===== Component: ",i,"=====\n")

                     (train,valid, test) = x_all
                     pca = PCA(n_components=i)
                     print("original shape:   ", train.shape)


                     pca.fit_transform(train)

                     train_PCA =  pca.transform(train)
                     train_new = pca.inverse_transform(train_PCA)

                     valid_PCA =  pca.transform(valid)
                     valid_new = pca.inverse_transform(valid_PCA)


                     test_PCA = pca.transform(test)
                     test_new = pca.inverse_transform(test_PCA)


                     print("transformed shape:", train_PCA.shape)
                     print("final shape:", train_new.shape)

                     tupla = (x_train_PCA, x_valid_PCA, x_test_PCA)␣
      ↪=train_new,valid_new,test_new

                     lis.append(tupla)
         return lis
```

```
[ ]: lis_PCA =  PCA_iter((x_train_flat_T,x_valid_flat_T, x_test_flat_T),10,211,20)
```

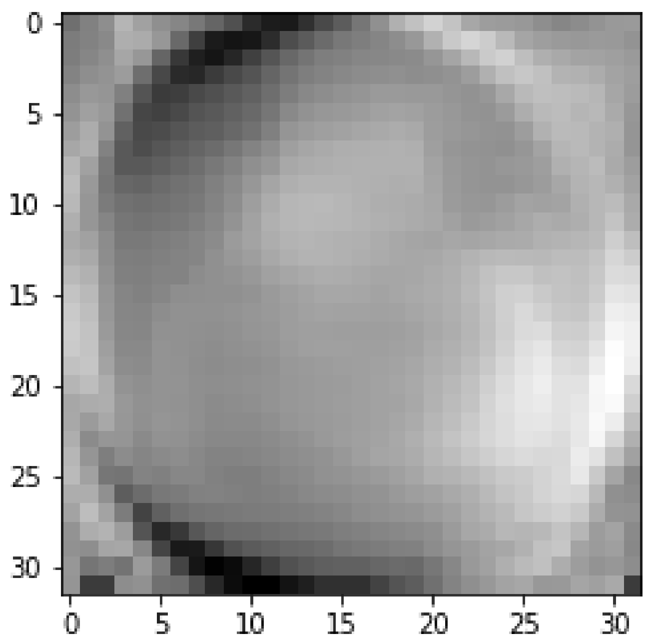
```
===== Component:  10 =====

original shape:    (32607, 1024)
transformed shape: (32607, 10)
final shape: (32607, 1024)


===== Component:  30 =====

original shape:    (32607, 1024)
transformed shape: (32607, 30)
```

```
final shape: (32607, 1024)


===== Component:  50 =====

original shape:    (32607, 1024)
transformed shape: (32607, 50)
final shape: (32607, 1024)


===== Component:  70 =====

original shape:    (32607, 1024)
transformed shape: (32607, 70)
final shape: (32607, 1024)


===== Component:  90 =====

original shape:    (32607, 1024)
transformed shape: (32607, 90)
final shape: (32607, 1024)


===== Component:  110 =====

original shape:    (32607, 1024)
transformed shape: (32607, 110)
final shape: (32607, 1024)


===== Component:  130 =====

original shape:    (32607, 1024)
transformed shape: (32607, 130)
final shape: (32607, 1024)


===== Component:  150 =====

original shape:    (32607, 1024)
transformed shape: (32607, 150)
final shape: (32607, 1024)


===== Component:  170 =====

original shape:    (32607, 1024)
```

```
transformed shape: (32607, 170)
final shape: (32607, 1024)


===== Component:  190 =====

original shape:    (32607, 1024)
transformed shape: (32607, 190)
final shape: (32607, 1024)


===== Component:  210 =====

original shape:    (32607, 1024)
transformed shape: (32607, 210)
final shape: (32607, 1024)
```

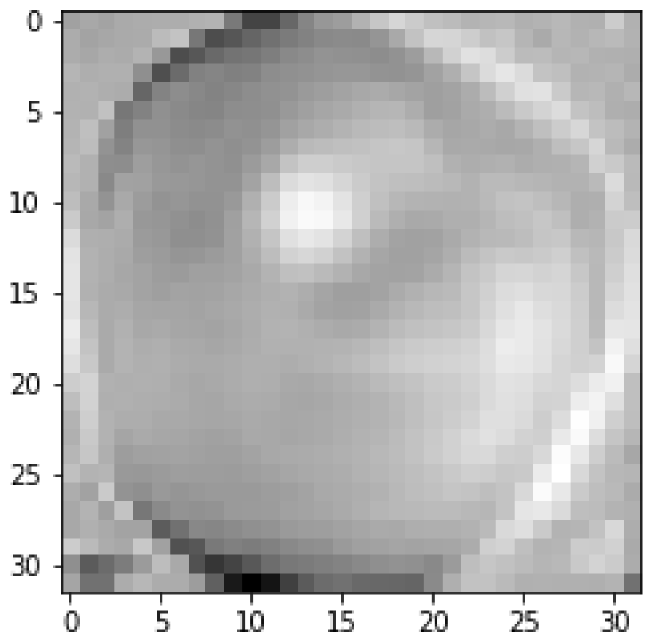
[ ]:
```
# 10 components example of the same image
tr,va,te  = lis_PCA[1]
plt.imshow(tr[2,:].reshape(32,32), cmap=plt.get_cmap("gray"))
```

[ ]: <matplotlib.image.AxesImage at 0x7f7296f20a20>



[ ]:
```
# 210 components example of an image
tr,va,te  = lis_PCA[len(lis_PCA)-1]
plt.imshow(tr[2,:].reshape(32,32), cmap=plt.get_cmap("gray"))
```

[ ]: <matplotlib.image.AxesImage at 0x7f7296e8c1d0>



4.1 Train feed-forward NN with reduced images

[ ]:
```
def FW_iter(lis_PCA, ep, bs):
    lis_FW = []
    epochs = ep
    batch_size = bs
    for itr in range(len(lis_PCA)):
        x_train_PCA, x_valid_PCA, x_test_PCA = lis_PCA[itr]


        print("FW- components: ",(itr+1)*20-10)
        #feed forward neural network
        model = tf.keras.Sequential([
          tf.keras.layers.Input(shape = (1024)),
          tf.keras.layers.Dense(32, activation = "relu"),
          tf.keras.layers.Dense(10, activation='softmax')
          ])
        model.compile(optimizer = "adam", loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])


        history = model.fit(x_train_PCA, y_train,
                            batch_size = bs,
```

```
                        epochs = epochs,
                        validation_data=(x_valid_PCA, y_valid),
                        verbose = 2

                      )

        y_pred = model.predict(x_test_PCA).round()

        zo_loss = zero_one(y_pred,y_test)
        print("Zero-one loss: ",zo_loss)
        tupla = (history, model, zo_loss)
        lis_FW.append(tupla)
    return lis_FW
```

[ ]:
```
epochs = 10
batch_size = 32
res = FW_iter(lis_PCA, epochs, batch_size)
```

```
FW- components:  10
Epoch 1/10
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.175393). Check your callbacks.
1019/1019 - 5s - loss: 0.8917 - accuracy: 0.6877 - val_loss: 1.3535 -
val_accuracy: 0.5962
Epoch 2/10
1019/1019 - 5s - loss: 0.6398 - accuracy: 0.7802 - val_loss: 1.3279 -
val_accuracy: 0.6405
Epoch 3/10
1019/1019 - 5s - loss: 0.5631 - accuracy: 0.8033 - val_loss: 1.2432 -
val_accuracy: 0.6723
Epoch 4/10
1019/1019 - 5s - loss: 0.5202 - accuracy: 0.8203 - val_loss: 1.2590 -
val_accuracy: 0.6750
Epoch 5/10
1019/1019 - 5s - loss: 0.4846 - accuracy: 0.8326 - val_loss: 1.3413 -
val_accuracy: 0.6714
Epoch 6/10
1019/1019 - 5s - loss: 0.4573 - accuracy: 0.8445 - val_loss: 1.2513 -
val_accuracy: 0.6985
Epoch 7/10
1019/1019 - 5s - loss: 0.4322 - accuracy: 0.8521 - val_loss: 1.2741 -
val_accuracy: 0.6983
Epoch 8/10
1019/1019 - 5s - loss: 0.4193 - accuracy: 0.8566 - val_loss: 1.3041 -
val_accuracy: 0.7063
Epoch 9/10
1019/1019 - 5s - loss: 0.4008 - accuracy: 0.8640 - val_loss: 1.3032 -
```

```
val_accuracy: 0.7003
Epoch 10/10
1019/1019 - 5s - loss: 0.3982 - accuracy: 0.8641 - val_loss: 1.3579 -
val_accuracy: 0.6978
Zero-one loss:  0.28689275893675525
FW- components:  30
Epoch 1/10
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.261349). Check your callbacks.
1019/1019 - 6s - loss: 0.6436 - accuracy: 0.7845 - val_loss: 1.0769 -
val_accuracy: 0.6961
Epoch 2/10
1019/1019 - 5s - loss: 0.3235 - accuracy: 0.8948 - val_loss: 1.0283 -
val_accuracy: 0.7299
Epoch 3/10
1019/1019 - 5s - loss: 0.2424 - accuracy: 0.9211 - val_loss: 1.0095 -
val_accuracy: 0.7387
Epoch 4/10
1019/1019 - 5s - loss: 0.1909 - accuracy: 0.9384 - val_loss: 0.9939 -
val_accuracy: 0.7633
Epoch 5/10
1019/1019 - 5s - loss: 0.1558 - accuracy: 0.9502 - val_loss: 1.1439 -
val_accuracy: 0.7466
Epoch 6/10
1019/1019 - 5s - loss: 0.1404 - accuracy: 0.9538 - val_loss: 1.0946 -
val_accuracy: 0.7713
Epoch 7/10
1019/1019 - 5s - loss: 0.1227 - accuracy: 0.9603 - val_loss: 1.1294 -
val_accuracy: 0.7809
Epoch 8/10
1019/1019 - 5s - loss: 0.1098 - accuracy: 0.9649 - val_loss: 1.1915 -
val_accuracy: 0.7785
Epoch 9/10
1019/1019 - 5s - loss: 0.1003 - accuracy: 0.9675 - val_loss: 1.1726 -
val_accuracy: 0.7894
Epoch 10/10
1019/1019 - 5s - loss: 0.0947 - accuracy: 0.9702 - val_loss: 1.2234 -
val_accuracy: 0.7822
Zero-one loss:  0.19798350137488543
FW- components:  50
Epoch 1/10
1019/1019 - 5s - loss: 0.5459 - accuracy: 0.8254 - val_loss: 0.8101 -
val_accuracy: 0.7436
Epoch 2/10
1019/1019 - 5s - loss: 0.2140 - accuracy: 0.9335 - val_loss: 0.8365 -
val_accuracy: 0.7656
Epoch 3/10
1019/1019 - 5s - loss: 0.1433 - accuracy: 0.9549 - val_loss: 0.8540 -
```

```
val_accuracy: 0.7918
Epoch 4/10
1019/1019 - 5s - loss: 0.1039 - accuracy: 0.9682 - val_loss: 0.8805 -
val_accuracy: 0.7915
Epoch 5/10
1019/1019 - 5s - loss: 0.0848 - accuracy: 0.9738 - val_loss: 0.9716 -
val_accuracy: 0.8034
Epoch 6/10
1019/1019 - 5s - loss: 0.0715 - accuracy: 0.9778 - val_loss: 0.9149 -
val_accuracy: 0.8155
Epoch 7/10
1019/1019 - 5s - loss: 0.0664 - accuracy: 0.9790 - val_loss: 1.1081 -
val_accuracy: 0.8093
Epoch 8/10
1019/1019 - 5s - loss: 0.0539 - accuracy: 0.9832 - val_loss: 1.1118 -
val_accuracy: 0.7929
Epoch 9/10
1019/1019 - 5s - loss: 0.0512 - accuracy: 0.9835 - val_loss: 1.0935 -
val_accuracy: 0.8323
Epoch 10/10
1019/1019 - 5s - loss: 0.0465 - accuracy: 0.9845 - val_loss: 1.0836 -
val_accuracy: 0.8343
Zero-one loss:  0.15902841429880843
FW- components:  70
Epoch 1/10
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.200094). Check your callbacks.
1019/1019 - 5s - loss: 0.4945 - accuracy: 0.8395 - val_loss: 0.8700 -
val_accuracy: 0.7372
Epoch 2/10
1019/1019 - 5s - loss: 0.1802 - accuracy: 0.9462 - val_loss: 0.7953 -
val_accuracy: 0.7839
Epoch 3/10
1019/1019 - 5s - loss: 0.1150 - accuracy: 0.9651 - val_loss: 0.8178 -
val_accuracy: 0.8015
Epoch 4/10
1019/1019 - 5s - loss: 0.0828 - accuracy: 0.9752 - val_loss: 0.8297 -
val_accuracy: 0.7953
Epoch 5/10
1019/1019 - 5s - loss: 0.0659 - accuracy: 0.9796 - val_loss: 0.7400 -
val_accuracy: 0.8329
Epoch 6/10
1019/1019 - 5s - loss: 0.0544 - accuracy: 0.9829 - val_loss: 0.8350 -
val_accuracy: 0.8300
Epoch 7/10
1019/1019 - 5s - loss: 0.0438 - accuracy: 0.9867 - val_loss: 0.8506 -
val_accuracy: 0.8344
Epoch 8/10
```

```
1019/1019 - 5s - loss: 0.0406 - accuracy: 0.9879 - val_loss: 0.8914 -
val_accuracy: 0.8357
Epoch 9/10
1019/1019 - 6s - loss: 0.0387 - accuracy: 0.9884 - val_loss: 0.8949 -
val_accuracy: 0.8410
Epoch 10/10
1019/1019 - 6s - loss: 0.0312 - accuracy: 0.9904 - val_loss: 0.9068 -
val_accuracy: 0.8454
Zero-one loss:  0.1457378551787351
FW- components:  90
Epoch 1/10
1019/1019 - 5s - loss: 0.5080 - accuracy: 0.8416 - val_loss: 0.7913 -
val_accuracy: 0.7591
Epoch 2/10
1019/1019 - 5s - loss: 0.1627 - accuracy: 0.9520 - val_loss: 0.7948 -
val_accuracy: 0.7869
Epoch 3/10
1019/1019 - 5s - loss: 0.0948 - accuracy: 0.9742 - val_loss: 0.7662 -
val_accuracy: 0.8137
Epoch 4/10
1019/1019 - 5s - loss: 0.0640 - accuracy: 0.9823 - val_loss: 0.7883 -
val_accuracy: 0.8290
Epoch 5/10
1019/1019 - 5s - loss: 0.0502 - accuracy: 0.9860 - val_loss: 0.7893 -
val_accuracy: 0.8356
Epoch 6/10
1019/1019 - 5s - loss: 0.0395 - accuracy: 0.9891 - val_loss: 0.8706 -
val_accuracy: 0.8399
Epoch 7/10
1019/1019 - 5s - loss: 0.0400 - accuracy: 0.9881 - val_loss: 1.0676 -
val_accuracy: 0.8155
Epoch 8/10
1019/1019 - 5s - loss: 0.0282 - accuracy: 0.9915 - val_loss: 1.0146 -
val_accuracy: 0.8299
Epoch 9/10
1019/1019 - 5s - loss: 0.0238 - accuracy: 0.9928 - val_loss: 0.9587 -
val_accuracy: 0.8548
Epoch 10/10
1019/1019 - 5s - loss: 0.0224 - accuracy: 0.9929 - val_loss: 1.0540 -
val_accuracy: 0.8471
Zero-one loss:  0.15627864344637946
FW- components:  110
Epoch 1/10
1019/1019 - 5s - loss: 0.4724 - accuracy: 0.8546 - val_loss: 0.8186 -
val_accuracy: 0.7603
Epoch 2/10
1019/1019 - 5s - loss: 0.1382 - accuracy: 0.9634 - val_loss: 0.7725 -
val_accuracy: 0.8089
```

```
Epoch 3/10
1019/1019 - 5s - loss: 0.0794 - accuracy: 0.9783 - val_loss: 0.8203 -
val_accuracy: 0.8191
Epoch 4/10
1019/1019 - 5s - loss: 0.0528 - accuracy: 0.9850 - val_loss: 0.9337 -
val_accuracy: 0.8204
Epoch 5/10
1019/1019 - 5s - loss: 0.0417 - accuracy: 0.9879 - val_loss: 0.9542 -
val_accuracy: 0.8313
Epoch 6/10
1019/1019 - 5s - loss: 0.0369 - accuracy: 0.9895 - val_loss: 0.9346 -
val_accuracy: 0.8412
Epoch 7/10
1019/1019 - 5s - loss: 0.0267 - accuracy: 0.9927 - val_loss: 1.0282 -
val_accuracy: 0.8474
Epoch 8/10
1019/1019 - 5s - loss: 0.0211 - accuracy: 0.9939 - val_loss: 1.1628 -
val_accuracy: 0.8286
Epoch 9/10
1019/1019 - 5s - loss: 0.0264 - accuracy: 0.9914 - val_loss: 1.1593 -
val_accuracy: 0.8408
Epoch 10/10
1019/1019 - 5s - loss: 0.0205 - accuracy: 0.9938 - val_loss: 1.2966 -
val_accuracy: 0.8436
Zero-one loss:  0.14848762603116408
FW- components:  130
Epoch 1/10
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.148986). Check your callbacks.
1019/1019 - 5s - loss: 0.5150 - accuracy: 0.8395 - val_loss: 0.7711 -
val_accuracy: 0.7811
Epoch 2/10
1019/1019 - 5s - loss: 0.1556 - accuracy: 0.9550 - val_loss: 0.7043 -
val_accuracy: 0.8064
Epoch 3/10
1019/1019 - 5s - loss: 0.0860 - accuracy: 0.9749 - val_loss: 0.7319 -
val_accuracy: 0.8285
Epoch 4/10
1019/1019 - 5s - loss: 0.0547 - accuracy: 0.9850 - val_loss: 0.8580 -
val_accuracy: 0.8292
Epoch 5/10
1019/1019 - 5s - loss: 0.0448 - accuracy: 0.9874 - val_loss: 0.9114 -
val_accuracy: 0.8353
Epoch 6/10
1019/1019 - 5s - loss: 0.0326 - accuracy: 0.9913 - val_loss: 0.9489 -
val_accuracy: 0.8368
Epoch 7/10
1019/1019 - 5s - loss: 0.0302 - accuracy: 0.9910 - val_loss: 0.8936 -
```

```
val_accuracy: 0.8608
Epoch 8/10
1019/1019 - 5s - loss: 0.0265 - accuracy: 0.9921 - val_loss: 0.9696 -
val_accuracy: 0.8584
Epoch 9/10
1019/1019 - 5s - loss: 0.0264 - accuracy: 0.9920 - val_loss: 1.0818 -
val_accuracy: 0.8422
Epoch 10/10
1019/1019 - 5s - loss: 0.0166 - accuracy: 0.9955 - val_loss: 1.1559 -
val_accuracy: 0.8453
Zero-one loss:  0.152153987167736
FW- components:  150
Epoch 1/10
1019/1019 - 5s - loss: 0.4667 - accuracy: 0.8549 - val_loss: 0.7300 -
val_accuracy: 0.7691
Epoch 2/10
1019/1019 - 5s - loss: 0.1365 - accuracy: 0.9635 - val_loss: 0.6678 -
val_accuracy: 0.8116
Epoch 3/10
1019/1019 - 5s - loss: 0.0762 - accuracy: 0.9800 - val_loss: 0.6308 -
val_accuracy: 0.8370
Epoch 4/10
1019/1019 - 5s - loss: 0.0458 - accuracy: 0.9884 - val_loss: 0.6961 -
val_accuracy: 0.8396
Epoch 5/10
1019/1019 - 5s - loss: 0.0426 - accuracy: 0.9883 - val_loss: 0.8004 -
val_accuracy: 0.8357
Epoch 6/10
1019/1019 - 5s - loss: 0.0267 - accuracy: 0.9929 - val_loss: 0.7318 -
val_accuracy: 0.8495
Epoch 7/10
1019/1019 - 5s - loss: 0.0267 - accuracy: 0.9921 - val_loss: 0.7954 -
val_accuracy: 0.8433
Epoch 8/10
1019/1019 - 5s - loss: 0.0232 - accuracy: 0.9931 - val_loss: 1.0402 -
val_accuracy: 0.8232
Epoch 9/10
1019/1019 - 5s - loss: 0.0190 - accuracy: 0.9945 - val_loss: 0.9076 -
val_accuracy: 0.8549
Epoch 10/10
1019/1019 - 5s - loss: 0.0184 - accuracy: 0.9949 - val_loss: 0.9122 -
val_accuracy: 0.8505
Zero-one loss:  0.14115490375802017
FW- components:  170
Epoch 1/10
1019/1019 - 5s - loss: 0.5013 - accuracy: 0.8487 - val_loss: 0.7488 -
val_accuracy: 0.7780
Epoch 2/10
```

```
1019/1019 - 5s - loss: 0.1291 - accuracy: 0.9651 - val_loss: 0.7400 -
val_accuracy: 0.8058
Epoch 3/10
1019/1019 - 5s - loss: 0.0664 - accuracy: 0.9830 - val_loss: 0.6988 -
val_accuracy: 0.8416
Epoch 4/10
1019/1019 - 5s - loss: 0.0447 - accuracy: 0.9883 - val_loss: 0.7572 -
val_accuracy: 0.8386
Epoch 5/10
1019/1019 - 5s - loss: 0.0339 - accuracy: 0.9907 - val_loss: 0.8077 -
val_accuracy: 0.8454
Epoch 6/10
1019/1019 - 5s - loss: 0.0254 - accuracy: 0.9930 - val_loss: 0.7187 -
val_accuracy: 0.8663
Epoch 7/10
1019/1019 - 5s - loss: 0.0213 - accuracy: 0.9939 - val_loss: 0.7830 -
val_accuracy: 0.8504
Epoch 8/10
1019/1019 - 5s - loss: 0.0189 - accuracy: 0.9946 - val_loss: 0.7266 -
val_accuracy: 0.8778
Epoch 9/10
1019/1019 - 5s - loss: 0.0124 - accuracy: 0.9968 - val_loss: 0.8611 -
val_accuracy: 0.8610
Epoch 10/10
1019/1019 - 5s - loss: 0.0178 - accuracy: 0.9946 - val_loss: 0.8310 -
val_accuracy: 0.8737
Zero-one loss:  0.12465627864344637
FW- components:  190
Epoch 1/10
1019/1019 - 5s - loss: 0.4545 - accuracy: 0.8607 - val_loss: 0.7529 -
val_accuracy: 0.7865
Epoch 2/10
1019/1019 - 5s - loss: 0.1264 - accuracy: 0.9652 - val_loss: 0.7090 -
val_accuracy: 0.8212
Epoch 3/10
1019/1019 - 5s - loss: 0.0705 - accuracy: 0.9815 - val_loss: 0.6724 -
val_accuracy: 0.8435
Epoch 4/10
1019/1019 - 5s - loss: 0.0461 - accuracy: 0.9876 - val_loss: 0.7778 -
val_accuracy: 0.8344
Epoch 5/10
1019/1019 - 5s - loss: 0.0383 - accuracy: 0.9895 - val_loss: 0.7378 -
val_accuracy: 0.8549
Epoch 6/10
1019/1019 - 5s - loss: 0.0264 - accuracy: 0.9932 - val_loss: 0.9976 -
val_accuracy: 0.8267
Epoch 7/10
1019/1019 - 5s - loss: 0.0244 - accuracy: 0.9929 - val_loss: 0.9187 -
```

```
val_accuracy: 0.8483
Epoch 8/10
1019/1019 - 5s - loss: 0.0181 - accuracy: 0.9950 - val_loss: 0.9842 -
val_accuracy: 0.8478
Epoch 9/10
1019/1019 - 6s - loss: 0.0234 - accuracy: 0.9936 - val_loss: 0.9568 -
val_accuracy: 0.8530
Epoch 10/10
1019/1019 - 6s - loss: 0.0140 - accuracy: 0.9960 - val_loss: 0.9279 -
val_accuracy: 0.8657
Zero-one loss:  0.12557286892758937
FW- components:  210
Epoch 1/10
1019/1019 - 5s - loss: 0.4542 - accuracy: 0.8597 - val_loss: 0.7646 -
val_accuracy: 0.7775
Epoch 2/10
1019/1019 - 5s - loss: 0.1189 - accuracy: 0.9685 - val_loss: 0.7023 -
val_accuracy: 0.8039
Epoch 3/10
1019/1019 - 5s - loss: 0.0641 - accuracy: 0.9839 - val_loss: 0.7556 -
val_accuracy: 0.8193
Epoch 4/10
1019/1019 - 5s - loss: 0.0412 - accuracy: 0.9896 - val_loss: 0.8243 -
val_accuracy: 0.8222
Epoch 5/10
1019/1019 - 5s - loss: 0.0262 - accuracy: 0.9936 - val_loss: 0.7042 -
val_accuracy: 0.8571
Epoch 6/10
1019/1019 - 5s - loss: 0.0262 - accuracy: 0.9922 - val_loss: 0.7838 -
val_accuracy: 0.8453
Epoch 7/10
1019/1019 - 5s - loss: 0.0157 - accuracy: 0.9956 - val_loss: 0.8158 -
val_accuracy: 0.8592
Epoch 8/10
1019/1019 - 5s - loss: 0.0245 - accuracy: 0.9926 - val_loss: 0.9040 -
val_accuracy: 0.8639
Epoch 9/10
1019/1019 - 5s - loss: 0.0147 - accuracy: 0.9961 - val_loss: 0.9469 -
val_accuracy: 0.8372
Epoch 10/10
1019/1019 - 5s - loss: 0.0174 - accuracy: 0.9947 - val_loss: 1.0241 -
val_accuracy: 0.8486
Zero-one loss:  0.1457378551787351
```

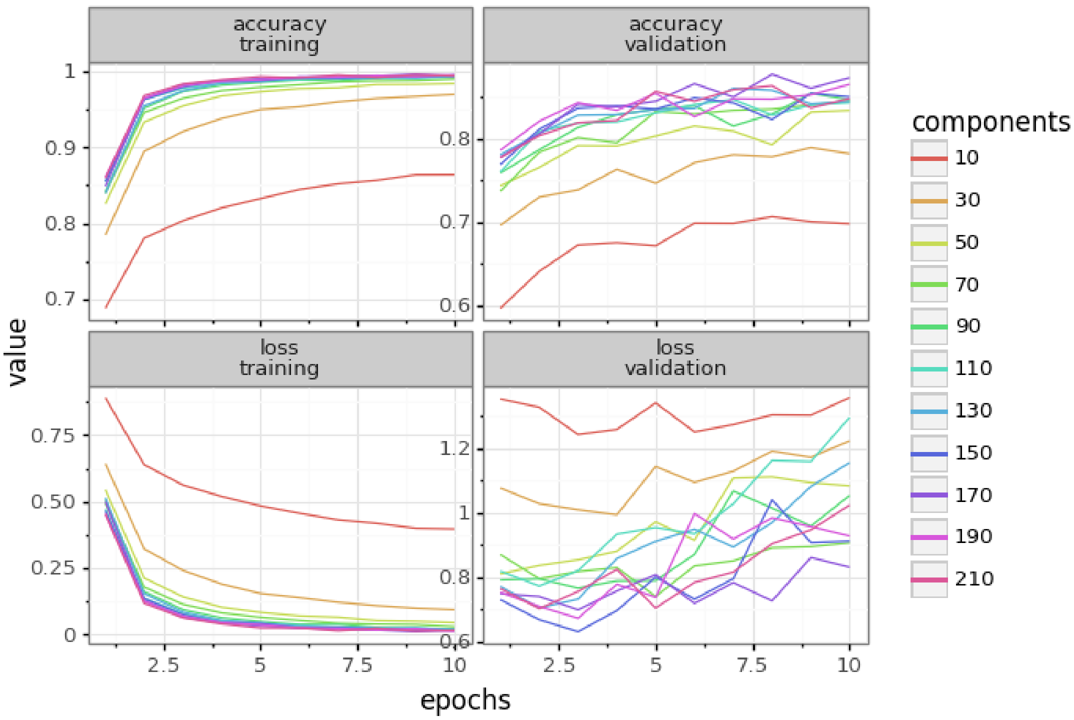
[ ]:
```
df = pd.DataFrame(
columns = ['epochs', 'valid', 'components', 'accuracy', 'value']
)
```

```
for itr in range(len(res)):

    time =  [i for i in range(1,epochs+1)]
    valids =  [0 for i in range(1,epochs+1)]
    components = [(itr+1)*20-10 for i in range(1,epochs+1)]


    accur = [1 for i in range(1,epochs+1)]
    acc = res[itr][0].history['accuracy']

    df1=  pd.DataFrame(data= np.vstack((time,valids,components,accur,acc)).T,␣
 ↪columns = ['epochs', 'valid', 'components', 'accuracy', 'value'])

    loss= res[itr][0].history['loss']
    accur = [0 for i in range(1,epochs+1)]
    df2=  pd.DataFrame(data= np.vstack((time,valids,components,accur,loss)).T,␣
 ↪columns = ['epochs', 'valid', 'components', 'accuracy', 'value'])

    valids =  [1 for i in range(1,epochs+1)]
    accur = [1 for i in range(1,epochs+1)]
    val_acc = res[itr][0].history['val_accuracy']

    df3=  pd.DataFrame(data= np.vstack((time,valids,components,accur,val_acc)).
 ↪T, columns = ['epochs', 'valid', 'components', 'accuracy', 'value'])

    accur = [0 for i in range(1,epochs+1)]

    val_loss = res[itr][0].history['val_loss']
    df4=  pd.DataFrame(data= np.vstack((time,valids,components,accur,val_loss)).
 ↪T, columns = ['epochs', 'valid', 'components', 'accuracy', 'value'])

    df = df.append(df1.append(df2).append(df3).append(df4))
```

[ ]:
```
df['components'] = df['components'].astype('category')
df = df.assign(accuracy = ['accuracy' if accuracy == 1. else 'loss' for␣
 ↪accuracy in df['accuracy']])
df = df.assign(valid = ['validation' if valid == 1. else 'training' for valid␣
 ↪in df['valid']])
df['accuracy'].unique()
```

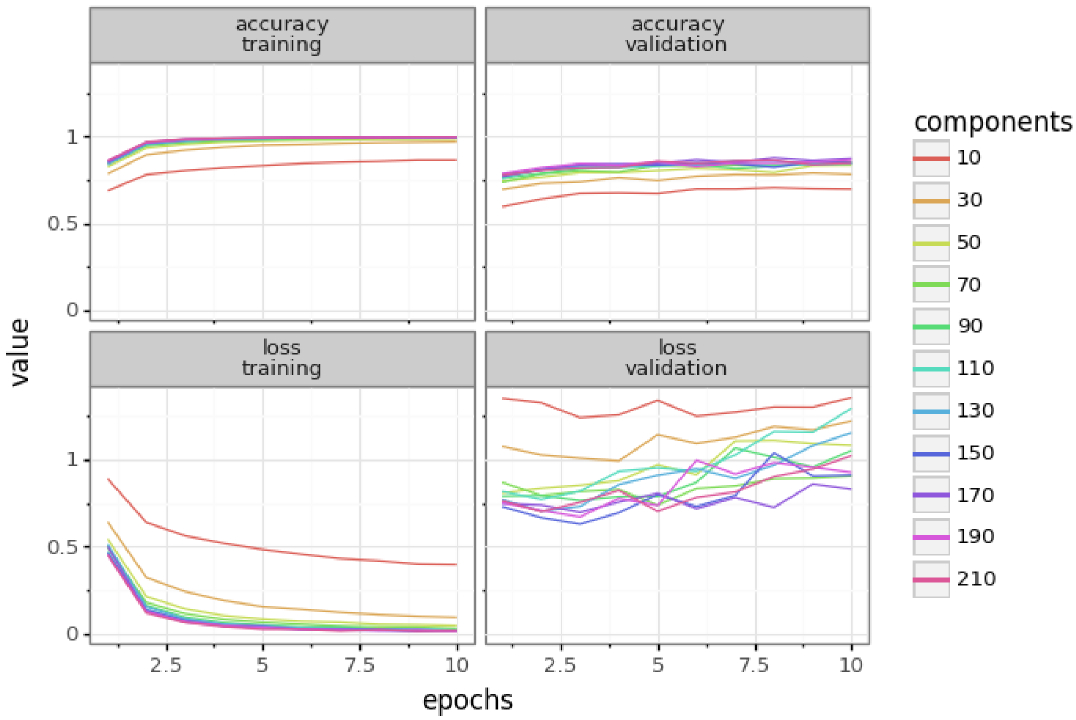
[ ]:
```
array(['accuracy', 'loss'], dtype=object)
```

[ ]:
```
ggplot(df, aes(x='epochs', y='value',color='components')) + \
    geom_line() + \
    facet_wrap(['accuracy','valid'],scales='free') +  theme_bw(base_size=12)
```



[ ]: <ggplot: (8758028328796)>

[ ]:
```
ggplot(df, aes(x='epochs', y='value',color='components')) + \
    geom_line() + \
    facet_wrap(['accuracy','valid']) + \
    theme_bw(base_size=12)
```



[ ]: <ggplot: (-9223363278826112789)>

4.2 Feed-Forward Zero-One Loss

[ ]:
```
df_loss = pd.DataFrame(
columns = ['components', 'zero_one']
)

losses =[]

for i in range(len(res)):
    losses.append(res[i][2])
components = [i for i in range(10,211,20)]
```

[ ]:
```
df_loss = pd.DataFrame( data = [components,losses], index = ['components',␣
↪'zero_one']).T
```

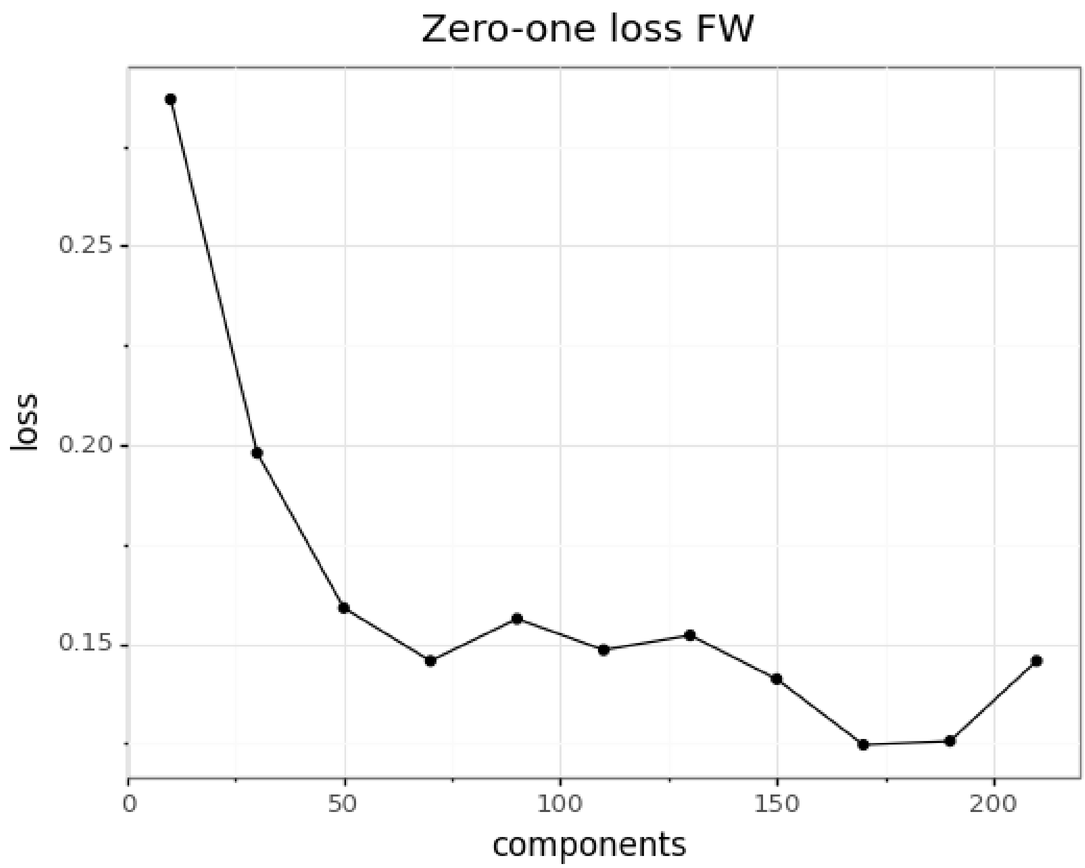
[ ]:
```
df_loss
```

[ ]:

	components	zero_one
0	10.0	0.286893
1	30.0	0.197984
2	50.0	0.159028

```
3          70.0  0.145738
4          90.0  0.156279
5         110.0  0.148488
6         130.0  0.152154
7         150.0  0.141155
8         170.0  0.124656
9         190.0  0.125573
10        210.0  0.145738
```

[ ]:
```
ggplot(df_loss, aes(x='components', y='zero_one')) + \
    geom_line() + \
    geom_point() + \
    theme_bw(base_size=12) + ggtitle("Zero-one loss FW") + ylab("loss")
```



[ ]: `<ggplot: (-9223363278721333202)>`

5 1.4 Convolutional Neural Newtworks

5.1 One VGG block CNN

[ ]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), padding = "same", activation='relu',␣
 ↪input_shape = (32, 32, 3)),
    tf.keras.layers.Conv2D(32, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation = 'relu'),
    tf.keras.layers.Dense(10, activation = 'softmax')
])

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

model.summary()
```

```
Model: "sequential_11"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense_22 (Dense)             (None, 128)               1048704
_________________________________________________________________
dense_23 (Dense)             (None, 10)                1290
=================================================================
Total params: 1,060,138
Trainable params: 1,060,138
Non-trainable params: 0
_________________________________________________________________
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
```

```
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense_22 (Dense)             (None, 128)               1048704
_________________________________________________________________
dense_23 (Dense)             (None, 10)                1290
=================================================================
Total params: 1,060,138
Trainable params: 1,060,138
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
history = model.fit(x_train,y_train,
        batch_size = 32,
        epochs=10,
        validation_data=(x_valid, y_valid),
        verbose=2
                   )
```

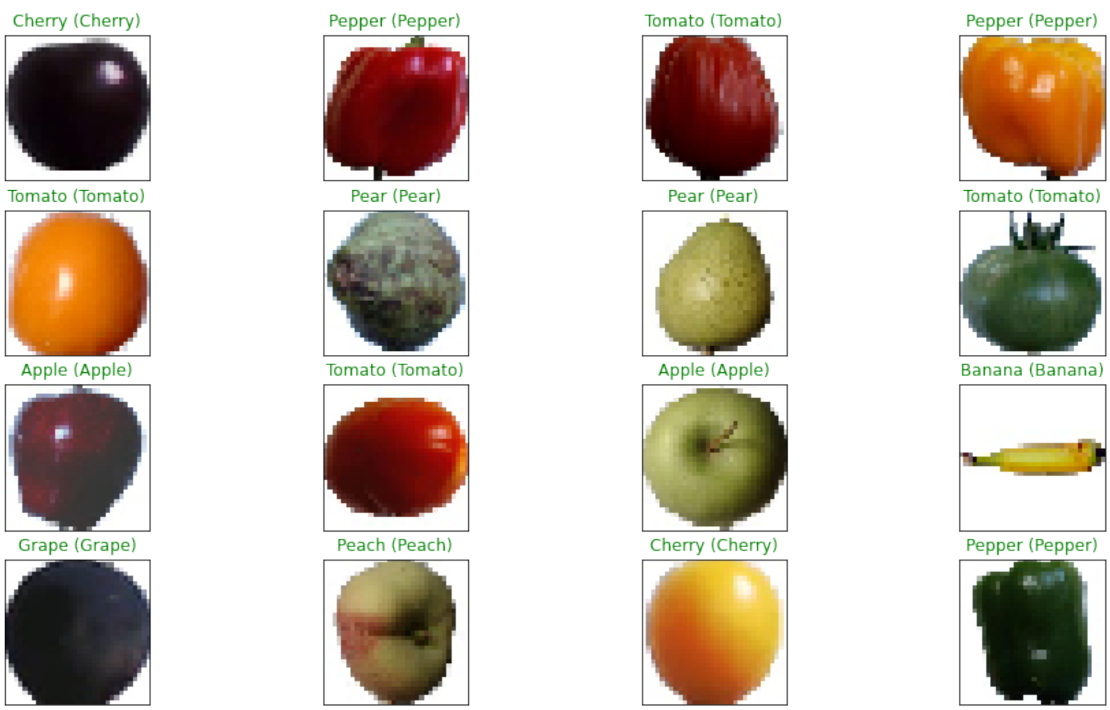
```
Epoch 1/10
1019/1019 - 12s - loss: 0.2258 - accuracy: 0.9251 - val_loss: 0.1457 -
val_accuracy: 0.9591
Epoch 2/10
1019/1019 - 11s - loss: 0.0122 - accuracy: 0.9964 - val_loss: 0.1096 -
val_accuracy: 0.9716
Epoch 3/10
1019/1019 - 11s - loss: 0.0083 - accuracy: 0.9981 - val_loss: 0.0620 -
val_accuracy: 0.9841
Epoch 4/10
1019/1019 - 11s - loss: 0.0058 - accuracy: 0.9989 - val_loss: 0.1105 -
val_accuracy: 0.9822
Epoch 5/10
1019/1019 - 11s - loss: 0.0071 - accuracy: 0.9986 - val_loss: 0.1179 -
val_accuracy: 0.9836
Epoch 6/10
1019/1019 - 11s - loss: 0.0080 - accuracy: 0.9989 - val_loss: 0.1146 -
val_accuracy: 0.9782
Epoch 7/10
1019/1019 - 11s - loss: 0.0039 - accuracy: 0.9994 - val_loss: 0.1395 -
val_accuracy: 0.9790
Epoch 8/10
1019/1019 - 11s - loss: 0.0020 - accuracy: 0.9995 - val_loss: 0.1240 -
val_accuracy: 0.9827
Epoch 9/10
1019/1019 - 11s - loss: 0.0045 - accuracy: 0.9995 - val_loss: 0.1455 -
val_accuracy: 0.9733
Epoch 10/10
1019/1019 - 11s - loss: 0.0029 - accuracy: 0.9994 - val_loss: 0.1433 -
```

```
val_accuracy: 0.9771
```

```
[ ]: y_pred = model.predict(x_test)

cnn_loss = []
zol = zero_one(y_pred, y_test)

print("Zero-one Loss: ", zol)
cnn_loss.append(zol)
```

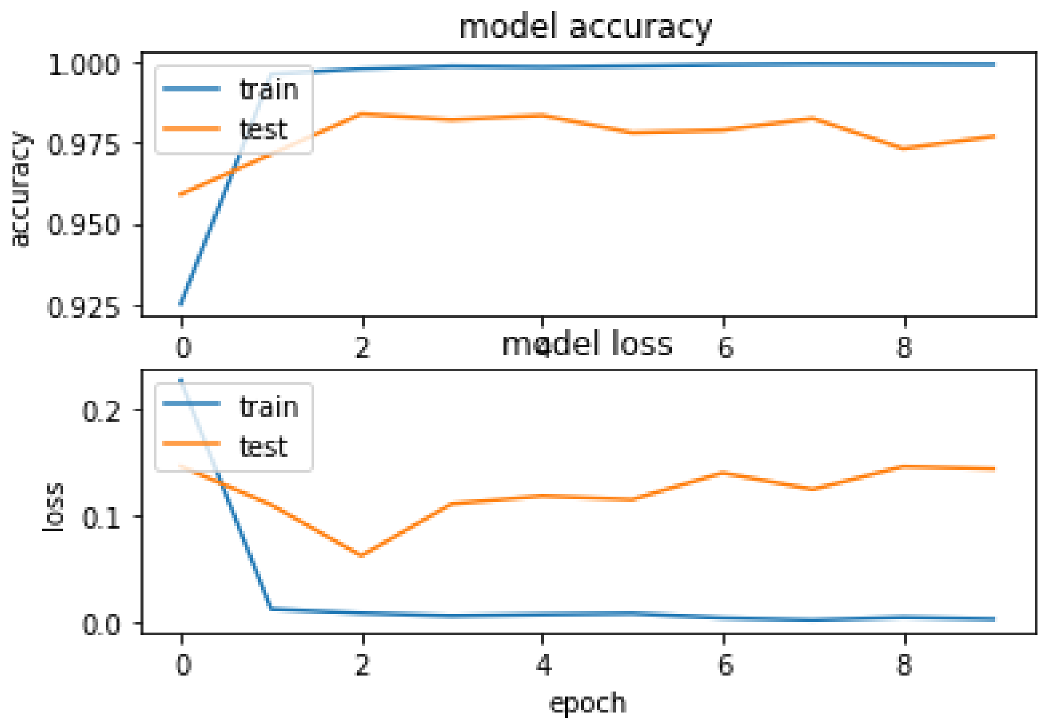
```
Zero-one Loss:  0.02153987167736022
```

```
[ ]: # plot a random sample of test images, their predicted labels, and ground truth
fig = plt.figure(figsize=(16, 9))
for i, idx in enumerate(np.random.choice(x_test.shape[0], size=16,␣
 ↪replace=False)):
    ax = fig.add_subplot(4, 4, i + 1, xticks=[], yticks=[])
    ax.imshow(np.squeeze(x_test[idx]))
    pred_idx = np.argmax(y_pred[idx])
    true_idx = np.argmax(y_test[idx])
    ax.set_title("{} ({})".format(TYPES[pred_idx], TYPES[true_idx]),
                 color=("green" if pred_idx == true_idx else "red"))
```



[ ]:

```
#Loss and accuracy visualisation

plt.figure(1)


plt.subplot(211)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')


plt.subplot(212)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



[ ]:

```

```

5.2 Two VGG blocks

[ ]:

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), padding = "same", activation='relu',␣
 ↪input_shape = (32, 32, 3)),
    tf.keras.layers.Conv2D(32, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.Conv2D(64, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation = 'relu'),
    tf.keras.layers.Dense(10, activation = 'softmax')
])

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

model.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_24 (Dense)             (None, 128)               524416
_________________________________________________________________
dense_25 (Dense)             (None, 10)                1290
=================================================================
Total params: 591,274
Trainable params: 591,274
Non-trainable params: 0
```

[ ]:
```
history2 = model.fit(x_train,y_train,
        batch_size = 32,
        epochs=10,
        validation_data=(x_valid, y_valid),
        verbose=2

                   )
```

```
Epoch 1/10
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.210605). Check your callbacks.
1019/1019 - 14s - loss: 0.2516 - accuracy: 0.9158 - val_loss: 0.1399 -
val_accuracy: 0.9559
Epoch 2/10
1019/1019 - 13s - loss: 0.0240 - accuracy: 0.9931 - val_loss: 0.0467 -
val_accuracy: 0.9812
Epoch 3/10
1019/1019 - 13s - loss: 0.0077 - accuracy: 0.9973 - val_loss: 0.0338 -
val_accuracy: 0.9900
Epoch 4/10
1019/1019 - 13s - loss: 3.4333e-05 - accuracy: 1.0000 - val_loss: 0.0311 -
val_accuracy: 0.9900
Epoch 5/10
1019/1019 - 13s - loss: 1.1592e-05 - accuracy: 1.0000 - val_loss: 0.0367 -
val_accuracy: 0.9900
Epoch 6/10
1019/1019 - 13s - loss: 5.8665e-06 - accuracy: 1.0000 - val_loss: 0.0342 -
val_accuracy: 0.9903
Epoch 7/10
1019/1019 - 13s - loss: 2.9794e-06 - accuracy: 1.0000 - val_loss: 0.0304 -
val_accuracy: 0.9903
Epoch 8/10
1019/1019 - 13s - loss: 1.6845e-06 - accuracy: 1.0000 - val_loss: 0.0328 -
val_accuracy: 0.9906
Epoch 9/10
1019/1019 - 13s - loss: 9.3372e-07 - accuracy: 1.0000 - val_loss: 0.0308 -
val_accuracy: 0.9905
Epoch 10/10
1019/1019 - 14s - loss: 5.2326e-07 - accuracy: 1.0000 - val_loss: 0.0337 -
val_accuracy: 0.9907
```

[ ]:
```
# evaluate zero-one loss
y_pred = model.predict(x_test)
zol = zero_one(y_pred, y_test)
```

```
print("Zero-one Loss: ", zol)

cnn_loss.append(zol)
```

```
Zero-one Loss:  0.00916590284142988
```

5.3 Three VGG blocks

[ ]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), padding = "same", activation='relu',␣
 ↪input_shape = (32, 32, 3)),
    tf.keras.layers.Conv2D(32, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.Conv2D(64, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Conv2D(128, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.Conv2D(128, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation = 'relu'),
    tf.keras.layers.Dense(10, activation = 'softmax')
])

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

model.summary()
```

```
Model: "sequential_13"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
```

```
conv2d_11 (Conv2D)           (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_26 (Dense)             (None, 128)               262272
_________________________________________________________________
dense_27 (Dense)             (None, 10)                1290
=================================================================
Total params: 550,570
Trainable params: 550,570
Non-trainable params: 0
_________________________________________________________________
```

[ ]:

```
history3 = model.fit(x_train,y_train,
        batch_size = 32,
        epochs=10,
        validation_data=(x_valid, y_valid),
        verbose=2

                   )
```

```
Epoch 1/10
1019/1019 - 19s - loss: 0.3207 - accuracy: 0.8886 - val_loss: 0.0736 -
val_accuracy: 0.9760
Epoch 2/10
1019/1019 - 18s - loss: 0.0340 - accuracy: 0.9895 - val_loss: 0.2608 -
val_accuracy: 0.9379
Epoch 3/10
1019/1019 - 18s - loss: 0.0234 - accuracy: 0.9932 - val_loss: 0.0518 -
val_accuracy: 0.9861
Epoch 4/10
1019/1019 - 18s - loss: 8.6958e-05 - accuracy: 1.0000 - val_loss: 0.0193 -
val_accuracy: 0.9917
Epoch 5/10
1019/1019 - 18s - loss: 9.9077e-06 - accuracy: 1.0000 - val_loss: 0.0171 -
val_accuracy: 0.9928
Epoch 6/10
1019/1019 - 18s - loss: 4.4342e-06 - accuracy: 1.0000 - val_loss: 0.0164 -
val_accuracy: 0.9925
Epoch 7/10
1019/1019 - 18s - loss: 2.2970e-06 - accuracy: 1.0000 - val_loss: 0.0168 -
val_accuracy: 0.9925
Epoch 8/10
1019/1019 - 18s - loss: 1.2628e-06 - accuracy: 1.0000 - val_loss: 0.0164 -
val_accuracy: 0.9932
Epoch 9/10
```

```
1019/1019 - 18s - loss: 6.9922e-07 - accuracy: 1.0000 - val_loss: 0.0156 -
val_accuracy: 0.9940
Epoch 10/10
1019/1019 - 18s - loss: 3.8787e-07 - accuracy: 1.0000 - val_loss: 0.0166 -
val_accuracy: 0.9938
```

[ ]:
```
# evaluate zero-one loss
y_pred = model.predict(x_test)

zol = zero_one(y_pred, y_test)

print("Zero-one Loss: ", zol)

cnn_loss.append(zol)
```

```
Zero-one Loss:  0.00916590284142988
```

5.4 Three VGG blocks with Dropout

[ ]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), padding = "same", activation='relu',␣
 ↪input_shape = (32, 32, 3)),
    tf.keras.layers.Conv2D(32, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(64, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.Conv2D(64, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(128, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.Conv2D(128, (3, 3), padding = "same", activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation = 'relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation = 'softmax')
])

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


model.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
conv2d_12 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_28 (Dense)             (None, 128)               262272
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_29 (Dense)             (None, 10)                1290
=================================================================
Total params: 550,570
Trainable params: 550,570
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```python
history4 = model.fit(x_train,y_train,
        batch_size = 32,
        epochs=10,
        validation_data=(x_valid, y_valid),
        verbose=2
                   )
```

```
Epoch 1/10
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.151422). Check your callbacks.
```

```
1019/1019 - 20s - loss: 0.4977 - accuracy: 0.8243 - val_loss: 0.1280 -
val_accuracy: 0.9546
Epoch 2/10
1019/1019 - 19s - loss: 0.0688 - accuracy: 0.9775 - val_loss: 0.1614 -
val_accuracy: 0.9452
Epoch 3/10
1019/1019 - 19s - loss: 0.0458 - accuracy: 0.9857 - val_loss: 0.0644 -
val_accuracy: 0.9795
Epoch 4/10
1019/1019 - 19s - loss: 0.0364 - accuracy: 0.9886 - val_loss: 0.0873 -
val_accuracy: 0.9727
Epoch 5/10
1019/1019 - 19s - loss: 0.0243 - accuracy: 0.9931 - val_loss: 0.1079 -
val_accuracy: 0.9783
Epoch 6/10
1019/1019 - 19s - loss: 0.0328 - accuracy: 0.9902 - val_loss: 0.0799 -
val_accuracy: 0.9782
Epoch 7/10
1019/1019 - 20s - loss: 0.0262 - accuracy: 0.9922 - val_loss: 0.2361 -
val_accuracy: 0.9506
Epoch 8/10
1019/1019 - 19s - loss: 0.0189 - accuracy: 0.9949 - val_loss: 0.1377 -
val_accuracy: 0.9713
Epoch 9/10
1019/1019 - 19s - loss: 0.0246 - accuracy: 0.9928 - val_loss: 0.0759 -
val_accuracy: 0.9793
Epoch 10/10
1019/1019 - 19s - loss: 0.0190 - accuracy: 0.9952 - val_loss: 0.3236 -
val_accuracy: 0.9211
```

[ ]:
```
# evaluate zero-one loss
y_pred = model.predict(x_test)
zol = zero_one(y_pred, y_test)

print("Zero-one Loss: ", zol)

cnn_loss.append(zol)
```

```
Zero-one Loss:  0.07561869844179651
```

5.5 Three VGG blocks with Dropout and Batch Normalization

[ ]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), padding = "same", use_bias=False,␣
 ↪input_shape = (32, 32, 3)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
```

```python
    tf.keras.layers.Conv2D(32, (3, 3), padding = "same", use_bias=False),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(64, (3, 3), padding = "same", use_bias=False),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.Conv2D(64, (3, 3), padding = "same", use_bias=False),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Conv2D(128, (3, 3), padding = "same", use_bias=False),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.Conv2D(128, (3, 3), padding = "same", use_bias=False),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, use_bias=False),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(10, activation = 'softmax')
])

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

model.summary()
```

```
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 32, 32, 32)        864
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 32)        128
_________________________________________________________________
activation (Activation)      (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 32, 32, 32)        9216
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128
```

```
_________________________________________________________________
activation_1 (Activation)    (None, 32, 32, 32)        0
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 16, 16, 64)        18432
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
activation_2 (Activation)    (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 16, 16, 64)        36864
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
activation_3 (Activation)    (None, 16, 16, 64)        0
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 8, 8, 128)         73728
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
activation_4 (Activation)    (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 8, 8, 128)         147456
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
activation_5 (Activation)    (None, 8, 8, 128)         0
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 4, 4, 128)         0
_________________________________________________________________
dropout_6 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_30 (Dense)             (None, 128)               262144
_________________________________________________________________
batch_normalization_6 (Batch (None, 128)               512
_________________________________________________________________
activation_6 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_7 (Dropout)          (None, 128)               0
```

```
_________________________________________________________________
dense_31 (Dense)             (None, 10)                1290
=================================================================
Total params: 552,298
Trainable params: 551,146
Non-trainable params: 1,152
_________________________________________________________________
```

[ ]:

```python

```

[ ]:

```python
history5 = model.fit(x_train,y_train,
        batch_size = 32,
        epochs=10,
        validation_data=(x_valid, y_valid),
        verbose=2
                   )
```

```
Epoch 1/10
1019/1019 - 22s - loss: 0.3911 - accuracy: 0.8740 - val_loss: 0.1592 -
val_accuracy: 0.9462
Epoch 2/10
1019/1019 - 21s - loss: 0.0870 - accuracy: 0.9735 - val_loss: 0.0608 -
val_accuracy: 0.9725
Epoch 3/10
1019/1019 - 21s - loss: 0.0421 - accuracy: 0.9875 - val_loss: 0.0188 -
val_accuracy: 0.9944
Epoch 4/10
1019/1019 - 21s - loss: 0.0428 - accuracy: 0.9876 - val_loss: 0.0137 -
val_accuracy: 0.9958
Epoch 5/10
1019/1019 - 21s - loss: 0.0259 - accuracy: 0.9927 - val_loss: 0.0061 -
val_accuracy: 0.9978
Epoch 6/10
1019/1019 - 22s - loss: 0.0252 - accuracy: 0.9924 - val_loss: 0.0168 -
val_accuracy: 0.9929
Epoch 7/10
1019/1019 - 21s - loss: 0.0234 - accuracy: 0.9928 - val_loss: 0.0430 -
val_accuracy: 0.9826
Epoch 8/10
1019/1019 - 21s - loss: 0.0239 - accuracy: 0.9925 - val_loss: 0.0387 -
val_accuracy: 0.9829
Epoch 9/10
1019/1019 - 21s - loss: 0.0169 - accuracy: 0.9948 - val_loss: 0.0207 -
val_accuracy: 0.9916
Epoch 10/10
1019/1019 - 21s - loss: 0.0140 - accuracy: 0.9956 - val_loss: 0.0163 -
val_accuracy: 0.9936
```

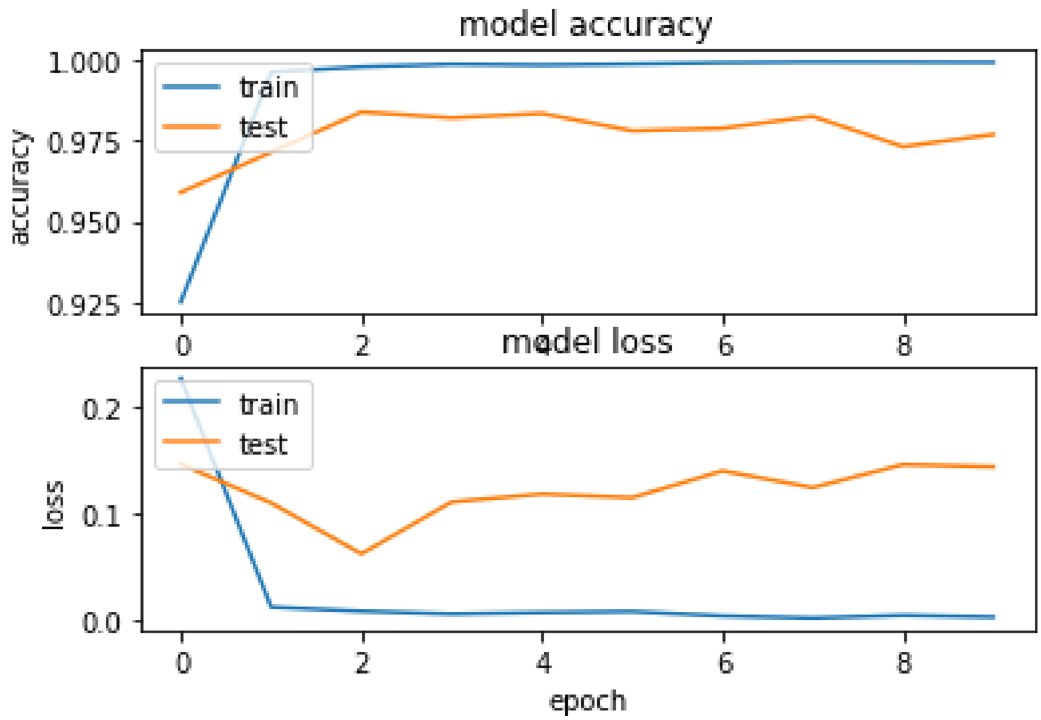
[ ]:
```python
# evaluate zero-one loss
y_pred = model.predict(x_test)
zol = zero_one(y_pred, y_test)

print("Zero-one Loss: ", zol)

cnn_loss.append(zol)
```

```
Zero-one Loss:  0.005957836846929423
```

5.6 VGG CNN results

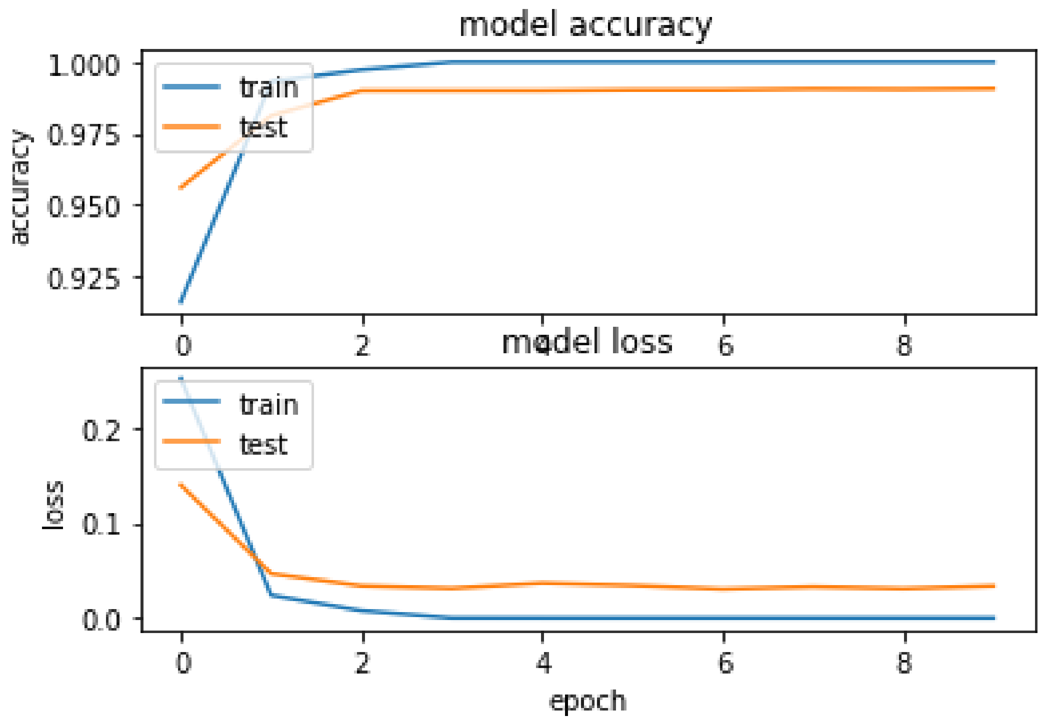
One VGG block

[ ]:
```python
plt.figure(1)



plt.subplot(211)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')


plt.subplot(212)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



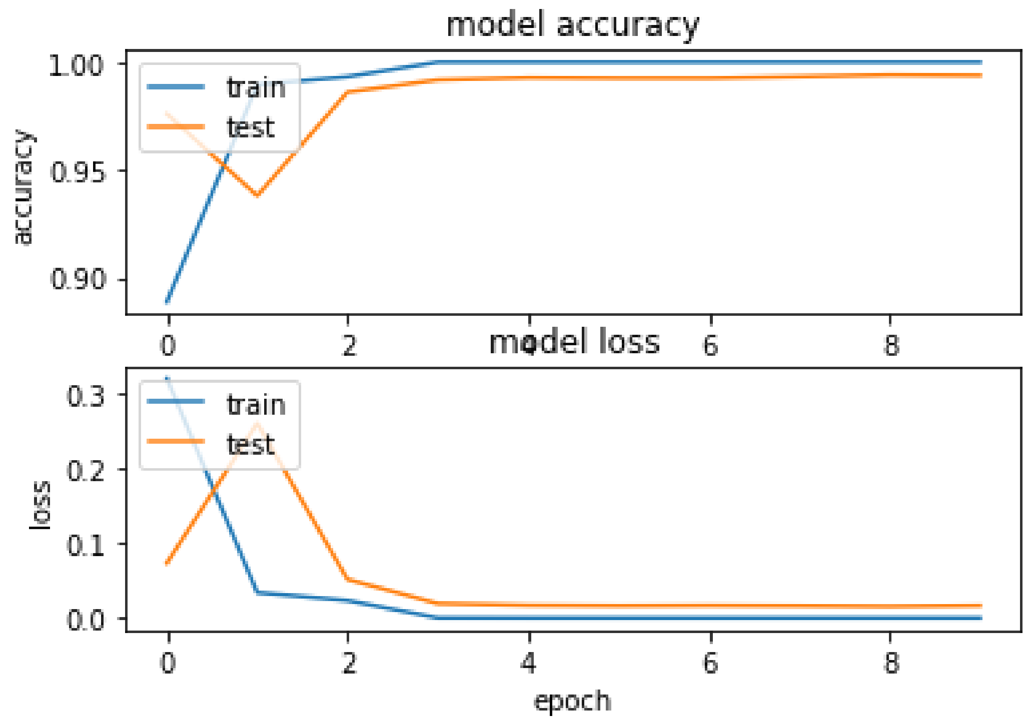
Two VGG block

[ ]:
```
plt.figure(1)


plt.subplot(211)
plt.plot(history2.history['accuracy'])
plt.plot(history2.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')


plt.subplot(212)
plt.plot(history2.history['loss'])
plt.plot(history2.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



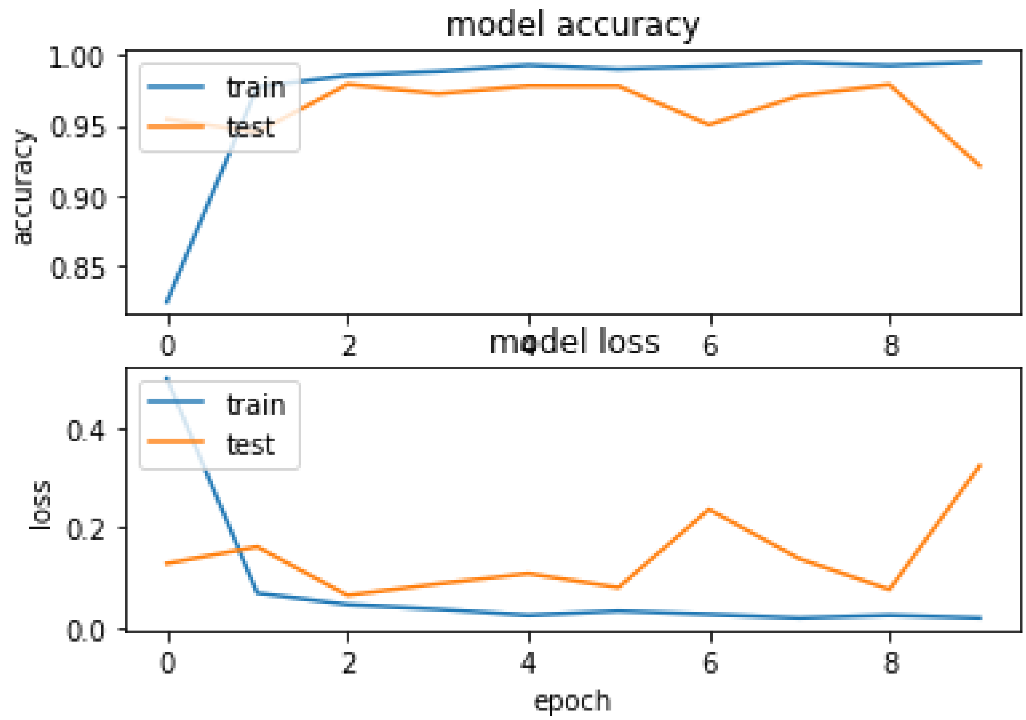
Three VGG block

[ ]:
```
plt.figure(1)


plt.subplot(211)
plt.plot(history3.history['accuracy'])
plt.plot(history3.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')


plt.subplot(212)
plt.plot(history3.history['loss'])
plt.plot(history3.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



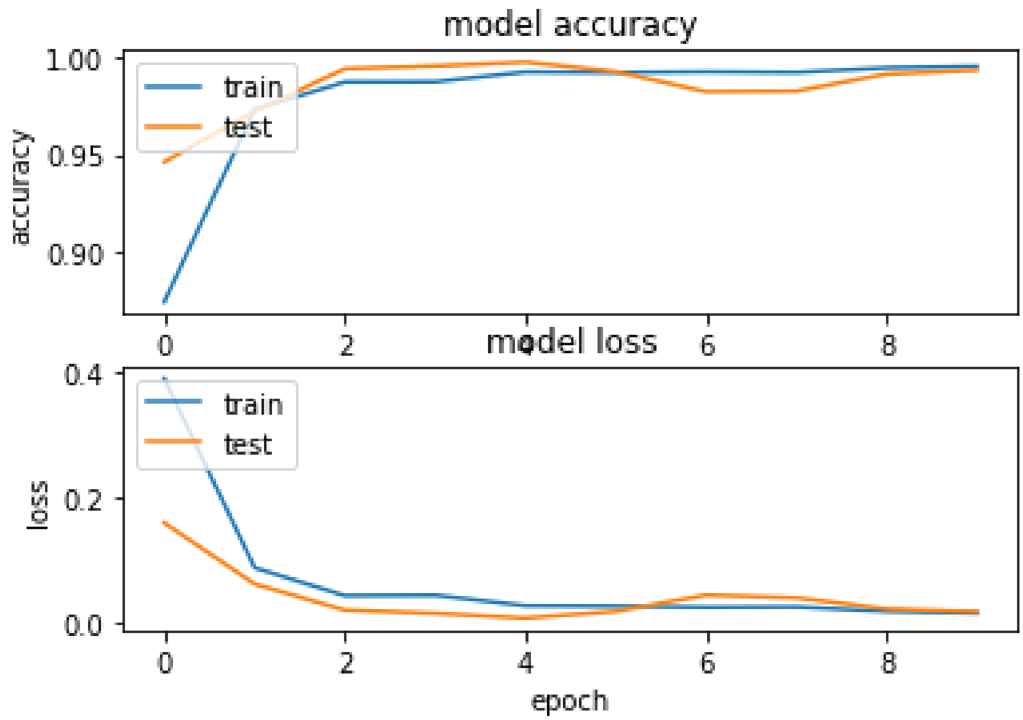
Three VGG block with Dropout

[ ]:
```
plt.figure(1)


plt.subplot(211)
plt.plot(history4.history['accuracy'])
plt.plot(history4.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')


plt.subplot(212)
plt.plot(history4.history['loss'])
plt.plot(history4.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



Three VGG block with Dropout and Batch Normalization

[ ]:
```
plt.figure(1)


plt.subplot(211)
plt.plot(history5.history['accuracy'])
plt.plot(history5.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')


plt.subplot(212)
plt.plot(history5.history['loss'])
plt.plot(history5.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



5.7 VGG Zero-One Loss

[ ]:
```
cnn_loss
```

[ ]:
```
[0.02153987167736022,
 0.00916590284142988,
 0.00916590284142988,
 0.07561869844179651,
 0.005957836846929423]
```

[ ]:
```
df_cnn_loss = pd.DataFrame(
columns = ['type', 'zero_one']
)

types = ["1VGG","2VGG","3VGG","3VGG-drop", "3VGG-drop-norm"]
df_cnn_loss = pd.DataFrame( data = [types,cnn_loss], index = ['types',␣
 ↪'zero_one']).T
```

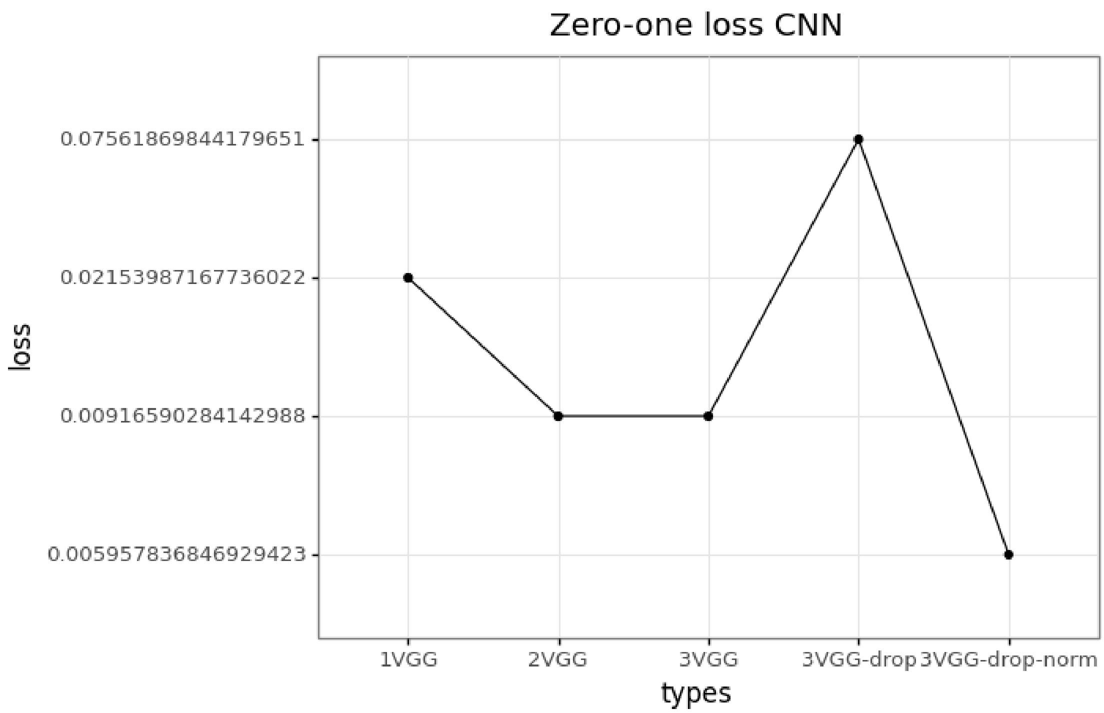
[ ]:
```
df_cnn_loss
```

[ ]:

	types	zero_one
0	1VGG	0.0215399
1	2VGG	0.0091659

```
2           3VGG    0.0091659
3       3VGG-drop    0.0756187
4  3VGG-drop-norm   0.00595784
```

[ ]:
```
ggplot(df_cnn_loss, aes(x='types', y='zero_one',group=1)) + \
    geom_point() + \
    geom_line() + \
    theme_bw(base_size=12) + ggtitle("Zero-one loss CNN") + ylab("loss")
```



[ ]: <ggplot: (8757990059065)>

6 1.5 LeNet Neural Networks

[ ]:
```
input_shape = (32,32,3)
num_classes = 10
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(6, kernel_size=(5, 5), strides=(1, 1),␣
 ↪activation='tanh', input_shape=input_shape, padding="same"),
    tf.keras.layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2),␣
 ↪padding='valid'),
    tf.keras.layers.Conv2D(16, kernel_size=(5, 5), strides=(1, 1),␣
 ↪activation='tanh', padding='valid'),
```

```
    tf.keras.layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2),␣
 ↪padding='valid'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(120, activation = 'tanh'),
    tf.keras.layers.Dense(84, activation = 'tanh'),
    tf.keras.layers.Dense(10, activation = 'softmax')
])

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

model.summary()
```

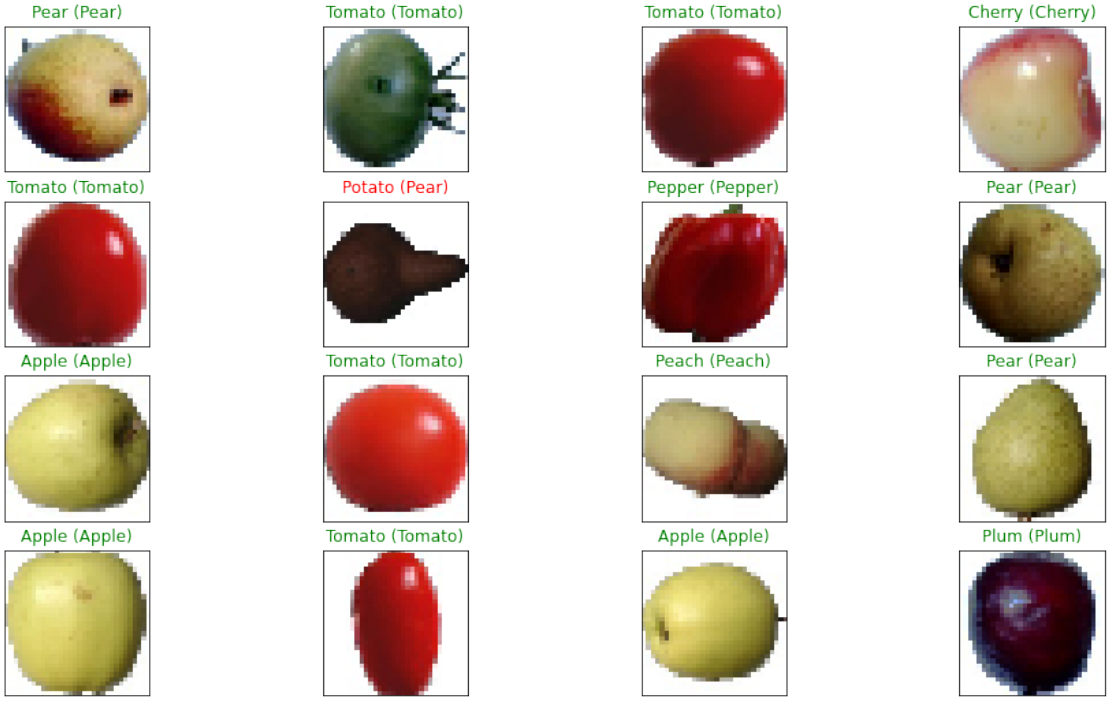
```
Model: "sequential_16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_24 (Conv2D)           (None, 32, 32, 6)         456
_________________________________________________________________
average_pooling2d (AveragePo (None, 16, 16, 6)         0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 12, 12, 16)        2416
_________________________________________________________________
average_pooling2d_1 (Average (None, 6, 6, 16)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_32 (Dense)             (None, 120)               69240
_________________________________________________________________
dense_33 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_34 (Dense)             (None, 10)                850
=================================================================
Total params: 83,126
Trainable params: 83,126
Non-trainable params: 0

_________________________________________________________________
```

[ ]:
```
history = model.fit(x_train, y_train,
                    batch_size = 32,
                    epochs = 10,
                    validation_data=(x_valid, y_valid),
                    verbose = 2

                    )
```

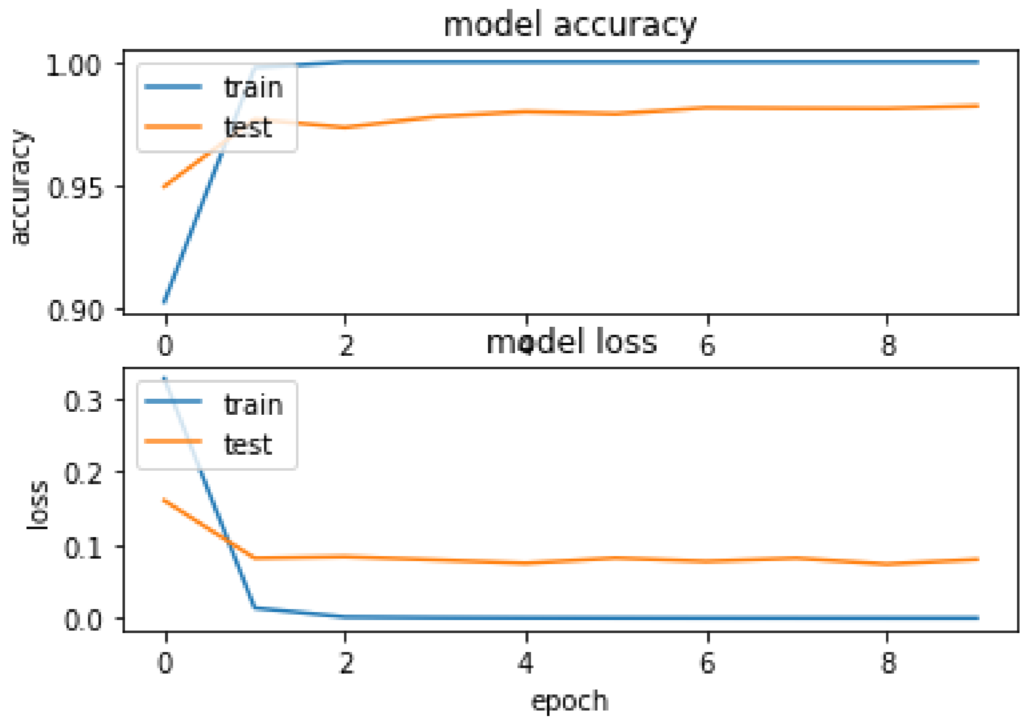
```
Epoch 1/10
```

```
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.219765). Check your callbacks.
1019/1019 - 8s - loss: 0.3274 - accuracy: 0.9028 - val_loss: 0.1608 -
val_accuracy: 0.9496
Epoch 2/10
1019/1019 - 7s - loss: 0.0138 - accuracy: 0.9977 - val_loss: 0.0819 -
val_accuracy: 0.9768
Epoch 3/10
1019/1019 - 9s - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0839 -
val_accuracy: 0.9735
Epoch 4/10
1019/1019 - 7s - loss: 6.4137e-04 - accuracy: 1.0000 - val_loss: 0.0796 -
val_accuracy: 0.9779
Epoch 5/10
1019/1019 - 7s - loss: 3.2755e-04 - accuracy: 1.0000 - val_loss: 0.0749 -
val_accuracy: 0.9799
Epoch 6/10
1019/1019 - 7s - loss: 1.7176e-04 - accuracy: 1.0000 - val_loss: 0.0822 -
val_accuracy: 0.9790
Epoch 7/10
1019/1019 - 7s - loss: 9.9440e-05 - accuracy: 1.0000 - val_loss: 0.0775 -
val_accuracy: 0.9815
Epoch 8/10
1019/1019 - 8s - loss: 5.2685e-05 - accuracy: 1.0000 - val_loss: 0.0819 -
val_accuracy: 0.9813
Epoch 9/10
1019/1019 - 7s - loss: 2.9746e-05 - accuracy: 1.0000 - val_loss: 0.0737 -
val_accuracy: 0.9812
Epoch 10/10
1019/1019 - 7s - loss: 1.6986e-05 - accuracy: 1.0000 - val_loss: 0.0801 -
val_accuracy: 0.9822
```

[ ]:
```
y_pred = model.predict(x_test)

# plot a random sample of test images, their predicted labels, and ground truth
fig = plt.figure(figsize=(16, 9))
for i, idx in enumerate(np.random.choice(x_test.shape[0], size=16,
 ↪replace=False)):
    ax = fig.add_subplot(4, 4, i + 1, xticks=[], yticks=[])
    ax.imshow(np.squeeze(x_test[idx]))
    pred_idx = np.argmax(y_pred[idx])
    true_idx = np.argmax(y_test[idx])
    ax.set_title("{} ({})".format(TYPES[pred_idx], TYPES[true_idx]),
                 color=("green" if pred_idx == true_idx else "red"))
```



[ ]:
```
plt.figure(1)


plt.subplot(211)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')


plt.subplot(212)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



[ ]:
```
y_pred = model.predict(x_test)
zol = zero_one(y_pred, y_test)

print("Zero-one Loss: ", zol)

cnn_loss.append(zol)
```

```
Zero-one Loss:  0.016498625114573784
```

7 1.6 MobileNetV2

[ ]:
```
model = MobileNetV2(input_shape=(32, 32, 3), alpha=1, weights=None,classes=10)
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
print(model.summary())
print('Compiled!')
```

```
Model: "mobilenetv2_1.00_32"

--------------------------------------------------------------------------------
------------------
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
```

```
==================
input_12 (InputLayer)          (None, 32, 32, 3)    0

--------------------------------------------------------------------------------
------------------
Conv1_pad (ZeroPadding2D)      (None, 33, 33, 3)    0           input_12[0][0]

--------------------------------------------------------------------------------
------------------
Conv1 (Conv2D)                 (None, 16, 16, 32)   864         Conv1_pad[0][0]

--------------------------------------------------------------------------------
------------------
bn_Conv1 (BatchNormalization)  (None, 16, 16, 32)   128         Conv1[0][0]

--------------------------------------------------------------------------------
------------------
Conv1_relu (ReLU)              (None, 16, 16, 32)   0           bn_Conv1[0][0]

--------------------------------------------------------------------------------
------------------
expanded_conv_depthwise (Depthw (None, 16, 16, 32)  288
Conv1_relu[0][0]

--------------------------------------------------------------------------------
------------------
expanded_conv_depthwise_BN (Bat (None, 16, 16, 32)  128
expanded_conv_depthwise[0][0]

--------------------------------------------------------------------------------
------------------
expanded_conv_depthwise_relu (R (None, 16, 16, 32)  0
expanded_conv_depthwise_BN[0][0]

--------------------------------------------------------------------------------
------------------
expanded_conv_project (Conv2D) (None, 16, 16, 16)   512
expanded_conv_depthwise_relu[0][0

--------------------------------------------------------------------------------
------------------
expanded_conv_project_BN (Batch (None, 16, 16, 16)  64
expanded_conv_project[0][0]

--------------------------------------------------------------------------------
------------------
block_1_expand (Conv2D)        (None, 16, 16, 96)   1536
expanded_conv_project_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_1_expand_BN (BatchNormali (None, 16, 16, 96)  384
block_1_expand[0][0]

--------------------------------------------------------------------------------
------------------
block_1_expand_relu (ReLU)     (None, 16, 16, 96)   0
block_1_expand_BN[0][0]

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

```
block_1_pad (ZeroPadding2D)     (None, 17, 17, 96)   0
block_1_expand_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_1_depthwise (DepthwiseCon (None, 8, 8, 96)     864
block_1_pad[0][0]

--------------------------------------------------------------------------------
------------------
block_1_depthwise_BN (BatchNorm (None, 8, 8, 96)     384
block_1_depthwise[0][0]

--------------------------------------------------------------------------------
------------------
block_1_depthwise_relu (ReLU)   (None, 8, 8, 96)     0
block_1_depthwise_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_1_project (Conv2D)        (None, 8, 8, 24)     2304
block_1_depthwise_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_1_project_BN (BatchNormal (None, 8, 8, 24)     96
block_1_project[0][0]

--------------------------------------------------------------------------------
------------------
block_2_expand (Conv2D)         (None, 8, 8, 144)    3456
block_1_project_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_2_expand_BN (BatchNormali (None, 8, 8, 144)    576
block_2_expand[0][0]

--------------------------------------------------------------------------------
------------------
block_2_expand_relu (ReLU)      (None, 8, 8, 144)    0
block_2_expand_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_2_depthwise (DepthwiseCon (None, 8, 8, 144)    1296
block_2_expand_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_2_depthwise_BN (BatchNorm (None, 8, 8, 144)    576
block_2_depthwise[0][0]

--------------------------------------------------------------------------------
------------------
block_2_depthwise_relu (ReLU)   (None, 8, 8, 144)    0
block_2_depthwise_BN[0][0]

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

```
block_2_project (Conv2D)        (None, 8, 8, 24)     3456
block_2_depthwise_relu[0][0]

__________________________________________________________________________________________________
block_2_project_BN (BatchNormal (None, 8, 8, 24)     96
block_2_project[0][0]

__________________________________________________________________________________________________
block_2_add (Add)               (None, 8, 8, 24)     0
block_1_project_BN[0][0]
block_2_project_BN[0][0]

__________________________________________________________________________________________________
block_3_expand (Conv2D)         (None, 8, 8, 144)    3456
block_2_add[0][0]

__________________________________________________________________________________________________
block_3_expand_BN (BatchNormali (None, 8, 8, 144)    576
block_3_expand[0][0]

__________________________________________________________________________________________________
block_3_expand_relu (ReLU)      (None, 8, 8, 144)    0
block_3_expand_BN[0][0]

__________________________________________________________________________________________________
block_3_pad (ZeroPadding2D)     (None, 9, 9, 144)    0
block_3_expand_relu[0][0]

__________________________________________________________________________________________________
block_3_depthwise (DepthwiseCon (None, 4, 4, 144)    1296
block_3_pad[0][0]

__________________________________________________________________________________________________
block_3_depthwise_BN (BatchNorm (None, 4, 4, 144)    576
block_3_depthwise[0][0]

__________________________________________________________________________________________________
block_3_depthwise_relu (ReLU)   (None, 4, 4, 144)    0
block_3_depthwise_BN[0][0]

__________________________________________________________________________________________________
block_3_project (Conv2D)        (None, 4, 4, 32)     4608
block_3_depthwise_relu[0][0]

__________________________________________________________________________________________________
block_3_project_BN (BatchNormal (None, 4, 4, 32)     128
block_3_project[0][0]

__________________________________________________________________________________________________
```

```
__________________
block_4_expand (Conv2D)         (None, 4, 4, 192)    6144
block_3_project_BN[0][0]
________________________________________________________________________________
__________________
block_4_expand_BN (BatchNormali (None, 4, 4, 192)    768
block_4_expand[0][0]
________________________________________________________________________________
__________________
block_4_expand_relu (ReLU)      (None, 4, 4, 192)    0
block_4_expand_BN[0][0]
________________________________________________________________________________
__________________
block_4_depthwise (DepthwiseCon (None, 4, 4, 192)    1728
block_4_expand_relu[0][0]
________________________________________________________________________________
__________________
block_4_depthwise_BN (BatchNorm (None, 4, 4, 192)    768
block_4_depthwise[0][0]
________________________________________________________________________________
__________________
block_4_depthwise_relu (ReLU)   (None, 4, 4, 192)    0
block_4_depthwise_BN[0][0]
________________________________________________________________________________
__________________
block_4_project (Conv2D)        (None, 4, 4, 32)     6144
block_4_depthwise_relu[0][0]
________________________________________________________________________________
__________________
block_4_project_BN (BatchNormal (None, 4, 4, 32)     128
block_4_project[0][0]
________________________________________________________________________________
__________________
block_4_add (Add)               (None, 4, 4, 32)     0
block_3_project_BN[0][0]
block_4_project_BN[0][0]
________________________________________________________________________________
__________________
block_5_expand (Conv2D)         (None, 4, 4, 192)    6144
block_4_add[0][0]
________________________________________________________________________________
__________________
block_5_expand_BN (BatchNormali (None, 4, 4, 192)    768
block_5_expand[0][0]
________________________________________________________________________________
__________________
block_5_expand_relu (ReLU)      (None, 4, 4, 192)    0
block_5_expand_BN[0][0]
```

```
__________________________________________________________________________________
__________________
block_5_depthwise (DepthwiseCon (None, 4, 4, 192)    1728
block_5_expand_relu[0][0]
__________________________________________________________________________________
__________________
block_5_depthwise_BN (BatchNorm (None, 4, 4, 192)    768
block_5_depthwise[0][0]
__________________________________________________________________________________
__________________
block_5_depthwise_relu (ReLU)   (None, 4, 4, 192)    0
block_5_depthwise_BN[0][0]
__________________________________________________________________________________
__________________
block_5_project (Conv2D)        (None, 4, 4, 32)     6144
block_5_depthwise_relu[0][0]
__________________________________________________________________________________
__________________
block_5_project_BN (BatchNormal (None, 4, 4, 32)     128
block_5_project[0][0]
__________________________________________________________________________________
__________________
block_5_add (Add)               (None, 4, 4, 32)     0
block_4_add[0][0]
block_5_project_BN[0][0]
__________________________________________________________________________________
__________________
block_6_expand (Conv2D)         (None, 4, 4, 192)    6144
block_5_add[0][0]
__________________________________________________________________________________
__________________
block_6_expand_BN (BatchNormali (None, 4, 4, 192)    768
block_6_expand[0][0]
__________________________________________________________________________________
__________________
block_6_expand_relu (ReLU)      (None, 4, 4, 192)    0
block_6_expand_BN[0][0]
__________________________________________________________________________________
__________________
block_6_pad (ZeroPadding2D)     (None, 5, 5, 192)    0
block_6_expand_relu[0][0]
__________________________________________________________________________________
__________________
block_6_depthwise (DepthwiseCon (None, 2, 2, 192)    1728
block_6_pad[0][0]
__________________________________________________________________________________
__________________
block_6_depthwise_BN (BatchNorm (None, 2, 2, 192)    768
```

```
block_6_depthwise[0][0]

--------------------------------------------------------------------------------
------------------
block_6_depthwise_relu (ReLU)   (None, 2, 2, 192)    0
block_6_depthwise_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_6_project (Conv2D)        (None, 2, 2, 64)     12288
block_6_depthwise_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_6_project_BN (BatchNormal (None, 2, 2, 64)     256
block_6_project[0][0]

--------------------------------------------------------------------------------
------------------
block_7_expand (Conv2D)         (None, 2, 2, 384)    24576
block_6_project_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_7_expand_BN (BatchNormali (None, 2, 2, 384)    1536
block_7_expand[0][0]

--------------------------------------------------------------------------------
------------------
block_7_expand_relu (ReLU)      (None, 2, 2, 384)    0
block_7_expand_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_7_depthwise (DepthwiseCon (None, 2, 2, 384)    3456
block_7_expand_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_7_depthwise_BN (BatchNorm (None, 2, 2, 384)    1536
block_7_depthwise[0][0]

--------------------------------------------------------------------------------
------------------
block_7_depthwise_relu (ReLU)   (None, 2, 2, 384)    0
block_7_depthwise_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_7_project (Conv2D)        (None, 2, 2, 64)     24576
block_7_depthwise_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_7_project_BN (BatchNormal (None, 2, 2, 64)     256
block_7_project[0][0]

--------------------------------------------------------------------------------
------------------
block_7_add (Add)               (None, 2, 2, 64)     0
```

```
block_6_project_BN[0][0]
block_7_project_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_8_expand (Conv2D)         (None, 2, 2, 384)    24576
block_7_add[0][0]
--------------------------------------------------------------------------------
------------------
block_8_expand_BN (BatchNormali (None, 2, 2, 384)    1536
block_8_expand[0][0]
--------------------------------------------------------------------------------
------------------
block_8_expand_relu (ReLU)      (None, 2, 2, 384)    0
block_8_expand_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_8_depthwise (DepthwiseCon (None, 2, 2, 384)    3456
block_8_expand_relu[0][0]
--------------------------------------------------------------------------------
------------------
block_8_depthwise_BN (BatchNorm (None, 2, 2, 384)    1536
block_8_depthwise[0][0]
--------------------------------------------------------------------------------
------------------
block_8_depthwise_relu (ReLU)   (None, 2, 2, 384)    0
block_8_depthwise_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_8_project (Conv2D)        (None, 2, 2, 64)     24576
block_8_depthwise_relu[0][0]
--------------------------------------------------------------------------------
------------------
block_8_project_BN (BatchNormal (None, 2, 2, 64)     256
block_8_project[0][0]
--------------------------------------------------------------------------------
------------------
block_8_add (Add)               (None, 2, 2, 64)     0
block_7_add[0][0]
block_8_project_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_9_expand (Conv2D)         (None, 2, 2, 384)    24576
block_8_add[0][0]
--------------------------------------------------------------------------------
------------------
block_9_expand_BN (BatchNormali (None, 2, 2, 384)    1536
block_9_expand[0][0]
--------------------------------------------------------------------------------
```

```
__________________
block_9_expand_relu (ReLU)      (None, 2, 2, 384)    0
block_9_expand_BN[0][0]
________________________________________________________________________________
__________________
block_9_depthwise (DepthwiseCon (None, 2, 2, 384)    3456
block_9_expand_relu[0][0]
________________________________________________________________________________
__________________
block_9_depthwise_BN (BatchNorm (None, 2, 2, 384)    1536
block_9_depthwise[0][0]
________________________________________________________________________________
__________________
block_9_depthwise_relu (ReLU)   (None, 2, 2, 384)    0
block_9_depthwise_BN[0][0]
________________________________________________________________________________
__________________
block_9_project (Conv2D)        (None, 2, 2, 64)     24576
block_9_depthwise_relu[0][0]
________________________________________________________________________________
__________________
block_9_project_BN (BatchNormal (None, 2, 2, 64)     256
block_9_project[0][0]
________________________________________________________________________________
__________________
block_9_add (Add)               (None, 2, 2, 64)     0
block_8_add[0][0]
block_9_project_BN[0][0]
________________________________________________________________________________
__________________
block_10_expand (Conv2D)        (None, 2, 2, 384)    24576
block_9_add[0][0]
________________________________________________________________________________
__________________
block_10_expand_BN (BatchNormal (None, 2, 2, 384)    1536
block_10_expand[0][0]
________________________________________________________________________________
__________________
block_10_expand_relu (ReLU)     (None, 2, 2, 384)    0
block_10_expand_BN[0][0]
________________________________________________________________________________
__________________
block_10_depthwise (DepthwiseCo (None, 2, 2, 384)    3456
block_10_expand_relu[0][0]
________________________________________________________________________________
__________________
block_10_depthwise_BN (BatchNor (None, 2, 2, 384)    1536
block_10_depthwise[0][0]
```

```
__________________________________________________________________________________________________
block_10_depthwise_relu (ReLU)  (None, 2, 2, 384)    0
block_10_depthwise_BN[0][0]
__________________________________________________________________________________________________
block_10_project (Conv2D)       (None, 2, 2, 96)     36864
block_10_depthwise_relu[0][0]
__________________________________________________________________________________________________
block_10_project_BN (BatchNorma (None, 2, 2, 96)     384
block_10_project[0][0]
__________________________________________________________________________________________________
block_11_expand (Conv2D)        (None, 2, 2, 576)    55296
block_10_project_BN[0][0]
__________________________________________________________________________________________________
block_11_expand_BN (BatchNormal (None, 2, 2, 576)    2304
block_11_expand[0][0]
__________________________________________________________________________________________________
block_11_expand_relu (ReLU)     (None, 2, 2, 576)    0
block_11_expand_BN[0][0]
__________________________________________________________________________________________________
block_11_depthwise (DepthwiseCo (None, 2, 2, 576)    5184
block_11_expand_relu[0][0]
__________________________________________________________________________________________________
block_11_depthwise_BN (BatchNor (None, 2, 2, 576)    2304
block_11_depthwise[0][0]
__________________________________________________________________________________________________
block_11_depthwise_relu (ReLU)  (None, 2, 2, 576)    0
block_11_depthwise_BN[0][0]
__________________________________________________________________________________________________
block_11_project (Conv2D)       (None, 2, 2, 96)     55296
block_11_depthwise_relu[0][0]
__________________________________________________________________________________________________
block_11_project_BN (BatchNorma (None, 2, 2, 96)     384
block_11_project[0][0]
__________________________________________________________________________________________________
block_11_add (Add)              (None, 2, 2, 96)     0
block_10_project_BN[0][0]
```

```
block_11_project_BN[0][0]
__________________________________________________________________________________
__________________
block_12_expand (Conv2D)        (None, 2, 2, 576)    55296
block_11_add[0][0]
__________________________________________________________________________________
__________________
block_12_expand_BN (BatchNormal (None, 2, 2, 576)    2304
block_12_expand[0][0]
__________________________________________________________________________________
__________________
block_12_expand_relu (ReLU)     (None, 2, 2, 576)    0
block_12_expand_BN[0][0]
__________________________________________________________________________________
__________________
block_12_depthwise (DepthwiseCo (None, 2, 2, 576)    5184
block_12_expand_relu[0][0]
__________________________________________________________________________________
__________________
block_12_depthwise_BN (BatchNor (None, 2, 2, 576)    2304
block_12_depthwise[0][0]
__________________________________________________________________________________
__________________
block_12_depthwise_relu (ReLU)  (None, 2, 2, 576)    0
block_12_depthwise_BN[0][0]
__________________________________________________________________________________
__________________
block_12_project (Conv2D)       (None, 2, 2, 96)     55296
block_12_depthwise_relu[0][0]
__________________________________________________________________________________
__________________
block_12_project_BN (BatchNorma (None, 2, 2, 96)     384
block_12_project[0][0]
__________________________________________________________________________________
__________________
block_12_add (Add)              (None, 2, 2, 96)     0
block_11_add[0][0]
block_12_project_BN[0][0]
__________________________________________________________________________________
__________________
block_13_expand (Conv2D)        (None, 2, 2, 576)    55296
block_12_add[0][0]
__________________________________________________________________________________
__________________
block_13_expand_BN (BatchNormal (None, 2, 2, 576)    2304
block_13_expand[0][0]
__________________________________________________________________________________
__________________
```

```
block_13_expand_relu (ReLU)     (None, 2, 2, 576)    0
block_13_expand_BN[0][0]
________________________________________________________________________________
__________________
block_13_pad (ZeroPadding2D)    (None, 3, 3, 576)    0
block_13_expand_relu[0][0]
________________________________________________________________________________
__________________
block_13_depthwise (DepthwiseCo (None, 1, 1, 576)    5184
block_13_pad[0][0]
________________________________________________________________________________
__________________
block_13_depthwise_BN (BatchNor (None, 1, 1, 576)    2304
block_13_depthwise[0][0]
________________________________________________________________________________
__________________
block_13_depthwise_relu (ReLU)  (None, 1, 1, 576)    0
block_13_depthwise_BN[0][0]
________________________________________________________________________________
__________________
block_13_project (Conv2D)       (None, 1, 1, 160)    92160
block_13_depthwise_relu[0][0]
________________________________________________________________________________
__________________
block_13_project_BN (BatchNorma (None, 1, 1, 160)    640
block_13_project[0][0]
________________________________________________________________________________
__________________
block_14_expand (Conv2D)        (None, 1, 1, 960)    153600
block_13_project_BN[0][0]
________________________________________________________________________________
__________________
block_14_expand_BN (BatchNormal (None, 1, 1, 960)    3840
block_14_expand[0][0]
________________________________________________________________________________
__________________
block_14_expand_relu (ReLU)     (None, 1, 1, 960)    0
block_14_expand_BN[0][0]
________________________________________________________________________________
__________________
block_14_depthwise (DepthwiseCo (None, 1, 1, 960)    8640
block_14_expand_relu[0][0]
________________________________________________________________________________
__________________
block_14_depthwise_BN (BatchNor (None, 1, 1, 960)    3840
block_14_depthwise[0][0]
________________________________________________________________________________
__________________
```

```
block_14_depthwise_relu (ReLU)  (None, 1, 1, 960)    0
block_14_depthwise_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_14_project (Conv2D)       (None, 1, 1, 160)    153600
block_14_depthwise_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_14_project_BN (BatchNorma (None, 1, 1, 160)    640
block_14_project[0][0]

--------------------------------------------------------------------------------
------------------
block_14_add (Add)              (None, 1, 1, 160)    0
block_13_project_BN[0][0]
block_14_project_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_15_expand (Conv2D)        (None, 1, 1, 960)    153600
block_14_add[0][0]

--------------------------------------------------------------------------------
------------------
block_15_expand_BN (BatchNormal (None, 1, 1, 960)    3840
block_15_expand[0][0]

--------------------------------------------------------------------------------
------------------
block_15_expand_relu (ReLU)     (None, 1, 1, 960)    0
block_15_expand_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_15_depthwise (DepthwiseCo (None, 1, 1, 960)    8640
block_15_expand_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_15_depthwise_BN (BatchNor (None, 1, 1, 960)    3840
block_15_depthwise[0][0]

--------------------------------------------------------------------------------
------------------
block_15_depthwise_relu (ReLU)  (None, 1, 1, 960)    0
block_15_depthwise_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_15_project (Conv2D)       (None, 1, 1, 160)    153600
block_15_depthwise_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_15_project_BN (BatchNorma (None, 1, 1, 160)    640
block_15_project[0][0]

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

```
------------------
block_15_add (Add)              (None, 1, 1, 160)    0
block_14_add[0][0]
block_15_project_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_16_expand (Conv2D)        (None, 1, 1, 960)    153600
block_15_add[0][0]
--------------------------------------------------------------------------------
------------------
block_16_expand_BN (BatchNormal (None, 1, 1, 960)    3840
block_16_expand[0][0]
--------------------------------------------------------------------------------
------------------
block_16_expand_relu (ReLU)     (None, 1, 1, 960)    0
block_16_expand_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_16_depthwise (DepthwiseCo (None, 1, 1, 960)    8640
block_16_expand_relu[0][0]
--------------------------------------------------------------------------------
------------------
block_16_depthwise_BN (BatchNor (None, 1, 1, 960)    3840
block_16_depthwise[0][0]
--------------------------------------------------------------------------------
------------------
block_16_depthwise_relu (ReLU)  (None, 1, 1, 960)    0
block_16_depthwise_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_16_project (Conv2D)       (None, 1, 1, 320)    307200
block_16_depthwise_relu[0][0]
--------------------------------------------------------------------------------
------------------
block_16_project_BN (BatchNorma (None, 1, 1, 320)    1280
block_16_project[0][0]
--------------------------------------------------------------------------------
------------------
Conv_1 (Conv2D)                 (None, 1, 1, 1280)   409600
block_16_project_BN[0][0]
--------------------------------------------------------------------------------
------------------
Conv_1_bn (BatchNormalization)  (None, 1, 1, 1280)   5120        Conv_1[0][0]
--------------------------------------------------------------------------------
------------------
out_relu (ReLU)                 (None, 1, 1, 1280)   0           Conv_1_bn[0][0]
--------------------------------------------------------------------------------
------------------
```

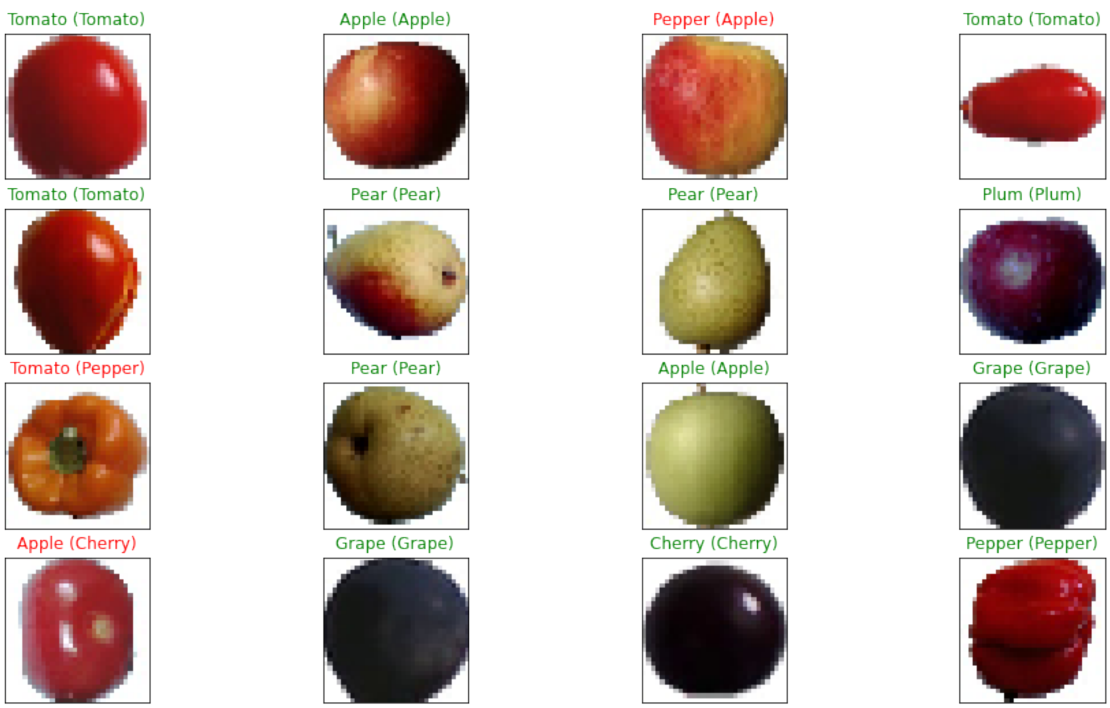
```
global_average_pooling2d_1 (Glo (None, 1280)         0           out_relu[0][0]
________________________________________________________________________________
__________________
Logits (Dense)                  (None, 10)           12810
global_average_pooling2d_1[0][0]
================================================================================
==================
Total params: 2,270,794
Trainable params: 2,236,682
Non-trainable params: 34,112
________________________________________________________________________________
__________________
None
Compiled!
```

[ ]:
```
history = model.fit(x_train,y_train,
        batch_size = 32,
        epochs=10,
        validation_data=(x_valid, y_valid),
        verbose=2  )
```

```
Train on 32607 samples, validate on 8724 samples
Epoch 1/10
 - 152s - loss: 0.6746 - accuracy: 0.7860 - val_loss: 2.2316 - val_accuracy:
0.1977
Epoch 2/10
 - 148s - loss: 0.1768 - accuracy: 0.9496 - val_loss: 2.2468 - val_accuracy:
0.1977
Epoch 3/10
 - 148s - loss: 0.1177 - accuracy: 0.9681 - val_loss: 2.2137 - val_accuracy:
0.1977
Epoch 4/10
 - 147s - loss: 0.1166 - accuracy: 0.9705 - val_loss: 2.1745 - val_accuracy:
0.1977
Epoch 5/10
 - 147s - loss: 0.0993 - accuracy: 0.9756 - val_loss: 0.8243 - val_accuracy:
0.7942
Epoch 6/10
 - 149s - loss: 0.0763 - accuracy: 0.9815 - val_loss: 0.9804 - val_accuracy:
0.7658
Epoch 7/10
 - 146s - loss: 0.0685 - accuracy: 0.9836 - val_loss: 2.2230 - val_accuracy:
0.6530
Epoch 8/10
 - 149s - loss: 0.0582 - accuracy: 0.9859 - val_loss: 9.0434 - val_accuracy:
0.4411
Epoch 9/10
 - 148s - loss: 0.0601 - accuracy: 0.9856 - val_loss: 4.6619 - val_accuracy:
```

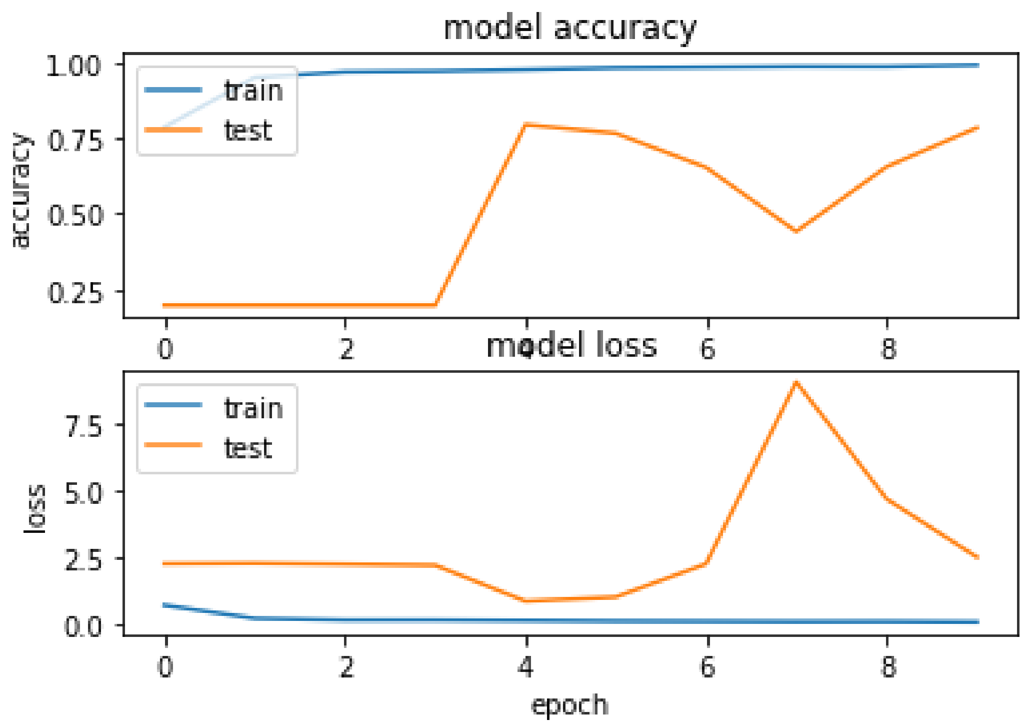
```
0.6551
Epoch 10/10
 - 149s - loss: 0.0429 - accuracy: 0.9905 - val_loss: 2.4783 - val_accuracy:
0.7839
```

[ ]:
```python
y_pred = model.predict(x_test)

# plot a random sample of test images, their predicted labels, and ground truth
fig = plt.figure(figsize=(16, 9))
for i, idx in enumerate(np.random.choice(x_test.shape[0], size=16,␣
 ↪replace=False)):
    ax = fig.add_subplot(4, 4, i + 1, xticks=[], yticks=[])
    ax.imshow(np.squeeze(x_test[idx]))
    pred_idx = np.argmax(y_pred[idx])
    true_idx = np.argmax(y_test[idx])
    ax.set_title("{} ({})".format(TYPES[pred_idx], TYPES[true_idx]),
                 color=("green" if pred_idx == true_idx else "red"))
```



[ ]:
```python
plt.figure(1)


plt.subplot(211)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```

```
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')


plt.subplot(212)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



[ ]:
```
y_pred = model.predict(x_test)
zol = zero_one(y_pred, y_test)
cnn_loss.append(zol)
print("Zero-one Loss: ", zol)
```

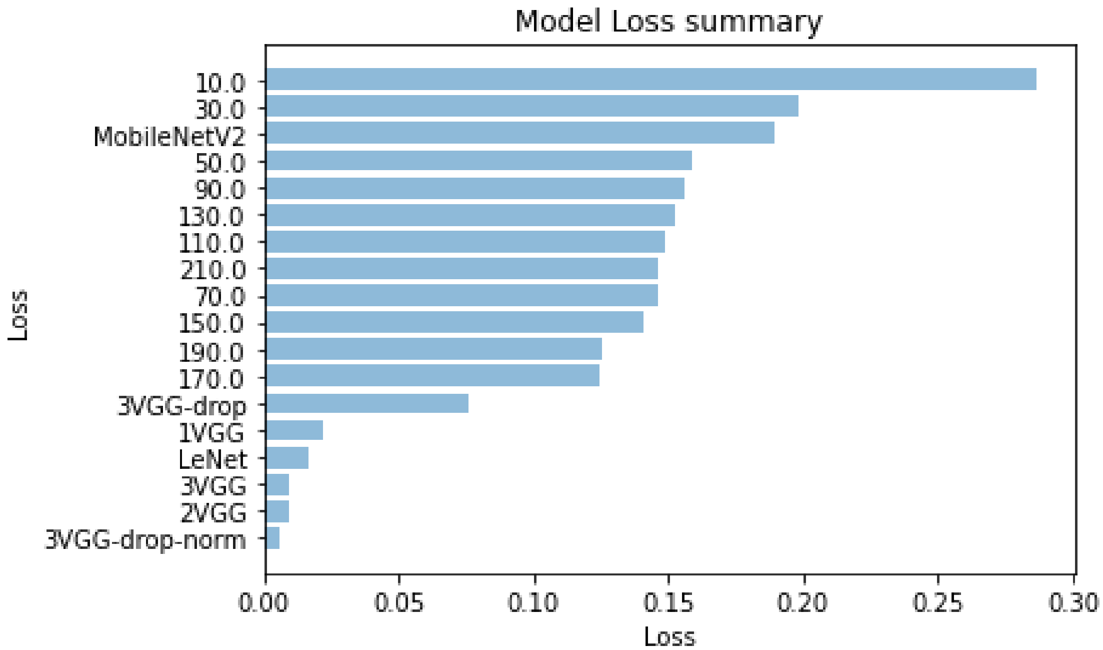
```
Zero-one Loss:  0.18973418881759854
```

8 1.7 Summary results

[ ]:
```
df_cnn_loss = pd.DataFrame(
columns = ['type', 'zero_one']
)

types = ["1VGG","2VGG","3VGG","3VGG-drop",␣
↪"3VGG-drop-norm","LeNet","MobileNetV2"]
df_cnn_loss = pd.DataFrame( data = [types,cnn_loss], index = ['types',␣
↪'zero_one']).T



df_new = df_loss.rename(columns={'components': 'types'})
df_new['types'] = df_new['types'].astype(str)
```

[ ]:
```
frames = [df_new, df_cnn_loss]
result = pd.concat(frames)
result = result.sort_values(by=['zero_one'])

result = result.reset_index(drop=True)
```

[ ]:
```
plt.barh(  result['types'].values,result['zero_one'].values , align='center',␣
↪alpha=0.5)
plt.xlabel('Loss')
plt.ylabel("Loss")
plt.title('Model Loss summary')

plt.show()
```


Model Loss summary
10.0
30.0
MobileNetV2
50.0
90.0
130.0
110.0
210.0
70.0
150.0
190.0
170.0
3VGG-drop
1VGG
LeNet
3VGG
2VGG
3VGG-drop-norm
Loss
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Loss