ResNet_50_contactless_final

May 13, 2022

1 Resnet 50

- Dataset: PolyU Contactless 2D to Contact-based 2D Fingerprint Images Database. http://www4.comp.polyu.edu.hk/~csajaykr/fingerprint.htm
- Keras: 2.8
- TensorFlow: 2.8
- Python: 3.9

Sandra Aguilar

2 Importing Libraries

[85]:
```
import keras
import numpy as np
import PIL
import tensorflow as tf
from tensorflow.python.keras.layers import Dense, Flatten
from keras.models import Sequential
from tensorflow.keras.optimizers import Adam
```

3 Data understanding /Data preparation

Dataset

[86]:
```
import pathlib
data_dir = r'C:
↪\Users\sandra\Documents\Biometrics\processed_contactless_2d_fingerprint_images\first_sessio
data_dir = pathlib.Path(data_dir)
```

[87]:
```
class_count = len(list(data_dir.glob('*/')))
print('Number of classes:' + str(class_count))
```

```
Number of classes:336
```

Showing one image of the dataset

[88]:
```
person1 = list(data_dir.glob('p1/*'))
print(person1[0])
```

```
PIL.Image.open(str(person1[0]))
```

```
C:\Users\sandra\Documents\Biometrics\processed_contactless_2d_fingerprint_images
\first_session\p1\p1.bmp
```

[88]:

3.1 Splitting the dataset

3.1.1 Training Dataset

80 % for training, 20% for validation

[89]:

```
img_height,img_width=350,225
batch_size=32
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.3,
  subset="training",
  seed=123,
  label_mode='categorical',
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 1680 files belonging to 336 classes.
Using 1176 files for training.
```

3.1.2 Validation dataset

[90]:
```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.3,
  subset="validation",
  seed=123,
  label_mode='categorical',
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 1680 files belonging to 336 classes.
Using 504 files for validation.
```

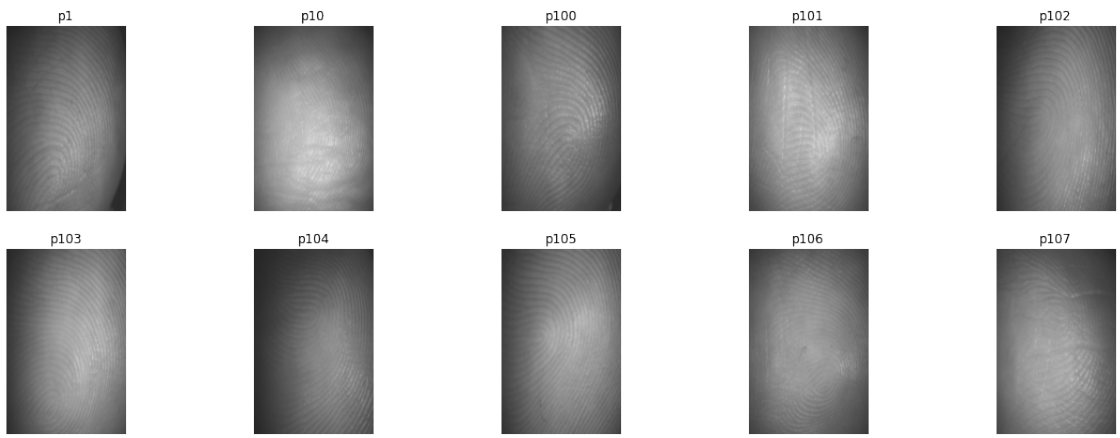
[91]:
```
class_names = train_ds.class_names
print('Number of classes found: ')
print(class_names)
```

```
Number of classes found:
['p1', 'p10', 'p100', 'p101', 'p102', 'p103', 'p104', 'p105', 'p106', 'p107',
'p108', 'p109', 'p11', 'p110', 'p111', 'p112', 'p113', 'p114', 'p115', 'p116',
'p117', 'p118', 'p119', 'p12', 'p120', 'p121', 'p122', 'p123', 'p124', 'p125',
'p126', 'p127', 'p128', 'p129', 'p13', 'p130', 'p131', 'p132', 'p133', 'p134',
'p135', 'p136', 'p137', 'p138', 'p139', 'p14', 'p140', 'p141', 'p142', 'p143',
'p144', 'p145', 'p146', 'p147', 'p148', 'p149', 'p15', 'p150', 'p151', 'p152',
'p153', 'p154', 'p155', 'p156', 'p157', 'p158', 'p159', 'p16', 'p160', 'p161',
'p162', 'p163', 'p164', 'p165', 'p166', 'p167', 'p168', 'p169', 'p17', 'p170',
'p171', 'p172', 'p173', 'p174', 'p175', 'p176', 'p177', 'p178', 'p179', 'p18',
'p180', 'p181', 'p182', 'p183', 'p184', 'p185', 'p186', 'p187', 'p188', 'p189',
'p19', 'p190', 'p191', 'p192', 'p193', 'p194', 'p195', 'p196', 'p197', 'p198',
'p199', 'p2', 'p20', 'p200', 'p201', 'p202', 'p203', 'p204', 'p205', 'p206',
'p207', 'p208', 'p209', 'p21', 'p210', 'p211', 'p212', 'p213', 'p214', 'p215',
'p216', 'p217', 'p218', 'p219', 'p22', 'p220', 'p221', 'p222', 'p223', 'p224',
'p225', 'p226', 'p227', 'p228', 'p229', 'p23', 'p230', 'p231', 'p232', 'p233',
'p234', 'p235', 'p236', 'p237', 'p238', 'p239', 'p24', 'p240', 'p241', 'p242',
'p243', 'p244', 'p245', 'p246', 'p247', 'p248', 'p249', 'p25', 'p250', 'p251',
'p252', 'p253', 'p254', 'p255', 'p256', 'p257', 'p258', 'p259', 'p26', 'p260',
'p261', 'p262', 'p263', 'p264', 'p265', 'p266', 'p267', 'p268', 'p269', 'p27',
'p270', 'p271', 'p272', 'p273', 'p274', 'p275', 'p276', 'p277', 'p278', 'p279',
'p28', 'p280', 'p281', 'p282', 'p283', 'p284', 'p285', 'p286', 'p287', 'p288',
'p289', 'p29', 'p290', 'p291', 'p292', 'p293', 'p294', 'p295', 'p296', 'p297',
'p298', 'p299', 'p3', 'p30', 'p300', 'p301', 'p302', 'p303', 'p304', 'p305',
'p306', 'p307', 'p308', 'p309', 'p31', 'p310', 'p311', 'p312', 'p313', 'p314',
'p315', 'p316', 'p317', 'p318', 'p319', 'p32', 'p320', 'p321', 'p322', 'p323',
'p324', 'p325', 'p326', 'p327', 'p328', 'p329', 'p33', 'p330', 'p331', 'p332',
'p333', 'p334', 'p335', 'p336', 'p34', 'p35', 'p36', 'p37', 'p38', 'p39', 'p4',
'p40', 'p41', 'p42', 'p43', 'p44', 'p45', 'p46', 'p47', 'p48', 'p49', 'p5',
'p50', 'p51', 'p52', 'p53', 'p54', 'p55', 'p56', 'p57', 'p58', 'p59', 'p6',
```

```
'p60', 'p61', 'p62', 'p63', 'p64', 'p65', 'p66', 'p67', 'p68', 'p69', 'p7',
'p70', 'p71', 'p72', 'p73', 'p74', 'p75', 'p76', 'p77', 'p78', 'p79', 'p8',
'p80', 'p81', 'p82', 'p83', 'p84', 'p85', 'p86', 'p87', 'p88', 'p89', 'p9',
'p90', 'p91', 'p92', 'p93', 'p94', 'p95', 'p96', 'p97', 'p98', 'p99']
```

[92]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(20, 7))
for images, labels in train_ds.take(1):
  for i in range(10):
    ax = plt.subplot(2, 5, i+1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[i])
    plt.axis("off")
```



4 Training The Model

[93]:
```
resnet_model = Sequential()

pretrained_model= tf.keras.applications.ResNet50(include_top=False,
                   input_shape=(img_height,img_width,3),
                   pooling='avg',
                   weights='imagenet')

for layer in pretrained_model.layers:
        layer.trainable=False

resnet_model.add(pretrained_model)
resnet_model.add(Flatten()) #this converts to 1D feature vectors
resnet_model.add(Dense(512, activation='relu'))
resnet_model.add(Dense(class_count, activation='softmax'))
```

[94]:
```
resnet_model.summary()
```

```
Model: "sequential_3"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 module_wrapper_9 (ModuleWra  (None, 2048)             0
 pper)

 module_wrapper_10 (ModuleWr  (None, 512)              1049088
 apper)

 module_wrapper_11 (ModuleWr  (None, 336)              172368
 apper)

=================================================================
Total params: 24,809,168
Trainable params: 1,221,456
Non-trainable params: 23,587,712
_________________________________________________________________
```

[95]:
```
resnet_model.compile(optimizer=Adam(learning_rate=0.
 ↪001),loss='categorical_crossentropy',metrics=['accuracy'])
```

[96]:
```
import time
epochs=15
start = time.time()
history = resnet_model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,

)
end = time.time()
print(end - start)
```

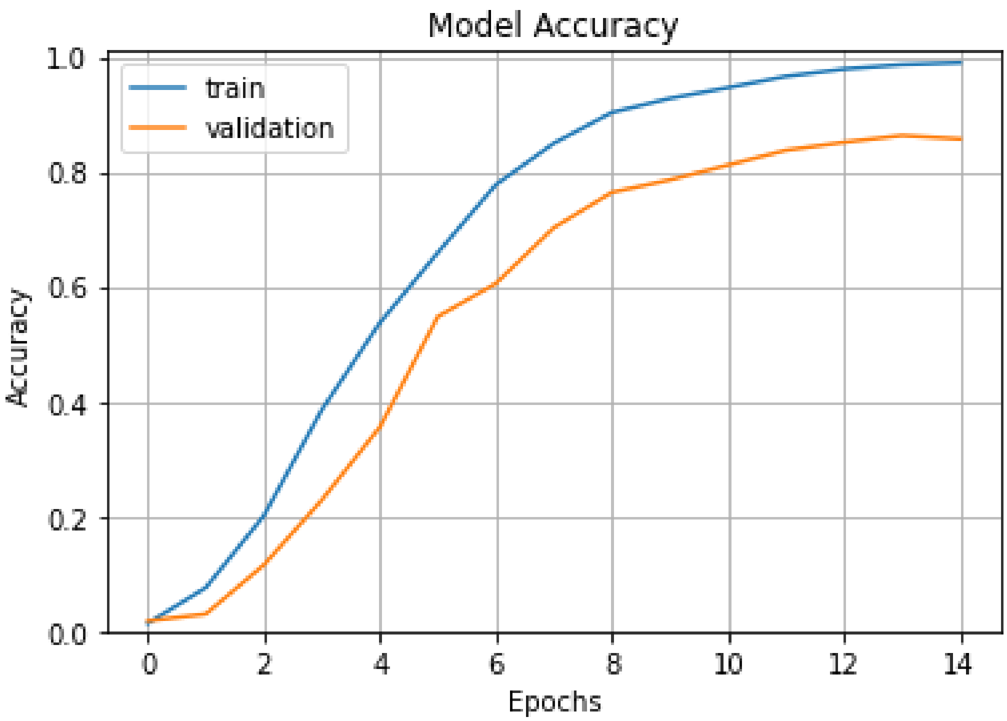
```
Epoch 1/15
37/37 [==============================] - 108s 3s/step - loss: 5.8776 - accuracy:
0.0136 - val_loss: 5.7019 - val_accuracy: 0.0179
Epoch 2/15
37/37 [==============================] - 104s 3s/step - loss: 5.2661 - accuracy:
0.0757 - val_loss: 5.3621 - val_accuracy: 0.0298
Epoch 3/15
37/37 [==============================] - 102s 3s/step - loss: 4.3856 - accuracy:
0.2007 - val_loss: 4.6946 - val_accuracy: 0.1151
Epoch 4/15
```

```
37/37 [==============================] - 99s 3s/step - loss: 3.4095 - accuracy:
0.3852 - val_loss: 3.7547 - val_accuracy: 0.2282
Epoch 5/15
37/37 [==============================] - 98s 3s/step - loss: 2.5809 - accuracy:
0.5366 - val_loss: 3.1191 - val_accuracy: 0.3552
Epoch 6/15
37/37 [==============================] - 98s 3s/step - loss: 1.9353 - accuracy:
0.6590 - val_loss: 2.3825 - val_accuracy: 0.5476
Epoch 7/15
37/37 [==============================] - 106s 3s/step - loss: 1.4119 - accuracy:
0.7772 - val_loss: 1.9653 - val_accuracy: 0.6052
Epoch 8/15
37/37 [==============================] - 100s 3s/step - loss: 1.0436 - accuracy:
0.8495 - val_loss: 1.6428 - val_accuracy: 0.7024
Epoch 9/15
37/37 [==============================] - 104s 3s/step - loss: 0.7732 - accuracy:
0.9031 - val_loss: 1.3442 - val_accuracy: 0.7639
Epoch 10/15
37/37 [==============================] - 99s 3s/step - loss: 0.5941 - accuracy:
0.9277 - val_loss: 1.1538 - val_accuracy: 0.7857
Epoch 11/15
37/37 [==============================] - 101s 3s/step - loss: 0.4470 - accuracy:
0.9464 - val_loss: 1.0242 - val_accuracy: 0.8115
Epoch 12/15
37/37 [==============================] - 105s 3s/step - loss: 0.3515 - accuracy:
0.9660 - val_loss: 0.9512 - val_accuracy: 0.8373
Epoch 13/15
37/37 [==============================] - 104s 3s/step - loss: 0.2733 - accuracy:
0.9787 - val_loss: 0.8549 - val_accuracy: 0.8512
Epoch 14/15
37/37 [==============================] - 105s 3s/step - loss: 0.2172 - accuracy:
0.9864 - val_loss: 0.8016 - val_accuracy: 0.8631
Epoch 15/15
37/37 [==============================] - 105s 3s/step - loss: 0.1820 - accuracy:
0.9898 - val_loss: 0.7638 - val_accuracy: 0.8571
1538.6198437213898
```

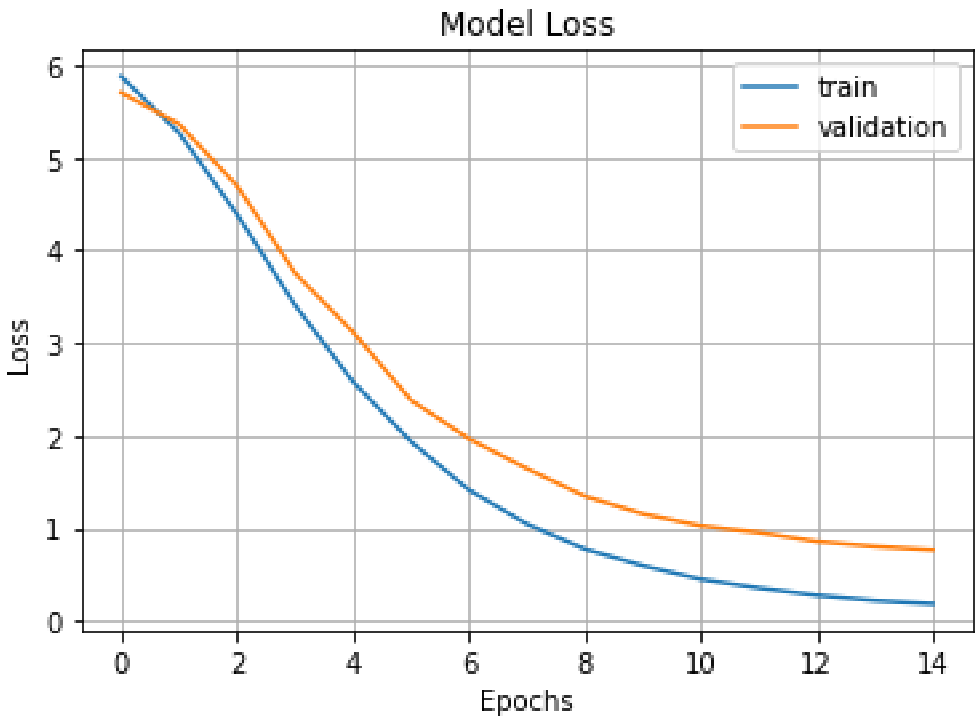
5 Evaluating The Model

[97]:
```
fig1 = plt.gcf()
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.axis(ymin=0,ymax=1.01)
plt.grid()
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
```

```
plt.legend(['train', 'validation'])
plt.show()
```



[98]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.grid()
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.show()
```



6 Saving Model

[113]:
```
tf.keras.models.save_model(
    resnet_model,
    "resnet50_contactless_final.model",
    overwrite=True,
    include_optimizer=True
)
```

```
WARNING:absl:Found untraced functions such as
flatten_3_layer_call_and_return_conditional_losses, flatten_3_layer_call_fn,
dense_6_layer_call_and_return_conditional_losses, dense_6_layer_call_fn,
dense_7_layer_call_and_return_conditional_losses while saving (showing 5 of 6).
These functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: resnet50_contactless_final.model\assets
```

```
INFO:tensorflow:Assets written to: resnet50_contactless_final.model\assets
```

7 Making Predictions

I was taken one image from dataset second_session for testing

[3]:
```
# model2 = tf.keras.models.load_model("resnet50_contactless_final.model")
```

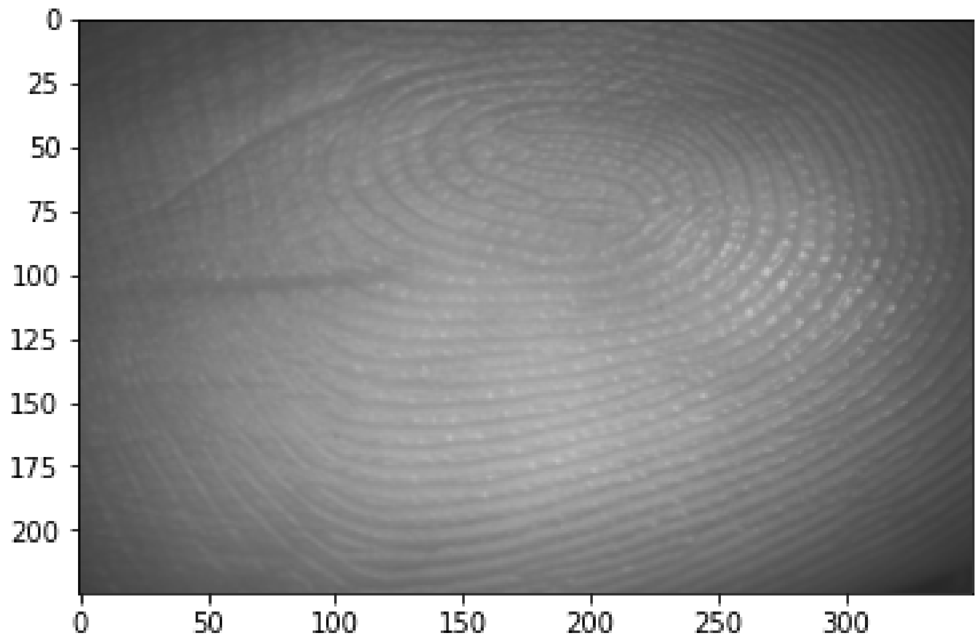
[114]:
```
import cv2
path=r'C:
↪\Users\sandra\Documents\Biometrics\processed_contactless_2d_fingerprint_images\test\p48\p6.
↪bmp'
image=cv2.imread(path)
plt.imshow(image)
image_resized= cv2.resize(image, (img_width,img_height))
image=np.expand_dims(image_resized,axis=0)

print(image.shape)
pred=resnet_model.predict(image)
print(pred)
```

```
(1, 350, 225, 3)
[[2.04394479e-09 3.19118271e-07 1.56376911e-10 4.77075046e-05
  1.63802861e-06 1.43274503e-09 1.23120486e-07 7.66018218e-08
  9.10660489e-08 1.12714060e-09 1.62233206e-07 6.81330192e-08
  4.20125446e-08 1.76640960e-08 2.81999805e-11 5.24919708e-09
  8.00743001e-05 3.59372194e-08 1.41690161e-05 5.33350204e-11
  3.03490274e-03 7.63439166e-05 1.50545702e-07 2.26169838e-09
  5.31825026e-05 2.66204783e-08 3.30943095e-07 1.10004713e-07
  1.97936839e-04 8.57597513e-07 1.66262853e-05 2.44684179e-05
  2.37141899e-06 1.51401964e-05 5.18455818e-06 7.35840558e-06
  9.07390870e-07 1.20865273e-09 9.12611082e-04 1.47482977e-04
  1.13222302e-06 5.34408230e-07 8.90072115e-05 3.21253744e-08
  7.04368285e-05 5.97560465e-05 8.46720241e-08 3.96286123e-05
  9.38835683e-07 1.16529136e-05 1.71598949e-04 1.60176228e-07
  9.33380591e-08 1.77991495e-07 2.80526319e-06 8.12636358e-09
  2.22459562e-06 2.09206971e-03 1.37207168e-03 1.15831397e-04
  2.30454003e-08 6.73606428e-06 8.14280909e-09 3.59368748e-08
  1.18234285e-08 2.39873771e-05 6.46673026e-04 5.24550444e-04
  3.94491330e-02 3.80781486e-08 1.87196965e-05 5.05741315e-09
  4.00216216e-09 9.38831146e-09 1.68044193e-04 3.47869695e-06
  1.77507191e-05 8.67989958e-10 2.35468076e-04 1.70152916e-07
  7.26470205e-07 9.85199762e-08 6.30495970e-06 2.07793215e-04
  3.14377321e-05 7.73985676e-08 2.65573158e-06 1.11402335e-06
  9.21051193e-04 4.03044542e-04 3.83048064e-06 1.54079299e-03
  8.21147842e-06 4.65935003e-03 2.41475180e-02 6.69284145e-06
  1.83625016e-04 1.10804023e-04 1.53457950e-06 5.78707784e-07
  1.06793323e-05 8.87679926e-05 3.29780392e-03 1.09835679e-03
  1.54249756e-05 4.56585155e-07 3.20486748e-09 4.59603733e-08
  4.52236108e-14 2.80530016e-10 4.59049006e-05 5.62248397e-06
  3.03857931e-04 5.80848400e-08 7.99883128e-06 1.49235380e-08
  3.60314516e-08 1.42590096e-07 3.75840415e-07 5.40374190e-09
  1.05924315e-04 2.49493969e-05 2.30335922e-04 8.64544709e-04
```

2.65346830e-06	1.42211002e-06	3.51051188e-09	2.06827693e-07
5.74719277e-04	3.59554433e-05	1.72446936e-03	1.11040635e-10
2.23129804e-09	8.34643217e-08	5.06174692e-04	4.96661094e-08
3.60294439e-09	3.35652339e-11	3.99240416e-05	2.63261427e-05
1.01766936e-05	5.99723062e-05	7.00225769e-07	8.85524033e-12
3.20959913e-07	2.25155288e-03	8.03143885e-10	1.04734488e-06
1.48933452e-11	7.48681384e-09	1.49734890e-12	1.07365783e-09
9.90840590e-07	4.67061145e-09	2.59692556e-09	1.77235258e-08
1.78267229e-02	4.89959821e-05	1.01762280e-05	3.92296158e-07
2.76072587e-08	3.14693182e-07	6.40276819e-04	2.61690457e-05
3.88530680e-06	7.72077125e-03	3.10073556e-05	9.44885699e-08
1.80354305e-02	1.47738829e-05	3.69692210e-09	2.85839405e-06
2.44238890e-05	1.35614874e-03	9.08866696e-07	2.33909868e-05
2.13941540e-08	1.91850904e-05	3.67479151e-05	1.04306082e-06
1.59239989e-05	2.28859676e-07	9.32948581e-07	3.10739782e-03
3.76787170e-07	1.15856797e-04	4.58903813e-08	2.40841640e-08
8.64319663e-05	2.12620489e-06	5.84517545e-10	7.83711075e-05
1.34357924e-06	4.16432641e-07	4.36002495e-07	1.86505567e-05
6.13966677e-07	5.13073064e-05	1.52714038e-05	6.58496749e-04
6.71705948e-07	1.93263884e-04	2.23480220e-06	1.53671976e-07
2.61139804e-10	1.01334706e-07	3.17305460e-10	2.32224573e-09
5.54787390e-13	1.37726426e-08	3.74962065e-06	1.14285825e-07
1.58732055e-05	1.50954525e-04	1.56539261e-06	3.16577243e-05
4.92233099e-09	4.71329251e-08	5.61764431e-08	1.68987463e-05
7.07707310e-04	7.61564115e-06	2.01322018e-05	6.54973383e-06
5.91885907e-07	1.96533627e-04	8.04580376e-02	1.41819119e-05
2.95042759e-04	1.03520470e-05	9.28368820e-07	7.26114768e-09
7.22382614e-12	4.55160034e-07	6.36262412e-04	4.43187106e-04
6.56713706e-09	6.58543172e-07	2.87819439e-05	6.70315103e-06
1.62640068e-10	9.32944033e-09	1.31605486e-06	1.46669026e-06
7.89354071e-02	2.77554820e-04	7.28410669e-05	1.63060511e-04
2.53522128e-08	1.02982049e-05	1.86960751e-05	1.33384175e-08
8.99701147e-11	2.82026478e-04	3.75699638e-05	9.11319489e-07
1.89798186e-08	2.50027608e-08	5.40635703e-10	1.54429025e-09
1.56334272e-06	7.57460424e-04	2.04798882e-04	5.47299860e-03
4.83211534e-06	5.55181032e-06	4.52710680e-07	1.38787035e-08
1.17573018e-06	2.91428034e-04	1.34917414e-07	5.87091723e-04
3.88237910e-04	1.73059976e-04	1.47470723e-06	7.38595554e-05
5.02639050e-05	4.05533356e-04	6.06233079e-04	6.44235432e-01
1.23735299e-05	2.13818055e-07	5.60611610e-08	7.02853140e-05
2.42864808e-05	6.51464163e-08	1.55169022e-04	5.60916196e-05
1.79143134e-03	1.00946141e-04	3.89450870e-05	1.93980689e-08
1.97135151e-07	1.20763303e-07	1.52502134e-05	1.01938294e-02
3.25057073e-03	3.76105418e-05	5.26543431e-08	3.54894851e-08
5.83194321e-07	1.84035223e-08	5.82475934e-08	1.08246226e-03
1.98983662e-06	7.81613693e-04	3.11867479e-04	7.33484683e-07
2.64951996e-05	2.23250718e-05	4.32529887e-05	8.95330857e-07
1.59131009e-02	2.26886594e-03	1.53117930e-03	5.69036405e-04

```
4.19114212e-06 2.69252467e-07 8.14379952e-10 4.27508651e-09
1.85544891e-07 3.19724654e-06 4.68111557e-06 6.05803370e-06
1.93961005e-05 1.89844904e-08 1.70720687e-05 2.83534813e-04
1.60627042e-05 1.43494213e-03 1.86520865e-05 1.58567229e-04
5.00120150e-05 2.56647018e-05 2.26158245e-05 3.07452794e-13]]
```



[115]:
```
output_class=class_names[np.argmax(pred)]
print("The predicted class is", output_class)
```

```
The predicted class is p48
```

[102]:
```
data_dir_test = r'C:
↪\Users\sandra\Documents\Biometrics\processed_contactless_2d_fingerprint_images\test'
data_dir_test = pathlib.Path(data_dir_test)
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_test,
  seed=123,
  label_mode='categorical',
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 336 files belonging to 336 classes.
```

[103]:
```
test_loss, test_acc  = resnet_model.evaluate(test_ds,verbose=2)
print(test_acc)
```

```
11/11 - 22s - loss: 0.6019 - accuracy: 0.8899 - 22s/epoch - 2s/step
0.8898809552192688
```