In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import keras
from keras import layers
from keras.models import Model
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from imgaug import augmenters as iaa

import random
```

Load Dataset

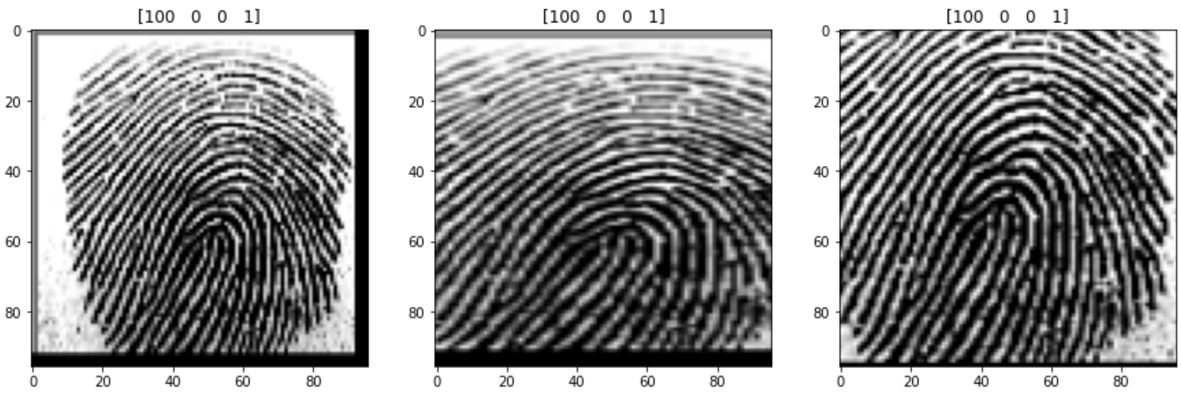
In [7]:

```
x_real = np.load('dataset/x_real.npz')['data']
y_real = np.load('dataset/y_real.npy')

x_zoom = np.load('dataset/x_zoom.npz')['data']
y_zoom = np.load('dataset/y_zoom.npy')

x_partial = np.load('dataset/x_partial.npz')['data']
y_partial = np.load('dataset/y_partial.npy')

print(x_zoom.shape, y_zoom.shape)
print(x_partial.shape, y_partial.shape)

plt.figure(figsize=(15, 10))
plt.subplot(1, 3, 1)
plt.title(y_Real[0])
plt.imshow(x_Real[0].squeeze(), cmap='gray')
plt.subplot(1, 3, 2)
plt.title(y_zoom[0])
plt.imshow(x_zoom[0].squeeze(), cmap='gray')
plt.subplot(1, 3, 3)
plt.title(y_partial[0])
plt.imshow(x_partial[0].squeeze(), cmap='gray')
```

```
(17998, 96, 96) (17998, 4)
(6000, 96, 96) (6000, 4)
```

Out[7]:

```
<matplotlib.image.AxesImage at 0x218f182ef88>
```



Train Test Split

In [8]:

```
x_train, x_val, label_train, label_val = train_test_split(x_zoom, y_zoom, test_siz
print(x_zoom.shape, y_zoom.shape)
print(x_train.shape, label_train.shape)
print(x_val.shape, label_val.shape)
```

```
(17998, 96, 96) (17998, 4)
(16198, 96, 96) (16198, 4)
(1800, 96, 96) (1800, 4)
```

Make Label Dictionary Lookup Table

In [9]:

```
# ID(3)性別(1)左右(1)指頭(1): index
# {'100001': 0, '100004': 1, '100002': 2, ....}
label_real_dict = {}

for i, y in enumerate(y_real):
    key = y.astype(str)
    key = ''.join(key).zfill(6)

    label_real_dict[key] = i
len(label_real_dict)
```

Out[9]:

```
6000
```

Data Generator

In [20]:

```
class DataGenerator(keras.utils.Sequence):
    def __init__(self, x, label, x_real, label_real_dict, batch_size=32, shuffle=T
        'Initialization'
        self.x = x
        self.label = label
        self.x_real = x_real
        self.label_real_dict = label_real_dict

        self.batch_size = batch_size
        self.shuffle = shuffle
        self.on_epoch_end()

    def __len__(self):
        'Denotes the number of batches per epoch'
        return int(np.floor(len(self.x) / self.batch_size))

    def __getitem__(self, index):
        'Generate one batch of data'
        # Generate indexes of the batch
        x1_batch = self.x[index*self.batch_size:(index+1)*self.batch_size]
        label_batch = self.label[index*self.batch_size:(index+1)*self.batch_size]

        x2_batch = np.empty((self.batch_size, 96, 96), dtype=np.float32)
        y_batch = np.zeros((self.batch_size, 1), dtype=np.float32)

        # augmentation
        if self.shuffle:
            seq = iaa.Sequential([
                iaa.GaussianBlur(sigma=(0, 0.5)),
                iaa.Affine(
                    scale={"x": (0.9, 1.1), "y": (0.9, 1.1)},
                    translate_percent={"x": (-0.1, 0.1), "y": (-0.1, 0.1)},
                    rotate=(-30, 30),
                    order=[0, 1],
                    cval=255
                )
            ], random_order=True)

            x1_batch = seq.augment_images(x1_batch)

        # pick matched images(label 1.0) and unmatched images(label 0.0) and put t
        # matched images must be all same, [subject_id(3), gender(1), left_right(1,
        for i, l in enumerate(label_batch):
            match_key = l.astype(str)
            match_key = ''.join(match_key).zfill(6)

            if random.random() > 0.5:
                # put matched image
                x2_batch[i] = self.x_real[self.label_real_dict[match_key]]
                y_batch[i] = 1.
            else:
                # put unmatched image
                while True:
                    unmatch_key, unmatch_idx = random.choice(list(self.label_real_

                    if unmatch_key != match_key:
                        break

                x2_batch[i] = self.x_real[unmatch_idx]
                y_batch[i] = 0.

        return [x1_batch.astype(np.float32) / 255., x2_batch.astype(np.float32) / 

    def on_epoch_end(self):
        if self.shuffle == True:
            self.x, self.label = shuffle(self.x, self.label)
```

In [21]:

```
train_gen = DataGenerator(x_train, label_train, x_real, label_real_dict, shuffle=Tr
val_gen = DataGenerator(x_val, label_val, x_real, label_real_dict, shuffle=False)
```

Create Model

In [22]:

```
x1 = layers.Input(shape=(96, 96, 1))
x2 = layers.Input(shape=(96, 96, 1))

# share weights both inputs
inputs = layers.Input(shape=(96, 96, 1))

feature = layers.Conv2D(32, kernel_size=3, padding='same', activation='relu')(inpu
feature = layers.MaxPooling2D(pool_size=2)(feature)

feature = layers.Conv2D(32, kernel_size=3, padding='same', activation='relu')(featu
feature = layers.MaxPooling2D(pool_size=2)(feature)

feature_model = Model(inputs=inputs, outputs=feature)

# 2 feature models that sharing weights
x1_net = feature_model(x1)
x2_net = feature_model(x2)

# subtract features
net = layers.Subtract()([x1_net, x2_net])
net = layers.Conv2D(32, kernel_size=3, padding='same', activation='relu')(net)
net = layers.MaxPooling2D(pool_size=2)(net)
net = layers.Flatten()(net)
net = layers.Dense(64, activation='relu')(net)
net = layers.Dense(1, activation='sigmoid')(net)

model = Model(inputs=[x1, x2], outputs=net)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
model.summary()
```

```
Model: "model_3"
__________________________________________________________________________________
________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================
================
 input_4 (InputLayer)           [(None, 96, 96, 1)]  0           []

 input_5 (InputLayer)           [(None, 96, 96, 1)]  0           []

 model_2 (Functional)           (None, 24, 24, 32)   9568        ['input_4[0][0]',
                                                                  'input_5[0][0]']

 subtract_1 (Subtract)          (None, 24, 24, 32)   0           ['model_2[0][0]',
                                                                  'model_2[1][0]']

 conv2d_5 (Conv2D)              (None, 24, 24, 32)   9248        ['subtract_1[0]
[0]']

 max_pooling2d_5 (MaxPooling2D)  (None, 12, 12, 32)  0           ['conv2d_5[0]
[0]']

 flatten_1 (Flatten)            (None, 4608)         0           ['max_pooling2d_5
[0][0]']

 dense_2 (Dense)                (None, 64)           294976      ['flatten_1[0]
[0]']

 dense_3 (Dense)                (None, 1)            65          ['dense_2[0][0]']

==================================================================================
================
Total params: 313,857
Trainable params: 313,857
Non-trainable params: 0
__________________________________________________________________________________
________________
```

Train

In [23]:

```
from keras.callbacks import EarlyStopping
# 建立 EarlyStopping 物件
es = EarlyStopping(monitor='val_loss', mode='min',
                   verbose=1, patience=2)

from keras.callbacks import ModelCheckpoint
# 建立 ModelCheckpoint 物件
filename = './data/Siamese_zoom.h5'
mc = ModelCheckpoint(filename, monitor='val_accuracy',
                     mode='max', verbose=0,
                     save_best_only=True)

history = model.fit(train_gen, epochs=10, validation_data=val_gen, callbacks=[es,
Epoch 1/10
WARNING:tensorflow:AutoGraph could not transform <function Model.make_train_functi
on.<locals>.train_function at 0x00000218ED7B8EE8> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity
to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with @tf.autograph.experimental.do_
not_convert
WARNING: AutoGraph could not transform <function Model.make_train_function.<locals
>.train_function at 0x00000218ED7B8EE8> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity
to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with @tf.autograph.experimental.do_
not_convert
506/506 [==============================] - ETA: 0s - loss: 0.3615 - acc: 0.8344WAR
NING:tensorflow:AutoGraph could not transform <function Model.make_test_function.<
locals>.test_function at 0x00000218FBBC0EE8> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity
to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with @tf.autograph.experimental.do_
not_convert
WARNING: AutoGraph could not transform <function Model.make_test_function.<locals
>.test_function at 0x00000218FBBC0EE8> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity
to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with @tf.autograph.experimental.do_
not_convert
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
506/506 [==============================] - 97s 191ms/step - loss: 0.3615 - acc: 0.
8344 - val_loss: 0.1771 - val_acc: 0.9208
Epoch 2/10
506/506 [==============================] - ETA: 0s - loss: 0.2697 - acc: 0.8861WAR
NING:tensorflow:Can save best model only with val_accuracy available, skipping.
506/506 [==============================] - 97s 191ms/step - loss: 0.2697 - acc: 0.
8861 - val_loss: 0.1221 - val_acc: 0.9665
Epoch 3/10
506/506 [==============================] - ETA: 0s - loss: 0.2400 - acc: 0.9000WAR
NING:tensorflow:Can save best model only with val_accuracy available, skipping.
506/506 [==============================] - 97s 193ms/step - loss: 0.2400 - acc: 0.
9000 - val_loss: 0.1274 - val_acc: 0.9554
Epoch 4/10
506/506 [==============================] - ETA: 0s - loss: 0.2128 - acc: 0.9145WAR
NING:tensorflow:Can save best model only with val_accuracy available, skipping.
506/506 [==============================] - 96s 190ms/step - loss: 0.2128 - acc: 0.
9145 - val_loss: 0.1096 - val_acc: 0.9621
Epoch 5/10
506/506 [==============================] - ETA: 0s - loss: 0.1848 - acc: 0.9247WAR
NING:tensorflow:Can save best model only with val_accuracy available, skipping.
506/506 [==============================] - 97s 192ms/step - loss: 0.1848 - acc: 0.
9247 - val_loss: 0.1052 - val_acc: 0.9621
Epoch 6/10
506/506 [==============================] - ETA: 0s - loss: 0.1711 - acc: 0.9339WAR
NING:tensorflow:Can save best model only with val_accuracy available, skipping.
506/506 [==============================] - 97s 191ms/step - loss: 0.1711 - acc: 0.
9339 - val_loss: 0.0939 - val_acc: 0.9732
Epoch 7/10
506/506 [==============================] - ETA: 0s - loss: 0.1608 - acc: 0.9373WAR
NING:tensorflow:Can save best model only with val_accuracy available, skipping.
506/506 [==============================] - 98s 193ms/step - loss: 0.1608 - acc: 0.
9373 - val_loss: 0.0642 - val_acc: 0.9782
Epoch 8/10
```

```
506/506 [==============================] - ETA: 0s - loss: 0.1378 - acc: 0.9471WAR
NING:tensorflow:Can save best model only with val_accuracy available, skipping.
506/506 [==============================] - 96s 190ms/step - loss: 0.1378 - acc: 0.
9471 - val_loss: 0.0456 - val_acc: 0.9827
Epoch 9/10
506/506 [==============================] - ETA: 0s - loss: 0.1322 - acc: 0.9493WAR
NING:tensorflow:Can save best model only with val_accuracy available, skipping.
506/506 [==============================] - 98s 193ms/step - loss: 0.1322 - acc: 0.
9493 - val_loss: 0.0542 - val_acc: 0.9805
Epoch 10/10
506/506 [==============================] - ETA: 0s - loss: 0.1269 - acc: 0.9518WAR
NING:tensorflow:Can save best model only with val_accuracy available, skipping.
506/506 [==============================] - 97s 193ms/step - loss: 0.1269 - acc: 0.
9518 - val_loss: 0.0530 - val_acc: 0.9805
Epoch 10: early stopping
```

save model

In [16]:

```python
# 儲存Keras模型
print('Saving Model: Siamese_zoom.h5 ...')
model.save('./data/Siamese_zoom.h5')
```

```
Saving Model: Siamese_zoom.h5 ...
```

load model

In [11]:

```python
model =  keras.models.load_model('./data/Siamese_zoom.h5')
```

Evaluation

In [17]:

```python
match = np.ones((6000,1))
match.shape
```

Out[17]:

```
(6000, 1)
```

In [25]:

```python
# 評估模型
print('\nTesting ...')
loss, accuracy = model.evaluate([x_partial.astype(np.float32) / 255.,x_real.astype
print('測試資料集的準確度 = {:.2f}'.format(accuracy))
```

```
Testing ...
188/188 [==============================] - 8s 43ms/step - loss: 0.9574 - acc: 0.69
17
測試資料集的準確度 = 0.69
```

In [29]:

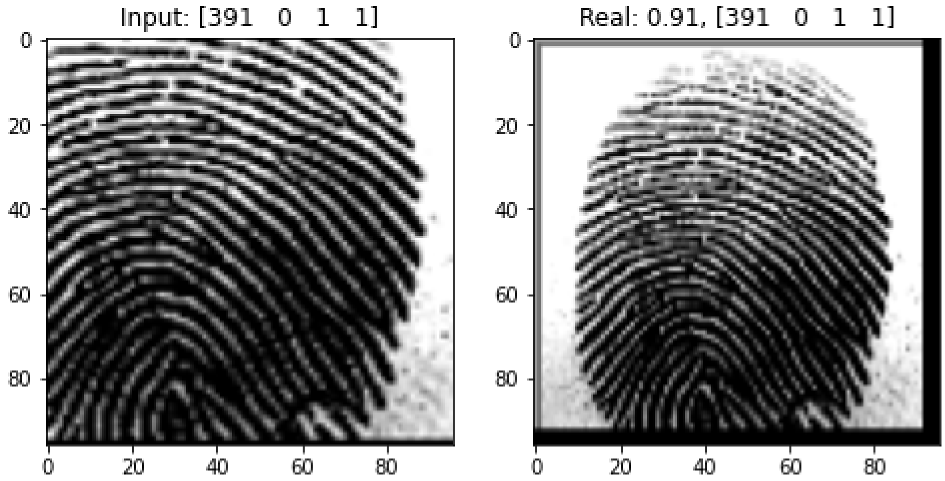
```python
# new user fingerprint input
random_idx = random.randint(0, len(x_partial))

random_img = x_partial[random_idx]
random_label = y_partial[random_idx]

random_img = random_img.reshape((1, 96, 96, 1)).astype(np.float32) / 255.

# matched image
match_key = random_label.astype(str)
match_key = ''.join(match_key).zfill(6)
```

```
rx = x_real[label_real_dict[match_key]].reshape((1, 96, 96, 1)).astype(np.float32)
ry = y_real[label_real_dict[match_key]]

pred_rx = model.predict([random_img, rx])
plt.figure(figsize=(8, 4))
plt.subplot(1, 2, 1)
plt.title('Input: %s' %random_label)
plt.imshow(random_img.squeeze(), cmap='gray')
plt.subplot(1, 2, 2)
plt.title('Real: %.02f, %s' % (pred_rx, ry))
plt.imshow(rx.squeeze(), cmap='gray')
```

```
1/1 [==============================] - 0s 22ms/step
```

Out[29]:

```
<matplotlib.image.AxesImage at 0x218fbeadf88>
```


Input: [391   0   1   1]
Real: 0.91, [391   0   1   1]