工業瑕疵資料分類

CNN小試身手

Environment Setup

- 開啟 Jupyter Notebook

Jupyter Notebook (Anaconda3)

應用程式

- 進入您存放檔案(資料集、code)的工作目錄
- 開啟cnn.ipynb檔進入Python編譯環境
- Jupyter Notebook教學在其他檔案有說明

實驗所需套件

- 實驗所需套件

```
import numpy as np
np.random.seed(1337)  # for reproducibility
from keras.datasets import mnist
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense, Activation, Convolution2D, MaxPooling2D, Flatten
from keras.optimizers import Adam
import matplotlib.pyplot as plt
%matplotlib inline
```

- 訓練過程繪圖函數

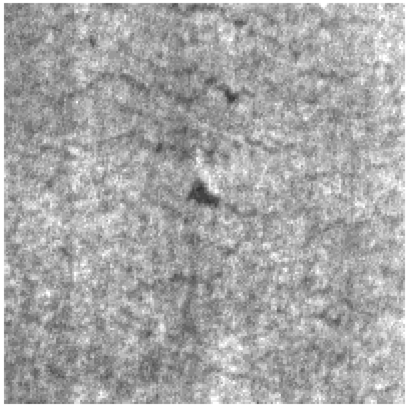
```
def show_train_history(train_history,train,validation):
    plt.plot(train_history.history[train])
    plt.plot(train_history.history[validation])
    plt.title('Train History')
    plt.ylabel('train')
    plt.xlabel('Epoch')
    plt.legend(['train','validation'],loc='upper left')
    plt.show()
```

CNN應用於工業

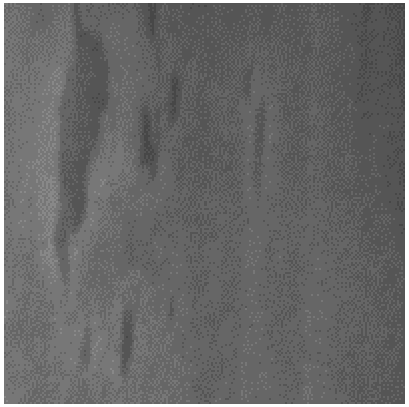
鋼板熱軋製程

- 熱軋鋼捲強度足、韌性佳、焊接性良、加工成形易，除可做為冷軋產品的主要原料外，更可廣泛用於五金、汽車、家電零件加工、結構用管件、建築用管件、加工為建築結構用鋼等

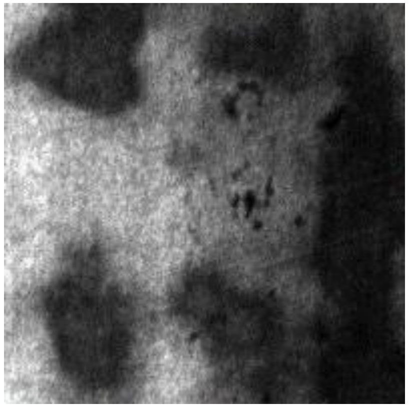
鋼板瑕疵辨識



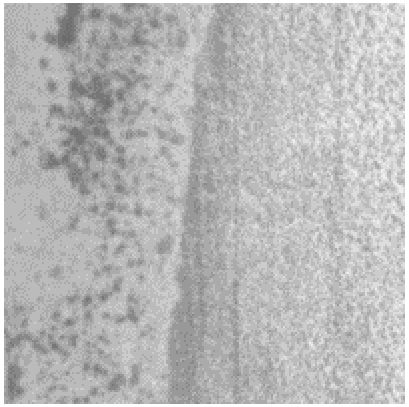
crazing (Cr)



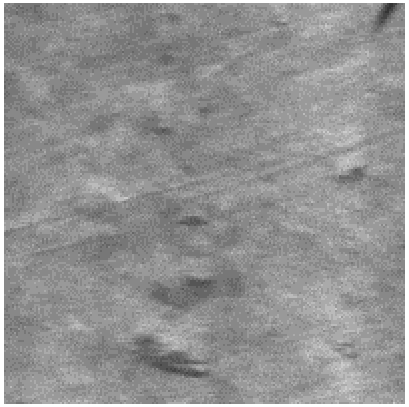
inclusion (In)



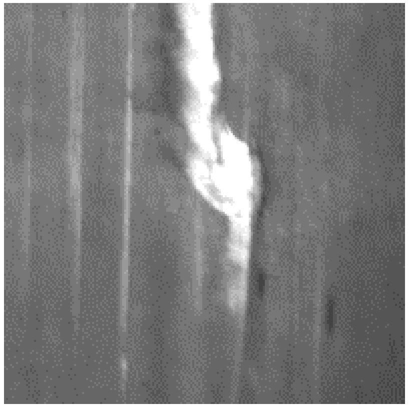
patches (Pa)



pitted surface (PS)



rolled-in scale (RS)



scratches (Sc)

實驗所需套件

- 準備實驗所需套件

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
from keras.models import Sequential
from keras.models import load_model
from keras.layers import Dense, Activation, Convolution2D, MaxPooling2D, Flatten,Dense
from keras.utils import np_utils
from keras.optimizers import Adam
from keras.preprocessing import image
from sklearn.preprocessing import OneHotEncoder
from sklearn.utils import shuffle
from keras.models import Model
```

```
pip install opencv-python
conda install -c conda-forge opencv
```

Load function

- 讀取label

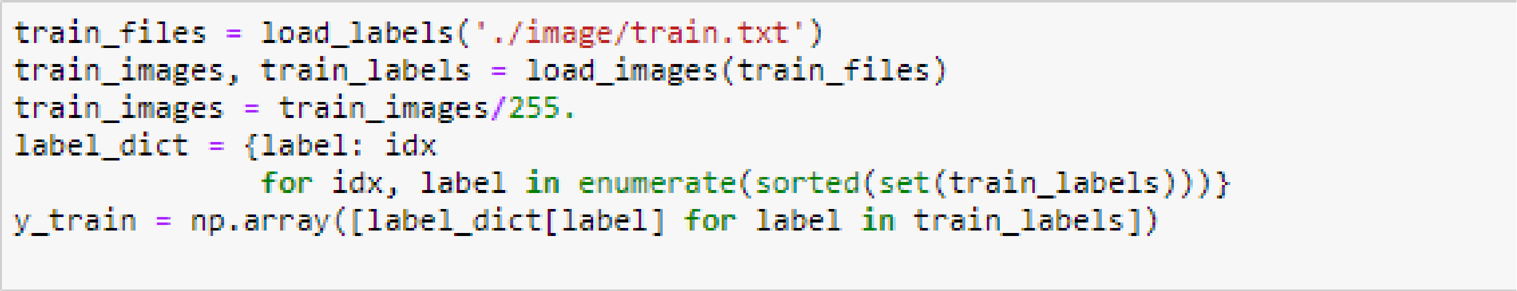
```
def load_labels(filepath):
    with open(filepath, 'r') as f:
        return [line.strip() for line in f]
```

- 讀取image

```
def load_images(image_files):
    img_list = []
    images = [cv2.imread('./image/file/{}'.format(p),0) for p in image_files]
    images = np.array(images)
    for image in images:
        img_list.append(cv2.resize(image,(100,100),interpolation=cv2.INTER_CUBIC))
    get_image_class = lambda path: path.split('_')[0]
    labels = list(map(get_image_class, image_files))
    return np.array(img_list), labels
```

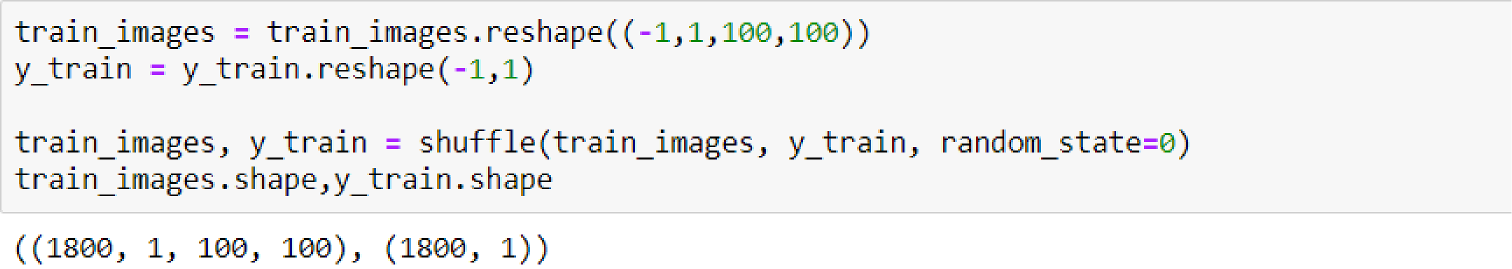
得到feature和label

- train.txt 記錄每一筆資料檔名，讀取並整理成array



```
train_files = load_labels('./image/train.txt')
train_images, train_labels = load_images(train_files)
train_images = train_images/255.
label_dict = {label: idx
              for idx, label in enumerate(sorted(set(train_labels)))}
y_train = np.array([label_dict[label] for label in train_labels])
```

- 將資料順序隨機打亂

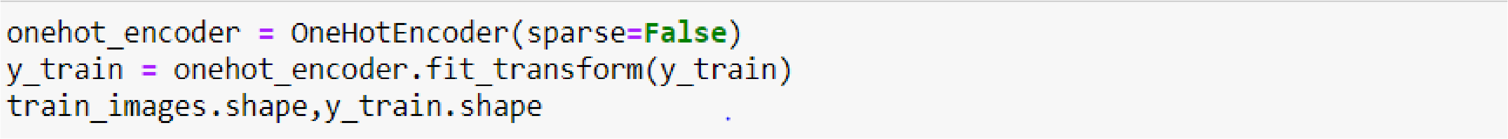


```
train_images = train_images.reshape((-1,1,100,100))
y_train = y_train.reshape(-1,1)

train_images, y_train = shuffle(train_images, y_train, random_state=0)
train_images.shape,y_train.shape
```

```
((1800, 1, 100, 100), (1800, 1))
```

- 將label 進行 one hot encoder



```
onehot_encoder = OneHotEncoder(sparse=False)
y_train = onehot_encoder.fit_transform(y_train)
train_images.shape,y_train.shape
```

可視化工具

- 將訓練過程可視化

```
def show_train_history(train_history,train,validation):
    plt.plot(train_history.history[train])
    plt.plot(train_history.history[validation])
    plt.title('Train History')
    plt.ylabel('train')
    plt.xlabel('Epoch')
    plt.legend(['train','validation'],loc='upper left')
    plt.show()
```

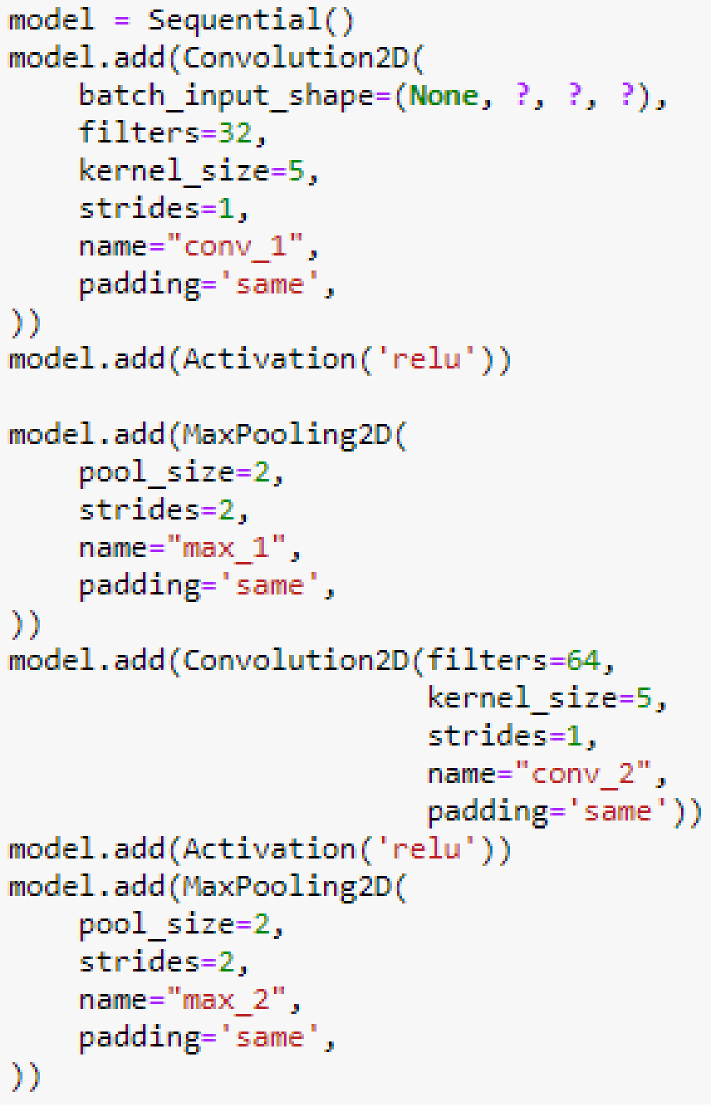
資料前置作業完畢!

開始搭建模型吧

撰寫模型

- 撰寫CNN模型
  - Layer: Conv -> Maxpooling -> Conv -> Maxpooling -> Flatten -> Dense
  - Filter, kernel size, stride都可以自己設定

撰寫模型



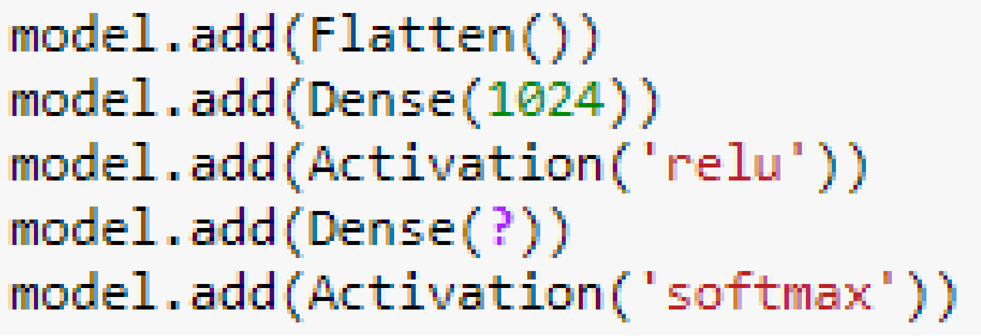
```
model = Sequential()
model.add(Convolution2D(
    batch_input_shape=(None, ?, ?, ?),
    filters=32,
    kernel_size=5,
    strides=1,
    name="conv_1",
    padding='same',
))
model.add(Activation('relu'))

model.add(MaxPooling2D(
    pool_size=2,
    strides=2,
    name="max_1",
    padding='same',
))
model.add(Convolution2D(filters=64,
                        kernel_size=5,
                        strides=1,
                        name="conv_2",
                        padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(
    pool_size=2,
    strides=2,
    name="max_2",
    padding='same',
))
```

Hint : 對照前面資料前置作業，想一下這裡要填多少(None,channel,height,width)

撰寫模型

- 撰寫CNN模型
  - Layer: Conv -> Maxpooling -> Conv -> Maxpooling -> Flatten -> Dense
  - Filter, kernel size, stride都可以自己設定

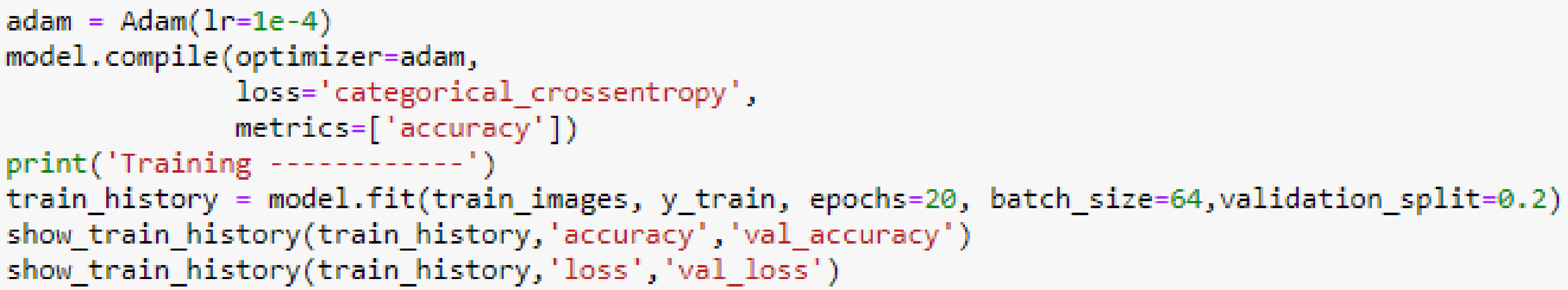


```
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation('relu'))
model.add(Dense(?))
model.add(Activation('softmax'))
```

Hint: defect總共有幾類?

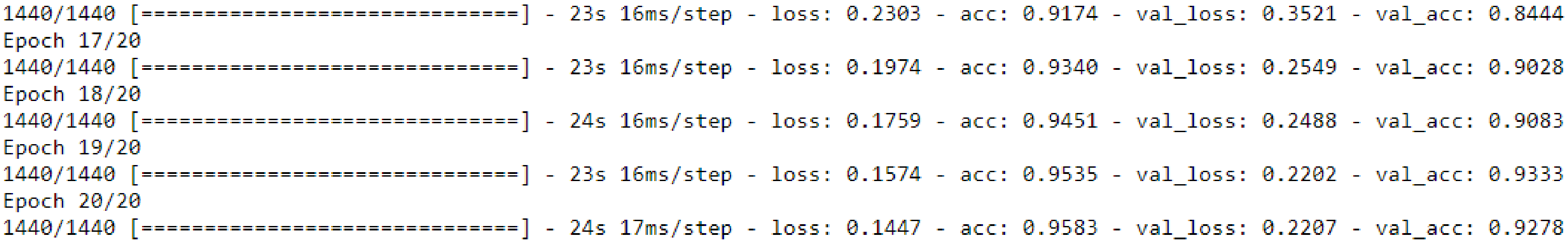
訓練模型

- 訓練模型
  - 以adam為optimizer
  - Loss以crossentropy計算

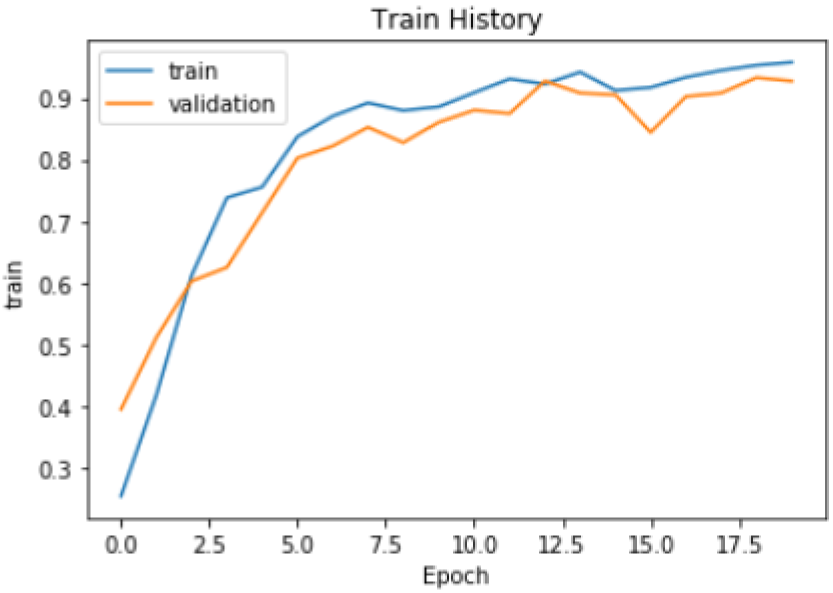


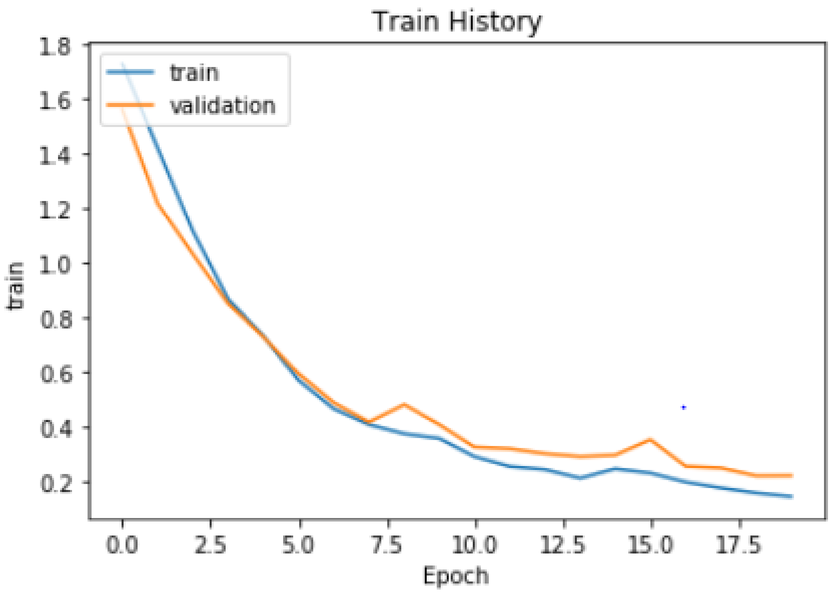
```
adam = Adam(lr=1e-4)
model.compile(optimizer=adam,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
print('Training ------------')
train_history = model.fit(train_images, y_train, epochs=20, batch_size=64,validation_split=0.2)
show_train_history(train_history,'accuracy','val_accuracy')
show_train_history(train_history,'loss','val_loss')
```

checkpoint



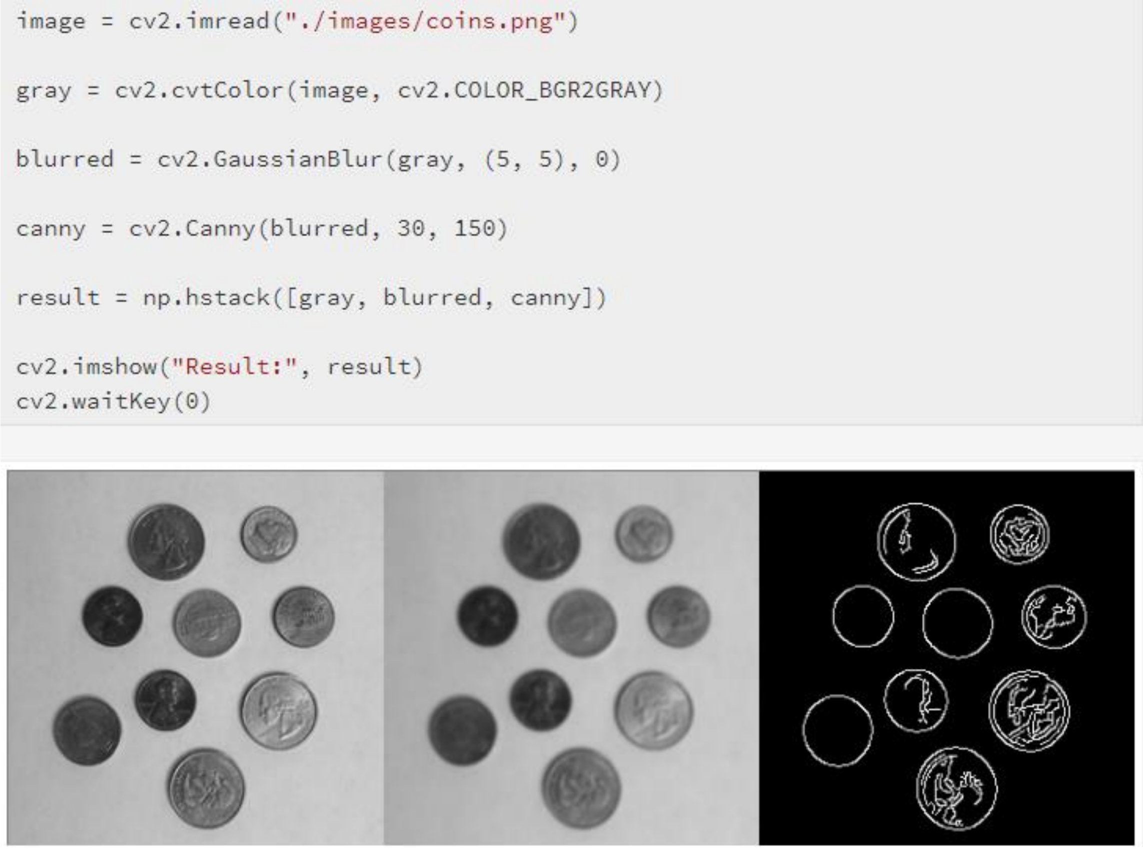
```
1440/1440 [==============================] - 23s 16ms/step - loss: 0.2303 - acc: 0.9174 - val_loss: 0.3521 - val_acc: 0.8444
Epoch 17/20
1440/1440 [==============================] - 23s 16ms/step - loss: 0.1974 - acc: 0.9340 - val_loss: 0.2549 - val_acc: 0.9028
Epoch 18/20
1440/1440 [==============================] - 24s 16ms/step - loss: 0.1759 - acc: 0.9451 - val_loss: 0.2488 - val_acc: 0.9083
Epoch 19/20
1440/1440 [==============================] - 23s 16ms/step - loss: 0.1574 - acc: 0.9535 - val_loss: 0.2202 - val_acc: 0.9333
Epoch 20/20
1440/1440 [==============================] - 24s 17ms/step - loss: 0.1447 - acc: 0.9583 - val_loss: 0.2207 - val_acc: 0.9278
```





Tutorial

- OpenCV
  - https://dotblogs.com.tw/coding4fun/2017/11/09/125723



```
image = cv2.imread("./images/coins.png")

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

canny = cv2.Canny(blurred, 30, 150)

result = np.hstack([gray, blurred, canny])

cv2.imshow("Result:", result)
cv2.waitKey(0)
```

Tutorial

- Defect segmentation
  - https://medium.com/@guildbilla/steel-defect-detection-image-segmentation-using-keras-dae8b4f986f0