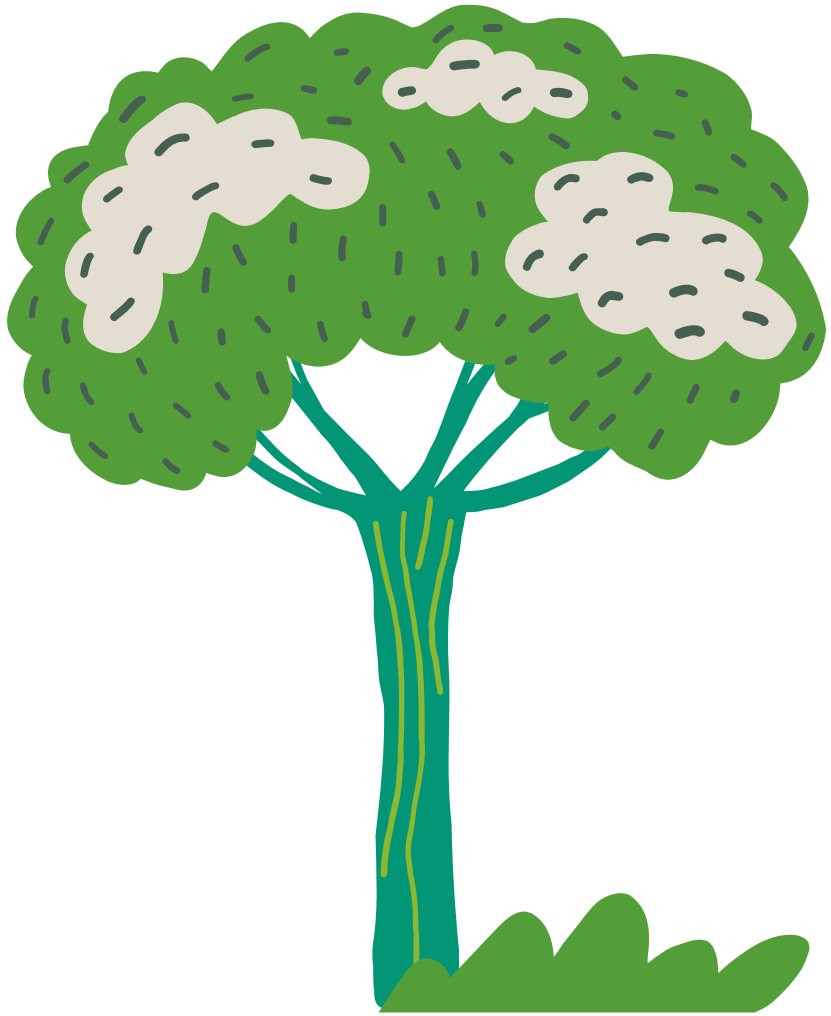
分類開心果品種

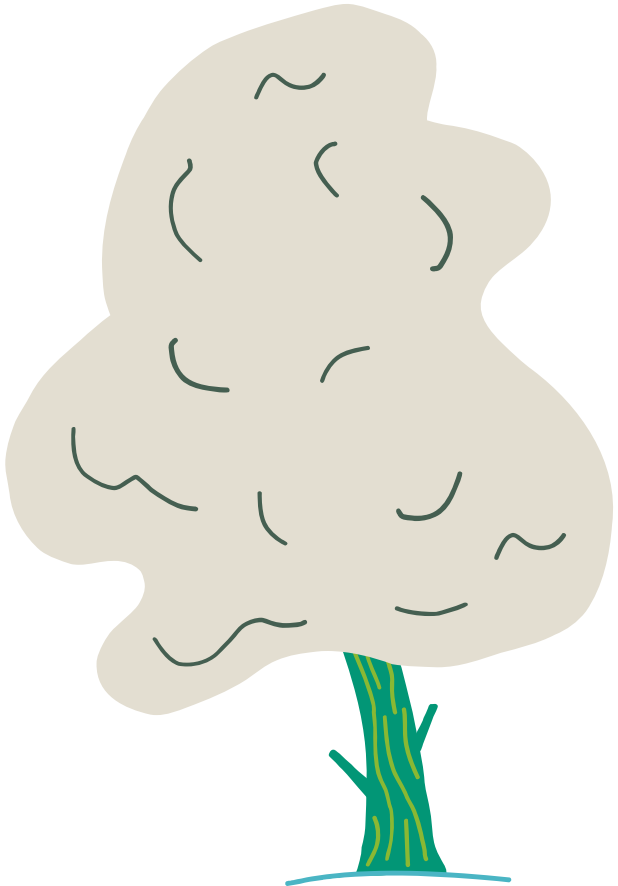
111426025 盧盈穎

2022/12/12

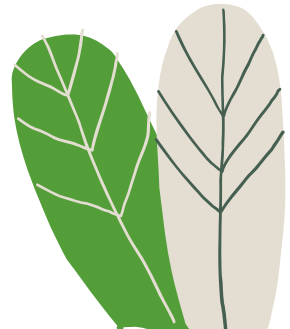


目錄

01 研究動機
Research motivation



目前農業利用科技輔助越來越成熟，運用影像辨識辨別作物，檢查是否有腐壞、不良品等，那此次資料集為開心果的辨識資料，此資料集有2種類別的開心果，期望透過貝式分類器及CNN訓練，達到可辨識的效果。



Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year	Download
Pistachio Image Dataset	2 Class	Classification Clustering	Image	2148	Image	2022	**Download** 4134 downloaded
Citation Request	**1:** SINGH D, TASPINAR YS, KURSUN R, CINAR I, KOKLU M, OZKAN IA, LEE H-N., (2022). Classification and Analysis of Pistachio Species with Pre-Trained Deep Learning Models, *Electronics*,11 (7), 981. https://doi.org/10.3390/electronics11070981. (Open Access) **DOI:** https://doi.org/10.3390/electronics11070981 **2:** OZKAN IA., KOKLU M. and SARACOGLU R. (2021). Classification of Pistachio Species Using Improved K-NN Classifier. *Progress in Nutrition*, Vol. 23, N. 2. https://doi.org/10.23751/pn.v23i2.9686. (Open Access) **DOI:** https://doi.org/10.23751/pn.v23i2.9686						

使用muratkoklu所提供Pistachio Image Dataset資料集

資料筆數為2148張圖片

https://www.muratkoklu.com/datasets/

02. 資料集介紹 / 資料觀察 / 資料前處理
Intro. to Datasets

Kirmizi_Pistachio

Siirt_Pistachio

上面兩張圖為開心果的兩種類別

模糊的圖片

物品邊緣有異物

物品邊緣不明顯

上面因素皆可能造成訓練成效不好，故會藉由影像強化，來增加影像辨識成功機率。

接下來將會藉由訓練影像強化前和強化後運用於不同種模型中進行比較。

02. 資料集介紹 / 資料觀察 / **資料前處理**

Intro. to Datasets

調整圖片大小，劃分訓練、測試集

```
print("Shape of Images:",train_images.shape)
print("Shape of Labels:",train_labels.shape)
```

```
Shape of Images: (1758, 224, 224, 3)
Shape of Labels: (1758,)
```

```
test_images = np.array(test_images)
test_labels = np.array(test_labels)

print("Shape of Images:",test_images.shape)
print("Shape of Labels:",test_labels.shape)
```

```
資料夾 Siirt_Pistachio 讀取中 ....
資料夾 Kirmizi_Pistachio 讀取中 ....
資料已讀取完成.
Shape of Images: (360, 224, 224, 3)
Shape of Labels: (360,)
```

保留各自類別的開心果圖片集 20筆，

當作後續預測用之資料。

Kirmizi_Pistachio

共20張圖

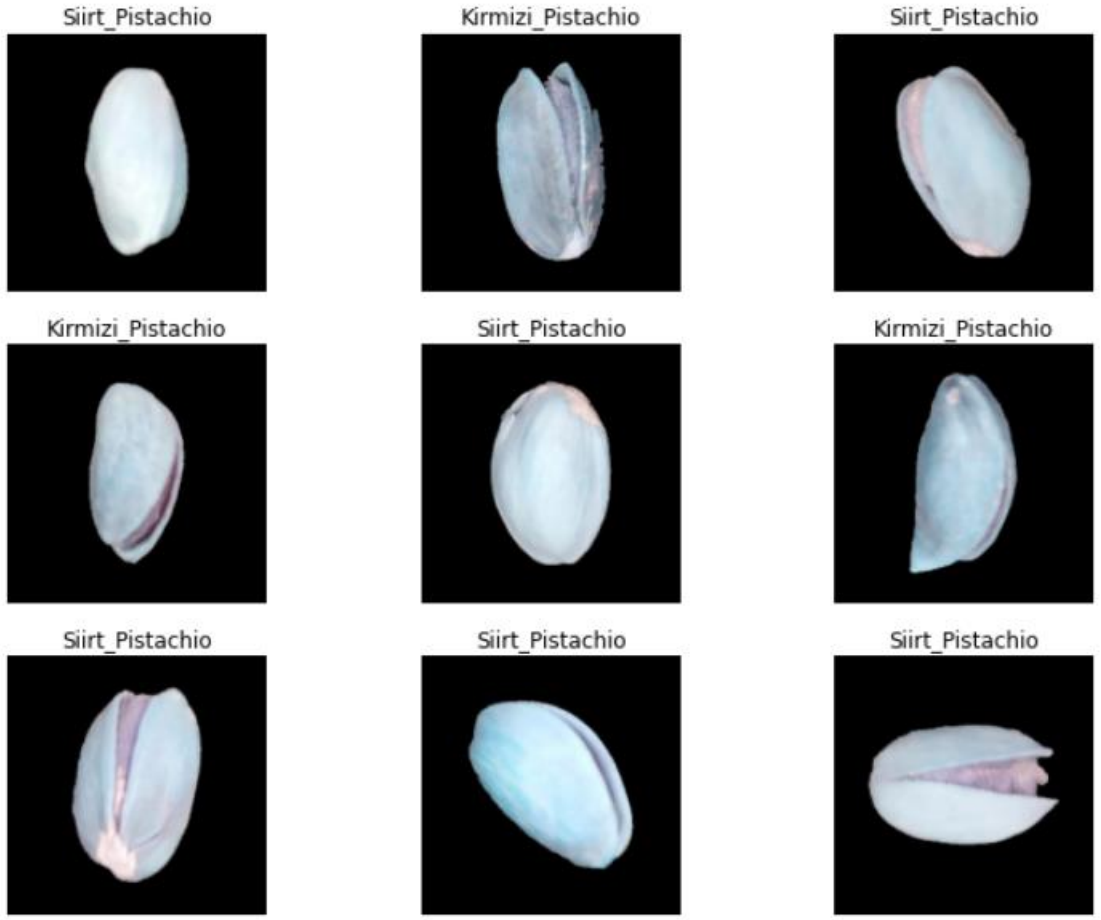
Siirt_Pistachio

共20張圖

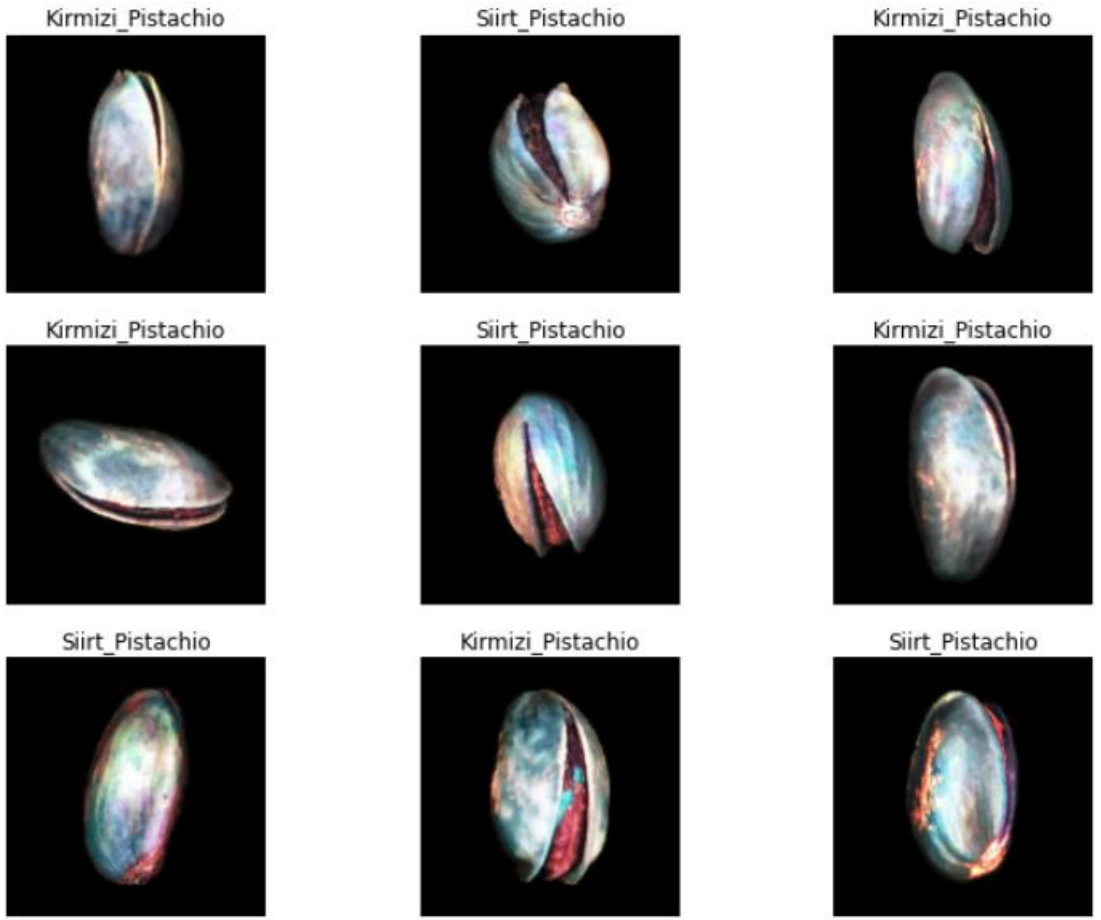
02. 資料集介紹 / 資料觀察 / **資料前處理**

Intro. to Datasets

進行圖片強化前



進行圖片強化後



03. 貝式分類器
Bayesian Classifier

進行圖片強化前

```
train_images = train_images.reshape(1758, 150528)
```

```
test_images = test_images.reshape(360, 150528)
```

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB().fit(train_images, train_labels)
```

```
score = model.score(test_images, test_labels)
score
```

```
0.7111111111111111
```

圖片強化後

```
train_images_r = train_images_r.reshape(1758, 150528)
test_images_r = test_images_r.reshape(360, 150528)
```

```
from sklearn.naive_bayes import GaussianNB
model_r = GaussianNB().fit(train_images_r, train_labels_r)
```

```
score_r = model_r.score(test_images_r, test_labels_r)
score_r
```

```
0.6111111111111112
```

在貝式分類器中，原始圖片在準確率方面表現比強化圖片之後來的好。

在準確率方面可以達到71.11%

04. CNN
CNN

建置CNN模型

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_1 (Conv2D)	(None, 112, 112, 128)	73856
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_2 (Conv2D)	(None, 56, 56, 256)	295168
conv2d_3 (Conv2D)	(None, 56, 56, 256)	590080
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_4 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_5 (Conv2D)	(None, 28, 28, 512)	2359808
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 512)	0
conv2d_6 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_7 (Conv2D)	(None, 14, 14, 512)	2359808
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 4096)	102764544
dense_1 (Dense)	(None, 4096)	16781312
dropout (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 2)	8194

Total params: 128,774,530
Trainable params: 128,774,530
Non-trainable params: 0

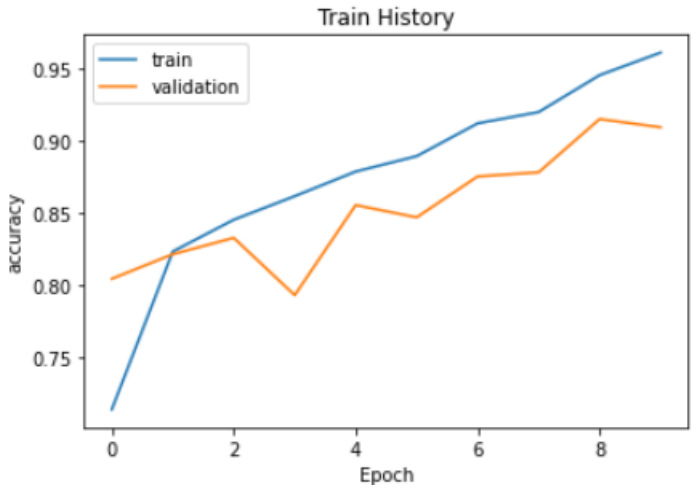
在此模型中，在每一層中都有增添加激活函數，以relu的方式進行，但在最一層全連階層時，透過激活函數softmax進行正規劃。

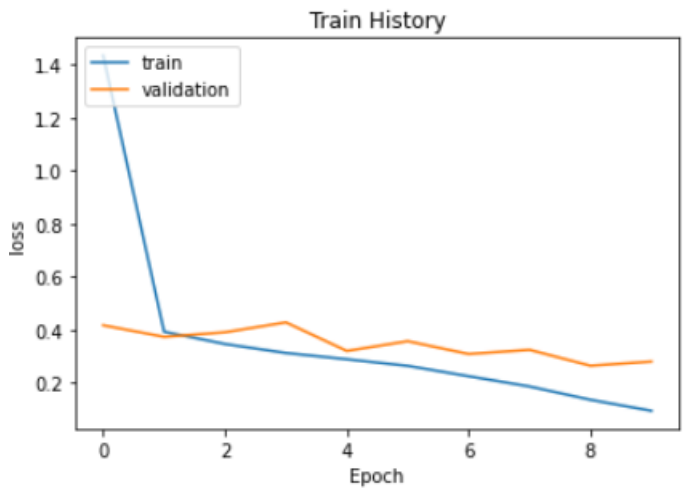
04. CNN
CNN

圖片強化前

```
scores = model.evaluate(train_images, train_labels)
print("\t[Info] Accuracy of testing data = {:2.1f}%".format(scores[1]*100.0))
```

```
55/55 [==============================] - 5s 82ms/step - loss: 0.0909 - accuracy: 0.9721
        [Info] Accuracy of testing data = 97.2%
```





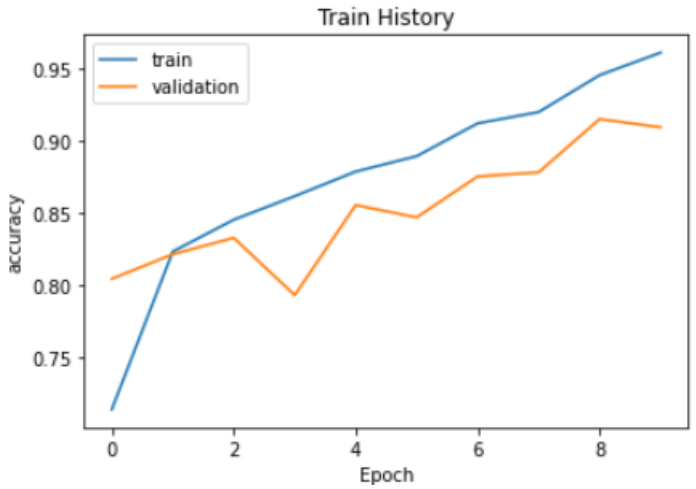
在CNN模型中，可以達到97.2%的準確率，且隨著epoch增加，測試集的準確率能同時上升，損失函數仍然能夠持續下降。

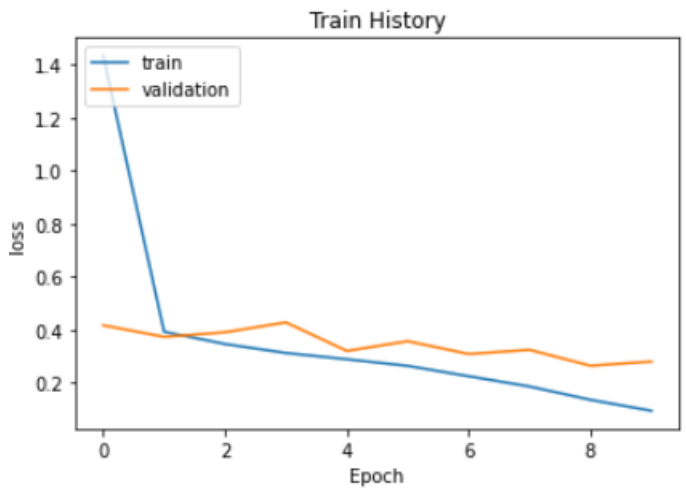
04. CNN
CNN

圖片強化前

```
model.evaluate(test_images,  test_labels)
```

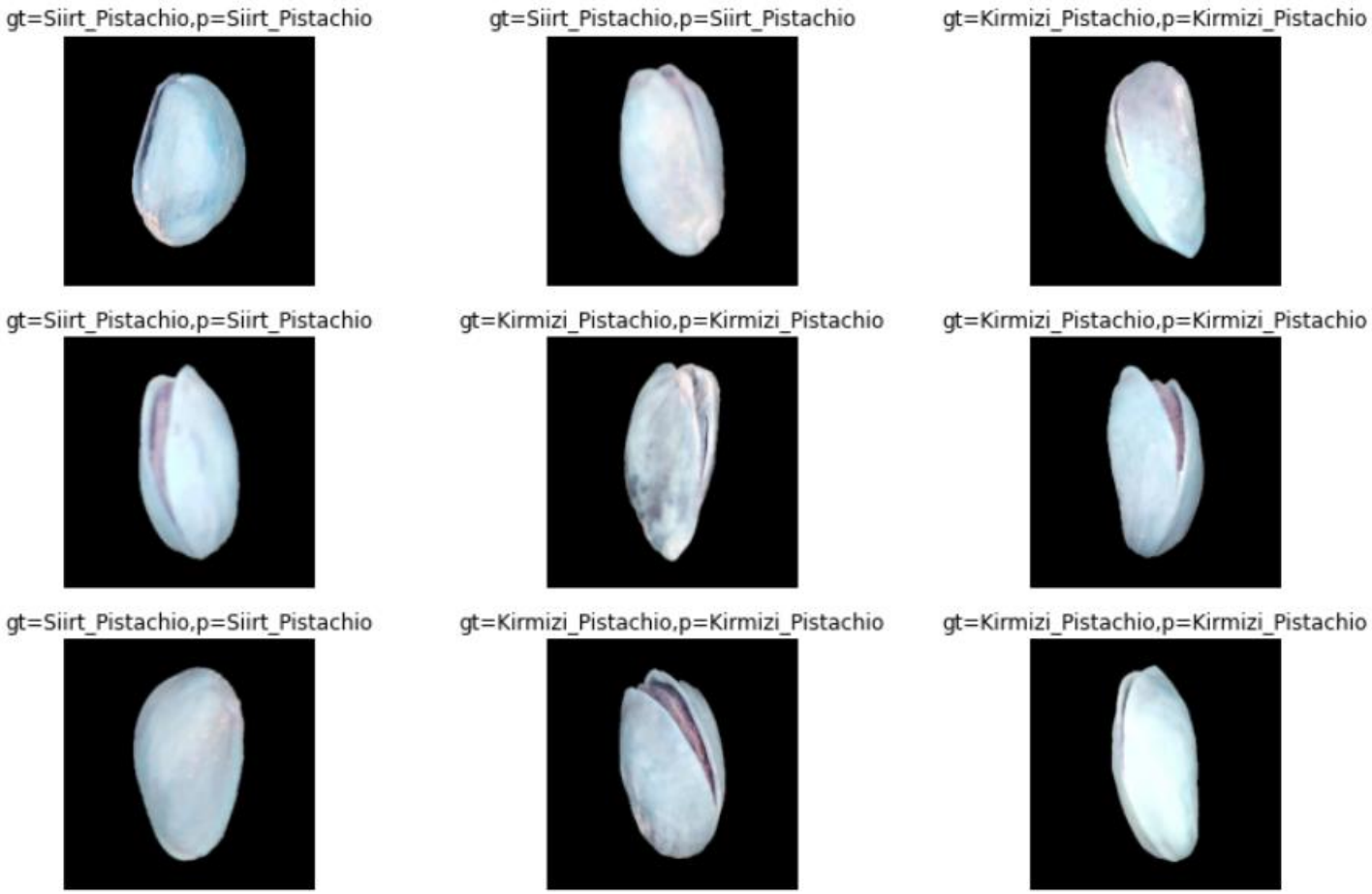
```
12/12 [==============================] - 2s 132ms/step - loss: 0.2111 - accuracy: 0.9250
[0.21114523708820343, 0.925000011920929]
```





在CNN模型中，可以達到92.5%的準確率，且隨著epoch增加，測試集的準確率能同時上升，損失函數仍然能夠持續下降。

圖片強化前 – 測試及判讀結果



gt : 為圖片的實際類別

p :為模型預測類別

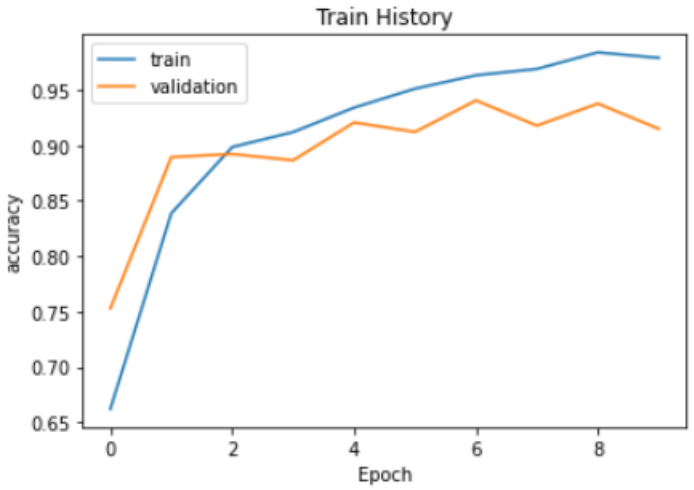
在左邊這個隨機挑選展示的例子中，可以看出CNN皆能準確判別出圖片的類別。

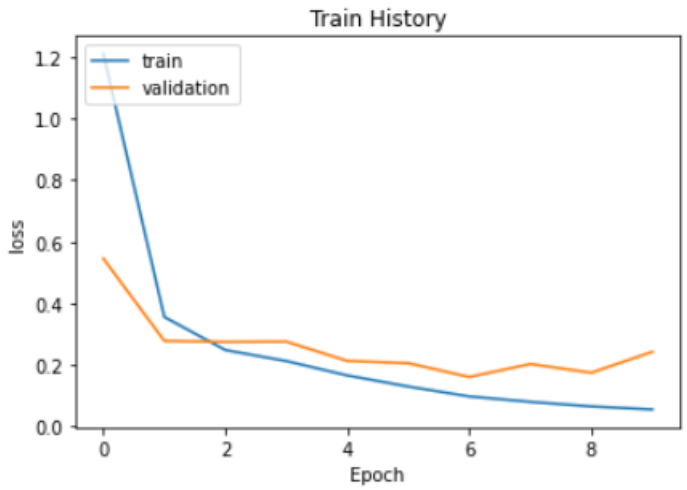
04. CNN
CNN

圖片強化後

```
model.evaluate(test_images,  test_labels)
```

```
12/12 [==============================] - 1s 80ms/step - loss: 0.1459 - accuracy: 0.9500
[0.1459493488073349, 0.949999988079071]
```

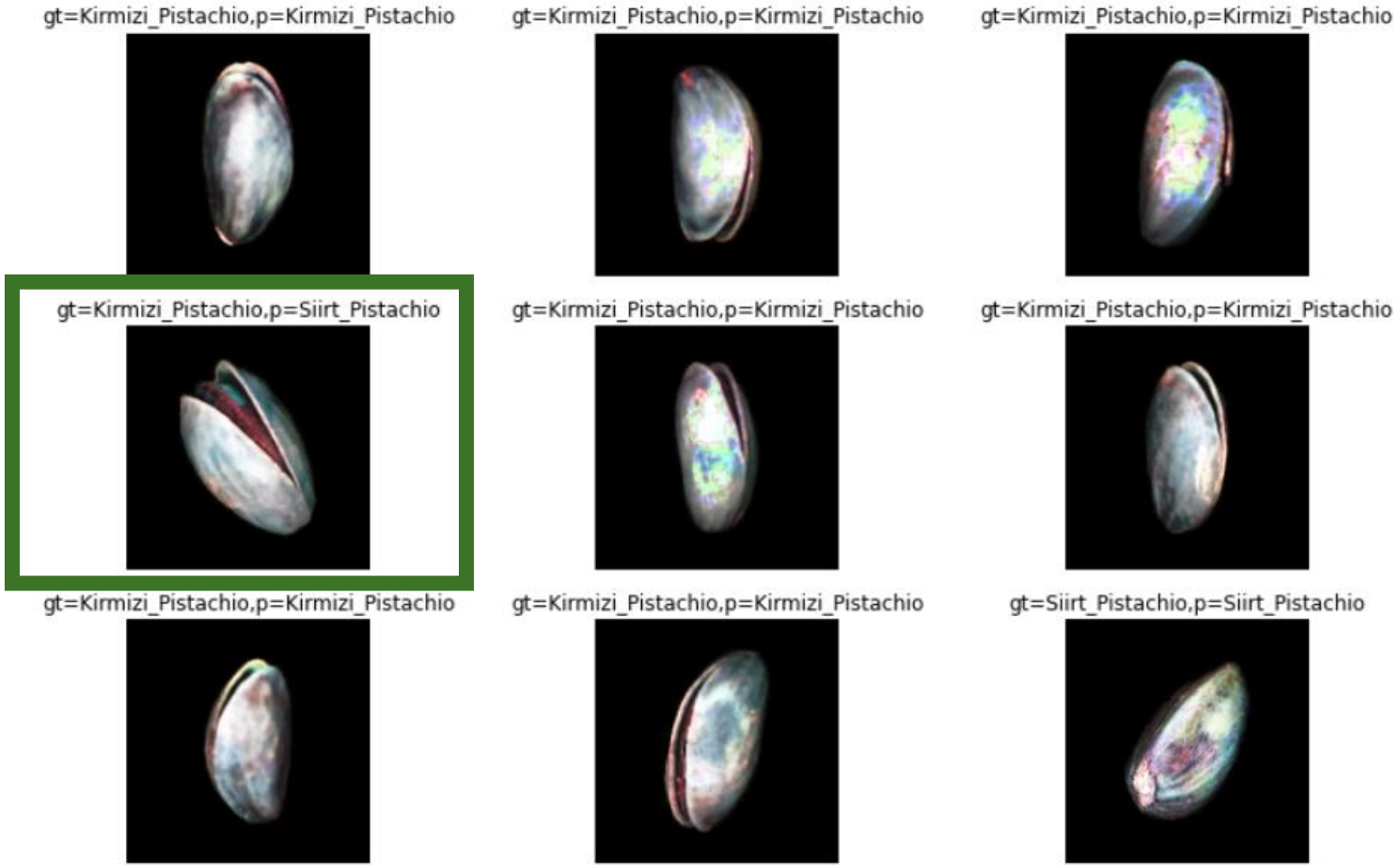




在CNN模型中，可以達到95%的準確率，且隨著epoch增加，測試集的準確率能同時上升，損失函數仍然能夠持續下降。

04. CNN
CNN

圖片強化後 – 測試及判讀結果



gt : 為圖片的實際類別

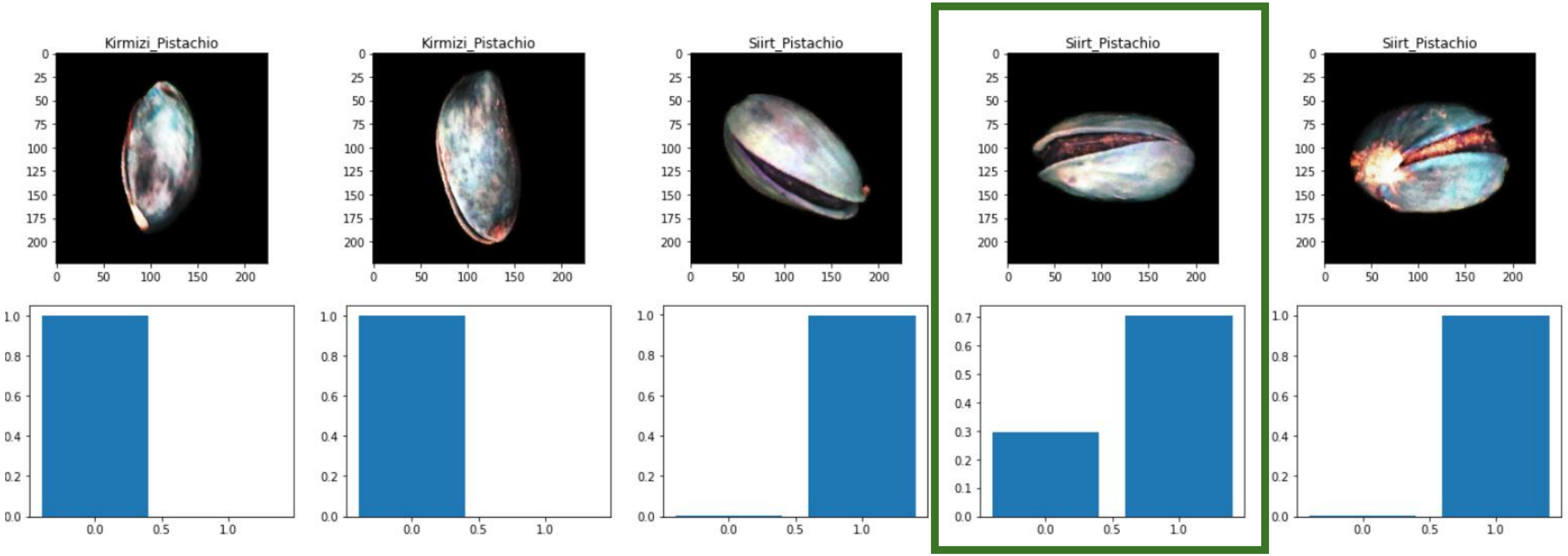
p :為模型預測類別

在左邊這個隨機挑選展示的例子中，綠色方框圈起處，為判讀失敗的部分。那這幾張圖可看出，大部分的影像相較於未強化前更加清晰，在邊緣方面有較突出。

最後選定模型為圖片強化後CNN模型

04. CNN
CNN

將保留為預測集資料放入強化圖片模型中進行判讀



上方圖片為預測集圖片，但圖片上的標籤來自於下方直方圖的機率，0:Kirmizi_Pistachio，1:Siirt_Pistachio，可以看出框起處，可能為模型較不容易判別出來的，較容易造成誤判。

05. 結論
Conclusion

透過貝式分類器和CNN的比較，可以得到在成效方面CNN的判讀結果比較好，但若考慮模型訓練時間的話，CNN所花費時間較貝式分類器來的長，且在貝式分類器中，圖片可以不用經過強化，仍可得到比強化後更好的結果，因此，若考慮時間成本的話，可能要選擇的是貝式分類器；但若是考慮誤判成本的話，便要選擇CNN。

謝謝大家

111426025 盧盈穎