ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Báo cáo đồ án

Môn học : Máy học CS114.K21.KHTN

Năm học: học kì II 2019-2020

Đề tài
Nhận diện trái cây qua hình ảnh

GVHD:
PGS,TS: Lê Đình Duy
ThS: Nguyễn Trường An

Sinh viên thực hiện
Phạm Lê Quang Nhật _ 18520120

Hồ Chí Minh , ngày 15 tháng 8 năm 2020

I Mở đầu

1. Giới thiệu đề tài

Hiện nay lĩnh vực công nghệ thông tin rất phát triển, được áp dụng trong nhiều lĩnh vực như : đời sống kinh tế, sản xuất, công nghiệp. Nông nghiệp cũng không ngoại lệ, áp dụng công nghệ thông tin vào nông nghiệp có hai lĩnh vực chính: chăn nuôi và trồng trọt, thì trong lĩnh vực trồng trọt bài toán nhận diện trái cây là một bài toán cơ bản đầu tiên trong nhiều bài toán lớn nhưng nó cũng rất là quan trọng.

2. Mục tiêu

Nhằm áp dụng những kiến thức trong Machine Learning vào giải quyết vấn đề thực tế, đồng thời cũng giúp em hiểu được những vấn đề khó khăn khi xây dựng một bài toán ML, em sẽ xây dựng một ứng dụng giúp nhận diện trái cây .

II Mô tả bài toán

Ở bài toán này, chúng tôi sẽ xây dựng mô hình phân loại trái cây. Với input là 1 bức ảnh chứa 1 loại trái cây, output là tên được dự đoán có tên trong số 12 loại trái cây sau: Apple, Avocado, Banana, Orange, Guava, Dragon Fruit, Coconut, Start Fruit,, Plum, Watermelon, Custard Apple

III Cách thức thu nhập và tiền xử lí dữ liệu

1. Cách thu thập bộ dữ liệu

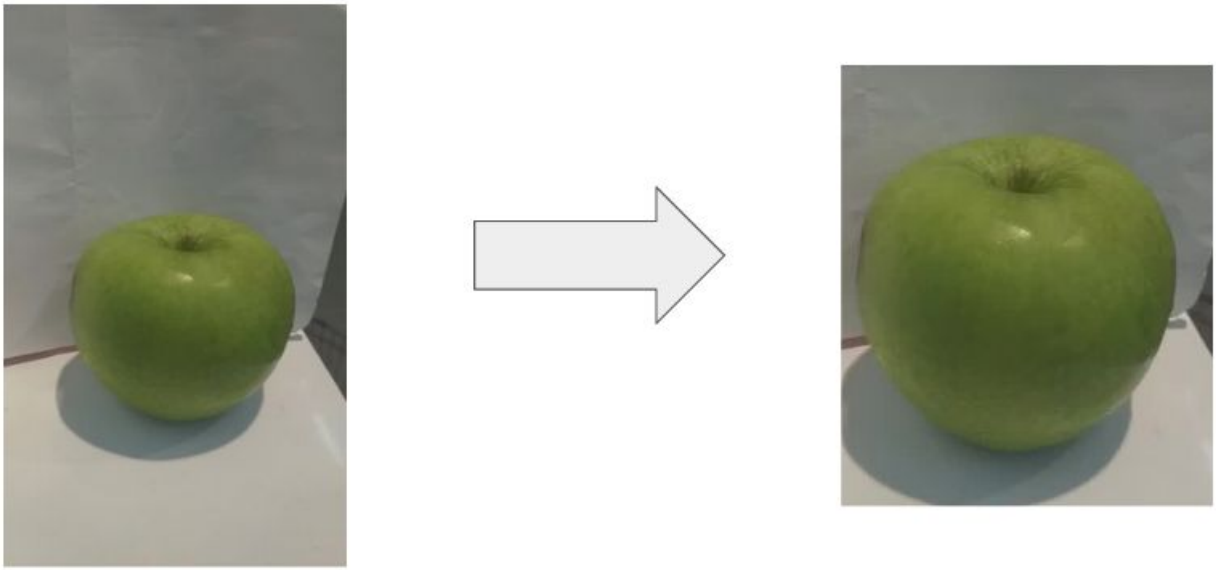
Đồ án này thu thập dữ liệu từ 2 nguồn chính là internet và tự thu thập. Với 95% dữ liệu là tự thu thập cùng với 2 bạn khác cùng lớp , còn lại là 5% lấy từ nguồn internet. Dữ liệu tự thu thập bao gồm dữ liệu quay video sau đó tách từng frame ảnh tuy giúp tiết kiệm thời gian thu thập và sinh ra nhiều

dữ liệu nhưng lại cho ảnh có chất lượng thấp. Vì vậy chúng tôi đã thu thập thêm dữ liệu bằng cách chụp trực tiếp từ điện thoại. Ảnh cho ra chất lượng cao, tuy nhiên phương pháp này đòi hỏi người thu thập phải cực khổ hơn phương pháp trước. Còn bộ dữ liệu trên Internet được tải trên kaggle [7], Cụ thể phân bố dữ liệu của chúng tôi như sau:

- Ở bộ dữ liệu train gồm 11020 bức ảnh .png được tách frame từ các video được quay bằng điện thoại, 2000 ảnh png được chụp bằng điện thoại, 700 ảnh được tải từ bộ dữ liệu trên kaggle [7]. Số lượng bức ảnh có trong 12 class:
    + Class Apple : 1159 ảnh
    + Class Avocado : 1478 ảnh
    + Class Banana : 1308 ảnh
    + Class Coconut : 1185 ảnh
    + Class Custard_apple : 941 ảnh
    + Class Dragon_fruit : 682 ảnh
    + Class Guava : 989 ảnh
    + Class Mango : 966 ảnh
    + Class Orange : 1488 ảnh
    + Class Plum : 1482 ảnh
    + Class Start_fruit : 1175 ảnh
    + Class Watermelon : 957 ảnh
- Bộ dữ liệu test thứ nhất gồm 2400 bức ảnh .png được chụp bằng điện thoại, được chia thành 12 class trái cây giống như tập train, mỗi loại gồm 200 bức ảnh.
- Bộ dữ liệu test thứ hai gồm 120 ảnh được down trực tiếp bằng tay từ trên google, chia đều vào 12 class trái cây, mỗi loại gồm 10 bức ảnh.

2. Tiền xử lí dữ liệu

- Bước 1: bộ dữ liệu train và dữ liệu test thứ nhất sẽ được xử lí cắt giảm background bằng tay từng bức ảnh



Hình 1: Hình bên trái là hình ảnh trước khi xử lí cắt giảm background, hình bên phải là hình ảnh sau khi xử lí cắt giảm background

- Bước 2: các hình ảnh được resize về kích thước 32 x 32.
    + Thông thường , resize một bức ảnh thường chúng ta không quan tâm đến tỉ số giữa chiều dài và chiều rộng (aspect ratio) của bức ảnh, điều này sẽ dẫn đến bức ảnh sau khi resize, đồ vật trong bức ảnh bị méo mó

Hình 2: Đối với hình ở giữa (ignore Aspect ratio) thì khi resize lại ảnh sẽ bị kéo dài ra hay bên, còn hình bên tay phải (Consider Aspect ratio) có hình dáng giống với ảnh gốc hơn

3 Rút trích đặc trưng

Về phần rút trích đặc trưng, tôi đề xuất thử với 3 kiểu rút trích đặc trưng:

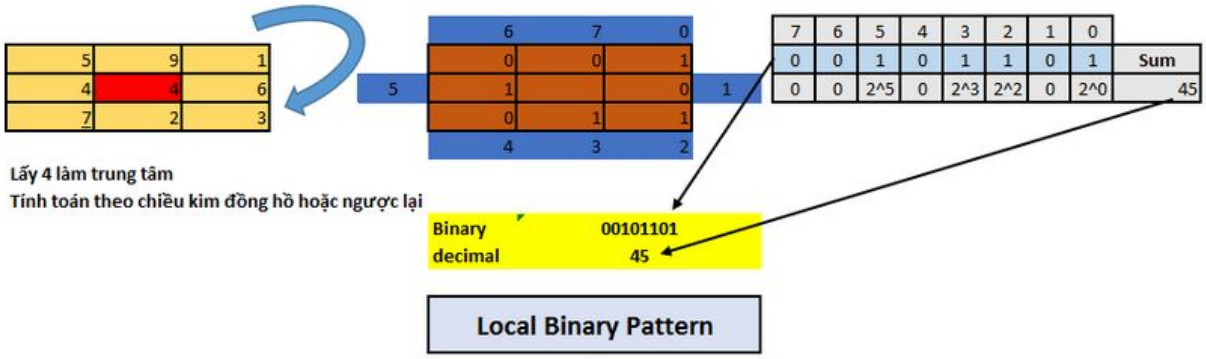
A. Vector normal:

Đây là các đơn giản nhất : Sau khi resize tất cả ảnh về cùng một kích thước (32* 32 *3). Coi mỗi pixel trong 1 bức ảnh là 1 feature. Resize bức ảnh về vector có kích thước (3072,1)

B. Local Binary Patterns: [1]

Local binary pattern nó là một thuật toán mô tả texture(cấu trúc) của một image. Ý tưởng cơ bản của nó là mô phỏng lại cấu trúc cục bộ (local texture) của image bằng cách so sánh mỗi pixel với các pixel lân cận nó(neighbors).Ta sẽ đặt một pixel là trung tâm(center) và so sánh với các pixel lân cận với nó, nếu pixel trung tâm lớn hơn hoặc bằng pixel lân cận thì nó sẽ trả về giá trị 1, ngược lại 0.

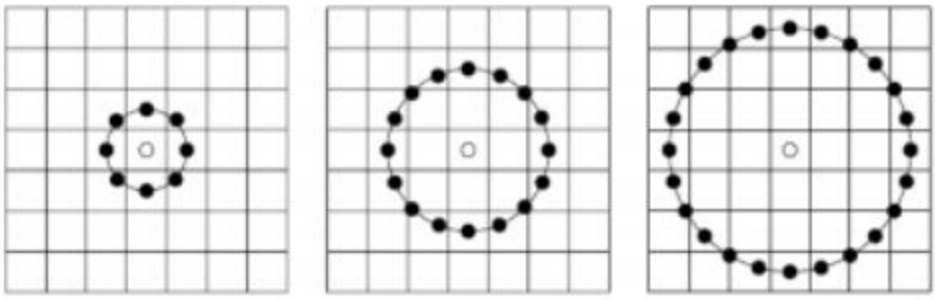
+ Ví dụ: chúng ta lấy bán kính 8 pixel lân cận thì lbp sẽ có dạng 11001111, là một chuỗi nhị phân để đơn giản và dễ đọc hơn ta sẽ chuyển về dạng decimal 207.

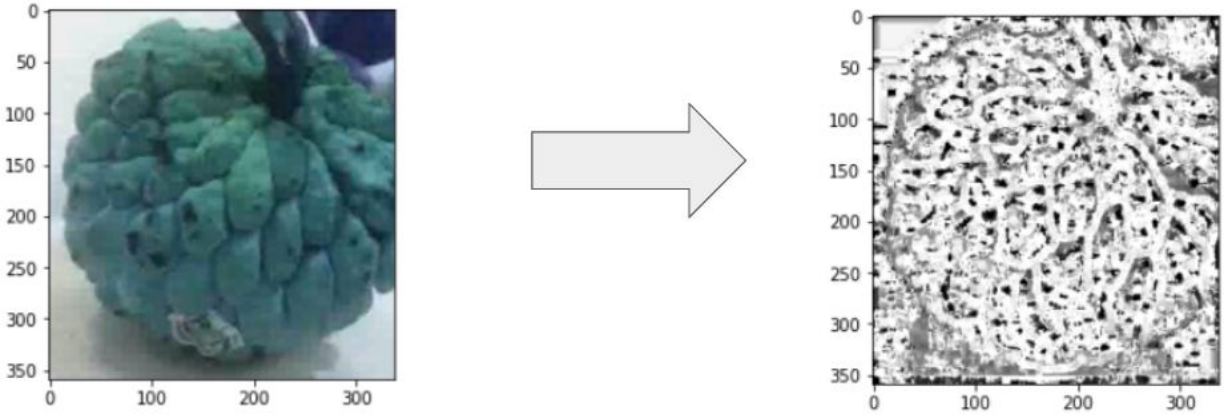


Hình 3: Ví dụ về cách tích trong Local Binary Pattern

+ Cách tính này có hạn chế đó là chỉ giới hạn 3x3 pixel không đủ để mô tả các cấu trúc large

scale nên người ta mở rộng khái niệm LBP bằng cách định nghĩa thêm 2 tham số là (P,R) trong P là số pixel lân cận xem xét và R là bán kính ta quét từ pixel trung tâm. Như hình bên dưới:





Hình 4: Ảnh bên trái là ảnh gốc, ảnh bên phải sau khi áp dụng LBP

C. Histogram Oriented of Gradient: [2]

+ HOG là viết tắt của Histogram of Oriented Gradient - một loại "feature descriptor". Mục đích của "feature descriptor" là trừu tượng hóa đối tượng bằng cách trích xuất ra những đặc trưng của đối tượng đó và bỏ đi những thông tin không hữu ích. Vì vậy, HOG được sử dụng chủ yếu để mô tả hình dạng và sự xuất hiện của một đối tượng trong ảnh.

+ Bản chất của phương pháp HOG là sử dụng thông tin về sự phân bố của các cường độ gradient (intensity gradient) hoặc của hướng biên (edge directions) để mô tả các đối tượng cục bộ trong ảnh.

IV Mô tả các thuật toán máy học

1. Chọn model

Về phần chọn model, tôi đề xuất thử với 2 model từ thư viện Sklearn và 2 model mạng Convolutional Neural Networks

- Đối với mỗi loại vector đặc trưng trong 3 kiểu vector normal, vector local binary patterns, vector hog, tôi sẽ sử dụng 5 model Sklearn sau:
    + LogisticRegression
    + DecisionTreeClassifier
    + KNeighborsClassifier
    + GaussianNB
    + SVM
- Đối với 2 model mạng Convolutional Neural Networks, tôi sẽ dụng 2 loại mạng sau với đầu vào của 2 loại mạng là ảnh có kích thước (32,32,3)
    + Mạng Shallownet: mạng neural 2 lớp
      Cấu trúc mạng như sau
      Input => Conv => RELU => FC
    + Mạng Lenet: Tôi sẽ thực hiện từ mạng Lenet gốc, được đề xuất bởi Yan LeCun trong một bài báo năm 1998 [3] [4]
      Cấu trúc mạng Lenet:

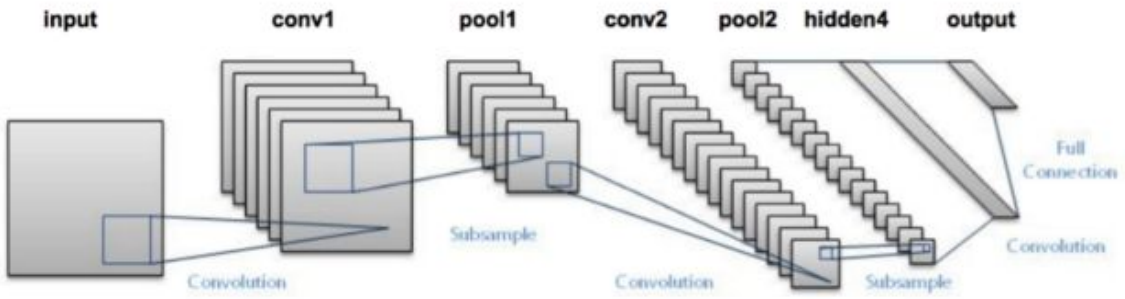


Figure 14.1: The LeNet architecture consists of two series of `CONV => TANH => POOL` layer sets followed by a fully-connected layer and softmax output. *Photo Credit: http://pyimg.co/ihjsx*

2 Các phương pháp đánh giá một hệ thống phân lớp [5]

Nhằm đánh giá hiệu quả của các thuật toán, tôi sẽ sử dụng một số phương pháp đo sau đây.

A . Accuracy

Đây là cách đơn giản giản và hay được sử dụng nhất . Cách đánh giá này đơn giản là tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử

+ Ví dụ: Ta có thể đếm được 6 điểm dữ liệu được dự đoán đúng trên tổng số 10 điểm. Vậy ta kết luận độ
+ chính xác của mô hình là 0.6 (hay 60%).

B. Confusion matrix

- Tuy nhiên cách tính sử dụng accuracy như ở trên chỉ cho chúng ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là confusion matrix.
- Về cơ bản, confusion matrix thể hiện có bao nhiêu điểm dữ liệu thực sự thuộc vào một class, và được dự đoán là rơi vào một class.
    + Ví dụ : Tập dữ liệu kiểm thử tổng cộng 10 điểm dữ liệu với 3 lớp dữ liệu. Ta xét ma trận tạo bởi các giá trị tại vùng 3x3 trung tâm của bảng

Total:10	Predicted as: 0	Predicted as: 1	Predicted as: 2	
True: 0	2	1	1	4
True: 1	1	2	0	3
True : 2	0	1	2	3

Ma trận thu được được gọi là confusion matrix. Nó là một ma trận vuông với kích thước mỗi chiều bằng số lượng lớp dữ liệu. Giá trị tại hàng thứ i, cột thứ j là số lượng điểm lẽ ra thuộc vào class i nhưng lại được dự đoán là thuộc vào class j. Như vậy, nhìn vào hàng thứ nhất (0), ta có thể thấy được rằng trong số bốn điểm thực sự thuộc lớp 0, chỉ có hai điểm được phân loại đúng, hai điểm còn lại bị phân loại nhầm vào lớp 1 và lớp 2.

C. True/False Positive/Negative

- Cách đánh giá này thường được áp dụng cho các bài toán phân lớp có hai lớp dữ liệu. Cụ thể hơn, trong hai lớp dữ liệu này có một lớp nghiêm trọng hơn lớp kia và cần được dự đoán chính xác. Ví dụ, trong bài toán xác định có bệnh ung thư hay không thì việc không bị sót (miss) quan trọng hơn là việc chẩn đoán nhầm âm tính thành dương tính. Trong bài toán xác định có mìn dưới lòng đất hay không thì việc bỏ sót nghiêm trọng hơn việc báo động nhầm rất nhiều. Hay trong bài toán lọc email rác thì việc cho nhầm email quan trọng vào thùng rác nghiêm trọng hơn việc xác định một email rác là email thường.
- Trong những bài toán này, người ta thường định nghĩa lớp dữ liệu quan trọng hơn cần được xác định đúng là lớp Positive (P-dương tính), lớp còn lại được gọi là Negative (N-âm tính). Ta định nghĩa

True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN) dựa trên confusion matrix chưa chuẩn hoá như sau:

	Predicted as Positive	Predicted as Negative
Actual: Positive	True Positive (TP)	False Nagative (FN)
Actual: Negative	False Positive (FP)	True Negative (TN)

- Người ta thường quan tâm đến TPR, FNR, FPR, TNR (R - Rate) dựa trên normalized confusion matrix như sau:

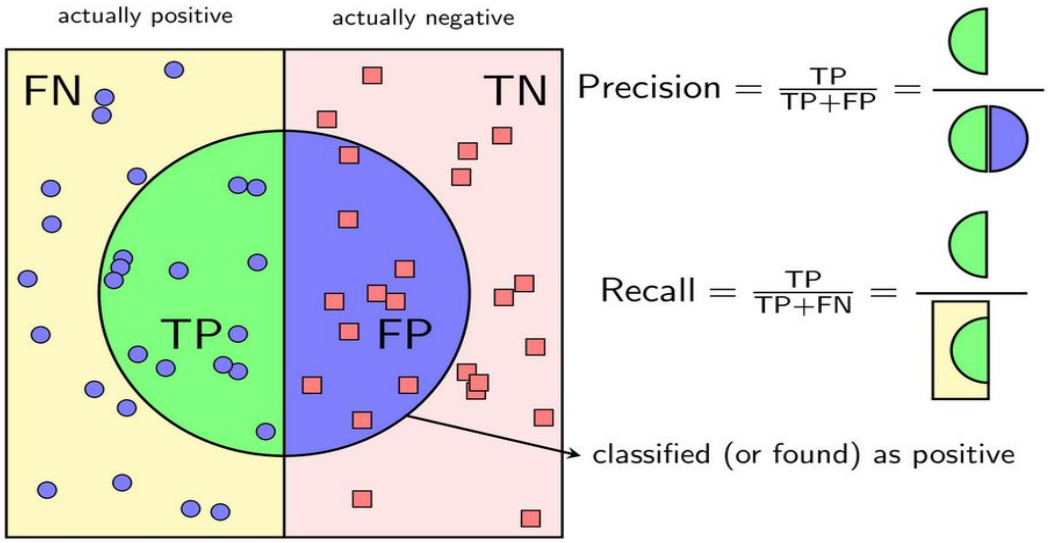
	Predicted as Positive	Predicted as Negative
Actual: Positive	TPR = TP/(TP + FN)	FNR = FN/(TP + FN)
Actual: Negative	FPR = FP/(FP + TN)	TNR = TN/(FP + TN)

- False Positive Rate còn được gọi là False Alarm Rate (tỉ lệ báo động nhầm), False Negative Rate còn được gọi là Miss Detection Rate (tỉ lệ bỏ sót). Trong bài toán dò mìn, thà báo nhầm còn hơn bỏ sót, tức là ta có thể chấp nhận False Alarm Rate cao để đạt được Miss Detection Rate thấp.

D. Precision và Recall

- Với bài toán phân loại mà tập dữ liệu của các lớp là chênh lệch nhau rất nhiều, có một phép đo hiệu quả thường được sử dụng là Precision-Recall.

- Trước hết xét bài toán phân loại nhị phân. Ta cũng coi một trong hai lớp là positive, lớp còn lại là negative.



Hình 5: Ví dụ minh hoạ về precision và recall

- Với một cách xác định một lớp là positive, Precision được định nghĩa là tỉ lệ số điểm true positive trong số những điểm được phân loại là positive (TP + FP).
- Recall được định nghĩa là tỉ lệ số điểm true positive trong số những điểm thực sự là positive (TP + FN).
- Khi Recall = 1, mọi điểm positive đều được tìm thấy. Tuy nhiên, đại lượng này lại không đo liệu có bao nhiêu điểm negative bị lẫn trong đó. Nếu mô hình phân loại mọi điểm là positive thì chắc chắn Recall = 1, tuy nhiên dễ nhận ra đây là một mô hình cực tồi.
- Một mô hình phân lớp tốt là mô hình có cả Precision và Recall đều cao, tức càng gần một càng tốt. Có cách đo chất lượng của bộ phân lớp dựa vào Precision và Recall: F1-score

$$\frac{2}{F_1} = \frac{1}{\text{precision}} + \frac{1}{\text{recall}}$$

3. Cross-validation [6]

- Để đánh giá model một cách khách quan, thay vì chia tập train thành 2 tập train và tập validation với kích thước cố định, cross validation chia tập training ra k tập con không có phần tử chung, có kích thước gần bằng nhau. Tại mỗi lần kiểm thử , được gọi là run, một trong số k tập con được lấy ra làm validataset. Mô hình sẽ được xây dựng dựa vào hợp của k−1 tập con còn lại. Mô hình cuối được xác định dựa trên trung bình của các train error và validation error. Cách làm này còn có tên gọi là k-fold cross validation.

Định nghĩa Stratified Cross Validation

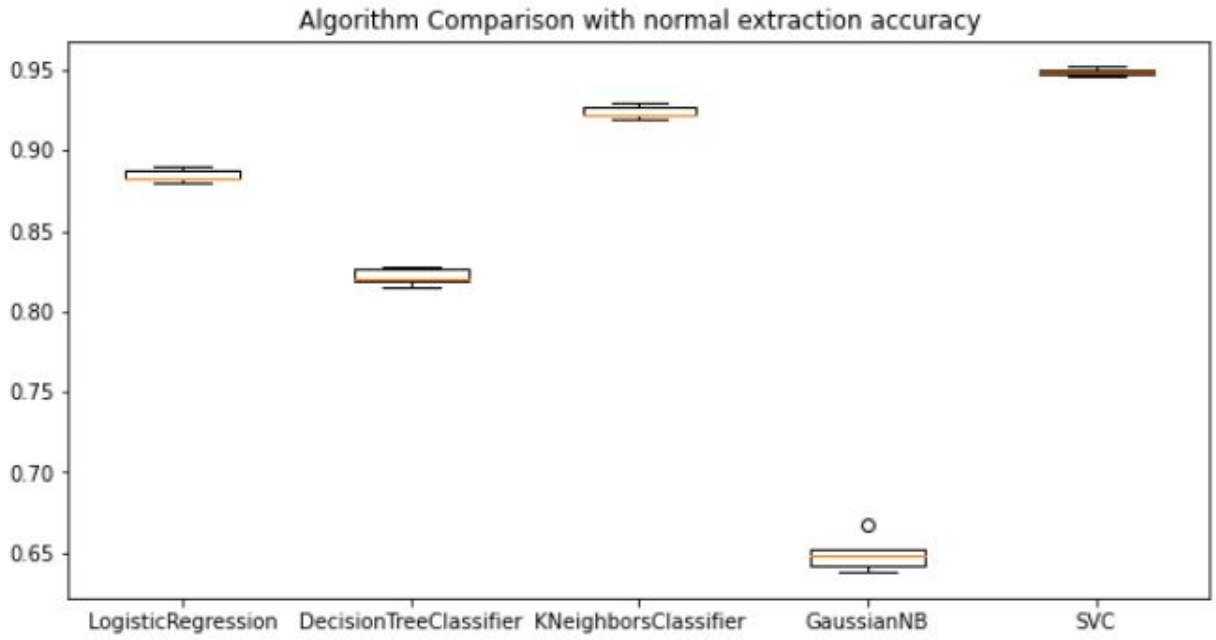
- Là phiên bản cải tiến của Cross validation, thay vì chia ngẫu nhiên tập dữ liệu train thành k fold (Cross validation), Cross Validation chia tập dữ liệu train thành k fold theo một phân phối xác suất (mean, variance,...)

4. Huấn luyện

Sau khi xác định các loại model sẽ dùng và thang độ đo được sử dụng để đo hiệu quả của model, tôi tiến hành train và đánh giá model trên tập dữ liệu training

A. Năm model Sklearn

A1. Dùng vector Normal

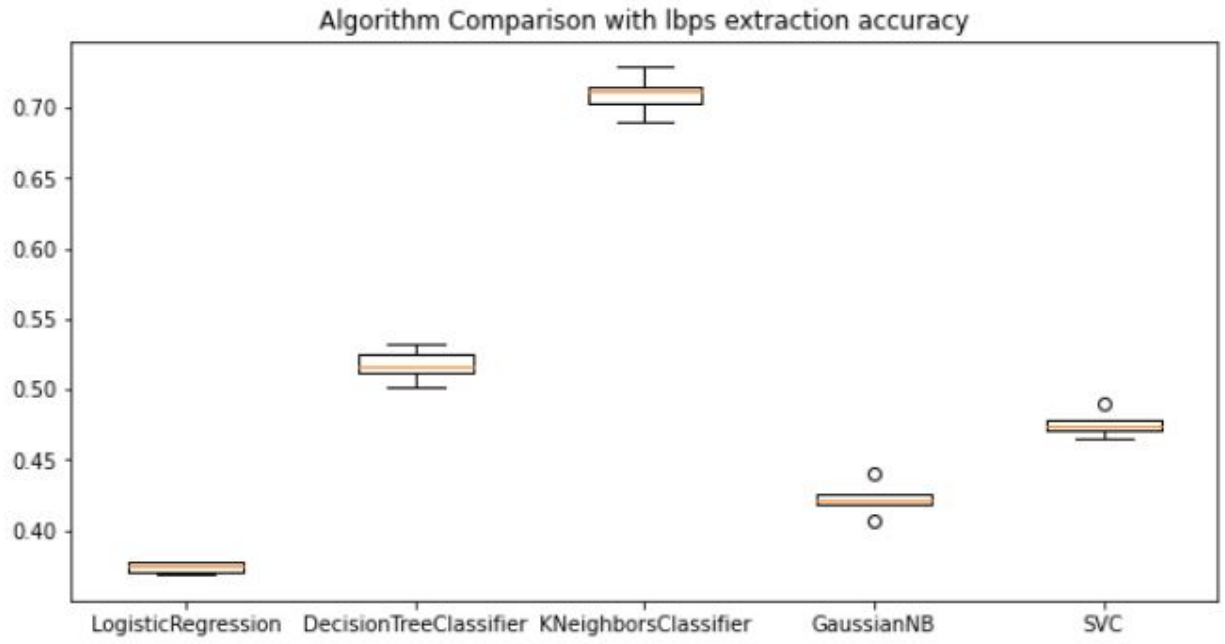


Hình 6: Kết quả trong quá trình huấn luyện 5 loại model sử dụng vector normal

Độ chính xác của các model:

+ LogisticRegression: 0.8844023323615161
+ DecisionTreeClassifier: 0.8215014577259476
+ KNeighborsClassifier: 0.9242711370262391
+ GaussianNB: 0.6494169096209912
+ SVC: 0.9487609329446064

A2. Dùng vector local binary patterns

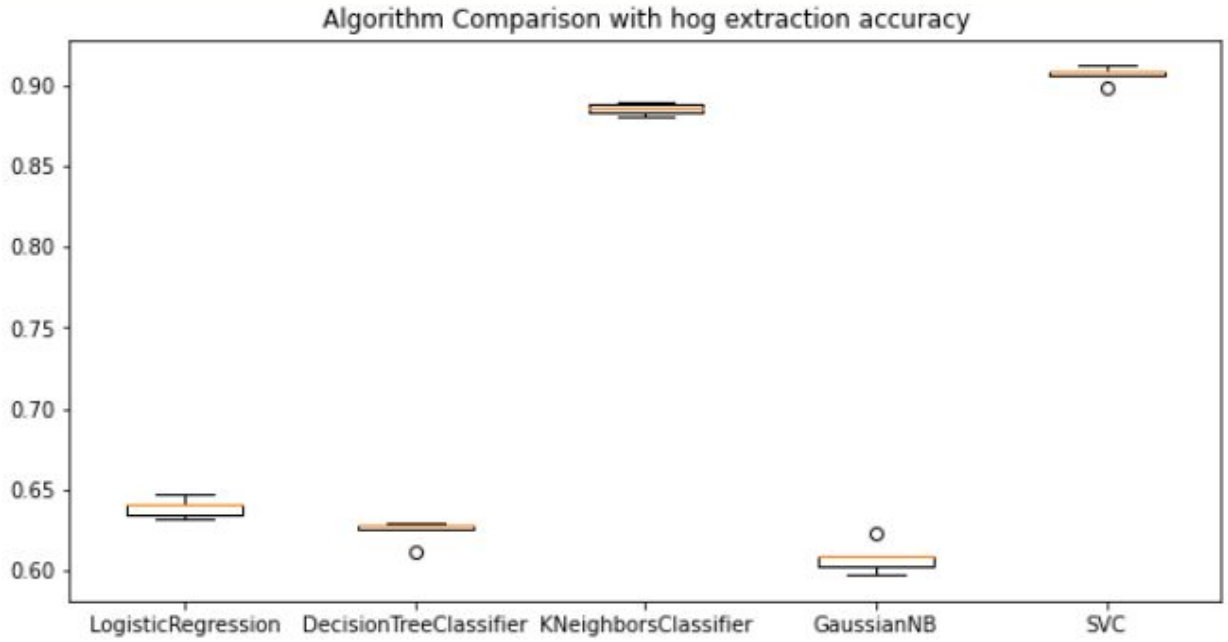


Hình 7: Kết quả trong quá trình huấn luyện 5 loại model sử dụng vector lbps

Độ chính xác của các model:

+ LogisticRegression: 0.37339650145772596
+ DecisionTreeClassifier: 0.517201166180758
+ KNeighborsClassifier: 0.7098396501457724
+ GaussianNB: 0.42274052478134105
+ SVC: 0.4759475218658892

A3. Dùng vector Histogram Oriented of Gradient



Hình 8: Kết quả trong quá trình huấn luyện 5 loại model sử dụng vector HOG

Độ chính xác của các model:

+ LogisticRegression: 0.63899416909621
+ DecisionTreeClassifier: 0.6244169096209913
+ KNeighborsClassifier: 0.8854956268221574
+ GaussianNB: 0.6079446064139942
+ SVC: 0.9069241982507288

B Hai model mạng Convolution Neural Network

B1. Mạng Shallownet

```
[INFO] serializing network ...
[INFO] evaluating network...
              precision    recall  f1-score   support

       Apple       0.97      0.98      0.98       242
     Avocado       0.97      0.96      0.97       299
      Banana       0.97      0.95      0.96       266
     Coconut       0.99      1.00      0.99       228
Custard_apple       0.99      0.98      0.99       182
 Dragon_fruit       0.98      0.95      0.97       137
       Guava       0.94      0.93      0.94       181
       Mango       0.92      0.94      0.93       184
      Orange       0.96      0.96      0.96       320
        Plum       0.98      1.00      0.99       287
 Start_fruit       0.98      0.98      0.98       233
  Watermelon       0.98      0.99      0.99       185

    accuracy                           0.97      2744
   macro avg       0.97      0.97      0.97      2744
weighted avg       0.97      0.97      0.97      2744
```

Hình 9: Kết quả trong quá trình huấn luyện mạng Shallownet

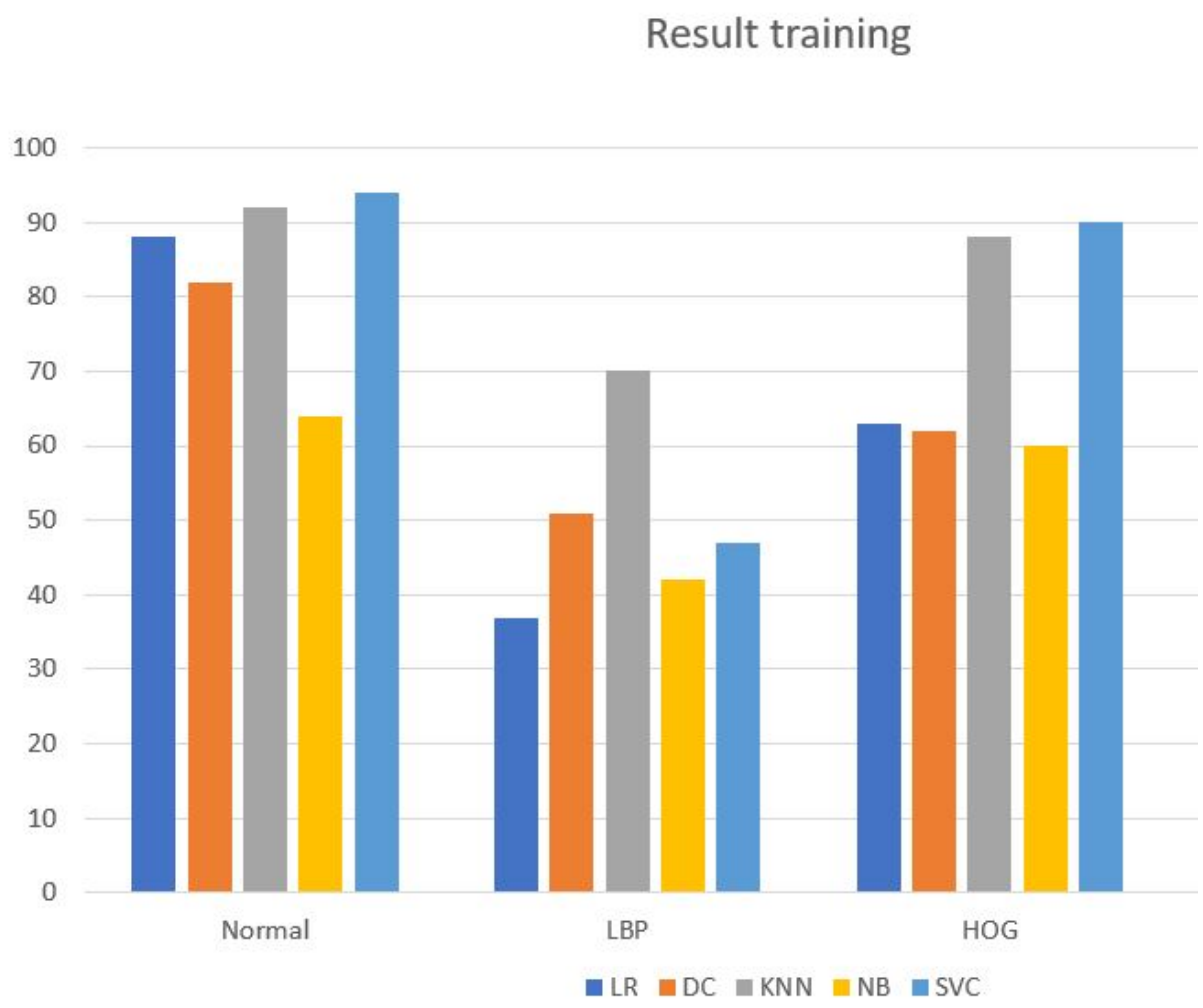
B2. Mạng Lenet

```
[INFO] serializing network ...
[INFO] evaluating network...
              precision    recall  f1-score   support

       Apple       1.00      0.98      0.99       242
     Avocado       0.98      0.99      0.98       299
      Banana       0.98      0.98      0.98       266
     Coconut       0.98      0.99      0.99       228
Custard_apple       0.99      0.98      0.99       182
 Dragon_fruit       0.98      0.96      0.97       137
       Guava       0.99      0.98      0.99       181
       Mango       0.98      0.97      0.98       184
      Orange       0.99      0.99      0.99       320
        Plum       0.98      1.00      0.99       287
  Start_fruit       1.00      0.99      1.00       233
  Watermelon       0.99      0.99      0.99       185

    accuracy                           0.99      2744
   macro avg       0.99      0.98      0.98      2744
weighted avg       0.99      0.99      0.99      2744
```

Hình 10: Kết quả trong quá trình huấn luyện mạng Lenet

5. Đánh giá

5A.Kết quả khi train tập huấn luyện trên 5 model Sklearn



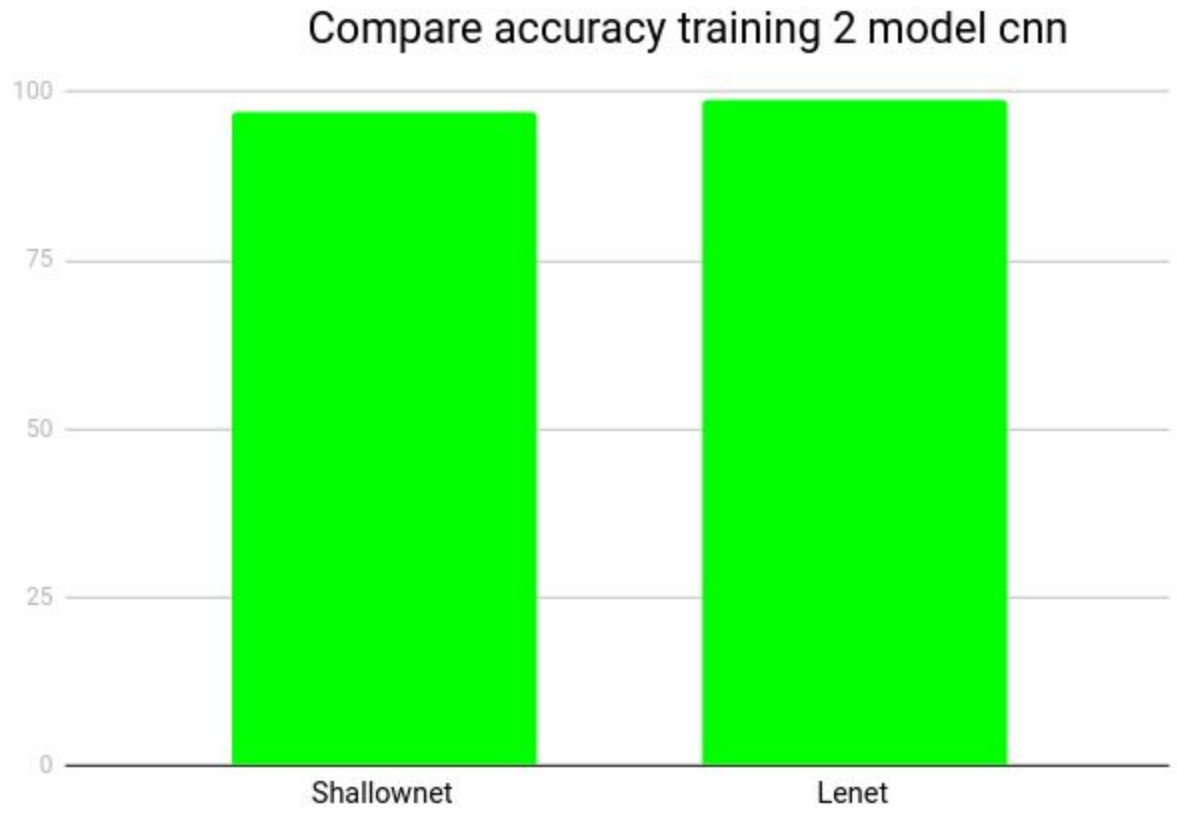
Hình 4.5.1 Biểu đồ tổng quan kết quả huấn luyện trên 5 model Sklearn

- Đánh giá về việc dùng vector đặc trưng khác nhau:
    + Sử dụng vector đặc trưng LBP cho cả 5 model sẽ cho độ chính xác thấp hơn so với sử dụng vector đặc trưng Normal và HOG. Do LBP là một thuật toán mô tả texture(cầu trúc) của một image, mà bộ dữ liệu trái cây, tất cả các trái đều có hình dạng tròn, điều đó khó phân biệt được các loại trái cây khác nhau

    + Sử dụng vector đặc trưng Normal cho cả 5 model có độ chính xác cao hơn so với vector HOG. Do resize ảnh ban đầu có kích thước 32 x 32 sẽ làm mất nhiều thông tin của bức ảnh (mất cạnh của loại trái có trong bức ảnh), nên việc dùng HOG để rút trích đặc trưng sẽ không hiệu quả bằng việc dùng vector normal (coi mỗi pixel của bức ảnh là một feature)

- Đánh giá về việc dùng nhiều loại model khác nhau:
    + Khi sử dụng vector Normal và vector HOG, model SVC cho ra độ chính xác cao nhất trong cả 5 model đã sử dụng
    + Tuy nhiên khi sử dụng vector LBP, model KNN cho ra độ chính xác cao nhất.
    + Trong cả 3 vector rút trích đặc trưng, model Gaussian Naive Bayes có độ chính xác thấp nhất, do model này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục, mà bài toán tôi đang giải quyết là bài toán có các biến rời rạc.

5B. Kết quả khi train tập huấn luyện trên 2 mạng cnn



Hình 11: Biểu đồ tổng quan kết quả huấn luyện trên 2 mạng CNN

Cả 2 mạng tôi đều cài đặt số lượng epoches = 35, sử dụng Stochastic Gradient Descent với learning_rate = 0.05. Kết quả độ chính xác của mạng Lenet cao hơn Shallownet 2% (99% so với 97%). Lí do là mạng Lenet có nhiều lớp hơn mạng Shallownet nên mạng Lenet học được nhiều đặc trưng ảnh hơn, từ đó dẫn đến độ chính xác cao hơn.

V. Cài đặt và tinh chỉnh tham số

* Từ 5 loại model sklearn train trên 3 loại vector normal, vector local binary patterns, HOG, tôi rút ra model có chỉ số accuracy tốt nhất trên mỗi loại vector đặc trưng đem đi tinh chỉnh tham số.

- Model SVM trên vector normal cho kết quả tốt nhất so với 4 model còn lại cùng sử dụng vector normal : 94.87%

- Model KNN trên vector local binary patterns cho kết quả tốt nhất so với 4 model còn lại cùng sử dụng vector local binary patterns : 70.98%
- Model SVM trên vector HOG cho kết quả tốt nhất so với 4 model còn lại cùng sử dụng vector HOG : 90.69%

* Đối với model mạng cnn Shallownet, Lenet do kết quả đánh giá sau khi train đạt lần lượt 97%, 99% nên tôi sẽ không tinh chỉnh tham số

1. RandomizedSearchCV

RandomizedSearchCV là một phương pháp tìm bộ hyperparameters bằng cách từ những giá trị parameter đã setting, Random Search sẽ chọn ngẫu nhiên các cặp parameter sao cho độ chính xác của tập cần dự đoán là lớn nhất theo một độ đo nhất định nào đó (F1-score, Accuracy, ROC,..)

Ưu điểm : RandomizedSearchCV thường có thời gian chạy nhanh

Nhược điểm: Vì random một số bộ parameter nên có khi kết quả không được tối ưu, và dẫn đến không được như ý muốn

2. Cài đặt

- Sử dụng RandomizedSearchCV cùng với StratifiedKFold(n_split=5) để tuning model

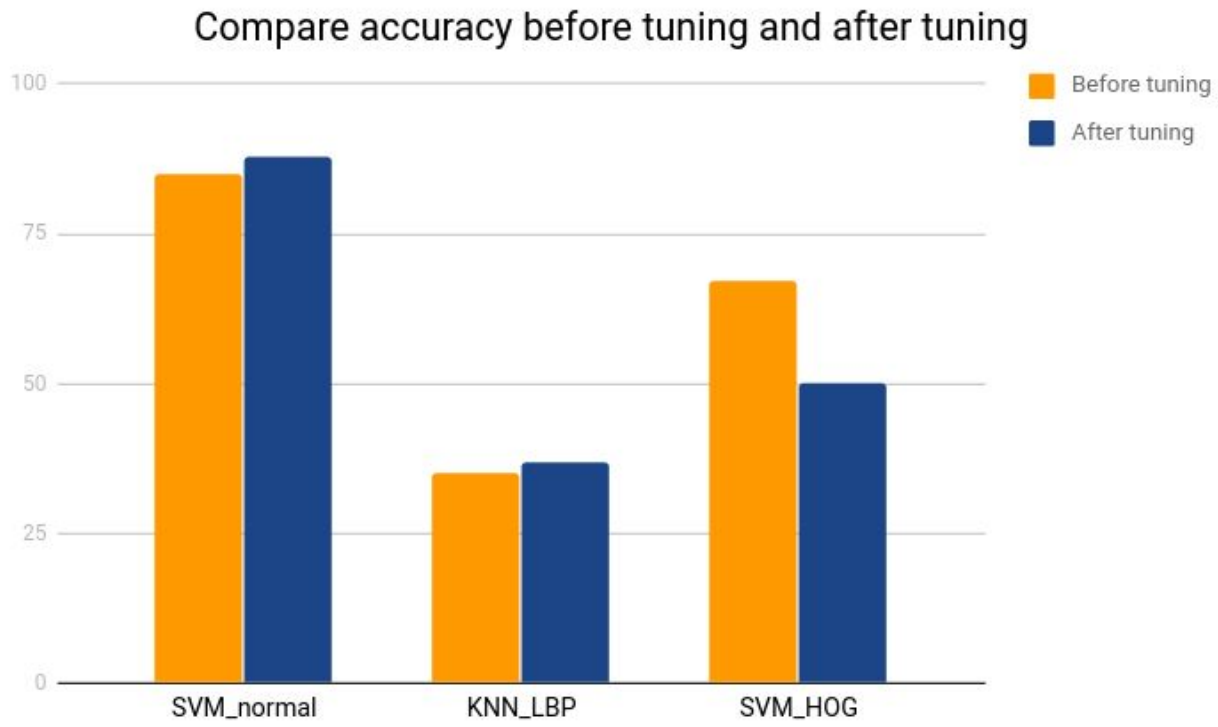
- Đối với model KNN tinh chỉnh bộ tham số:
    + leaf_size = list(range(1,50)),
    + n_neighbors = list(range(1,30)),
    + p=[1,2]

- Đối với model SVM tinh chỉnh bộ tham số:
    + C: [0.1 , 1],

+ gamma: [1,0.1],
+ kerner: [‘linear’,’poly’,’rbf’]

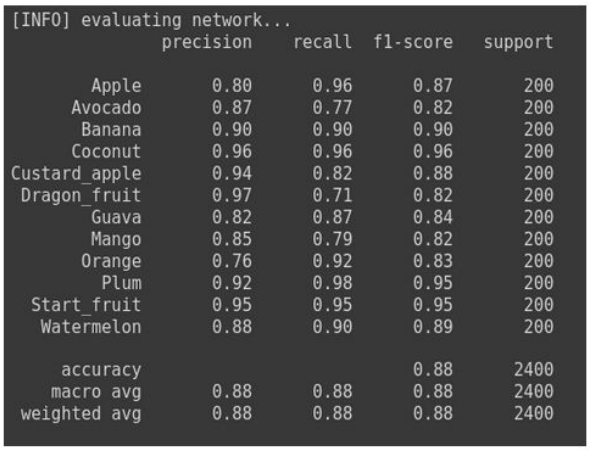
3. Kết quả trước khi tuning và sau khi tuning model sklearn

- Để đánh giá kết quả hiệu quả của model trước khi tuning và sau khi tuning, tôi sử dụng bộ dữ liệu test thứ 1 (gồm 2400 bức ảnh).

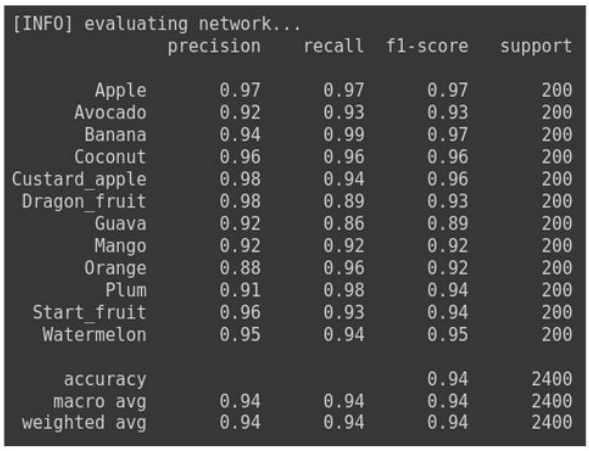


Hình 12: So sánh độ chính xác của model trước khi tuning và sau khi tuning

- Tôi cũng sẽ thực hiện việc đo độ chính xác của 2 mạng Shallownet và Lenet trên bộ data test thứ nhất



```
[INFO] evaluating network...
              precision    recall  f1-score   support

       Apple       0.80      0.96      0.87       200
     Avocado       0.87      0.77      0.82       200
      Banana       0.90      0.90      0.90       200
     Coconut       0.96      0.96      0.96       200
Custard_apple       0.94      0.82      0.88       200
 Dragon_fruit       0.97      0.71      0.82       200
       Guava       0.82      0.87      0.84       200
       Mango       0.85      0.79      0.82       200
      Orange       0.76      0.92      0.83       200
        Plum       0.92      0.98      0.95       200
  Start_fruit       0.95      0.95      0.95       200
   Watermelon       0.88      0.90      0.89       200

    accuracy                           0.88      2400
   macro avg       0.88      0.88      0.88      2400
weighted avg       0.88      0.88      0.88      2400
```



```
[INFO] evaluating network...
              precision    recall  f1-score   support

       Apple       0.97      0.97      0.97       200
     Avocado       0.92      0.93      0.93       200
      Banana       0.94      0.99      0.97       200
     Coconut       0.96      0.96      0.96       200
Custard_apple       0.98      0.94      0.96       200
 Dragon_fruit       0.98      0.89      0.93       200
       Guava       0.92      0.86      0.89       200
       Mango       0.92      0.92      0.92       200
      Orange       0.88      0.96      0.92       200
        Plum       0.91      0.98      0.94       200
  Start_fruit       0.96      0.93      0.94       200
   Watermelon       0.95      0.94      0.95       200

    accuracy                           0.94      2400
   macro avg       0.94      0.94      0.94      2400
weighted avg       0.94      0.94      0.94      2400
```

Hình 13: Hình bên trái là kết quả của model mạng Shallownet, hình bên phải là kết quả của model mạng Lenet

Kết quả : Mạng Lenet cho độ chính xác cao hơn so với mạng Shallownet (94% so với 88%)

4. Nhận xét:

- Sau khi tuning, model SVM sử dụng vector Normal có độ chính xác cải thiện hơn hơn trước khi tuning (lần lượt 85% - 88%).
- Tương tự, sau khi tuning, model KNN sử dụng vector LBP có độ chính xác cao hơn trước khi tuning (lần lượt 35% - 39%)
- Riêng model SVM sử dụng vector HOG sau khi tuning thấp hơn so với trước khi tuning (lần lượt 67% - 50%), do RandomizedSearchCV chỉ chọn random một số bộ parameter để lấy ra bộ parameter cho kết quả tốt nhất chứ không thể vét cạn tất cả bộ parameter giống GridSearchCV

VI. Đánh giá kết quả, kết luận

Sau quá trình tuning model sklearn, tôi chọn ra model SVM sử dụng vector normal ; đối với model CNN , tôi chọn ra model mạng Lenet. Cả 2 model đều được đánh giá trên bộ test thứ hai (gồm 120 bức hình)

	precision	recall	f1-score	support
Apple	0.50	0.64	0.56	11
Avocado	0.33	0.10	0.15	10
Banana	0.26	0.60	0.36	10
Coconut	0.00	0.00	0.00	9
Custard_apple	0.20	0.10	0.13	10
Dragon_fruit	0.29	0.50	0.37	10
Guava	0.09	0.10	0.10	10
Mango	0.15	0.20	0.17	10
Orange	0.27	0.30	0.29	10
Plum	0.50	0.20	0.29	10
Start_fruit	0.42	0.50	0.45	10
Watermelon	0.33	0.20	0.25	10
accuracy			0.29	120
macro avg	0.28	0.29	0.26	120
weighted avg	0.28	0.29	0.27	120

Hình 6.1 Kết quả đánh giá của model SVM sử dụng vector normal trên bộ data 120 bức ảnh

	precision	recall	f1-score	support
Apple	0.71	0.45	0.56	11
Avocado	0.67	0.20	0.31	10
Banana	0.25	0.70	0.37	10
Coconut	0.17	0.11	0.13	9
Custard_apple	0.67	0.20	0.31	10
Dragon_fruit	0.42	1.00	0.59	10
Guava	0.25	0.10	0.14	10
Mango	0.07	0.10	0.08	10
Orange	0.27	0.40	0.32	10
Plum	0.00	0.00	0.00	10
Start_fruit	0.17	0.10	0.12	10
Watermelon	0.29	0.20	0.24	10
accuracy			0.30	120
macro avg	0.33	0.30	0.26	120
weighted avg	0.33	0.30	0.27	120

Hình 6.2 Kết quả đánh giá của model mạng Lenet trên bộ data 120 bức ảnh

1. Nhận xét

- Từ hai biểu biểu đồ trên ta thấy độ chính xác khi sử dụng model SVM trên vector normal đạt độ chính xác 29%, độ chính xác khi sử dụng model mạng Lenet đạt 30%.
- Trong khi cũng 2 model đó khi đánh giá model SVM - vector normal trên bộ data train (13720 ảnh) và bộ data test thứ nhất (2400 ảnh) cho ra kết quả lần lượt là 94% - 88% , tương tự với model mạng Lenet là (99% - 94%)
- Điều đó chứng tỏ 2 model của tôi bị overfiting, có 4 nguyên nhân dẫn đến hiện tượng này:
    + Lúc train, đầu vào model của tôi nhận được bức ảnh chỉ chứa 1 loại trái cây, bộ data test thứ nhất bức ảnh chỉ chứa 1 loại trái cây, trong khi ở bộ data test thứ hai có những bức chứa một trùm trái cây

+ Bộ dữ liệu validation (lấy từ tập train) với data test thứ nhất tôi đã loại bỏ background gây nhiễu, trong khi bộ data test thứ hai có nhiều ảnh có background gây nhiễu
+ Bộ dữ liệu train do đa số ảnh được cắt từ frame video, nên có rất nhiều ảnh trùng nhau, nên model không học được nhiều đặc trưng, dẫn đến hiệu quả không cao khi sử dụng ngoài thực tế
+ Bộ dữ liệu train có chất lượng ảnh không tốt, trong khi bộ dữ liệu test có chất lượng ảnh tốt

2. Cách khắc phục:

- Thu thập thêm nhiều bức ảnh có chất lượng ảnh rõ nét bổ sung vào data train
- Chụp nhiều bức ảnh có độ đa dạng hơn, thay vì bức ảnh chứa một trái thì bức ảnh sẽ chứa một trùm trái
- Áp dụng những kĩ thuật tiên tiến, tôi đang thử nghiệm dùng **Transfer learning** cho đề tài của tôi, nhưng do tôi dùng mạng VGG16 đã được giải quyết cho bài toán phân loại khác, khi áp dụng vô bài toán của tôi, bắt buộc phải resize ảnh về 224x224, và số lượng bộ data train của tôi nhiều, train trên colab nên dẫn tới tràn bộ nhớ gram.

3.Kết luận

Qua việc sử dụng 5 model Sklearn và 2 mạng CNN cùng với 3 loại rút trích đặc trưng khác nhau, tôi rút ra một số kinh nghiệm sau:

- Không nên sử dụng model Gaussian Naive Bayes cho bài toán có biến rời rạc, lí do tôi đã giải thích ở (Phần IV, mục 5A)
- Sử dụng mạng CNN có số lớp càng sẽ cho kết quả cao hơn so với mạng có số lớp ít , đã chứng minh kết quả ở (Phần IV, mục 5B)
- Tùy bài toán có thể sử dụng model Sklearn kết hợp rút trích đặc trưng cho phù hợp với bài toán. Ví dụ như đề tài tôi đang nghiên cứu dùng vector LBP thấp hơn hẳn so với 2 đặc trưng khác. Dùng SVM

kết hợp với vector normal cho kết quả cao nhất, lí do tôi đã giải thích ở (Phần IV, mục 5A)

- Dùng RandomSearchCV để tinh chỉnh tham số cho model Sklearn nhanh hơn dùng GridSearchCV nhưng có một số trường hợp cho kết quả tuning thấp hơn so với trước tuning, đã chứng minh kết quả ở (Phần V, mục 3)

VII Phát triển phần mềm

Tôi sẽ xây dựng 2 phần mềm ứng dụng demo đồ án nhận diện trái cây:

1. Demo trên giao diện web trên framework flask
2. Demo trên giao diện app trên framework tkinkter

Link demo :
https://www.youtube.com/watch?v=Xtyh55GQP-E&feature=youtu.be

VIII Nguồn tham khảo

[1] https://thorpham.github.io/blog/2018/02/22/feature-exaction/
[2]Tìm hiểu về phương pháp mô tả đặc trưng HOG (Histogram of Oriented Gradients)
[3] http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
[4] Sách Deep Learning for Compter Vision (Adrian RoseBrock)
[5] https://machinelearningcoban.com/2017/08/31/evaluation/
[6]https://stats.stackexchange.com/questions/49540/understanding-stratified-cross-validation?fbclid=IwAR2e1X1Sp-81Hy5E3Ooe9BQ_cb_arElXr0SWp8QxZ_S46T1o-zBd651PjF0
[7] https://www.kaggle.com/moltean/fruits