BÁO CÁO DỰ ÁN
PHÂN LOẠI 26 LOẠI CÂY KHU VỰC ĐÔNG NAM Á

MÔN DSP305X

SINH VIÊN: LƯU TRUNG HIẾU

MENTOR HƯỚNG DẪN: NGUYỄN HẢI NAM

- hieultfx10546@funix.edu.vn

TỔNG QUAN

Khu vực Đông Nam Á được thiên nhiên ưu ái có hệ động thực vật đa dạng, nhiều loài có giá trị kinh tế cao. Nền kinh tế của khu vực này phụ thuộc rất lớn vào nông nghiệp, tuy nhiên không phải ai cũng phân biệt được các loại cây trồng với nhau. Áp dụng học máy và học sâu vào bài toán phân loại các loại cây trồng khu vực Đông Nam Á, có thể giúp cho người dân khu vực này và cũng như người dân trên thế giới có thể ứng dụng vào sản xuất, kinh doanh trong thực tế

Agenda (Mục lục)

- I. Exploratory data analysis (Phân tích dữ liệu khám phá)
- II. Modelling & Evaluation (Lập mô hình & Đánh giá)
- III. Model improvement (Cải thiện mô hình)
- IV. Model Serving (triển khai mô hình)
- V. Conclusion (Kết luận)

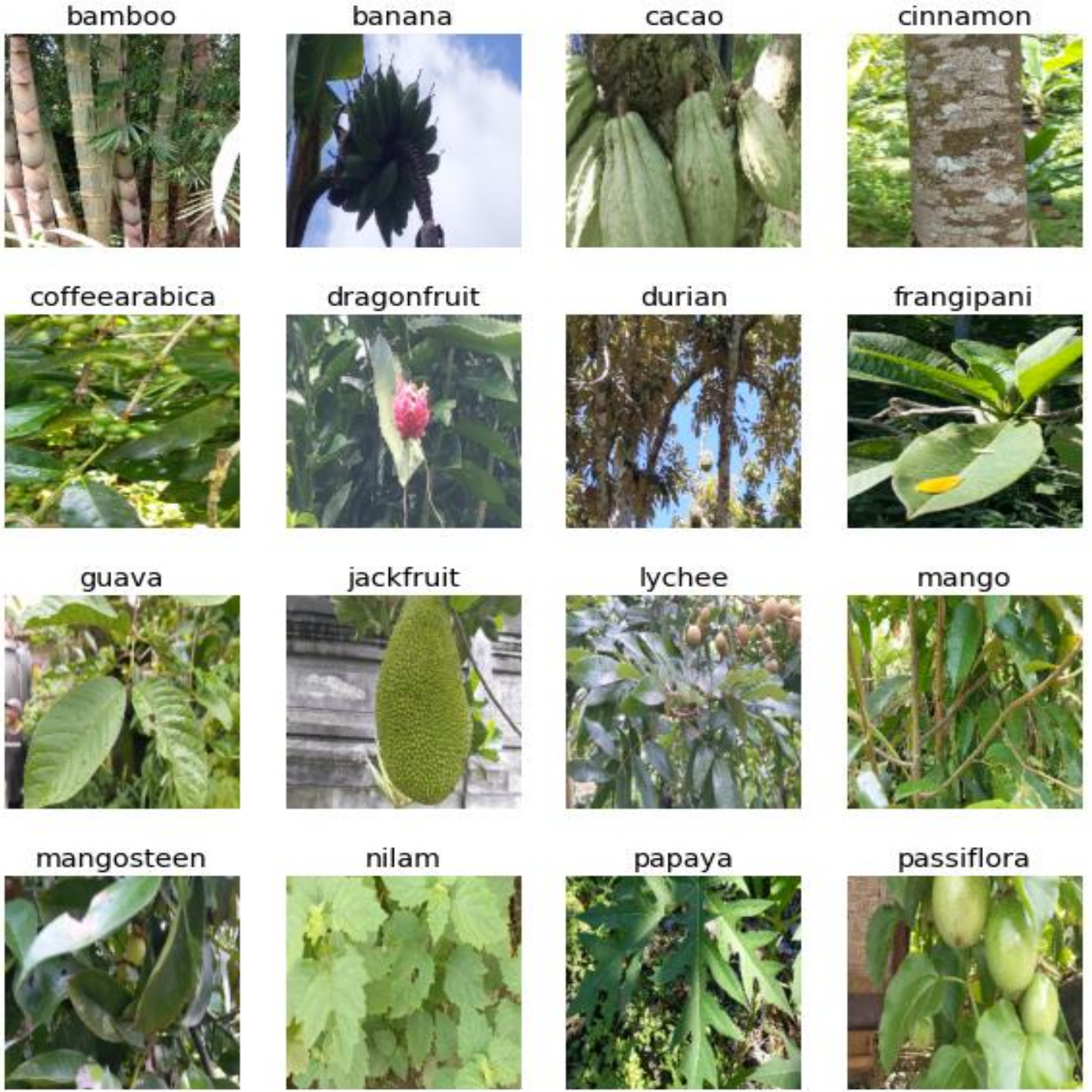
I. Exploratory data analysis (Phân tích dữ liệu khám phá)

1. **HIỂU BIẾT VỀ KINH DOANH (BUSINESS UNDERSTANDING)**
2. **HIỂU BIẾT VỀ DỮ LIỆU (DATA UNDERSTANDING)**
3. **PHÂN TÍCH DỮ LIỆU (DATA ANALYSIS)**
4. **PHƯƠNG PHÁP GIẢI QUYẾT VẤN ĐỀ (SOLUTION APPROACH)**
5. **TIỀN XỬ LÝ DỮ LIỆU (PREPROCESSING)**

BUSINESS UNDERSTANDING

Mục đích bài toán: Xác định 26 loại cây trồng dựa trên tập hình ảnh=> đây là bài toán Phân loại nhiều lớp

Mô hình sử dụng accuracy-score (mục tiêu 99%)



HIEULTFX10546@FUNIX.EDU.VN

2. Data Understanding

(Hiểu biết về Dữ liệu)

- Tập Train có tất cả 41607 hình ảnh được dán nhãn của 26 loại cây. Tập dữ liệu được lấy từ cuộc thi của kaggle
- Các cây được chụp ở các vị trí khác nhau: thân, hoa, quả, lá, ….
- Cấu trúc dữ liệu hình ảnh được tổ chức dưới dạng một ma trận (số học) 3 chiều (height, width, channel). Mỗi phần tử trong ma trận này có kiểu dữ liệu là số nguyên (0-255) mô tả giá trị của mức sáng

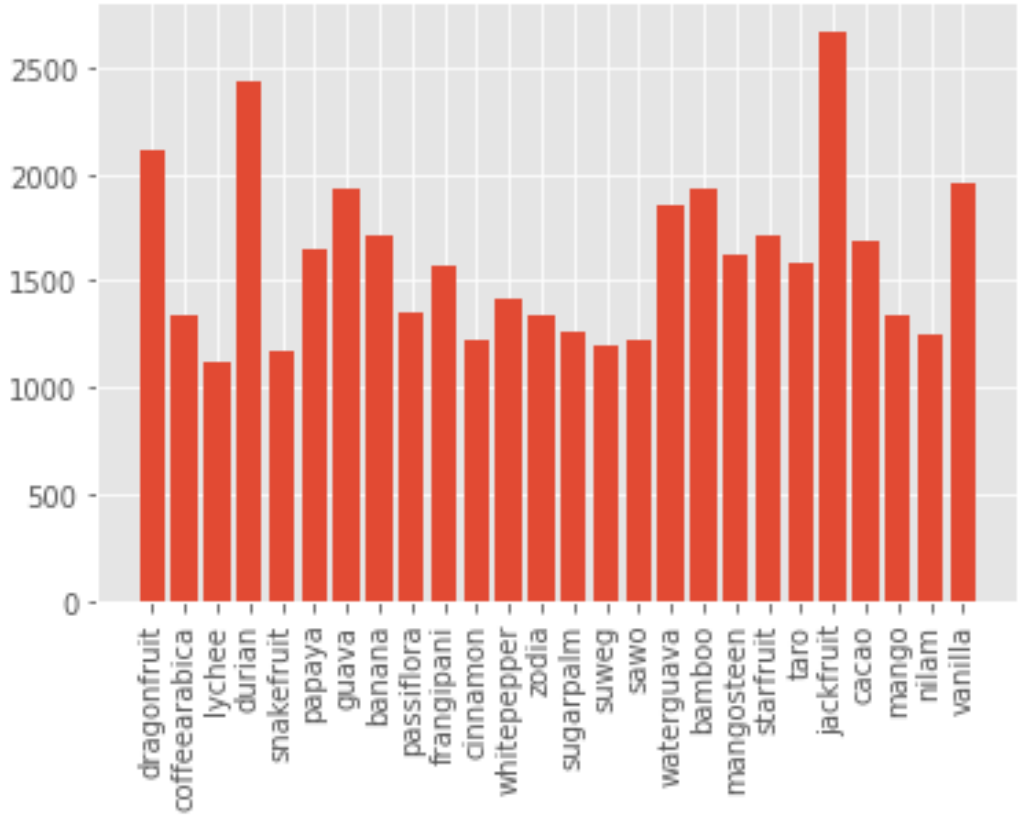
3. DATA ANALYSIS (PHÂN TÍCH DỮ LIỆU)

- Nhóm có số lượng ảnh ít nhất là cây vải (lychee) với khoảng 1,100 ảnh
- Nhóm có số lượng ảnh nhiều nhất là cây mít (jackfruit) với khoảng 2,600 ảnh
- Các nhóm không có sự chênh lệch về số lượng quá đáng kể
- Các ảnh chủ yếu là có kích thước 1920x1080

đối với ảnh chụp thẳng đứng hoặc 1080 x 1920

đối với ảnh chụp ngang, cần đưa ảnh về cùng

kích thước để model có thể train được



4. SOLUTION APPROACH (PHƯƠNG PHÁP GIẢI QUYẾT VẤN ĐỀ)

Với bài toán phân loại hình ảnh có nhiều nhãn => chọn mô hình CNN (convolutional neural network)

- **Mô tả kỹ thuật:**
  - Ngôn ngữ lập trình: Python
  - Packages, framework: tensorflow, numpy, matplotlib, opencv
  - Chạy trên google colab để có thể tận dụng được GPU giúp quá trình training nhanh chóng hơn

III. 5 . TIỀN XỬ LÝ DỮ LIỆU (PREPROCESSING)

Nhận thấy trong tập dữ liệu có những ảnh tương đồng nhau, có thể sử dụng các phương pháp để xóa bỏ những ảnh tương tự này.

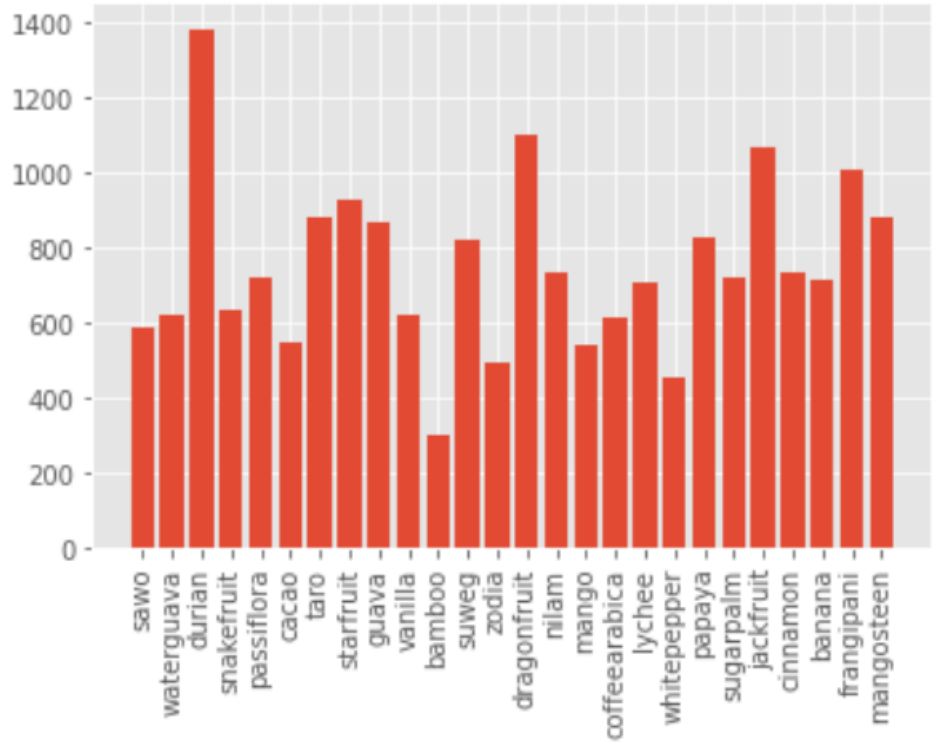
Có thể sử dụng thư viện imagehash để giải quyết vấn đề này. Sau quá trình xóa những ảnh tương tự , tập dữ liệu mới còn 34,510 ảnh, giảm gần 20% so với tập ban đầu

```
def remove_sameimage_whash(link):
  img_names = os.listdir(link)
  print(link, len(img_names))
  lst = []
  for i in img_names:
    hash = imagehash.whash(Image.open(f'{link}/{i}'))
    if hash in lst:
      os.remove(f'{link}/{i}')
    else:
      lst.append(hash)
  print('after',len(os.listdir(link)))
```

Hai ảnh có đặc điểm tương tự nhau

III. 5 . TIỀN XỬ LÝ DỮ LIỆU (PREPROCESSING)

Dữ liệu sau khi lọc không bị tình trạng imbalace



5 . TIỀN XỬ LÝ DỮ LIỆU (PREPROCESSING)

-Sử dụng chức năng ImageDataGenerator của tensorflow để preprocessing dữ liệu như:

- rescale: chuyển giá trị của pixel về khoảng từ 0 đến 1
- Resize (target size): chuyển hình ảnh về cùng kích thước
- Label biểu diễn dưới dạng one hot vector

Augmentation : tăng cường dữ liệu

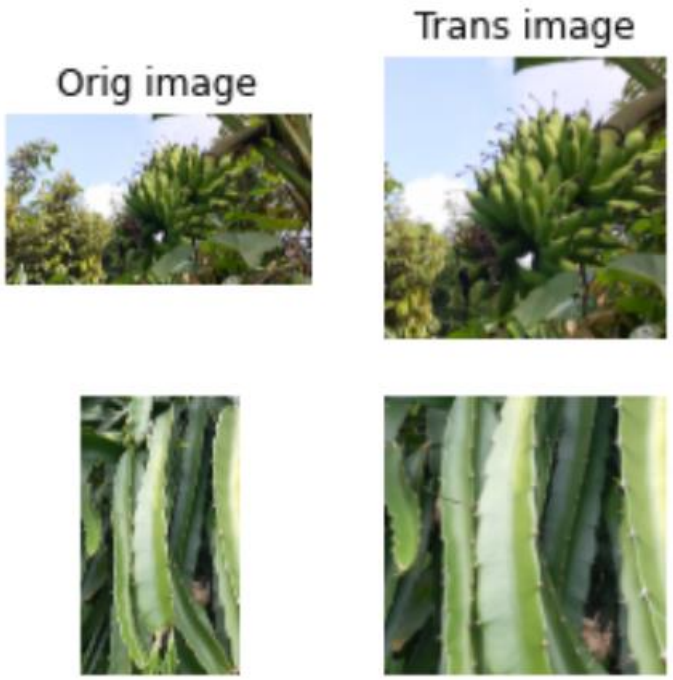
- width/height Shift: Dịch theo chiều ngang/dọc ngẫu nhiên trong một phạm vi nào đó
- Zoom : thực hiện zoom ngẫu nhiên trong một phạm vi nào đó
- Shear: làm méo ảnh
- brightness: thay đổi độ sáng của ảnh trong một phạm vi nào đó
- flip: lật ảnh ngẫu nhiên

```
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255, rotation_range=5,
    width_shift_range=0.05,
    height_shift_range=0.05,
    shear_range=0.05,
    zoom_range=0,
    brightness_range=(1, 1.3),
    horizontal_flip=True,
    fill_mode='nearest'
  ).flow_from_directory('/content/bali-26_train/train_val/train',
      target_size=(128, 128),batch_size = 32)
```

5 . TIỀN XỬ LÝ DỮ LIỆU (PREPROCESSING)

Một số Agumentation khác như

- Center crop: cắt ở giữa bức ảnh



- Mixup: trộn 2 bức ảnh vào nhau

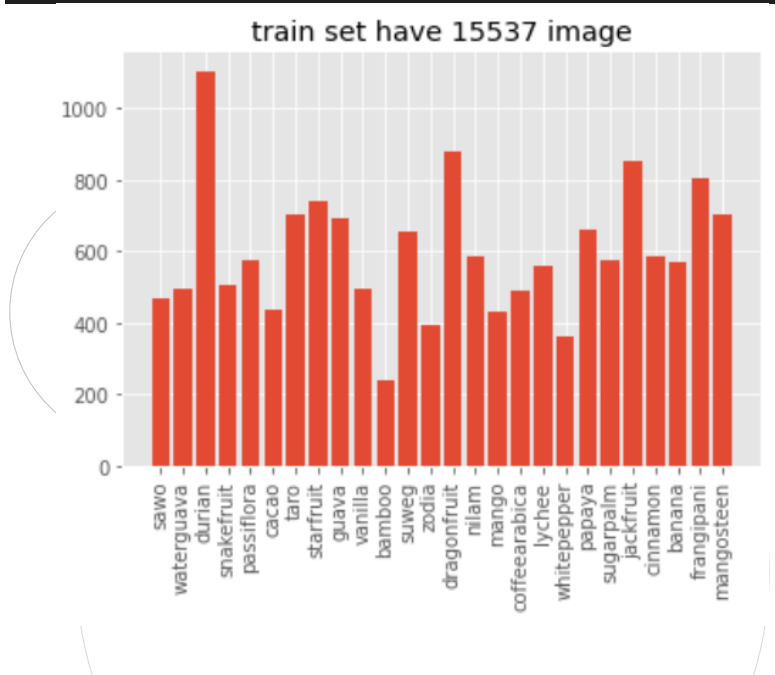
- cutmix: thay thế ô vuông bằng ảnh khác

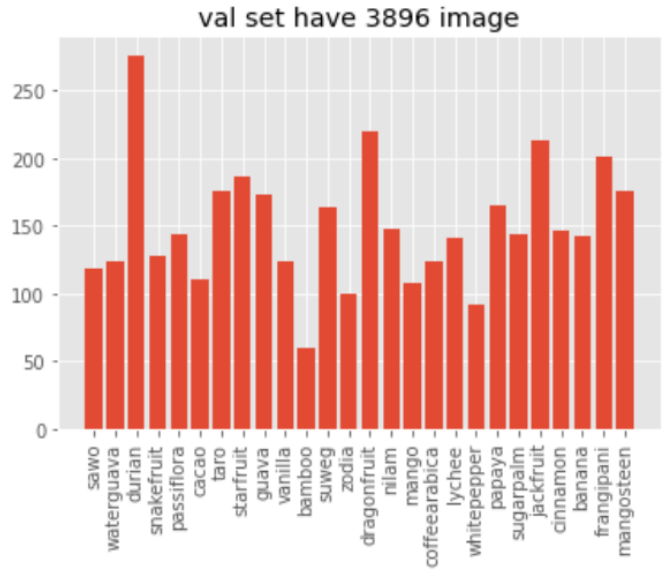
5 . TIỀN XỬ LÝ DỮ LIỆU (PREPROCESSING)

Chia dữ liệu thành tập train và validation

- Sử dụng thư viện split foldes để tách dữ liệu thành tập train và val theo tỉ lệ 8/2
- Số lượng của các nhãn được chia ra theo đúng tỉ lệ

```
# sử dụng split folder để tách thành tập train và val
!pip -q install split_folders
import splitfolders
input_folder = "/content/bali-26_train/bali-26_train"
output = "/content/bali-26_train/train_val/"

splitfolders.ratio(input_folder, output=output, seed=42, ratio=(.8, .2))
```





II. MODELLING & EVALUATION (LẬP MÔ HÌNH & ĐÁNH GIÁ)

1 Thử nghiệm:

Xây dựng mô hình CNN có cấu trúc như sau:

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(8,(3,3),padding = 'same',activation = 'relu', input_shape = train_data_gen.image_shape,name = 'Layer1'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(16,(3,3),padding = 'same',activation = 'relu',name = 'Layer2'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D((2,2),name = 'Layer3'),

    tf.keras.layers.Conv2D(32,(3,3),padding = 'same',activation = 'relu',name = 'Layer4'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(64,(3,3),padding = 'same',activation = 'relu',name ='Layer5'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D((3,3),name = 'Layer6'),

    tf.keras.layers.Flatten(name = 'Layer7'),
    tf.keras.layers.Dense(100,'relu',name = 'Layer8'),
    tf.keras.layers.BatchNormalization(),

    tf.keras.layers.Dense(26,'softmax',name = 'Layer9')
])
```

II. MODELLING & EVALUATION (LẬP MÔ HÌNH & ĐÁNH GIÁ)

1 Xác định các yếu tố:

- Xác định optimizer, loss function và metrics cho mocel

Xây dựng 1 số callbacks như:

- tensorboard (trực quan performance)
- modelcheckpoint (lưu kết quả)
- Learning rate schedule: set up learning rate
- Reduce lr on plateau: giảm learning rate khi kết quả không cải thiện
- Early stopping: dừng sớm

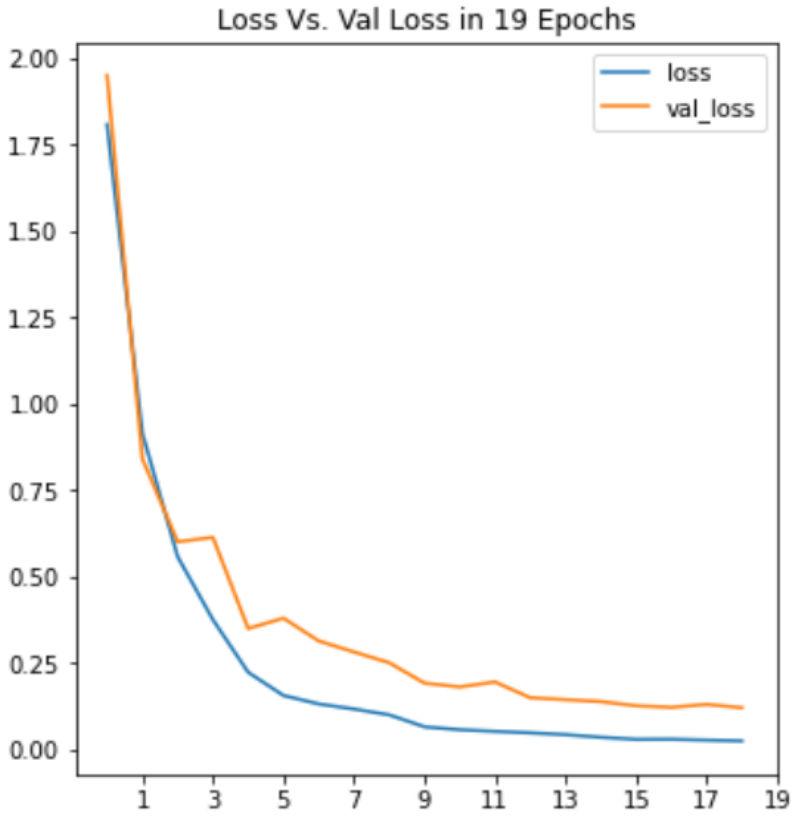
```
model.compile('adam','categorical_crossentropy',['accuracy'])
```

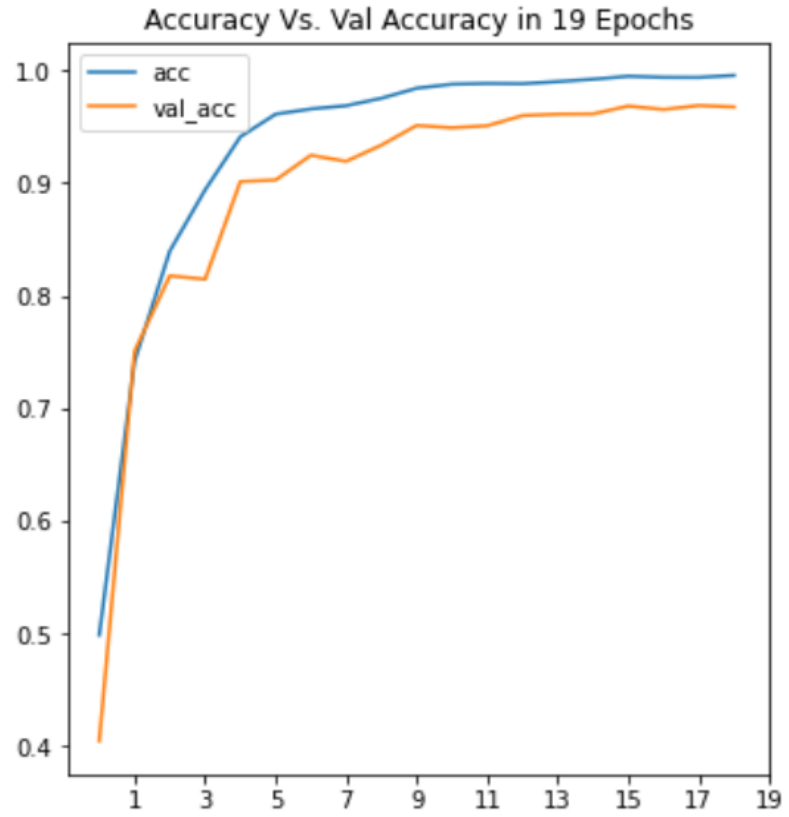
```
# tensorboard
logdir = os.path.join('logs2', datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tsb=tf.keras.callbacks.TensorBoard(log_dir=logdir,histogram_freq=1)
# model checkpoint
mdcp = tf.keras.callbacks.ModelCheckpoint('/content/drive/MyDrive/asia_plant_nocrop.h5',verbose = 1,
                                          monitor = 'val_accuracy',save_best_only=True)
# reducelronplateau
rdlr = tf.keras.callbacks.ReduceLROnPlateau(monitor ='val_accuracy', patience = 3)
# early stopping
es = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', min_delta=0.05,patience=3)
# learning rate schedule
def step_decay(epoch):
   initial_lrate = 1e-4/2
   drop = 0.5
   epochs_drop = 5
   lrate = initial_lrate * math.pow(drop,
           math.floor((1+epoch)/epochs_drop))
   return lrate
lr_schedule = tf.keras.callbacks.LearningRateScheduler(step_decay)
```

II. MODELLING & EVALUATION (LẬP MÔ HÌNH & ĐÁNH GIÁ)

2 Kết quả Đánh giá: .

- Sau khi huấn luyện mô hình, có kết quả tương đối tốt,
- model đạt accuracy trên tập train là 0.9936 và tập test là 0.9687





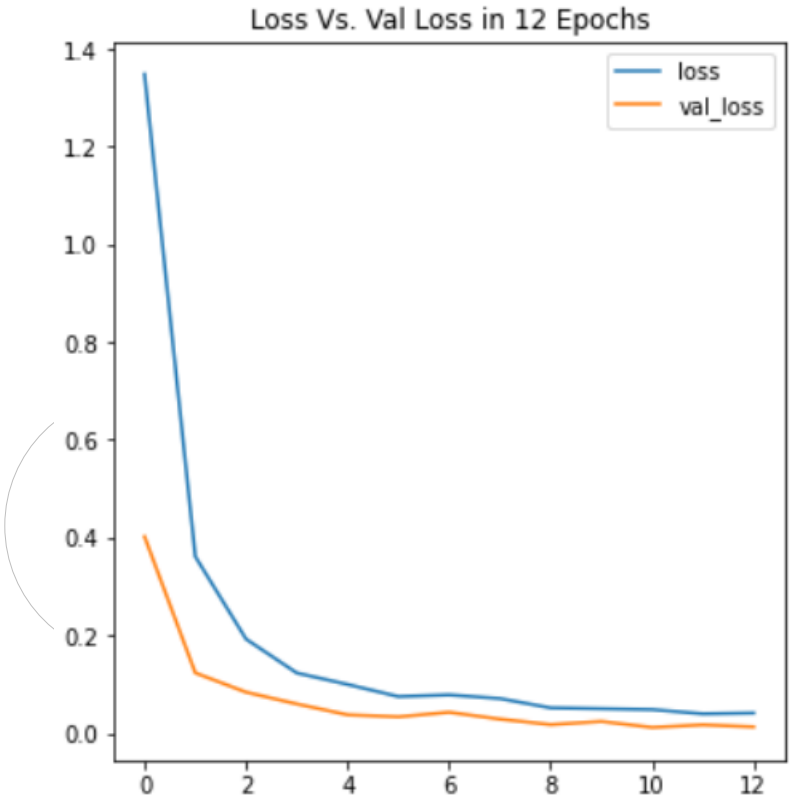
- Đưa mô hình dự đoán tập test trên kaggle cũng được kết quả khá tốt là 0,96809

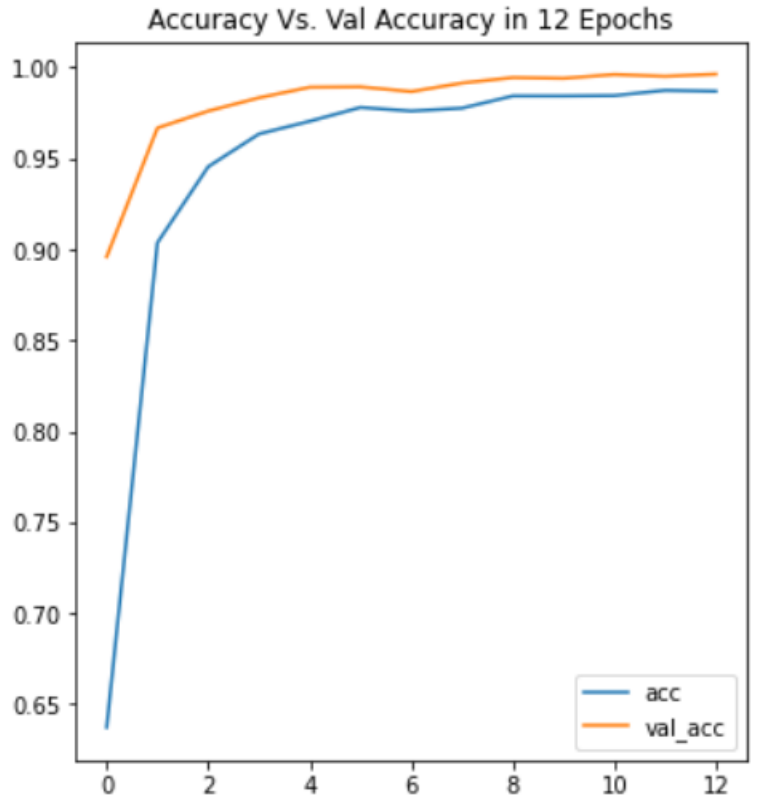
III. CẢI THIỆN MÔ HÌNH

3.1 : Thêm 1 convolutional layer như hình:

```
tf.keras.layers.Conv2D(128,(3,3),padding = 'same',activation = 'relu',name = 'Layer7'),
tf.keras.layers.BatchNormalization(),
```

Kết quả:
Sau 12 epochs, model đã tốt lên: accuracy trên tập train là 0.9897 trong khi accuracy trên tập test là 0.9866



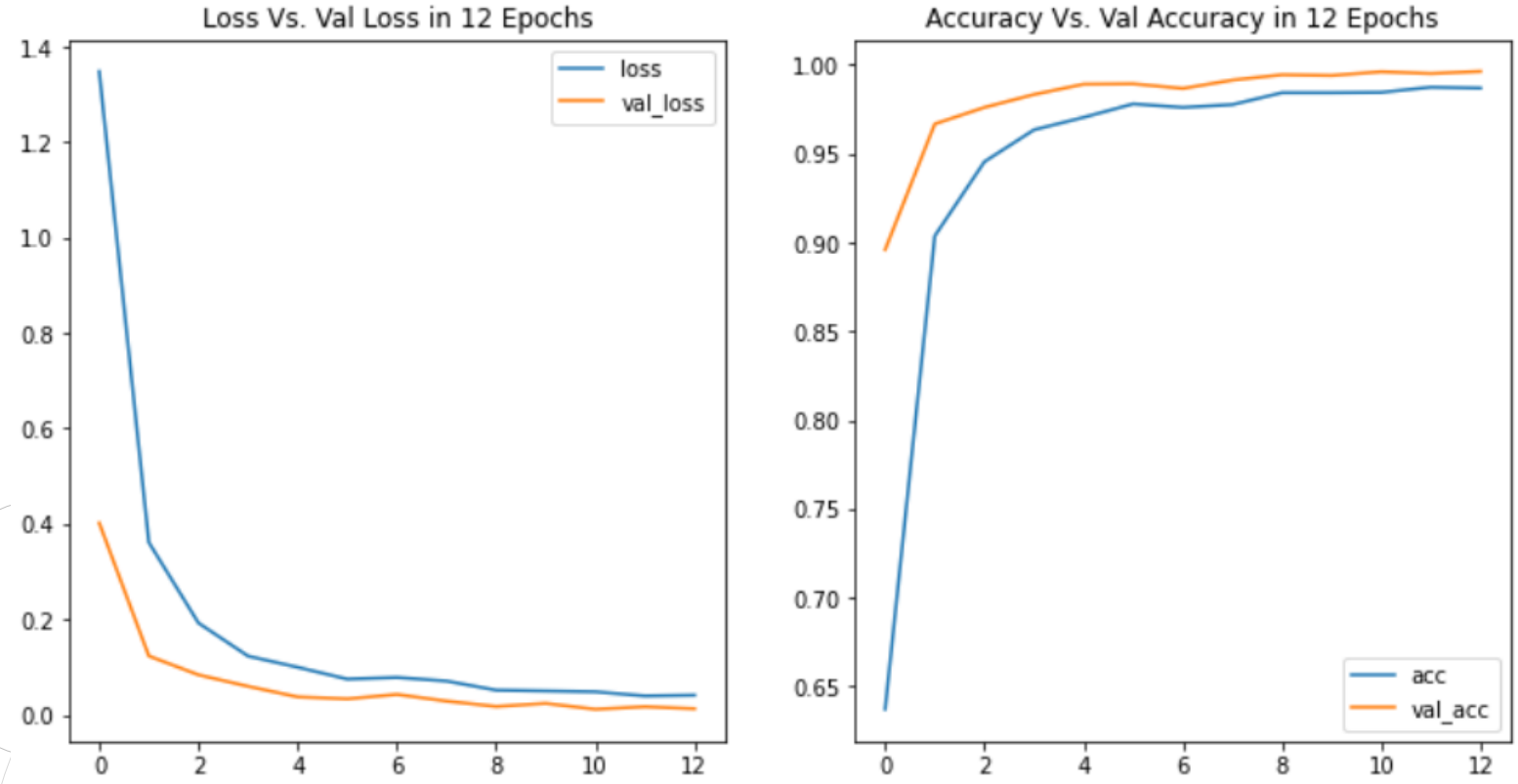


III. CẢI THIỆN MÔ HÌNH

```
tf.keras.layers.Conv2D(256,(3,3),padding = 'same',activation = 'relu',name = 'Layer8'),
tf.keras.layers.BatchNormalization(momentum=0.95, epsilon=0.005, beta_initializer=tf.ker
tf.keras.layers.MaxPooling2D((5,5),name = 'Layer9'),

tf.keras.layers.Dropout(rate = 0.5,name = 'Layer12'),
```

3.2 : Tiếp tục thêm 1 convolutional layer và maxpooling như hình:

- Thêm dropout layer vào trước lớp output như hình



Kết quả:
Sau 12 epochs, model đã tốt lên: accuracy của tập validation đạt được 0.9964 sau 12 epochs

- Đưa mô hình dự đoán tập test trên kaggle cũng được kết quả cải thiện là 0,99740, tăng 2.9% so với model ban đầu

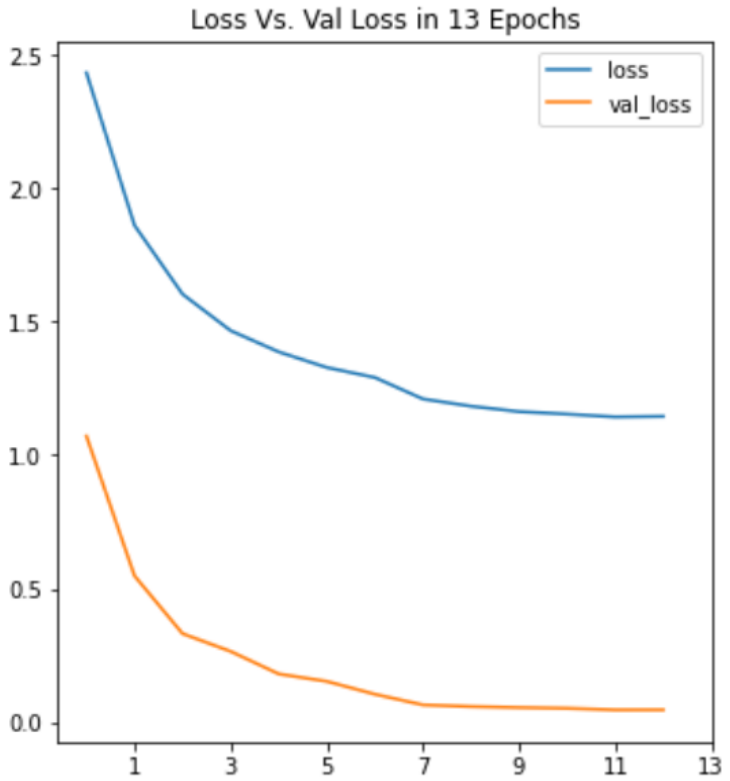
III. CẢI THIỆN MÔ HÌNH

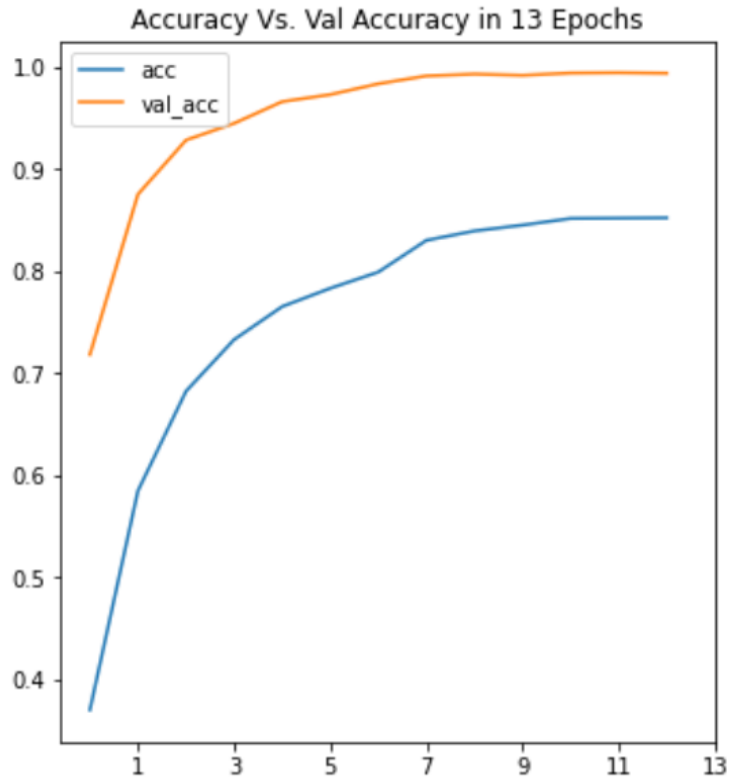
3.3 : Sử dụng cutmix

CutMix: Phương pháp này vừa là kết hợp giữa Cutout và Mixup. Theo đó chúng ta thay thế những ô vuông trên một ảnh bằng những ô vuông có cùng diện tích của một ảnh khác thuộc các nhãn còn lại. Nhãn mới được tạo thành cũng là nhãn mềm được kết hợp tuyến tính từ hai nhãn.

Sau 13 epochs, model sử dụng cutmix đạt accuracy trên tập val là 0.9942, kết quả khá tốt

Áp dụng trên tập test ta được kết quả là 0,99480





III. CẢI THIỆN MÔ HÌNH

3.4 : Pretrain với mạng EfficientnetB0

Mạng Efficiecntnet đạt được độ chính xác tốt hơn nhiều so với các kiến trúc CNN trước đó.

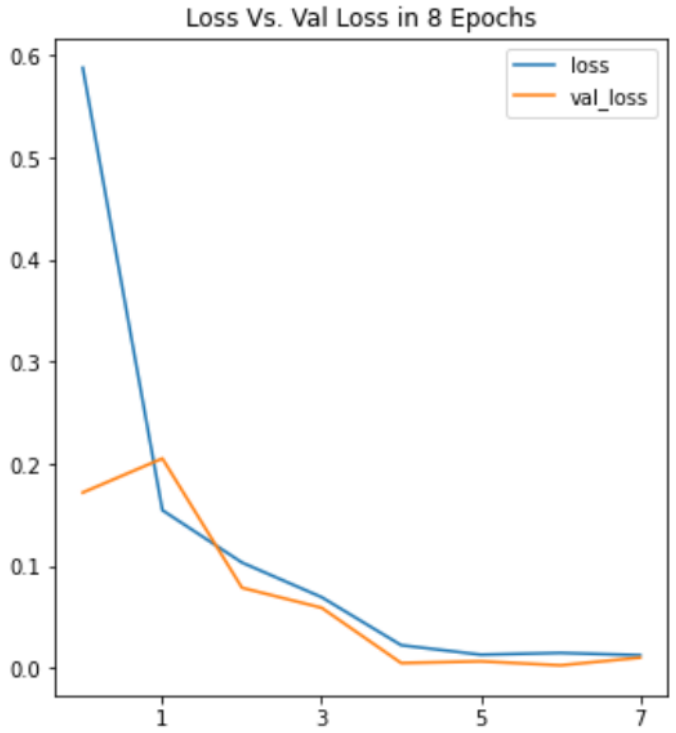
Ý tưởng của EfficientNet là thay vì tìm ra một kiến trúc tối ưu từ đầu thì nó xuất phát từ một Base model F, sau đó dần dần mở rộng, cải tiến nó dần lên. EfficientNet tiến hành Scale-up đồng thời cả 3 thành phần, gọi là Compound Scaling.

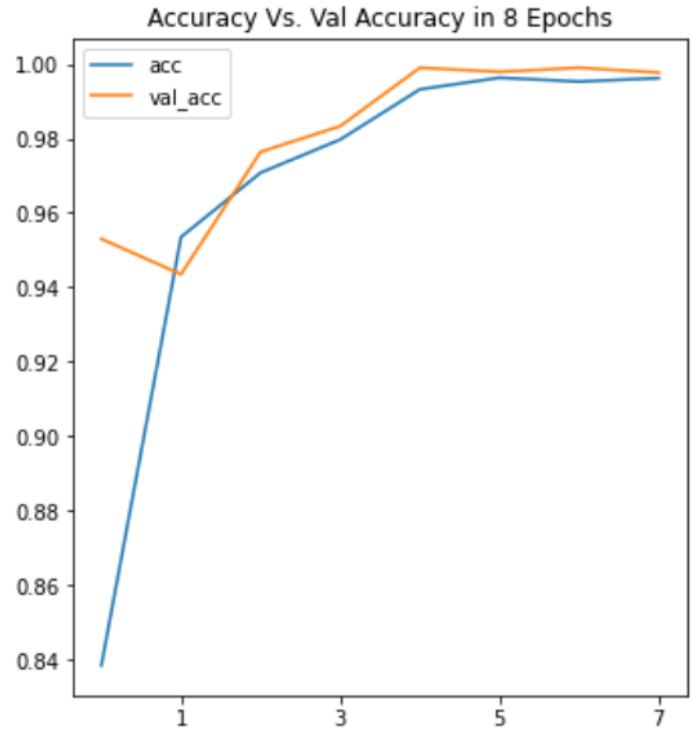
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb0 (Functional)  (None, 4, 4, 1280)       4049571

 global_average_pooling2d_2   (None, 1280)             0
 (GlobalAveragePooling2D)

 dropout_4 (Dropout)         (None, 1280)              0

 dense_5 (Dense)             (None, 26)                33306

=================================================================
Total params: 4,082,877
Trainable params: 4,040,854
Non-trainable params: 42,023
_________________________________________________________________
```

III. CẢI THIỆN MÔ HÌNH

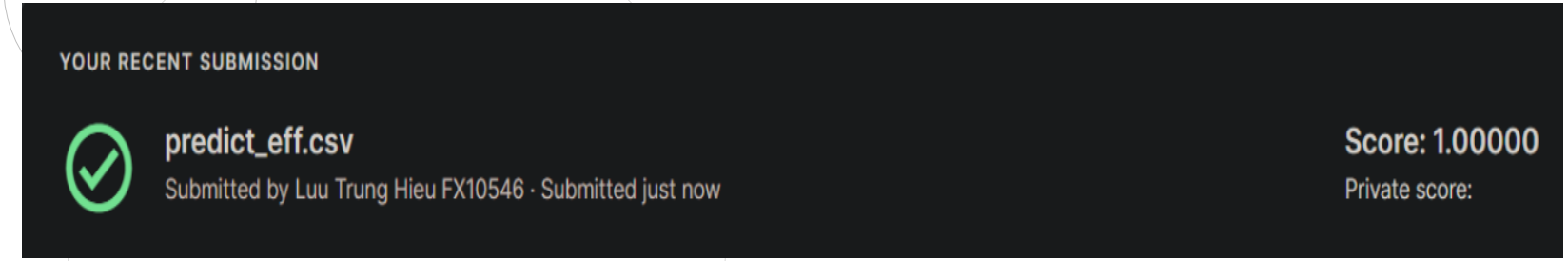
3.4 : Pretrain với mạng EfficientnetB0

Mạng hoạt động rất tốt trên bộ dữ liệu, đạt accuracy trên tập valid là 0,99 khi đến epochs thứ 4





Áp dụng model trên với tập test trên kaggle đạt được độ chính xác là 1,000



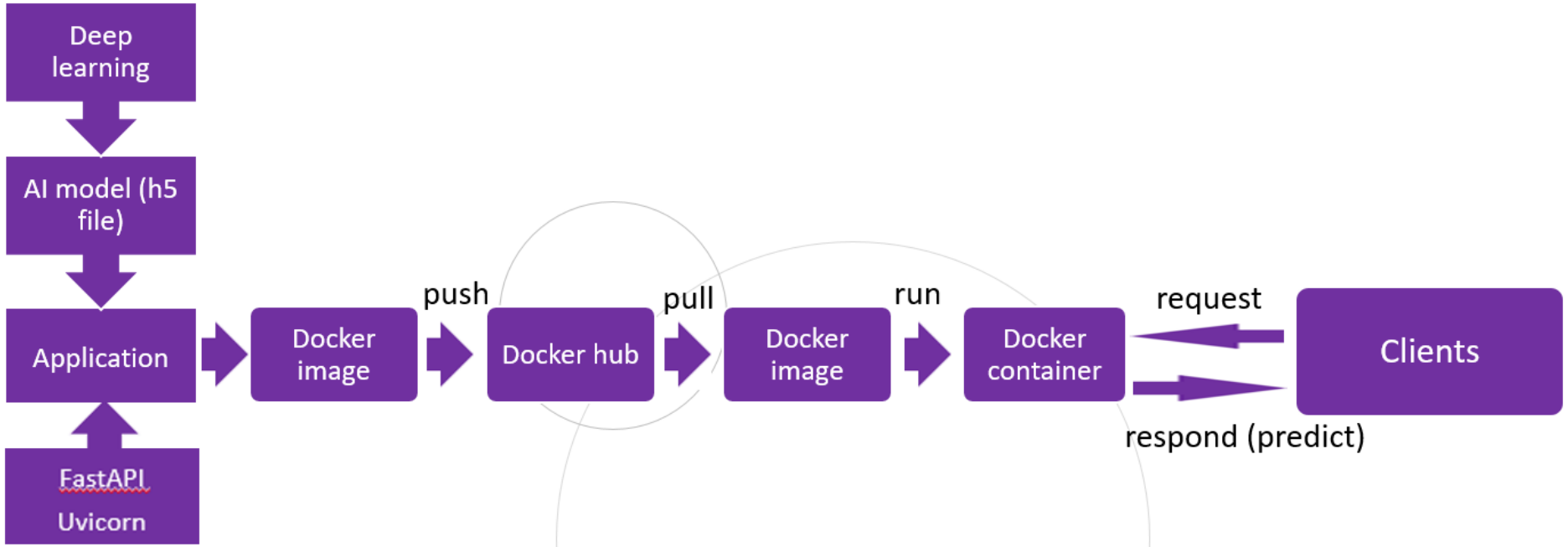
BẢNG TỔNG HỢP KẾT QUẢ

	Thêm layer	agumentation	pretrain	Kết quả (accuracy val/test)
Model baseline				0,9687 (19 epochs)
model1	Thêm 1 conv2d layer			0.9866 (12 epochs)
Model2	Thêm 2 conv2d, 1 maxpooling, 1 dropout	Crop center		0,99221 (10 epochs)
Model2	Thêm 2 conv2d, 1 maxpooling, 1 dropout			0,99740(12 epochs)
Resnet50_base		Resnet preprocess input	resnet50	0,9938 (5 epochs)
cutmix_model	Thêm 2 conv2d, 1 maxpooling, 1 dropout	cutmix		0.9942 (12 epochs)
Efficientnetb0_base		Efficientnet preprocess input	Efficientnetb0	1,0000 (5 epochs)
Mixup_model	Thêm 2 conv2d, 1 maxpooling, 1 dropout	mixup		0.9942 (12 epochs)

IV. Model Serving (triển khai model)

Sau khi xây dựng xong model thì phải triển khai model, tức là đưa model lên môi trường production, từ đó có thể kết hợp với những services khác để làm việc với người dùng cuối. Trong dự án này, em tạo 1 ứng dụng bằng tiêu chuẩn RESTful API và đóng gói ứng dụng này

Cụ thể, em sử dụng dùng FastAPI, để xử lý best model thành 1 docker image. Sau đó chạy docker image này dưới dạng container chạy trên localhost để nhận đầu vào là hình ảnh như lưu đồ sau

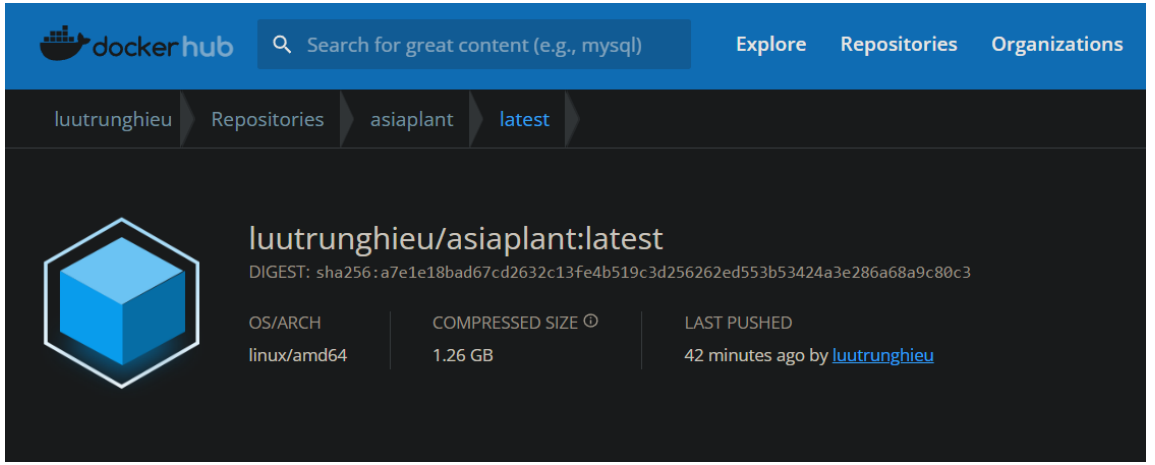


IV. Model Serving (triển khai model)

build docker image : [link github]

Vào terminal/window powershell gõ lệnh sau:

docker build -t hieu/asiaplant:no-batch03 .



Push docker images lên docker hub sử dụng lệnh sau:

docker tag hieu/asiaplant:no-batch03 luutrunghieu/asiaplant:latest

docker push luutrunghieu/asiaplant:latest

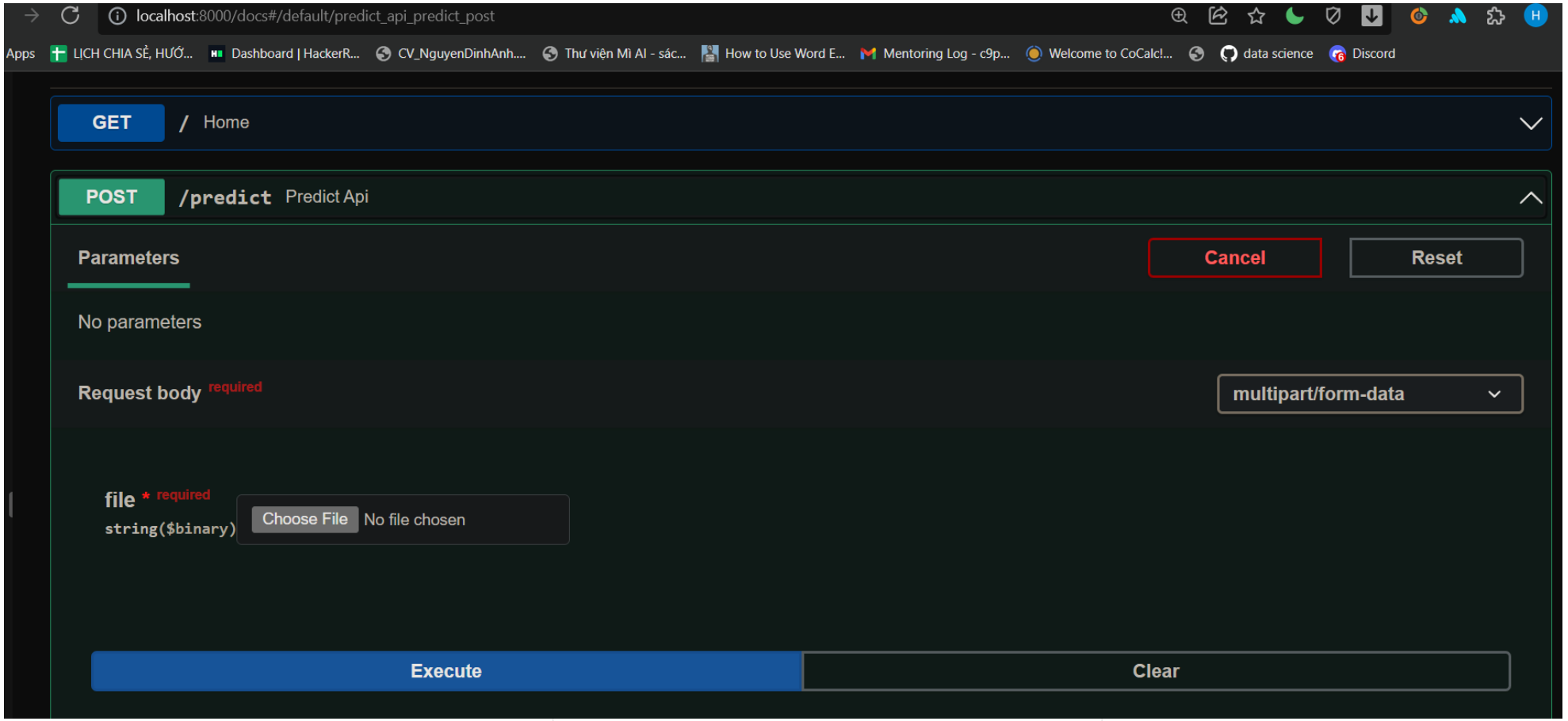
Kết quả như hình bên cạnh:

Để pull docker image ta dùng lệnh: *docker pull luutrunghieu/asiaplant:latest*

Chạy docker image (tạo docker container) *: docker run -- rm –p 8000:8000 luutrunghieu/asiaplant:latest*

IV. Model Serving (triển khai model)

Truy Cập vào địa chỉ http://localhost:8000/docs để thu được kết quả



IV. Model Serving (triển khai model)

Nhập ảnh sau và thu được kết quả

Curl

```
curl -X 'POST' \
  'http://localhost:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@100.jpg;type=image/jpeg'
```

Request URL

```
http://localhost:8000/predict
```

Server response

Code | Details

200

Response body


```
{
  "filename": "100.jpg",
  "prediction": [
    {
      "guava": "99.96 %"
    },
    {
      "mangosteen": "0.04 %"
    }
  ],
  "encoded_img": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAIBAQEBAQIBAQECAgICAgQDAgICAgUEBAMEBgUGBgYFBgYGBwkIBgcJBwYGCAsICQoKCgoKBggLDAsKDAkKCgr/2wBDAQICAgICAgUDAwUKBwYHCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgr/wAARCAEAAQADASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREAAgECBAQDBAcFBAAAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwD5b+9ooooA/9k="
}
```

V. CONCLUSION (KẾT LUẬN)

SO VỚI MÔ HÌNH BASELINE, CÁC MÔ HÌNH CẢI THIỆN CÓ ĐỘ CHÍNH XÁC TĂNG KHOẢNG 3%

SAU NÀY, VỚI LƯỢNG DỮ LIỆU NHIỀU VÀ ĐA DẠNG HƠN, MÔ HÌNH CÓ THỂ DỰ ĐOÁN ĐƯỢC NHIỀU LOẠI CÂY HƠN VÀ DỰ ĐOÁN CHÍNH XÁC HƠN

THANK YOU

hieultfx10546@funix.edu.vn