

AI VIET NAM – COURSE 2022

Basic CNN - Exercise

Ngày 11 tháng 12 năm 2022

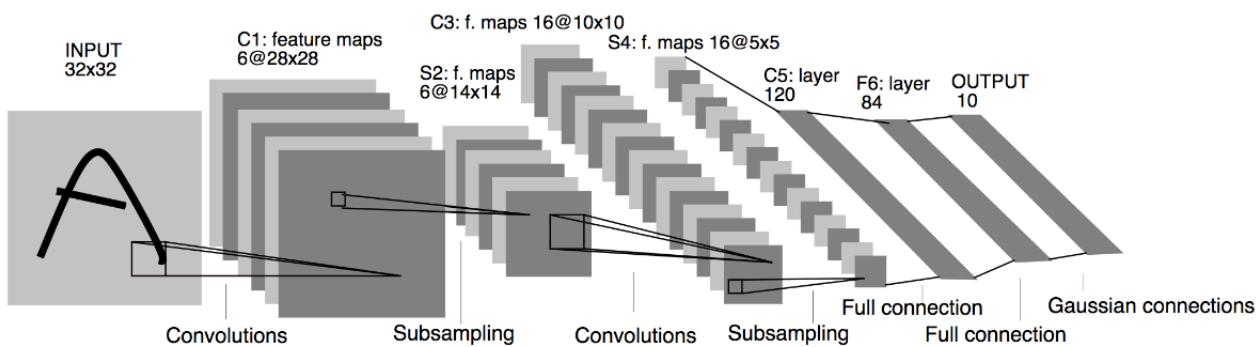
Phần I: Cài đặt thuật toán

Problem 1 - Build Convolution Neural Network with Lenet Model

- Mô tả bài toán

LeNet là một trong những mạng CNN lâu đời nổi tiếng nhất được Yann LeCun phát triển vào những năm 1998s. Một số điểm lưu ý về kiến trúc LeNet:

- Kiến trúc gồm 7 layers, trong đó có 3 convolutional layers, 2 subsampling layers, và 2 fully connected layers.
- Khi xây dựng kiến trúc, dùng 2 loại layers mang ý nghĩa chính là Convolution layers và Subsampling layers.



Hình 1: LeNet model, 1998

Kiến trúc mô hình chi tiết:

| Layer | Layer type | Feature Map | Size | Kernel | Stride | Activation | Padding |
|--------|--------------------------|-------------|-------|--------|--------|------------|---------|
| Input | Image | 1 | 32x32 | - | - | - | - |
| C1 | Convolution (Conv2D) | 6 | 28x28 | 5x5 | 1 | relu | no |
| S2 | Sub sampling (AvgPool2D) | 6 | 14x14 | 2x2 | 2 | - | same |
| C3 | Convolution (Conv2D) | 16 | 10x10 | 5x5 | 1 | relu | no |
| S4 | Sub sampling (AvgPool2D) | 16 | 5x5 | 2x2 | 2 | - | same |
| C5 | Convolution | 120 | 1x1 | 5x5 | 1 | relu | |
| F6 | Fully connected | - | 84 | - | - | relu | |
| Output | Fully connected | - | 10 | - | - | softmax | |

Hình 2: LeNet Architecture

- Triển khai thuật toán

```

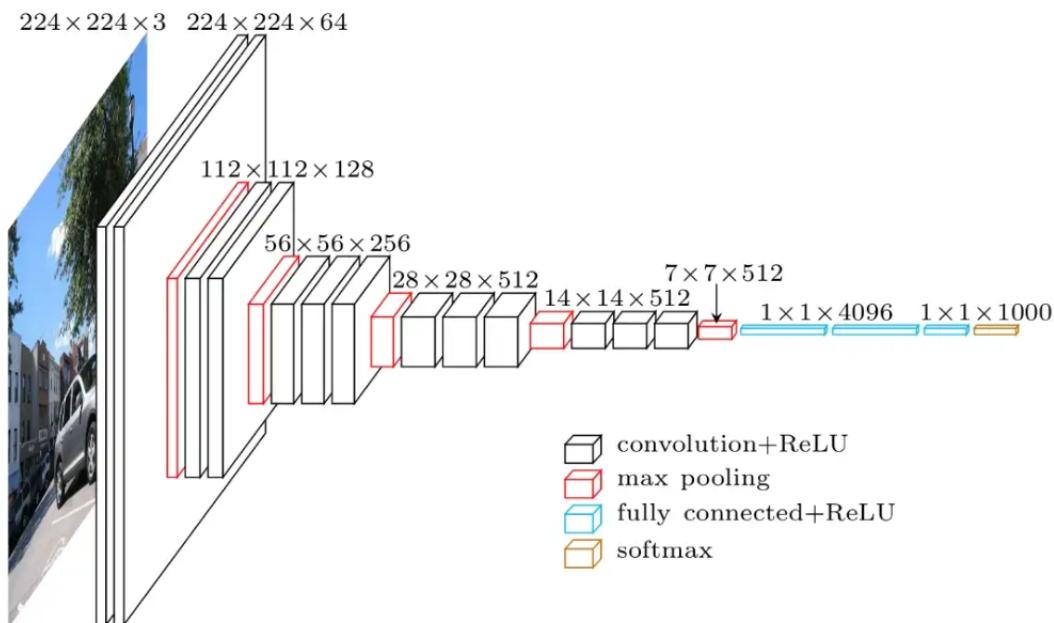
1 lenet_model = tf.keras.models.Sequential([
2     tf.keras.layers.Conv2D(), # C1
3     tf.keras.layers.AvgPool2D(), # S2
4     tf.keras.layers.Conv2D(), # C3
5     tf.keras.layers.AvgPool2D(), # S4
6     tf.keras.layers.Conv2D(), # C5
7     tf.keras.layers.Flatten(),
8     tf.keras.layers.Dense(), # F6
9     tf.keras.layers.Dense() # Output layer
10]

```

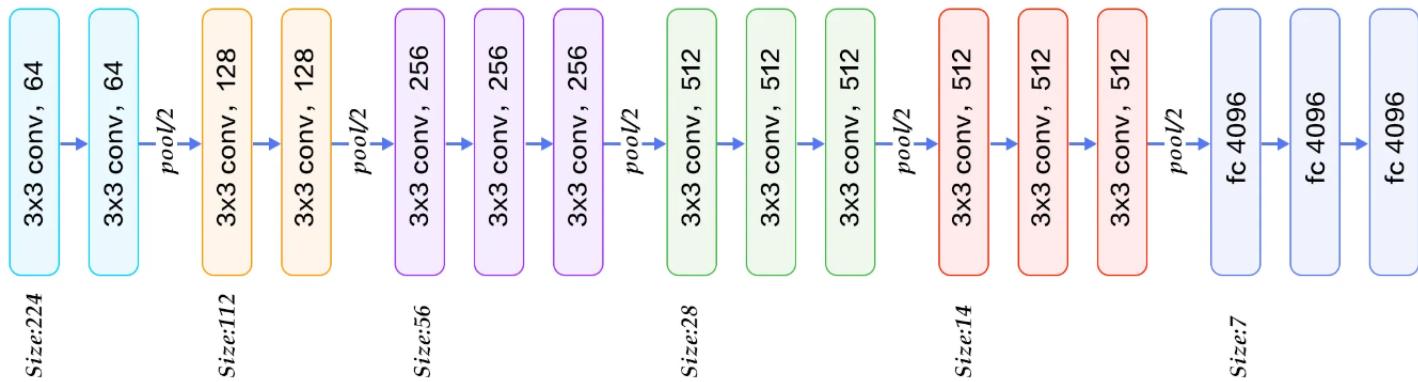
Problem 2 - Build Networks Using Blocks (VGG16)

- Kiến trúc mô hình VGG-16 gồm 2 thành phần chính:

- Thành phần thứ nhất là chuỗi các khối VGG, trong mỗi khối VGG bao gồm một chuỗi các convolutional layers, sau đó là maximum pooling layer.
- Thành phần thứ hai là một khối lớp dense gồm 3 fully connected layers.



Hình 3: VGG-16 Architecture, 2014



Hình 4: VGG-16 Architecture, 2014

- Kiến trúc mô hình chi tiết:

| Layer | Layer type | Feature map | Size | Kernel | Pooling size | Stride | Activation |
|--------|-----------------|-------------|-------------|--------|--------------|--------|------------|
| Input | Image | 1 | 224x224x3 | - | - | - | - |
| 1 | 2 x Convolution | 64 | 224x224x64 | 3x3 | - | 1 | relu |
| 2 | Max pooling | 64 | 112x112x64 | - | 2 | 2 | - |
| 3 | 2 x Convolution | 128 | 112x112x128 | 3x3 | - | 1 | relu |
| 4 | Max pooling | 128 | 56x56x128 | - | 2 | 2 | - |
| 5 | 2 x Convolution | 256 | 56x56x256 | 3x3 | - | 1 | relu |
| 6 | Max pooling | 256 | 28x28x256 | - | 2 | 2 | - |
| 7 | 3 x Convolution | 512 | 28x28x512 | 3x3 | - | 1 | relu |
| 8 | Max pooling | 512 | 14x14x512 | - | 2 | 2 | - |
| 9 | 3 x Convolution | 512 | 14x14x512 | 3x3 | - | 1 | relu |
| 10 | Max pooling | 512 | 7x7x512 | - | 2 | 2 | - |
| 11 | Fully connected | - | 25088 | - | - | - | relu |
| 12 | Fully connected | - | 4096 | - | - | - | relu |
| 13 | Fully connected | - | 4096 | - | - | - | relu |
| Output | Fully connected | - | 1000 | - | - | - | softmax |

Hình 5: VGG-16, Architecture detail

- Triển khai thuật toán

```
1 vgg16_model = tf.keras.models.Sequential([
2     # 1st Conv Block
3     tf.keras.layers.Conv2D(),
4     tf.keras.layers.Conv2D(),
5     tf.keras.layers.MaxPool2D(),
6
7     # 2nd Conv Block
8     tf.keras.layers.Conv2D(),
9     tf.keras.layers.Conv2D(),
10    tf.keras.layers.MaxPool2D(),
11
12    # 3rd Conv block
13    tf.keras.layers.Conv2D(),
14    tf.keras.layers.Conv2D(),
15    tf.keras.layers.Conv2D(),
16    tf.keras.layers.MaxPool2D(),
17
18    # 4th Conv block
19    tf.keras.layers.Conv2D(),
20    tf.keras.layers.Conv2D(),
21    tf.keras.layers.Conv2D(),
22    tf.keras.layers.MaxPool2D(),
23
24    # 5th Conv block
25    tf.keras.layers.Conv2D(),
26    tf.keras.layers.Conv2D(),
27    tf.keras.layers.Conv2D(),
28    tf.keras.layers.MaxPool2D(),
29
30    # Fully connected layers
31    tf.keras.layers.Flatten(),
32    tf.keras.layers.Dense(),
33    tf.keras.layers.Dense(),
34    tf.keras.layers.Dense()
35 ])
```

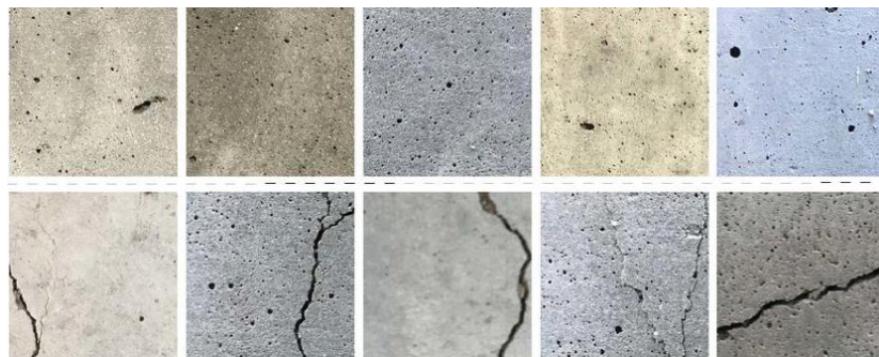
Problem 3 - Ứng dụng LeNet và VGG-16 vào bài toán phân loại ảnh dựa trên 2 dataset, Cassava Leaf Disease và Concrete Crack

- Cassava Leaf Disease Dataset gồm 5 classes
 - CBB: Cassava bacterial blight
 - CBSD: Cassava brown streak disease
 - CGM: Cassava green mottle
 - CMD: Cassava mosaic disease
 - Healthy



Hình 6: Cassava Leaf Disease

- **Concrete Crack:** gồm 2 class là crack và no crack



Hình 7: Concrete crack

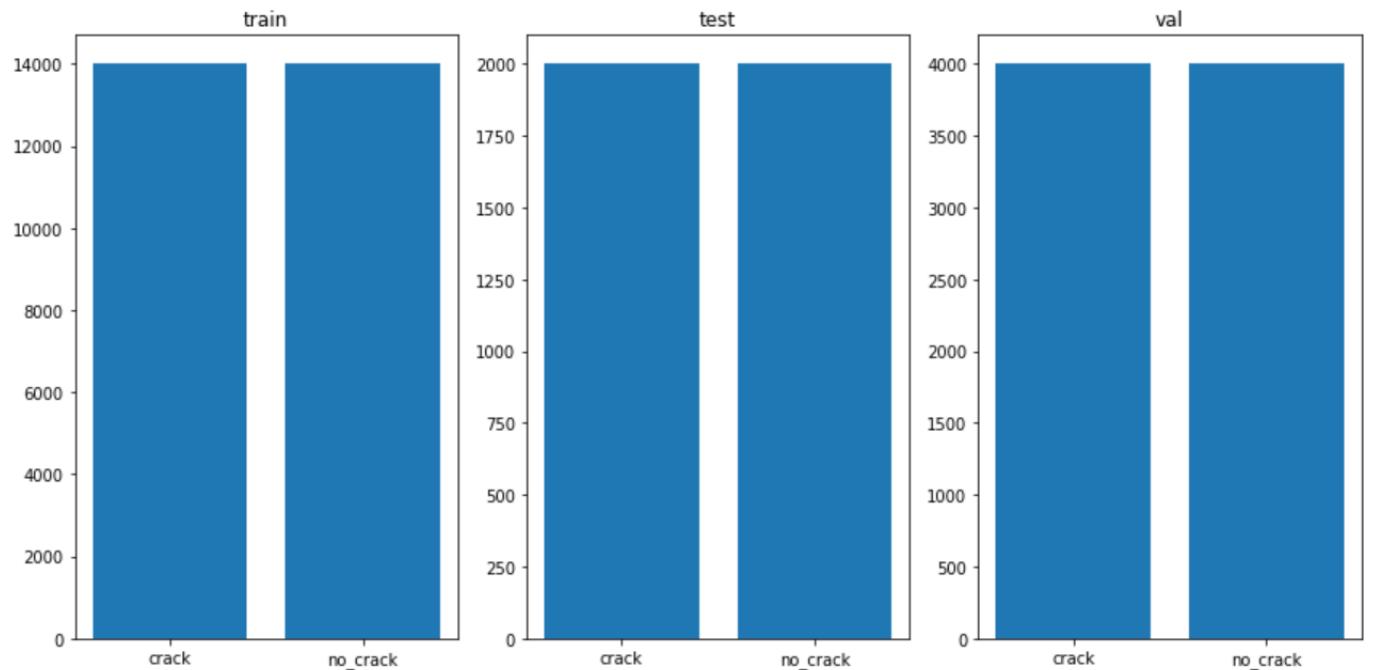
Các bước thực hiện như sau:

- Chuẩn bị dữ liệu: chia dataset thành train, validation và test
- Chuẩn hóa các điểm ảnh về khoảng giá trị 0 và 1
- Tăng cường dữ liệu với các phép xoay, phóng, ...
- Huấn luyện dữ liệu trên 2 model LeNet và VGG-16
- Dánh giá mô hình trên tập test

- Triển khai thuật toán với Concrete Crack dataset

1. Load data

```
1 import os
2 import cv2
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import matplotlib.image as mpimg
6 from imutils import paths
7 from sklearn.metrics import classification_report, confusion_matrix
8
9 import tensorflow as tf
10 from keras.preprocessing.image import ImageDataGenerator
11 from keras.models import load_model
12
13
14 !pip install GitPython
15 from git import Repo
16
17 filepath = 'temp_concrete_crack'
18 Repo.clone_from('https://github.com/bimewok/Concrete-Crack-Image-Classifier',
19                 filepath)
20
21 base_dir = '/content/temp_concrete_crack/data/concrete_images'
22 train_dir = os.path.join(base_dir, 'train')
23 valid_dir = os.path.join(base_dir, 'val')
24 test_dir = os.path.join(base_dir, 'test')
25
26 data = os.listdir(base_dir)
27 crack = os.path.join(data[0], 'crack')
28 img_path = base_dir + '/' + crack + '/' + os.listdir(os.path.join(base_dir,
29             crack))[0]
30 img = cv2.imread(img_path)
31 print(img.dtype, img.shape)
32
33
34 def show_labels(data_dir):
35     data = os.listdir(data_dir)
36     fig, ax = plt.subplots(1, len(data), figsize=(12,6))
37     for idx in range(len(data)):
38         sub_dir = os.path.join(data_dir, data[idx])
39         labels = os.listdir(sub_dir)
40         list_data = []
41         for label in labels:
42             image_files = list(paths.list_images(os.path.join(sub_dir, label)))
43         list_data.append(len(image_files))
44         ax[idx].bar(labels, list_data)
45         ax[idx].set_title(data[idx])
46         # ax[idx].axis('off')
47     plt.tight_layout()
48     plt.show()
49
50 show_labels(base_dir)
51
```



Hình 8: Train, Validation, Test of Concrete Crack

2. Preprocessing

```

1 img_size = img.shape[0] # = 227
2 output_size = 1 #Use sigmoid function
3 batch_size = 256
4
5 train_datagen = ImageDataGenerator( rescale=1.0/255.0,
6                                     rotation_range=30,
7                                     zoom_range=0.15,
8                                     width_shift_range=0.2,
9                                     height_shift_range=0.2,
10                                    shear_range=0.15,
11                                    horizontal_flip=True,
12                                    fill_mode="nearest" )
13
14 val_datagen = ImageDataGenerator(rescale=1.0/255.0)
15
16 test_datagen = ImageDataGenerator(rescale=1.0/255.0)
17
18 # prepare iterators
19 train_dataloader = train_datagen.flow_from_directory(train_dir,
20                                                       class_mode='binary',
21                                                       batch_size=batch_size,
22                                                       target_size=(img_size,
23                                                       img_size))
24 valid_dataloader = val_datagen.flow_from_directory(valid_dir,
25                                                       class_mode='binary',
26                                                       batch_size=batch_size,
27                                                       target_size=(img_size, img_size))
28
29 test_dataloader = test_datagen.flow_from_directory(test_dir,
30                                                       class_mode='binary',

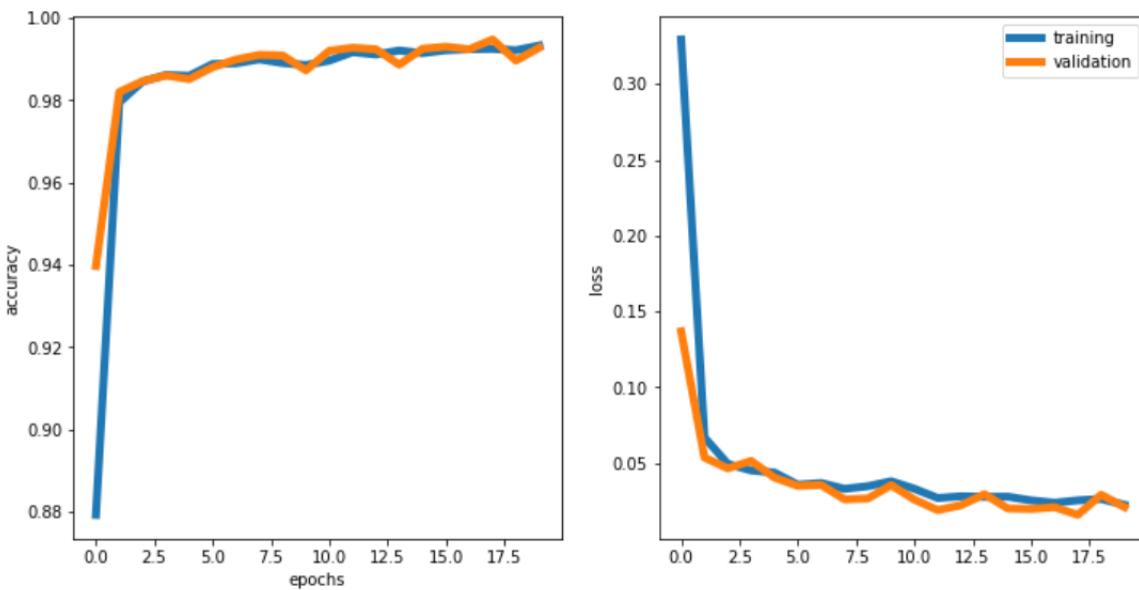
```

```

31
32
33
batch_size=batch_size,
target_size=(img_size, img_size))

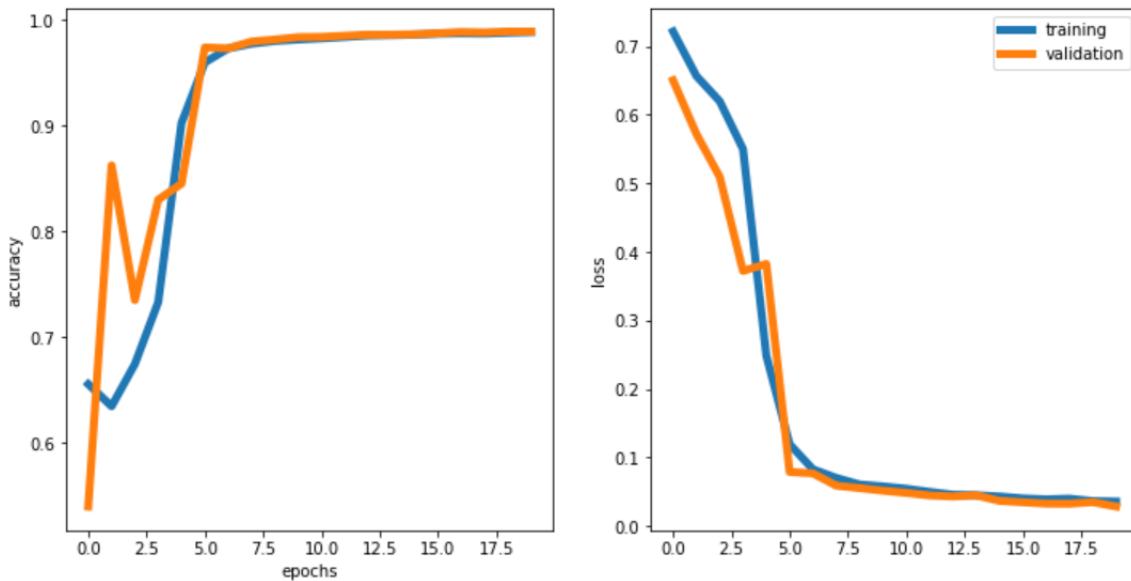
```

3. Áp dụng model LeNet.



Hình 9: LeNet results

4. Áp dụng model VGG-16.



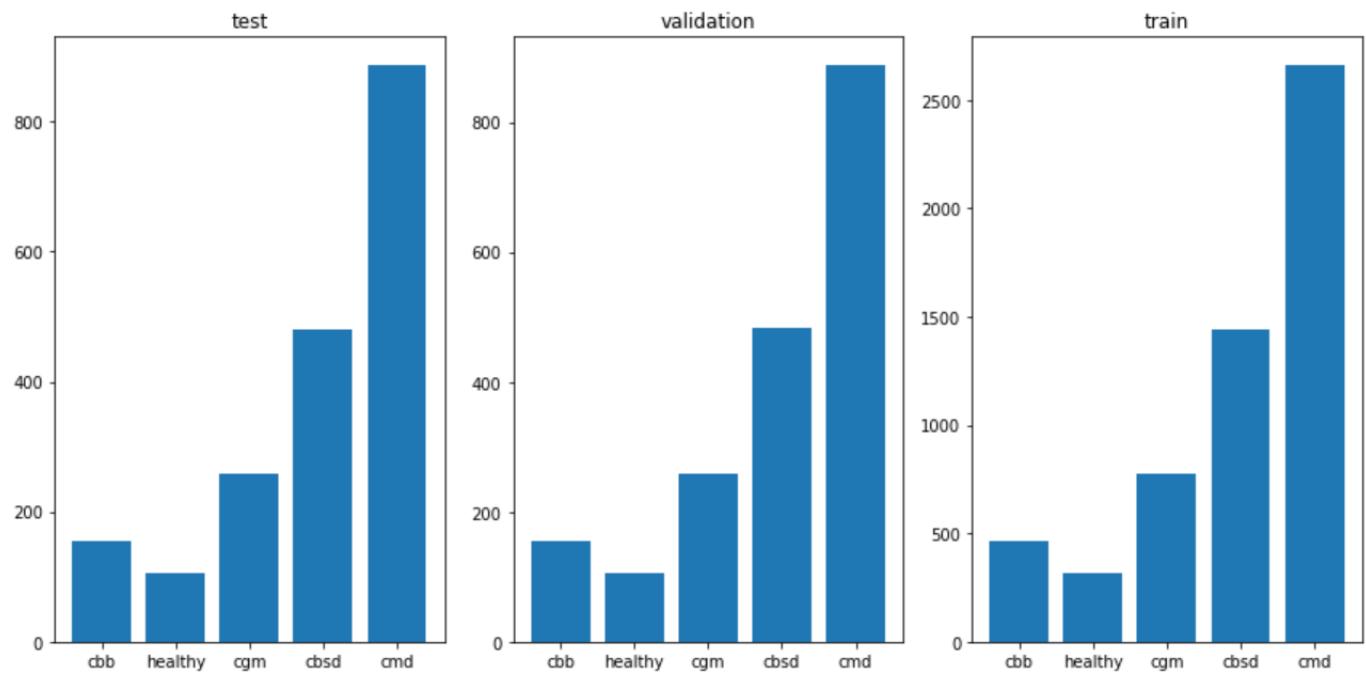
Hình 10: VGG-16 results

- Triển khai thuật toán với Cassava Leaf Disease dataset

1. Load data

```
1 !wget --no-check-certificate https://storage.googleapis.com/emcassavadata/
      cassavaleafdata.zip \
2          -O /content/cassavaleafdata.zip
3 !unzip '/content/cassavaleafdata.zip'
4
5 base_dir = '/content/cassavaleafdata'
6 train_dir = os.path.join(base_dir, 'train')
7 valid_dir = os.path.join(base_dir, 'validation')
8 test_dir = os.path.join(base_dir, 'test')
9
10 labels_dict = {
11     "cbb": "Cassava Bacterial Blight (CBB)",
12     "cbsd": "Cassava Brown Streak Disease (CBSD)",
13     "cgm": "Cassava Green Mottle (CGM)",
14     "cmd": "Cassava Mosaic Disease (CMD)",
15     "healthy": "Healthy"
16 }
17
18

1 def show_labels(data_dir):
2     data = os.listdir(data_dir)
3     fig, ax = plt.subplots(1, len(data), figsize=(12,6))
4     for idx in range(len(data)):
5         sub_dir = os.path.join(data_dir, data[idx])
6         labels = os.listdir(sub_dir)
7         list_data = []
8         for label in labels:
9             image_files = list(paths.list_images(os.path.join(sub_dir, label)))
10        list_data.append(len(image_files))
11    ax[idx].bar(labels, list_data)
12    ax[idx].set_title(data[idx])
13    # ax[idx].axis('off')
14    plt.tight_layout()
15    plt.show()
16 show_labels(base_dir)
17
```



Hình 11: Train, Validation, Test of Cassava Leaf Disease

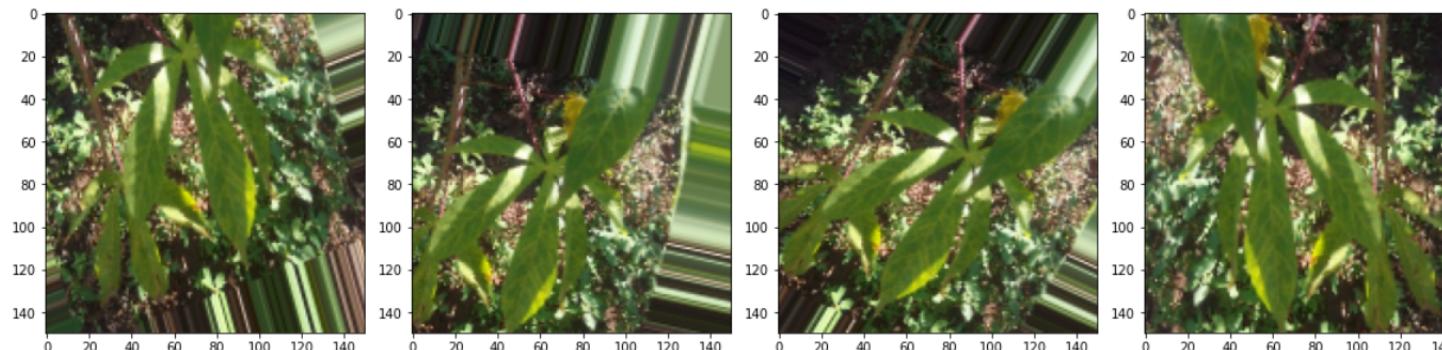
2. Preprocessing

```

1 img_size = 150
2 output_size = 5 # Number of output
3 batch_size = 128
4
5 train_datagen = ImageDataGenerator( rescale=1.0/255.0,
6                                     rotation_range=30,
7                                     zoom_range=0.15,
8                                     width_shift_range=0.2,
9                                     height_shift_range=0.2,
10                                    shear_range=0.15,
11                                    horizontal_flip=True,
12                                    fill_mode="nearest" )
13
14 val_datagen = ImageDataGenerator(rescale=1.0/255.0)
15
16 test_datagen = ImageDataGenerator(rescale=1.0/255.0)
17
18 # prepare iterators
19 train_dataloader = train_datagen.flow_from_directory(train_dir,
20                                                       batch_size=batch_size,
21                                                       target_size=(img_size,
22                                                       img_size))
23 valid_dataloader = val_datagen.flow_from_directory(valid_dir,
24                                                       batch_size=batch_size,
25                                                       target_size=(img_size, img_size))
26
27 test_dataloader = test_datagen.flow_from_directory(test_dir,
28                                                       batch_size=batch_size,
29                                                       shuffle = False,
30                                                       target_size=(img_size, img_size))
31

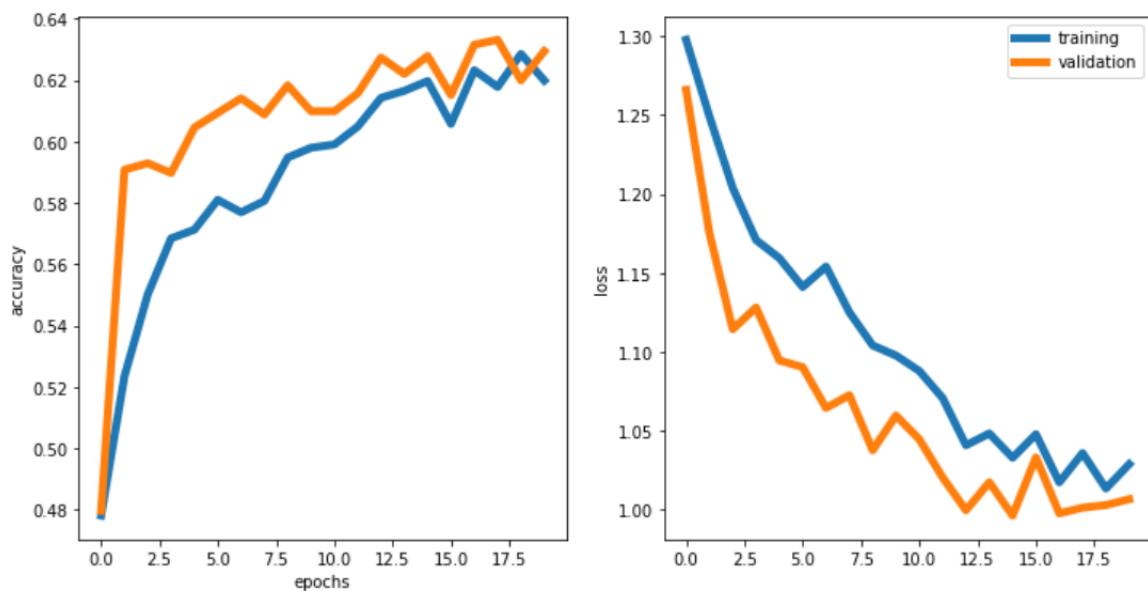
```

```
1 def show_aug(images):
2     fig, axis = plt.subplots(1, 5, figsize=(20, 20))
3     axis = axis.flatten()
4     for img, ax in zip(images, axis):
5         ax.imshow(img)
6     plt.tight_layout()
7     plt.show()
8
9 imgs = [train_dataloader[0][0][0] for i in range(5)]
10 show_aug(imgs)
11
```



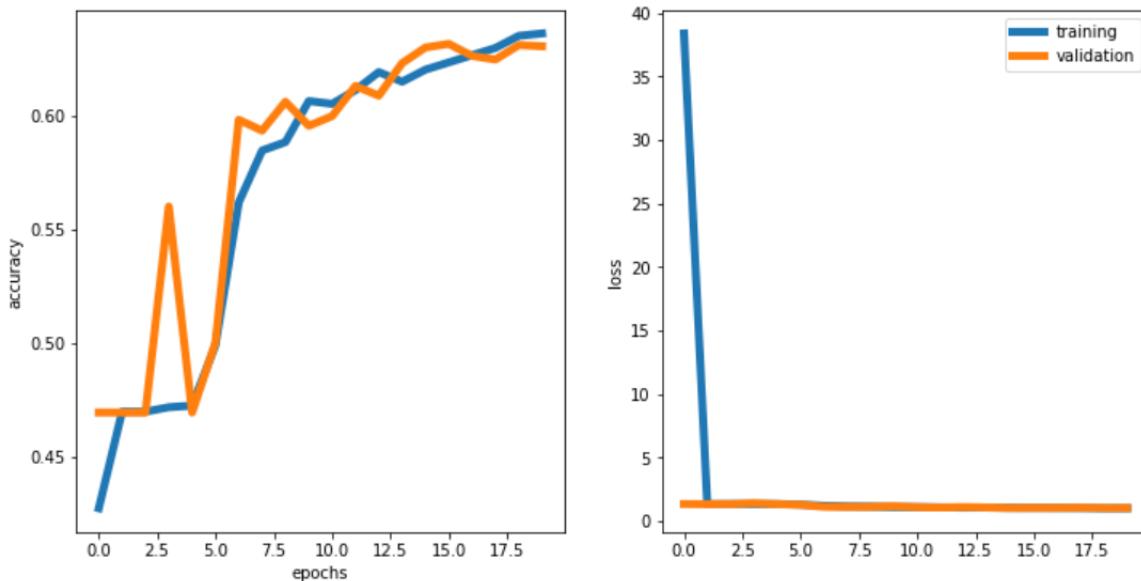
Hình 12: Cassava Leaf Disease after preprocessing

3. Áp dụng model LeNet.



Hình 13: LeNet results

4. Áp dụng model VGG-16.



Hình 14: VGG-16 results

Reading Assignment - Các bạn đọc thêm và suy nghĩ với cách dùng filter 1x1

- [Đọc bài viết tại đây](#)
- Conv 1x1 có thể thay thế fully-connected layer (FCL). FCL yêu cầu biết trước kích thước (số node) của input, trong khi Conv không yêu cầu điều này. Do đó, dùng conv 1x1 khi thay thế FCL giúp model chạy được với dynamic input size.
- Conv 1x1 còn giúp thay đổi số filter (độ sâu) của input feature map.

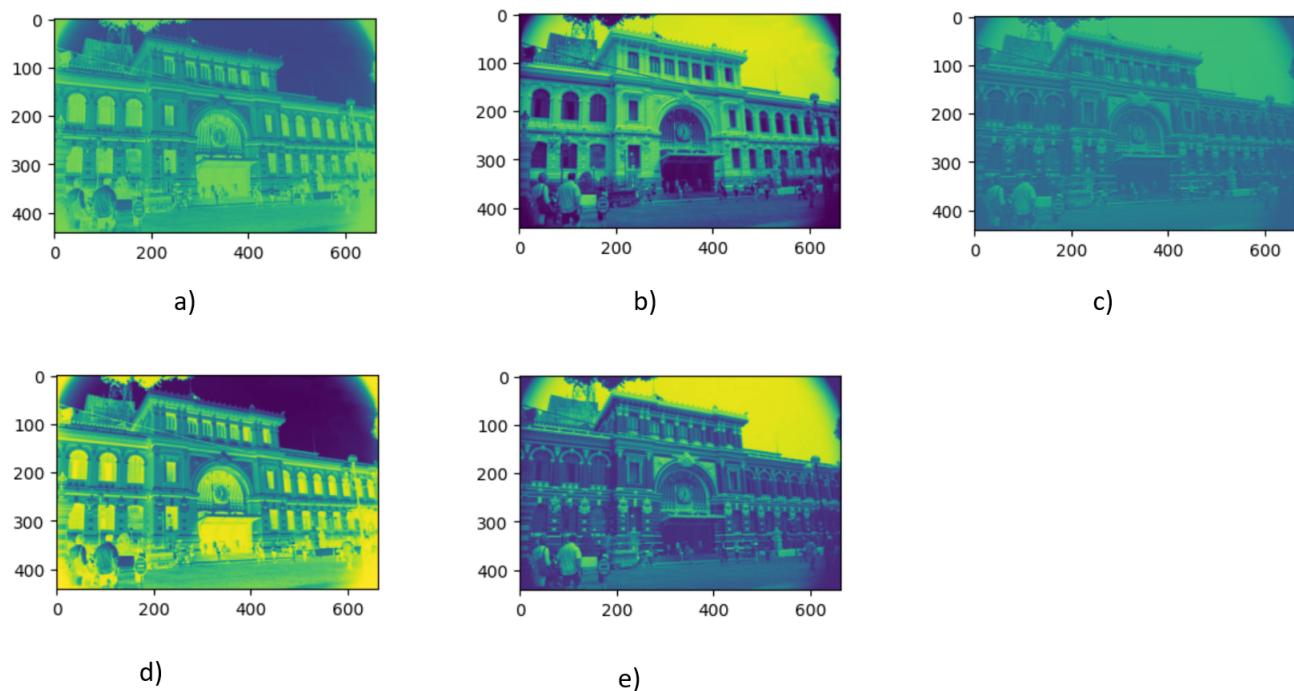
Optional - Các bạn đọc hiểu bài Tutorial Basic CNN và rút ra những điểm cơ bản

Phần II: Trắc nghiệm

1. Các hình bên dưới, đâu là hình của feature map thứ 3 khi áp dụng kernel=2x2, stride=1.



Hình 15: Bưu điện TP.HCM



Hình 16: 5 feature-map

2. Pooling có tác dụng:

- (a) tăng kích thước ảnh
- (b) giảm kích thước ảnh
- (c) cả hai câu trên đều đúng
- (d) tất cả đều sai

3. Kết quả phép tính convolution của input và kernel sẽ là

$$\begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline 3 & 4 & 5 \\ \hline 6 & 7 & 8 \\ \hline \end{array} * \begin{array}{|c|c|} \hline -1 & 1 \\ \hline 2 & 2 \\ \hline \end{array} =$$

Hình 17: Convolution

| | |
|----|----|
| 16 | 19 |
| 27 | 31 |

| | |
|----|----|
| 15 | 19 |
| 27 | 31 |

| | |
|----|----|
| 16 | 19 |
| 28 | 31 |

| | |
|----|----|
| 15 | 20 |
| 26 | 31 |

a) b) c) d)

Hình 18: Chọn kết quả phù hợp

4. Kết quả của phép tính Max pooling sẽ là

| | | |
|----|---|---|
| 10 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

2 x 2
Max

Hình 19: Max pooling

| | |
|----|----|
| 10 | 19 |
| 7 | 8 |

a)

| | |
|----|---|
| 10 | 5 |
| 7 | 8 |

b)

| | |
|----|---|
| 16 | 5 |
| 8 | 8 |

c)

| | |
|----|----|
| 15 | 10 |
| 6 | 7 |

d)

Hình 20: Chọn kết quả phù hợp

5. Kết quả của phép tính Avg pooling sẽ là

| | | |
|----|---|---|
| 10 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

2 x 2
avg

Hình 21: Avg pooling

| | |
|---|-----|
| 4 | 4.5 |
| 3 | 8 |

a)

| | |
|---|---|
| 5 | 5 |
| 5 | 8 |

b)

| | |
|---|---|
| 4 | 5 |
| 7 | 8 |

c)

| | |
|-----|---|
| 4.5 | 3 |
| 5 | 6 |

d)

Hình 22: Chọn kết quả phù hợp

6. Kết quả của phép tính Convolution với kernel 3x3 sẽ là

| | | | | |
|-----|-----|-----|-----|-----|
| 52 | 151 | 246 | 207 | 90 |
| 250 | 421 | 236 | 144 | 41 |
| 35 | 44 | 228 | 102 | 43 |
| 150 | 124 | 214 | 59 | 52 |
| 28 | 23 | 21 | 144 | 254 |

*

| | | |
|---|---|----|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

=

Hình 23: Convolution với kernel 3x3

| | | |
|------|------|------|
| -373 | 162 | -160 |
| -200 | 280 | 540 |
| -250 | -140 | 113 |

| | | |
|-----|-----|------|
| 135 | 50 | -129 |
| 128 | 359 | -150 |
| 250 | 70 | 90 |

| | | |
|------|------|-----|
| -373 | 163 | 536 |
| -243 | 284 | 542 |
| -250 | -114 | 114 |

| | | |
|-----|-----|------|
| 113 | 50 | -190 |
| 115 | 70 | -21 |
| 120 | 100 | -51 |

a)

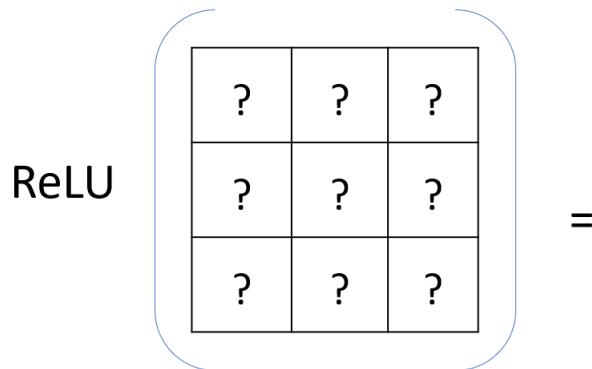
b)

c)

d)

Hình 24: Chọn kết quả phù hợp

7. Lấy kết quả thu được ở câu 6, cho qua hàm ReLU, với `max_value=None`, sẽ thu được kết quả:



Hình 25: ReLU activation

| | | |
|---|-----|-----|
| 0 | 163 | 536 |
| 0 | 284 | 542 |
| 0 | 0 | 114 |

a)

| | | |
|---|-----|-----|
| 0 | 150 | 534 |
| 0 | 280 | 534 |
| 0 | 0 | 113 |

b)

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

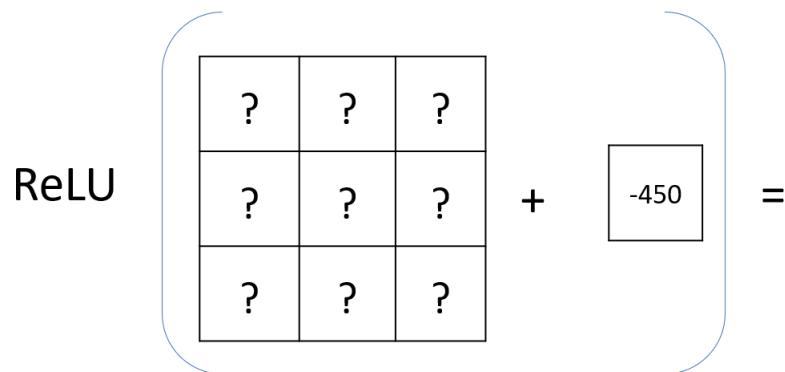
c)

| | | |
|---|-----|----|
| 0 | 240 | 44 |
| 0 | 30 | 28 |
| 0 | 0 | 10 |

d)

Hình 26: Chọn kết quả phù hợp

8. Lấy kết quả thu được ở câu 7, cho qua hàm ReLU, với `max_value=None` và `Bias=-450`, sẽ thu được kết quả:



Hình 27: ReLU activation với Bias

| | | |
|---|----|----|
| 0 | 0 | 58 |
| 0 | 10 | 46 |
| 0 | 0 | 14 |

| | | |
|---|---|----|
| 0 | 0 | 86 |
| 0 | 0 | 92 |
| 0 | 0 | 0 |

| | | |
|---|---|----|
| 0 | 1 | 20 |
| 0 | 1 | 89 |
| 0 | 0 | 30 |

| | | |
|---|----|----|
| 0 | 20 | 46 |
| 0 | 10 | 28 |
| 0 | 0 | 17 |

a) b) c) d)

Hình 28: Chọn kết quả phù hợp

9. Trở lại mô hình LeNet, nếu thay số lượng filter ở layer C1 thành 12, thì layer S2 sẽ có size x feature map là:

| Layer | Layer type | Feature Map | Size | Kernel | Stride | Activation | Padding |
|-------|--------------------------|-------------|-------|--------|--------|------------|---------|
| Input | Image | 1 | 32x32 | - | - | - | - |
| C1 | Convolution (Conv2D) | 12 | 28x28 | 5x5 | 1 | relu | no |
| S2 | Sub sampling (AvgPool2D) | 1 | 14x14 | 2x2 | 2 | - | same |

Hình 29: Thay đổi số lượng filter

- (a) (14, 14, 12)
- (b) (14, 14, 6)
- (c) (7, 7, 12)
- (d) (14, 14, 14)

10. Tiếp tục kết quả câu 9, nếu thay Stride = 1 tại layer S4, thì output của layer này sẽ có size x feature map là:

| Layer | Layer type | Feature Map | Size | Kernel | Stride | Activation | Padding |
|-------|--------------------------|-------------|-------|--------|--------|------------|---------|
| Input | Image | 1 | 32x32 | - | - | - | - |
| C1 | Convolution (Conv2D) | 12 | 28x28 | 5x5 | 1 | relu | no |
| S2 | Sub sampling (AvgPool2D) | 1 | 14x14 | 2x2 | 2 | - | same |
| C3 | Convolution (Conv2D) | 16 | 10x10 | 5x5 | 1 | relu | no |
| S4 | Sub sampling (AvgPool2D) | 1 | 5x5 | 2x2 | 1 | - | same |

Hình 30: Thay đổi bước trượt Stride

- (a) (5, 5, 16)
- (b) (7, 7, 16)
- (c) (9, 9, 16)
- (d) (5, 5, 8)