Activity #4

2D CONVOLUTION (CNN SIMULATION)

Topics

4.1 VGG16 Model Parameters

4.2 Image Preparation (from scratch)

4.3 Cov2D

Libraries

1. import numpy as np
2. import cv2
3. from matplotlib import pyplot as plt
4. from keras.models import Model
5. from tensorflow.keras.applications.vgg16 import VGG16
6. from keras.applications.vgg16 import preprocess_input
7. from keras.preprocessing.image import img_to_array
8. from numpy import expand_dims
9. from scipy import signal

4.1 VGG16 MODEL PARAMETERS

4.1 VGG16 Model Parameters

1

- Read image file

2

- # Load VGG16 model from tensorflow.keras
- model = VGG16()
- # model detail
- model.summary()

3

- # retrieve kernel weights from the 1st Convolutional layer
- kernels, biases = model.layers[1].get_weights()
- # View CNN layer 1 architecture
- model.layers[1].get_config()

4

- # Preprocess Image using keras and numpy
- # convert the image to an array
- img = img_to_array(img)
- # expand dimensions so that it represents a single 'sample'
- # -> reshape 3D(H,W,Ch) image to 4D image (sample,H,W,Ch)
- img = expand_dims(img, axis=0)
- # prepare the image (e.g. scale pixel values for the vgg)
- img_ready = preprocess_input(img)

4.1

VGG16 Model Parameters

```
kernel #  0
************************************
Channel #  0
[[ 0.42947057  0.373467   -0.06136011]
 [ 0.27476987  0.03868078 -0.36722335]
 [-0.05746817 -0.26224968 -0.35009676]]
Min coefficients  -0.36722335
----------------------------------------
Channel #  1
[[ 0.55037946  0.44007453 -0.08138704]
 [ 0.34573907  0.04063221 -0.4535013 ]
 [-0.05863491 -0.33066967 -0.4850302 ]]
Min coefficients  -0.4850302
----------------------------------------
Channel #  2
[[ 0.4800154   0.4085474  -0.06514555]
 [ 0.31047726  0.05020237 -0.40338343]
 [-0.05087169 -0.2852275  -0.41851634]]
```

```
kernel #  1
************************************
Channel #  0
[[0.11727387 0.16206263 0.135694  ]
 [0.14835016 0.20229845 0.16168842]
 [0.12934428 0.17157242 0.13871045]]
Min coefficients  0.11727387
----------------------------------------
Channel #  1
[[0.02087744 0.04734124 0.04185439]
 [0.03104937 0.06581022 0.0462575 ]
 [0.03167877 0.05471011 0.04231958]]
Min coefficients  0.020877438
----------------------------------------
Channel #  2
[[-0.17269668 -0.17037505 -0.15435153]
 [-0.18760149 -0.17757156 -0.17439997]
 [-0.16600266 -0.16666673 -0.1570488 ]]
```

View Filter Kernels

4.1

VGG16 Model Parameters

```
{'name': 'block1_conv1',
 'trainable': True,
 'dtype': 'float32',
 'filters': 64,
 'kernel_size': (3, 3),
 'strides': (1, 1),
 'padding': 'same',
 'data_format': 'channels_last',
 'dilation_rate': (1, 1),
 'groups': 1,
 'activation': 'relu',
 'use_bias': True,
 'kernel_initializer': {'class_name': 'GlorotUniform',
  'config': {'seed': None}},
 'bias_initializer': {'class_name': 'Zeros', 'config': {}},
 'kernel_regularizer': None,
 'bias_regularizer': None,
 'activity_regularizer': None,
 'kernel_constraint': None,
 'bias_constraint': None}
```

View CNN Layer 1 Architecture of VGG16

4.1 VGG16 Model Parameters

5

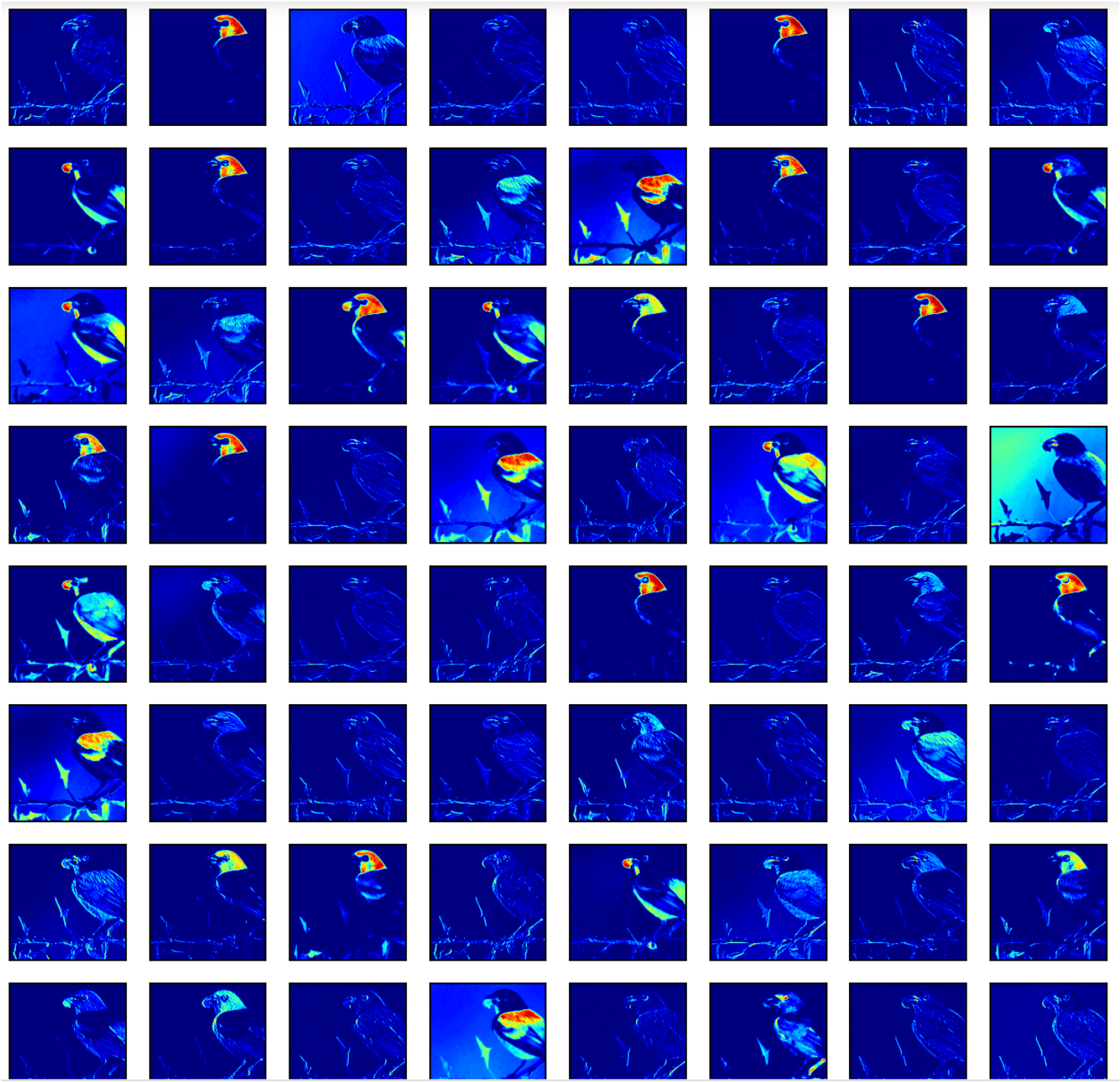
- # Extract Model CNN Layer 1
- model = Model(inputs=model.inputs, outputs=model.layers[1].output)
- model.summary()

6

- # Extract Results from CNN Layer 1 called feature map (shape = (sample = 1, 224, 224, n_filters))
- # CNN Layer 1 -> n_filters = 64
- feature_maps = model.predict(img_ready)

7

- # Display images of feature_maps
  - Subplot() 8 x 8 images



4.1

VGG16 Model Parameters

View Results of CNN layer 1

(from 64 filter kernels)

4.2 IMAGE PREPARATION (FROM SCRATCH)

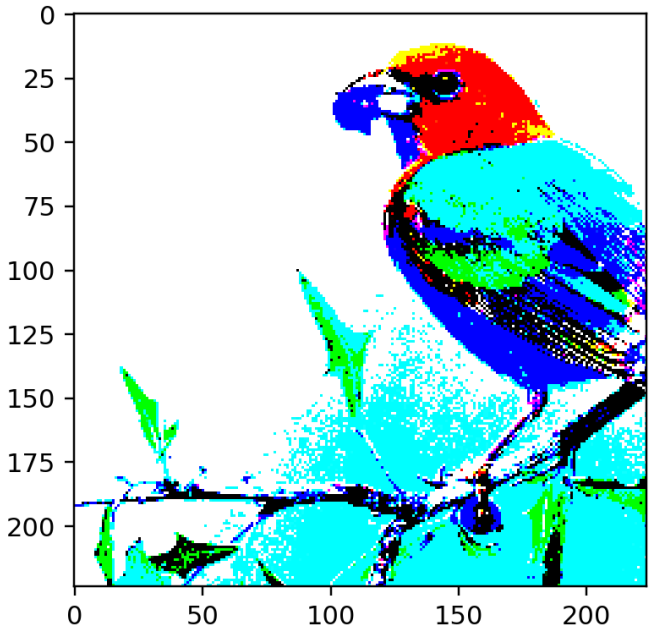
4.2 Prepare input image from scratch

1
- # Image reshape from 3D image (H, W, Ch) -> 4D image (1, H, W, Ch)
  - Read image
  - img.reshape()

2
- # Image resize (H, W) -> (224, 224)
  - # Be careful aspect ratio
  - cv2.resize()

3
- # Image subtract dataset mean of R, G, B
  - img_mean = [123.68, 116.779, 103.939] -> [meanR, meanG, meanB]
  - Image – img_mean ทำด้วย mean แต่ละ channel R, G, B

4
- # Color conversion
  - RGB -> BGR



4.2

Prepare input image from scratch

Image after preprocess with
- mean subtraction
- RGB -> BGR

4.3 CONV2D()

4.3 Conv2D()

1

- # operate 2D convolution to image from 4.2 (imgBGR)
  - # image convolution with kernel แยกแต่ละ color channel (ทำทุก color channel)
  - Img_result[: , :, 0] = signal.convolve2d(imgBGR[: , : , 0], kernels[: , : , 0, i] , mode='same',boundary='fill', fillvalue=0)
    - # -> zero padding

2

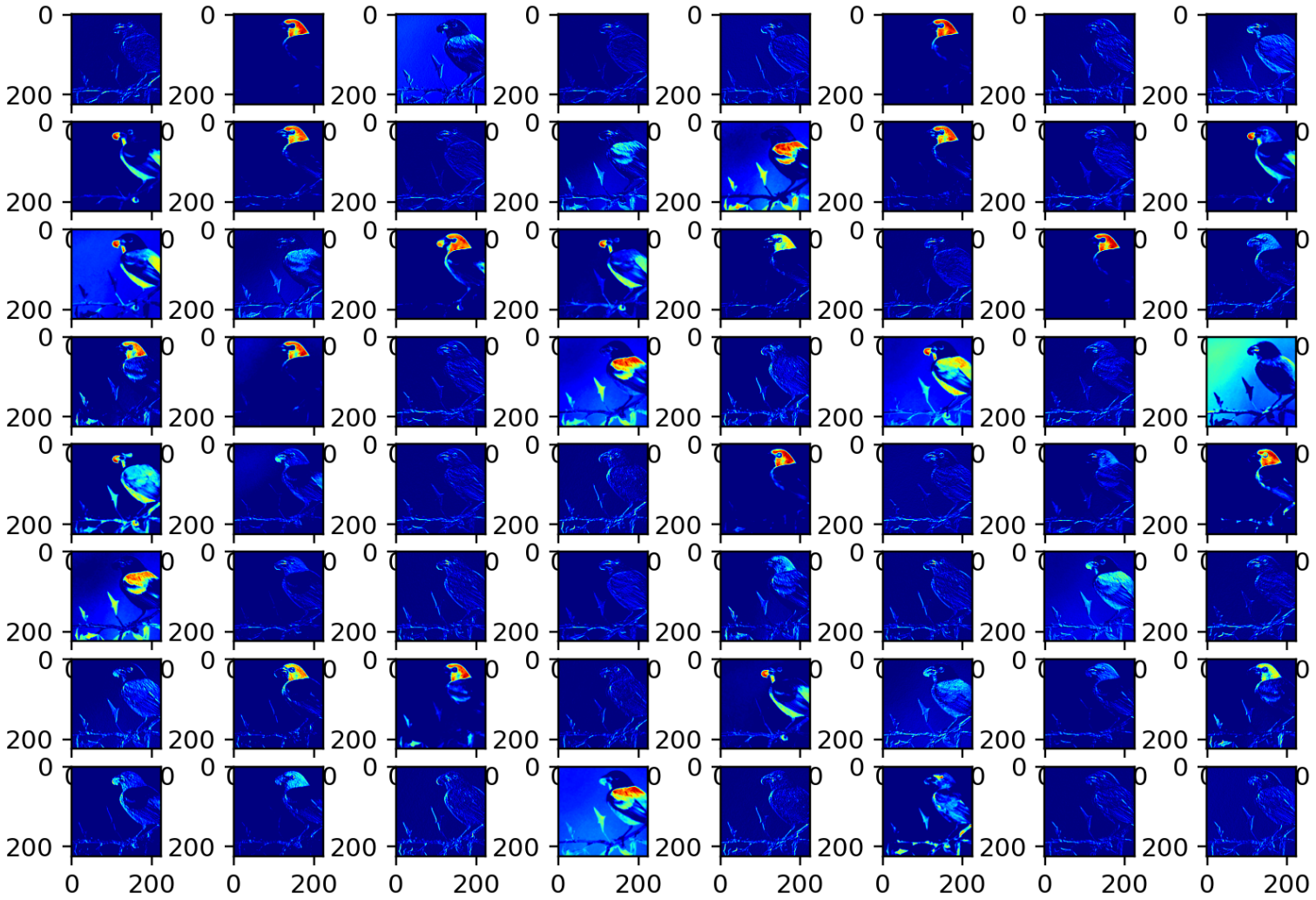
- # Sum image convolutional results of B,G,R
  - Image_sum = img_result[: , : , B] + img_result[: , : , G] + img_result[: , : , R]

3

- # Activation Function
  - ถ้าค่าใน image_sum < 0 -> ให้เปลี่ยนค่าเป็น 0

4

- # Display images of feature_maps
  - Subplot() 8 x 8 images



4.3

conv2D()

View Results of Conv2D

(from 64 filter kernels)