Lab #3

2D CONVOLUTION (CNN SIMULATION)

Topics

3.1 VGG16 Model Parameters

3.2 Image Preparation (from scratch)

3.3 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

3.1 VGG16 MODEL PARAMETERS

3.1 VGG16 Model Parameter

- แสดงพารามิเตอร์โครงสร้างโมเดล VGG16
  - # Load VGG16 model from tensorflow.keras
    - model = VGG16()
  - แสดงโครงสร้าง filter kernel แต่ละ Layer ของ VGG16
    - model.summary()
  - ดึง Array ของ weight และ bias ใน Layer 1 ของ VGG16
    - # retrieve kernel weights from the 1st Convolutional layer
    - kernels, biases = model.layers[1].get_weights()
  - ดึงโครงสร้างการทำงานทั้งหมดของ VGG16
    - # View CNN layer 1 architecture
    - model.layers[1].get_config()

3.1

VGG16 Model Weights

```
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 ]]
```

```
Bias = [ 0.73429835  0.09340367  0.06775674  0.8862966   0.25994542  0.66426694
 -0.01582893  0.3249065   0.68600726  0.06247932  0.58156496  0.2361475
  0.69694996  0.19451167  0.4858922   0.44571847  0.5113422   0.208576
  0.57557714  0.33199573  0.4997983   0.7117759   0.30284074  0.7082712
  0.04548979  0.7446502   0.29845494  0.48211655  0.81658626  0.62603897
  0.3768093   2.064037    0.77311045  0.3459577   0.6130958   0.65459156
  0.39045632  0.50869167  0.2625384   0.23669638  0.07971057  1.1179353
  0.26129362  0.8697589   0.21543622  0.78007823  0.37015367  0.47993386
  0.4313978   0.5084194   0.23049663  0.7636527   0.35419866  0.45794216
  0.4662595   0.09850298  0.3803252   0.66880196  0.4015123   0.90510356
  0.43166816  1.302014    0.5306885   0.48993504]
```

View Filter Kernels

3.1

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

3.1 VGG16 Model Parameter

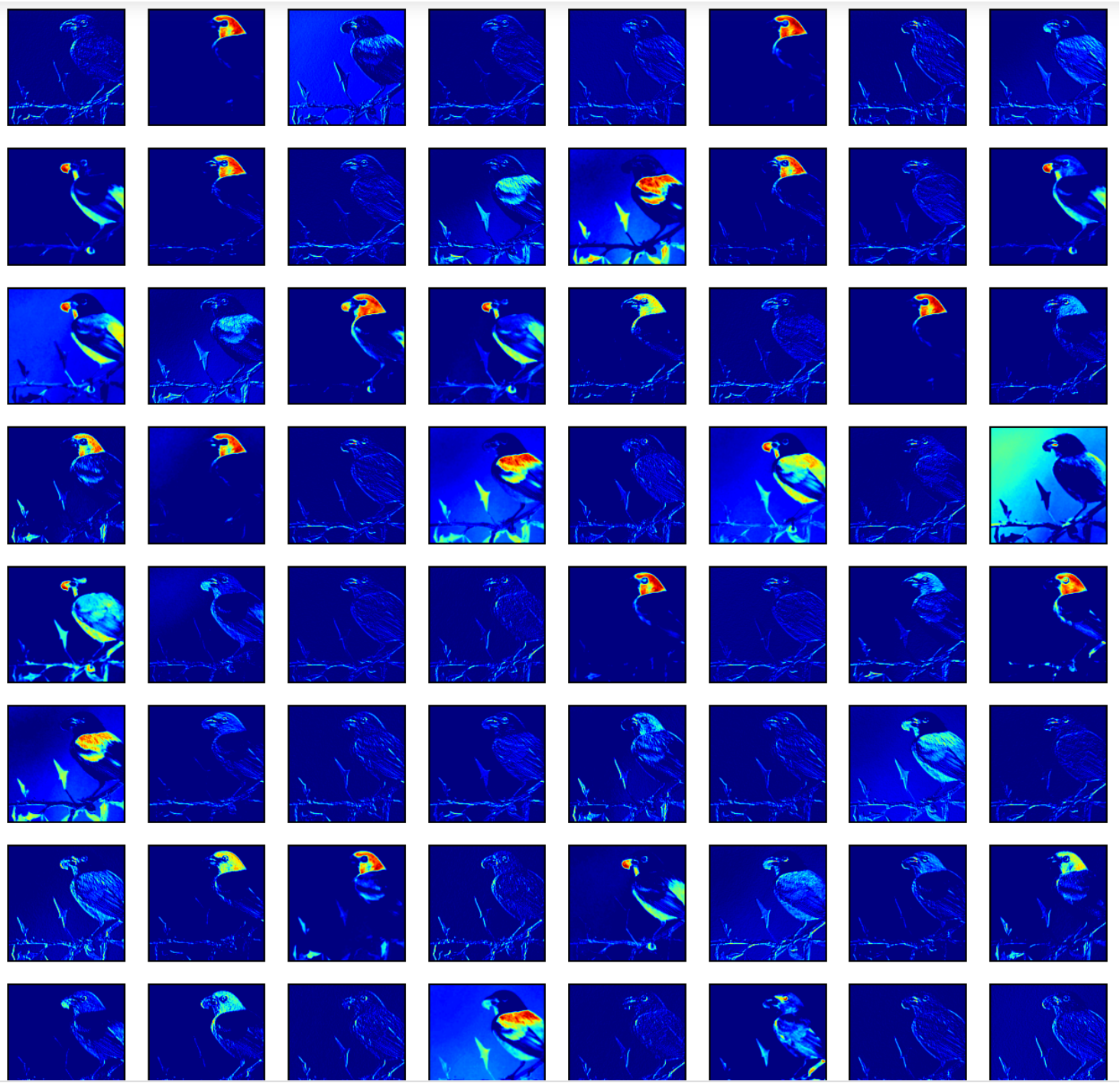
- อ่านไฟล์ภาพที่ต้องการนำเข้าโมเดล
- ปรับโครงสร้าง Image Array from 3D to 4D
  - # convert the image to an array
    - img = img_to_array(img)
  - # expand dimensions so that it represents a single 'sample’
  - # reshape 3D(H,W,Ch) image to 4D image (sample=1,H,W,Ch)
    - img = expand_dims(img, axis=0)
- ทำ Preprocess Image using keras and numpy
  - # prepare the image (e.g. scale pixel values for the vgg)
    - img = preprocess_input(img)
- แสดงภาพหลังทำ preprocess

3.1 VGG16 Model Parameter

- ดึงโมเดล VGG16 เฉพาะ CNN Layer 1 มาใช้
  - model = Model(inputs=model.inputs, outputs=model.layers[1].output)
  - model.summary()

- ทำ model prediction เพื่อดึง Feature Map (ภาพผลลัพธ์ จากแต่ละ CNN node หลัง Activation function)
  - # 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)

- แสดงภาพ Feature map ผลลัพธ์จาก 64 node ของ CNN Layer 1

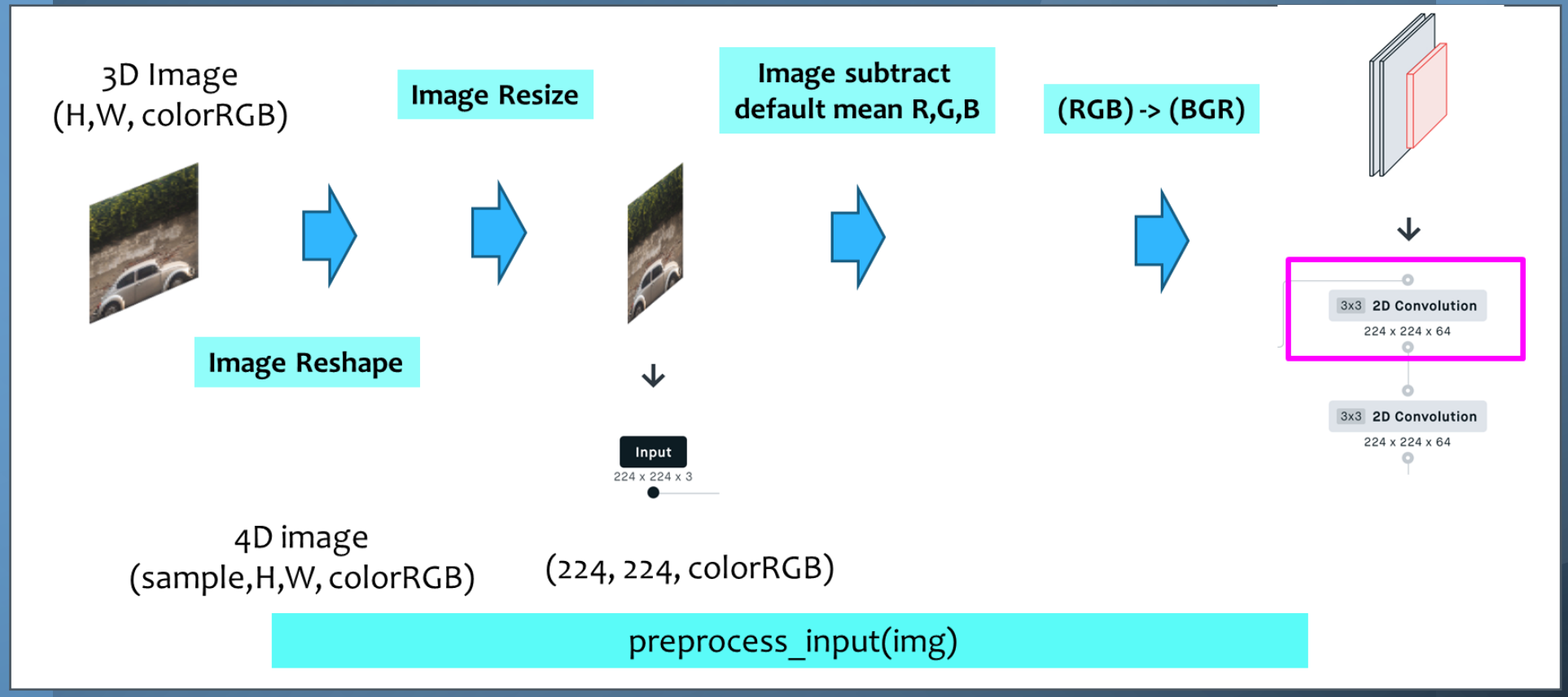


3.1

VGG16 Model Parameters

View Results of CNN layer 1

(from 64 filter kernels)




```
tf.keras.applications.vgg16.VGG16(
    include_top=True,
    weights='imagenet',
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation='softmax'
)
```

3.2 IMAGE PREPARATION (FROM SCRATCH)

★ **Note:** each Keras Application expects a specific kind of input preprocessing. For VGG16, call `tf.keras.applications.vgg16.preprocess_input` on your inputs before passing them to the model. `vgg16.preprocess_input` will convert the input images from RGB to BGR, then will zero-center each color channel with respect to the ImageNet dataset, without scaling.

https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg16/VGG16

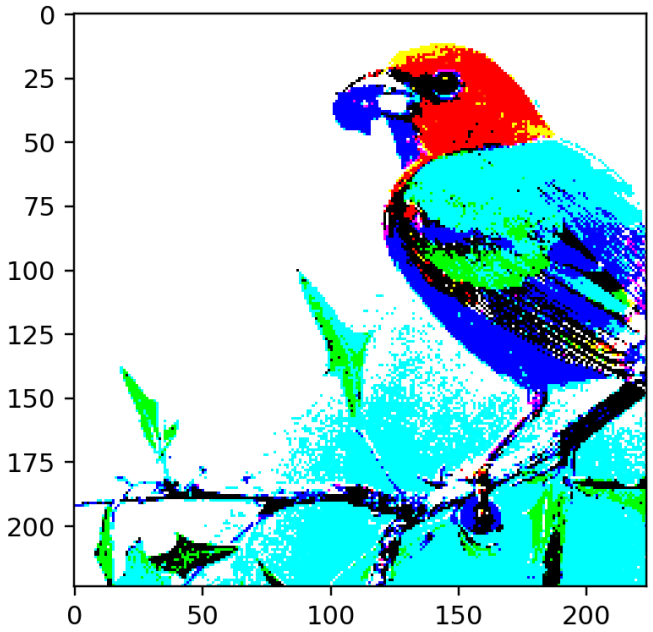
3.2 Image Preparation (from scratch)

- อ่านไฟล์ภาพที่ต้องการนำเข้าโมเดล
- ปรับโครงสร้าง Image Array from 3D to 4D
  - # Image reshape from 3D image (H, W, Ch) -> 4D image (1, H, W, Ch)
  - img.reshape()

- ทำการปรับขนาดภาพเพื่อให้ตรงกับขนาดอินพุทของโมเดล VGG16
  - # Image resize (H, W) -> (224, 224)
  - # Be careful aspect ratio
    - cv2.resize()

- ทำการเลื่อนระดับเฉดสี ให้อ้างอิงกับค่า เฉลี่ยเฉดสีของชุดข้อมูลที่ถูกใช้ Pretrain (ค่าอ้างอิงเฉลี่ยที่ให้ ImageNet) -> mean shift (subtraction)
  - # Image subtract dataset mean R, G, B of ImageNet dataset
  - img_mean = [123.68, 116.779, 103.939] -> [meanR, meanG, meanB]
  - คำนวน image – img_mean ทำแต่ละ channel R, G, B

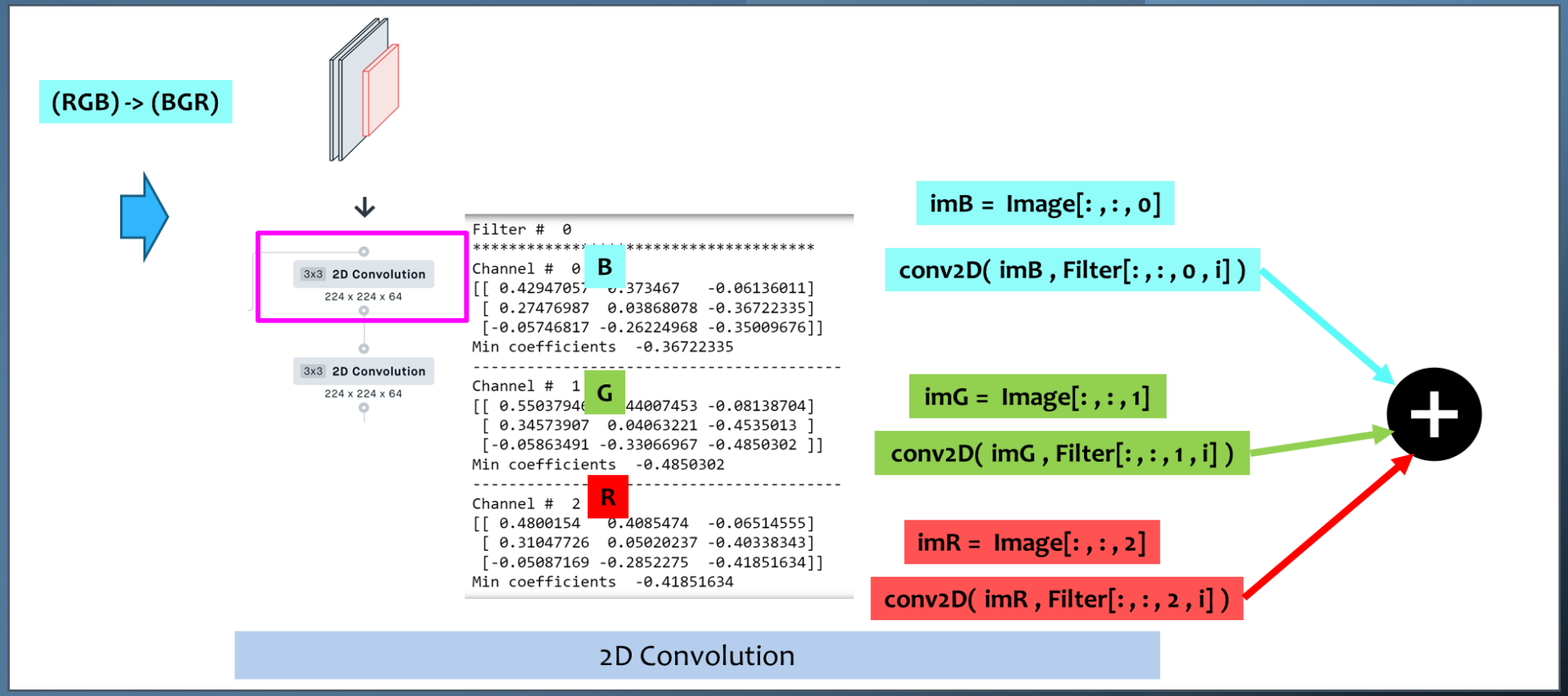
- แสดงภาพผลลัพธ์



3.2

Prepare input image from scratch

Image after preprocess with
- mean subtraction


(RGB) -> (BGR)
3x3 2D Convolution
224 x 224 x 64
3x3 2D Convolution
224 x 224 x 64
Filter # 0
Channel # 0
B
[[0.42947057 0.373467 -0.06136011]
[0.27476987 0.03868078 -0.36722335]
[-0.05746817 -0.26224968 -0.35009676]]
Min coefficients -0.36722335
Channel # 1
G
[[0.5503794 44007453 -0.08138704]
[0.34573907 0.04063221 -0.4535013]
[-0.05863491 -0.33066967 -0.4850302]]
Min coefficients -0.4850302
Channel # 2
R
[[0.4800154 0.4085474 -0.06514555]
[0.31047726 0.05020237 -0.40338343]
[-0.05087169 -0.2852275 -0.41851634]]
Min coefficients -0.41851634
imB = Image[: , : , 0]
conv2D(imB , Filter[: , : , 0 , i])
imG = Image[: , : , 1]
conv2D(imG , Filter[: , : , 1 , i])
imR = Image[: , : , 2]
conv2D(imR , Filter[: , : , 2 , i])
2D Convolution

3.3 CONV2D()

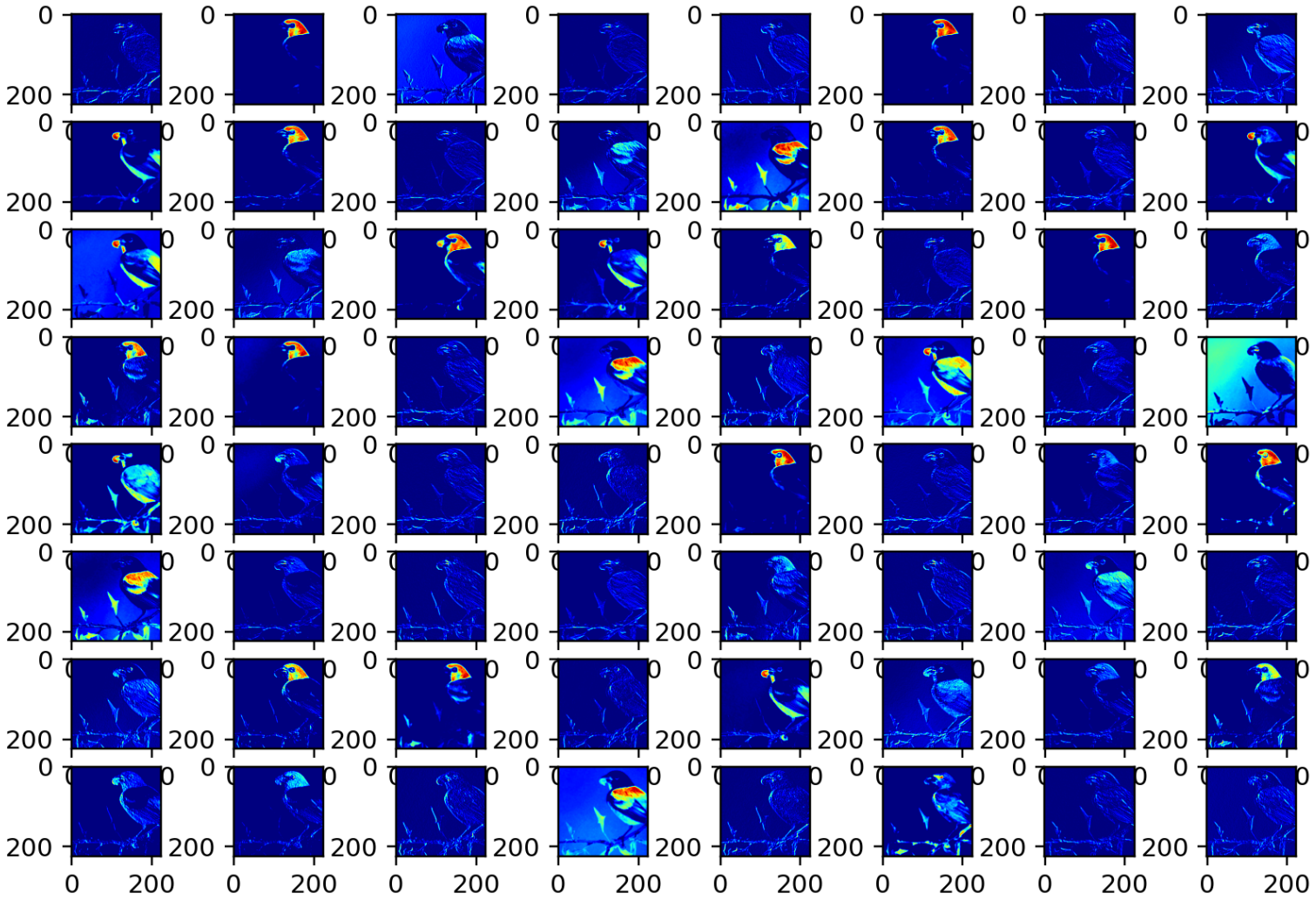
3.3 2DImage Convolution using filter kernels from VGG16 Layer 1

- ทำ convolution กับภาพที่ผ่านการทำ mean shift (subtraction) ในขั้นตอน Preprocess 3.2
  - # image convolution ด้วย kernel แยกแต่ละ color channel (ทำทุก color channel)
  - Ex. 2Dconv(Ch #0 (Blue) with filter kernel ลำดับ i สำหรับ (Ch #0 (Blue)) โดย 0 <= i <64
    - img_result[: , :, 0] = signal.convolve2d(imgBGR[: , : , 0], kernels[: , : , 0, i] , mode='same',boundary='fill', fillvalue=0)

- ทำการรวมภาพผลลัพธ์จากทุก color channel
  - # Sum image convolutional results of B,G,R
  - image_sum = img_result[: , : ,0] + img_result[: , : , 1] + img_result[: , : , 2]

- ทำการคำนวน Activation Function กำหนดให้ใช้ฟังก์ชัน Relu
  - Relu = max(img, 0)
    - Ex. ตัวอย่างคำสั่ง Replace array component with a value
      - my_array[my_array > 8] = 20

- แสดงภาพผลลัพธ์



3.3

conv2D()

View Results of Conv2D

(from 64 filter kernels)