Convolution Neural Network

Dr. Kelly Trinh

Housekeeping

- Assessment 2 is due on Sunday, Week 4

Housekeeping

- Assessment 2 is due on Sunday, Week 4
- To avoid a surcharge from AWS, please
    - ensure to stop notebook instance and delete endpoints when they are not used

Housekeeping

- Assessment 2 is due on Sunday, Week 4
- To avoid a surcharge from AWS, please
  - ensure to stop notebook instance and delete endpoints when they are not used
  - request SageMaker Training Jobs or SageMaker HyperParameter Optimization only request SageMaker Notebook instance if you work with large data (i.e. $>$1GB)

Housekeeping

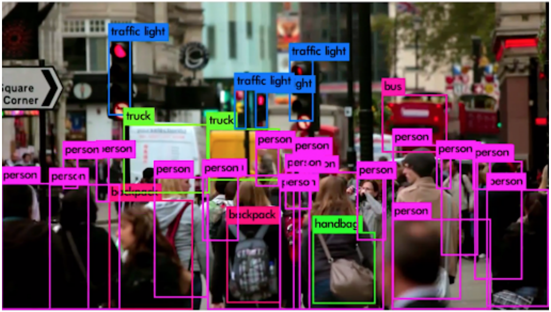
- Assessment 2 is due on Sunday, Week 4
- To avoid a surcharge from AWS, please
  - ensure to stop notebook instance and delete endpoints when they are not used
  - request SageMaker Training Jobs or SageMaker HyperParameter Optimization only request SageMaker Notebook instance if you work with large data (i.e. $>$1GB)
  - set up a budget and threshold for alter. AWS will notify when the charges reach the threshold of your budget.

Housekeeping

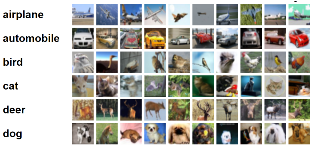
- Assessment 2 is due on Sunday, Week 4
- To avoid a surcharge from AWS, please
  - ensure to stop notebook instance and delete endpoints when they are not used
  - request SageMaker Training Jobs or SageMaker HyperParameter Optimization only request SageMaker Notebook instance if you work with large data (i.e. $>$1GB)
  - set up a budget and threshold for alter. AWS will notify when the charges reach the threshold of your budget.

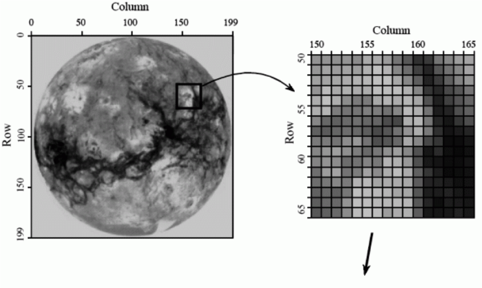
- Convolution neural network (CNN) (Week 6 online content)
- Advanced CNN

- Image classification: classify an image into a class. In other words, provide probability that image belong to a particular class
- Object localization: image classification + identifying a location of one object with bounding box.
- Object detection : image classification + identifying location of each object with bounding boxes.



Source: Medium



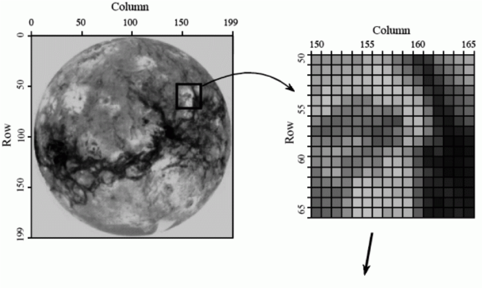


Column

Row	150					155					160					165
50	183	183	181	184	177	200	200	189	159	135	94	105	160	174	191	196
	186	195	190	195	191	205	216	206	174	153	112	80	134	157	174	196
	194	196	198	201	206	209	215	216	199	175	140	77	106	142	170	186
	184	212	200	204	201	202	214	214	214	205	173	102	84	120	134	159
	202	215	203	179	165	165	199	207	202	208	197	129	73	112	131	146
55	203	208	166	159	160	168	166	157	174	211	204	158	69	79	127	143
	174	149	143	151	156	148	146	123	118	203	208	162	81	58	101	125
	143	137	147	153	150	140	121	133	157	184	203	164	94	56	66	80
	164	165	159	179	188	159	126	134	150	199	174	119	100	41	41	58
	173	187	193	181	167	151	162	182	192	175	129	60	88	47	37	50
60	172	184	179	153	158	172	163	207	205	188	127	63	56	43	42	55
	156	191	196	159	167	195	178	203	214	201	143	101	69	38	44	52
	154	163	175	165	207	211	197	201	201	199	138	79	76	67	51	53
	144	150	143	162	215	212	211	209	197	198	133	71	69	77	63	53
	140	151	150	185	215	214	210	210	211	209	135	80	45	69	66	60
65	135	143	151	179	213	216	214	191	201	205	138	61	59	61	77	63

Source: Venus planet from Steven Smith, 1999

- Computers see images as a matrix of numbers (pixels) in the range of [0,255]. Each pixel represents a number.

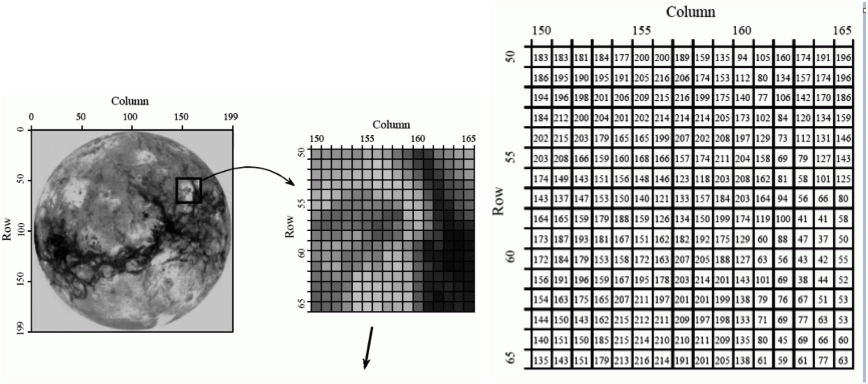


Column

Row	150					155					160					165
50	183	183	181	184	177	200	200	189	159	135	94	105	160	174	191	196
	186	195	190	195	191	205	216	206	174	153	112	80	134	157	174	196
	194	196	198	201	206	209	215	216	199	175	140	77	106	142	170	186
	184	212	200	204	201	202	214	214	214	205	173	102	84	120	134	159
	202	215	203	179	165	165	199	207	202	208	197	129	73	112	131	146
55	203	208	166	159	160	168	166	157	174	211	204	158	69	79	127	143
	174	149	143	151	156	148	146	123	118	203	208	162	81	58	101	125
	143	137	147	153	150	140	121	133	157	184	203	164	94	56	66	80
	164	165	159	179	188	159	126	134	150	199	174	119	100	41	41	58
	173	187	193	181	167	151	162	182	192	175	129	60	88	47	37	50
60	172	184	179	153	158	172	163	207	205	188	127	63	56	43	42	55
	156	191	196	159	167	195	178	203	214	201	143	101	69	38	44	52
	154	163	175	165	207	211	197	201	201	199	138	79	76	67	51	53
	144	150	143	162	215	212	211	209	197	198	133	71	69	77	63	53
	140	151	150	185	215	214	210	210	211	209	135	80	45	69	66	60
65	135	143	151	179	213	216	214	191	201	205	138	61	59	61	77	63

Source: Venus planet from Steven Smith, 1999

- Computers see images as a matrix of numbers (pixels) in the range of [0,255]. Each pixel represents a number.
- Image are often represented as 3 dimensional array of *height* $\times$ *width* $\times$ *channels*



Row \ Column	150					155					160					165
50	183	183	181	184	177	200	200	189	159	135	94	105	160	174	191	196
	186	195	190	195	191	205	216	206	174	153	112	80	134	157	174	196
	194	196	198	201	206	209	215	216	199	175	140	77	106	142	170	186
	184	212	200	204	201	202	214	214	214	205	173	102	84	120	134	159
	202	215	203	179	165	165	199	207	202	208	197	129	73	112	131	146
55	203	208	166	159	160	168	166	157	174	211	204	158	69	79	127	143
	174	149	143	151	156	148	146	123	118	203	208	162	81	58	101	125
	143	137	147	153	150	140	121	133	157	184	203	164	94	56	66	80
	164	165	159	179	188	159	126	134	150	199	174	119	100	41	41	58
	173	187	193	181	167	151	162	182	192	175	129	60	88	47	37	50
60	172	184	179	153	158	172	163	207	205	188	127	63	56	43	42	55
	156	191	196	159	167	195	178	203	214	201	143	101	69	38	44	52
	154	163	175	165	207	211	197	201	201	199	138	79	76	67	51	53
	144	150	143	162	215	212	211	209	197	198	133	71	69	77	63	53
	140	151	150	185	215	214	210	210	211	209	135	80	45	69	66	60
65	135	143	151	179	213	216	214	191	201	205	138	61	59	61	77	63

Source: Venus planet from Steven Smith, 1999

- Computers see images as a matrix of numbers (pixels) in the range of [0,255]. Each pixel represents a number.
- Image are often represented as 3 dimensional array of *height* $\times$ *width* $\times$ *channels*
  - Grayscale picture: $400 \times 600 \times 1$
  - Color picture: $400 \times 600 \times 3$, represents for 3 channels Red, Green and Blue (RGB)

- `keras.preprocessing.image` in Keras to convert image into array

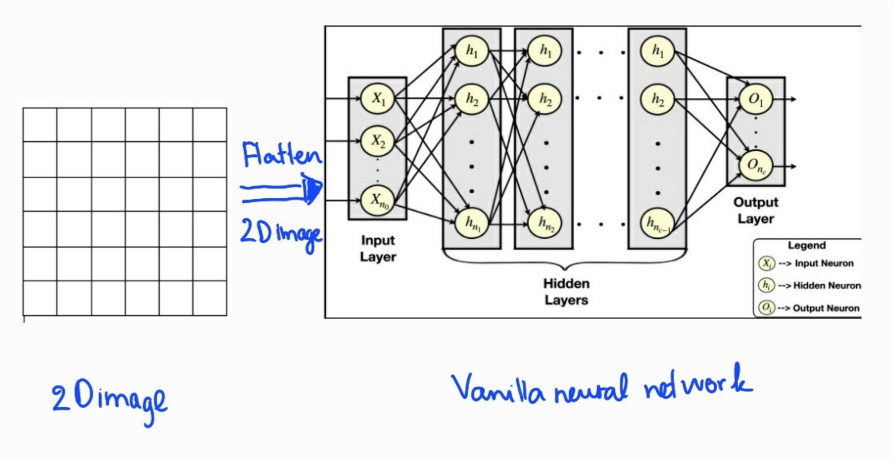
```
from keras.preprocessing.image import load_img
img = load_img('Basecamp.jpg')
print(type(img))
print(img.format)
print(img.size)

# convert to numpy array
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import array_to_img
# convert to numpy array
img_array = img_to_array(img)
print(img_array.dtype)
print(img_array.shape)
print(img_array)

# convert back to image

img = array_to_img(img_array)
#**************************************************
```

Can a vanilla neural network learn features of an image?



- Flat 2D image into 1 dimension inputs of pixel values
- All spatial information between pixels in 2D dimensional image is removed.
- **How can we preserve the spatial location?**

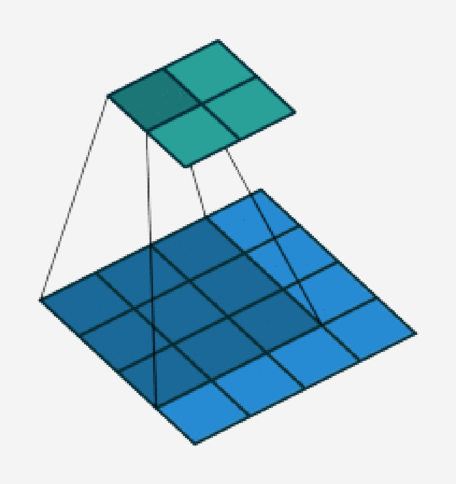
How a CNN extract features and do classification?

- Convolution layers
- Pooling layer
- Convolution neural network

Convolution layers

- Extract features from an image
- Preserving spatial information of an image

- Ideas: Given 2D dimensional image, each neuron in the convolution layer will connect to a patch of image using a kernel (filters). The whole spatial connection of the image is extracted by simply sliding the same filter across the image.



Link

- Use a kernel of 3×3: 9 weights which is used to learn the feature.
- Apply the same kernel across the image to extract a feature image of 2×2
- Multiple kernels are used to learn many features of image.
- The operation is known as convolution. The output of the convolution layer is known as **feature maps**

$$\begin{bmatrix} 3 & 3 & 2 & 1 \\ 0 & 0 & 1 & 3 \\ 3 & 1 & 2 & 2 \\ 2 & 0 & 0 & 2 \end{bmatrix} \odot \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 12 & 12 \\ 10 & 17 \end{bmatrix}$$

$$\begin{bmatrix} \boxed{\begin{matrix} 3 & 3 & 2 \\ 0 & 0 & 1 \\ 3 & 1 & 2 \end{matrix}} & \begin{matrix} 1 \\ 3 \\ 2 \end{matrix} \\ \begin{matrix} 2 & 0 & 0 \end{matrix} & 2 \end{bmatrix} \odot \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix} = \begin{aligned} &3\times 0 + 3\times 1 + 2\times 2 \\ &+ 0\times 2 + 0\times 2 + 1\times 0 \\ &+ 3\times 0 + 1\times 1 + 2\times 2 \end{aligned}$$

$$= 12$$

Manual filters

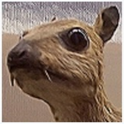
What are kernel types?

- This kernels are constructed manually to learn particular features such as sharpen, edge detection, and strong edge

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Source: Wikipedia

Manual Filterss

Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

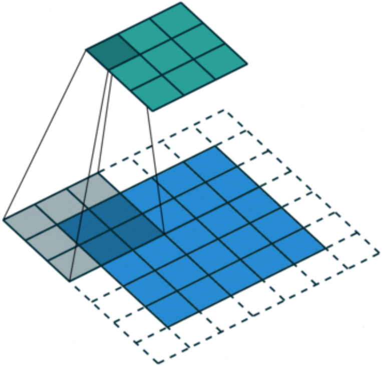
Source: Wikipedia

Kernels

- This kernels are constructed manually to learn particular features such as sharpen, edge detection, and strong edge
- Convolution neural network will estimate weights of the kernels, learn features of an image in the most optimal way.
- Keras syntax: `tf.keras.layers.Conv2D`

```
# Input is an image 28x28
input_shape = (1, 28, 28, 3)
x = tf.random.normal(input_shape)
tf.keras.layers.Conv2D(filters=32, kernel_size=(3,3), strides=(1,1), input_shape=input_shape[1:])(image)
```

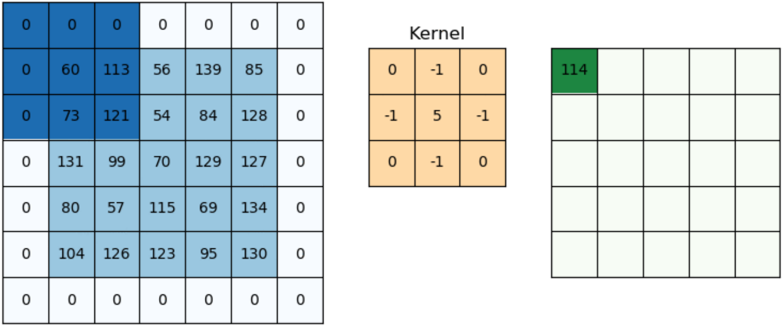
- Strike: The number of rows and columns traversed per slide.



Link

```
input_shape = (1, 28, 28, 3)
x = tf.random.normal(input_shape)
tf.keras.layers.Conv2D(filters=32, kernel_size=(3,3), strides=(2,2), input_shape=input_shape[1:])(image)
```

- Padding: to ensure that the size of kernel and the output the same or independently, and to extract the border of the input



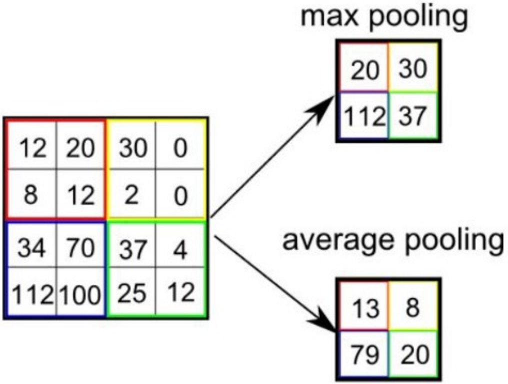
Source: Towardsdatascience

```
input_shape = (1, 28, 28, 3)
x = tf.random.normal(input_shape)
tf.keras.layers.Conv2D(filters=32, kernel_size=(3,3), strides=(2,2), padding ='same',
                       input_shape=input_shape[1:])(image)
```

-

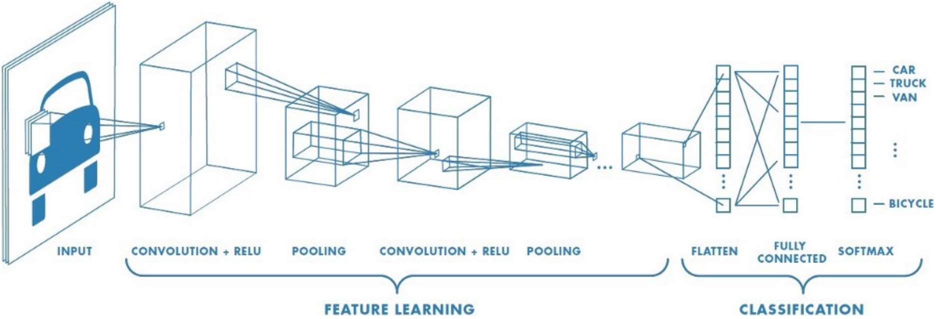
Pooling layer

- Down-sampling the number of feature maps to further reduce computational costs and memory space.
- Ensuring that the output feature maps are invariant to small translations of the input. Invariance to translation means that if the input is translated (e.g. changed in size, or rotated), the output feature maps do not change. This is not the case for the output feature maps obtained from the convolution layer
- 2 types of pooling: max pooling and average pooling



Source: TowardsDataScience

Convolution layers



Source: TowardsDataScience

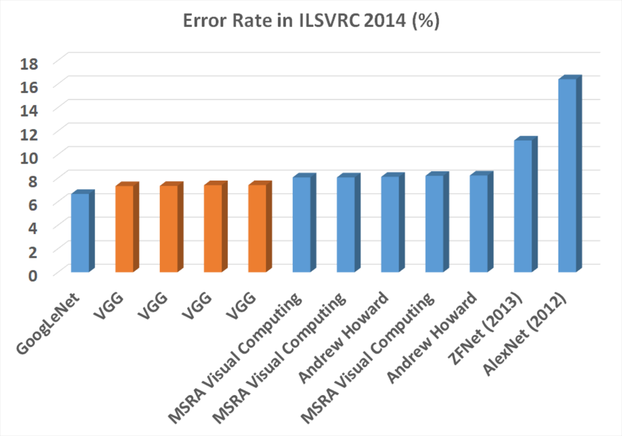
- Convolution can be classified into 2 stages. The first stage is to learn features, and the second stage is to do classification
- Convolution layer
- Pooling layer: Downsampling operation on each feature map.
    - We can use multiple layers to learn many features of an image. The output of the convolution layers will be a 3 dimensional array for gray scale picture (height, width and depth (number of kernels) and 4-dimensional array for color (height, width, channels and depth)
- Non-linearty: the feature extracted are highly non-linear

Advanced CNN

- AlexNet
- VGG
- GoogLeNet
- ResNet

ImageNet

- The ImageNet Large Scale Visual Recognition Challenge (ILSVRC), is an annual competition that uses subsets from ImageNetdateset, a large collection of human annotated photographs.



Source: Kaggle

VGG

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG

- Use of very small convolution filters, e.g. 3×3 and 1×1 with a stride of one.
- Use of max pooling with a size of 2×2 and a stride of the same dimensions.
- Stacking convolution layers together before using a pooling layer to define a block. The idea is that convolution layers with smaller filters approximate the effect of one convolution with a larger sized filter.
- The number of filters increases with the depth of the model, starting with 64, 128, 256 and 512 filters.
- Development of very deep (16 and 19 layer) models.

VGG on Keras

```
#*************************************************************
import tensorflow as tf
from keras.models import Model
from keras.layers import Input
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
# function for creating a vgg block
def vgg_block(layer_in, n_filters, n_conv):
    # add convolutional layers
    for _ in range(n_conv):
        layer_in = Conv2D(n_filters, (3,3), padding='same', activation='relu')(layer_in)
    # add max pooling layer
    layer_in = MaxPooling2D((2,2), strides=(2,2))(layer_in)
    return layer_in
# define model input
visible = Input(shape=(256, 256, 3))
# add vgg module
layer = vgg_block(visible, 64, 2)
# add vgg module
layer = vgg_block(layer, 128, 2)
# add vgg module
layer = vgg_block(layer, 256, 4)
# create model
model = Model(inputs=visible, outputs=layer)
# summarize model
model.summary()
```

040

Summary

- Convolution layers
- Pooling layer
- Convolution neural network

Lab session

- Implementation of CNN in AWS using script mode and built-in algorithm in SageMaker
- The technical collaborate session is on Monday (31/5) next week.