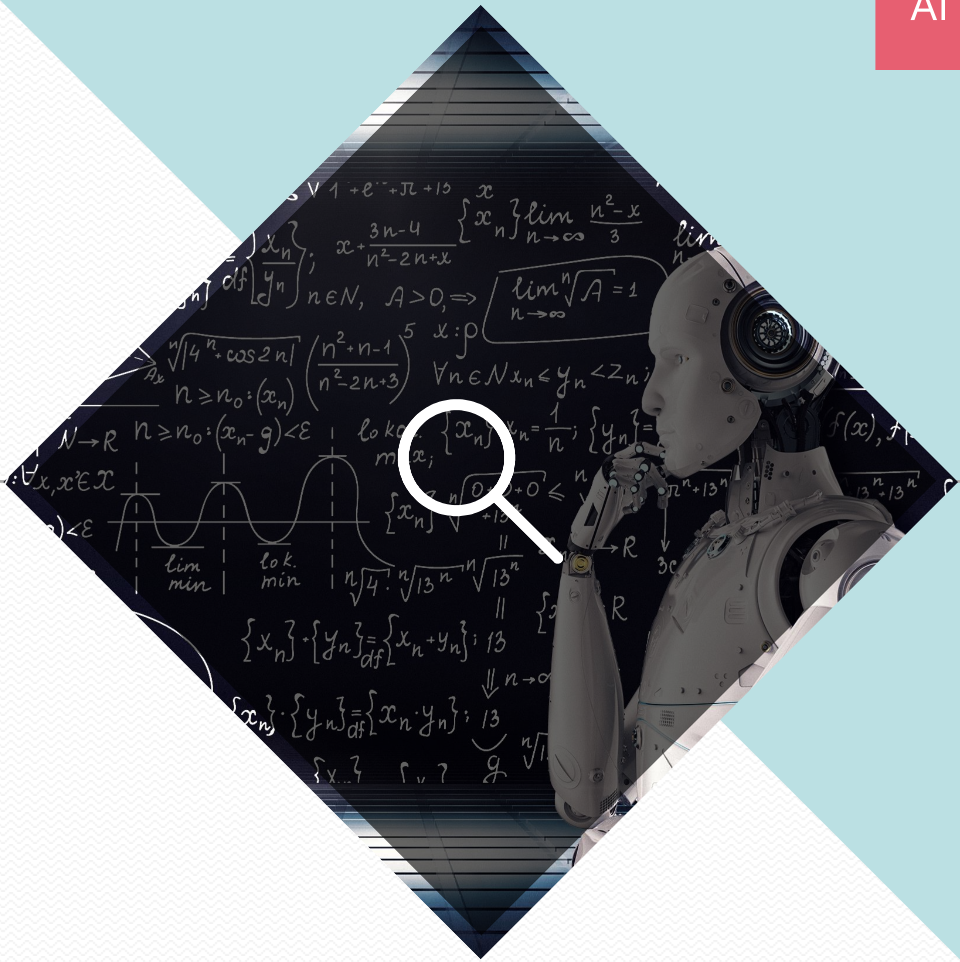


Classification and Detection hands on

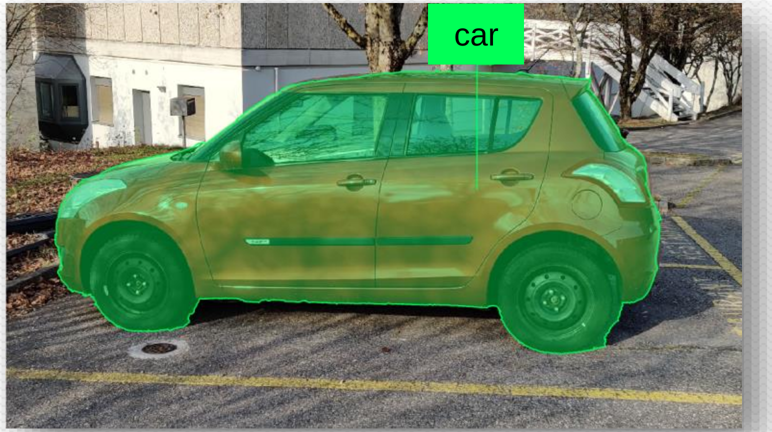
Before Start

WHAT YOU WILL LEARN FROM THIS SESSION

- Hands on practice of adopting CNN knowledge.
- Designing & building a model for image classification.
- Training & inferencing workflows in TensorFlow.
- Getting help on TensorFlow modules & functions.
- Tuning hyper-parameter.

AI Use Case

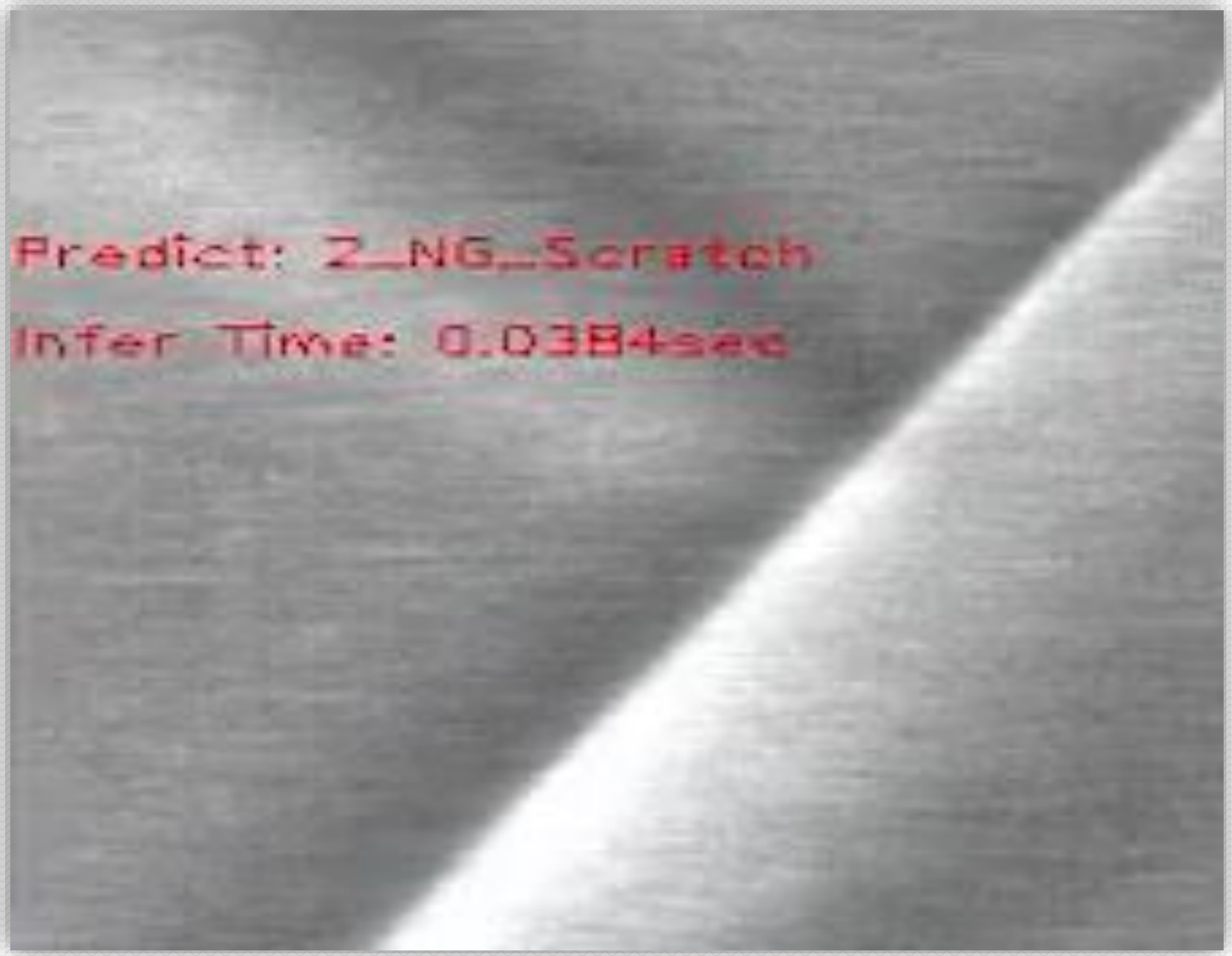
- Classification
  - Classify the whole that one or identify it in an image
- Detection
  - Track that object in a video or identify it in an image
- Segmentation
  - Highlight the isolated region and groups every pixel in that delineated shape for later processing



Classification

- Classify the whole that one or identify it in an image

Example



Classification Annotation

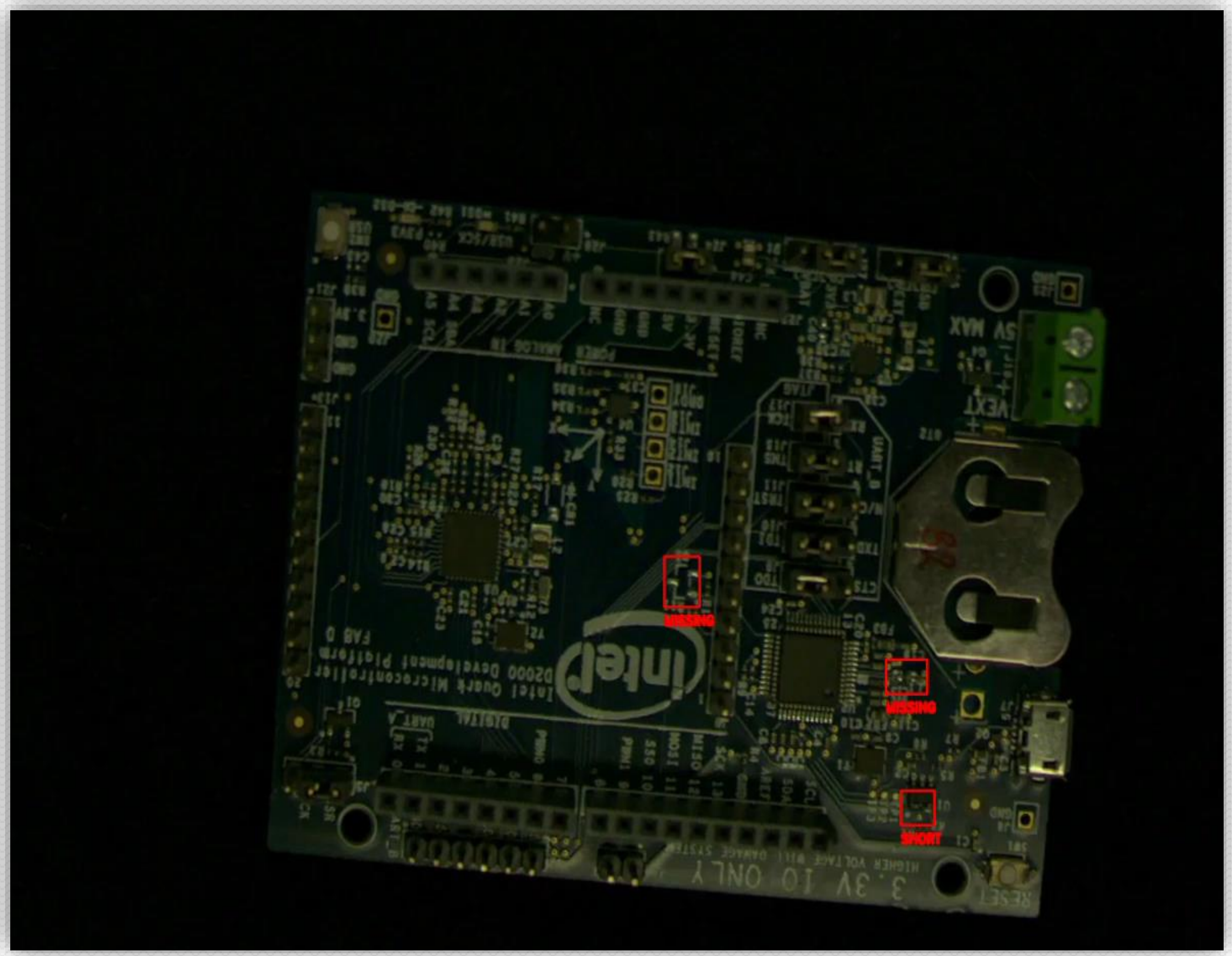
- Classification
  - Directory name is label name.

Car

Detection

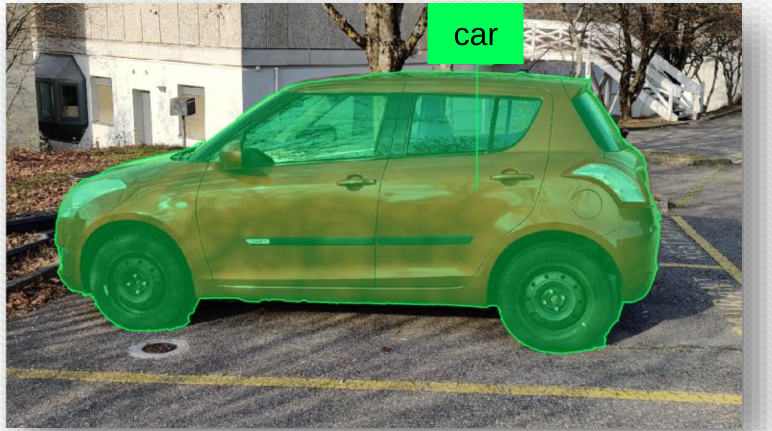
- Track that object in a video or identify it in an image

Example

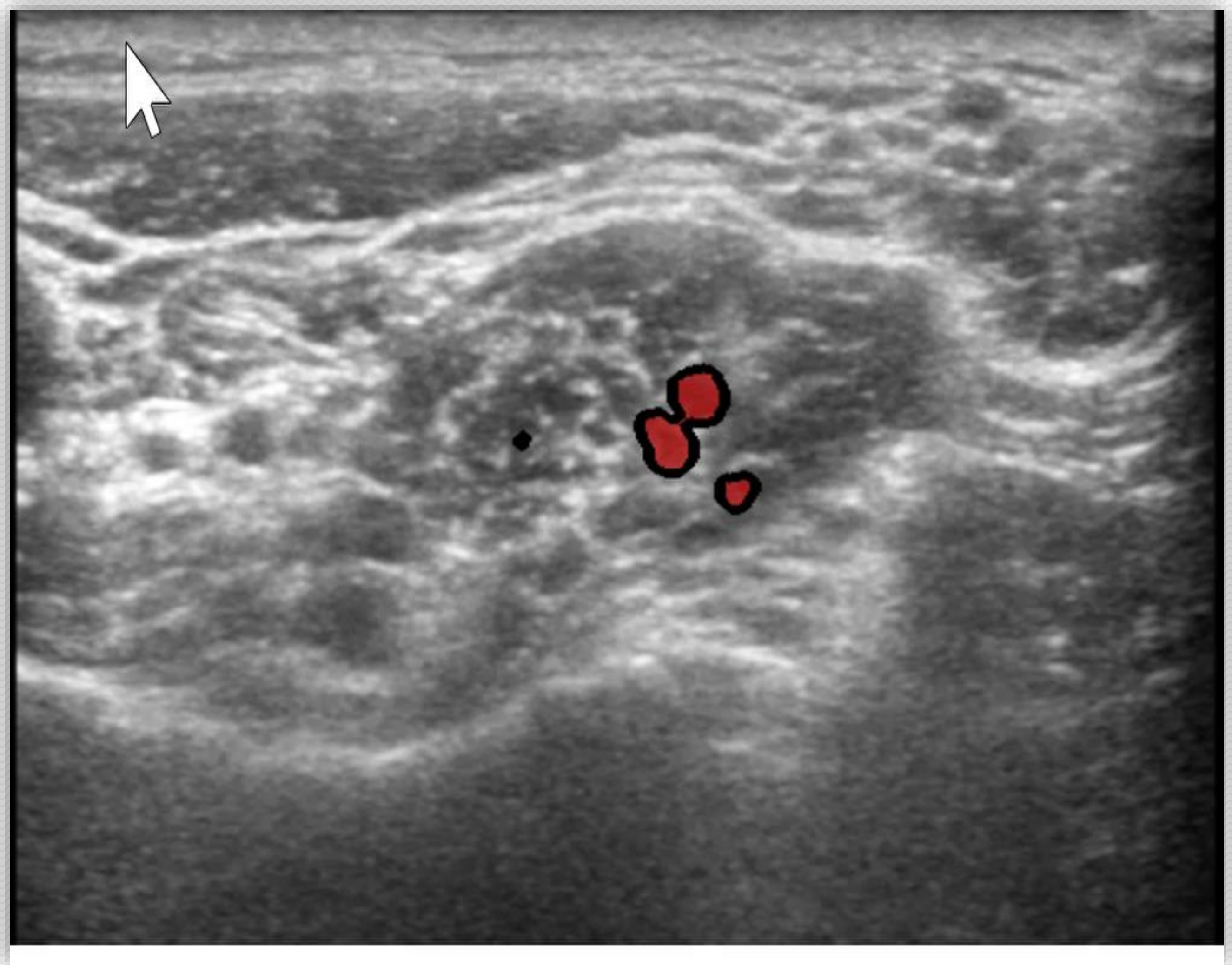


Segmentation

- Highlight the isolated region and groups every pixel in that delineated shape for later processing



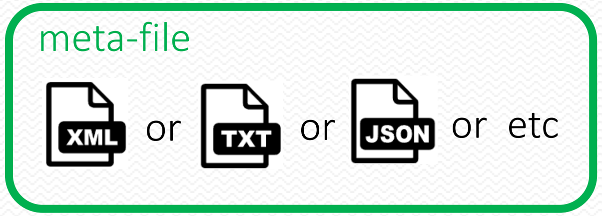
Example

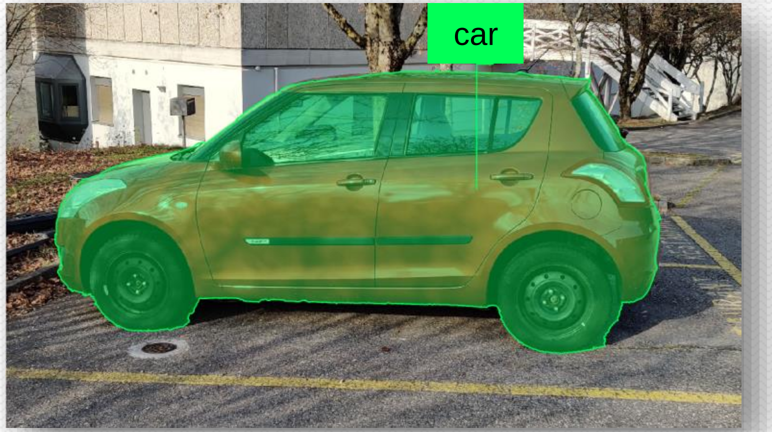


Detection / Segmentation Annotation

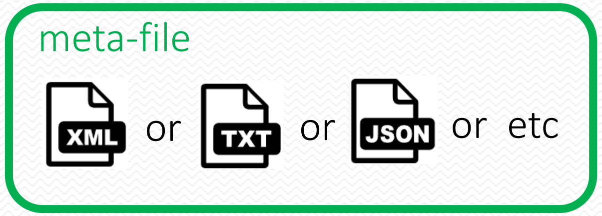
- Object Detection / Segmentation
  - Dedicated meta-file to contain label info

+





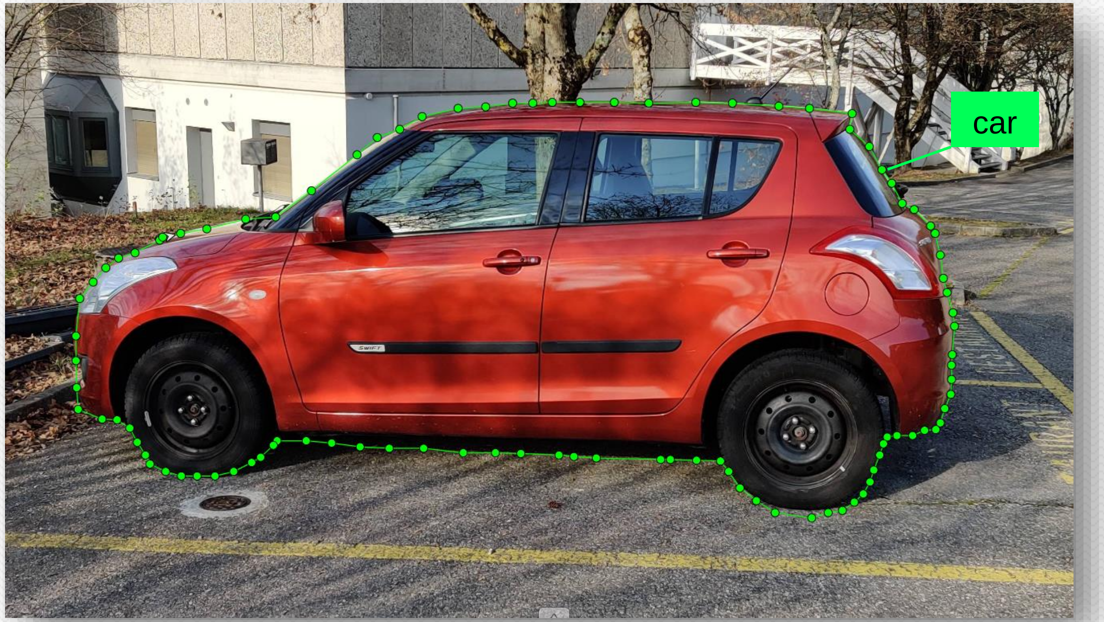
+



Detection / Segmentation Annotation

- Example
  - Segmentation
  - COCO format

IMG_20201123_132631.png



instances.json

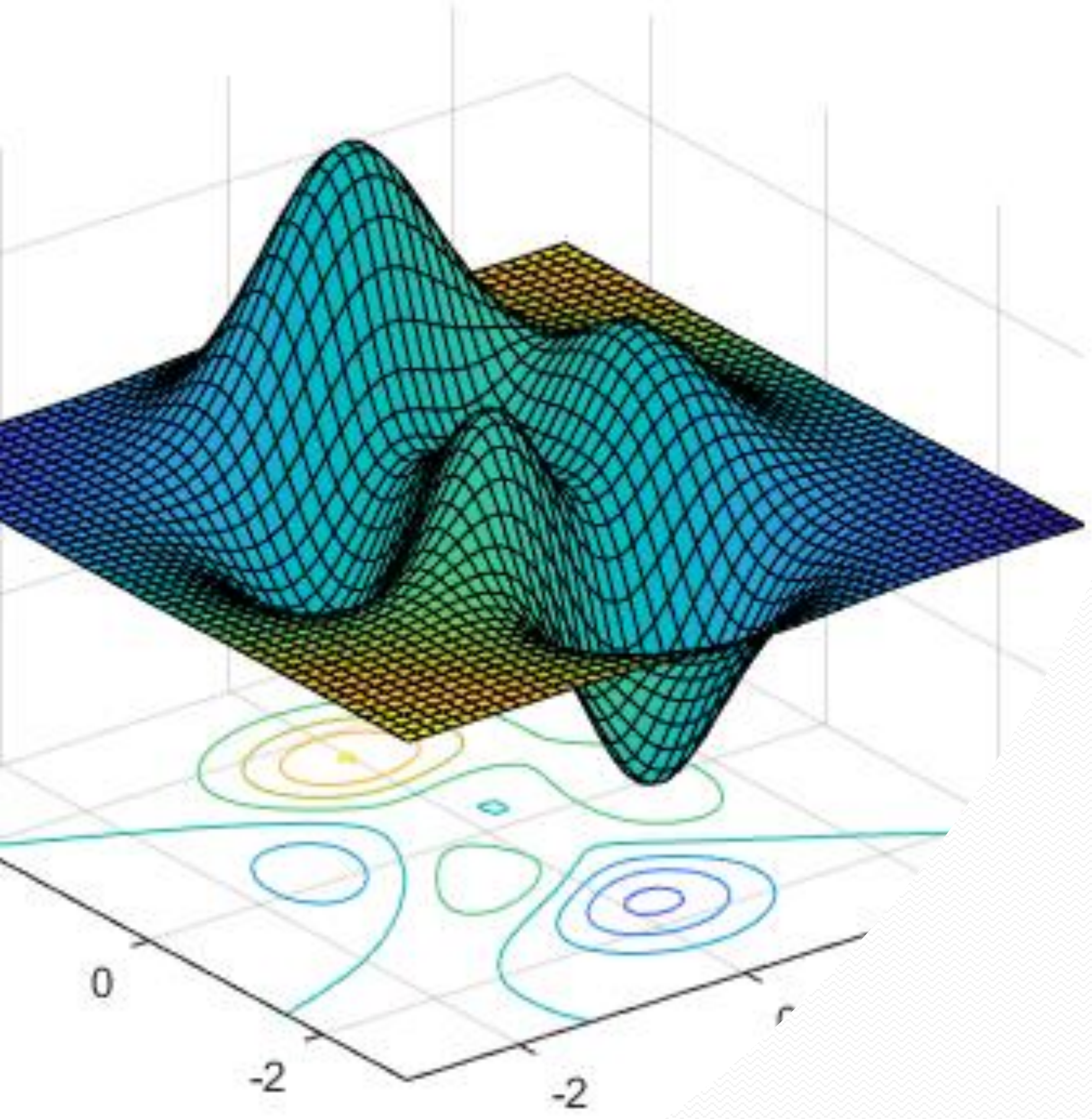

```
{
 ...(Snipped)...
 "categories": [
   {
     "id": 1,
     "name": "human",
     "supercategory": ""
   },
   {
     "id": 2,
     "name": "car",
     "supercategory": ""
   },
 ],
 "images": [
   {
     "id": 1,
     "width": 2666,
     "height": 1540,
     "file_name": "IMG_20201123_132631.png",
     "license": 0,
     "flickr_url": "",
     "coco_url": "",
     "date_captured": 0
   }
 ],
 "annotations": [{
   "segmentation": [[395.29, 591.17, 447.18, 578.20, 512.04, 567.08, 608.40, 542.99, 682.53, 537.43, 771.48,
   472.57, 884.53, 383.61, 934.57, 340.99, 990.16, 320.60, 1045.76, 289.10, 1134.71, 268.71, 1203.28,
   265.01, 1269.99, 255.74, 1320.03, 255.74, 1368.21, 253.89, 1436.78, 253.89, 1522.03, 253.89, 1607.27,
   255.74, 1724.02, 253.89, 1814.83, 255.74, 1929.73, 263.16, 2005.71, 268.71, 2111.34, 281.69, 2105.78,
   320.60, 2153.96, 363.23, 2185.47, 420.68, 2209.56, 454.03, 2235.50, 504.07, 2265.16, 518.90, 2305.93,
   559.67, 2317.04, 578.20, 2330.02, 631.94, 2337.43, 689.39, 2346.70, 724.60, 2361.52, 774.64, 2365.23,
   807.99, 2359.67, 878.41, 2357.81, 904.36, 2357.81, 939.57, 2355.96, 976.63, 2341.14, 1011.84, 2330.02,
   1047.05, 2318.90, 1063.73, 2291.10, 1065.59, 2261.45, 1078.56, 2222.53, 1078.56, 2198.44,
   ...(Snipped)..., 1091.53, 686.24, 1091.53, 676.97, 1102.65, 645.47, 1132.30, 623.23, 1145.27, 578.75,
   1167.51, 532.42, 1178.63, 487.95, 1178.63, 449.03, 1178.63, 410.11, 1167.51, 371.20, 1148.98, 360.08,
   1126.74, 339.69, 1095.24, 321.16, 1060.03, 289.66, 1039.64, 252.59, 1037.79, 193.29, 1013.70, 189.58,
   959.95, 187.73, 893.24, 187.73, 832.08, 197.00, 739.42, 230.35, 698.65, 261.86, 663.44, 293.36, 639.35,
   334.13, 626.38, 395.29, 594.88, 402.70, 587.46]],
   "area": 600.4,
   "iscrowd": 1,
   "Image_id:" 2,
   "bbox": [473.05, 395.45, 38.65, 28.92],
   "category_id": 15,
   "id": 934}]
}
```


Label name

Image information

All of points info to close contour for object

* COCO (Common Objects in Context) : https://cocodataset.org/



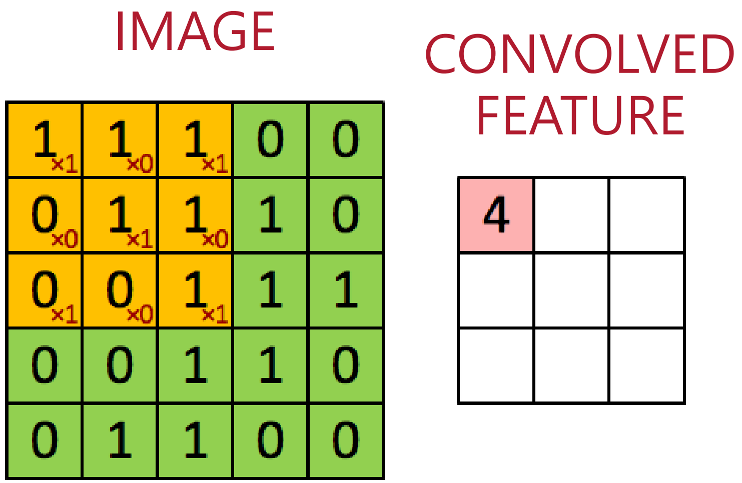
CONVOLUTION

CONVOLUTION

MATHEMATICAL OPERATION

Slides one function over another

e.g., Photo filters

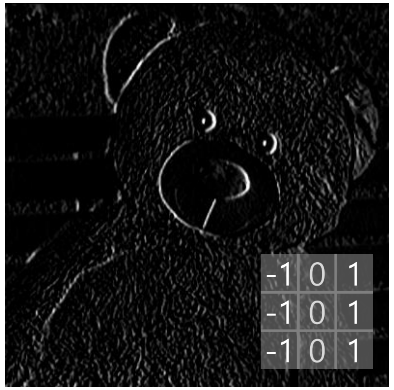


ORIGINAL

SHARPEN

EMBOSS

EXTRACT HORIZONTAL LINE



EXTRACT VERTICAL LINE

CONVOLUTION

MATHEMATICAL OPERATION

1차 미분 연산자이용한 Edge detect

IMAGE

$1_{\times1}$	$1_{\times0}$	$1_{\times1}$	0	0
$0_{\times0}$	$1_{\times1}$	$1_{\times0}$	1	0
$0_{\times1}$	$0_{\times0}$	$1_{\times1}$	1	1
0	0	1	1	0
0	1	1	0	0

CONVOLVED FEATURE

4		

Original

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

Edge Detection

CONVOLUTION

```python
import cv2
import numpy as np

img = cv2.imread('lena.png', cv2.IMREAD_GRAYSCALE)
kernel = np.array([[1, 1, 1],[1, -8, 1], [1, 1, 1]])
print(kernel)
output = cv2.filter2D(img, -1, kernel)
cv2.imshow('edge', output)
cv2.waitKey(0)
```

Homework #1

- 아래 나열된 필터를 적용해 보고, 해당 코드를 github에 upload
- https://en.wikipedia.org/wiki/Kernel_(image_processing)

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

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)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

VIRTUAL ENVIRONMENT

- Python3 virtual environment.
  - `$mkdir intro && cd intro`
  - `$sudo apt install python3-venv`
  - `$python3 -m venv .venv`
  - `$source .venv\bin\activate`

- Download & install python modules.
  - `(.venv)$ pip install tensorflow`
  - `(.venv)$ pip install numpy`
  - `(.venv)$ pip install matplotlib`
  - `(.venv)$ pip install pyQt5==5.14`

CONTENTS

IMAGE CLASSIFICATION TRAINING WORKFLOW

- 데이터셋 준비
  - 학습용 / 검증용
- 학습 모델 형성
  - Convolution layer
  - Pooling layer
  - Dense(FC) layer
- 컴파일
- 학습 및 테스트

train.py

```
# Prepare datasets.
# Construct a model.
# Compile the model.
# Train & evaluate
# Save the model.
```

VIRTUAL ENVIRONMENT

- Python3 virtual environment.
  - `$mkdir intro && cd intro`
  - `$sudo apt install python3-venv`
  - `$python3 -m venv .venv`
  - `$source .venv\bin\activate`

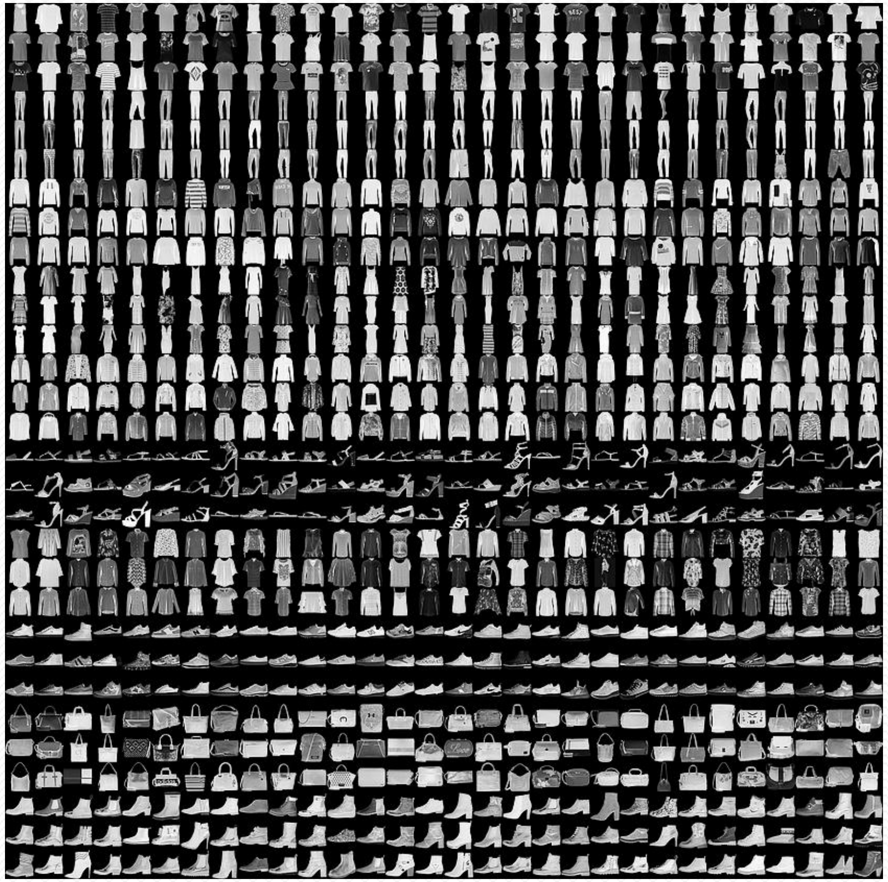
- Download & install python modules.
  - `(.venv)$ pip install tensorflow`
  - `(.venv)$ pip install numpy`
  - `(.venv)$ pip install matplotlib`
  - `(.venv)$ pip install pyQt5==5.14`

INITIAL SETTING & IMPORT MODULES

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
```

DATASET PREPARATION (Hand and Fashion)

```
# Model load: MNIST / Fashion MNIST Dataset
mnist = tf.keras.datasets.mnist
fashion_mnist = tf.keras.datasets.fashion_mnist
```



DATASET PREPARATION

mnist 혹은 fashion_mnist 데이터셋을 준비하고, training할 이미지셋과 test할 이미지 셋을 구분해준다.
그 후 이미지의 색상을 정규화를 시켜주기 위해 255로 색상 값을 나눈다.

```
# Model load: MNIST / Fashion MNIST Dataset
fashion_mnist = tf.keras.datasets.fashion_mnist
(f_image_train, f_label_train), (f_image_test, f_label_test) = fashion_mnist.load_data()
# normalized iamges
f_image_train, f_image_test = f_image_train / 255.0, f_image_test / 255.0
```

DATASET PREPARATION (cont’d)

fashion_mnist 의 레이블은 숫자로 저장이 되어 있기 때문에 레이블과 클래스 이름을 매핑을 해줘야 한다.

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

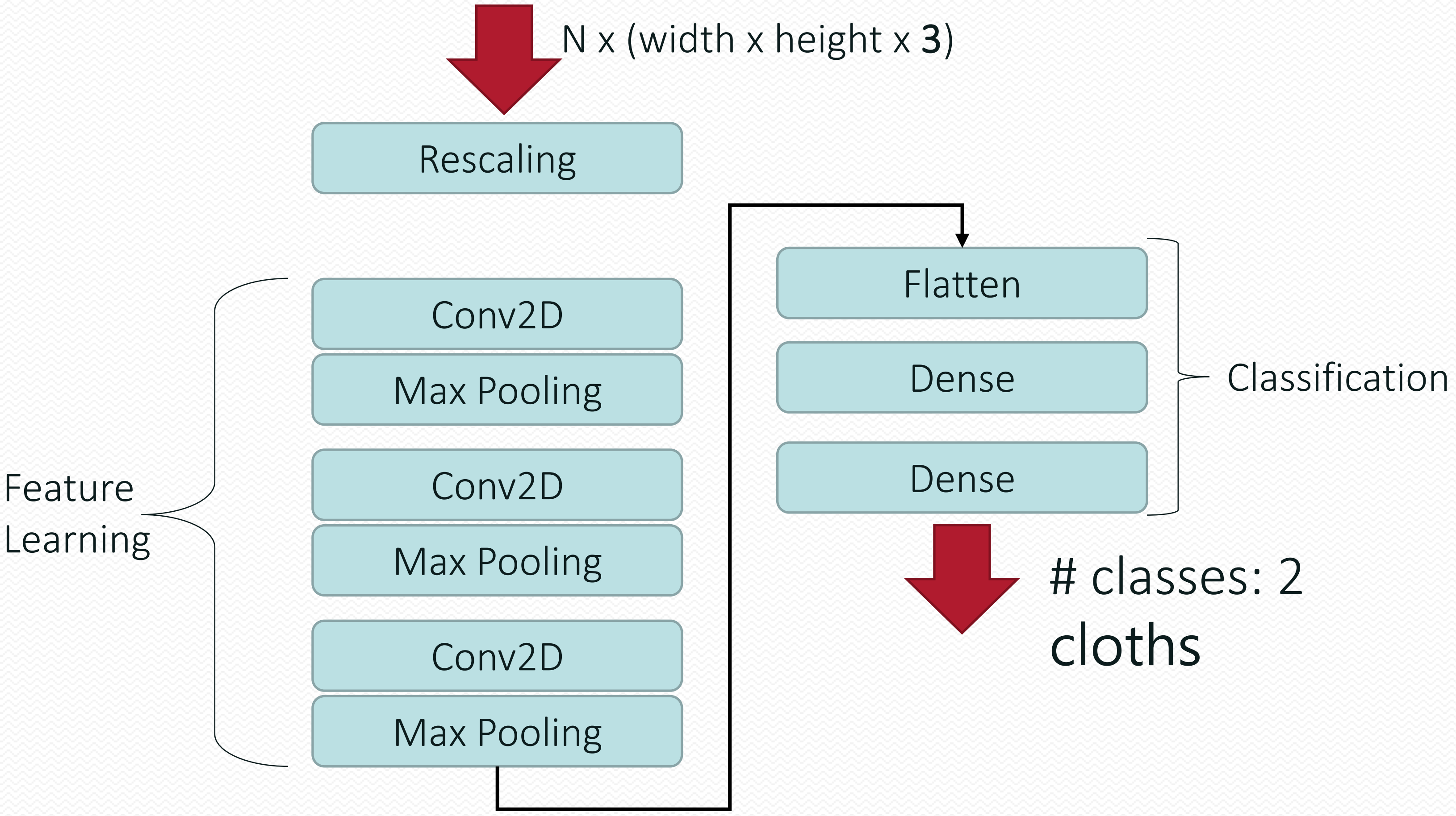
레이블	클래스
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

DATASET PREPARATION (show)

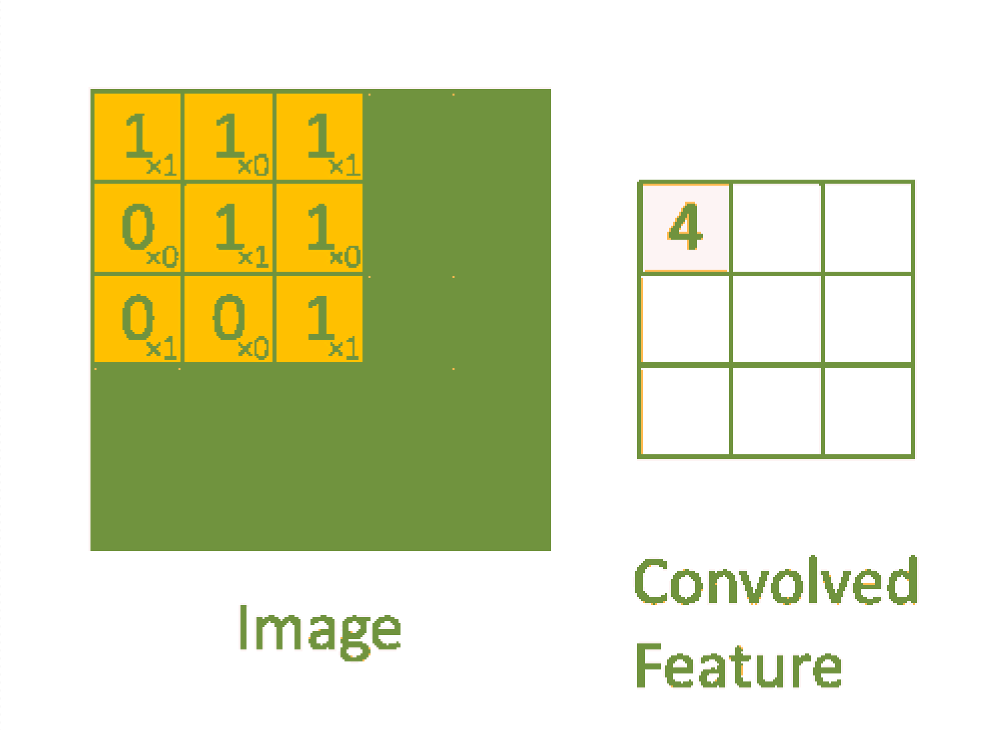
```
plt.figure(figsize=(10,10))
for i in range(10):
    plt.subplot(3,4,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(f_image_train[i])
    plt.xlabel(class_names[f_label_train[i]])
plt.show()
```

MODEL CONSTRUCTION

- [tf.keras.Sequential()]
- [tf.keras.layers]
  - Rescaling
  - Convolution
  - Pooling
  - Flatten
  - Dense(ANN/FC)



CONVOLUTION LAYER

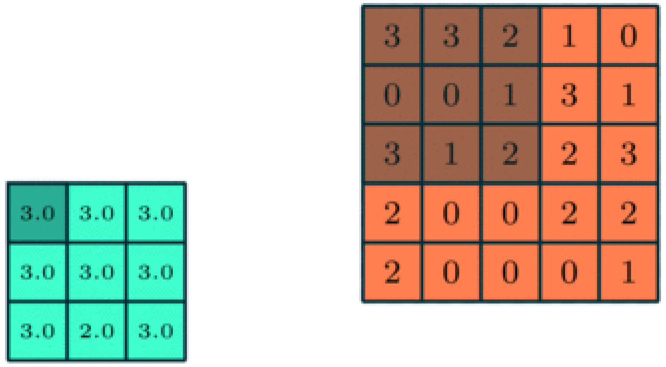


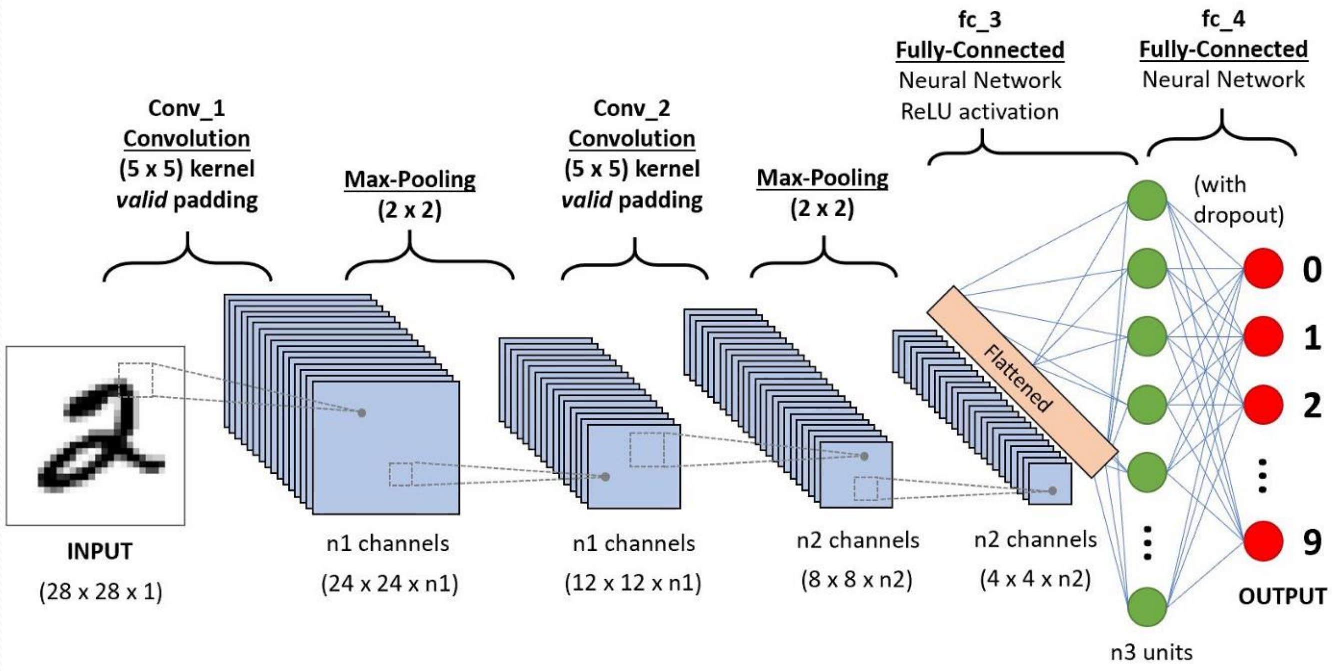
- Image: 5x5, 1 channel
- Kernel: 3x3
- Stride: 1
- Padding: 0

*Animations from [A Comprehensive Guide to Convolutional Neural Networks]()

POOLING LAYER

- Feature map: 5x5, 1 channel
- Kernel: 3x3
- Stride: 1
- Padding: 0

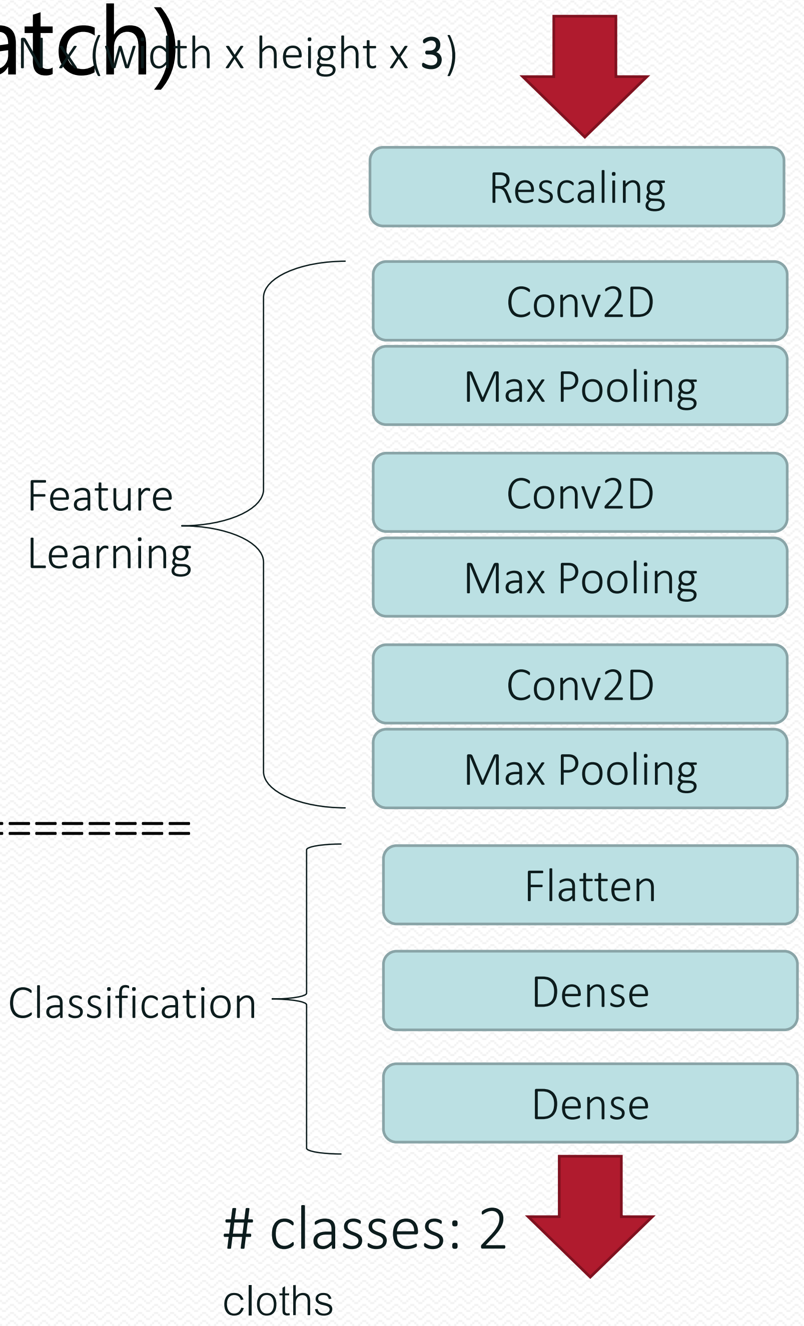




- Max pooling
- Average pooling

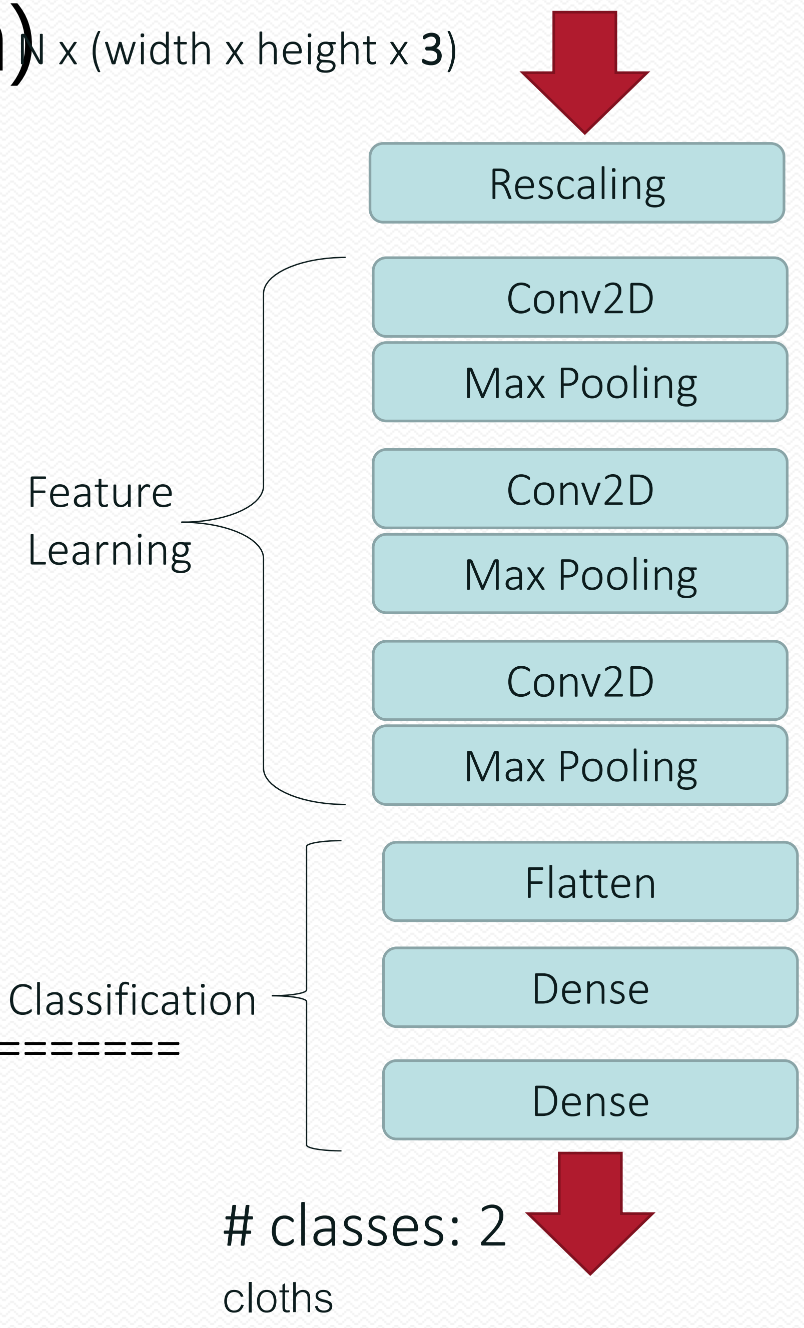
MODEL CONSTRUCTION (without batch)

```python
# CNN
model = Sequential()
model.add(Conv2D(32,(2,2),activation="sigmoid",
input_shape=(28,28,1)))
model.add(Conv2D(64,(2,2),activation="sigmoid"))
model.add(Conv2D(128,(2,2),2,activation="sigmoid"))
model.add(Conv2D(32,(2,2),activation="sigmoid"))
model.add(Conv2D(64,(2,2),activation="sigmoid"))
model.add(Conv2D(128,(2,2),2,activation="sigmoid"))
=========================================================
# ANN
model.add(Flatten())
model.add(Dense(128,activation="sigmoid"))
model.add(Dense(10,activation="softmax"))
```



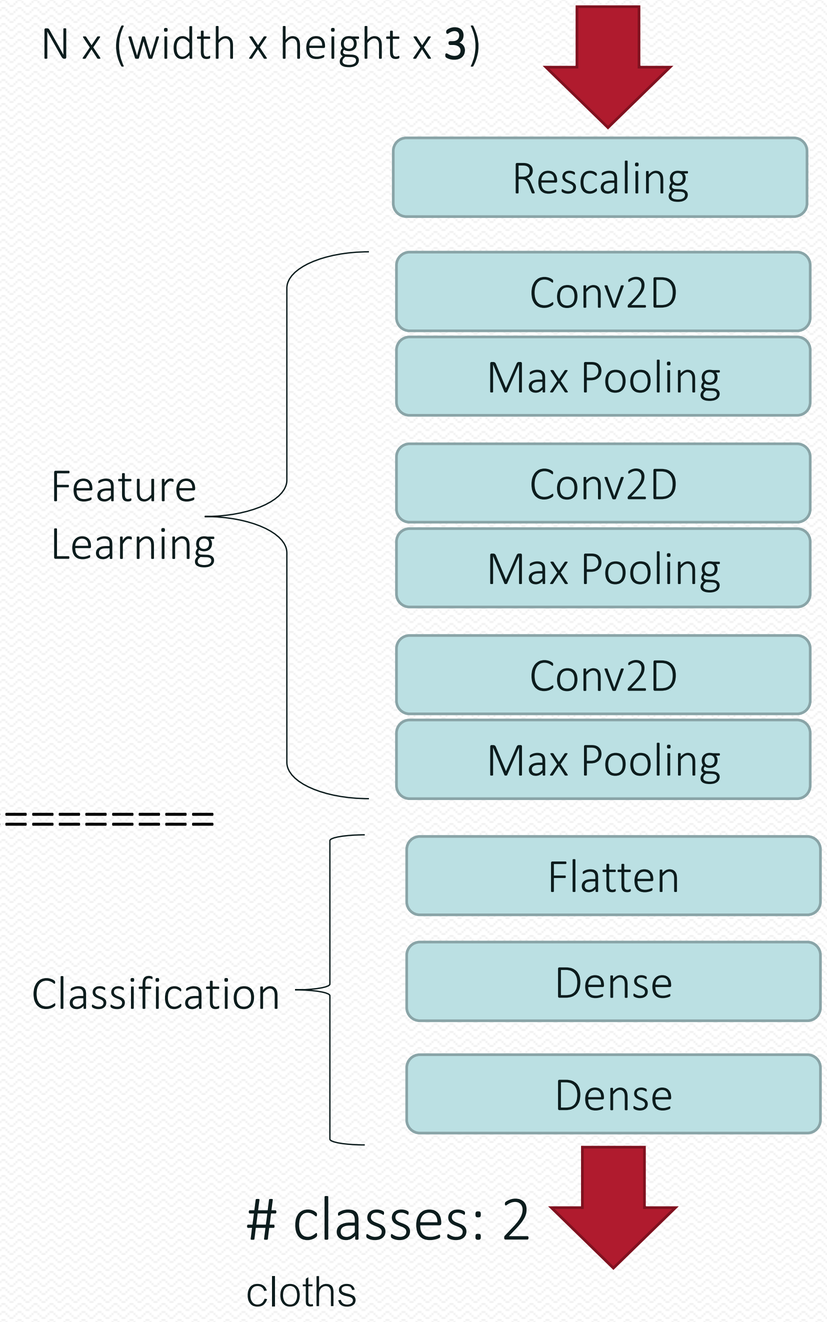
MODEL CONSTRUCTION (with batch)

```
# CNN
model = Sequential()
model.add(Conv2D(32,(2,2),activation="sigmoid",
input_shape=(28,28,1)))
model.add(BatchNormalization())
model.add(Conv2D(64,(2,2),activation="sigmoid"))
model.add(BatchNormalization())
model.add(Conv2D(128,(2,2),2,activation="sigmoid"))
model.add(BatchNormalization())
model.add(Conv2D(32,(2,2),activation="sigmoid"))
model.add(BatchNormalization())
model.add(Conv2D(64,(2,2),activation="sigmoid"))
model.add(BatchNormalization())
model.add(Conv2D(128,(2,2),2,activation="sigmoid"))
model.add(BatchNormalization())
=================================================================
# ANN
model.add(Flatten())
model.add(Dense(128,activation="sigmoid"))
model.add(Dense(10,activation="softmax"))
```



MODEL CONSTRUCTION (relu)

```
# CNN
model = Sequential()
model.add(Conv2D(32,(2,2),activation="relu",
input_shape=(28,28,1)))
model.add(Conv2D(64,(2,2),activation="relu"))
model.add(Conv2D(128,(2,2),2,activation="relu"))
model.add(Conv2D(32,(2,2),activation="relu"))
model.add(Conv2D(64,(2,2),activation="relu"))
model.add(Conv2D(128,(2,2),2,activation="relu"))
==========================================================
# ANN
model.add(Flatten())
model.add(Dense(128,activation="sigmoid"))
model.add(Dense(10,activation="softmax"))
```



MODEL COMPILATION

- tf.keras.Sequential.compile()
  - optimizer
  - loss
  - Metrics
- tf.model.fit()
  - validation_data
  - epochs
  - Hyper parameters
    - Epochs
    - Dataset size
    - Batch size
    - Optimizer / loss function
- Save model
  - tf.keras.Model.save()

```
model.compile(
        loss = "categorical_crossentropy",
        optimizer = "adam",
        metrics=["accuracy"])
history = model.fit(image_train, label_train, epochs=10, batch_size=10)
model.summary()
model.save('fashion_mnist.h5')
with open('historyBatchReLu', 'wb') as file_pi:
    pickle.dump(history.history, file_pi)
```

MODEL TRAINING

```
Epoch 1/10
250/250 [==============================] - 210s 840ms/step - loss: 0.6527 -
accuracy: 0.6428 - val_loss: 0.5764 - val_accuracy: 0.6965
Epoch 2/10
250/250 [==============================] - 222s 888ms/step - loss: 0.5223 -
accuracy: 0.7293 - val_loss: 0.5381 - val_accuracy: 0.7275
Epoch 3/10
250/250 [==============================] - 225s 898ms/step - loss: 0.4456 -
accuracy: 0.7894 - val_loss: 0.5187 - val_accuracy: 0.7500
Epoch 4/10
250/250 [==============================] - 224s 895ms/step - loss: 0.3441 -
accuracy: 0.8429 - val_loss: 0.5366 - val_accuracy: 0.7545
Epoch 5/10
250/250 [==============================] - 225s 899ms/step - loss: 0.2362 -
accuracy: 0.9046 - val_loss: 0.6528 - val_accuracy: 0.7575
Epoch 6/10
250/250 [==============================] - 238s 953ms/step - loss: 0.1323 -
accuracy: 0.9465 - val_loss: 0.8491 - val_accuracy: 0.7460
Epoch 7/10
250/250 [==============================] - 232s 927ms/step - loss: 0.0696 -
accuracy: 0.9734 - val_loss: 1.0140 - val_accuracy: 0.7570
Epoch 8/10
250/250 [==============================] - 238s 952ms/step - loss: 0.0591 -
accuracy: 0.9801 - val_loss: 1.0195 - val_accuracy: 0.7415
Epoch 9/10
250/250 [==============================] - 242s 967ms/step - loss: 0.0339 -
accuracy: 0.9893 - val_loss: 1.2181 - val_accuracy: 0.7460
Epoch 10/10
250/250 [==============================] - 248s 991ms/step - loss: 0.0341 -
accuracy: 0.9875 - val_loss: 1.3117 - val_accuracy: 0.7545
```

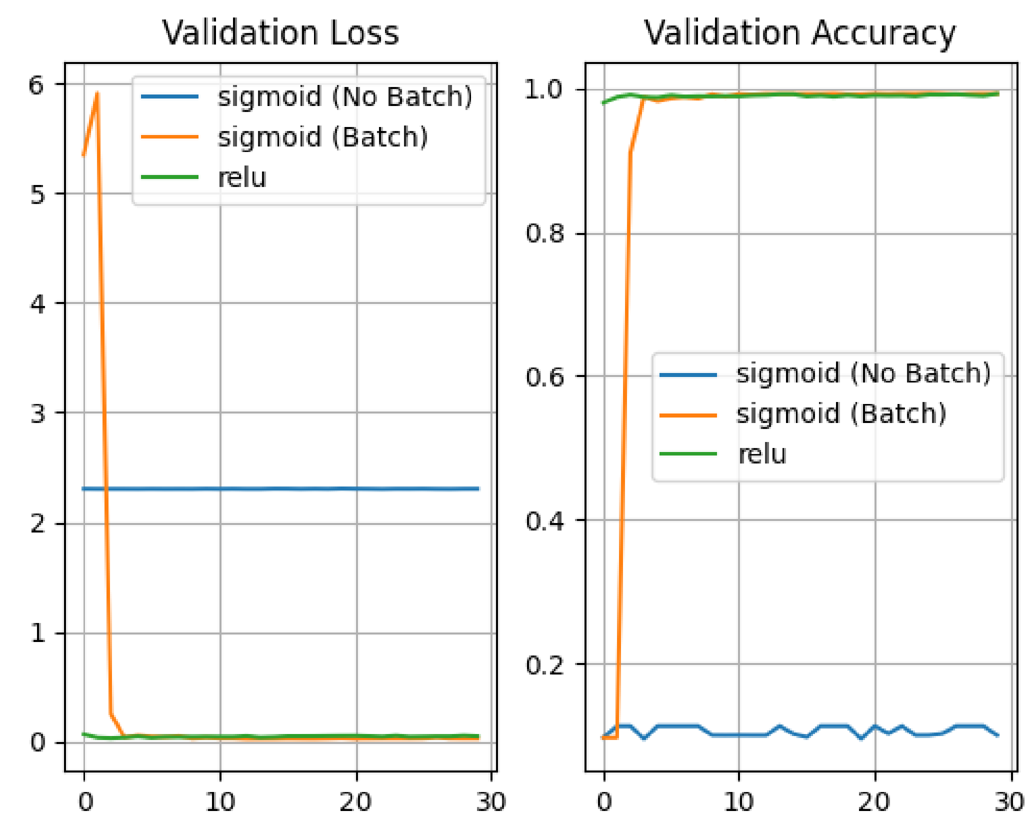
INFERENCE TRAINED MODELS

- Load model
  - tf.keras.models.load_model()

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import cv2

model = tf.keras.models.load_model(‘./fashion_mnist.h5')
fashion_mnist = tf.keras.datasets.fashion_mnist
(f_image_train, f_label_train), (f_image_test, f_label_test) =
fashion_mnist.load_data()

f_image_train, f_image_test = f_image_train / 255.0, f_image_test / 255.0

num = 10
predict = model.predict(f_image_train[:num])
print(f_label_train[:num])
print(" * Prediction, ", np.argmax(predict, axis = 1))
```

Homework #2

- No Batch, Batch, relu를 사용했을 때 성능을 비교.



SAVE MODEL SPECIFICATION (1)

```
import pickle
import numpy as np
import matplotlib.pyplot as plt

historyNoBatch = pickle.load(open('./historyNoBatch', "rb"))
historyBatch = pickle.load(open('./historyBatch', "rb"))
historylelu = pickle.load(open('./historyBatchReLu', "rb"))

val_accNB = historyNoBatch["val_accuracy"]
val_lossNB= historyNoBatch["val_loss"]
val_lossB = historyBatch["val_loss"]
val_accB = historyBatch["val_accuracy"]
val_lossL = historylelu["val_loss"]
val_accL = historylelu["val_accuracy"]
```

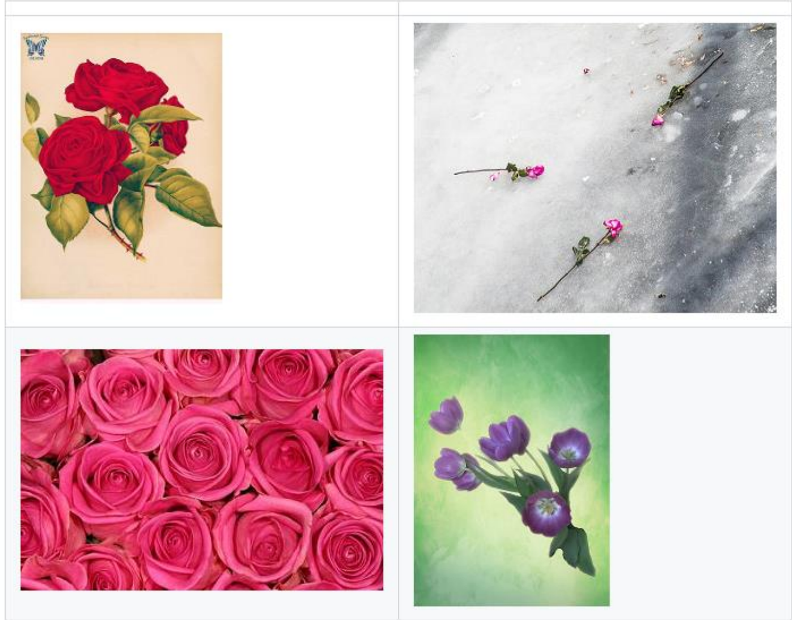
SAVE MODEL SPECIFICATION (2)

```
plt.subplot(1,2,1)
plt.title('Validation Loss')
plt.plot(range(len(val_lossNB)),val_lossNB,label = "sigmoid (No Batch)")
plt.plot(range(len(val_lossB)),val_lossB,label = "sigmoid (Batch)")
plt.plot(range(len(val_lossL)),val_lossL,label = "relu")
plt.grid()
plt.legend()
plt.subplot(1,2,2)
plt.title('Validation Accuracy')
plt.plot(range(len(val_accNB)),val_accNB,label = "sigmoid (No Batch)")
plt.plot(range(len(val_accB)),val_accB,label = "sigmoid (Batch)")
plt.plot(range(len(val_accL)),val_accL,label = "relu")
plt.grid()
plt.legend()
plt.show()
plt.savefig("Summary.png")
```

cnn overview (openvino notebooks ex)flower classification)

[openvino_notebooks/notebooks/301-tensorflow-training-openvino at 2024.0 · openvinotoolkit/openvino_notebooks (github.com)]

Training to Deployment with TensorFlow and OpenVINO™



In this directory, you will find two Jupyter notebooks. The first is an end-to-end deep learning training tutorial which borrows the open source code from the TensorFlow image classification tutorial, demonstrating how to train the model and then convert to OpenVINO Intermediate Representation (OpenVINO IR). It leverages the tf_flowers dataset which includes about 3,700 photos of flowers.

The second notebook demonstrates how to quantize the OpenVINO IR model that was created in the first notebook. Post-training quantization speeds up inference on the trained model. The quantization is performed with the Post-training Quantization with NNCF from OpenVINO Toolkit. A custom dataloader and a metric will be defined, and accuracy and performance will be computed for the original OpenVINO IR model and the quantized model on CPU and iGPU (if available).

Model training with OTX

Build
- Chooses models with custom dataset

Train
- Trains a model on a dataset
- HPO is optional

Eval
- Runs evaluation of trained model on a particular dataset

Export
- Exports a trained pytorch model to ONNX or OpenVINO

Optimize
- Optimizes the model with NNCF or POT

Deploy
- Deploy model with sample visualizer code

Disable Nouveau & Install nVidia GPU driver

```
$ lsmod|grep nouveau
nouveau              2285568  20
mxm_wmi                16384  1 nouveau
drm_ttm_helper         16384  1 nouveau
ttm                    86016  2 drm_ttm_helper,nouveau
drm_kms_helper        307200  1 nouveau
i2c_algo_bit           16384  1 nouveau
drm                   618496  11
drm_kms_helper,drm_ttm_helper,ttm,nouveau
video                  61440  1 nouveau
wmi                    32768  3 wmi_bmof,mxm_wmi,nouveau
```

```
sudo su
echo -e "\n\nblacklist nouveau" >> /etc/modprobe.d/blacklist.conf

echo "options nouveau modeset=0" > /etc/modprobe.d/nouveau-kms.conf
update-initramfs -u

reboot

sudo ubuntu-drivers autoinstall
```

Verify nVidia GPU driver installation

```
$ glxinfo|grep -i "opengl renderer"
OpenGL renderer string: NVIDIA GeForce GTX 1660/PCIe/SSE2
```

```
$ nvidia-smi
Sat Aug  5 12:07:21 2023
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.86.05              Driver Version: 535.86.05    CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA GeForce GTX 1660        Off | 00000000:01:00.0  On |                  N/A |
|  0%   54C    P0              50W / 130W |   3403MiB /  6144MiB |     98%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|    0   N/A  N/A      1191      G   /usr/lib/xorg/Xorg                          428MiB |
|    0   N/A  N/A      1524      G   /usr/bin/gnome-shell                         88MiB |
|    0   N/A  N/A      2983      G   ...irefox/2952/usr/lib/firefox/firefox      173MiB |
|    0   N/A  N/A      5327      G   ...sion,SpareRendererForSitePerProcess      103MiB |
|    0   N/A  N/A     19790      C   ...otx-classification/.otx/bin/python3     2604MiB |
+---------------------------------------------------------------------------------------+
```

Install CUDA 11.7

```
wget
https://developer.download.nvidia.com/compute/cuda/11.7.0/local_installers/cuda_11.7.0_515.43.04_linux.run

sudo sh cuda_11.7.0_515.43.04_linux.run
```

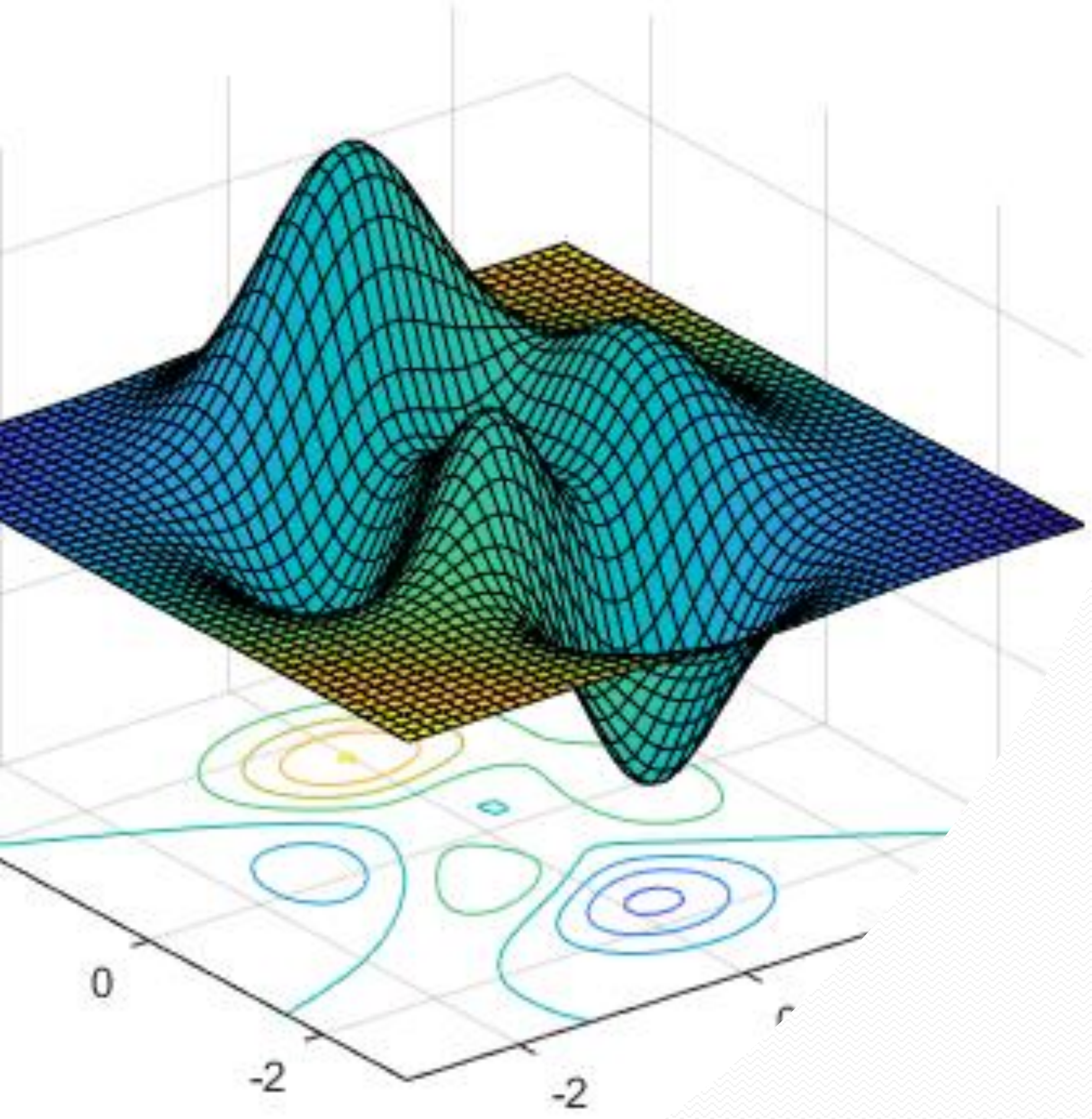
```
Existing package manager installation of the driver found. It is strongly
recommended that you remove this before continuing.
Abort
Continue
```

```
End User License Agreement
--------------------------

NVIDIA Software License Agreement and CUDA Supplement to
Software License Agreement. Last updated: October 8, 2021

The CUDA Toolkit End User License Agreement applies to the
NVIDIA CUDA Toolkit, the NVIDIA CUDA Samples, the NVIDIA
Display Driver, NVIDIA Nsight tools (Visual Studio Edition),
and the associated documentation on CUDA APIs, programming
model and development tools. If you do not agree with the
terms and conditions of the license agreement, then do not
download or use the software.

Last updated: October 8, 2021.


Preface
-------

Do you accept the above EULA? (accept/decline/quit):
accept
```

```
CUDA Installer
- [ ] Driver
     [ ] 515.43.04
+ [X] CUDA Toolkit 11.7
  [X] CUDA Demo Suite 11.7
  [X] CUDA Documentation 11.7
  Options
  Install
```

Install CUDA 11.7 (Cont'd)

```
Edit .bashrc
export PATH=/usr/local/cuda-11.7/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.7/lib64:$LD_LIBRARY_PATH

Or
echo "export PATH=/usr/local/cuda-11.7/bin:\$PATH" >> ~/.bashrc
echo "export LD_LIBRARY_PATH=/usr/local/cuda-11.7/lib64:\$LD_LIBRARY_PATH" >>
~/.bashrc

sudo reboot
```

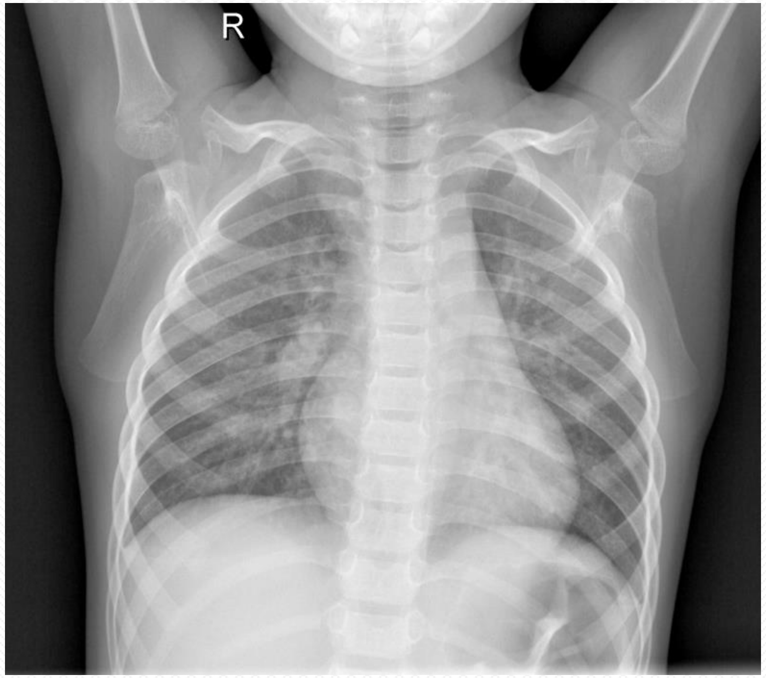


HOMEWORK #3

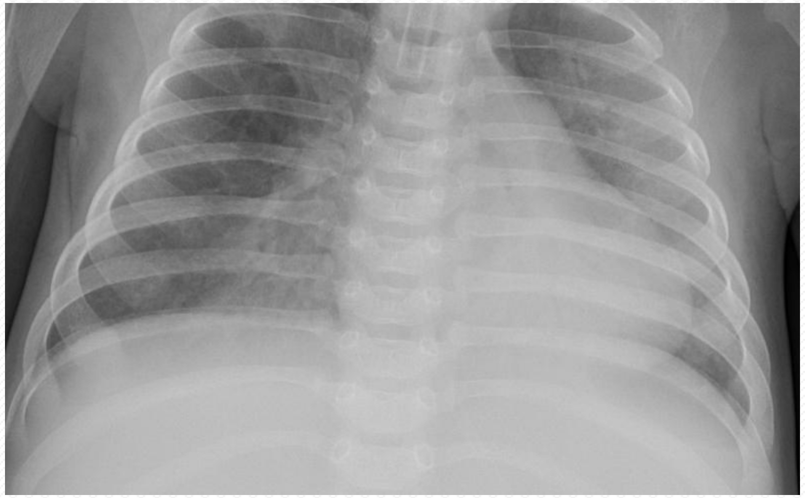
MEDICAL IMAGE CLASSIFICATION: DATASET PREPARATION

https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

Normal



Pneumonia



MEDICAL IMAGE CLASSIFICATION: IMPORT

```
import numpy as np # forlinear algebra
import matplotlib.pyplot as plt #for plotting things
import os
from PIL import Image # for reading images

# Keras Libraries <- CNN
import tensorflow as tf
from tensorflow.keras import datasets, layers, models, Model, Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, AveragePooling2D,
Flatten, Dense, Input, BatchNormalization, Concatenate, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
#from sklearn.metrics import classification_report, confusion_matrix # <- define
evaluation metrics
```

MEDICAL IMAGE CLASSIFICATION: DATASET PATH (1)

학습할 이미지들의 파일경로를 가져오기

```
mainDIR = os.listdir('./chest_xray')
print(mainDIR)
train_folder= './chest_xray/train/'
val_folder = './chest_xray/val/'
test_folder = './chest_xray/test/'
# train
os.listdir(train_folder)
train_n = train_folder+'NORMAL/'
train_p = train_folder+'PNEUMONIA/'
#Normal pic
print(len(os.listdir(train_n)))
rand_norm= np.random.randint(0,len(os.listdir(train_n)))
norm_pic = os.listdir(train_n)[rand_norm]
print('normal picture title: ',norm_pic)
norm_pic_address = train_n+norm_pic
#Pneumonia
rand_p = np.random.randint(0,len(os.listdir(train_p)))
sic_pic =  os.listdir(train_p)[rand_norm]
sic_address = train_p+sic_pic
print('pneumonia picture title:', sic_pic)
```

MEDICAL IMAGE CLASSIFICATION: DATASET SHOW (2)

학습할 이미지를 불러오고 **PLOT**

```
# Load the images
norm_load = Image.open(norm_pic_address)
sic_load = Image.open(sic_address)

#Let's plt these images
f = plt.figure(figsize= (10,6))
a1 = f.add_subplot(1,2,1)
img_plot = plt.imshow(norm_load)
a1.set_title('Normal')

a2 = f.add_subplot(1, 2, 2)
img_plot = plt.imshow(sic_load)
a2.set_title('Pneumonia')
plt.show()
# let's build the CNN model
```

MEDICAL IMAGE CLASSIFICATION: BUILD MODEL

Homework #3

지난시간에 이어 CNN LAYER를 추가.

MEDICAL IMAGE CLASSIFICATION: DATASET PREPARATION

```
num_of_test_samples = 600
batch_size = 32

# Fitting the CNN to the images
# The function ImageDataGenerator augments your image by iterating through image as
your CNN is getting ready to process that image

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)  #Image normalization.

training_set = train_datagen.flow_from_directory('./chest_xray/train',
                                                 target_size = (64, 64),
                                                 batch_size = 32,
                                                 class_mode = 'binary')
```

MEDICAL IMAGE CLASSIFICATION: DATASET PREPARATION (cont'd)

```
validation_generator = test_datagen.flow_from_directory('./chest_xray/val/',
    target_size=(64, 64),
    batch_size=32,
    class_mode='binary')

test_set = test_datagen.flow_from_directory('./chest_xray/test',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')
model_fin.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               802944
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 129
=================================================================
Total params: 813,217
Trainable params: 813,217
Non-trainable params: 0
_________________________________________________________________
```

DATASET PREPARATION: fit (training)

```
cnn_model = model_fin.fit(training_set,
                         steps_per_epoch = 163,
                         epochs = 10,
                         validation_data = validation_generator,
                         validation_steps = 624)

test_accu = model_fin.evaluate(test_set,steps=624)

model_fin.save('medical_ann.h5')
print('The testing accuracy is :',test_accu[1]*100, '%')
Y_pred = model_fin.predict(test_set, 100)
y_pred = np.argmax(Y_pred, axis=1)
max(y_pred)
```

DATASET PREPARATION

- **Homework #4**
  - 학습한 모델의 ANN 성능과 과 CNN 성능을 비교

HOMEWORK SUMMARY

- **Homework**
  - 이미지 필터 적용해보기 (Homework 1).
  - No Batch, Batch, relu를 사용했을 때 성능을 비교 (Homework 2).
  - MEDICAL 이미지 classification을 위한 cnn layer 추가 (Homework #3).
  - 학습한 모델의 ANN 성능과 과 CNN 성능을 비교 (Homework #4).

THANK YOU