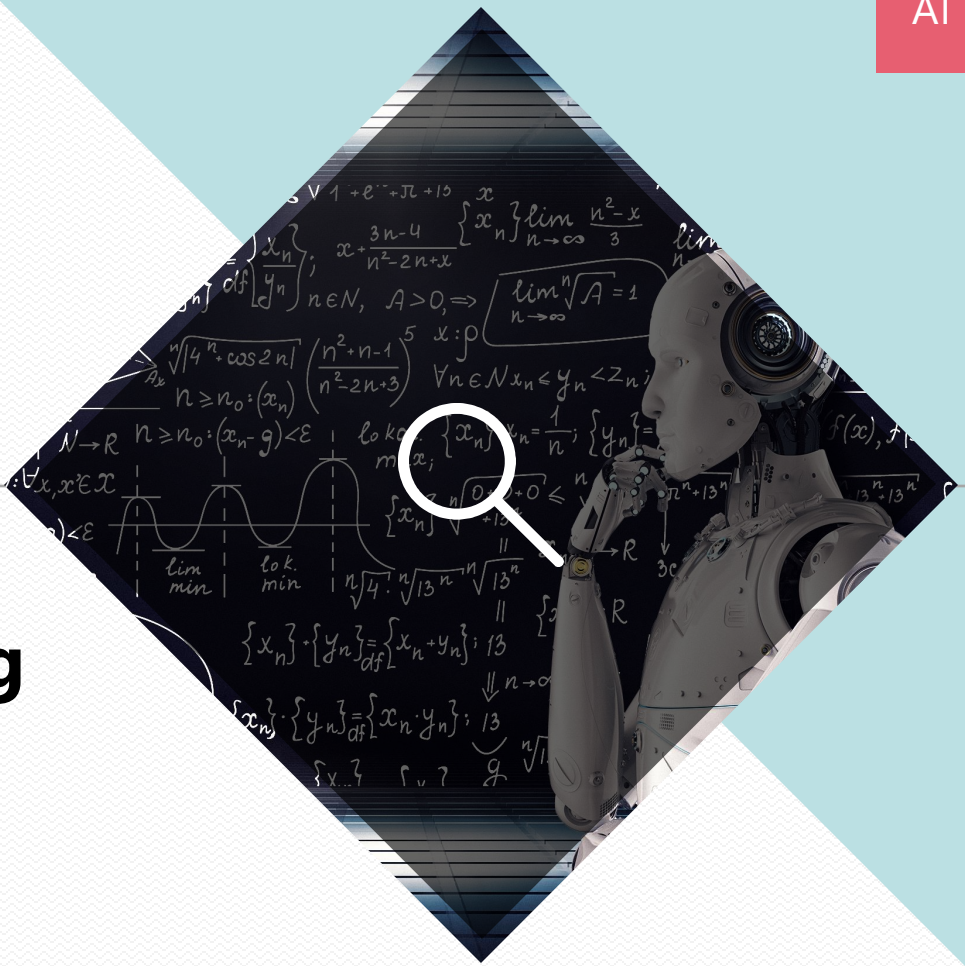
AI TRAINING SERIES



Smart Factory – Model Training

Objectives

- AI model을 training하는 방법을 학습한다
  - 직접 수집한 불량/양품 데이터로 AI 모델을 트레이닝하고 application에 적용할 수 있다
  - 원하는 기능을 수행하는 AI model을 training하고 프로젝트에 적용할 수 있다
  - OTX(OpenVINIO Training Extension) 사용 방법을 알 수 있다
  - Hyperparameter에 따른 성능 변화를 관찰 한다

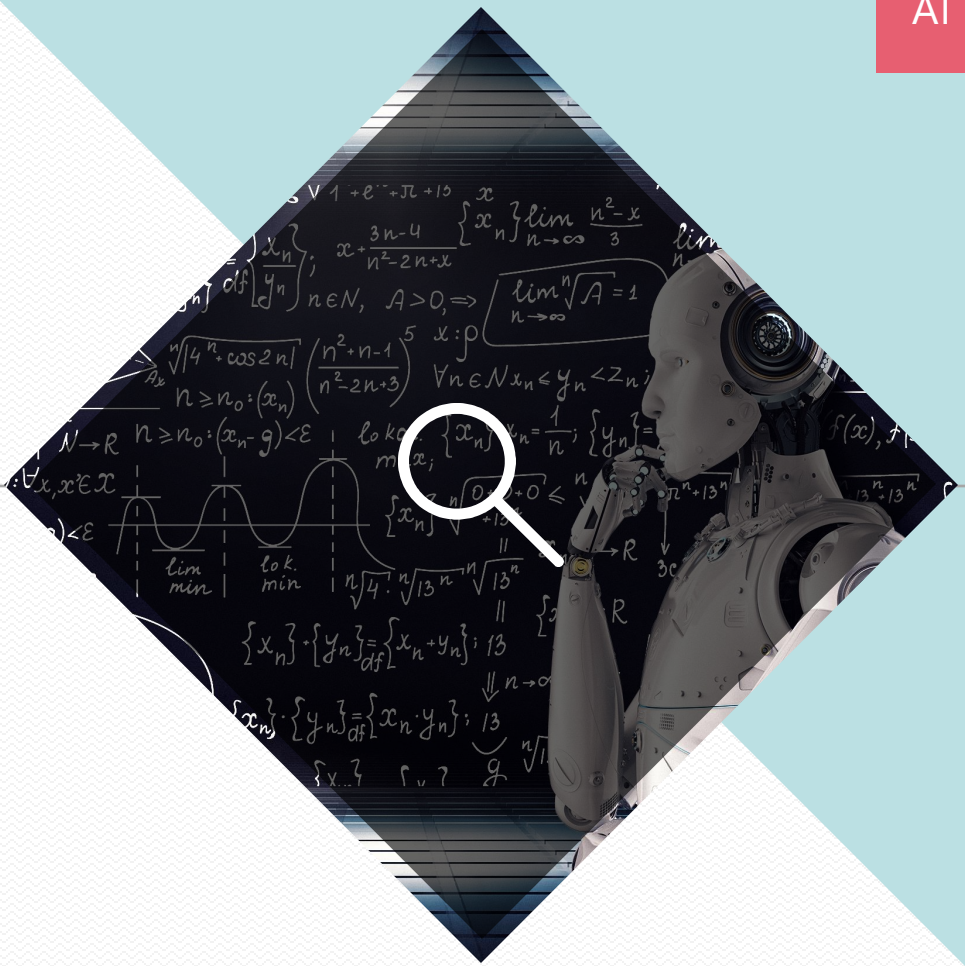
Software Dependencies

- Ubuntu 22.04
- Python 3.10
  - sudo apt install python3 python3-dev python3-venv
- OpenVINO Training Extensions
- Refer otx.md

```
sudo apt-get install g++ freeglut3-dev build-essential libx11-dev
libxmu-dev libxi-dev libglu1-mesa libglu1-mesa-dev gcc-multilib dkms
mesa-utils
```

OpenVINO Training Extensions(OTX)

- Low-code **transfer learning** framework for Computer Vision
- Diverse combinations of model architectures, learning methods, and task types based on PyTorch and OpenVINO toolkit
- Provides model templates
- Supports Hyper Parameter Optimization(HPO)



Nvidia GPU

Python 3.8

Cuda 11.7 **에서만 하기 내용으로 동작함.**

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
```

OTX Install

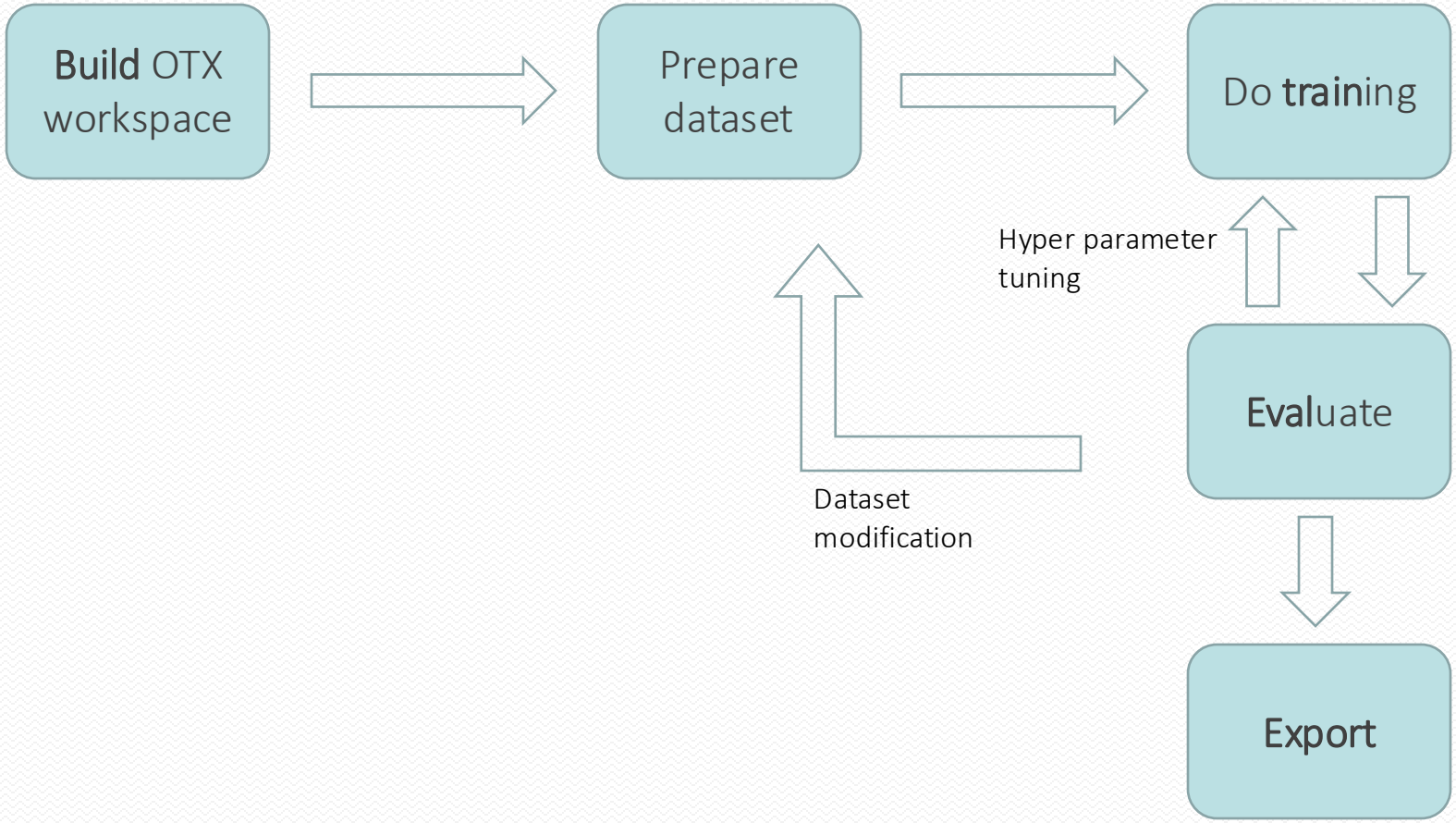
```
# Create a virtual env.
python -m venv .otx

# Activate virtual env.
source .otx/bin/activate
```

```
pip install wheel setuptools

# install command for torch==1.13.1 for CUDA 11.7:
pip install torch==1.13.1 torchvision==0.14.1 --extra-index-url https://download.pytorch.org/whl/cu117
pip install otx[full]
```

OTX Training Steps



OTX find

- ‘find’ gives supported templates list

```
otx find
otx find | awk ‘{print $2}’ |uniq
otx find --task classification
```

```
TASK

ACTION_CLASSIFICATION
ACTION_DETECTION
CLASSIFICATION
ANOMALY_CLASSIFICATION
ANOMALY_DETECTION
ANOMALY_SEGMENTATION
ROTATED_DETECTION
DETECTION
INSTANCE_SEGMENTATION
SEGMENTATION
```

OTX Classification Example

- Download the **flower_photos** dataset

```
mkdir -p ~/workspace/otx-flowers && cd $_
wget http://download.tensorflow.org/example_images/flower_photos.tgz -O - |tar xvz

tree ./ -d
./
└── flower_photos
    ├── daisy
    ├── dandelion
    ├── roses
    ├── sunflowers
    └── tulips

6 directories
```

```
find ./ -maxdepth 2 -type d | while read -r dir; do printf "%s:\t" "$dir"; find "$dir" -type f | wc
-l; done

./flower_photos/:   3671
./flower_photos/dandelion:    898
./flower_photos/tulips:       799
./flower_photos/daisy:        633
./flower_photos/sunflowers:   699
./flower_photos/roses:        641
```

OTX Classification Example - build

- Make sure otx virtual environment was loaded

```
source .otx/bin/activate
```

- Create a workspace for classification

```
(.otx)$ mkdir ~/workspace/otx-flowers && cd $_
(.otx)$ otx build --train-data-roots ./flower_photos/ --model MobileNet-V3-large-1x --workspace
./classification-task
```

- Explore the workspace

```
(.otx)$ cd classification-task
(.otx)$ cat template.yaml
(.otx)$ ds_count ./splitted_dataset 1
./splitted_dataset/:          3670
./splitted_dataset/train:     2936
./splitted_dataset/val:       734
```

Manage Your Favorite Functions

- You can create a bash function for your own comfort

```
mkdir -p ~/bin/
echo -e "\n\nsource ~/bin/my_funcs.sh" >> ~/.bashrc
touch ~/bin/my_funcs.sh
```

```
cat ~/bin/my_funcs.sh

ds_count() {
    find $1 -maxdepth $2 -type d | \
    while read dir
        do print "%s:\t" "$dir"
        find "$dir" -type f | wc -l
    done
}
```

```
# Log out/in or source the bash resource file
source ~/.basrhrc

# Newly defined dataset counting function should work
ds_count ./ 2
```

Troubleshoot - **NotADirectoryError**

```
$ otx build --train-data-roots ~/dataset/flower_photos --model EfficientNet-V2-S --workspace classification-task
...

[*] Detected dataset format: imagenet
Traceback (most recent call last):
  File "/home/intel/workspace/otx-classification/.otx/lib/python3.10/site-packages/datumaro/components/dataset.py", line 728, in import_from
    env.make_extractor(src_conf.format, src_conf.url, **extractor_kwargs)
  File "/home/intel/workspace/otx-classification/.otx/lib/python3.10/site-packages/datumaro/components/environment.py", line 283, in make_extractor
    return self.extractors.get(name)(*args, **kwargs)
  File "/home/intel/workspace/otx-classification/.otx/lib/python3.10/site-packages/datumaro/plugins/data_formats/imagenet.py", line 40, in __init__
    self._categories = self._load_categories(path)
  File "/home/intel/workspace/otx-classification/.otx/lib/python3.10/site-packages/datumaro/plugins/data_formats/imagenet.py", line 47, in _load_categories
    raise NotADirectoryError(errno.ENOTDIR, os.strerror(errno.ENOTDIR), path)
NotADirectoryError: [Errno 20] Not a directory: '/home/intel/dataset/flower_photos'

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  ...
```

Troubleshoot – **NotADirectoryError**(Cont'd)

- Trigger PDB Python debugger

import pdb; pdb.set_trace()

```
42
43      def _load_categories(self, path):
44          label_cat = LabelCategories()
45          for dirname in sorted(os.listdir(path)):
46              if not os.path.isdir(os.path.join(path, dirname)):
47                  import pdb; pdb.set_trace()
48                  raise NotADirectoryError(errno.ENOTDIR, os.strerror(errno
```

```
[*]     - Updated: classificatoin-task/compression_config.json
[*] Detected dataset format: imagenet
> /home/intel/workspace/otx-classification/.otx/lib/python3.10/site-packages/datumaro/plugins/data_
-> raise NotADirectoryError(errno.ENOTDIR, os.strerror(errno.ENOTDIR), path)
(Pdb) l
 43         def _load_categories(self, path):
 44             label_cat = LabelCategories()
 45             for dirname in sorted(os.listdir(path)):
 46                 if not os.path.isdir(os.path.join(path, dirname)):
 47                     import pdb; pdb.set_trace()
 48  ->                 raise NotADirectoryError(errno.ENOTDIR, os.strerror(errno.ENOTDIR), path)
 49                 if dirname != ImagenetPath.IMAGE_DIR_NO_LABEL:
 50                     label_cat.add(dirname)
 51             return {AnnotationType.label: label_cat}
 52
 53         def _load_items(self, path):
(Pdb) p path
'/home/intel/dataset/flower_photos'
(Pdb) p dirname
'LICENSE.txt'
(Pdb) os.path.isdir(os.path.join(path, dirname))
False
(Pdb) p os.path.join(path, dirname)
'/home/intel/dataset/flower_photos/LICENSE.txt'
(Pdb)
```

OTX Classification Example - train

- Start training

```
(.otx)$ otx train
…
2023-08-01 11:37:14,947 | INFO : Epoch [21][46/46]          lr: 5.122e-03, eta: 0:29:50, time:
0.254, data_time: 0.109, memory: 1855, current_iters: 965, loss: 0.0109, sharpness: 0.0383,
max_loss: 0.0492
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 734/734, 420.5 task/s, elapsed: 2s, ETA:
0s
2023-08-01 11:37:16,826 | INFO :
Early Stopping at :20 with best accuracy: 0.9618528747558593
2023-08-01 11:37:16,826 | INFO : Exp name: outputs/20230801_113219_train/logs
2023-08-01 11:37:16,826 | INFO : Epoch(val) [21][46]        accuracy_top-1: 0.9578, accuracy_top-5:
1.0000, daisy accuracy: 0.9606, dandelion accuracy: 0.9944, roses accuracy: 0.9457, sunflowers
accuracy: 0.9640, tulips accuracy: 0.9187, mean accuracy: 0.9567, accuracy: 0.9578, current_iters:
966
2023-08-01 11:37:16,828 | INFO : MemCacheHandlerBase uses 0 / 0 (0.0%) memory pool and store 0
items.
2023-08-01 11:37:18,115 | INFO : called save_model
2023-08-01 11:37:18,283 | INFO : Final model performance: Performance(score: 0.9618528747558593,
dashboard: (18 metric groups))
2023-08-01 11:37:18,283 | INFO : train done.
otx train time elapsed:  0:04:59.112100
otx train CLI report has been generated: outputs/20230801_113219_train/cli_report.log
```

Troubleshoot – **loss: nan**

```
| INFO : Exp name: outputs/20230805_125525_train/logs
| INFO : Epoch [1][46/46]          lr: 1.000e+00, eta: 0:00:00, time: 0.263, data_time: 0.005, memory: 1831, current_iters:
45, loss: nan
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>] 734/734, 660.9 task/s, elapsed: 1s, ETA:     0s
| INFO : Saving best checkpoint at 1 epochs
| INFO : Exp name: outputs/20230805_125525_train/logs
| INFO : Epoch(val) [1][46]          accuracy_top-1: 0.0000, accuracy_top-5: 0.0000, daisy accuracy: 0.0000, dandelion
accuracy: 0.0000, roses accuracy: 0.0000, sunflowers accuracy: 0.0000, tulips accuracy: 1.0000, mean accuracy: 0.2000,
accuracy: 0.0000, current_iters: 46
| INFO : MemCacheHandlerBase uses 0 / 0 (0.0%) memory pool and store 0 items.
| INFO : called save_model
```

Troubleshoot – **loss: nan** (Cont'd)

- Try to adjust **learning rate** if nan happens during a training
  - calculation result diverged
  - divided by zero

$$w' = w - \alpha \frac{\partial J}{\partial w}$$

learning rate

$$b' = b - \alpha \frac{\partial J}{\partial b}$$

learning rate

[template.yaml]

```
learning_rate:
    default_value: 0.0058
    auto_hpo_state: POSSIBLE
```

```
# Maximum learning_rate: 1.0 -- configuration.yaml
$ otx train params --learning_parameters.learning_rate 1.0 --learning_parameters.num_iters 1

# Minimum learning_rate: 1e-07 – configuration.yaml
$ otx train params --learning_parameters.learning_rate 1e-07 --learning_parameters.num_iters 1
```

Troubleshoot – **loss: nan** (Cont'd)

- Try to adjust batch size
  - depends on GPU's hardware capability
  - smaller batch size will take longer time to achieve high accuracy

[template.yaml]

```
batch_size:
    default_value: 64
    auto_hpo_state: POSSIBLE
```

```
# Half size of default batch size(64)
$ otx train params --learning_parameters.batch_size 32 --learning_parameters.num_iters 1

# Try more half
$ otx train params --learning_parameters.batch_size 16 --learning_parameters.num_iters 1
```

OTX Classification Example - train

- Training outputs

```
(.otx)$ tree outputs/latest_trained_model/
outputs/latest_trained_model/
├── cli_report.log
├── logs
│   ├── 20230801_113221.log.json
│   ├── best_epoch_13.pth
│   ├── epoch_21.pth
│   ├── latest.pth -> epoch_21.pth
│   └── tf_logs
│       └── events.out.tfevents.1690857143.intel-SYS-220U-TNR.4000570.0
└── models
    ├── label_schema.json
    └── weights.pth
```

OTX Classification Example – eval(optional)

- Evaluate trained model for test dataset

```
(.otx)$ otx eval --test-data-roots ./splitted_dataset/val --load-weights
./outputs/latest_trained_model/logs/best_epoch_13.pth
```

OTX Classification Example – export

- Export trained model to OpenVINO format

```
$ otx export

…
Find more information about API v2.0 and IR v11 at
https://docs.openvino.ai/latest/openvino_2_0_transition_guide.html
[ SUCCESS ] Generated IR version 11 model.
[ SUCCESS ] XML file: /home/litcoder/workspace/otx-flowers/classification-
task/outputs/20230801_140718_export/logs/model.xml
[ SUCCESS ] BIN file: /home/litcoder/workspace/otx-flowers/classification-
task/outputs/20230801_140718_export/logs/model.bin

2023-08-01 14:07:26,630 - mmdeploy - INFO - Successfully exported OpenVINO model:
outputs/20230801_140718_export/logs/model_ready.xml
2023-08-01 14:07:26,749 | INFO : Exporting completed
```

OTX Classification Example – Application

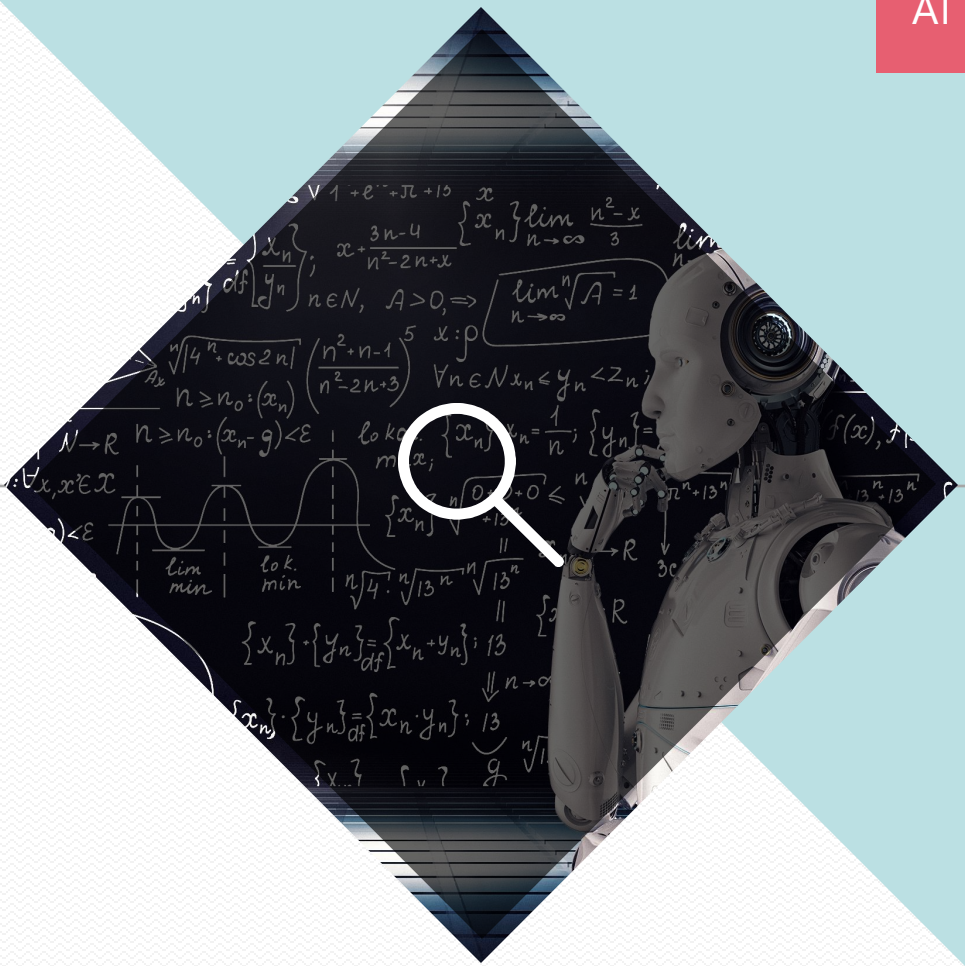
- Test the trained model with any downloaded flower images

```
(.otx) $ wget
https://raw.githubusercontent.com/openvinotoolkit/openvino/master/samples/python/hello_classificati
on/hello_classification.py

(.otx) $ python hello_classification.py ./outputs/20230801_140718_export/openvino/openvino.xml
./test/1.jpg "CPU"
[ INFO ] Creating OpenVINO Runtime Core
[ INFO ] Reading the model: ./outputs/20230801_140718_export/openvino/openvino.xml
[ INFO ] Loading the model to the plugin
[ INFO ] Starting inference in synchronous mode
[ INFO ] Image path: ./test/1.jpg
[ INFO ] Top 10 results:
[ INFO ] class_id probability
[ INFO ] --------------------
[ INFO ] 3        3.5824265
[ INFO ] 0        1.9845574
[ INFO ] 2        -0.5692880
[ INFO ] 1        -2.5086317
[ INFO ] 4        -2.9615641
[ INFO ]
[ INFO ] This sample is an API example, for any performance measurements please use the dedicated
benchmark_app tool
```

'daisy', 'dandelion', 'roses', 'sunflowers', 'tulips'

AI TRAINING SERIES



Intel arc GPU (XPU)

Install Intel arc graphic deriver in ubuntu (1)

url: https://www.intel.co.kr/content/www/kr/ko/download/747008/intel-arc-graphics-driver-ubuntu.html

인텔® Arc™ 그래픽 드라이버 - Ubuntu*

ID	날짜	버전
747008	11/2/2023	Latest(최신)

소개

이 릴리스는 인텔® Arc™ A-시리즈 그래픽의 Ubuntu* 23.04 및 22.04용 패키지 업데이트를 제공합니다.

세부 설명

인텔® Arc™ A-시리즈 그래픽용 Linux* 기반 운영 체제 배포용 범용 GPU(GPGPU) 기능을 활성화하기 위해 인텔 소프트웨어 설치, 배포 및 업데이트하는 방법에 대한 설명서 및 지침은 여기 https://dgpu-docs.intel.com/driver/client/overview.html 찾을 수 있습니다.

클릭

OS 지원

- 우분투* 22.04
- 우분투* 23.04

Install Intel arc graphic deriver in ubuntu (2)

url: https://dgpu-docs.intel.com/driver/client/overview.html

Ubuntu 버전 24.10

Installing Client GPUs on Ubuntu Desktop 24.10

Support for Lunar Lake and initial support for Battle Mage has been backported from kernel version 6.12 to version 6.11, which is included in Ubuntu 24.10. However, as this version of Ubuntu does not include the latest compute and media-related packages, we offer the intel-graphics Personal Package Archive (PPA). The PPA provides early access to newer packages, along with additional tools and features such as EU debugging.

Use the following commands to install the intel-graphics PPA and the necessary compute and media packages:

```
apt-get update
apt-get install -y software-properties-common

# Add the intel-graphics PPA for 24.10
add-apt-repository -y ppa:kobuk-team/intel-graphics

# Install the compute-related packages
apt-get install -y libze-intel-gpu1 libze1 intel-ocloc intel-opencl-icd clinfo

# Install the media-related packages
apt-get install -y intel-media-va-driver-non-free libmfx1 libmfx-gen1.2 libvpl2 libvpl-tools libva-glx2 va-driver-all vainfo
```

Install Intel arc graphic deriver in ubuntu (3)

url: https://dgpu-docs.intel.com/driver/client/overview.html

Ubuntu 버전 24.04

Installing Client GPUs on Ubuntu Desktop 24.04 LTS

The Ubuntu 24.04 repositories contain compute packages for various Intel graphics products. To install those packages, use the following commands:

```
# Install the Intel graphics GPG public key
wget -qO - https://repositories.intel.com/gpu/intel-graphics.key | \
  sudo gpg --yes --dearmor --output /usr/share/keyrings/intel-graphics.gpg

# Configure the repositories.intel.com package repository
echo "deb [arch=amd64,i386 signed-by=/usr/share/keyrings/intel-graphics.gpg] https://repositories.intel.com/gpu/ubuntu noble
  sudo tee /etc/apt/sources.list.d/intel-gpu-noble.list

# Update the package repository meta-data
sudo apt update

# Install the compute-related packages
apt-get install -y libze1 intel-level-zero-gpu intel-opencl-icd clinfo
```

Install Intel arc graphic deriver in ubuntu (4)

url: https://dgpu-docs.intel.com/driver/client/overview.html

Ubuntu 버전 22.04

Installing Client GPUs on Ubuntu Desktop 22.04 LTS

The Ubuntu 22.04 repositories do not contain compute packages for various Intel graphics products. To install these packages, you can use Intel's dedicated package repository.

```
# Install the Intel graphics GPG public key
wget -qO - https://repositories.intel.com/gpu/intel-graphics.key | \
  sudo gpg --yes --dearmor --output /usr/share/keyrings/intel-graphics.gpg

# Configure the repositories.intel.com package repository
echo "deb [arch=amd64,i386 signed-by=/usr/share/keyrings/intel-graphics.gpg] https://repositories.intel.com/gpu/ubuntu jammy
  sudo tee /etc/apt/sources.list.d/intel-gpu-jammy.list

# Update the package repository meta-data
sudo apt update

# Install the compute-related packages
apt-get install -y libze1 intel-level-zero-gpu intel-opencl-icd clinfo
```

Install openvino training extension (otx)(1)

github: https://github.com/openvinotoolkit/training_extensions

installation guide :
https://openvinotoolkit.github.io/training_extensions/latest/guide/get_started/installation.html

Otx 설치 가이드 페이지 이동

Installation

Please refer to the installation guide. If you want to make changes to the library, then a local installation is recommended.

클릭

- Install from PyPI
- Install from source

Install openvino training extension (otx)(2)

installation guide :
https://openvinotoolkit.github.io/training_extensions/latest/guide/get_started/installation.html

OpenVINO™ Training Extensions

Search Ctrl + K

Prerequisites

The current version of OpenVINO™ Training Extensions was tested in the following environment:

- Ubuntu 20.04
- Python >= 3.10

Install OpenVINO™ Training Extensions for users (CUDA/CPU)

1. Install OpenVINO™ Training Extensions package:

- A local source in development mode

Install openvino training extension (otx)(3)

installation guide :
https://openvinotoolkit.github.io/training_extensions/latest/guide/get_started/installation.html

중요! - Intel arc gpu의 경우 반드시 XPU devices로 설치!!

Install OpenVINO™ Training Extensions for users (XPU devices)

1. Install OpenVINO™ Training Extensions from source to use XPU functionality.

Minimum requirements

```
# Clone the training_extensions repository with the following command:
git clone https://github.com/openvinotoolkit/training_extensions.git
cd training_extensions

pip install -e '.[base]' --extra-index-url https://download.pytorch.org/whl/test/xpu
```

Note

Please, refer to the PyTorch official documentation guide to install prerequisites and resolve possible issues.

Install openvino training extension (otx)(4)

installation guide :
https://openvinotoolkit.github.io/training_extensions/latest/guide/get_started/installation.html

Install pytorch for Intel arc graphic

Install OpenVINO™ Training Extensions for users (XPU devices)

1. Install OpenVINO™ Training Extensions from source to use XPU functionality.

Minimum requirements

```
# Clone the training_extensions repository with the following command:
git clone https://github.com/openvinotoolkit/training_extensions.git
cd training_extensions

pip install -e '.[base]' --extra-index-url https://download.pytorch.org/whl/test/xpu
```

Note

Please, refer to the PyTorch official documentation guide to install prerequisites and resolve possible issues.

클릭

Install openvino training extension (otx)(5)

Getting Started on Intel GPU on pytorch:
https://pytorch.org/docs/stable/notes/get_start_xpu.html

Install pytorch for Intel arc graphic

Binaries

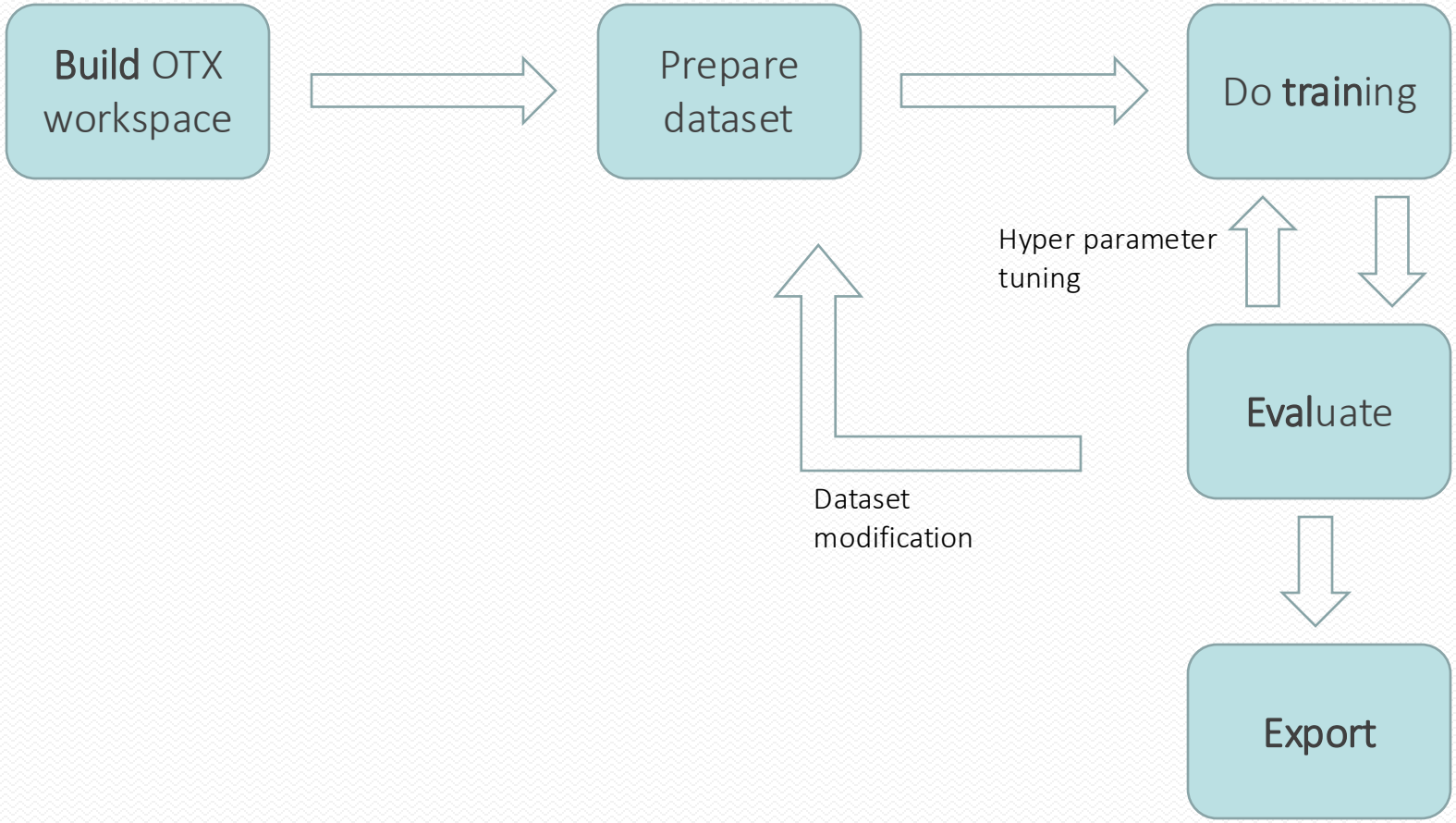
Platform Linux

Now we have all the required packages installed and environment activated. Use the following commands to install `pytorch`, `torchvision`, `torchaudio` on Linux.

For preview wheels

```
pip3 install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/test/xpu
```

OTX Training Steps



Classification using OTX (1)

installation guide :
https://openvinotoolkit.github.io/training_extensions/latest/guide/get_started/installation.html

OpenVINO™ Training Extensions

Get Started

Introduction

Installation

CLI Guide

API Quick-Guide

Tutorials

Base Tutorials

Training to deployment tutorials

Classification model

Object Detection model

Instance Segmentation model

Semantic Segmentation model

Anomaly Detection Tutorial

Action Classification model

클릭

Installation

Prerequisites

The current version of OpenVINO™ Training Extensions was tested in the following environment:

- Ubuntu 20.04
- Python >= 3.10

Install OpenVINO™ Training Extensions for users (CUDA/CPU)

1. Install OpenVINO™ Training Extensions package:

- A local source in development mode

Homework #3 – Train your own AI models

- Collect dataset for Smart Factory defect classification
- Train your own model
  - Try all classification models
  - Tune hyper parameters for highest accuracy
  - Measure FPS(Frame Per Seconds: inference speed)
  - Use hw03_template.md for reporting out
- Modify inference code as necessary

Homework03

Classification model	Accuracy	FPS	Training time	Batch size	Learning rate	Other prams
EfficientNet-V2-S						
EfficientNet-B0						
DeiT-Tiny						
MobileNet-V3-large-1x						

FPS 측정 방법

THANK YOU