intel AI

ARTIFICIAL INTELLIGENCE

Inference and OpenVINO

2018. Dec. 20

Intel Korea 이인구

인공지능 딥 러닝 구현 순서

구현 순서

- ✓ Framework 과 Network 모델을 선정한다. Image classification? Object Detection?
  - TF, Caffe, Mxnet, CNTK, AlexNet, LeNet, SSD, Yolo, GoogleNet, etc.

- ✓ Dataset 준비
  - 이미 학습에 사용된 Dataset에 추가 학습하거나 (예: Cifar, MNIST, ImageNet, etc)
  - 직접 Dataset을 구축 합니다.

- ✓ 학습 & 테스트
  - Deep Learning Framework 선택 (Caffe, Tensorflow, etc).
  - 학습과 최적화를 통해 최종적으로 Network 구성하고, Weights 값을 갖습니다. .

- ✓ 추론
  - Deep Learning Framework 선택 (Caffe, Tensorflow, **OpenVINO**, etc).
  - 모델 트레이닝에서 얻은 Network, Weights 값를 사용해서 주어진 입력 데이터가 훈련된 오브젝트중 에디에 가까운지을 분석해 줍니다.

인공지능 딥 러닝 구현 순서

1. Framework 과
Network 모델을 선정합니다.

- ✓ Framework - TF, Caffe, Mxnet, BigDL, CN TK….
- ✓ Network topology – ResNet50, Inception V3, MobileNet, SSD300….
- ✓ Category
- ✓ Image classification, Object Detection, Im age segmentation, Speech Recognition,…

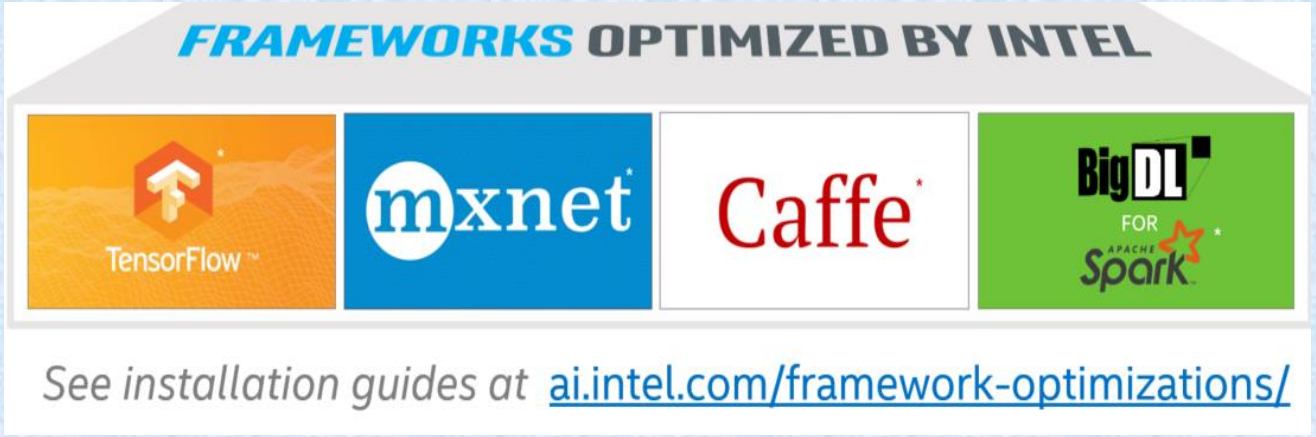


Image Recognition		
ResNet50*	✓	✓
InceptionV3*	✓	✓
Inception_ResV2*	Q2	2H
InceptionV4	Q2	2H
MobileNet	Q2	2H
SqueezeNets	Q2	2H
DenseNet	Q2	2H

Object Detection & Localization		
SSD-VGG16*	✓	✓
R-FCN	Q2	2H
Faster-RCNN	Q2	Q2
YoloV2	Q2	2H

Image Segmentation		
MaskRCNN	Q2	2H
U-Net	Q2	2H
3D-Unet*	Q2	2H

Image Generation		
DRAW	Q2	2H

Speech Recognition		
Deep Speech	Q2	Q2
Transformer	Q2	Q2

Language Translation		
GNMT*	✓	✓
Transformer	Q2	Q2

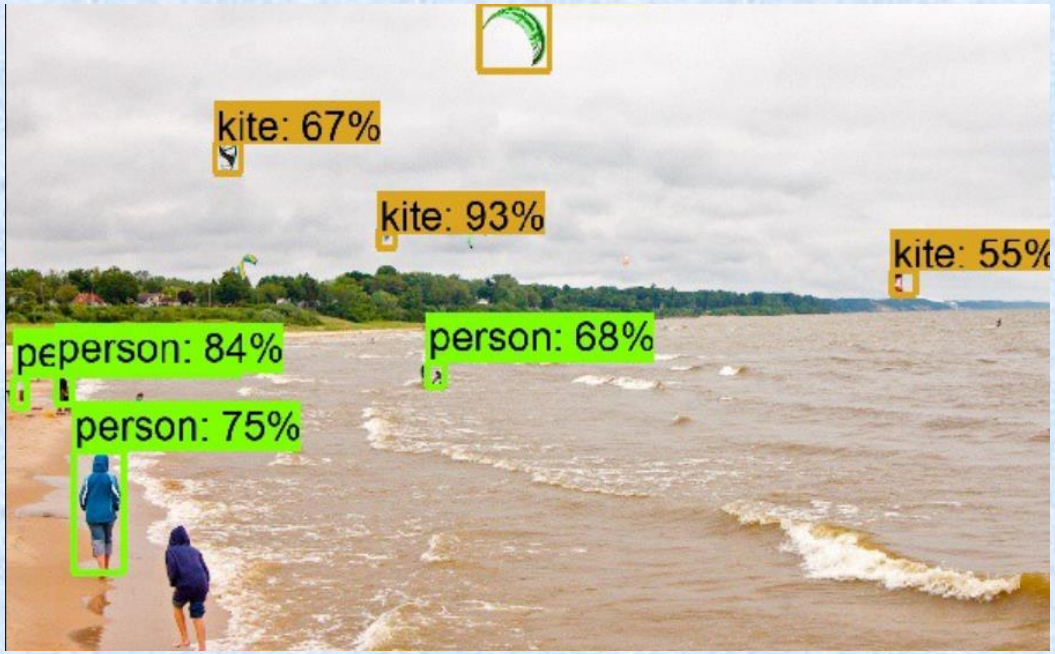
Adversarial networks		
DCGAN	Q2	Q2
3DGAN	Q2	2H

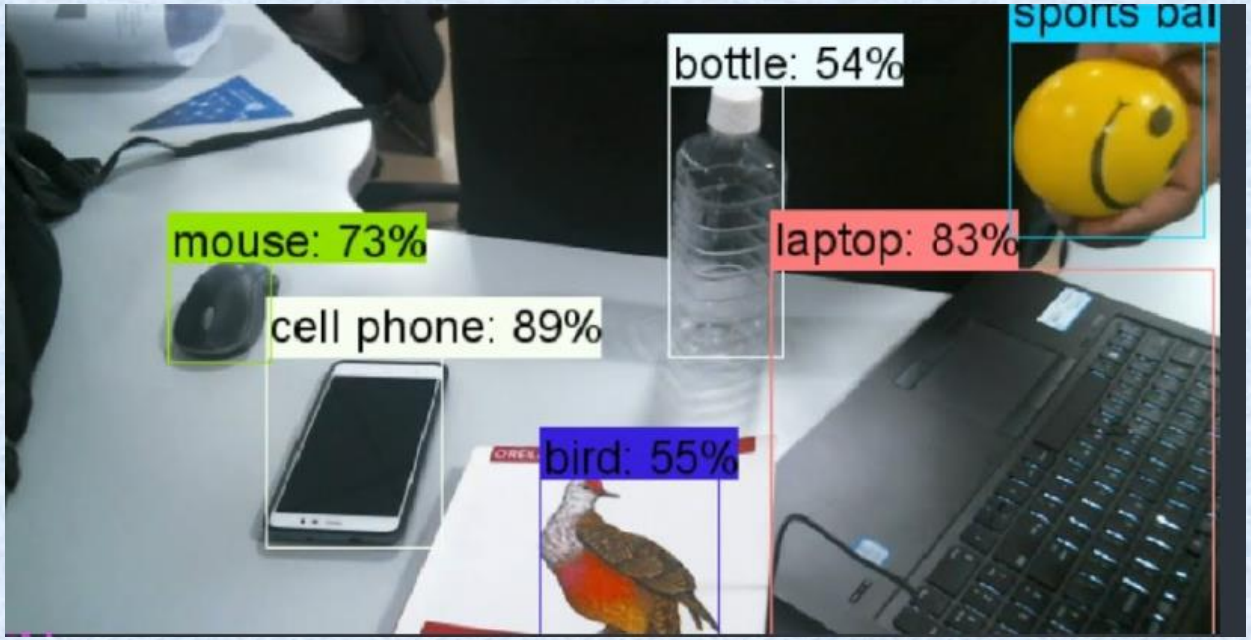
Recommender Systems		
Wide & Deep*	Q2	2H

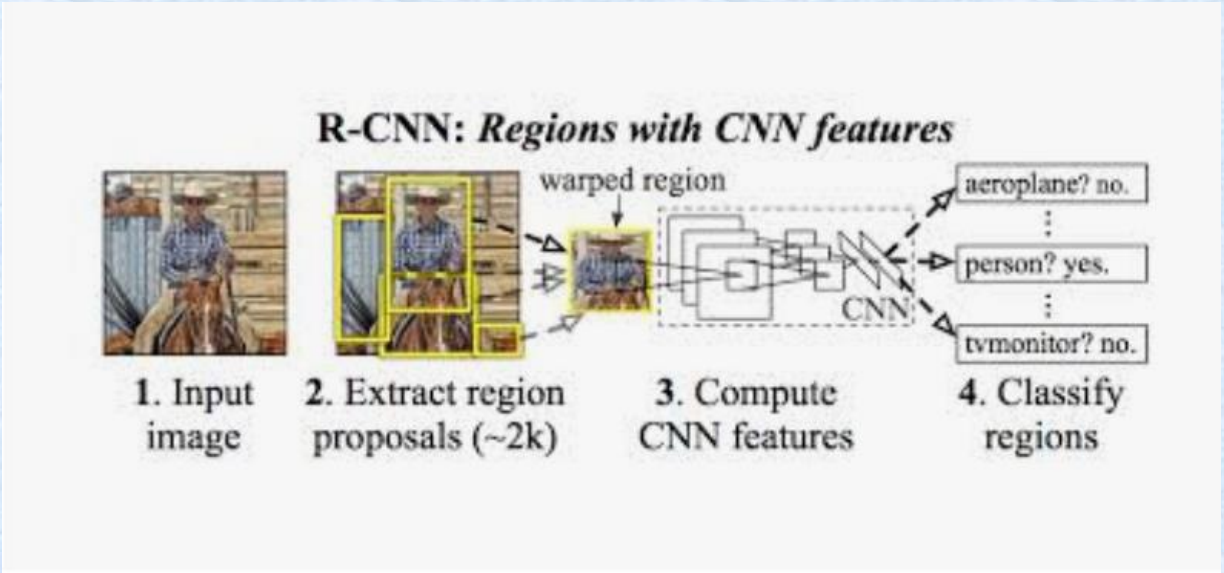
Text To Speech		
WaveNet	Q2	2H

Reinforcement Learning		
A3C	Q2	2H

Object detection







Detect and Classify Species of Fish from Fishing Vessels with Modern ...

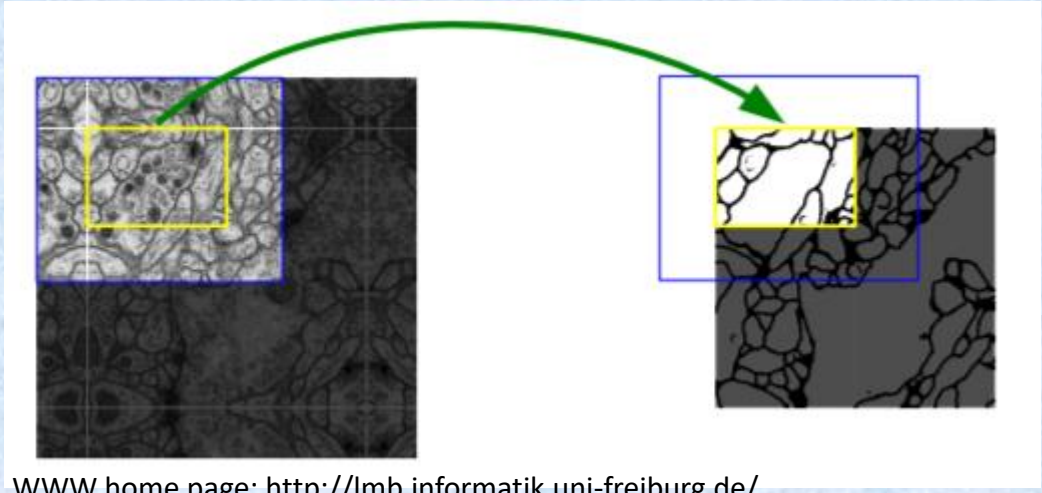
Image Recognition

Image
Pattern
Face

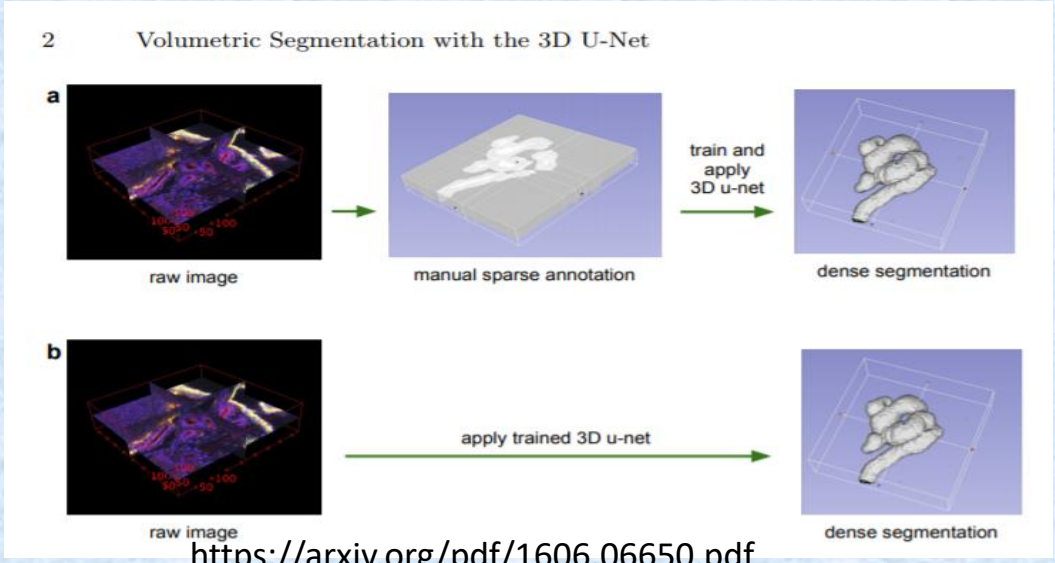
https://www.tensorflow.org/tutorials/images/image_recognition

Image segmentation

U-Net: Convolutional Networks for Biomedical Image Segmentation



WWW home page: http://lmb.informatik.uni-freiburg.de/



https://arxiv.org/pdf/1606.06650.pdf

Mask R-CNN

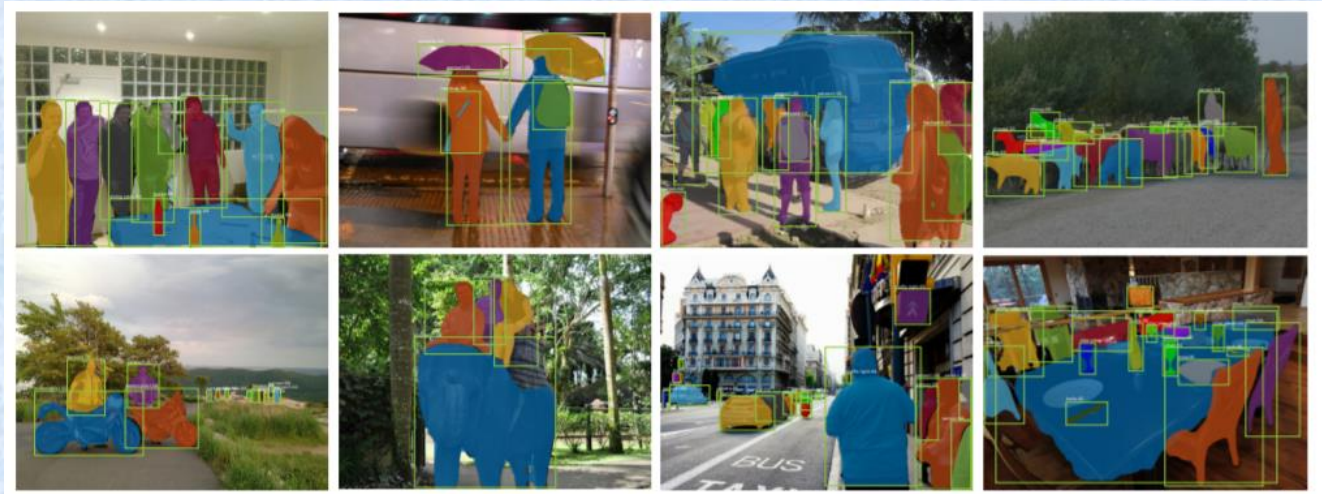
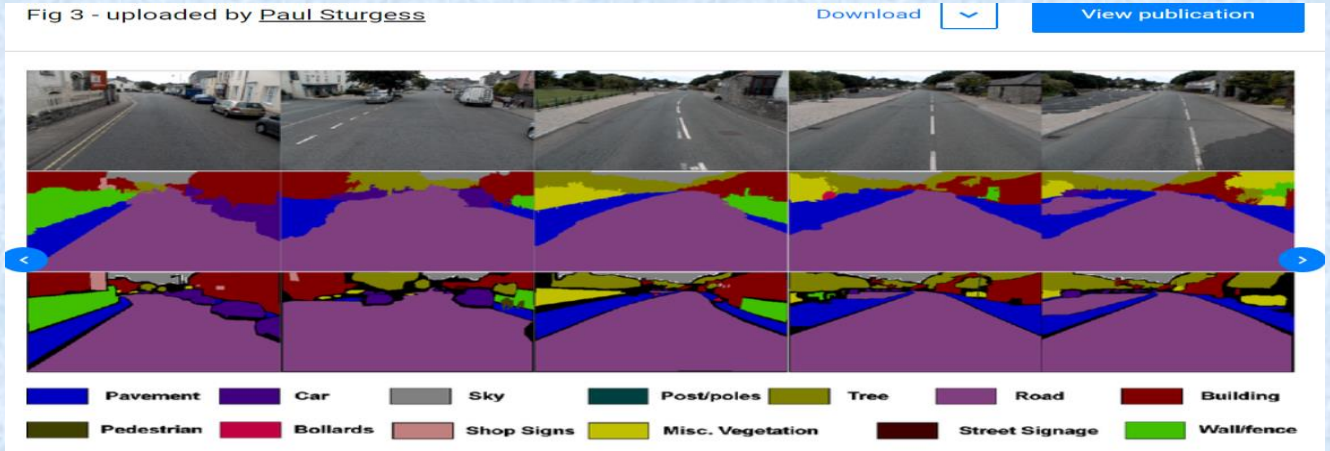


Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.



Wide & Deep Learning for Recommender Systems

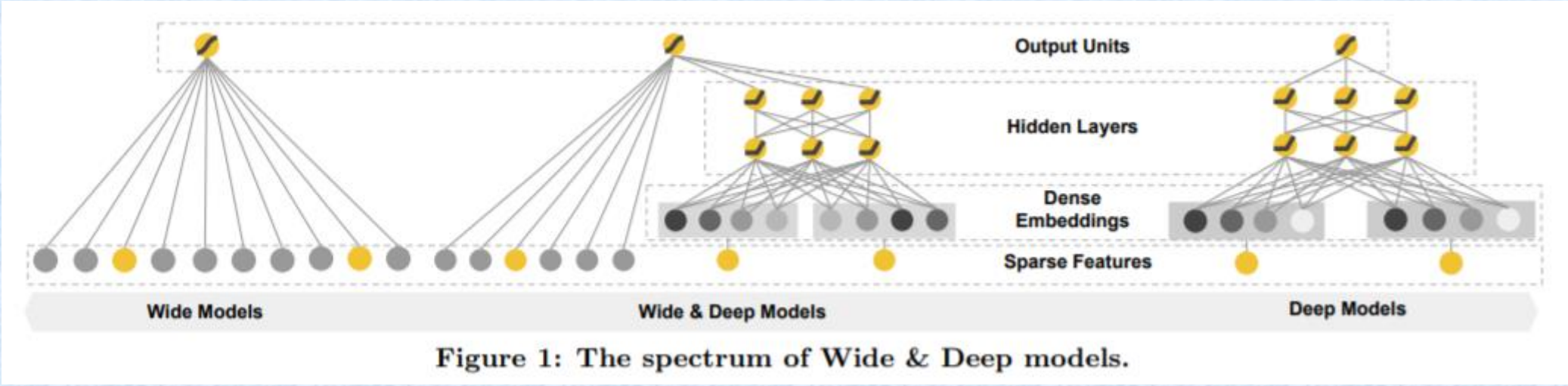


Figure 1: The spectrum of Wide & Deep models.

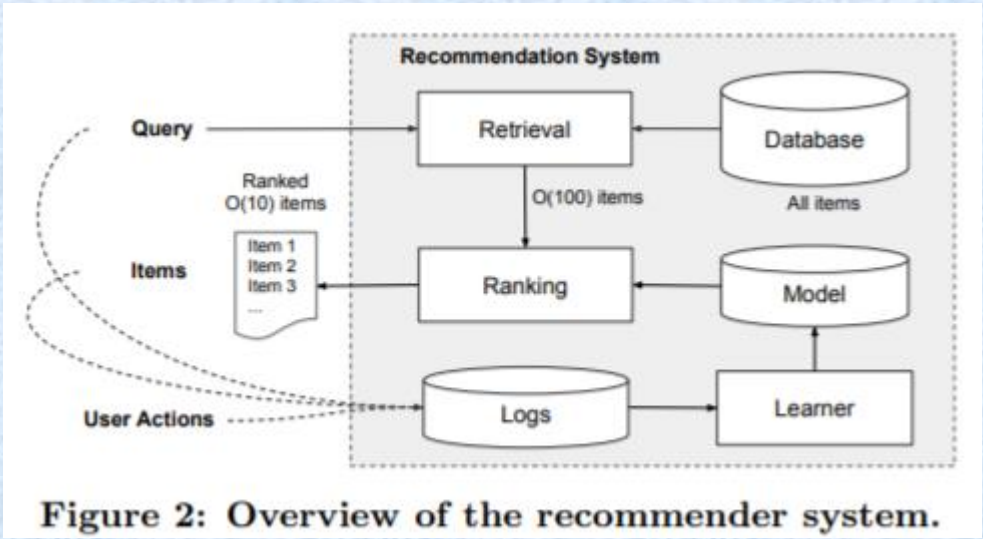
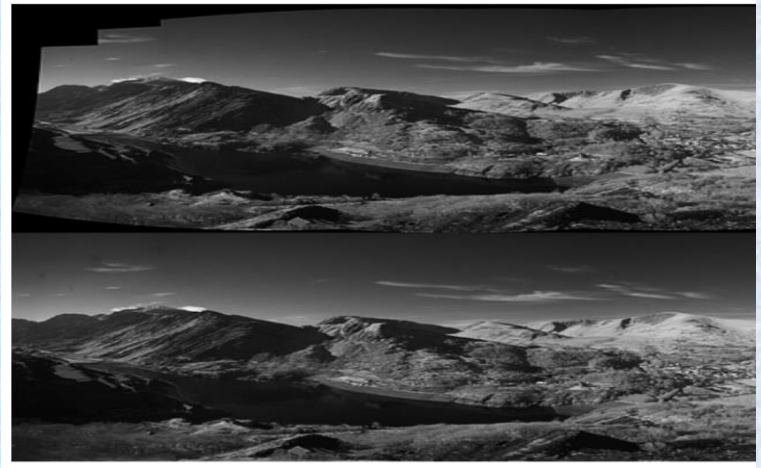


Figure 2: Overview of the recommender system.

Source : Google Inc.

Adversarial Networks

DCGAN (Deep Convoultional Generative adversarial Network)

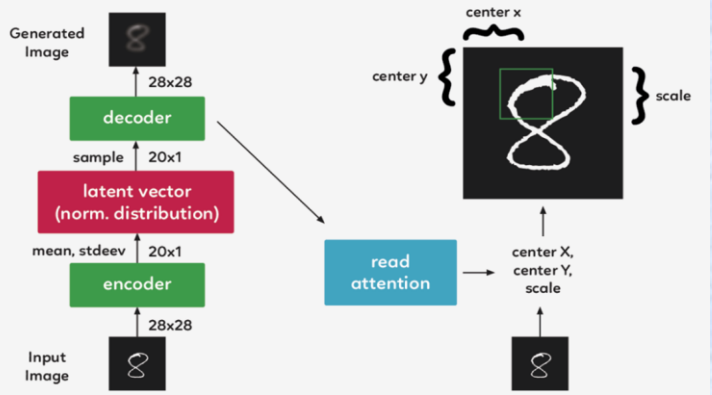


출처 : https://dev-strender.github.io/articles/2017-07/decan-introduction

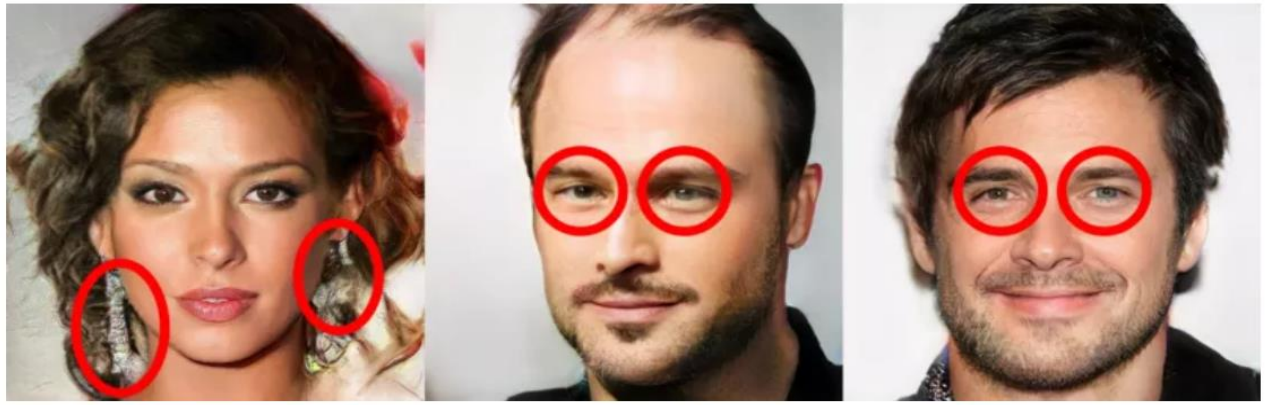
Image Generation

DRAW (A Recurrent Neural Network For Image Generation)

Figure 6. **Generated MNIST images.** All digits were generated by DRAW except those in the rightmost column, which shows the training set images closest to those in the column second to the right (pixelwise L^2 is the distance measure). Note that the network was trained on binary samples, while the generated images are mean probabilities.



The faces on the left were created by AI in 2014; on the right are ones made by AI in 2018. | Image: Goodfellow et al;



Some examples of AI-generated faces with obvious asymmetrical features. | Image by Kyle McDonald

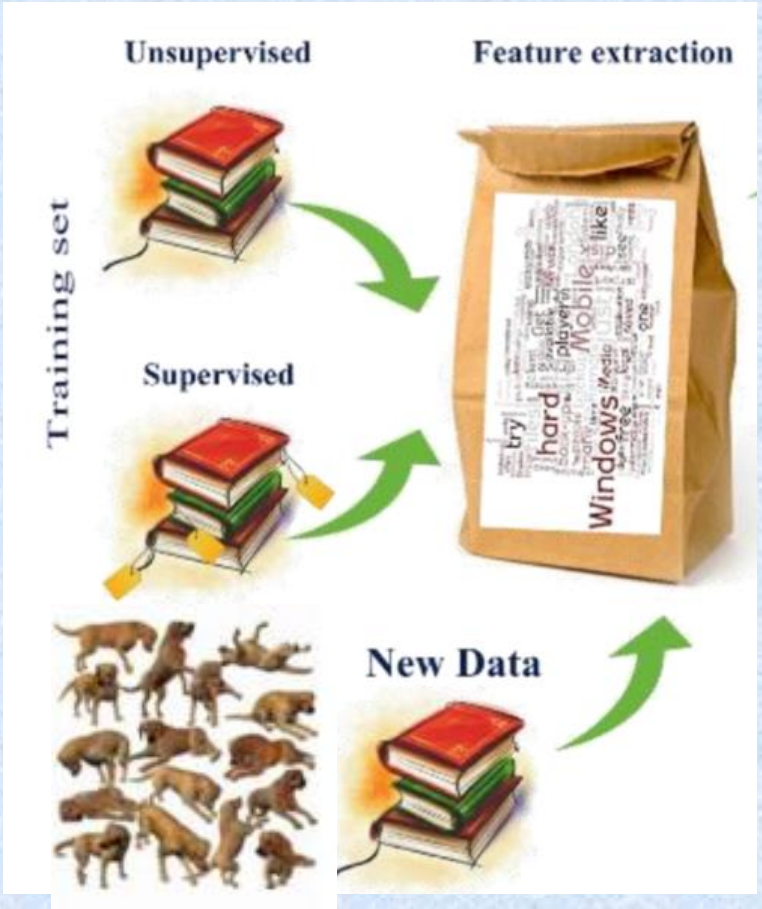
인공지능 딥 러닝 구현 순서

2. Dataset 준비

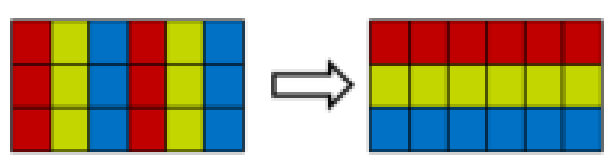
- ✓ Unsupervised data,
- ✓ Supervised data
- ✓ New Data.

Data pre-processing

Decode
Resize
Color conversion
(Interleave -> Planar)



Data Pre-processing

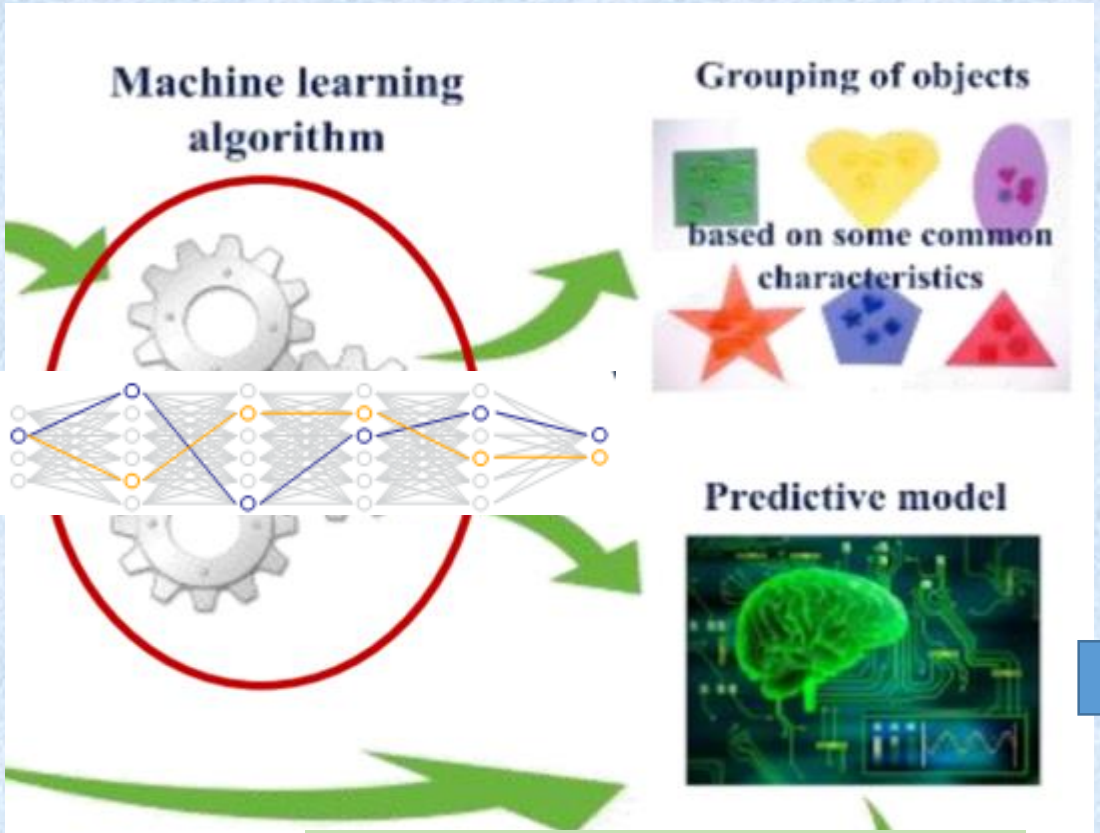


인공지능 딥 러닝 구현 순서

3. 학습 & 테스트

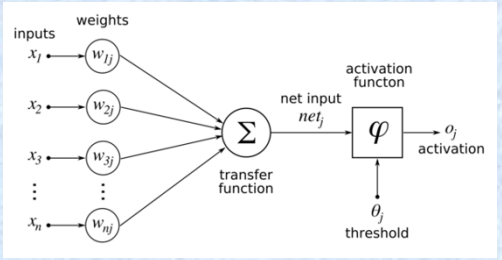
- ✓ 학습
- ✓ 모델만들기
- ✓ 테스트
- ✓ 최적화

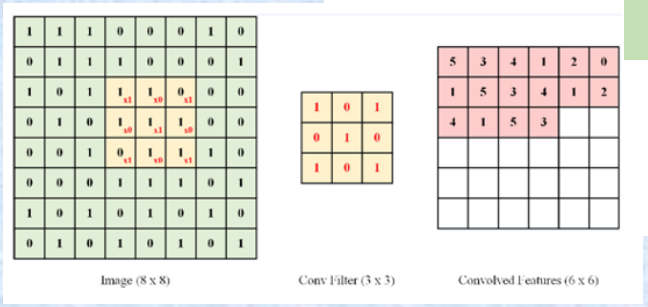
가중치값 생성



모델, 가중치값

Convolution, Pooling,
Softmax, fully-connected NN





인공지능 딥 러닝 구현 순서

4. 추론

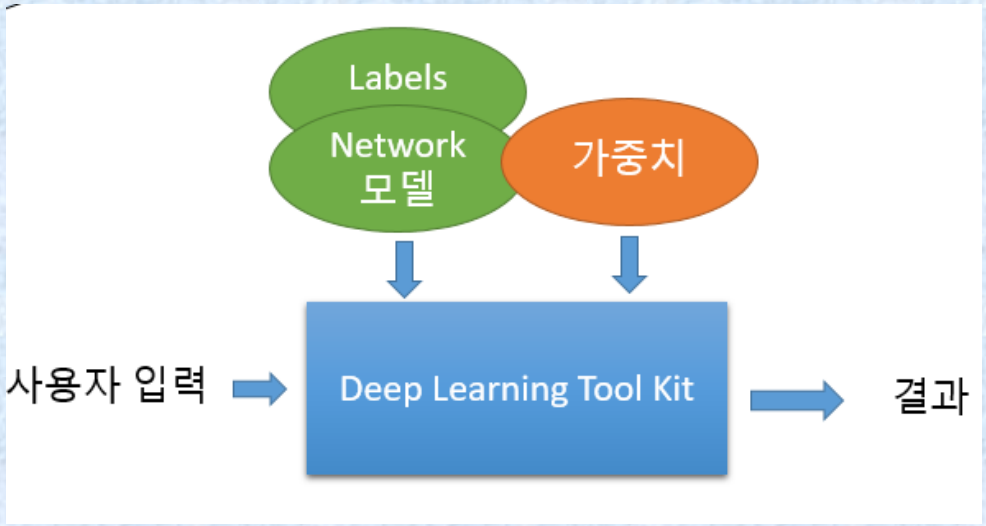


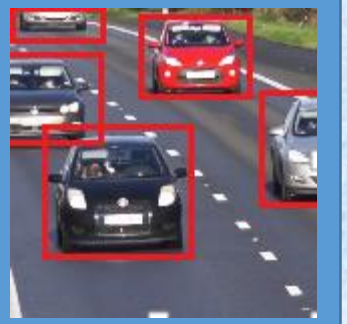
Image Classification

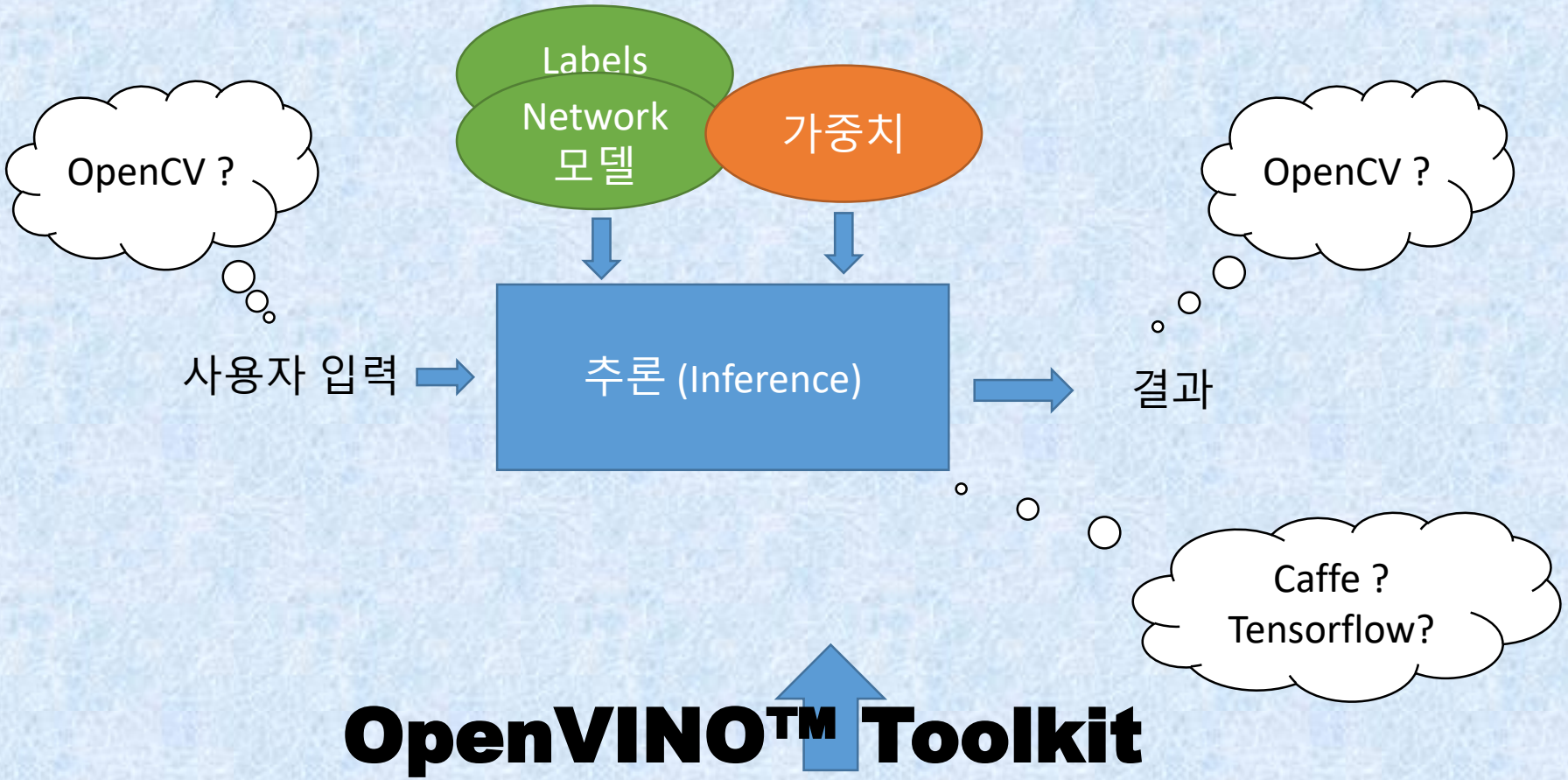
추론
Inference

가위?, 보?
(분류)

Object Detection

추론
Inference



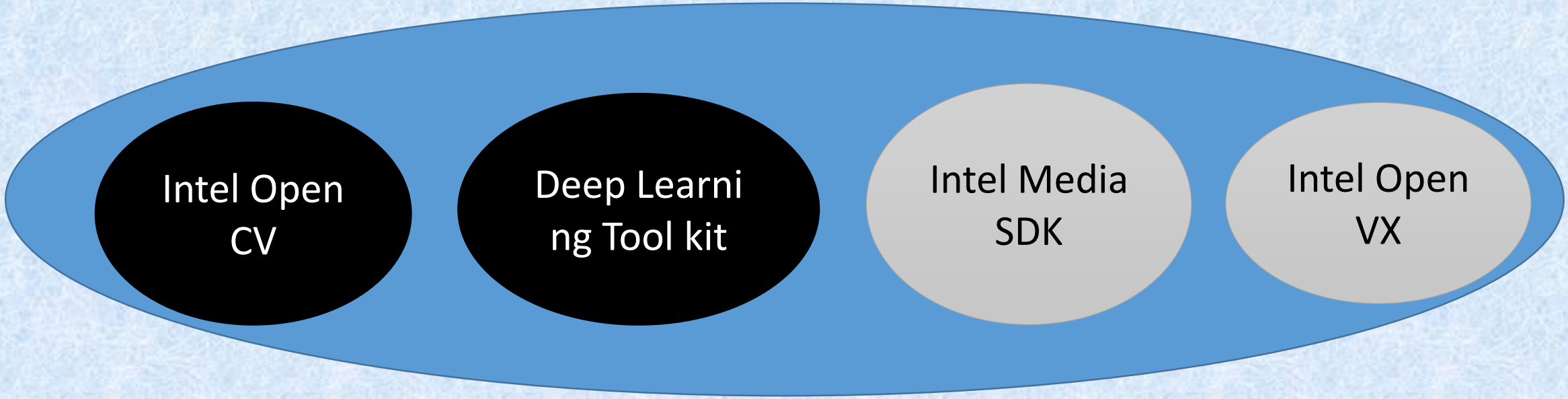


OpenVINO™ Toolkit

Open **V**isual **I**nference and **N**eural Network **O**ptimization

영상을 추론하고 신경망을 최적화 하는 도구 모음

OpenVINO™ Toolkit 의 구성



Inference Engine

1. 연산 장치들을 사용하여 추론을 수행한다.
2. Library형태로 제공된다.

Model Optimizer

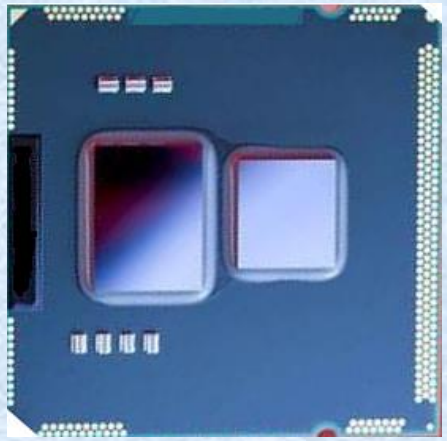
1. 현재 일반적으로 사용되고 있는 Deep Learning Framework의 결과물을 Inference Engine이 인식할 수 있는 포맷으로 전환한다.
2. 전환하는 과정에서 Network 모델의 최적화를 수행하여 Inference Engine으로 더 나은 Performance를 낼 수 있도록한다.

OpenVINO 등장의 배경

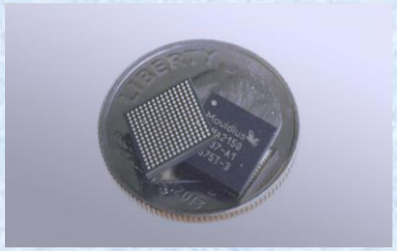
- ✓ 인텔이 제공하는 다양한 연산 장치들을 딥 러닝에 사용할 수 있게합니다.
  - CPU
  - GPU
  - FPGA (Field programmable gate array)
  - Movidius (VPU, a low power system on chip)
- ✓ 이를 위한 사용자 인터페이스를 제공합니다.
  - 하나의 인터페이스로 다수의 연산 장치를 효율적으로 사용할 수 있게 합니다.
- ✓ 최적의 성능을 제공합니다.
  - 각 연산 장치에 최적화된 plug-in을 제공합니다.
    - mklDNN, clDNN, DLNA,
  - IA (인텔 Architecture)에 최적화된 OpenCV, OpenCL, MediaSDK 같은 기타 Tool (SDK)과도 연동할 수 있게 합니다.

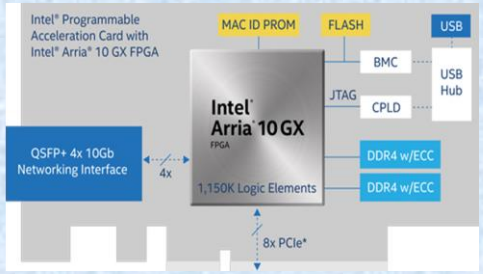
인텔의 연산 장치들

CPU



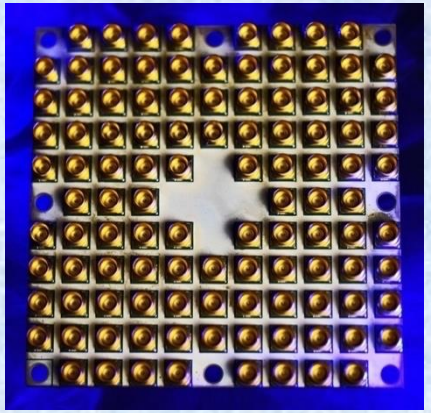
Movidius (VPU)





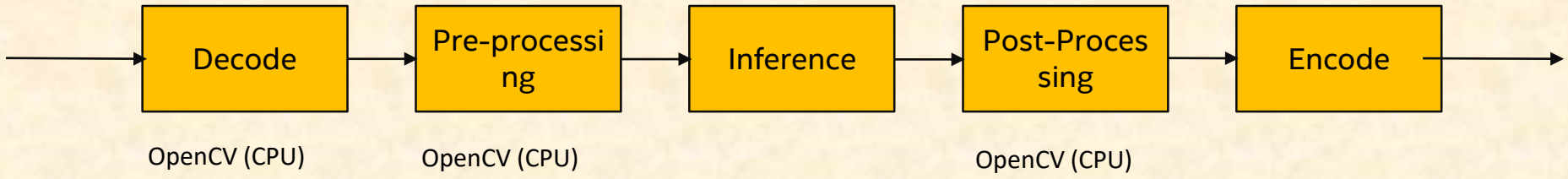
FPGA

Neuromorphic



Quantum Computing

인공지능 딥 러닝 구현 순서



Demo: Auto Recognition Demo Using Deep Learning Deployment Toolkit

인공지능 딥 러닝 추론 구현 순서

구현 순서

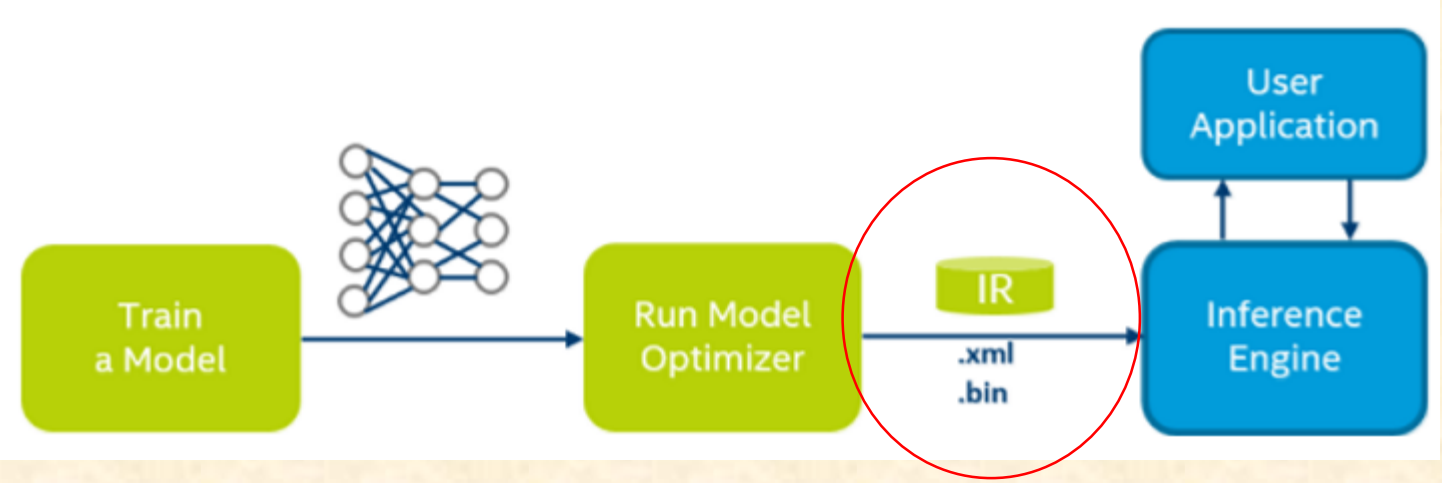
1. 학습을 통해 만들어진 모델을 가져오기 – 가중치 file, *.prototxt, *.caffemodel
1. OpenVINO 에서 Python을 이용해서 모델을 컨버젼 하기 -- *.bin, *.xml
1. 입력 데이터 위치 확인 하기- 이미지, 동영상, 카메라
1. 사용할 HW 을 선택하기 – CPU, GPU, Movidius
1. 입력 데이터를 가공하기 – Decode, Re-size, Color conversion
1. 인퍼런스 - OpenVINO or Streamer – Image classification, Object Detection.
1. 데이터 출력 - Boxing, Encoding
1. 실행 결과물

코딩하기

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

2. 모델 Conversion하기

Inference Engine이 원하는 포맷으로 Network 모델과 가중치 정보를 전환합니다.



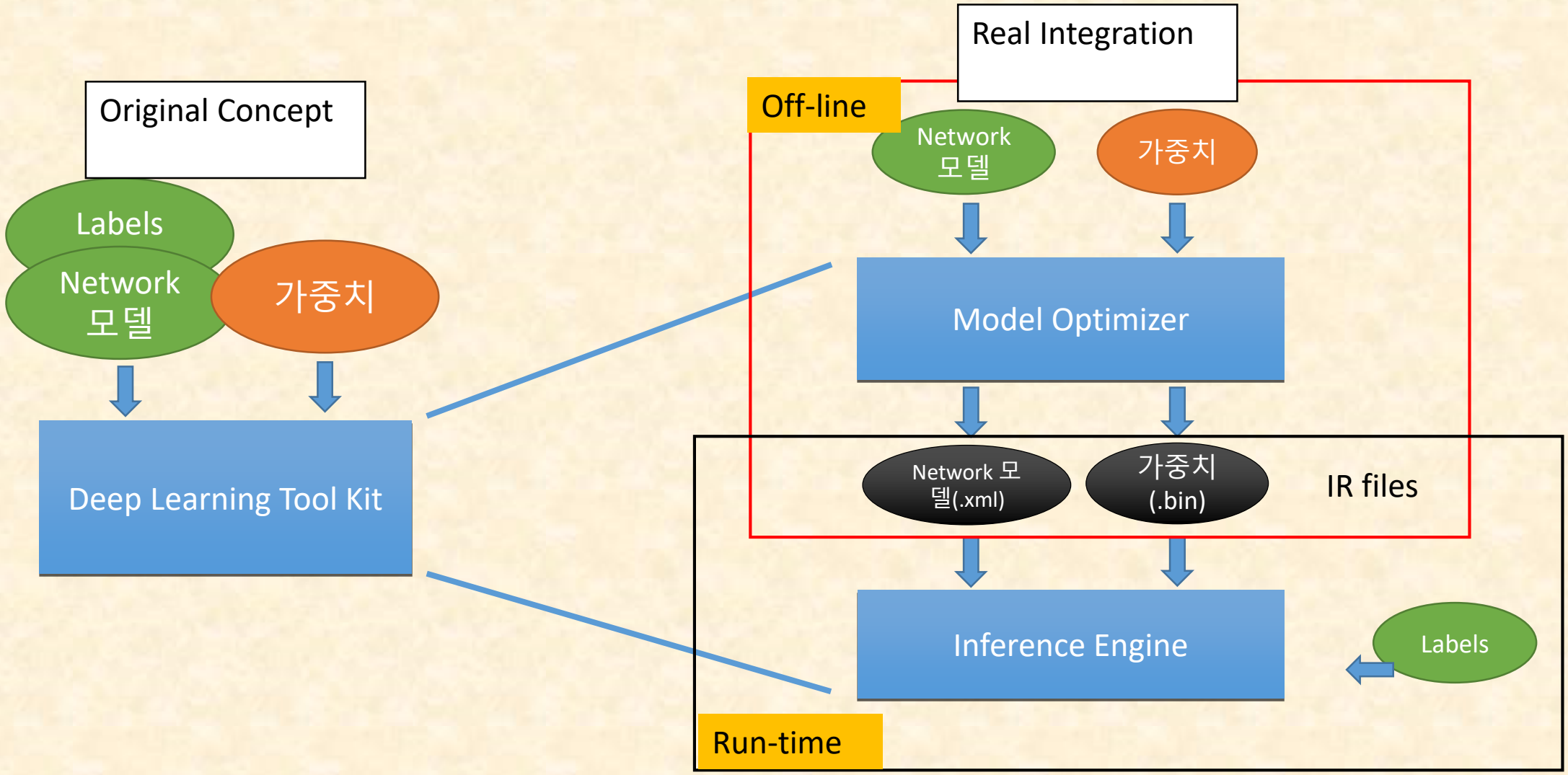
.caffemodel
.prototxt

Intermediate Representation

Deep Learning Framework으로 어떤 것을 선택하여 작업하더라도 OpenVINO를 사용하여 Inference 작업을 하기 위해서는 Inference Engine이 원하는 Format으로 바꿔줘야 합니다.

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

2. 모델 Conversion하기



OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

2. 모델 Conversion하기

/home/intel/my_model/ 의 위치에 Caffe 로 트레이닝을 해서 만들어진 모델 test.caffemodel과 test.prototxt가 있다.

```
$   conda deactivate
$   cd /opt/intel/openvino/deployment_tools/model_optimizer
$   python3 mo.py --input_model ~/my_model/test.caffemodel --output_dir ~/my_model
```

모델 전환이 성공적으로 이루어 졌다면
~/my_model 폴더에 test.xml, test.bin file들이 생성된다.

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

3. 입력 파일 위치 확인

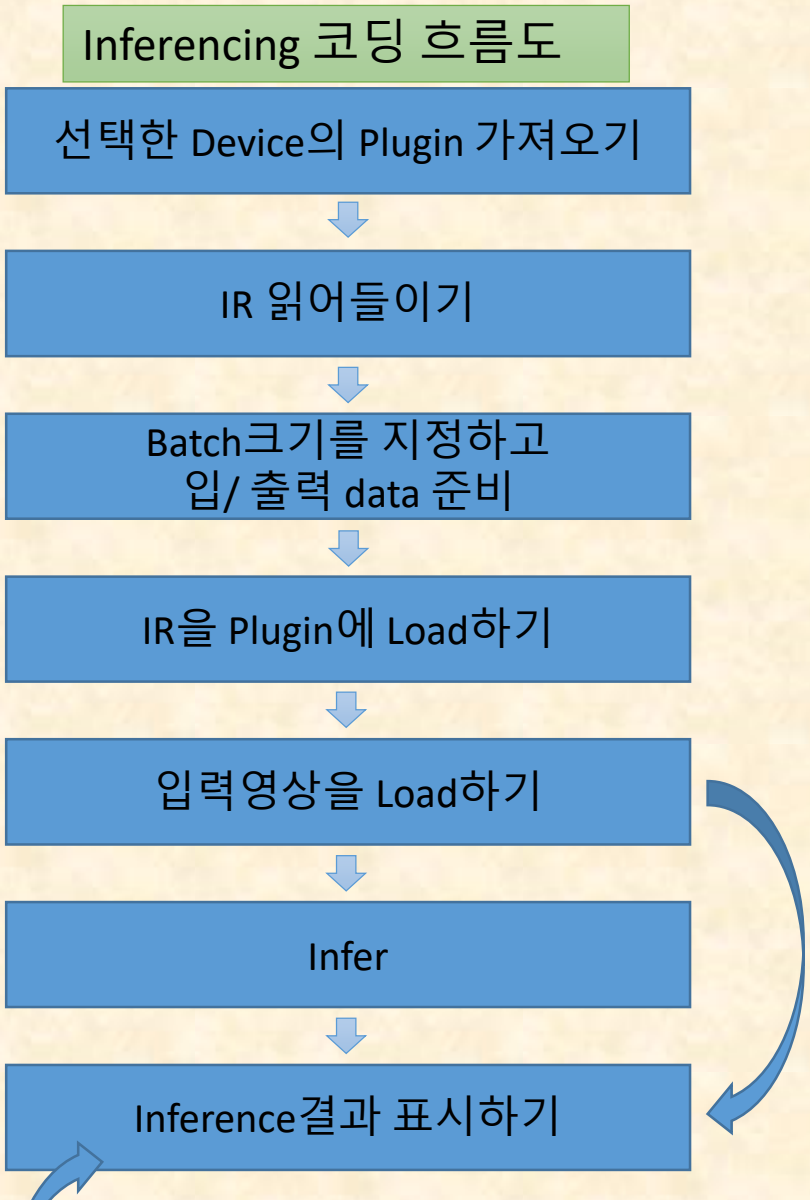
1. IR 파일 및 input image 파일 확인하기

```
$ cd /home/intel/my_model
$ ls
```

여기에 IR 파일과 test할 input image 파일이 있는지 확인 합니다.
(default: test.xml, test.bin, input.jpg)

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

4~7. 코딩 하기



1. /home/intel 에 “sample” 폴더 생성하고 이동하기

```
$ cd /home/intel
$ mkdir sample; cd sample
```

1. /home/intel/sample 에 “main.cpp” 파일 생성하기

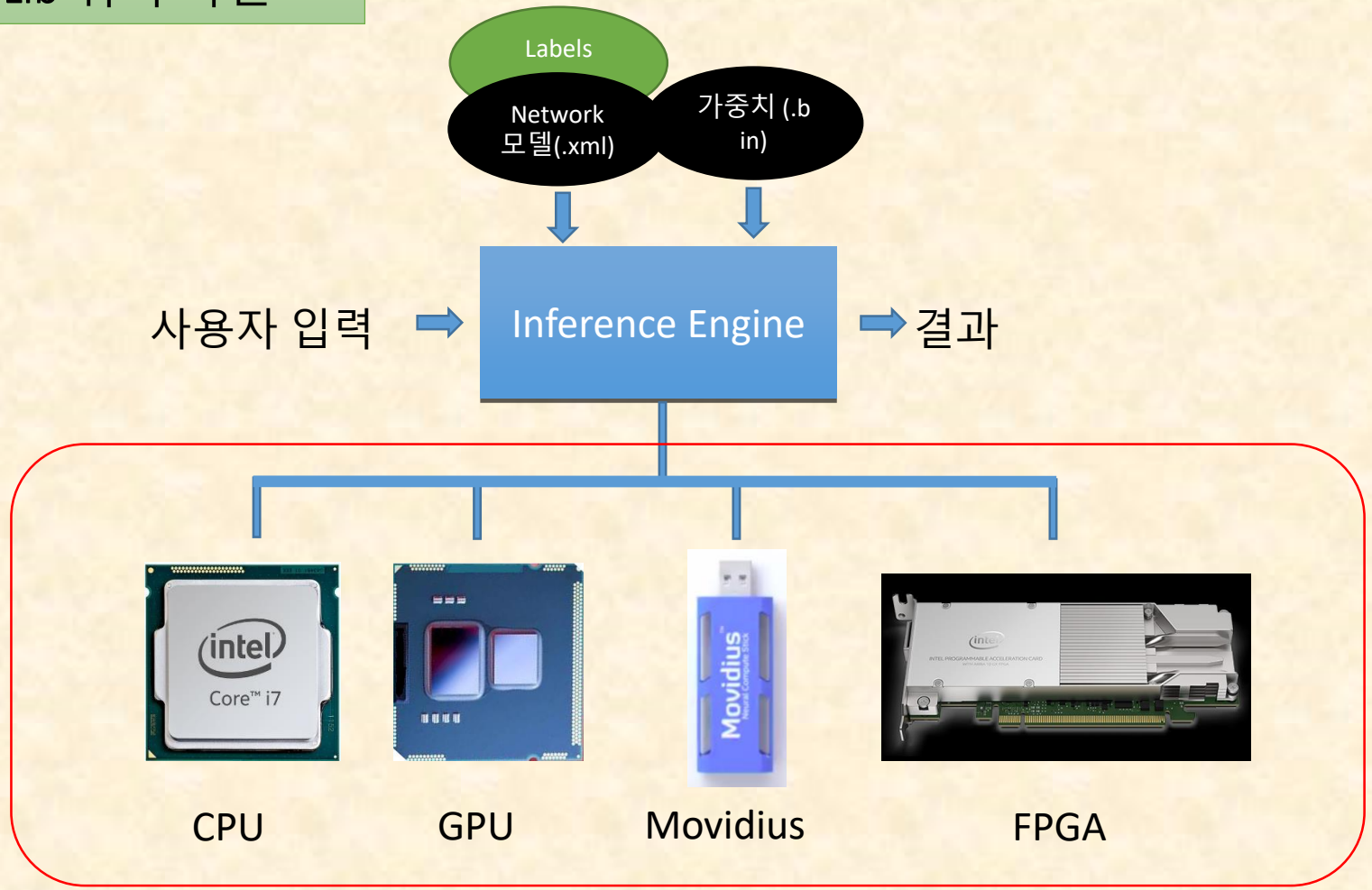
```
$ gedit main.cpp
```

1. main.cpp에 기본 main frame 입력하기

```
int main(int argc, char *argv[]) {

        // 코드를 여기에 입력합니다.
 //home/intel/sample_code/Inference.doc 참조

        return EXIT_SUCCESS;
}
```

OpenVINO의 Deep Learning Toolkit을 사용하여 S/W 구현하기

4. 사용할 HW 선택/ Lib 위치 확인




```
$ cd /opt/intel/computer_vision_sdk/deployment_tools/inference_engine/lib/
ubuntu_16.04/intel64
$ ls *.so
```

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

4~7. 코딩 설명(main.cpp안에 있는 내용 설명)

선택한 Device의 Plugin 가져오기

Plugin들이 위치한 폴더를 입력한다.
프로그램 시작전에 setupvars.sh를 실행하면 “”도 OK

```
// ---------------------Load A Plugin for Inference Engine-----------
InferenceEngine::PluginDispatcher dispatcher({""});
InferencePlugin plugin(dispatcher.getSuitablePlugin(TargetDevice::eCPU));
```

libMKLDNNPlugin.so 를 Loading한다.

CPU를 사용하여 Inference를 수행한다.

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

4~7. 코딩 설명(main.cpp안에 있는 내용 설명)

선택한 Device의 Plugin 가져오기

IR 읽어들이기

```
// ----Load IR Generated by ModelOptimizer (.xml and .bin files)----------

CNNNetReader network_reader;

network_reader.ReadNetwork("/home/intel/my_model/test.xml");
network_reader.ReadWeights("/home/intel/my_model/test.bin");
```

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

4~7. 코딩 설명(main.cpp안에 있는 내용 설명)

선택한 Device의 Plugin 가져오기

↓

IR 읽어들이기

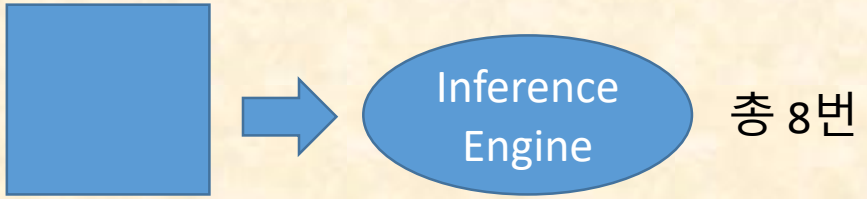
↓

Batch크기를 지정하고
입/ 출력 data 준비

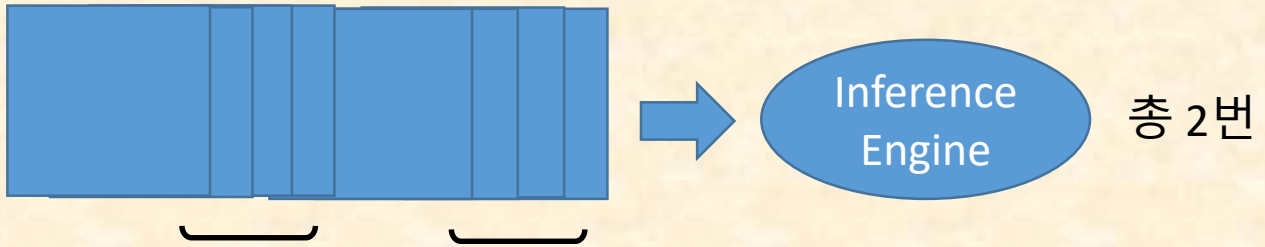
↓

만약,입력 data가 8장이라면

Batch Size = 1 인 경우,



Batch Size = 4 인 경우,




```
// ----------------Set batch size-----------------------------------
CNNNetwork network = network_reader.getNetwork();
network.setBatchSize(1);
// ----------------Prepare input blobs-----------------------------auto input_info = netw
ork.getInputsInfo().begin()->second;
auto input_name = network.getInputsInfo().begin()->first;
input_info->setPrecision(Precision::U8);
// ----------------Prepare output blobs----------------------------auto output_info = net
work.getOutputsInfo().begin()->second;
auto output_name = network.getOutputsInfo().begin()->first;
output_info->setPrecision(Precision::FP32);
```

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

4~7. 코딩 설명(main.cpp안에 있는 내용 설명)

선택한 Device의 Plugin 가져오기

↓

IR 읽어들이기

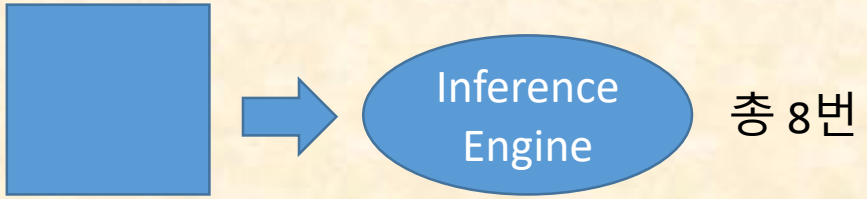
↓

Batch크기를 지정하고
입/ 출력 data 준비

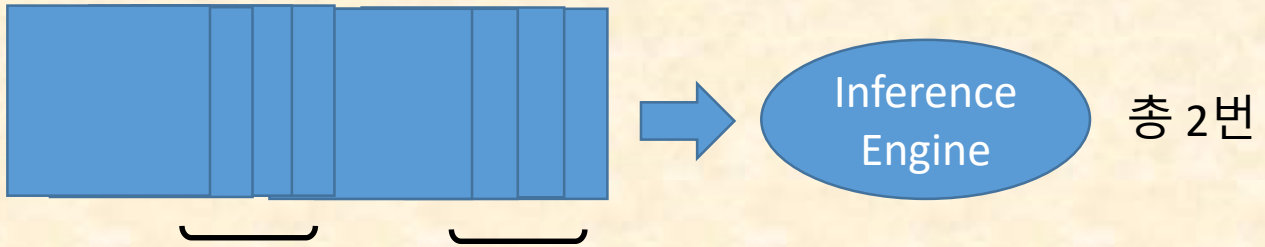
↓

만약, 입력 data가 8장이라면

Batch Size = 1 인 경우,



Batch Size = 4 인 경우,




```
// ----------------Set batch size-----------------------------------
CNNNetwork network = network_reader.getNetwork();
```

고려사항:

1) End 2 End의 지연 상황 : batch size가 커지면 Inference engine이 batch가 모두 끝날 때 까지 기다려야 하므로 연산처리 장치의 performance에 따라 지연 현상으로 인한 제품에 품질에 영향을 줄 수 있다.

2) 시스템 리소스 (메모리) : batch가 끝날 때 까지 memory를 hold하고 있어야 하므로 batch size가 커지면 system memory가 부족해 질 수 있고 이로 인해 system전체의 성능이 떨어질 수 있다.

결국, 제품의 목적과 사용 중인 시스템의 리소스를 고려하여 적절할 Batch size를 정해야 한다.

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

4~7. 코딩 설명(main.cpp안에 있는 내용 설명)

선택한 Device의 Plugin 가져오기

IR 읽어들이기

Batch크기를 지정하고
입/ 출력 data 준비

IR을 Plugin에 Load하기

```
// ---------Loading model to the plugin----------------------------
auto executable_network = plugin.LoadNetwork(network, {});
```

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

4~7. 코딩 설명(main.cpp안에 있는 내용 설명)

선택한 Device의 Plugin 가져오기

⇩

IR 읽어들이기

⇩

Batch크기 입/ 출력 data

⇩

IR을 Plugin에 Load하기

⇩

입력영상을 Load하기

⇩

Infer

```
auto infer_request = executable_network.CreateInferRequest();
auto input = infer_request.GetBlob(input_name);
auto input_data = input->buffer().as<PrecisionTrait<Precision::U8>::value_type*>();

/* Copying data from image to the input blob */
cv::Mat ori_image, infer_image;
ori_image = cv::imread("/home/intel/my_model/input.jpg");
cv::resize(ori_image, infer_image, cv::Size(input_info->getDims()[0], input_info->getDims()[1]
));

size_t channels_number = input->dims()[2];
size_t image_size = input->dims()[1] * input->dims()[0];

for (size_t pid = 0; pid < image_size; ++pid) {
    for (size_t ch = 0; ch < channels_number; ++ch) {
        input_data[ch * image_size + pid] = infer_image.at<cv::Vec3b>(pid)[ch];
    }
}
/* Running the request synchronously */
infer_request.Infer();
```

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

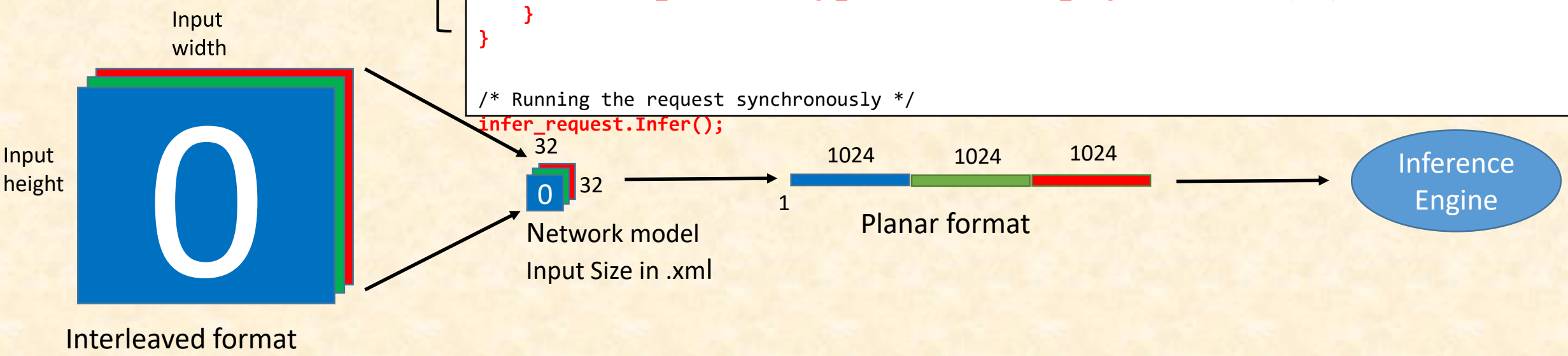
4~7. 코딩 설명(main.cpp안에 있는 내용 설명)

입력영상을 Load하기

```
auto infer_request = executable_network.CreateInferRequest();
auto input = infer_request.GetBlob(input_name);
auto input_data = input->buffer().as<PrecisionTrait<Precision::U8>::value_type*>();


/* Copying data from image to the input blob */
cv::Mat ori_image, infer_image;
ori_image = cv::imread("/home/intel/my_model/input.jpg");
cv::resize(ori_image, infer_image, cv::Size(input_info->getDims()[0], input_info->getDims()[1]));


size_t channels_number = input->dims()[2];
size_t image_size = input->dims()[1] * input->dims()[0];


for (size_t pid = 0; pid < image_size; ++pid) {
    for (size_t ch = 0; ch < channels_number; ++ch) {
            input_data[ch * image_size + pid] = infer_image.at<cv::Vec3b>(pid)[ch];
    }
}


/* Running the request synchronously */
infer_request.Infer();
```

1. Resizing

2. Color format conversion



OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

3~7. 코딩 설명(main.cpp안에 있는 내용 설명)

선택한 Device의 Plugin 가져오기

⇩

IR 읽어들이기

⇩

Batch크기를 지정하고 입/출력 data 준비

⇩

IR을 Plugin에 Load하기

⇩

입력영상을 Load하기

⇩

Infer

⇩

Inference결과 표시하기

```
// ----------Postprocess output blobs----------------------
auto output = infer_request.GetBlob(output_name);
auto output_data = output->buffer().as<PrecisionTrait<Precision::FP32>::v
alue_type*>();

/* Sort output probabilities and put it to results vector */
std::vector<unsigned> results;
TopResults(10, *output, results);

std::cout << std::endl << "Top 10 results:" << std::endl << std::endl;

for (size_t id = 0; id < 10; ++id) {
    std::cout.precision(7);
    auto result = output_data[results[id]];
    std::cout << std::left << std::fixed << result << " label #" << resul
ts[id] << std::endl;
}
```

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

3~7. 코딩 설명(코딩 마무리하고 컴파일 하기)

1. main.cpp에 header 와 InferenceEngine namespace 추가하기

```
#include <vector>
#include <memory>
#include <string>
#include <opencv2/opencv.hpp>
#include <inference_engine.hpp>
using namespace InferenceEngine;
```

2. **컴파일하기**

```
$   cd ~/sample
$   g++ -o sample1 -std=c++11 main_test.cpp -I$INTEL_CVSDK_DIR/opencv/include -I$INTE
L_CVSDK_DIR/deployment_tools/inference_engine/include -L$IE_PLUGINS_PATH -L$INTEL
_CVSDK_DIR/opencv/lib -ldl -lopencv_core -lopencv_imgproc -lopencv_imgcodecs -linferen
ce_engine -lopencv_video -lopencv_highgui -lopencv_videoio
```

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

8. 실행하기

1. 실행 및 결과

```
$  ./sample1
```

```
Top 10 results:
1.0000000 label #0
0.0000000 label #4
0.0000000 label #8
0.0000000 label #9
0.0000000 label #7
0.0000000 label #3
0.0000000 label #1
0.0000000 label #5
0.0000000 label #2
0.0000000 label #6
intel@nuc:~/sample$ _
```

Network 모델과 가중치 정보로 입력 영상에 대한 추론 결과 10개를 보여 준다.
확률 값이며 모두 합치면 1이 된다.
가장 높은 확률을 가지는 Label을 선택하면 된다.
여기서는 “0”.

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

8. 실행 (영상확인 하기)

1. Resize된 영상 확인해 보기 – main.cpp를 수정합니다.

```
cv::resize(ori_image, infer_image, cv::Size(input_info->getDims()
[0], input_info->getDims()[1]));

cv::namedWindow("resized_image", cv::WINDOW_NORMAL);
cv::resizeWindow("resized_image", 600,600);
cv::imshow("resized_image", infer_image);
cv::waitKey(0);
```

1. 컴파일하기

```
intel@nuc:~/sample$ intel@nuc:~/sample$ g++ -o sample1 -std=c++11 main.cpp -I$INT
EL_CVSDK_DIR/opencv/include -I$INTEL_CVSDK_DIR/deployment_tools/inference_engine/
include -L$IE_PLUGINS_PATH -L$INTEL_CVSDK_DIR/opencv/lib -ldl -lopencv_core -lope
ncv_imgproc -lopencv_imgcodecs -linference_engine -lopencv_video -lopencv_highgui
-lopencv_videoio
```

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

1. CPU 대신 Movidius 을 사용하기 위해서는 FP16 으로 컨버젼 해야합니다.

1. **Movidius는 FP 16bit 모델을 요구** 하므로 test.caffemodel 을 mo.py 로 FPS 16bit용 IR 파일들을 생성 하셔야 합니다.

```
$ cd /opt/intel/openvino/deployment_tools/model_optimizer
$ python3 mo.py --input_model ~/my_model/test.caffemodel --output_dir ~/my_model --data_type FP
16
```

그러면 ~/my_model/ 에 test.xml/bin 파일이 생성되면 이건 FP16 bit version 입니다.
data_type을 지정하지 않으면 FP32 bit version이 기본으로 생성됩니다.

2. Plugin 을 **"eMYRIAD" 로 변경합니다**. (GPU 를 사용하기 위해서는 "eGPU" 로 변경)

```
InferencePlugin plugin(dispatcher.getSuitablePlugin(TargetDevice::eCPU));
➔
InferencePlugin plugin(dispatcher.getSuitablePlugin(TargetDevice::eMYRIAD));
```

OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

2. 코드 작성

```
$ cd /home/intel/
$ mkdir sample
$ cd sample/
$ gedit main.cpp
```

2. 컴파일)

```
$    cd ~/sample/
$    g++ -o sample1 -std=c++11 main.cpp -I$INTEL_CVSDK_DIR/opencv/include -I$INTEL
_CVSDK_DIR/deployment_tools/inference_engine/include -L$IE_PLUGINS_PATH -L$INT
EL_CVSDK_DIR/opencv/lib -ldl -lopencv_core -lopencv_imgproc -lopencv_imgcodecs -li
nference_engine -lopencv_video -lopencv_highgui -lopencv_videoio
```

3. 실행 (FP16)

```
$    ./sample1
```

SSD300 Pre-trained model 사용하기

Caffe 에서 Pre-trained된 model SSD300 을 이용하여 OpenVION에서 추론 하기

-. Model conversion

-. model_ssd.ccp만들기

-. 컴파일 하기

-. ./sampel1 실행

OpenVINO 을 이용한 추론 – SSD300 모델

Object Detection 코딩 하기 – 대부분의 이전 Image Classification code를 사용한다.

1. Model: SSD300 caffemodel from model_download – 이미 되어 있습니다.

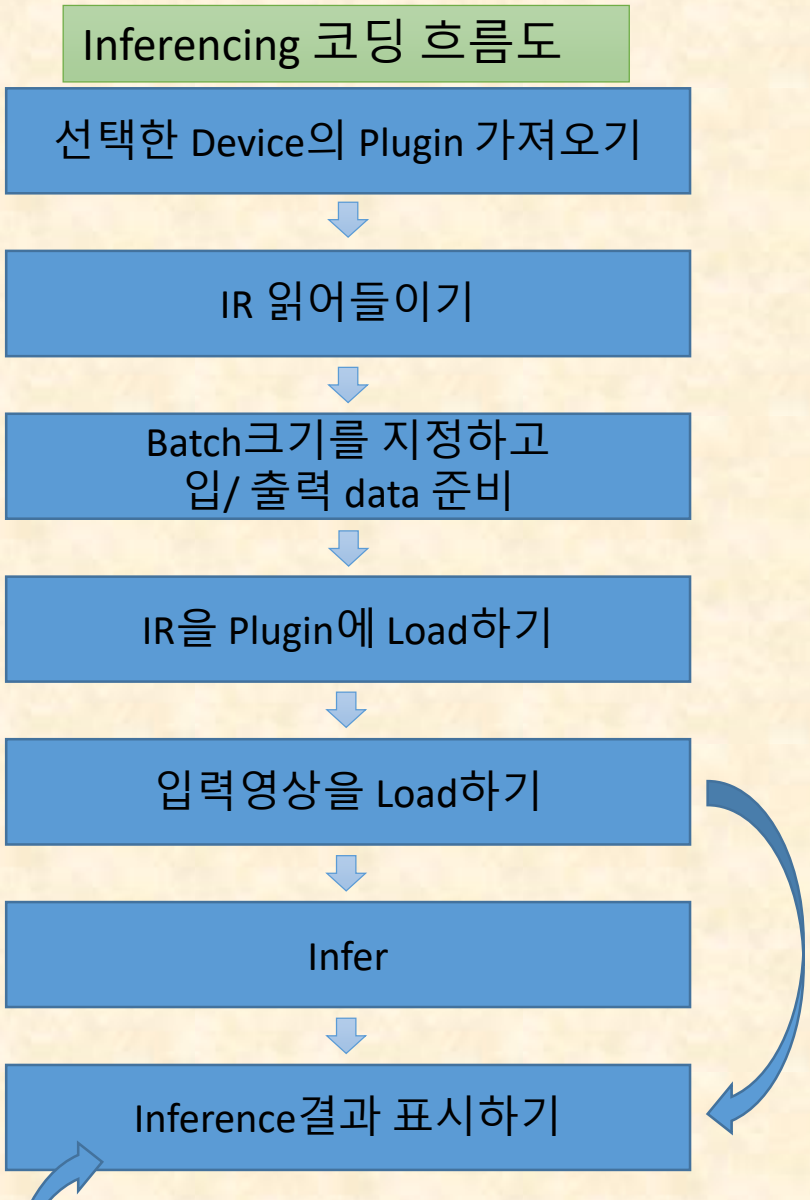
```
$ cd /opt/intel/computer_vision_sdk/deployment_tools/model_downloader
$ sudo python3 ./downloader.py
$ cp ./ssd300* ~/my_model
```

2. 모델 Conversion하기

```
$ cd /opt/intel/openvino/deployment_tools/model_optimizer
$ python3 mo.py --input_model ~/my_model/ssd300.caffemodel --output_dir ~/my_model
--data_type FP16
```

OpenVINO 을 이용한 추론 – SSD300 모델

3. 코딩 하기



1. /home/intel 에 “sample” 폴더 생성하고 이동하기

```
$ cd /home/intel
$ mkdir sample; cd sample
```

2. /home/intel/sample 에 “main.cpp” 파일 생성하기

```
$ gedit main_ssd.cpp
```

3. main.cpp에 기본 main frame 입력하기

```
int main(int argc, char *argv[]) {

        // 코드를 여기에 입력합니다.
  //home/intel/sample_code/Inference.doc 참조


        return EXIT_SUCCESS;
}
```

4.코드를 입력후 저장하고 종료하고 콘솔로 이동.

OpenVINO 을 이용한 추론 – SSD300 모델

3. 코딩 하기 | 1. CPU Extension library 지정하기

```
// -----Load A Plugin for Inference Engine-------------------------
InferenceEngine::PluginDispatcher dispatcher({""});
InferencePlugin plugin(dispatcher.getSuitablePlugin(TargetDevice::eCPU));

// ++++++++++++++++++++ Load layer extension for CPU ++++++++++++++++++++
auto extension_ptr = make_so_pointer<InferenceEngine::IExtension>("/opt/intel/computer_vision_sdk
/deployment_tools/inference_engine/lib/ubuntu_16.04/intel64/libcpu_extension.so");
plugin.AddExtension(extension_ptr);
// ----Load IR Generated by ModelOptimizer (.xml and .bin files)------
CNNNetReader network_reader;
```

Inference engine이 CPU인 경우 (mklDNN), libmklDNN.so 는 기본적인 layer들만 지원하고 사용자에 의해 추가되고 변형된 layer는 지원하지 않는다.
그래서 그런 확장된 layer를 지원하는 모듈을 따로 제작해야 하며 그 모듈을 지정해 줘야 합니다. “libcpu_extension.so” 는 Inference engine sample을 빌드하면 생성되며 기본적인 방법을 보여줍니다.
여기서는 SSD의 “detection_out” layer를 지원합니다.

OpenVINO 을 이용한 추론 – SSD300 모델

3 코딩 하기

2 SSD 모델 읽어들이기 (IR)

```
// -----Load IR Generated by ModelOptimizer (.xml and .bin files)--------

CNNNetReader network_reader;

network_reader.ReadNetwork("/home/intel/my_model/ssd300.xml");
network_reader.ReadWeights("/home/intel/my_model/ssd300.bin");
```

3. 새로운 이미지 읽어들이기

```
cv::Mat ori_image, infer_image;
ori_image = cv::imread("/home/intel/my_model/cars_768x768.jpg"); //jpg
VideoCapture cap("/dev/video0");   //camera
```

OpenVINO 을 이용한 추론 – SSD300 모델

3. 코딩 하기 | 4. Inference 출력 데이타 처리하기

```
infer_request.Infer();
auto output = infer_request.GetBlob(output_name);
// ++++++++++++ check proposal count and objectsize of each proposal ++++++++++
const int maxProposalCount = output->dims()[1];
const int objectSize = output->dims()[0];
const Blob::Ptr output_blob = output;
const float* detection = static_cast<PrecisionTrait<Precision::FP32>::value_type*>(output_blob->buffer());
/* Each detection has image_id that denotes processed image */
for (int curProposal = 0; curProposal < maxProposalCount; curProposal++) {
    float image_id = detection[curProposal * objectSize + 0];
    float label = detection[curProposal * objectSize + 1];
    float confidence = detection[curProposal * objectSize + 2];

    if (image_id < 0 || confidence == 0)
        continue;

    float xmin = detection[curProposal * objectSize + 3] * ori_image.size().width;
    float ymin = detection[curProposal * objectSize + 4] * ori_image.size().height;
    float xmax = detection[curProposal * objectSize + 5] * ori_image.size().width;
    float ymax = detection[curProposal * objectSize + 6] * ori_image.size().height;

    cout << "[" << curProposal << "," << label << "] element, prob = " << confidence << " (" << xmin << ","
<< ymin << ")-(" << xmax << "," << ymax << ")" << " batch id : " << image_id;
```

OpenVINO 을 이용한 추론 – SSD300 모델

3. 코딩 하 header 와 InferenceEngine namespace 추가하기

```
#include <vector>
#include <memory>
#include <string>
#include <opencv2/opencv.hpp>
#include <inference_engine.hpp>
using namespace InferenceEngine;
```

4. 컴파일 하기

```
$    cd ~/sample
$    g++ -o sample1 -std=c++11 main_ssd.cpp -I /opt/intel/openvino/opencv/include -I
/opt/intel/openvino/deployment_tools/inference_engine/include -L$IE_PLUGINS_PATH -L
/opt/intel/openvino/opencv/lib -ldl -lopencv_core -lopencv_imgproc -lopencv_imgcodecs
-linference_engine -lopencv_video -lopencv_highgui -lopencv_videoio
```

5. 실행 (SSD300)

```
$    ./sample1
```

Mobilenet-SSD Pre-trained model 사용하기

Caffe 에서 Pre-trained된 model mobilenet-ssd 을 이용하여 OpenVION에서 추론 하기

-. Model conversion

-. model_mobilenet.ccp만들기

-. 컴파일 하기

-. ./sampel1 실행

OpenVINO 을 이용한 추론 – Mobilenet-SSD 모델

1. Mobilenet-ssd 모델 Conversion하기

```
$    cd /opt/intel/openvino/deployment_tools/model_optimizer
$    python3 mo.py --framework caffe --input_model ~/my_model/mobilenet-ssd.caffemodel
--output_dir ~/my_model --data_type FP16
```

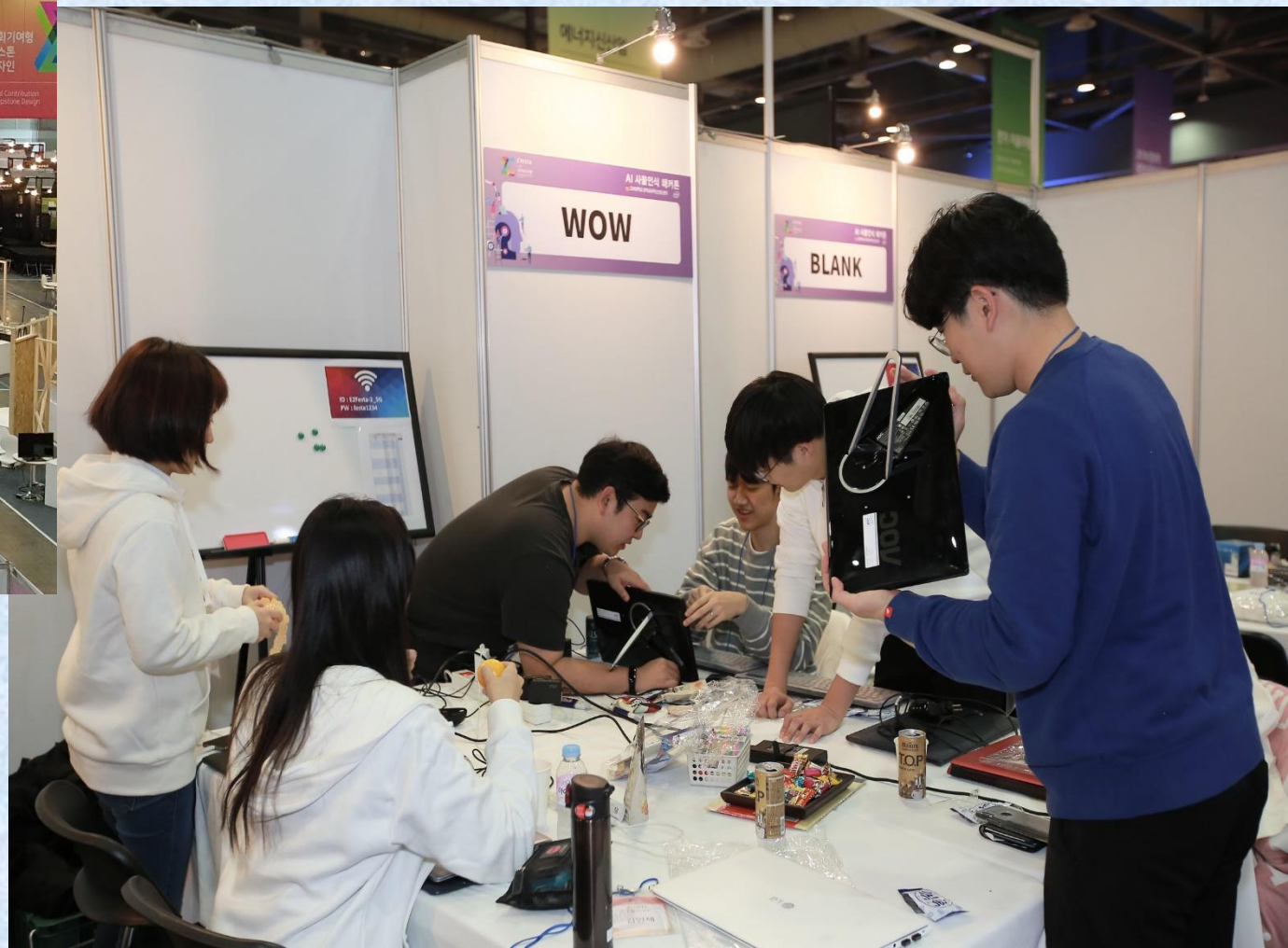
2. 컴파일(mobilenet)

```
$    cd ~/sample/
$    g++ -o sample1 -std=c++11 main_mobilenet.cpp -I /opt/intel/openvino/opencv/include
-I /opt/intel/openvino/deployment_tools/inference_engine/include -L$IE_PLUGINS_PATH -
L /opt/intel/openvino/opencv/lib -ldl -lopencv_core -lopencv_imgproc -lopencv_imgcodecs
-linference_engine -lopencv_video -lopencv_highgui -lopencv_videoio
```

3. 실행 (mobilenet)

```
$    ./sample1
```


IoT
IoT Maker
AI 사물인식 해커톤
AI Hackathon
드론축구대회
Drone Soccer League
에너지산업별
사회기여형
캡스톤
디자인


WOW
BLANK

OpenVINO 을 이용한 추론 – App 으로 연결

1. Mobilenet-ssd 모델을 이용한 시나리오 구현 사례

```
$ cd /opt/intel/openvino/deployment_tools/model_optimizer
$ python3 mo.py --framework caffe --input_model ~/my_model/mobilenet-ssd.caffemodel
--output_dir ~/my_model --data_type FP16
```

2. 컴파일(mobilenet)

```
$    cd ~/sample/
$    g++ -o sample2 -std=c++11 main_mobilenet_wori.cpp -I$INTEL_CVSDK_DIR/opencv/
include -I$INTEL_CVSDK_DIR/deployment_tools/inference_engine/include -L$IE_PLUGIN
S_PATH -L$INTEL_CVSDK_DIR/opencv/lib -ldl -lopencv_core -lopencv_imgproc -lopencv
_imgcodecs -linference_engine -lopencv_video -lopencv_highgui -lopencv_videoio -lX11
```

3. 실행 (mobilenet)

```
$    ./sample2
```

CIFAR-10 으로 트레이닝한 model 사용하기

Caffe CifarNet 으로 training 한 model cifar.caffemodel 을 이용하여 OpenVION에서 추론 하기

-. Model conversion

-. model_cifar.ccp만들기

-. 컴파일 하기

-. ./sampel1 실행

OpenVINO 을 이용한 추론 – CIFAR-10 모델

1. File copy 준비 작업

```
$    cd  ~/sample_code/opt-intel-computer_vision_sdk-deployment_tools-model_optimizer-mo-front-caffe-proto/
$    sudo cp ./caffe_pb2.py /opt/intel/openvino/deployment_tools/model_optimizer/mo/front/caffe/proto/
```

2. Cifar10 모델 Conversion하기

```
$    cd /opt/intel/openvino/deployment_tools/model_optimizer
$    python3 mo.py --framework caffe --input_model ~/my_model/cifar.caffemodel --output_dir ~/my_model
```

3. 컴파일(Cifar10)

```
$    cd ~/sample
$    g++ -o sample1 -std=c++11 main_cifar.cpp -I /opt/intel/openvino/opencv/include -I /opt/intel/openvino/deployment_tools/inference_engine/include -L$IE_PLUGINS_PATH -L /opt/intel/openvino/opencv/lib -ldl -lopencv_core -lopencv_imgproc -lopencv_imgcodecs -linference_engine -lopencv_video -lopencv_highgui -lopencv_videoio
```

OpenVINO 을 이용한 추론 – CIFAR-10 모델

4. 실행 (Cifar10)

```
$    ./sample1
```

5. 컴파일(Cifar10- Video input)

```
$    cd /sample
$    g++ -o sample1 -std=c++11 main_cifar_camera.cpp -I$INTEL_CVSDK_DIR/opencv/include -I$INTEL_CVSDK_DIR/deployment_tools/inference_engine/include -L$IE_PLUGINS_PATH -L$INTEL_CVSDK_DIR/opencv/lib -ldl -lopencv_core -lopencv_imgproc -lopencv_imgcodecs -linference_engine -lopencv_video -lopencv_highgui -lopencv_videoio -lX11
```

6. 실행 (Cifar10- Video input)

```
$    ./sample1
```

7. File copy 복원

```
$    cd ~/sample_code/r1_caffe_pb2
$    sudo cp ./caffe_pb2.py /opt/intel/openvino/deployment_tools/model_optimizer/mo/front/caffe/proto/
```

수고 하셨습니다!