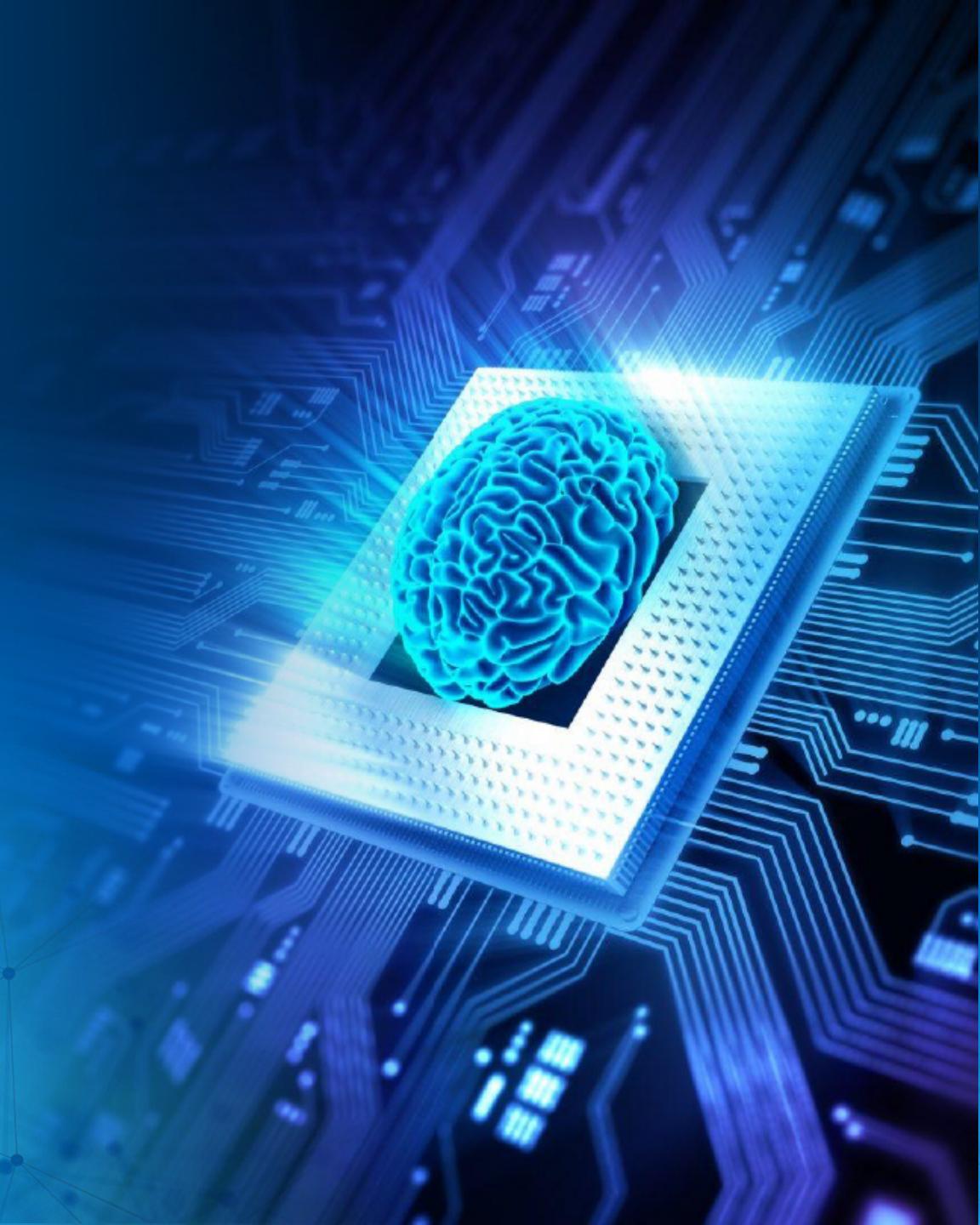




# ARTIFICIAL INTELLIGENCE



# 파이썬 특징

- “Hello World” 를 출력하는 문법 비교

## Java Language

```
public class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

## C Language

```
#include <stdio.h>
int main()
{
    printf("Hello World");
    return 0;
}
```

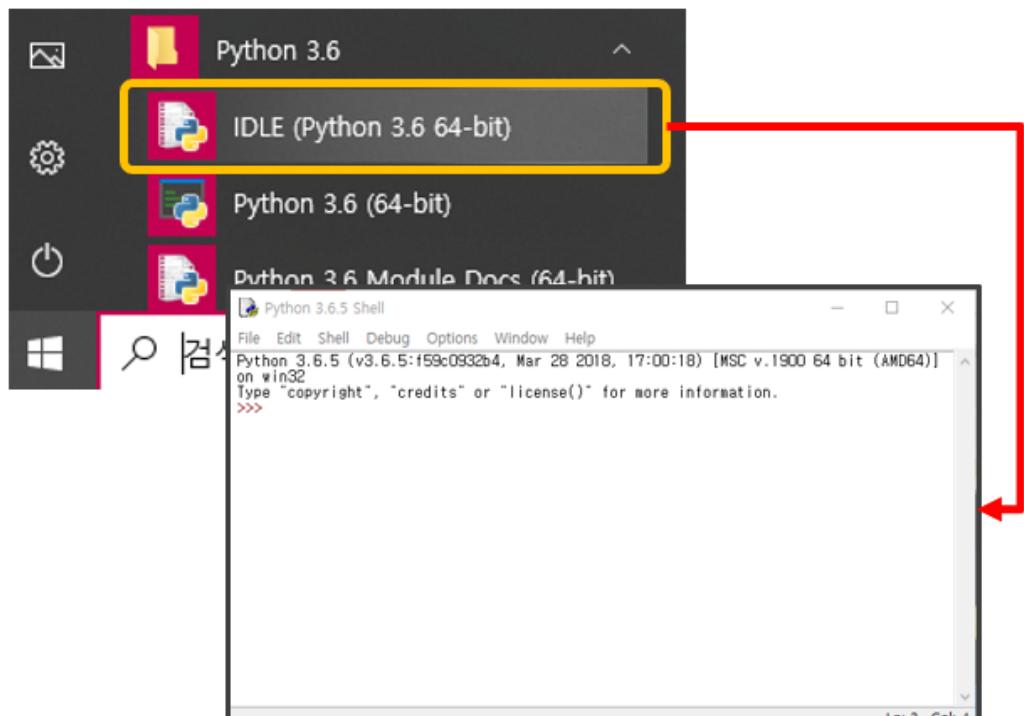
## Python Language

```
print("Hello World")
```

# 파이썬 개발 환경

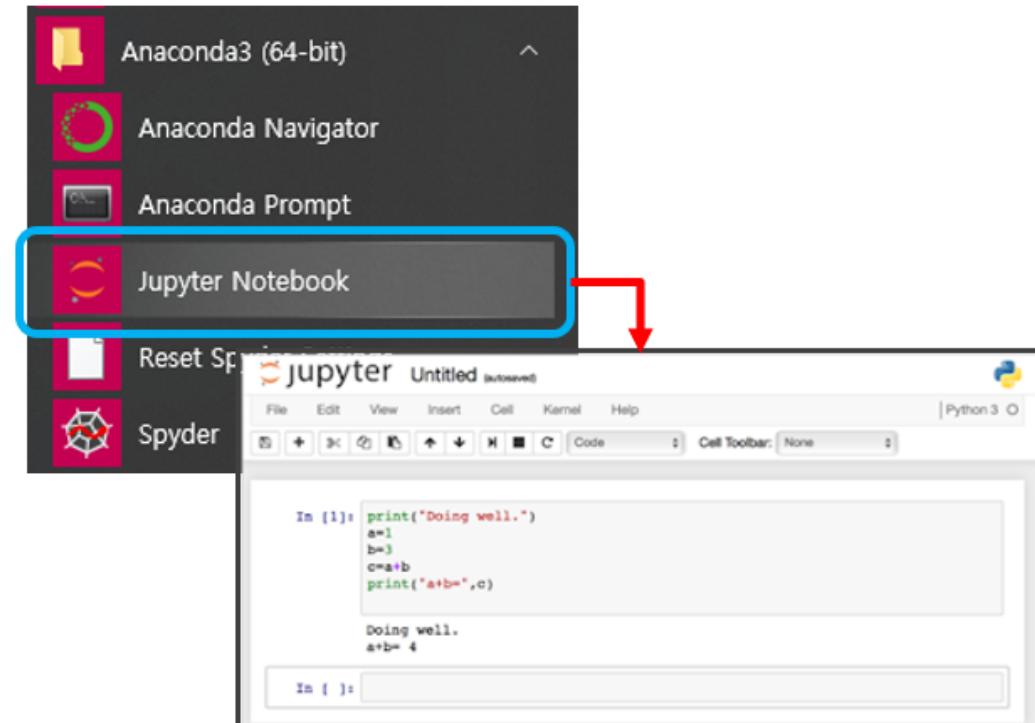
## <Python IDLE>

- Python 을 설치하면 제공되는 에디터임
- 가볍고 사용이 간단하나 디버깅이나 파일형태 작업하는데 불편함



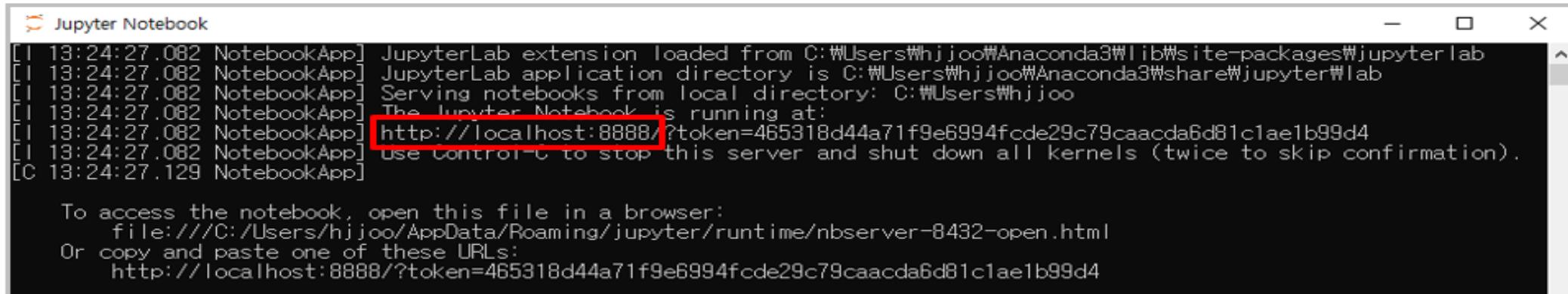
## Jupyter notebook

- 웹상에서 에디팅과 실행이 가능한 우수한 에디터
- 아나콘다를 설치하면 같이 설치됨



# 주피터 노트북 실행(1)

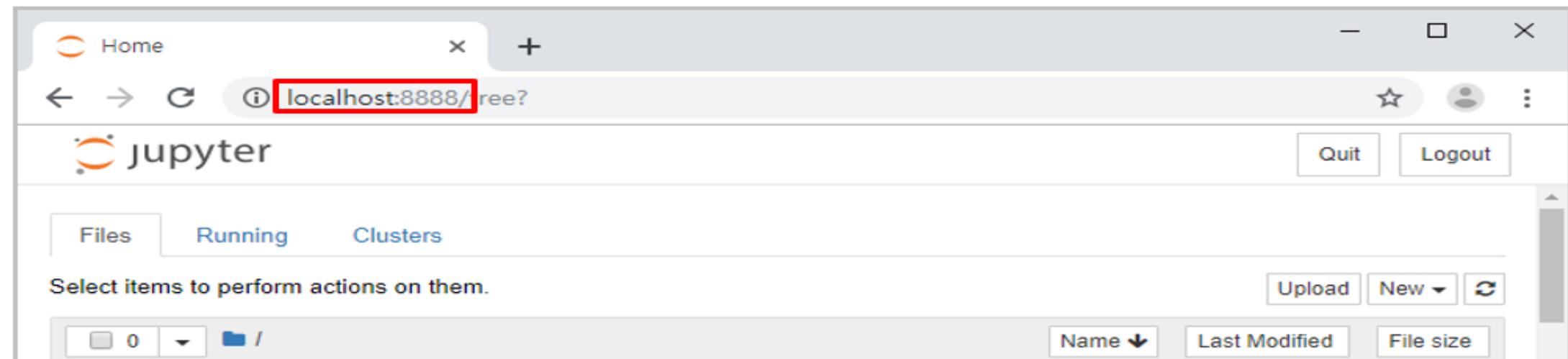
## 1 주피터 노트북 콘솔



```
Jupyter Notebook
[| 13:24:27.082 NotebookApp] JupyterLab extension loaded from C:\Users\hj.joo\Anaconda3\lib\site-packages\jupyterlab
[| 13:24:27.082 NotebookApp] JupyterLab application directory is C:\Users\hj.joo\Anaconda3\share\jupyter\lab
[| 13:24:27.082 NotebookApp] Serving notebooks from local directory: C:\Users\hj.joo
[| 13:24:27.082 NotebookApp] The Jupyter Notebook is running at:
[| 13:24:27.082 NotebookApp] http://localhost:8888/?token=465318d44a71f9e6994fcde29c79caacda6d81c1ae1b99d4
[| 13:24:27.082 NotebookApp] use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:24:27.129 NotebookApp]

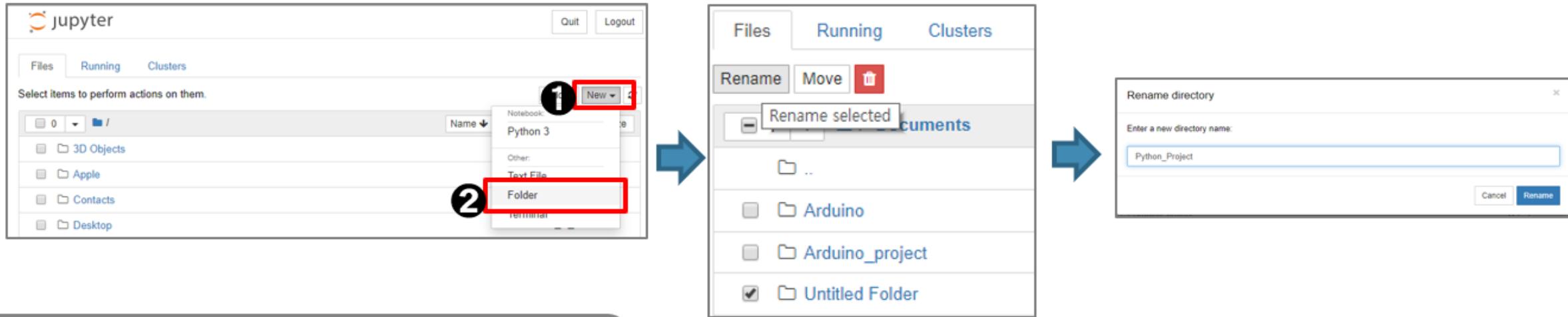
To access the notebook, open this file in a browser:
  file:///C:/Users/hj.joo/AppData/Roaming/jupyter/runtime/nbserver-8432-open.html
Or copy and paste one of these URLs:
  http://localhost:8888/?token=465318d44a71f9e6994fcde29c79caacda6d81c1ae1b99d4
```

## 2 주피터노트북 실행

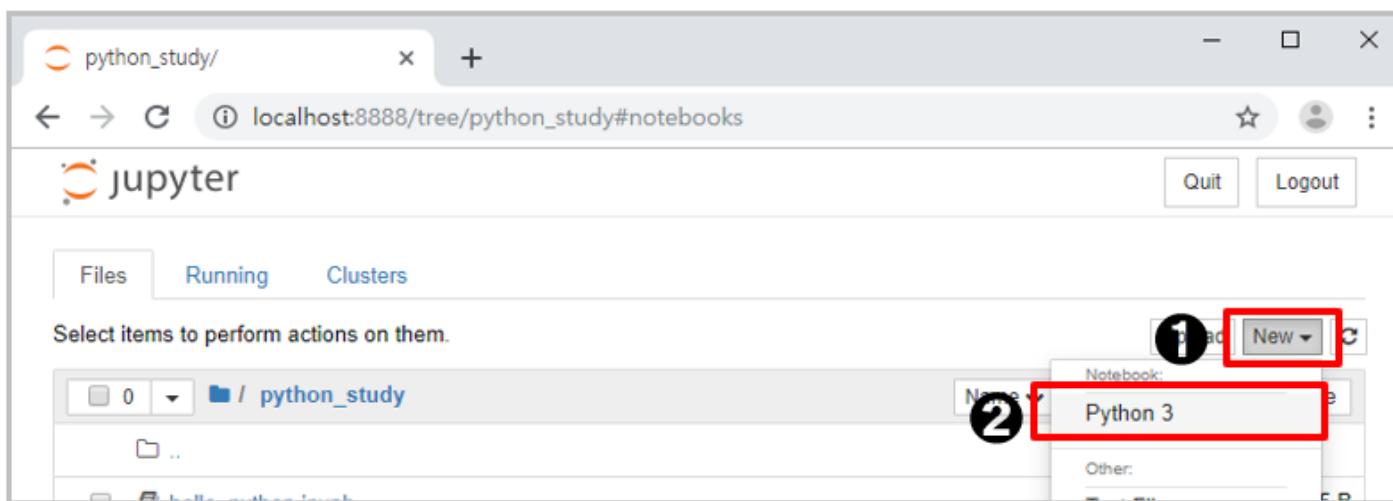


# 주피터 노트북 실행(2)

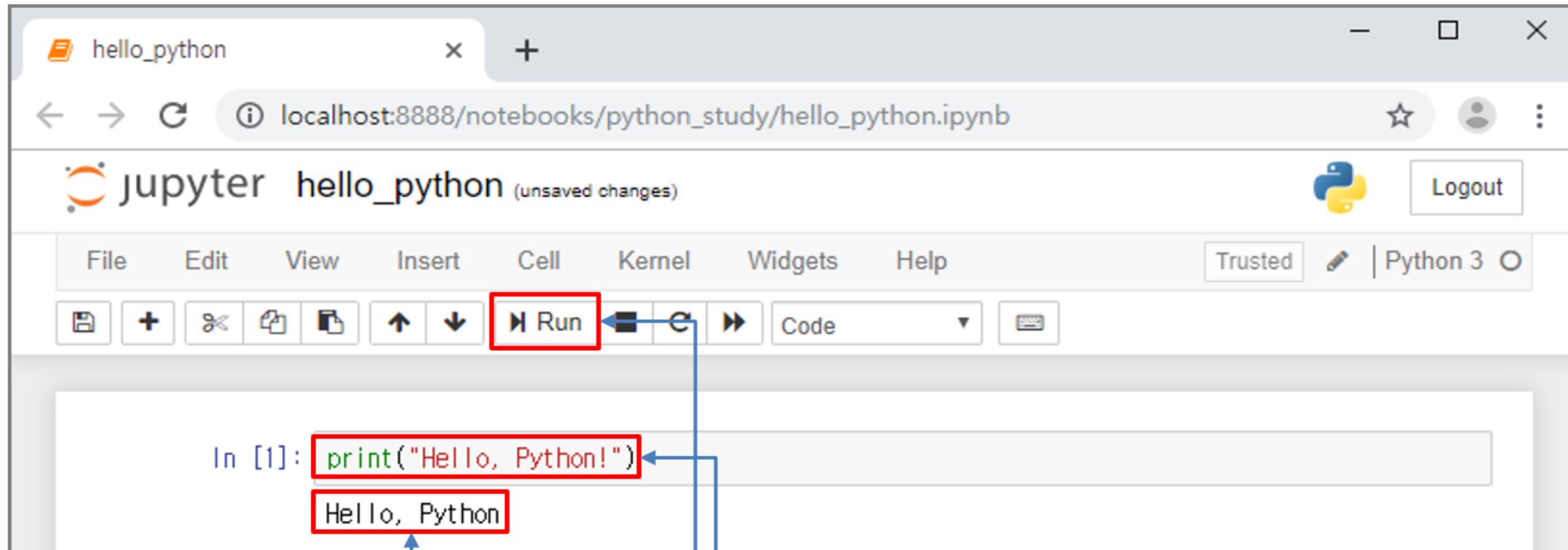
## ③ 작업할 디렉토리 생성



## ④ 파이썬 파일 생성



# 주피터 노트북 사용하기(1)

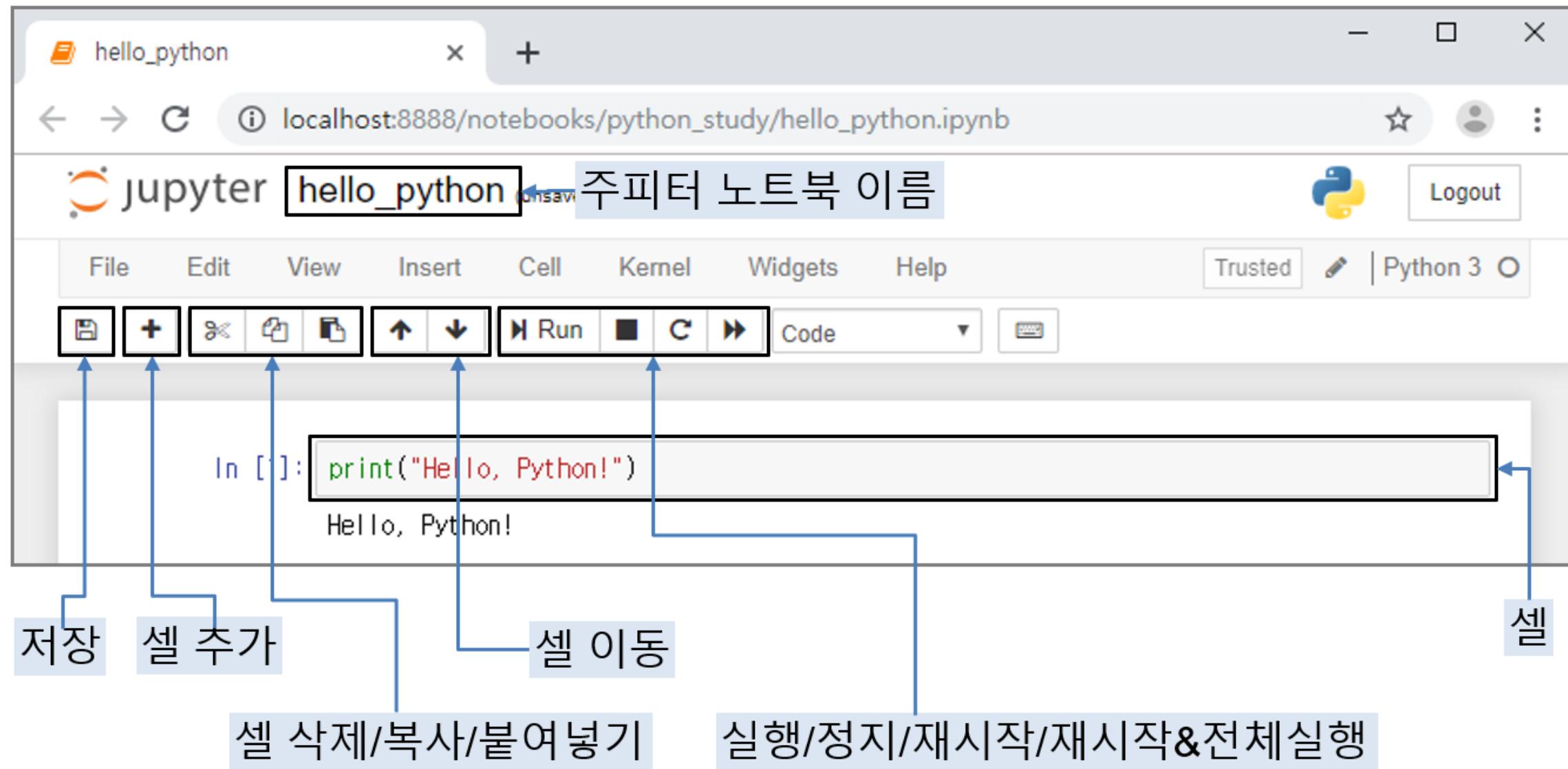


① 코딩

② [Run] 클릭 혹은 [Shift+Enter]

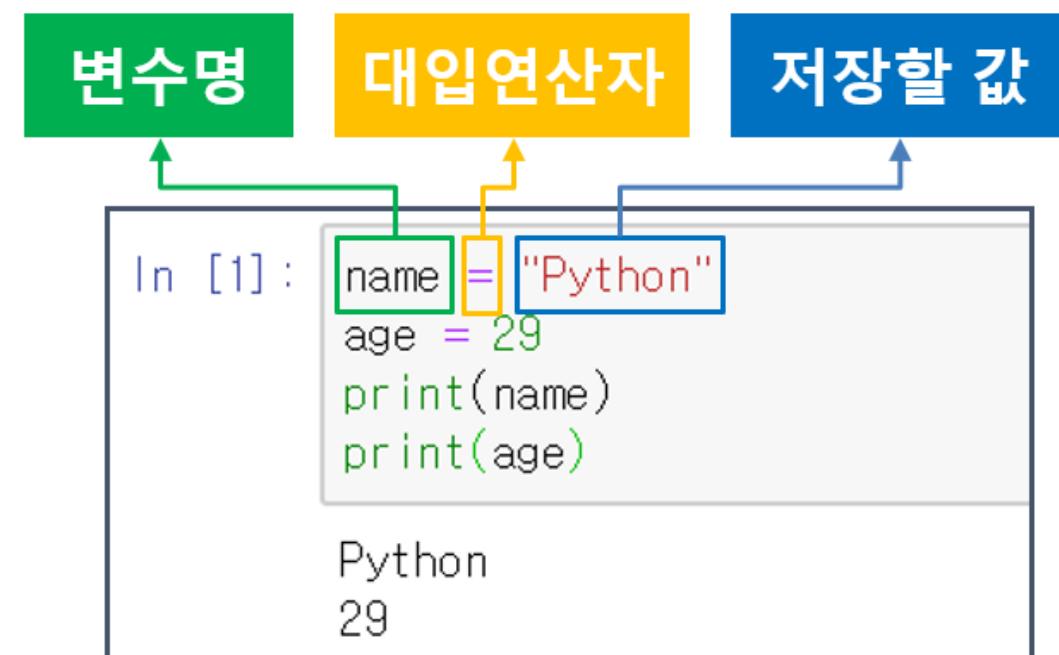
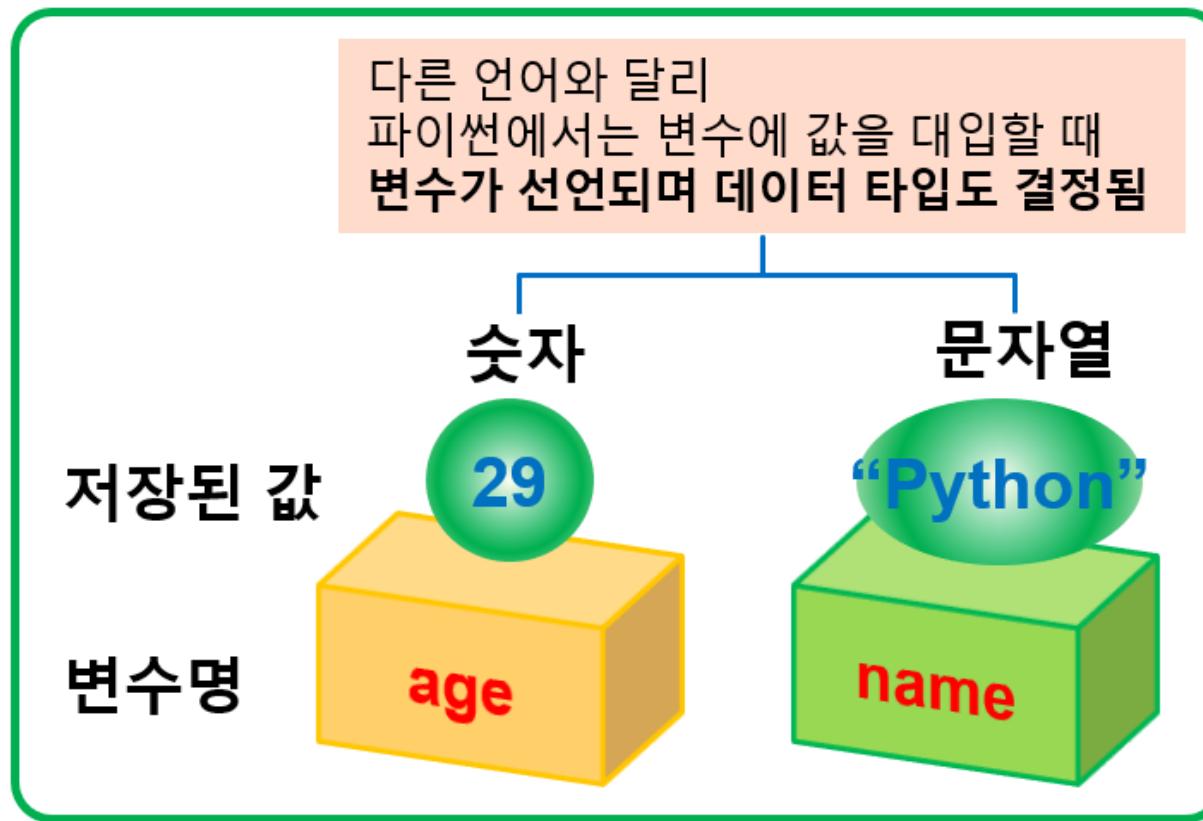
③ 실행결과

# 주피터 노트북 사용하기(2)



# 파이썬 기초 - 변수

- 프로그램을 실행하는데 필요한 정보를 저장하는 공간
- 필요에 따라 저장된 값의 변경이 가능
- 숫자, 문자열 저장 (문자열 지정 시에는 따옴표(' or " )로 구분)



# 파이썬 기초 - 변수

- 숫자 맞추기 게임 : 컴퓨터의 random 기능을 사용하여 1부터 50사이의 정수 하나를 임의로 선택하게 하고 사용자가 컴퓨터가 선택한 수를 맞추는 프로그램

모듈 : 파이썬 정의와 문장들을 담고 있는 파일(random.py)로 import 하여 사용함

```
1 import random
2
3 randomNum = random.randint(1,50)
4
5 guess = int(input('숫자를 맞춰보세요(1~50): '))
6 count = 1
7
8 while guess != randomNum:
9     if guess > randomNum:
10         print('더 작은 수를 선택하세요')
11     else:
12         print('더 큰 수를 선택하세요')
13     guess = int(input('숫자를 맞춰보세요(1~50): '))
14     count += 1
15
16 print(count, '번 만에 맞혔습니다. 입력한 숫자는', randomNum, '입니다')
```

함수 : 프로그래밍의 복잡도를 낮추기 위해 특정 기능을 하나로 묶어서 따로 관리하기 위해 사용

반복문: 조건이 만족하는 한 계속 반복

조건문: if ~ else 문

# 파이썬 기초 - 함수

- 함수 사용 방법

## 사용 형식

```
def 함수이름(매개변수 목록):  
    # 명령 블록  
    return 결과
```

```
1 # 함수의 호출과 반환  
2 def find_max(a, b) :  
3     if a > b :  
4         y = a  
5     else :  
6         y = b  
7     return y  
8  
9 find_max(10, 20)
```

20

```
1 # 반환값이 여러개인 경우  
2 def add_multiply(x, y) :  
3     plus = x + y  
4     gup = x * y  
5  
6     return plus, gup # 튜플로 반환  
7  
8 m, n = add_multiply(10, 20)  
9 print(m, n)
```

30 200

# 파이썬 기초 - 모듈

- 모듈 사용 방법

| import문을 사용 방법1  | import문을 사용 방법2- 함수 일부만 불러올 경우   |
|--|--|
| <b>import 모듈명</b>  | <b>from 모듈명 import</b> 변수 또는 함수  |
| import calculator<br><br>print(calculator.plus(10, 5))<br><br>print(calculator.minus(10, 5))<br><br>print(calculator.multiply(10, 5))<br><br>print(calculator.divide(10, 5)) | 1) <b>from calculator import</b> plus<br><b>from calculator import</b> minus<br><b>from calculator import</b> multiply<br><b>from calculator import</b> divide<br><br>2) <b>from calculator import</b> plus, minus, multiply, divide<br><br>3) <b>from calculator import</b> * |
| import문을 사용 방법3  |  |
| <b>import 모듈명 as 다른이름</b>  | print(plus(10, 5))<br><br>print(minus(10, 5))<br><br>print(multiply(10, 5))<br><br>print(divide(10, 5))  |
| <b>import calculator as c</b><br><br>print(c.plus(10, 5))  |  |

# 파이썬 기초 - 모듈

- 모듈 사용 방법

```
1 # 복불복 게임
2 import random
3 import time
4 play = ['까나리액젓', '콜라', '레몬', '커피', '간장'] ←
5 print('복불복게임을 시작합니다. 1초후에 알려줄께요')
6 time.sleep(1)
7 print('바로바로', random.choice(play))
```

복불복게임을 시작합니다. 1초후에 알려줄께요  
바로바로 까나리액젓

- 리스트 선언

변수는 데이터를 하나씩  
저장하나 리스트는 여러  
개를 저장할 수 있음

# Inference with OpenVINO

2020. Jan.  
인텔코리아 이인구 전무

# 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 값을 사용해서 주어진 입력 데이터가 훈련된 오브젝트중 에디에 가까운지를 분석해 줍니다.

# 인공지능 딥 러닝 구현 순서

## 1. Framework 과 Network 모델을 선정합니다.

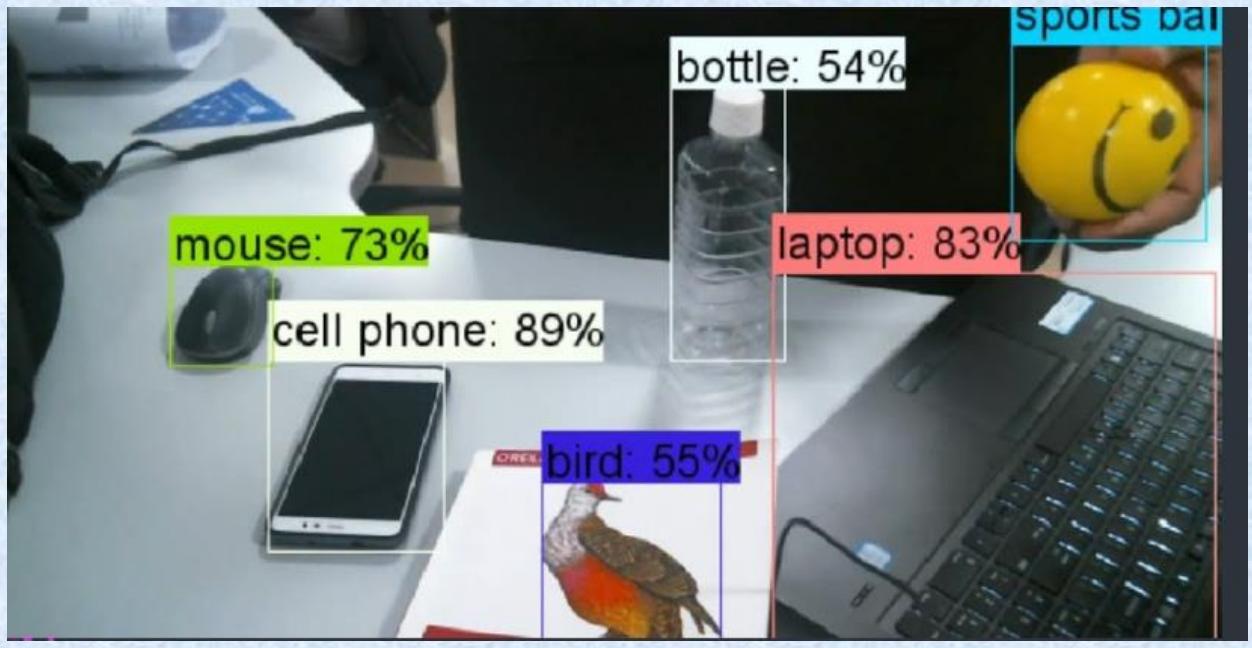
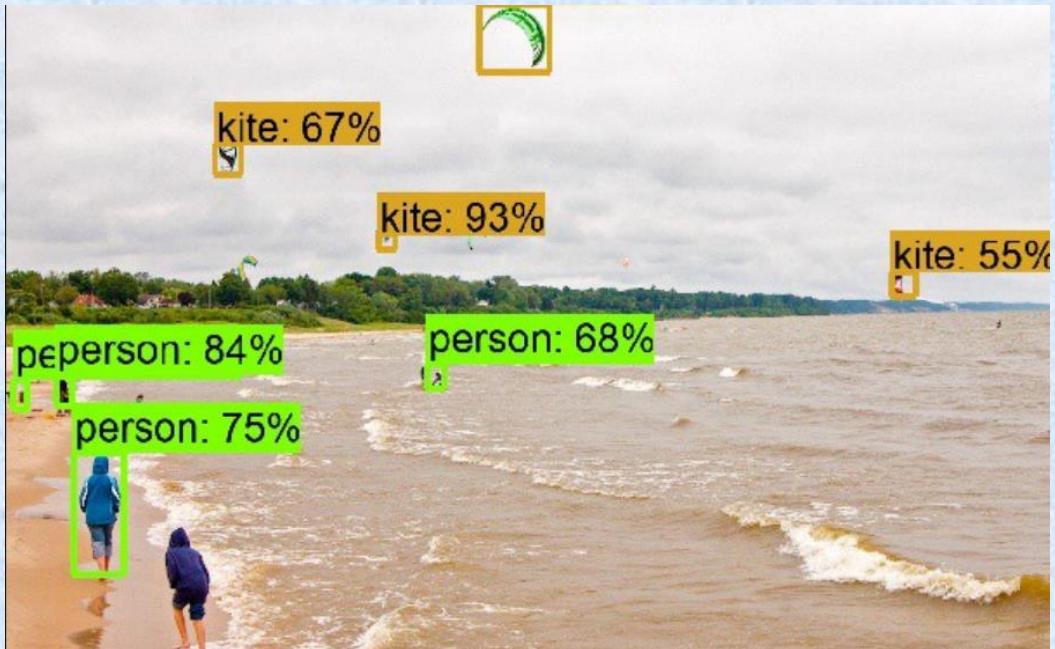
- ✓ Framework - TF, Caffe, Mxnet, BigDL,CNTK....
- ✓ Network topology – ResNet50, Inception V3, MobileNet, SSD300....
- ✓ Category
- ✓ Image classification, Object Detection, Image segmentation, Speech Recognition,...

**FRAMEWORKS OPTIMIZED BY INTEL**

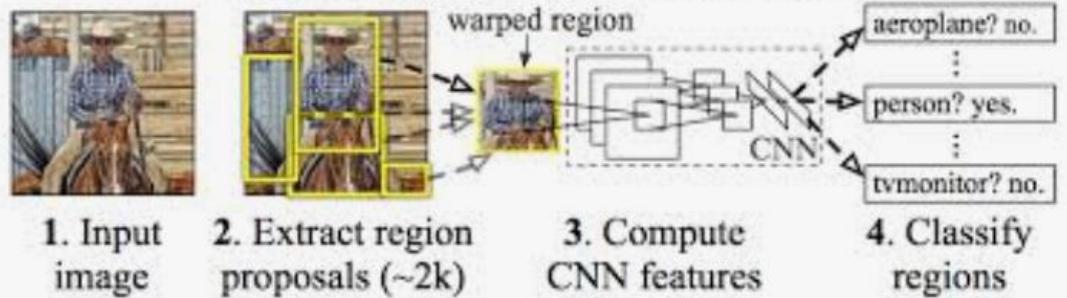
See installation guides at [ai.intel.com/framework-optimizations/](http://ai.intel.com/framework-optimizations/)

| Image Recognition                |    |    | Object Detection & Localization |    |    | Image Segmentation       |    |    |
|----------------------------------|----|----|---------------------------------|----|----|--------------------------|----|----|
| <a href="#">ResNet50*</a>        | ✓  | ✓  | <a href="#">SSD-VGG16*</a>      | ✓  | ✓  | <a href="#">MaskRCNN</a> | Q2 | 2H |
| <a href="#">InceptionV3*</a>     | ✓  | ✓  | <a href="#">R-FCN</a>           | Q2 | 2H | <a href="#">U-Net</a>    | Q2 | 2H |
| <a href="#">Inception ResV2*</a> | Q2 | 2H | <a href="#">Faster-RCNN</a>     | Q2 | Q2 | <a href="#">3D-Unet*</a> | Q2 | 2H |
| <a href="#">InceptionV4</a>      | Q2 | 2H | <a href="#">YoloV2</a>          | Q2 | 2H |                          |    |    |
| Speech Recognition               |    |    | Language Translation            |    |    | Image Generation         |    |    |
| <a href="#">Deep Speech</a>      | Q2 | 2H | <a href="#">GNMT*</a>           | ✓  | ✓  | <a href="#">DRAW</a>     | Q2 | 2H |
| <a href="#">Transformer</a>      | Q2 | 2H | <a href="#">Transformer</a>     | Q2 | Q2 |                          |    |    |
| Recommender Systems              |    |    | Text To Speech                  |    |    | Adversarial networks     |    |    |
| <a href="#">Wide &amp; Deep*</a> | Q2 | 2H | <a href="#">WaveNet</a>         | Q2 | 2H | <a href="#">DCGAN</a>    | Q2 | Q2 |
|                                  |    |    |                                 |    |    | <a href="#">3DGAN</a>    | Q2 | 2H |
| Reinforcement Learning           |    |    |                                 |    |    |                          |    |    |
|                                  |    |    |                                 |    |    | <a href="#">A3C</a>      | Q2 | 2H |

# Object detection



## R-CNN: Regions with CNN features



# Image Recognition

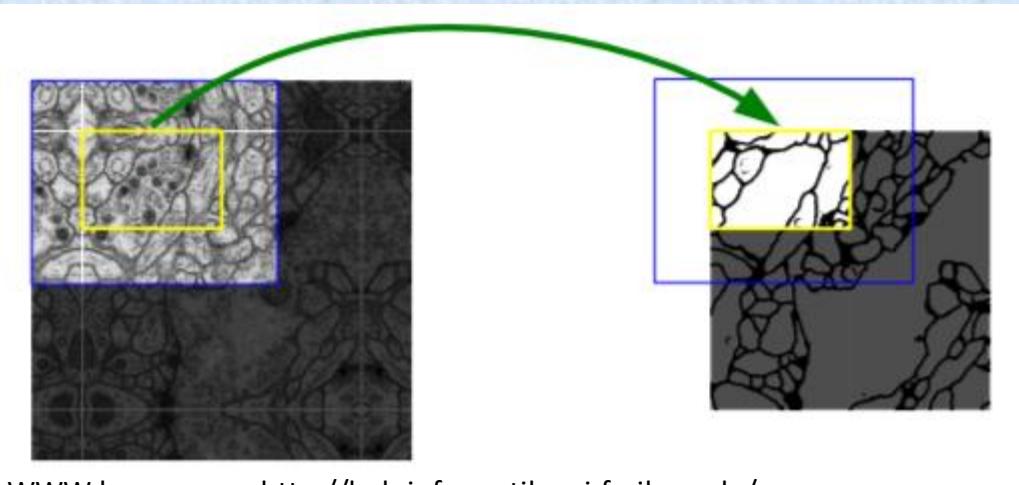
Image  
Pattern  
Face



[https://www.tensorflow.org/tutorials/images/image\\_recognition](https://www.tensorflow.org/tutorials/images/image_recognition)

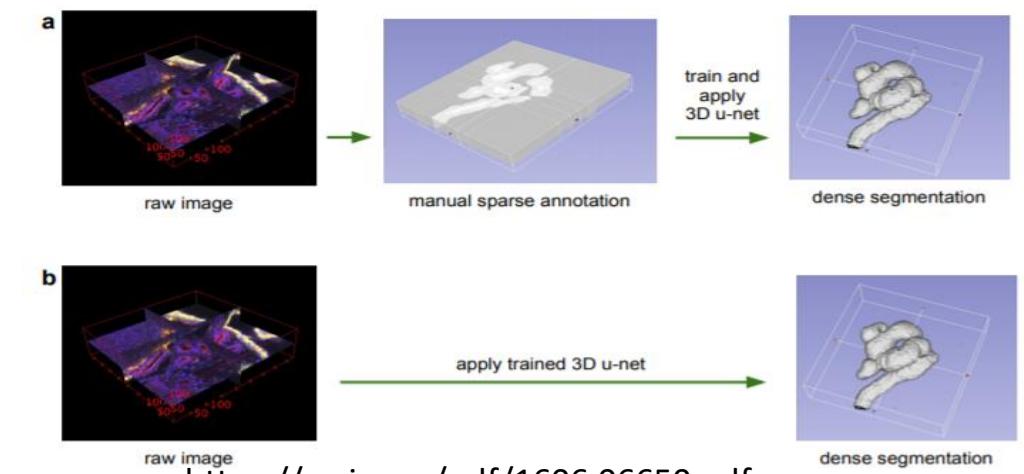
# Image segmentation

U-Net: Convolutional Networks for Biomedical Image Segmentation



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

## 2 Volumetric Segmentation with the 3D U-Net



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

## Mask R-CNN

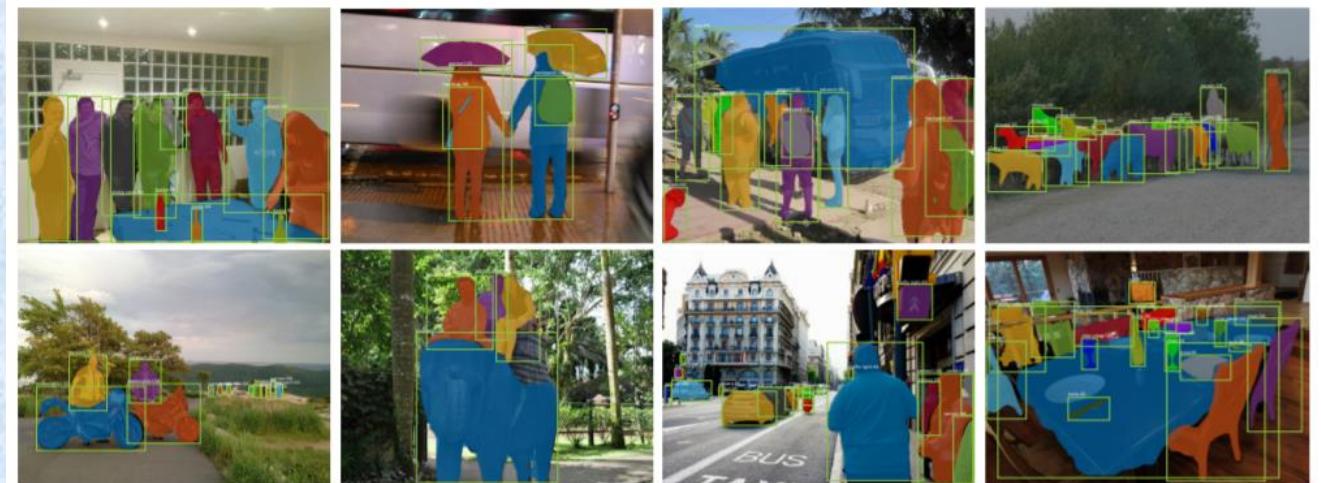
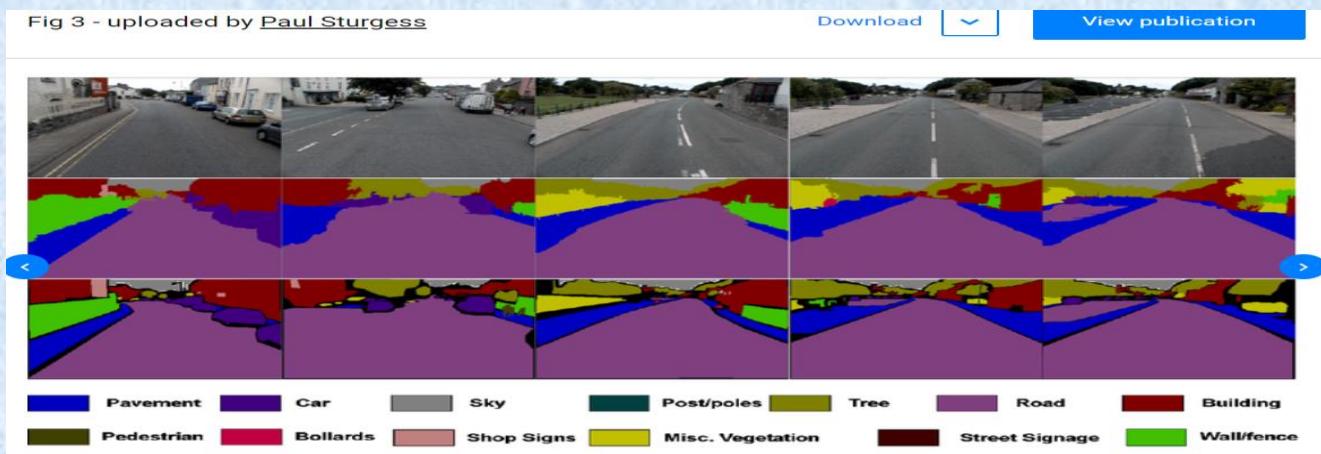


Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask AP* of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.



# Wide & Deep Learning for Recommender Systems

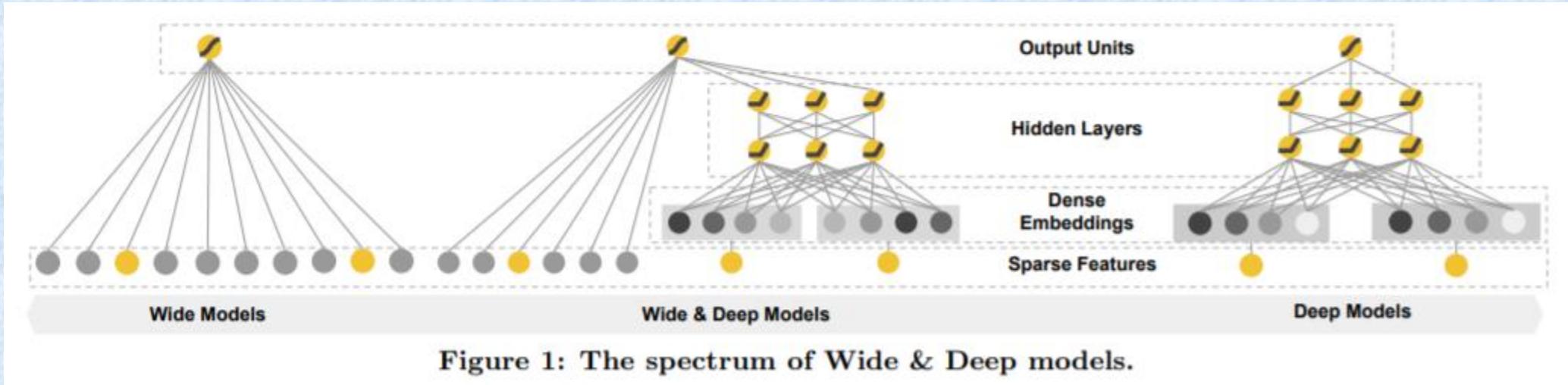


Figure 1: The spectrum of Wide & Deep models.

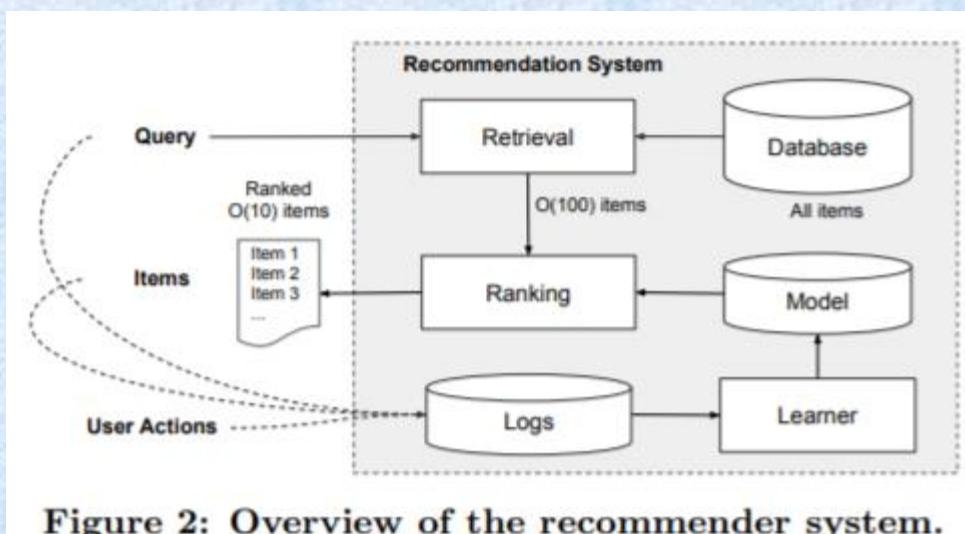


Figure 2: Overview of the recommender system.

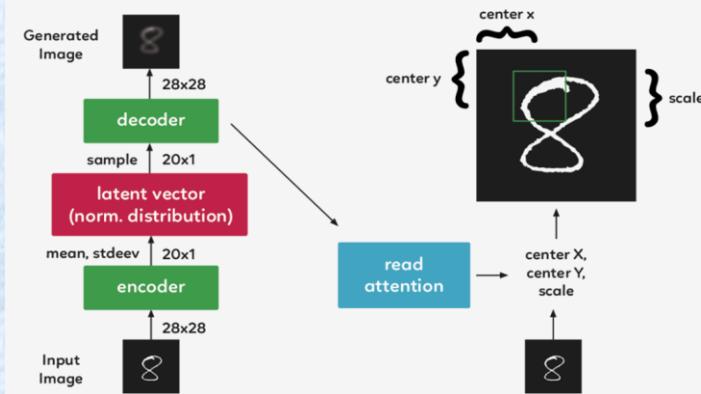
Source : Google Inc.

# 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.



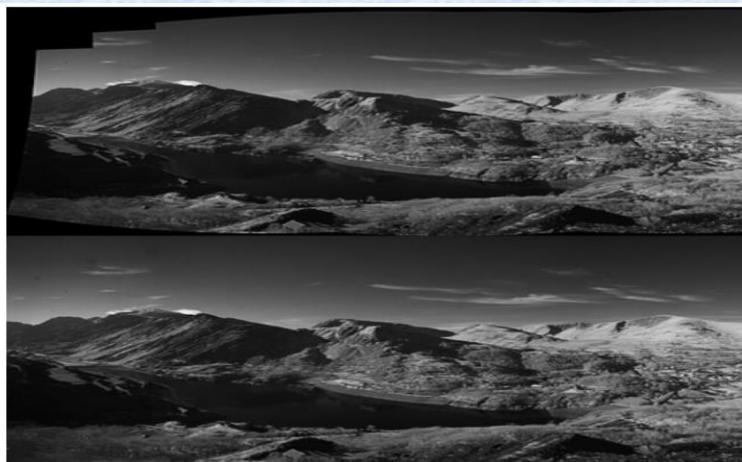
*The faces on the left were created by AI in 2014; on the right are ones made by AI in 2018. | Image: Goodfellow et al;*



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

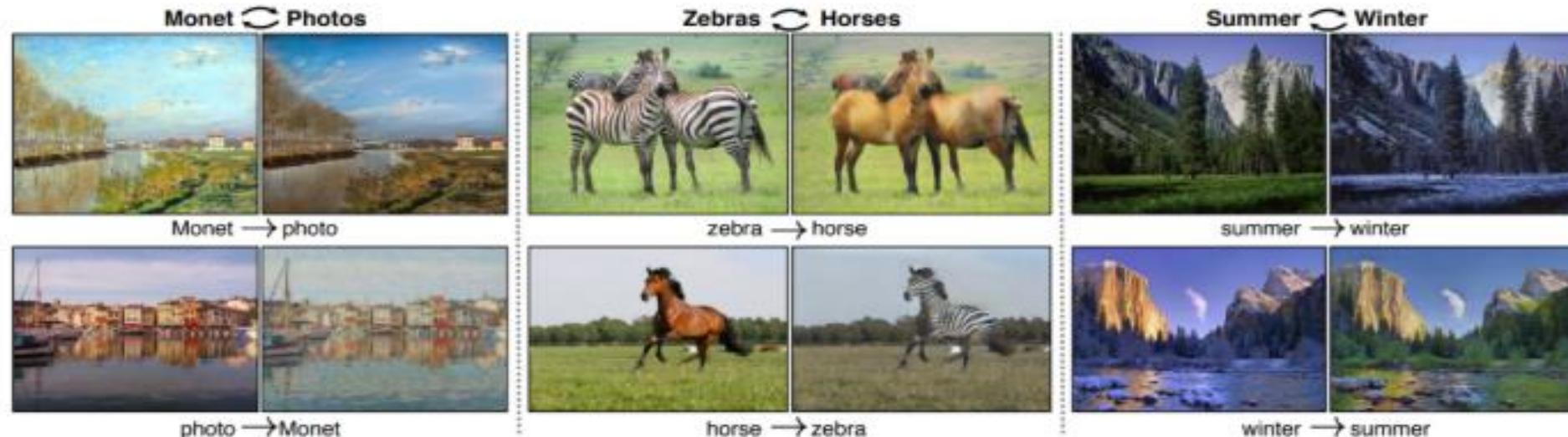
# Adversarial Networks

DCGAN (Deep Convoultional Generative adversarial Network)



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

# DC-GAN의 활용



[그림4 - cycleGAN을 통한 Image Translation]

GAN이 Image translation에서 적용된다. 흑백사진을 컬러사진으로, 간단한 일러스트를 구체적인 사진으로 만들어내는 등 우리가 원하는 대로 이미지를 바꾸어 재생성 한다.

\* 이미지 출처

그림1: (논문) [Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks](#). Alec Radford, Luke Metz, Soumith Chintala.

그림2: (논문) [Synthesizing Obama: Learning Lip Sync from Audio](#). Supasorn Suwajanakorn, Steven M. Seitz, Ira Kerner, Imacher-Shlizerman.

그림3: (논문) [Eye In-Painting with Exemplar Generative Adversarial Networks](#). Brian Dolhansky, Cristian Canton Ferrer.

그림4: (논문) [Unpaired Image-to-Image Translation using Cycle-consistent Adversarial Networks](#). Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros.

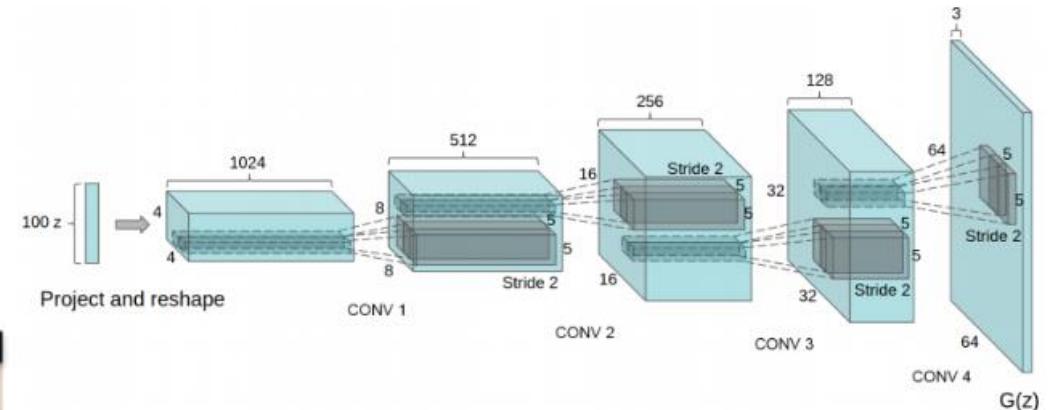
<https://m.post.naver.com/viewer/postView.nhn?volumeNo=16706734&memberNo=36733075&vType=VERTICAL>

# DC-GAN의 활용



[그림2 - GAN을 통해 합성한 오바마 전 미국 대통령의 연설 영상]

음성을 추출한후, 음성에 맞게 입 모양을 생성하도록 GAN을 학습하여 가짜영상을 만들어 낸다.



[그림1 - DCGAN의 생성자 구조도(100차원 latent vector z가 64\*64 픽셀의 이미지를 생성)]

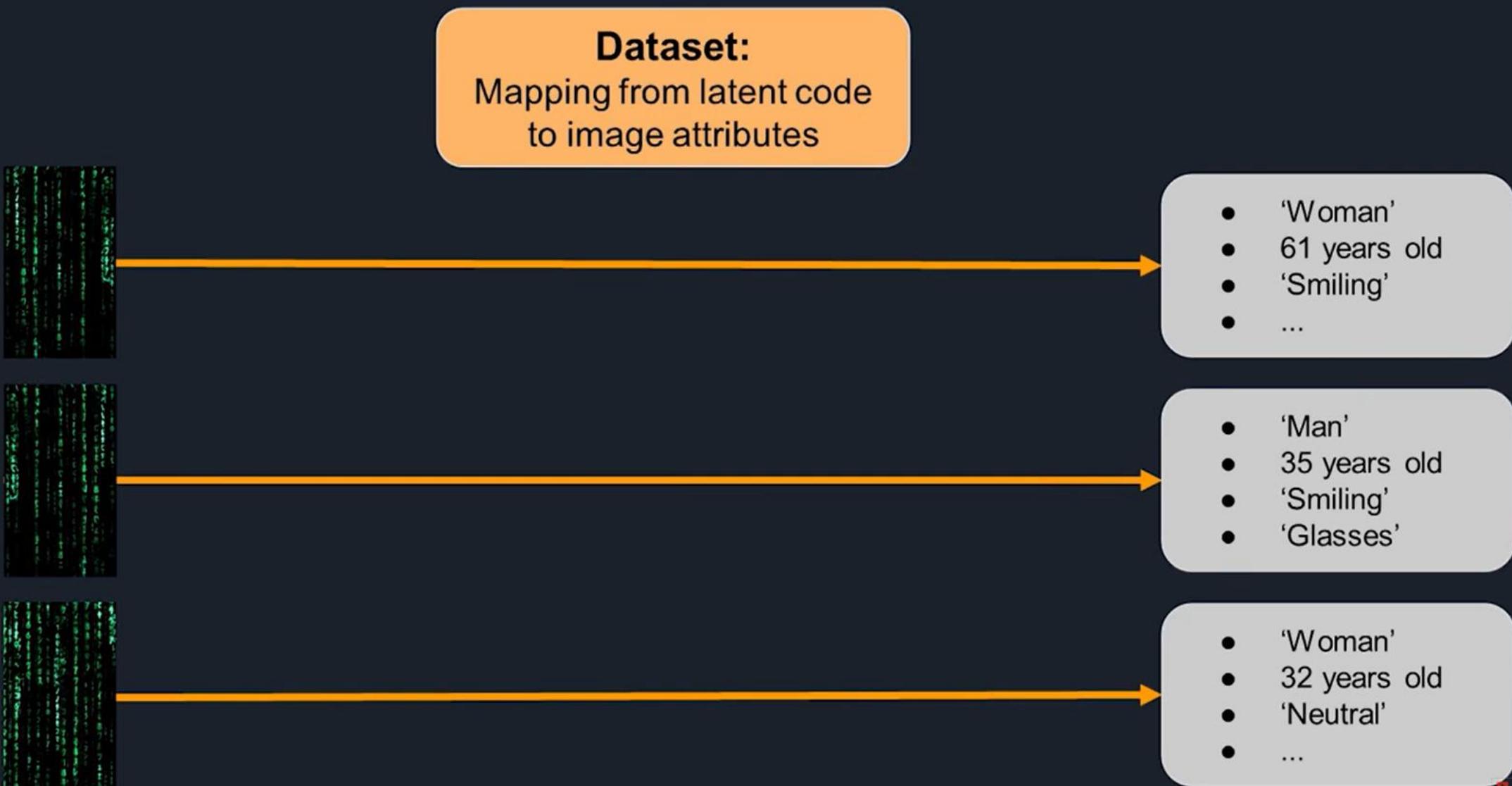


(a) (b) (c) (d)

페이스북에서 개발한 Real-eye-opener : 눈을 감은 사진에 가짜 눈을 생성하여 눈을 뜨고 있는 사진을 만들어 준다. 단순히 포토샵을 이용해 합성한 사진에 비해 훨씬 자연스럽다.

[그림3 - 페이스북의 real-eye-opener]

# Let's create another dataset!



# 인공지능 딥 러닝 구현 순서

## 2. Dataset 준비

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

## Data pre-processing

Decode

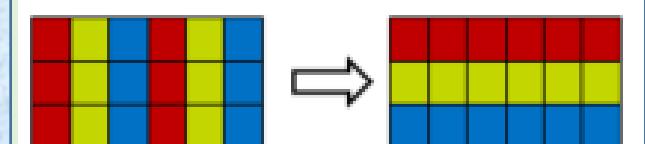
Resize

Color conversion

(Interleave -> Planar)



## Data Pre-processing



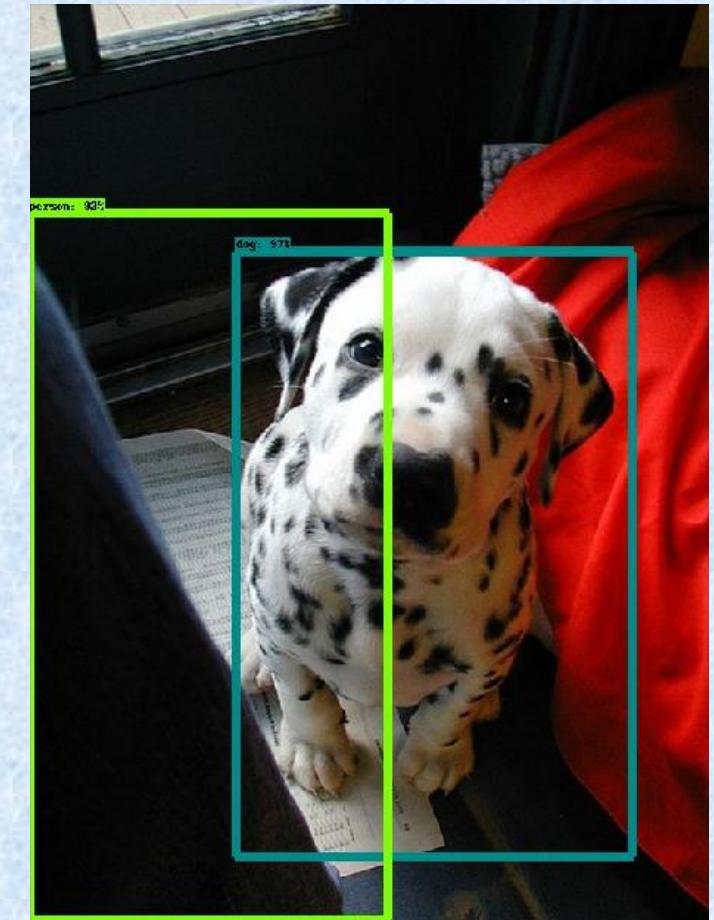
# Object detection

- ✓ Bounding box  
 $(x, y, w, h)$
- ✓ Classification  
(dog, cat, person, ...)
- ✓ Multi-Object  
 $(x_1, y_1, w_1, h_1) \rightarrow \text{Dog}$   
 $(x_2, y_2, w_2, h_2) \rightarrow \text{Person}$

Input data



Result



# Object detection

- ✓ Classification task  
Label -> Class
- ✓ Object Detection task  
Label -> Class + Bounding box



```
<annotation>
  <folder>train</folder>
  <filename>cat1.jpg</filename>
  <path>C:\Users\cat1.jpg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>440</width>
    <height>440</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>cat</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>81</xmin>
      <ymin>116</ymin>
      <xmax>377</xmax>
      <ymax>424</ymax>
    </bndbox>
  </object>
</annotation>
```

# Object Detection Labeling

Image crawler 설치 (icrawler 패키지 사용)

```
$ mkdir labeling; cd labeling
$ git clone https://github.com/eogns282/lecture\_0
$ mv lecture_0/* .
$ pip3 install icrawler --user
```

Image crawling 후, labeling 프로그램 실행 (labelImg)

```
$ python3 crawl.py
$ bash crawl.sh --user
$ cd labelImg
$ python3 labelImg.py
```

crawl.py 내부

```
$ greedy_crawler = GreedyImageCrawler(storage={'root_dir': 'crawl_image/dog'})
$ greedy_crawler.crawl(domains='http://amazing-creature.blogspot.com/2013/03/cute-puppy-pictures-30-pics.html', max_num=10, min_size=(200, 200))
```

# Object Detection Labeling

## 1. Image data labeling



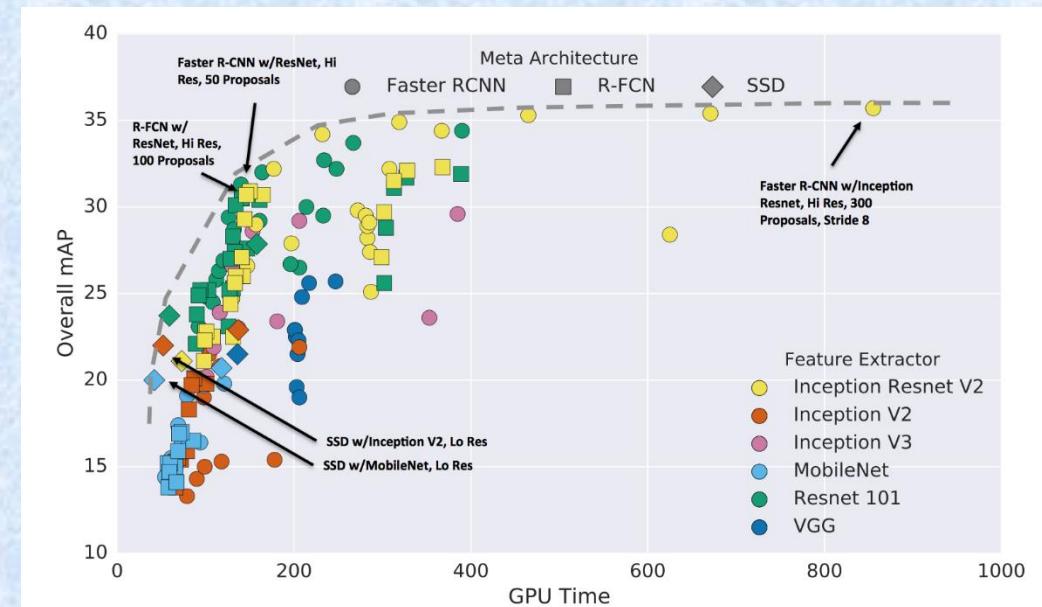
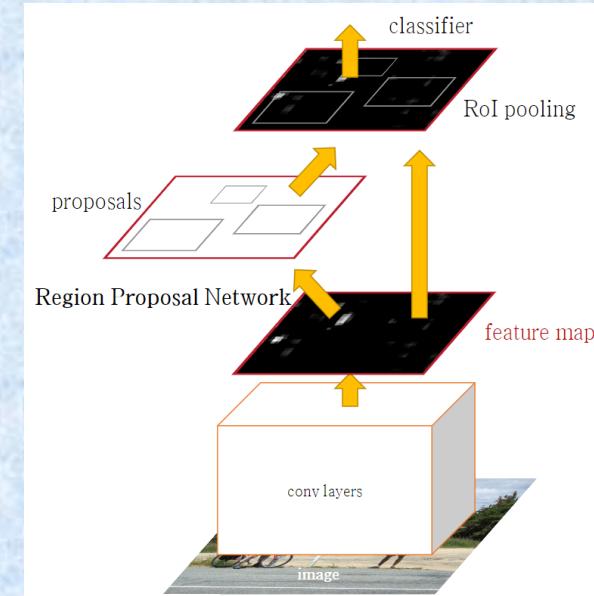
단축키

W - Bounding box 활성화  
Ctrl + s - 저장

10장 labeling 해보기

# Object detection

- ✓ R-CNN 계열 – R-CNN, Fast R-CNN, Faster R-CNN, ...  
( Class와 Bounding box를 따로 계산 )
- ✓ Single-shot 계열 – SSD, YOLO, ...  
( Class와 Bounding box를 한번에 계산 )
- ✓ Accuracy vs Speed tradeoff

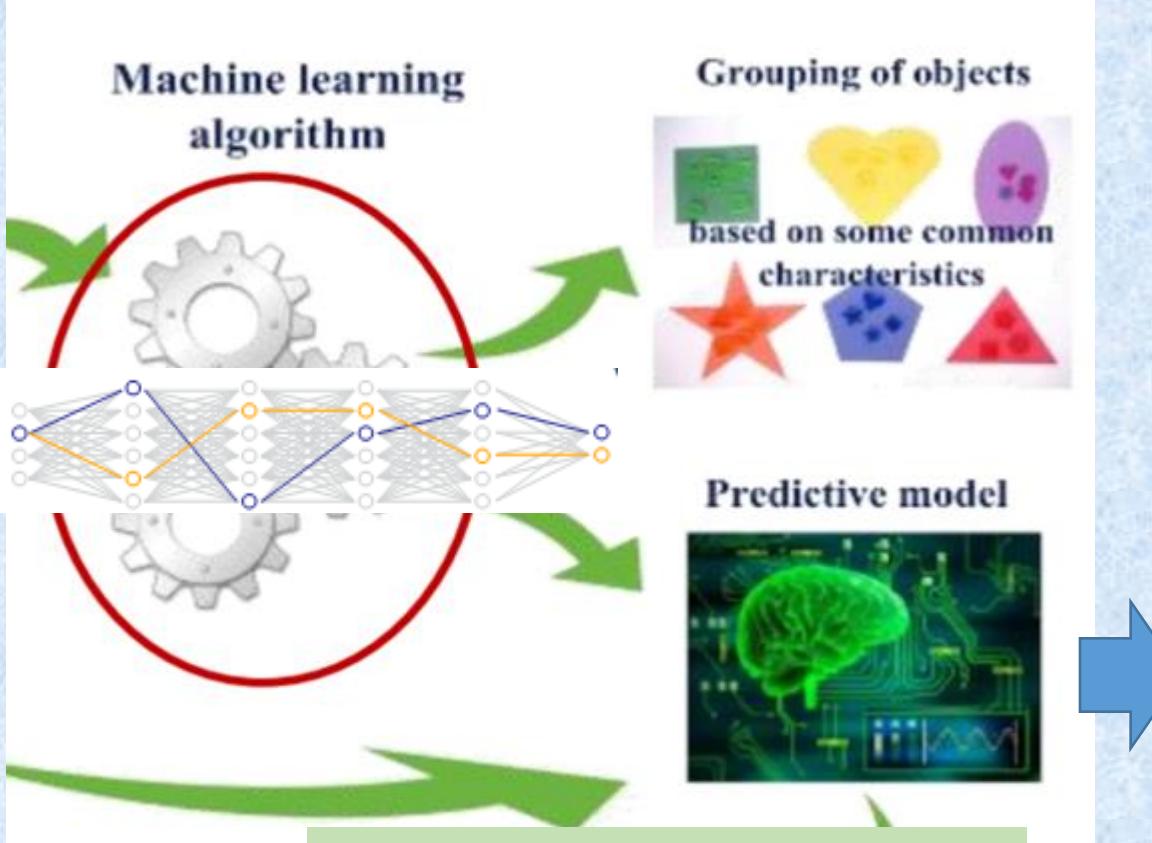
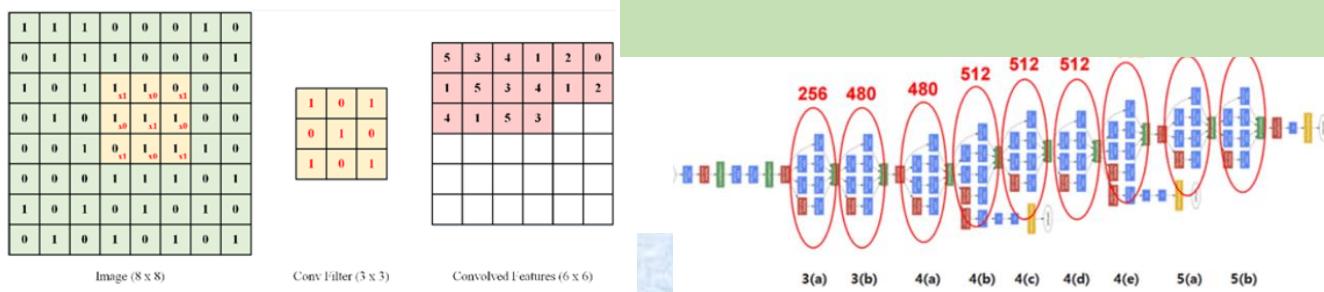
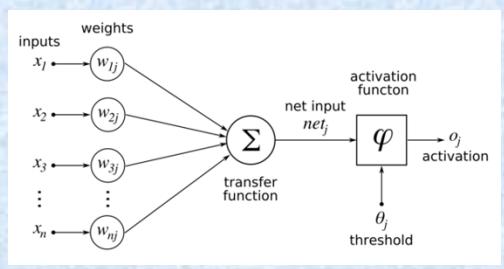


# 인공지능 딥 러닝 구현 순서

## 3. 학습 & 테스트

- ✓ 학습
- ✓ 모델만들기
- ✓ 테스트
- ✓ 최적화

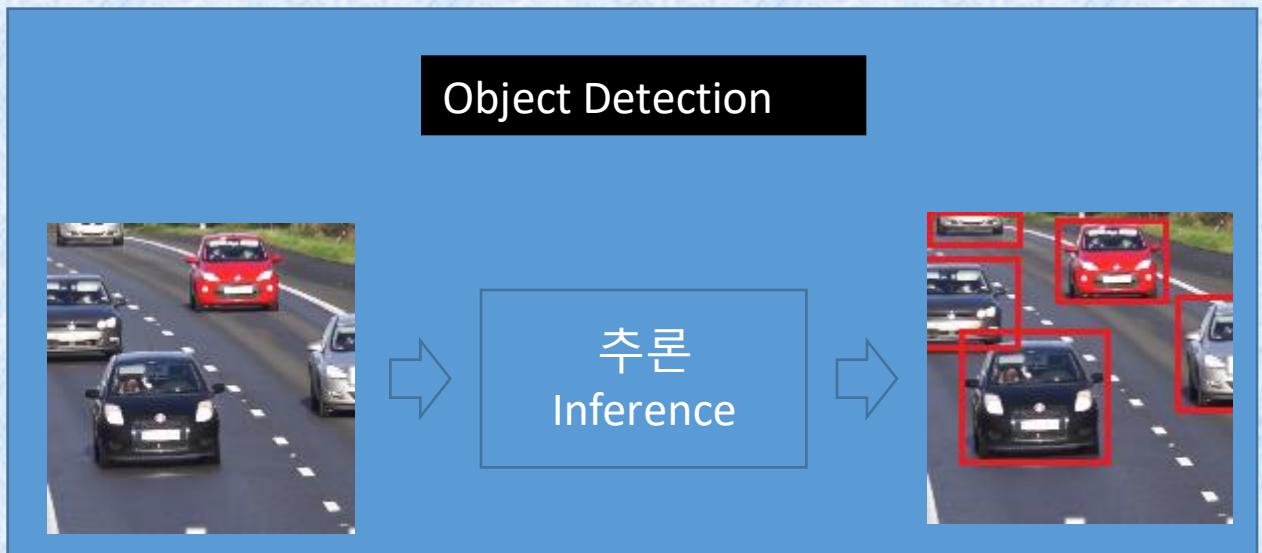
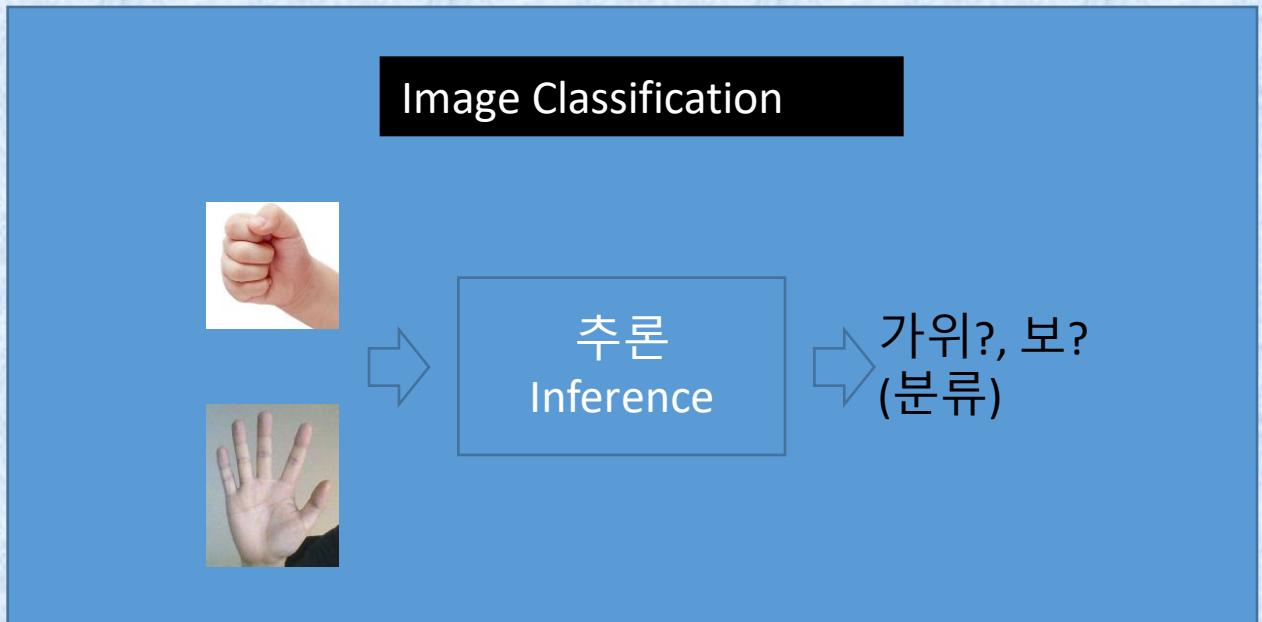
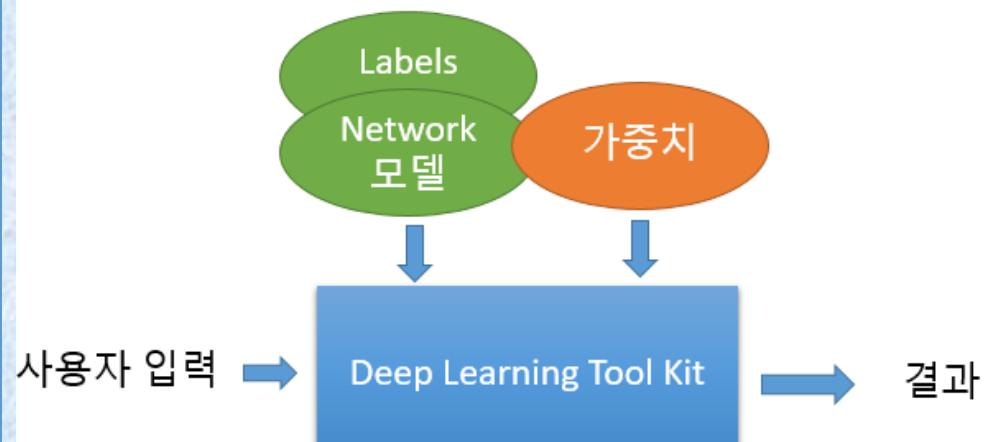
## 가중치값 생성

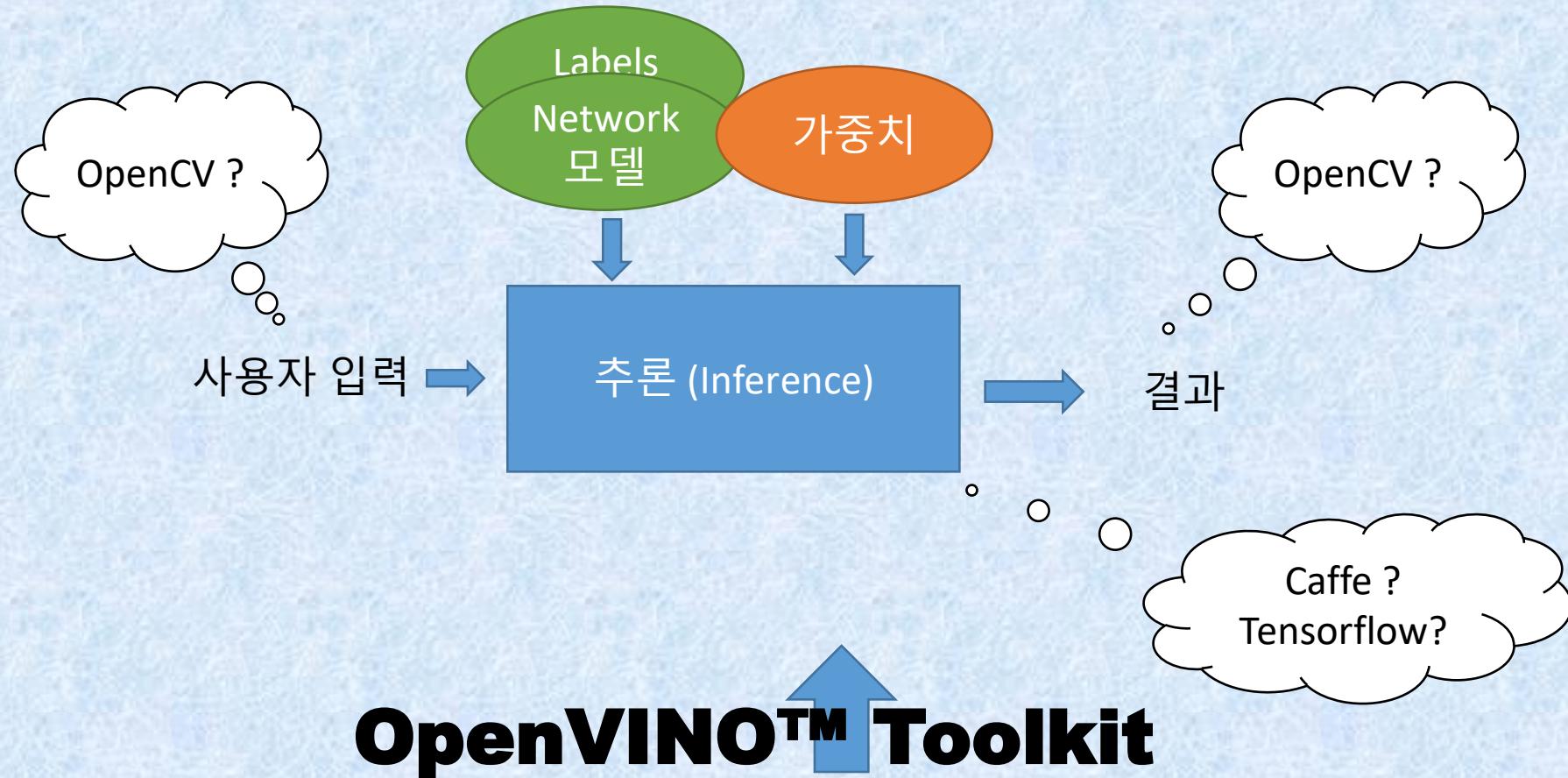


모델, 가중치값

# 인공지능 딥 러닝 구현 순서

## 4. 추론





Open **V**isual **I**nference and **N**eural Network **O**ptimization  
영상을 추론하고 신경망을 최적화 하는 도구 모음

# OPENVINO™ TOOLKIT

## 딥러닝 인퍼런스 툴킷 : Computer vision tools +DL Inference performance

### Intel® Deep Learning Deployment Toolkit

**Model Optimizer**  
Convert & Optimize



**Inference Engine**  
Optimized Inference

Code Samples & 10 Pre-trained Models

IR = Intermediate Re  
presentation file



### Traditional Computer Vision Tools & Libraries

#### Optimized Libraries

OpenCV\*

OpenVX\*

Photography Vi  
sion

Code Samples

For Intel® CPU & CPU with integrated graphics

#### Increase Media/Video/Graphics Performance

**Intel® Media SDK**  
Open Source version

**OpenCL™  
Drivers & Runtimes**

For CPU with integrated graphics

#### Optimize Intel® FPGA

FPGA RunTime Environment  
(from Intel® FPGA SDK for OpenCL™)

Bitstreams

FPGA – Linux\* only

**Intel® Architecture-Based  
Platforms Support**



[software.intel.com/openvino-toolkit](http://software.intel.com/openvino-toolkit)

All products, computer systems, dates, and figures are preliminary based on current expectations, and are subject to change without notice.

OS Support CentOS\* 7.4 (64 bit) Ubuntu\* 16.04.3 LTS (64 bit) Microsoft Windows\* 10 (64 bit) Yocto Project\* version Polk Jethro v2.0.3 (64 bit)

OpenVX and the OpenVX logo are trademarks of the Khronos Group Inc.

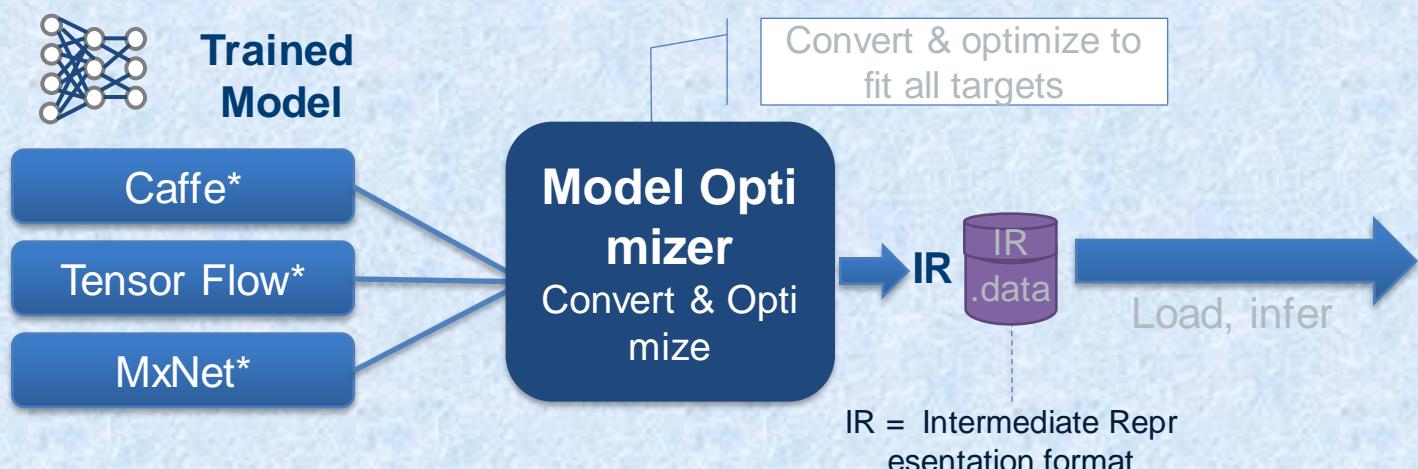
OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos

# INTEL® DEEP LEARNING DEPLOYMENT TOOLKIT (DLDT)

다양한 프레임워크와 다양한 디바이스를 쉽게 사용할수 있도록한 Inference tool

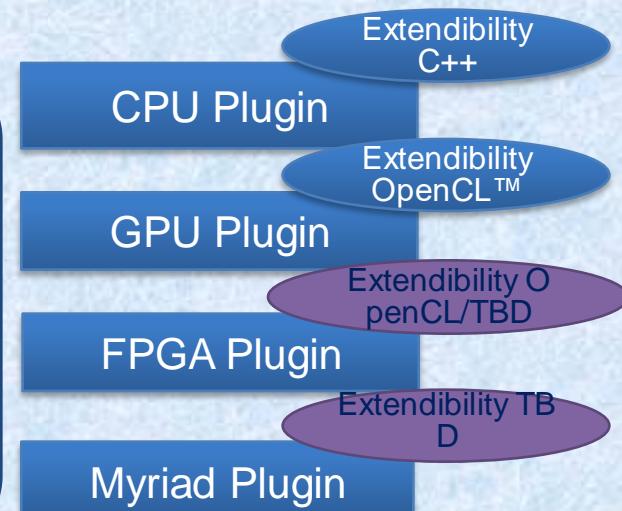
## Model Optimizer

준비 단계-> 훈련 된 모델 가져와서 토폴로지 변환을 통해 성능 / 공간을 최적화합니다. 즉, HW와 일치하는 데이터 형식으로 변환하는 것입니다.



## Inference Engine

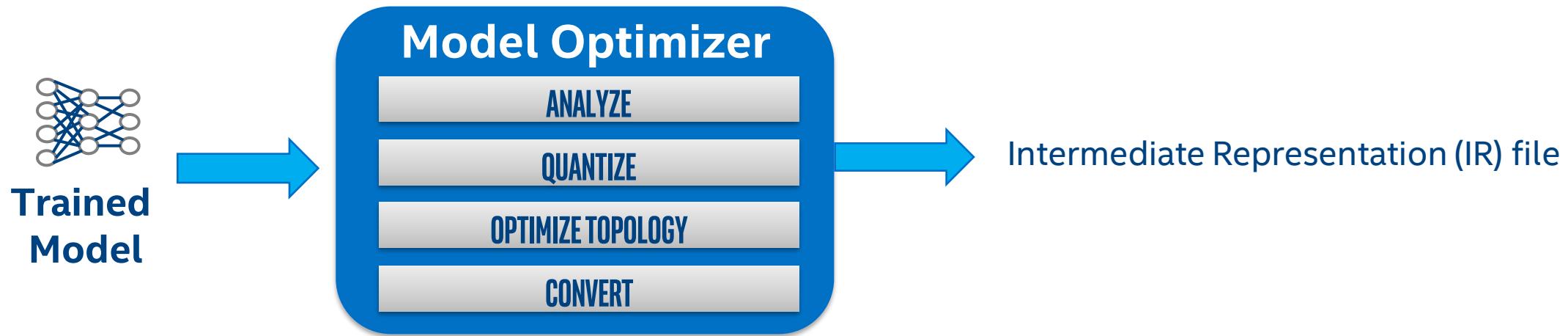
추론 API : 인터페이스는 각 HW유형에 대해 동적으로 로드되는 플러그인으로 구현됩니다. 사용자가 여러 코드 경로를 구현하지 않고도 각 유형에 대해 최상의 성능을 제공합니다.



GPU = Intel CPU with integrated graphics processing unit/Intel® Processor Graphics

All products, computer systems, dates, and figures are preliminary based on current expectations, and are subject to change without notice.

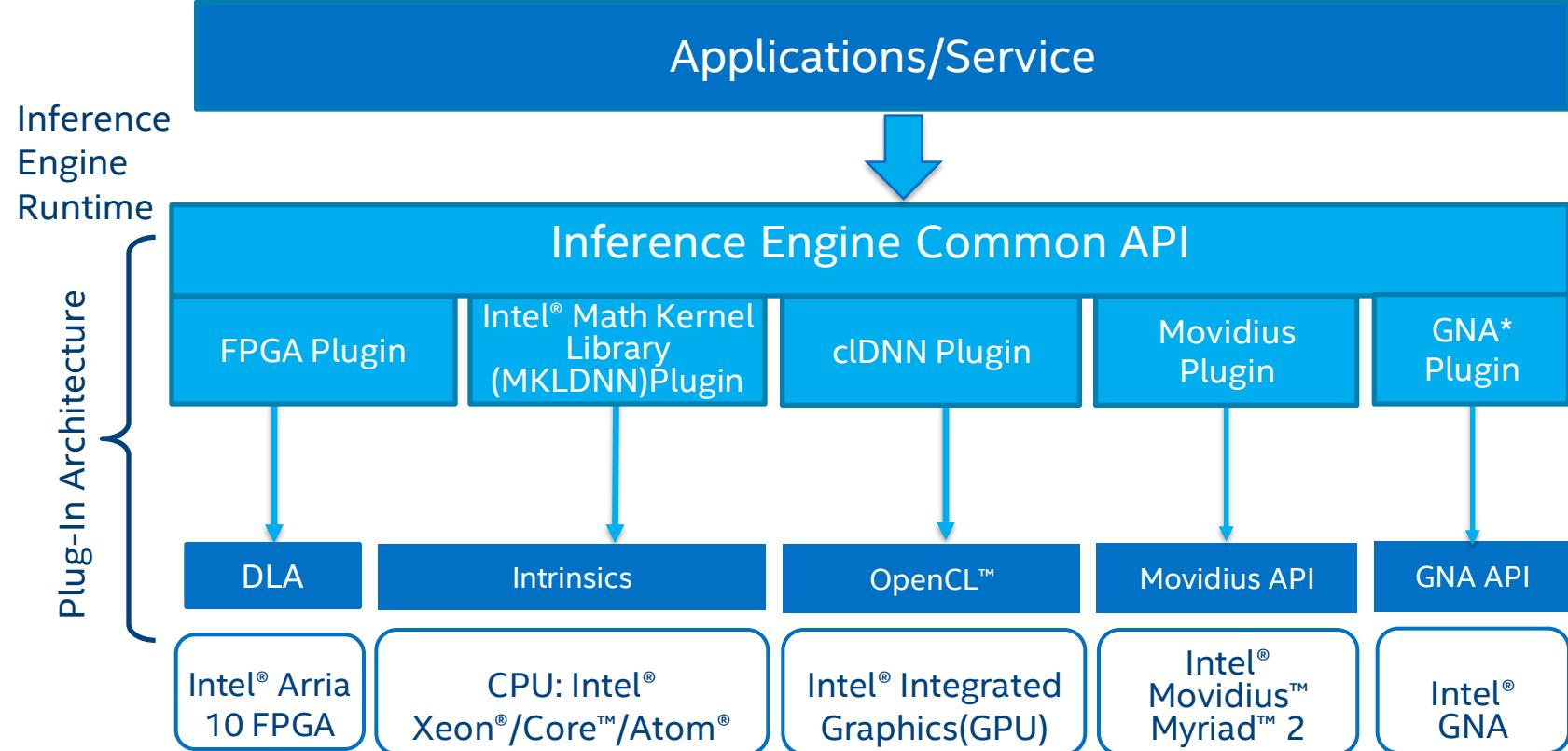
# Improve Performance with Model Optimizer



- Python \* 기반 워크 플로우는 프레임 워크 재구성이 필요없음.
- 다양한 프레임 워크에서 모델 가져 오기 (Caffe \*, TensorFlow \*, MXNet \*, Kaldi \* 등)
- Caffe, MXNet 및 TensorFlow를 위한 100 개 이상의 모델 검증.

# Optimal Model Performance Using the Inference Engine

- 모든 인텔® 아키텍처에서 추론을 위한 단순하고 통합 된 API
- 대규모 IA 하드웨어 타겟 (CPU / GEN / FPGA)에 대한 최적화 된 추론
- 이질성 지원으로 하드웨어 유형에 따른 레이어 실행 가능
- 비동기 실행으로 성능 향상
- 미래의 인텔® 프로세서를 위한 미래형 / 확장형 개발



Transform Models & Data into Results & Intelligence

GPU = Intel CPU with integrated graphics processing unit/Intel® Processor Graphics/GEN  
GNA = Gaussian mixture model and Neural Network Accelerator

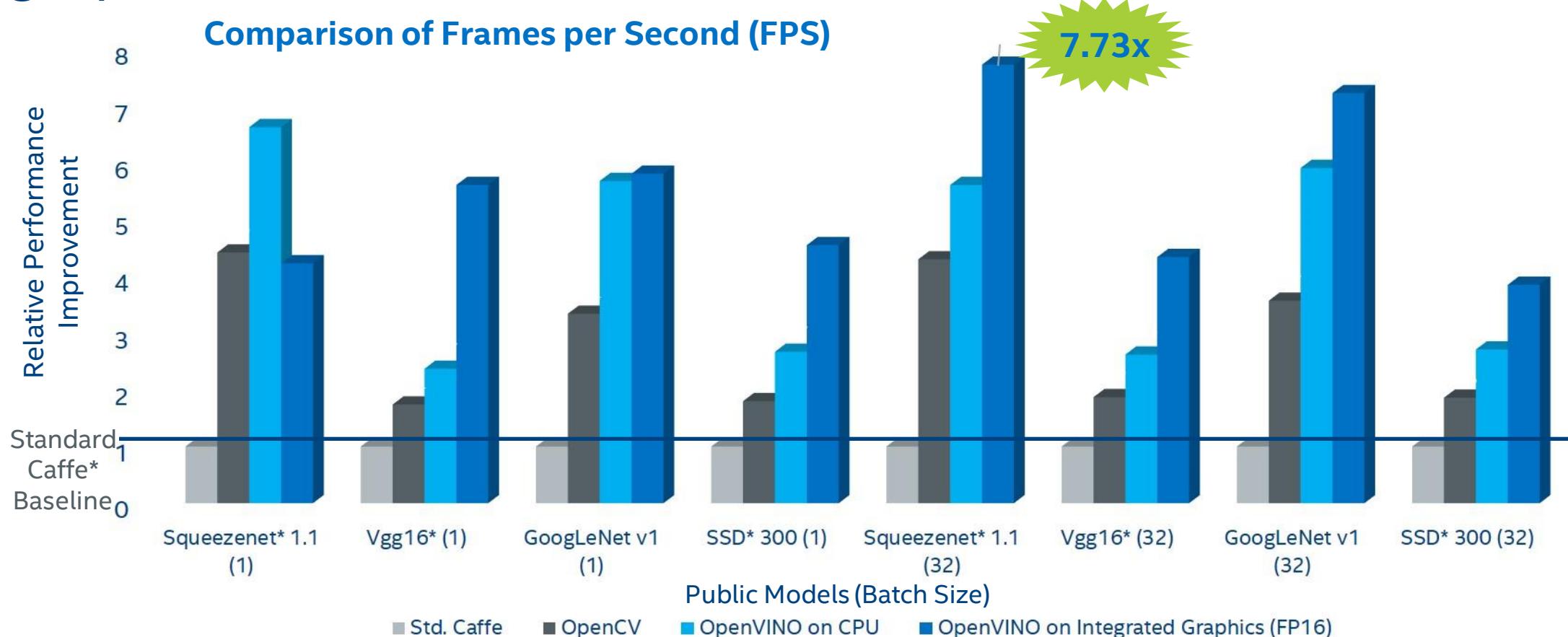
## Optimization Notice

Copyright © 2018, Intel Corporation. All rights reserved.  
\*Other names and brands may be claimed as the property of others.

OpenVX and the OpenVX logo are trademarks of the Khronos Group Inc.  
OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos



# Increase Deep Learning Workload Performance on Public Models using OpenVINO™ toolkit & Intel® Architecture



Fast Results on Intel Hardware, even before using Accelerators

# Deep Learning Samples & Models

## Face Detection



- **Retail Environment Model**
- **Head Position**
- **Emotion Recognition**

승객이 차량에 있는지 또는 실내 보행자 수를 세는 것과 같은 다양한 용도로 얼굴을 식별하십시오. 그것을 사람 탐지기와 결합하여 누구가오고 가는지 확인하십시오.

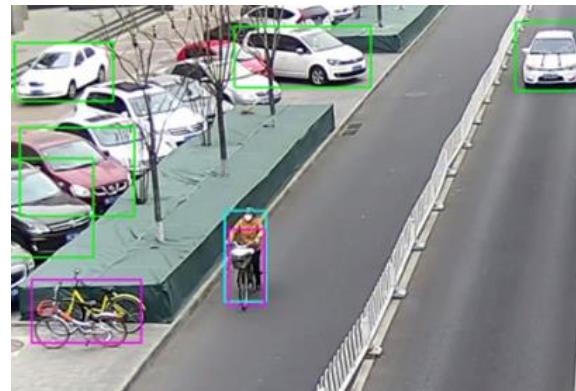
### Pre-Trained Models

- 연령 및 성별
- 얼굴 인식 - 표준 & 고급
- 머리 위치
- 인간 탐지 - 눈높이 & 고각 탐지
- 사람, 차량 및 자전거 감지
- 차량 번호판 탐지
- 차량 메타 데이터
- 차량 탐지
- 리테일 환경
- 보행자 감지
- 보행자 및 차량 감지
- 사람 속성 인정 사거리
- 감정 인식
- 다른 동영상으로 누군가 식별 - 표준 및 고급
- 길가 객체 식별
- 진보 된 길가 식별

# Human Detection

## Detect People, Vehicles, & Bikes

사람, 자전거 타는 사람, 자전거 만 혼자하는 사람과 차량을 구별. 이 모델의 다양한 조명 조건은 일광, 어두움 및 날씨 변화의 정확성을 향상시킴.



- Eye-Level Detection
- High-Angle Detection
- Pedestrian Detection
- Pedestrian & Vehicle Detection
- Identify Someone in Different Videos
- Pedestrian Attributes

### Optimization Notice

Copyright © 2018, Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.

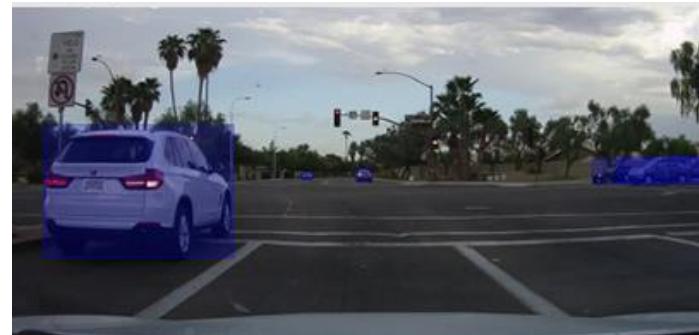
OpenVX and the OpenVX logo are trademarks of the Khronos Group Inc.



# Vehicle Feature Recognition

## Vehicle Detection

튜닝 된 MobileNetV1을 사용하여 기능을 추출하는 SSD 프레임 워크를 적용하여 차량을 식별.  
다운로드 : \ deployment\_tools \ intel\_models \ vehicle-detection-adas-0002



Type: regular  
Color: black

- License Plate Detection: Small-Footprint Network
- License Plate Detection: Front-Facing Camera
- Vehicle Metadata
- Identify Roadside Objects
- Advanced Roadside Identification

Load Input Image(s)

Run Inference 1:  
Model  
vehicle-license-plate-detection-barrier-0007

Detects Vehicles



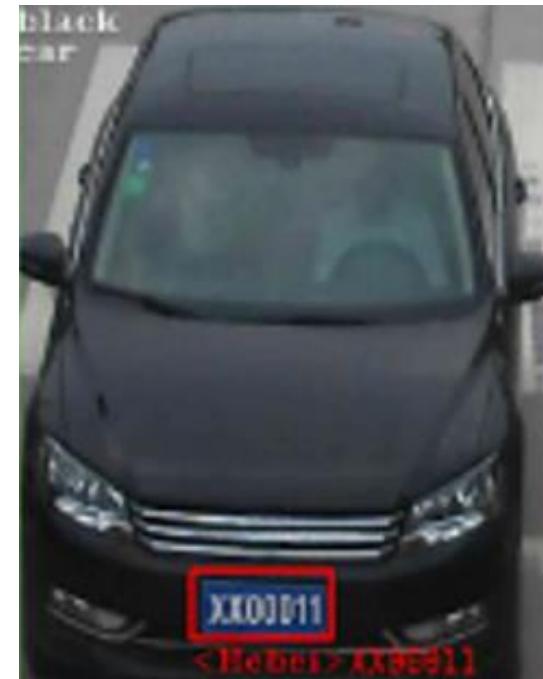
Run Inference 2:  
Model  
vehicle-attributes-recognition-barrier-0010

Classifies vehicle attributes

Run Inference 3:  
Model  
license-plate-recognition-barrier-0001

Detects License Plates

Vehicle Detection Time : 30.10 ms (33.23 fps)  
Vehicle Attribs Time (averaged over 2 detections) : 6.26 ms (159.71 fps)  
LPR Time (averaged over 1 detection) : 5.04 ms (198.43 fps)



Display Results

[Optimization Notice](#)

Copyright © 2018, Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.



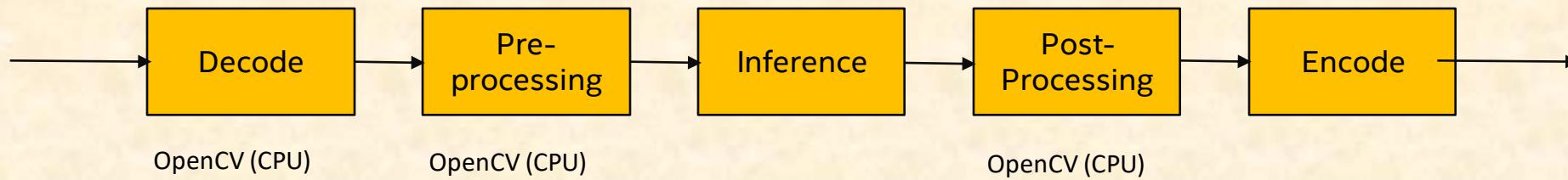
# OpenVINO 설치하기

웹브라우저에서 Intel Distribution OpenVINO 검색.

Get start → OpenVINO download → 설치 가이드 따라서 설치



# 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
2. OpenVINO에서 Python을 이용해서 모델을 컨버젼 하기 -- \*.bin, \*.xml
3. Inference를 위한 코딩
  - 입력 데이터 위치 확인하기 - 이미지, 동영상, 카메라
  - 사용할 HW을 선택하기 – CPU, GPU, Movidius
  - 입력 데이터를 가공하기 – Decode, Re-size, Color conversion
  - 인퍼런스 - OpenVINO – Image classification, Object Detection
  - 데이터 출력 - Boxing, Encoding
4. 실행 – 결과물

코딩하기

# OpenVINO 을 이용한 추론 – Mobilenet-SSD 모델

## 1. 모델 가져오기

Mobilenet은 20가지 사물을 학습시킨 image classification 모델로 가볍게 사용이 가능합니다.

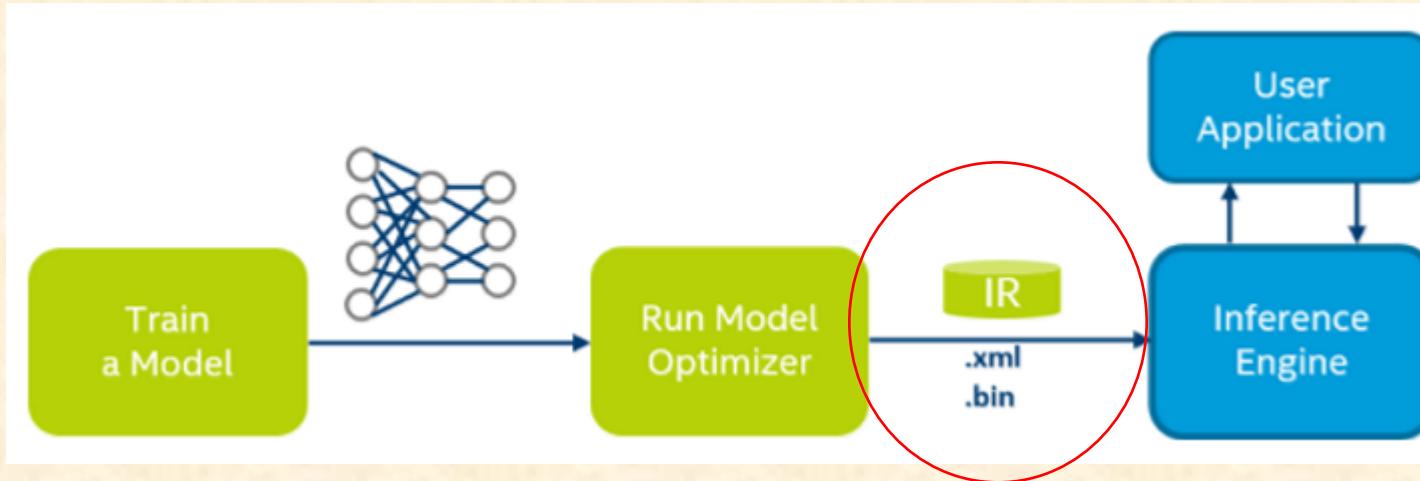
Caffe에서 Pre-trained된 model [mobilenet-ssd.caffemodel](#)을 이용하여 OpenVION에서 추론 하기.

- Model conversion/optimization
- 입출력, 네트워크 모델 지정 하기.
- 추론 및 지정한 이미지 저장하기 .

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 2. 모델 Conversion하기

Inference Engine이 원하는 포맷으로 Network 모델과 가중치 정보를 전환합니다.



.caffemodel  
.prototxt

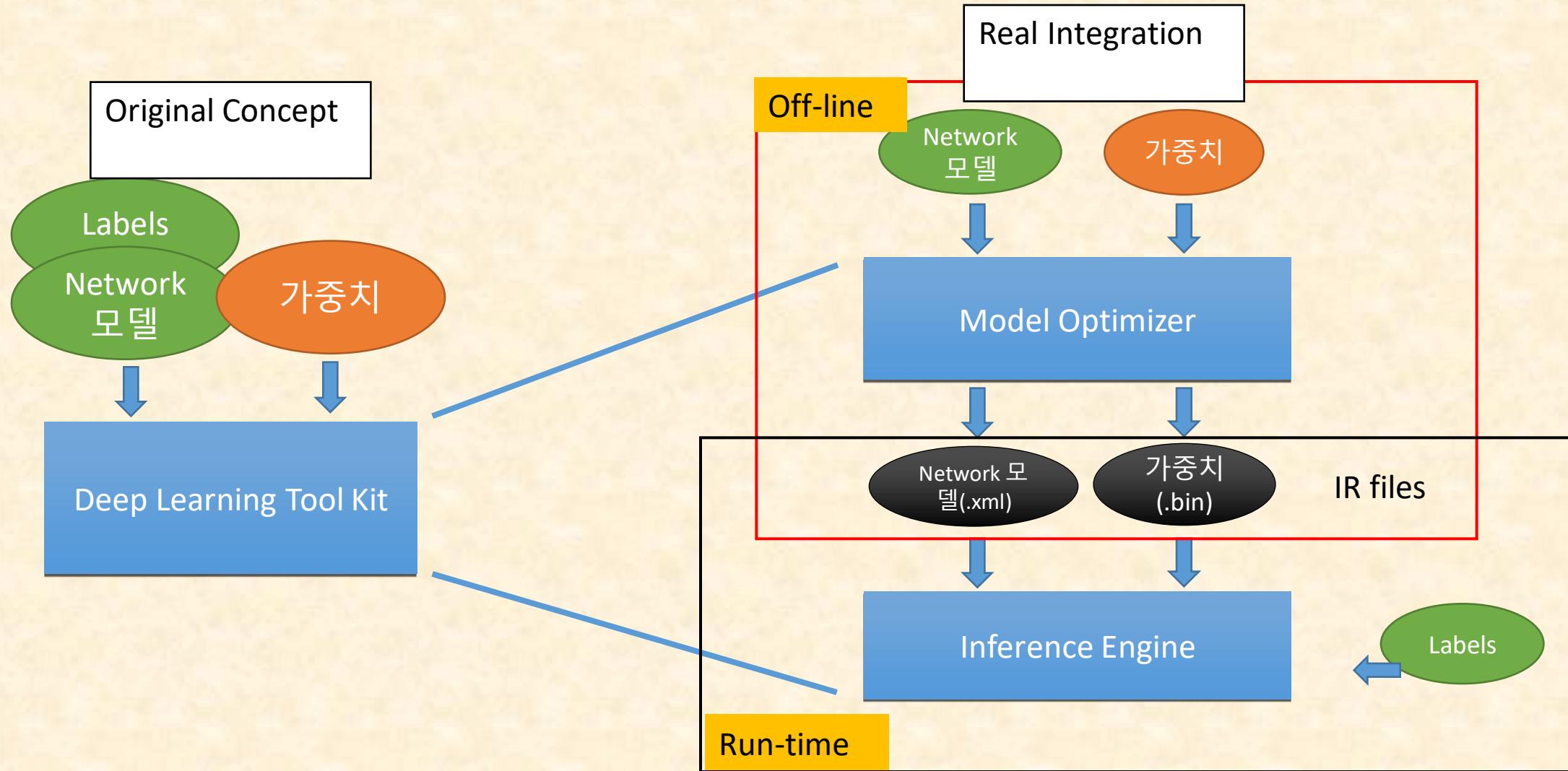
Intermediate Representation



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 Conversion 하기

/home/intel/my\_model/의 위치에 Caffe로 트레이닝을 해서 만들어진 모델 mobilenet-ssd.caffemodel과 mobile-ssd.prototxt가 있습니다.

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

모델 전환이 성공적으로 이루어 졌다면  
~/my\_model 폴더에 mobile-ssd.xml, mobile-ssd.bin file들이 생성됩니다.

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 2. 모델 Conversion 하기\_FP16의 경우

1. CPU는 FP32 data format 을 이용하고 Movidius 와 GPU는 FP16 data forma 의 IR 파일들 사용며, Data forma을 data\_type FP16 으로 지정해주면 FP16 format 으로 생성됩니다.

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

그러면 ~/my\_model/ 에 test.xml/bin 파일이 FP16 bit version 으로 생성됩니다.  
data\_type을 지정하지 않으면 FP32 bit version이 기본으로 생성됩니다.

2. FP16으로 생성했으면 Plugin 을 “eMYRIAD” 로 변경합니다.  
(GPU 를 사용하기 위해서는 “eGPU” 로 변경)

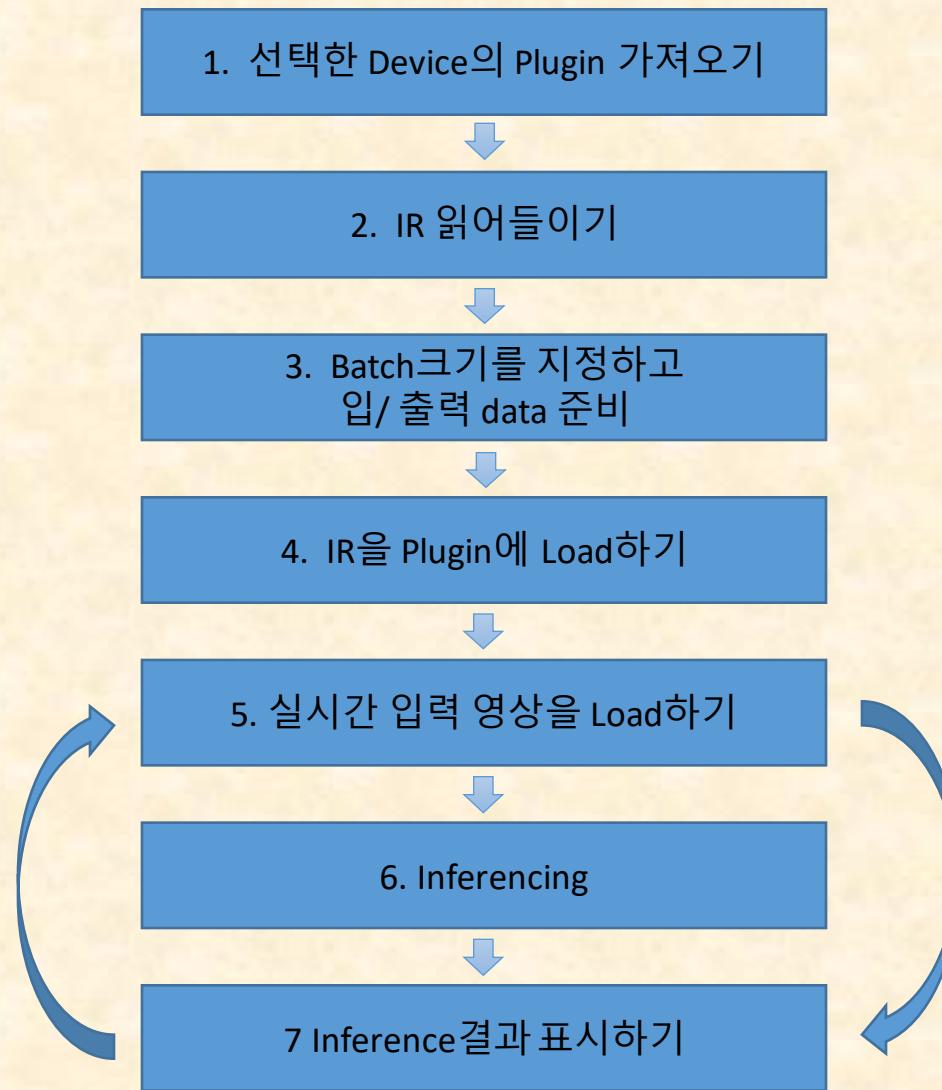
```
InferencePlugin plugin(dispatcher.getSuitablePlugin(TargetDevice::eCPU));
```



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

# OpenVINO의 Deep Learning Toolkit을 사용하여 s/w 구현하기

## 3. Inference 코딩 하기



# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

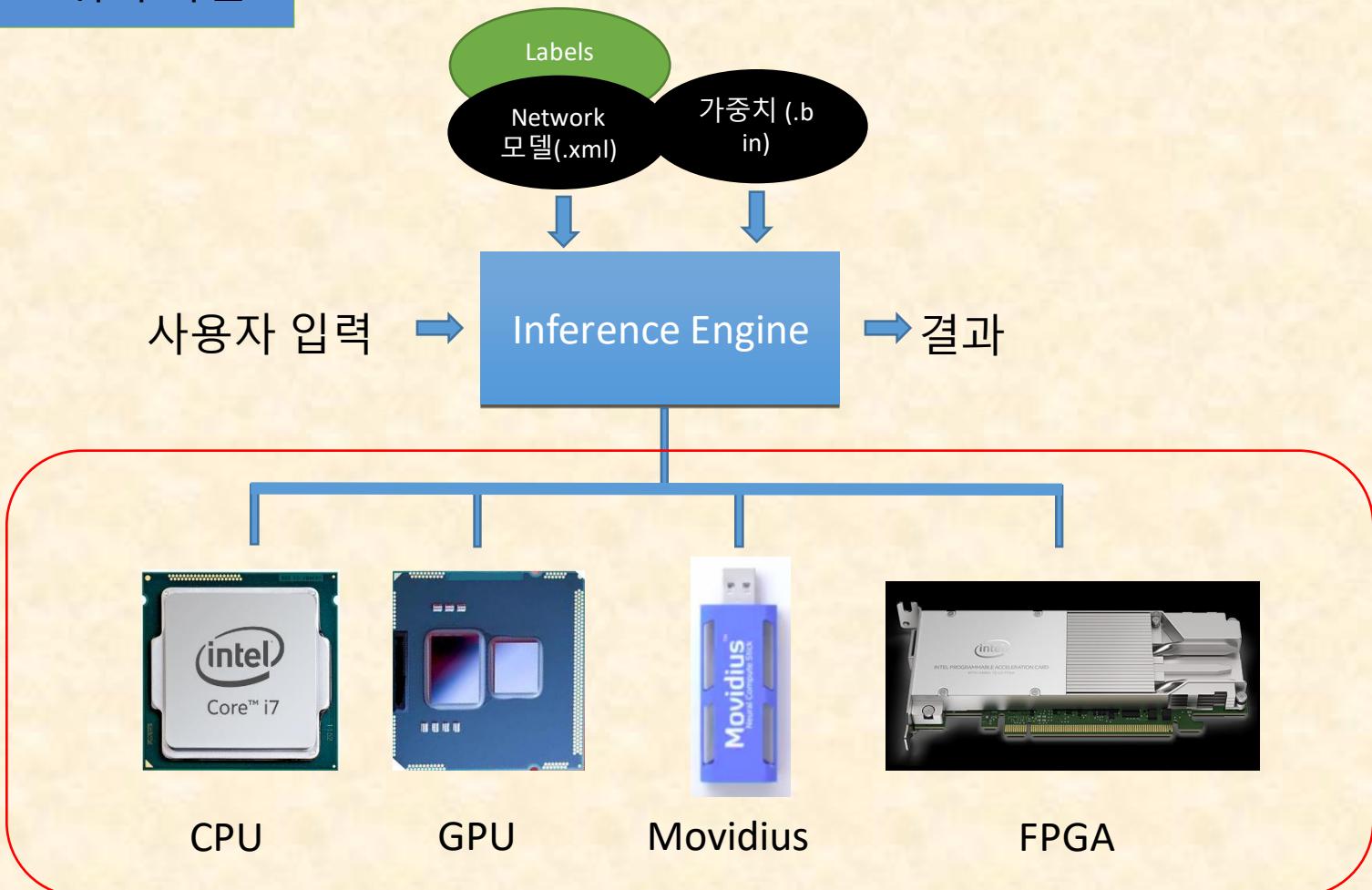
## 3. 추론 하기 (코드 실행)

### 코드 실행

```
$ cd ~/sample  
$ python3 mobilenet_woori.py
```

# OpenVINO의 Deep Learning Toolkit을 사용하여 s/w 구현하기

사용할 HW 선택/ Lib 위치 확인



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

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 1. 선택한 Device의 Plugin 가져오기

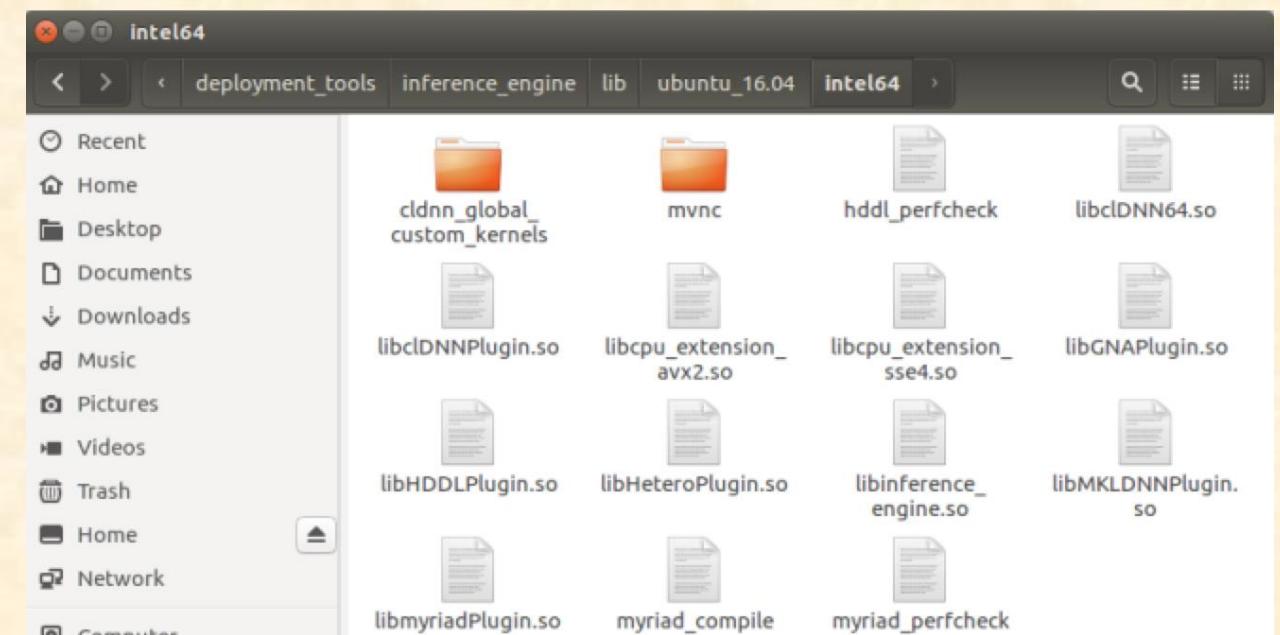
추론 엔진 가져오기

```
from openvino.inference_engine import IENetwork, IEPlugin
```

플러그인 로딩

```
plugin = IEPlugin("CPU",
"/opt/intel/openvino_2019.1.133/deployment_tools/inference_engine/lib/ubuntu_16.04/intel64")
```

CPU를 사용하여 Inference를 수행한다.



# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 2. IR 읽어들이기

플러그인에 extension library 로드

```
plugin.add_cpu_extension("/opt/intel/openvino/deployment_tools/inference_engine/lib/intel64/libcpu_extension_sse4.so")
```

모델 로딩

```
model_xml = "/home/intel/my_model/FP32/mobilenet-ssd.xml"
model_bin = "/home/intel/my_model/FP32/mobilenet-ssd.bin"
print("Loading network files:{}\n{}".format(model_xml, model_bin))
```

네트워크 정보 로딩

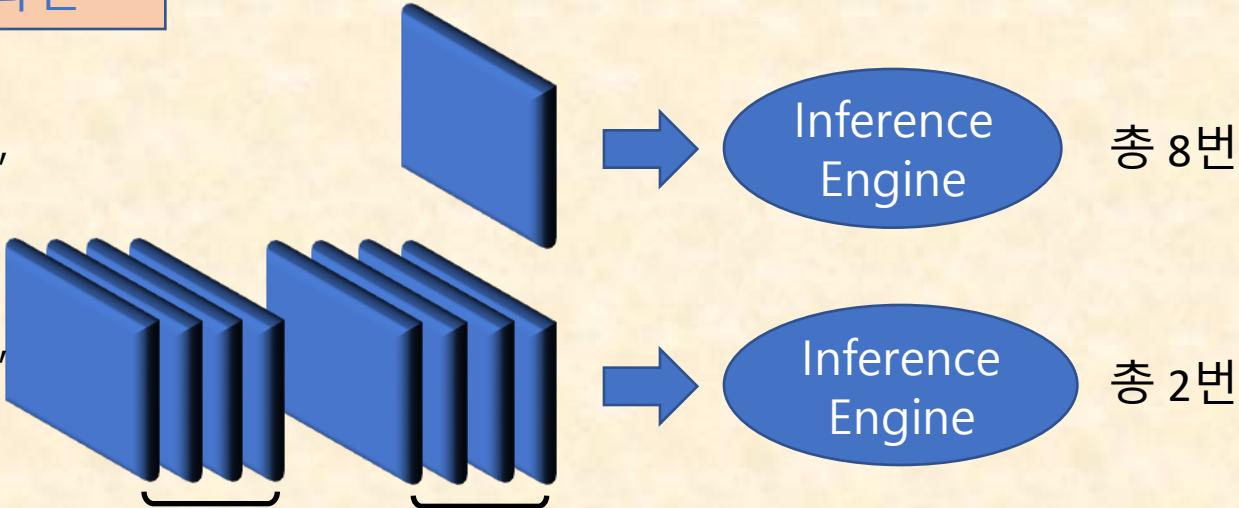
```
net = IENetwork(model=model_xml, weights=model_bin)
```

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 3. Batch크기를 지정하고 입/ 출력 data 준비

만약, 입력 data가 8장이라면

Batch Size = 1 인 경우,



배치 사이즈 설정

```
net.batch_size = 1
```

입출력 데이터 정보 설정

```
input_blob = next(iter(net.inputs))  
out_blob = next(iter(net.outputs))
```

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 4. IR을 Plugin에 Load하기

실제 실행 네트워크로 로딩

```
exec_net = plugin.load(network=net)
```

## 5. 실시간 입력 영상을 Load하기

VideoCapture object 생성

cv2.VideoCapture(0) 카메라 영상을 가져옴.

cv2.VideoCapture("동영상제목")으로 동영상 파일을 가져올 수도 있음

```
cap = cv2.VideoCapture(0)
if not cap.isOpened(): sys.exit('camera error')
```

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 6. Inferencing

Detect 가능한 사물 리스트

```
labels = ["plane", "bicycle", "bird", "boat", "bottle", "bus", "car", "cat", "chair", "cow", "table", "dog",  
"horse", "motorcycle", "person", "plant", "sheep", "sofa", "train", "monitor"]
```

같은 물체를 여러번 찍지 않기 위한 flag

```
captured = False
```

15프레임 이상 detect 돼야 프레임 캡처를 하기 위한 카운터

```
detect_count = 15
```

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 6. Inferencing – while문 내부

카메라의 한 프레임을 읽어옴.  
읽는데 실패했을 시 다음 루프로 넘어감

```
ret, frame = cap.read()  
if not ret: continue
```

Esc를 누르면 루프 탈출해 프로그램 종료

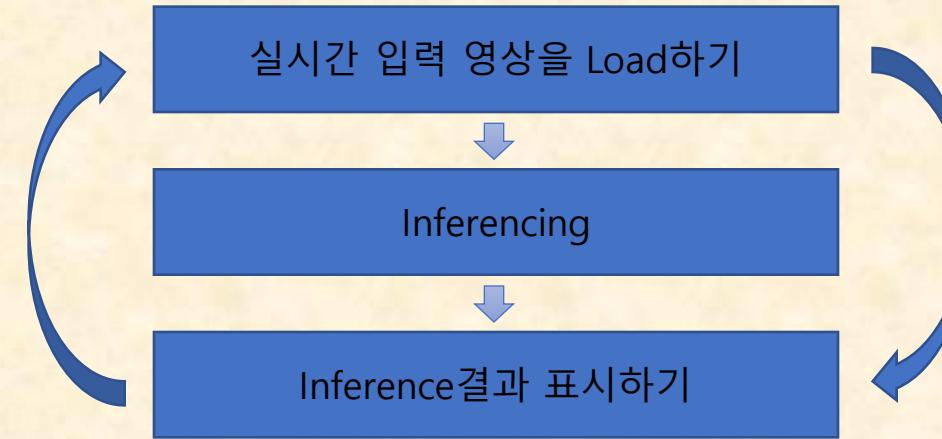
```
ch = cv2.waitKey(1) & 0xFF  
if ch == 27: break
```

첫 사진 캡처 여부를 저장하기 위한 flag

```
is_first = True
```

같은 물체를 여러 번 캡처하지 않기 위한 flag

```
captured = False
```



# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 6. Inferencing

15프레임 이상 detect 돼야 프레임 캡처를 하기 위한 카운터

```
detect_count = 15
```

사진 캡쳐 시간을 저장하는 변수

```
first_detect_time = 0  
last_detect_time = 0
```

Input에 맞게 이미지 사이즈 조정 및 색상 변환

```
n, c, h, w = net.inputs[input_blob].shape
```

```
images = np.ndarray(shape=(n, c, h, w))  
images[0] = cv2.resize(frame, (300, 300)).transpose((2,0,1))
```

이미지 추론

```
res = exec_net.infer(inputs={input_blob: images})  
detections = res[out_blob][0][0]
```

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 7 Inference 결과 표시하기

추론 결과처리 후 일정 인식률(confidence) 이상인 데이터만 이후 코드 실행

```
for i, detect in enumerate(detections):
```

```
    image_id = float(detect[0])
    label_index = int(detect[1])
    confidence = float(detect[2])
```

```
    if image_id < 0 or confidence == 0.:
        continue
```

```
    if confidence > 0.7:
```

```
        ...
```

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 7 Inference 결과 표시하기

Confidence가 0.7보다 큰 데이터에 대해

원하는 물체가 인식됐을 시 detect\_count를 1씩 증가.

0.7보다 작으면 detect\_count가 0이 되므로 15프레임 이상 연속으로 detect되어야 프레임을 캡처한다.

```
if confidence > 0.7:  
    detected = labels[label_index-1]  
    print(detected, detect_count)  
    #if detected == "cat":  
    #    print("cat confidence:", confidence)  
    if detected == "person" or detected == "dog" or detected == "plane" or detected == "horse":  
        detect_count += 1  
        if detect_count > 15:  
            ...  
            ...  
  
    else:  
        detect_count = 0  
        captured = False
```

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 7 Inference 결과 표시하기

### 이미지 캡처 코드

cv2.imwrite 함수로 이미지를 저장한다.

Timestamp에 캡처 시간을 분 초 100분의 1초를 저장해  
사진 파일의 제목으로 사용한다.

캡쳐 후 captured를 True로 만들어 한 상황에 한 번만 캡처가 되도록 한다.

```
if detect_count > 15:  
    font = cv2.FONT_HERSHEY_SIMPLEX  
    msg = "{} image saved".format(detected)  
  
    if not captured:  
        now = time.time()  
        localtime = datetime.datetime.fromtimestamp(int(now))  
        hundred = str(now).split('.')[1][:2]  
        timestamp = "%02d%02d%s" % (localtime.minute, localtime.second, hundred)  
        cv2.imwrite("{}{}.jpeg".format(detected, timestamp), frame)  
  
    captured = True  
    if is_first:  
        first_detect_time = time.time()  
        is_first = False  
    else:  
        last_detect_time = time.time()  
cv2.putText(frame, msg, (30, 30), font, 1.5, (255,255,255), 2, cv2.LINE_AA)
```

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 7 Inference 결과 표시하기

### 이미지 캡처 코드

처음 이미지 캡처가 된 시간을 first\_detect\_time에 저장한다.

is\_first를 False로 만들어 그 후부터는 last\_detect\_time에 업데이트 된다.

cv2.putText 함수로 사진이 캡처 됐음을 메세지로 띄운다.

```
if detect_count > 15:  
    font = cv2.FONT_HERSHEY_SIMPLEX  
    msg = "{} image saved".format(detected)  
  
    if not captured:  
        now = time.time()  
        localtime = datetime.datetime.fromtimestamp(int(now))  
        hundred = str(now).split('.')[1][:2]  
        timestamp = "%02d%02d%s" % (localtime.minute, localtime.second, hundred)  
        cv2.imwrite("{}{}.jpeg".format(detected, timestamp), frame)  
  
    captured = True  
    if is_first:  
        first_detect_time = time.time()  
        is_first = False  
    else:  
        last_detect_time = time.time()  
cv2.putText(frame, msg, (30, 30), font, 1.5, (255,255,255), 2, cv2.LINE_AA)
```

# OpenVINO 을 이용한 추론 – Mobilenet-SSD 모델

SSD300은 20가지 사물을 학습시킨 object detection 모델로 가볍게 사용이 가능합니다.

Caffe 에서 Pre-trained된 model SSD300을 이용하여 OpenVION에서 추론 하기.

- Model conversion/optimization
- 입출력, 네트워크 모델 지정 하기.
- 추론 및 지정한 이미지 저장 하기 .

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 1. 모델 Conversion 하기

/home/intel/my\_model/ 의 위치에 Caffe 로 트레이닝을 해서 만들어진 모델 ssd300.caffemodel과 ssd300.prototxt가 있다.

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

모델 전환이 성공적으로 이루어 졌다면 ~/my\_model 폴더에 test.xml, test.bin file들이 생성된다.

## 2. Coding 하기 – 입출력, 모델, 디바이스 지정.

## 3. 추론 하기 – main\_ssd.py

```
$ cd ~/sample  
$ python3 main_ssd.py
```

# OpenVINO 을 이용한 추론 – Mobilenet-SSD 모델

MNIST LeNet은 0-9까지 숫자를 학습시킨 image classification 모델로 가볍게 사용이 가능합니다.

Caffe 에서 Pre-trained된 model mnist을 이용하여 OpenVION에서 추론 하기.

- Model conversion/optimization
- 입출력, 네트워크 모델 지정 하기.
- 추론 및 지정한 이미지 저장 하기 .

# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 1. 모델 Conversion 하기

/home/intel/my\_model/ 의 위치에 Caffe 로 트레이닝을 해서 만들어진 모델 lenet.caffemodel과 lenet.prototxt가 있다.

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

모델 전환이 성공적으로 이루어 졌다면 ~/my\_model 폴더에 lenet.xml, lenet.bin file들이 생성된다.

# OpenVINO의 Deep Learning Toolkit을 사용하여 S/W 구현하기

## 2. Coding 하기 – 입출력, 모델, 디바이스 지정.

입력영상을 Load하기

1. Resizing

2. Color format conversion

Input width

Input height



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

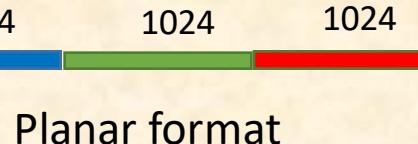
/* Copying data from image to the input blob */
cv::Mat ori_image, infer_image;
ori_image = cv::imread("/home/intel/my_model/input.jpg");
cv::resize(ori_image, infer_image, cv::Size(input_info->getDims()[0], input_info->getDims()[1]));

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();
```

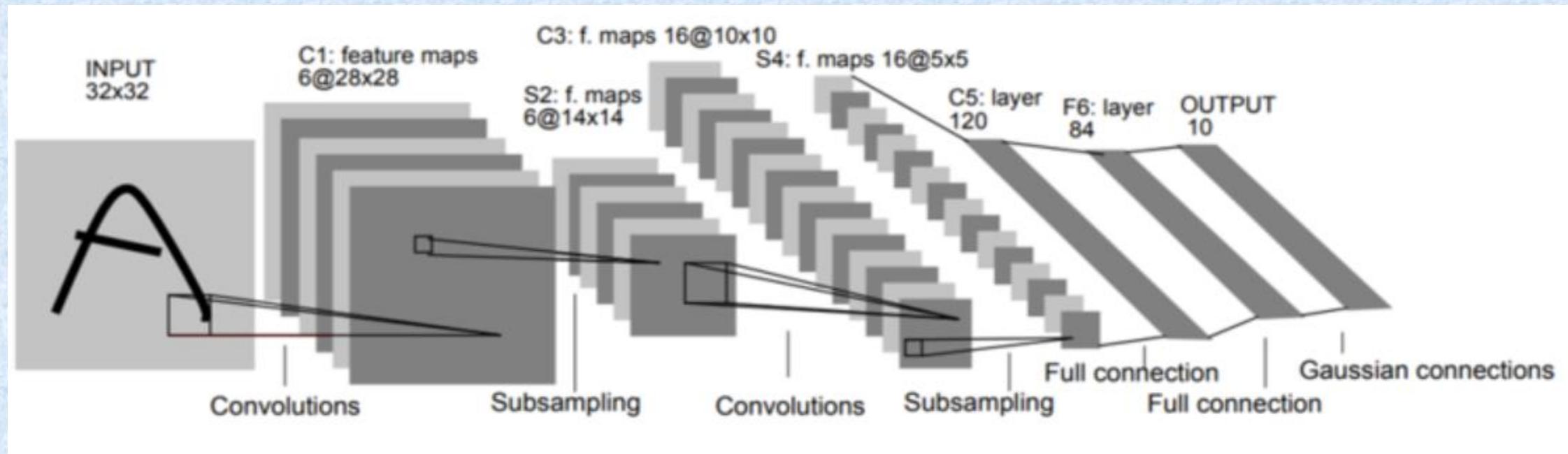
Network model  
Input Size in .xml



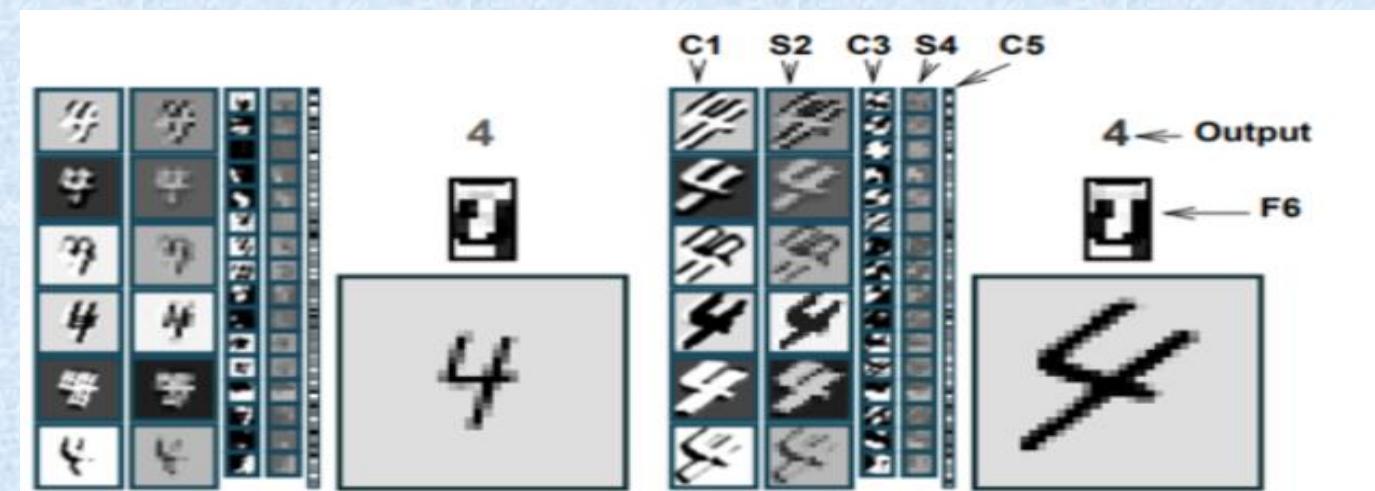
Inference Engine

Interleaved format

# LeNet-5 구조



입력 영상에 대해  
각각의 단계별 영상.



# OpenVINO Deep Learning Toolkit을 이용한 딥러닝 추론

## 3. 추론 하기 – lenet.py

```
$ cd ~/sample  
$ python3 lenet.py
```

### 1. 실행 및 결과

```
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 보여줍니다.
- 확률 값이며 모두 합치면 1이 됩니다.
- 가장 높은 확률을 가지는 Label을 선택합니다.
- 여기서는 “0”입니다.

ap  
apt install python-tk

1. 그림판에 마우스로 숫자를 써서 추론이 잘 되는지 확인 합니다 .

```
$ cd ~/sample  
$ mkdir img  
$ cp input.jpg ./img/test0.jpg  
$ cp input(1).jpg ./img/test1.jpg
```

```
$ cd ~/imobilenet  
$ source myvenv/bin/activate  
$ pip3 install pygame  
$ sudo apt-get install python3-tk  
$ cd ~/sample  
$ python3 lenet.py  
$ (myvenv) python3 app.py
```

A photograph showing a man from behind, wearing a white t-shirt, operating a drone from a control station. He is seated at a desk with three computer monitors displaying flight data and code. A laptop and a keyboard are also on the desk. In the foreground, there is a graphic of a quadcopter drone with glowing blue outlines against a dark background.

intel x LG전자 x KiSTi

2019 INTEL® AI DRONE FESTIVAL

# 드론 카메라을 이용한 영상 추론 하기

바탕화면에서 왼쪽에 있는 메뉴에서 파일 클릭하고 home/AI\_DRONE/DroneFestival\_MS로 이동한다음, 여기서 터미널을 열어서 하기코드를 실행

```
$ python3 Tello_Gui_M_ver.py True
```

드론에 베터리 삽입하는곳에 적혀 있는 본인의 텔로 드론의 ID 을 기억하고,  
베터리 삽입하고 전원을 켭니다.

(처음 드론을 사용하는경우 핸드폰에 텔로(Tello) 드론 어플을 설치해서 드론을  
Activation 시킵니다.) 핸드폰에서 텔로드론 연결을 종료하고.,

NUC mini PC 에서 드론 와이파이 ID를 찾아서 연결합니다.

GUI 에서 Connect 를 클릭하여 창에 ok 가 떨어지고 스크린에 동영상이 스트리밍 되면 드론을  
날려 영상을 추론할 준비가 된것입니다.

# 드론 카메라을 이용한 영상 추론 하기

## 드론 영상 추론하기

드론 카메라에 GUI 우측에 있는 영상을 비추어 추론이 잘 되는지 확인 해보세요.

사물이 인식이 되면 인식한 사물을 GUI 있는곳에 드론 영상을 캡춰해줍니다.

감사합니다.