



Inference with pre-trained models

Intel Korea

Haeyoung Lee

순서

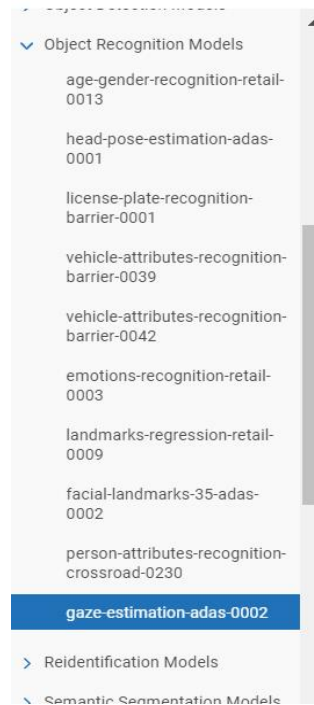
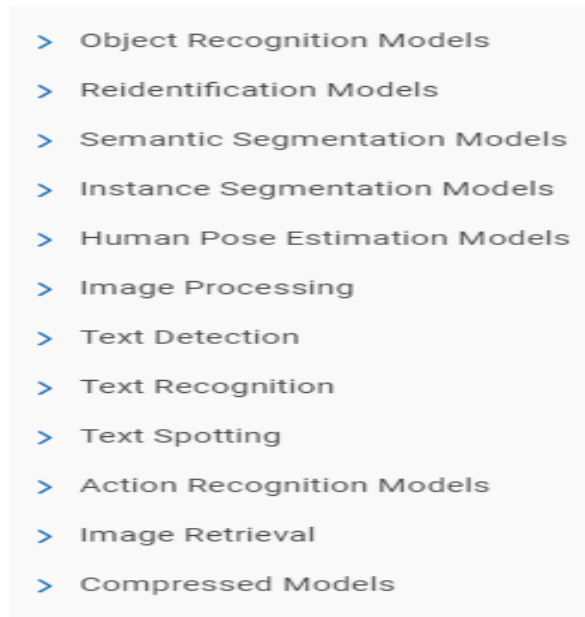
1. Intel Model Zoo 에서 사용하고자 하는 pre-trained model 을 고릅니다.
2. Model Downloader 을 통해서 pre-trained model 을 알맞은 형태로 다운로드 해 줍니다.
3. 샘플 추론 코드를 실행하여 다운받은 모델로 원하는 사진 또는 영상을 입력하여 추론해봅니다.

1. Pre-trained model 고르기

1. 먼저 사용하고자 하는 Pre-trained 모델을 골라야 합니다. 아래 링크에 들어가시면 다양한 Pretrained Models 의 상세한 설명을 볼 수 있으니 적절한 모델을 골라줍니다.

이번 자료에서는 gaze-estimation-adas-0002 를 골라 데모를 보여드리겠습니다.

https://docs.openvinotoolkit.org/latest/omz_models_intel_index.html

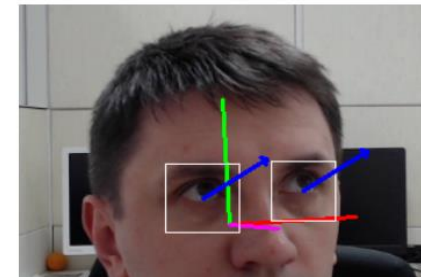


gaze-estimation-adas-0002

Use Case and High-Level Description

This is a custom VGG-like convolutional neural network for gaze direction estimation.

Example and Gaze Vector Definition



1. Pre-trained model 고르기

2. Intel 에서 제공하는 Open_model_zoo github 를 사용하셔도 좋습니다. 아래 링크에 들어가시면 다양한 데모를 read me 를 통해서 살펴볼 수 있습니다.

https://github.com/openvinotoolkit/open_model_zoo

위 링크의 다양한 데모 중 예시로 **Gaze estimation** 데모를 실행해보도록 하겠습니다.

IRDonch Merge pull request #1323 from opencv/release ... 366130d 25 days ago 3,320 commits		
ci	Update ci/requirements-*.txt using a prerelease build of OpenVINO 202...	2 months ago
demos	Remove links to the deleted Face Recognition demo	last month
models	Merge pull request #1224 from Aya-Zibra/patch-1	last month
tools	AC docs: added note about BF16 (#1289)	last month
.editorconfig	Publishing R3	2 years ago
classification_demo	Reverse changes in main.cpp	
common	Merge pull request #1074 from AnthonyQua	
crossroad_camera_demo	Replace rwmmap() calls with rmap and wmap	
gaze_estimation_demo	Merge pull request #1231 from IRDonch/zer	
human_pose_estimation_demo	Replace rmap with wmap in human_pose_es	
interactive_face_detection_demo	Fix some dangling pointer problems	

위 링크에 접속하셔서 **demos** 를 클릭해주고,
gaze_estimation_demo 로 들어가줍니다.

1. Pre-trained model 고르기

README.md

Gaze Estimation Demo

This demo showcases the work of gaze estimation model. The corresponding pre-trained model `gaze-estimation-adas-0002` is delivered with the product.

The demo also relies on the following auxiliary networks:

- `face-detection-retail-0004` OR `face-detection-adas-0001` detection networks for finding faces
- `head-pose-estimation-adas-0001`, which estimates head pose in Tait-Bryan angles, serving as an input for gaze estimation model
- `facial-landmarks-35-adas-0002`, which estimates coordinates of facial landmarks for detected faces. The keypoints at the corners of eyes are used to locate eyes regions required for the gaze estimation model
- `open-closed-eye-0001`, which estimates eyes state of detected faces.

For more information about the pre-trained models, refer to the [model documentation](#).

Other demo objectives are:

- Video/Camera as inputs, via OpenCV*
- Visualization of gaze estimation results, and, optionally, results of inference on auxiliary models

3. 스크롤을 내리시면 README.md 의 파일을 보여주는데, 여기에 Gaze Estimation Demo 에 대한 상세한 설명이 있습니다.

이번 데모를 실행하기 위해서는 총 5가지 모델이 필요합니다.

face-detection-retail-0004

facial-landmarks-35-adas-0002

gaze-estimation-adas-0002

head-pose-estimation-adas-0001

open-closed-eye-0001

Model Downloader을 통해 위 5개의 모델 xml, bin 파일을 다운로드 해줍니다. (다음 슬라이드 참고)

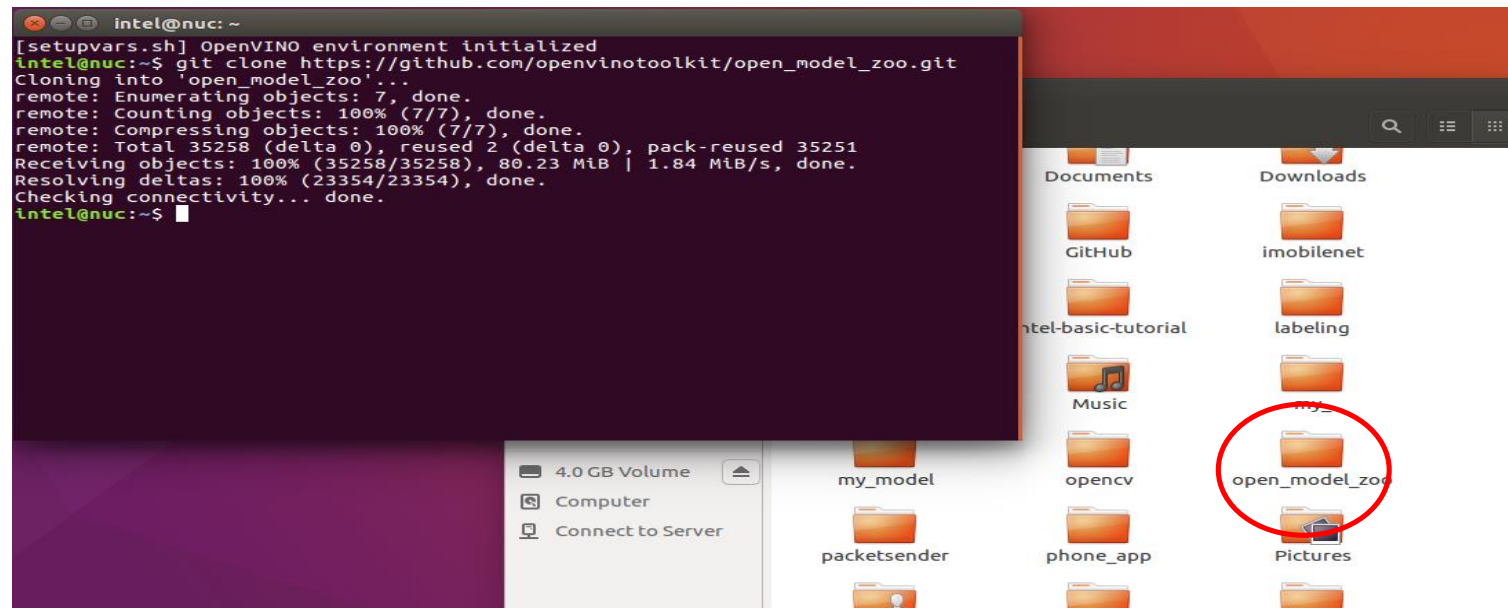
2. Model Downloader로 모델 다운로드하기

```
$ git clone https://github.com/openvinotoolkit/open\_model\_zoo.git
```

1. 위의 명령어를 입력해 필요한 코드들이 담겨있는 open_model_zoo 디렉토리를 다운로드 해줍니다.

이 때 인터넷이 연결되어 있어야 합니다. (원래 OpenVINO 를 설치하면 OpenVINO 경로에 open_model_zoo 가 있지만 경로가 복잡하므로 git clone 을 통해 home 밑에 다운로드해줍니다.)

다운로드가 완료되면 아래와 같이 home 디렉토리 아래 **open_model_zoo** 폴더를 확인할 수 있습니다.



2. Model Downloader로 모델 다운로드하기

2. 오른쪽과 같은 모델들을 다운로드하기 위해서는 OpenVINO 의 툴인 Model Downloader 를 사용해야 합니다.

- Model Downloader는 파이썬 파일로 존재하며, 위치는 아래 사진과 같이 **/home/intel/open_model_zoo/tools/downloader** 에 있습니다.

face-detection-retail-0004

facial-landmarks-35-adas-0002

gaze-estimation-adas-0002

head-pose-estimation-adas-0001

open-closed-eye-0001

3. 위의 경로로 들어가 downloader.py 를 실행하고, 명령을 입력할 때 '**--name**' 와 함께 원하는 모델을 넣어줍니다.

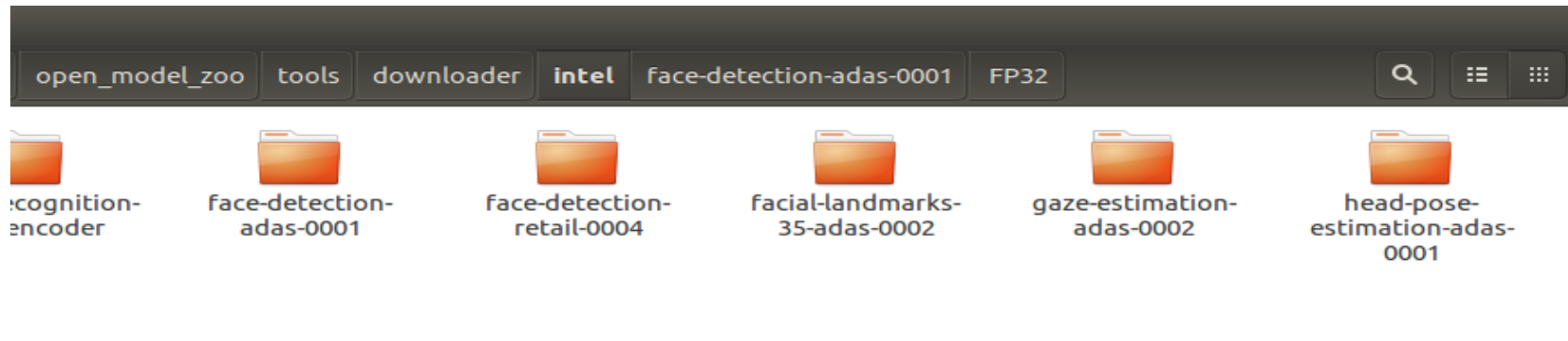
2. Model Downloader로 모델 다운로드하기

```
$ cd /home/intel/open_model_zoo/tools/downloader
```

```
$ sudo python3 downloader.py --name gaze-estimation-adas-0002 ( --name 뒤의 모델 이름 바꿔서 5개 모두 진행 )
```

4. 위와 같은 명령어를 입력해 앞의 5개의 모델을 전부 다운로드해줍니다.

다운로드를 완료하면 아래의 그림과 같이 downloader 아래 intel 폴더 안에 모델들이 다운로드된 것을 확인할 수 있습니다.

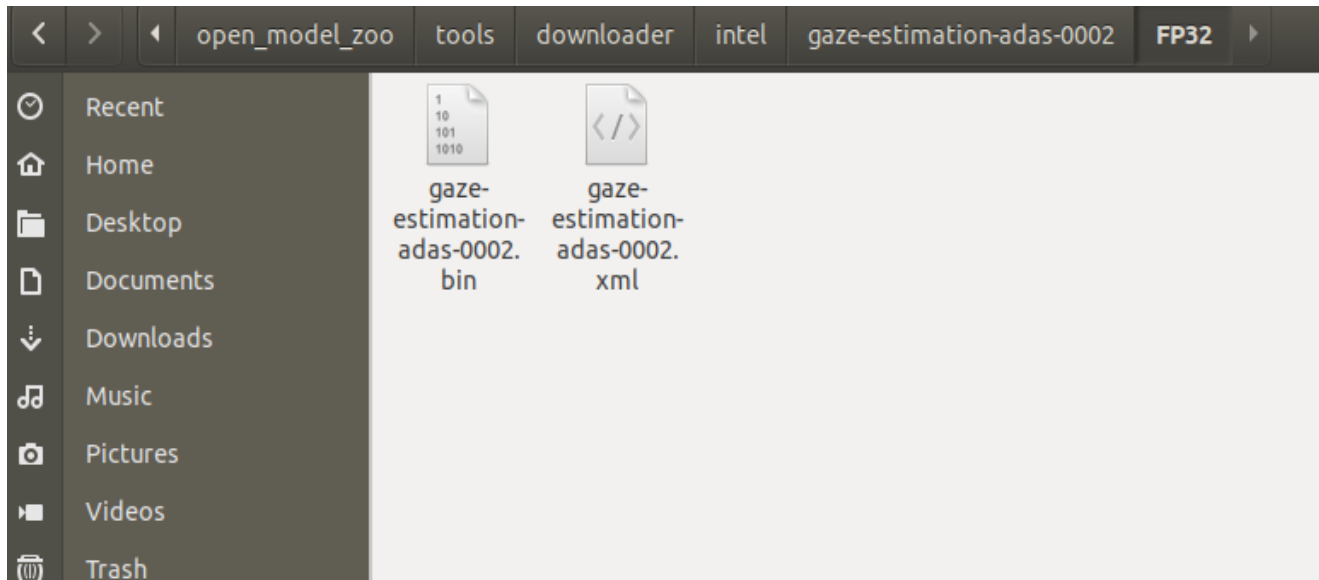


2. Model Downloader로 모델 다운로드하기

5. 다운로드가 완료된 모델을 클릭해보면 선택한 모델이 FP16, FP16-INT8, FP32 의 다양한 포맷으로 다운받아진 것을 확인할 수 있습니다.

6. 여기서 FP32 에 들어가 선택한 모델이 xml, bin 파일로 잘 변환된 것을 확인해줍니다.

(오픈비노를 이용해 추론하기 위해서는 Inference Engine이 인식할 수 있는 IR 파일로 변환이 되어야 합니다.)



7. 다운받은 모델들의 경로를 간단하게 하기 위해 FP32 아래의 xml과 bin파일을 **home/intel/model** 로 복사해줍니다.

2. Model Downloader로 모델 다운로드하기

```
$ sudo python3 converter.py --name open-closed-eye-0001 --mo /opt/intel/openvino/deployment_tools/model_optimizer/mo.py
```

8. 5개의 모델 중 'open-closed-eye-0001' 모델은 다른 모델과 다르게 xml, bin 파일로 다운로드되지 않고 onnx 파일로 다운로드됩니다. 따라서 OpenVINO 추론을 위해서는 onnx 파일을 다시 xml, bin파일로 변환해주어야 합니다. 이 onnx 파일은 아래 다운 완료된 코드에서 확인할 수 있듯이 downloader 밑에 intel 이 아닌 public 에 저장되어 있습니다.

```
intel@intel-NUC7i7BNH:~/open_model_zoo/tools/downloader$ python3 downloader.py --name open-closed-eye-0001
#####| Downloading models |#####

===== Downloading /home/intel/open_model_zoo/tools/downloader/public/open-closed-eye-0001/open-closed-eye.onnx
... 100%, 45 KB, 759 KB/s, 0 seconds passed

#####| Post-processing |#####

intel@intel-NUC7i7BNH:~/open_model_zoo/tools/downloader$
```

```
intel@intel-NUC7i7BNH:~/open_model_zoo/tools/downloader$ python3 converter.py --name open-closed-eye-0001
===== Converting open-closed-eye-0001 to IR (FP16)
Conversion command: /usr/bin/python3 -- /opt/intel/openvino_2020.4.287/deployment_tools/model_optimizer/mo.py --framework=onnx --data_type=FP16 --output_dir=/home/intel/open_model_zoo/tools/downloader/public/open-closed-eye-0001/FP16 --model_name=open-closed-eye-0001 '--mean_values=[127.0, 127.0, 127.0]' '--scale_values=[255, 255, 255]' --output=19 --input_model=/home/intel/open_model_zoo/tools/downloader/public/open-closed-eye-0001/open-closed-eye.onnx

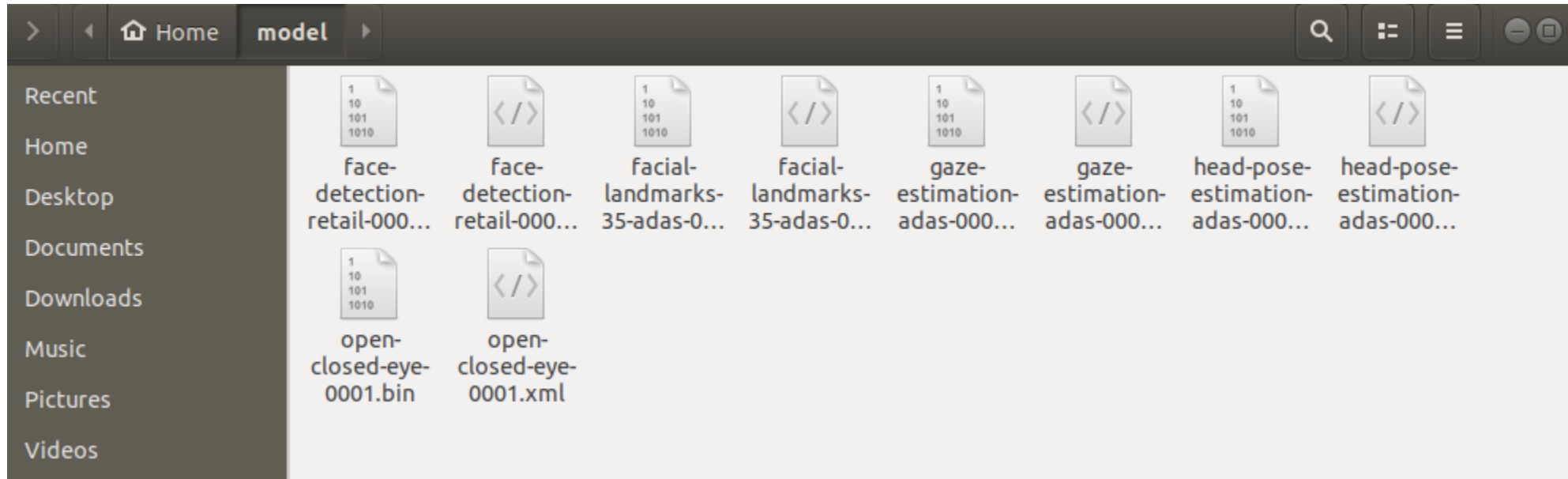
Model Optimizer arguments:
Common parameters:
- Path to the Input Model:      /home/intel/open_model_zoo/tools/downloader/public/open-closed-eye-0001/open-closed-eye.onnx
- Path for generated IR:       /home/intel/open_model_zoo/tools/downloader/public/open-closed-eye-0001/FP16
- IR output name:              open-closed-eye-0001
- Log level:                   ERROR
- Batch:                       Not specified, inherited from the model
- Input layers:                Not specified, inherited from the model
- Output layers:               19
- Input shapes:                Not specified, inherited from the model
- Mean values:                 [127.0, 127.0, 127.0]
- Scale values:                [255, 255, 255]
- Scale factor:                Not specified
- Precision of IR:             FP16
- Enable fusing:               True
- Enable grouped convolutions fusing: True
```

9. 위의 코드를 통해 onnx 파일을 xml, bin 파일로 변환해주면 public/open-closed-eye-0001 밑에 FP32 가 생성되고, FP32 밑에 open-closed-eye.xml 과 bin 파일을 /home/intel/model 로 복사해 줍니다.

2. Model Downloader로 모델 다운로드하기

10. 아래와 같이 /home/intel/model 아래에 5개의 모델이 다 준비가 되었다면,
모델 다운로드 단계는 완료되었습니다.

이제 5개의 모델을 이용한 gaze estimation demo 를 실행하기 위해 준비해야합니다.



3. Demo 실행하기

```
$ cd open_model_zoo/demos
$ mkdir build
$ cd build
$ cmake ../
$ make -j (에러 발생시 이 명령어 반복 실행)
```

```
intel@nuc:~/open_model_zoo/demos/build$ make -j
Scanning dependencies of target monitors
Scanning dependencies of target gflags_nothreads_static
[ 1%] Building CXX object thirdparty/gflags/CMakeFiles/gflags_nothreads_static.dir/src/gflags_completions.cc.o
[ 3%] Building CXX object thirdparty/gflags/CMakeFiles/gflags_nothreads_static.dir/src/gflags_reporting.cc.o
[ 3%] Building CXX object thirdparty/gflags/CMakeFiles/gflags_nothreads_static.dir/src/gflags.cc.o
[ 5%] Building CXX object common/monitors/CMakeFiles/monitors.dir/cpu_monitor.cpp.o
[ 5%] Building CXX object common/monitors/CMakeFiles/monitors.dir/memory_monitor.cpp.o
[ 6%] Building CXX object common/monitors/CMakeFiles/monitors.dir/presenter.cpp.o
[ 7%] Linking CXX static library ../../intel64/Release/lib/libgflags_nothreads.
```

1. open_model_zoo 아래 demos 폴더에는 다양한 데모들이 준비되어 있습니다. 이러한 데모들 중 c 로 개발된 데모를 실행하기 위해서는 먼저 데모파일들을 빌드해서 실행가능하도록 해주어야 합니다.

- 위의 명령어들을 하나씩 입력해서 데모를 빌드해줍니다.

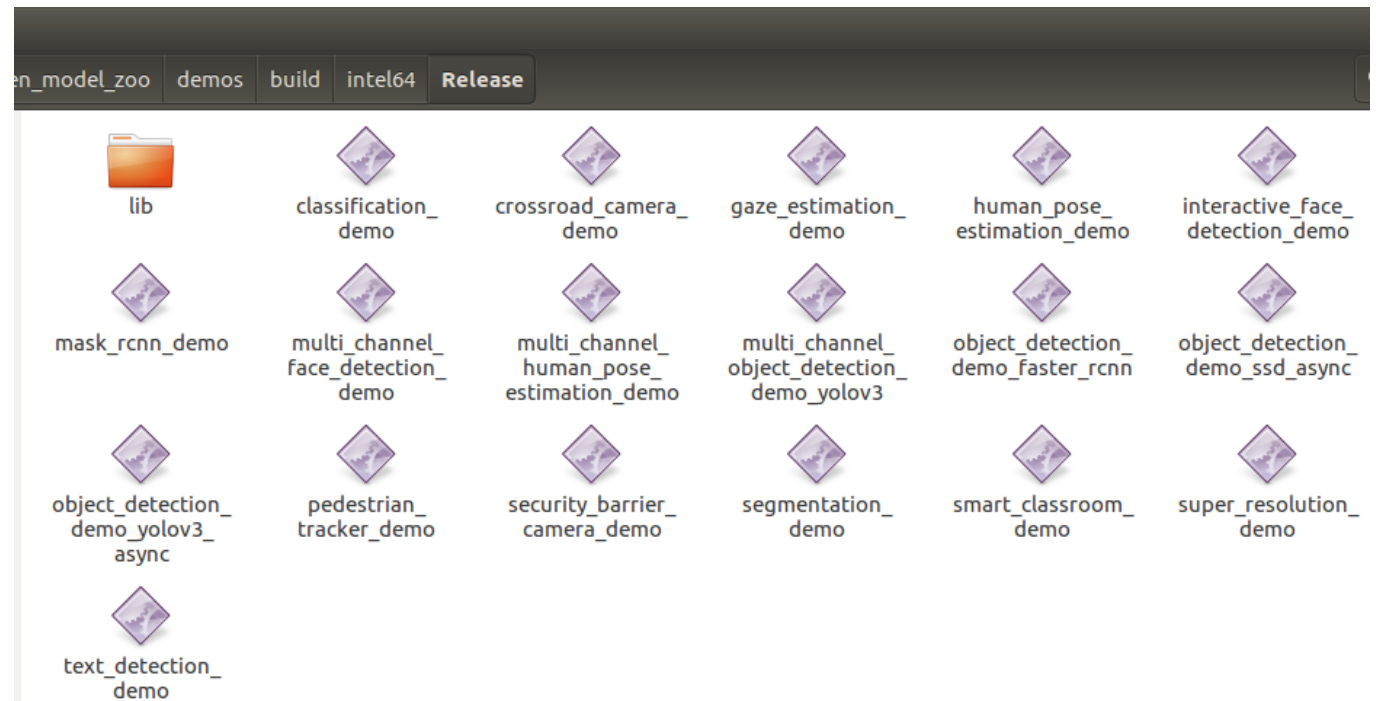
“make -j” 를 통해 데모를 하나씩 빌드해줄 수 있는데, 여기서 빌드가 완료된 데모 실행 파일은 **/home/intel/open_model_zoo/demos/build/intel64/Release** 에 저장됩니다.

- 이 때 make -j 의 명령어에서 에러가 발생할 수 있는데, 에러가 발생하면 다시 한번 “make -j” 명령어를 입력해주고, **100%** 다운로드가 완료될 때까지 반복해서 입력해줍니다.

3. Demo 실행하기

2. 빌드가 100% 완료되면 아래와 같이 `/home/intel/open_model_zoo/demos/build/intel64/Release` 아래 다양한 실행파일들이 빌드된 것을 확인할 수 있습니다.

```
[ 85%] Linking CXX executable ../intel64/Release/mask_rcnn_demo
[ 85%] Built target mask_rcnn_demo
[ 86%] Linking CXX executable ../intel64/Release/object_detection_demo_faster_rcnn
[ 86%] Built target object_detection_demo_faster_rcnn
[ 87%] Linking CXX executable ../intel64/Release/pedestrian_tracker_demo
[ 94%] Built target pedestrian_tracker_demo
[ 95%] Linking CXX executable ../intel64/Release/object_detection_demo_yolov3_async
[ 95%] Built target object_detection_demo_yolov3_async
[ 96%] Linking CXX executable ../intel64/Release/multi_channel_face_detection_demo
[ 97%] Linking CXX executable ../intel64/Release/multi_channel_human_pose_estimation_demo
[ 97%] Built target multi_channel_face_detection_demo
[ 97%] Built target multi_channel_human_pose_estimation_demo
[ 98%] Linking CXX executable ../intel64/Release/security_barrier_camera_demo
[ 98%] Built target security_barrier_camera_demo
[100%] Linking CXX executable ../intel64/Release/multi_channel_object_detection_demo_yolov3
[100%] Built target multi_channel_object_detection_demo_yolov3
Scanning dependencies of target ie_samples
[100%] Built target ie_samples
intel@nuc:~/open_model_zoo/demos/build$
```



3. Demo 실행하기

```
$ cd intel64/Release
$ ./gaze_estimation_demo -m /home/intel/model/gaze-estimation-adas-0002.xml -m_fd /home/intel/model/face-
detection-retail-0004.xml -m_hp /home/intel/model/head-pose-estimation-adas-0001.xml -m_lm
/home/intel/model/facial-landmarks-35-adas-0002.xml -m_es /home/intel/model/open-closed-eye-0001.xml
```

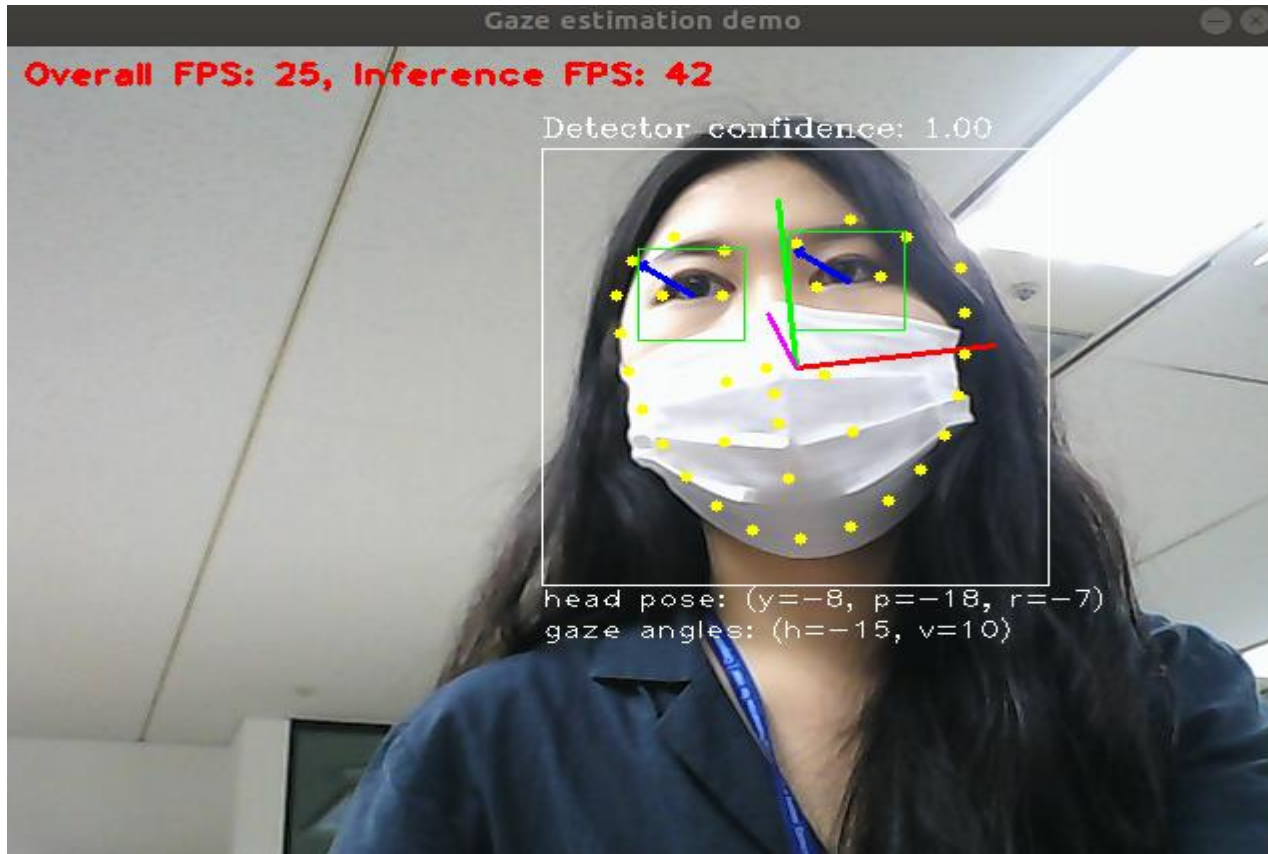
3. 이제 Gaze estimation DEMO 실행을 위해 위의 명령어를 입력해줍니다.

```
intel@intel-NUC7i7BNH:~/open_model_zoo/demos/build/intel64/Release$ ./gaze_estim
ation_demo -m /home/intel/model/gaze-estimation-adas-0002.xml -m_fd /home/intel/
model/face-detection-retail-0004.xml -m_hp /home/intel/model/head-pose-estimatio
n-adas-0001.xml -m_lm /home/intel/model/facial-landmarks-35-adas-0002.xml -m_es
/home/intel/model/open-closed-eye-0001.xml
InferenceEngine: 0x7f037f731030
[ INFO ] Parsing input parameters
[ INFO ] Reading input
[ INFO ] Loading device CPU
CPU
MKLDNNPlugin version ..... 2.1
Build ..... 2020.4.0-359-21e092122f4-releases/2020/4
```

3. Demo 실행하기

4. 아래와 같이 gaze estimation DEMO 가 잘 실행됩니다.

이 때 오른쪽 아래 사진과 같이 해당 키를 누르면 다양한 옵션이 on / off 됩니다. (A 누르면 전체 on, N 누르면 전체 off 등)



```
80 The demo allows you to control what information is display
81 The following keys are supported:
82 * G - to toggle displaying gaze vector
83 * B - to toggle displaying face detector bounding boxes
84 * O - to toggle displaying head pose information
85 * L - to toggle displaying facial landmarks
86 * E - to toggle displaying eyes state
87 * A - to switch on displaying all inference results
88 * N - to switch off displaying all inference results
89 * F - to flip frames horizontally
90 * Esc - to quit the demo
q1
```


* BERT Demo 실행

```
$ cd /home/intel/open_model_zoo/tools/downloader  
$ python3 downloader.py --name bert-small-uncased-whole-word-masking-squad-0001
```

- BERT demo 는 question-answering demo 로, 질문을 입력하면 대답을 해주는 데모입니다.
- 먼저 input 으로 질문을 할 내용이 담긴 웹사이트 주소와 모델을 주고, 질문을 입력하면 대답을 해줍니다.

1. 앞의 방법과 동일하게 이번 'Bert question-answering demo' 를 위해서 'bert-small-uncased-whole-word-masking-squad-0001' 모델을 다운로드해주고, 다운로드 완료된 FP32 밑의 xml과 bin 파일을 /home/intel/model/의 경로에 복사해줍니다.

* BERT Demo 실행

```
$ cd /home/intel/open_model_zoo/demos/python_demos/bert_question_answering_demo
$ python3 bert_question_answering_demo.py -m /home/intel/model/bert-small-uncased-whole-word-masking-squad-0001.xml -v /home/intel/open_model_zoo/models/intel/bert-small-uncased-whole-word-masking-squad-0001/vocab.txt -i https://en.Wikipedia.org/wiki/Artificial\_neural\_network -q 1
```

- 이 데모는 앞의 gaze_estimation_demo 와 달리 python 으로 개발된 데모입니다. 따라서 python_demo 에 가서 demo를 찾아 실행해줍니다.
- 또 input 으로 -v 가 필요한데, 이는 BERT 모델을 train 할 때 이용했던 단어 목록으로, /home/intel/open_model_zoo/models/intel/bert-small-uncased-whole-word-masking-squad-0001/vocab.txt 에서 확인할 수 있고, 인풋으로 입력해야합니다.

* BERT Demo 실행

```
intel@nuc: ~/open_model_zoo-master/demos/python_demos/bert_question_answering_demo
[setupvars.sh] OpenVINO environment initialized
intel@nuc:~/open_model_zoo-master/demos/python_demos/bert_question_answering_demo$ python3 bert_question_answering_demo.py -m /home/intel/Desktop/BERT/bert-small-uncased-whole-word-masking-squad-0001.xml -v /home/intel/open_model_zoo-master/demos/python_demos/bert_question_answering_demo/vocab.txt -i https://en.wikipedia.org/wiki/Artificial_neural_network -q 1
[ INFO ] Loading vocab file: /home/intel/open_model_zoo-master/demos/python_demos/bert_question_answering_demo/vocab.txt
[ INFO ] 30522 tokens loaded
[ INFO ] Get context from https://en.wikipedia.org/wiki/Artificial_neural_network
[ INFO ] Size: 35655 chars
[ INFO ] Context: Topics
Collective intelligence
Collective action
Self-organized criticality
Herd mentality
Phase transition
Agent-based modelling
Synchronization
Ant colony optimization
Particle swarm optimization
Swarm behaviour
Social network analysis
Small-world networks
Community identification
Centrality
Motifs
Graph Theory
Scaling
Robustness
Systems biology
Dynamic networks
Evolutionary computation
Genetic algorithms
Genetic programming
Artificial life
Machine learning
Evolutionary developmental biology
Artificial intelligence
Evolutionary robotics
Reaction-diffusion systems
Partial differential equations
Dissipative structures
Percolation
Cellular automata
Spatial ecology
Self-replication
Spatial evolutionary biology
Operationalization
Feedback
Self-reference
Goal-oriented
System dynamics
Sensemaking
```

- 데모를 실행하면 왼쪽과 같이 웹사이트 내용을 읽어오고, 질문을 입력하는 창이 뜹니다.
- 해당 내용에 맞는 질문을 입력하시면, 정답 확률과 함께 정답을 찾아주게 됩니다.

* BERT Demo 실행

```
intel@nuc: ~/open_model_zoo-master/demos/python_demos/bert_question_answering_demo
is deprecated. To access DataPtrs user need to use 'input_data' property of InputInfoPtr object
which can be accessed by 'input_info' property.
if ie_encoder.inputs.keys() != set(input_names) or ie_encoder.outputs.keys() != set(output_names):
[ INFO ] Loading model to the CPU
Type question (empty string to exit):what is a constant parameter whose value is set before the
learning process begins?
[ INFO ] Size: 15 tokens
[ INFO ] Sequence of length 384 is processed with 3.35 requests/sec (0.3 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.23 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 4.87 requests/sec (0.21 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.18 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 4.94 requests/sec (0.2 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.14 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.19 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.21 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.22 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.28 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.26 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 4.85 requests/sec (0.21 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.26 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.26 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.25 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.30 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.19 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.20 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.15 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.03 requests/sec (0.2 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.12 requests/sec (0.2 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.14 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.24 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.23 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.21 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.25 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.27 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.23 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.22 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.26 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.22 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.25 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.26 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.23 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.25 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.27 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.20 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.24 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.24 requests/sec (0.19 sec per request)
[ INFO ] Sequence of length 384 is processed with 5.21 requests/sec (0.19 sec per request)
[ INFO ] The performance below is reported only for reference purposes, please use the benchmark
tool (part of the OpenVINO samples) for any actual measurements.
[ INFO ] 40 requests of 384 length were processed in 7.86sec (0.2sec per request)
[ INFO ] ---answer: 0.60 hyperparameter
[ INFO ] A hyperparameter is a constant parameter whose value is set before the learning process
begins
[ INFO ] ---answer: 0.59 weight
[ INFO ] Neurons and edges typically have a weight that adjusts as learning proceeds
[ INFO ] ---answer: 0.24 weight
[ INFO ] Each link has a weight, which determines the strength of one node's influence on
another
Type question (empty string to exit):
```

네이버 통합검색 | '아마존-MS-인텔-엔디비아' | Artificial neural network

Wikipedia.org/wiki/Artificial_neural_network

ANNs are composed of **artificial neurons** which are conceptually derived from biological **neurons**. Each artificial neuron has inputs and produce a signal which can be sent to multiple other neurons. The inputs can be the feature values of a sample of external data, such as images or documents, or the outputs of other neurons. The outputs of the final *output neurons* of the neural net accomplish the task, such as recognizing an object in an image.

To find the output of the neuron, first we take the weighted sum of all the inputs, weighted by the *weights* of the *connections* from the inputs to the neuron. We then add a *bias* term to this sum. This weighted sum is sometimes called the *activation*. This weighted sum is then passed through a (usually nonlinear) *activation function* to produce the output. The initial inputs are external data, such as images and documents. The ultimate outputs accomplish the task, such as recognizing an object in an image.^[39]

Connections and weights [edit]

The network consists of connections, each connection providing the output of one neuron as an input to another neuron. Each connection is assigned a weight that represents its relative importance.^[37] A given neuron can have multiple input and output connections.^[40]

Propagation function [edit]

The *propagation function* computes the input to a neuron from the outputs of its predecessor neurons and their connections as a weighted sum.^[37] The *propagation function* can be added to the result of the propagation.^[41]

Organization [edit]

The neurons are typically organized into multiple layers, especially in **deep learning**. Neurons of one layer connect only to neurons of the immediate preceding and immediately following layers. The layer that receives external data is the *input layer*. The layer that produces the ultimate result is the *output layer*. In between them are zero or more *hidden layers*. Single layer and unlayered networks are also used. Between two layers, multiple connections are possible. They can be *fully connected*, with every neuron in one layer connecting to every neuron in the next layer. They can be *pooling*, where multiple neurons in one layer connect to a single neuron in the next layer, thereby reducing the number of neurons in that layer.^[42] Neurons with only one connection to the next layer form a *directed acyclic graph* and are known as *feedforward networks*.^[43] Alternatively, networks that allow connections between neurons in the same or previous layers are known as *recurrent networks*.^[44]

Hyperparameter [edit]

Main article: *Hyperparameter (machine learning)*

A hyperparameter is a constant **parameter** whose value is set before the learning process begins. The values of parameters are derived via learning. Examples of hyperparameters include **learning rate**, the number of hidden layers and batch size.^[45] The values of some hyperparameters can be dependent on those of other hyperparameters. For example, the size of some layers can depend on the overall number of layers.

Learning [edit]

This section includes a **list of references**, related reading or **external links**, but its sources remain **unclear because it lacks inline citations**. Please help to **improve** this section by **introducing more precise citations**. (August 2019) (Learn how and when to remove this template message)

See also: *Mathematical optimization, Estimation theory, and Machine learning*

Learning is the adaptation of the network to better handle a task by considering sample observations. Learning involves adjusting the weights (and biases) of the network. This is done by minimizing the observed errors. Learning is complete when examining additional observations does not usefully reduce the error. If after learning, the error rate is too high, the network typically must be redesigned. Practically this is done by defining a *cost function* that the output continues to decline, learning continues. The cost is frequently defined as a **statistic** whose value can only be approximated. The outputs are between the output (almost certainly a cat) and the correct answer (cat) is small. Learning attempts to reduce the total of the differences across the straightforwared application of *optimization theory and statistical estimation*.

– Hyperparameter
의 정의를 주고 무엇인지 물었을 때
답이
hyperparameter
라고 하는 예시입니다.



Thank you