

AIPro iNet Solution Demo Guide (API Version)

Part I. Demo Program Installation

Part II. iNet Solution Guide

System Requirement & Dependency

Category	Content
Models	Object Detection, Object Tracking, Person Attribute Recognition, Pose Estimation, Action Recognition
Program Language	C/C++
OS	Windows 11
Environment	Visual Studio 2022 CUDA 11.6.2 cuDNN 8.4.0
Demo Dependency	OpenCV-4.5.5(included), TensorRT-8.4.2.4 (included), Openvino(included)
GPU	RTX 2070 or newer

Date: Sept. 2022

Author: Chun-Su Park

E-mail: cspk@skku.edu

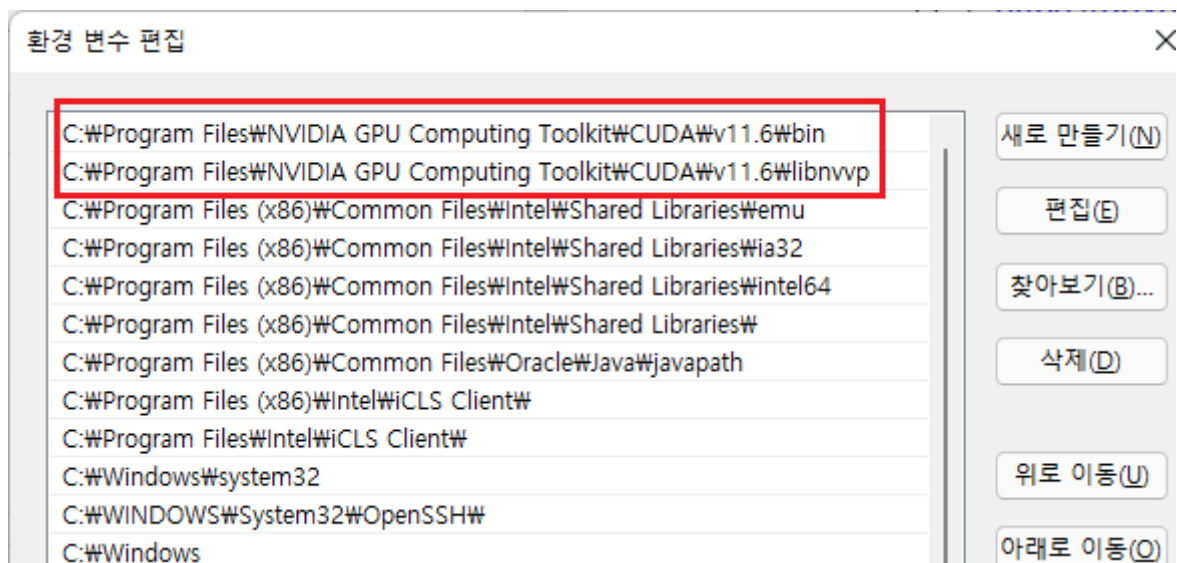
Part I. Demo Program Installation

1. Visual Studio 2022 Installation

- A. Install Community Version(free)
- B. (Important!!) You must install Visual Studio before installing CUDA and cuDNN

2. CUDA Installation

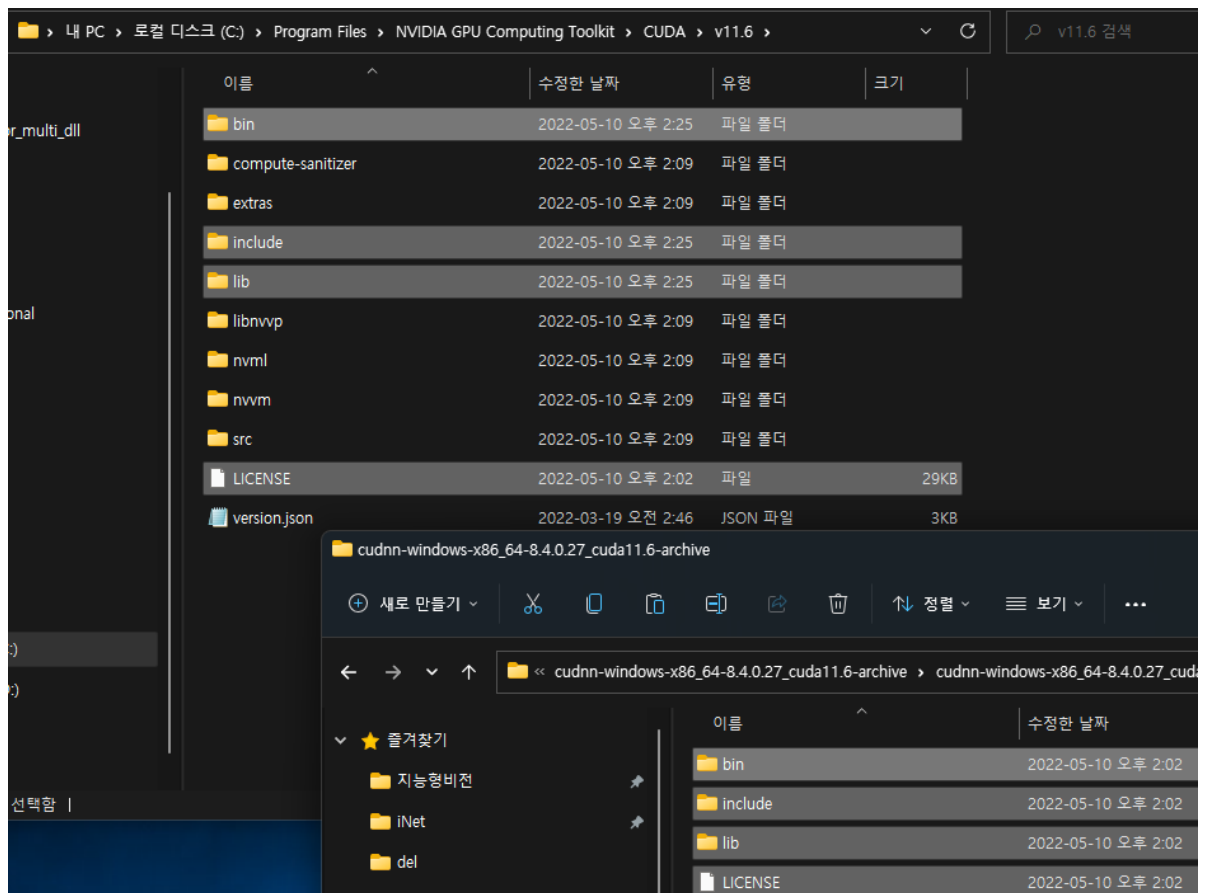
- A. Install CUDA Toolkit 11.6.2
 - Link: <https://developer.nvidia.com/cuda-toolkit-archive>
 - File name: cuda_11.6.2_511.65_windows.exe
 - Use default path
- B. Add the followings to PATH
 - C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.6\bin
 - C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.6\libnvvp



3. cuDNN Installation

- A. Install cuDNN 8.4.0
 - Link: <https://developer.nvidia.com/rdp/cudnn-download>
 - File name: cudnn-windows-x86_64-8.4.0.27_cuda11.6.exe
 - You need to sign up to NVIDIA to download cuDNN

- Unzip the downloaded file. Then, copy and paste bin, include, and lib directories to “C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.6”



B. Install zlibwapi.dll

- cudnn 8.4 uses zlibwapi.dll of zlib internally
 - If zlibwapi.dll is not installed, you can see the following error message

“Could not locate zlibwapi.dll. Please make sure it is in your library path!”

- Download zlib123dllx64.zip by using the ZLIB_DLL link in the following page
 - <https://docs.nvidia.com/deeplearning/cudnn/install-guide/index.html#install-zlib-windows>

3.1.3. Installing zlib

zlib is a data compression software library that is needed by cuDNN.

Procedure

1. Download and extract the zlib package from **ZLIB DLL**. Users with a 32-bit machine should download

Note: If using Chrome, the file may not automatically download. If this happens, right-click the link

2. Add the directory path of `zlibwapi.dll` to the environment variable PATH.

- Unzip `zlib12dllx64.zip`. Then, copy `zlibwapi.dll` in the `dll_x64` directory to "C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.6\bin"

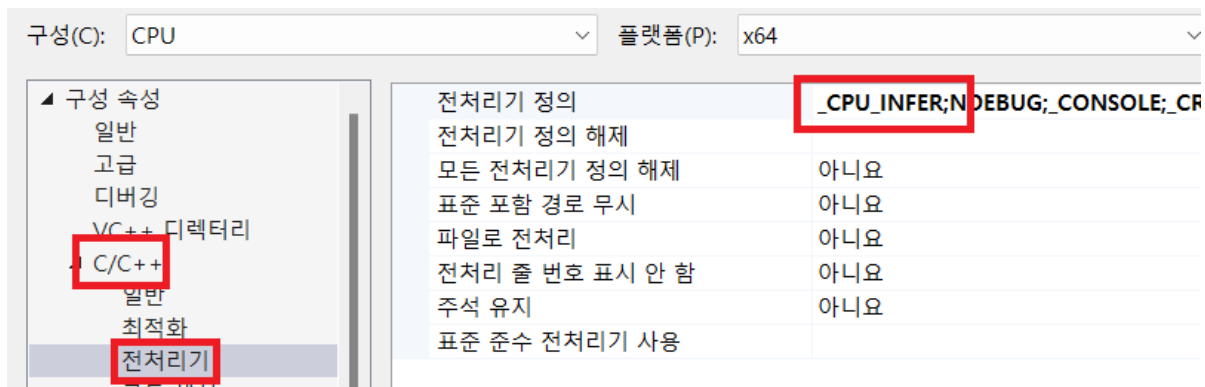
4. Visual Studio 2022 Setting

A. Clone or download iNet-API-Demo repository

- Repository Address: <https://github.com/AIProCo/iNet-API-Demo>

B. Open iNet-API-Demo.sln and set environment

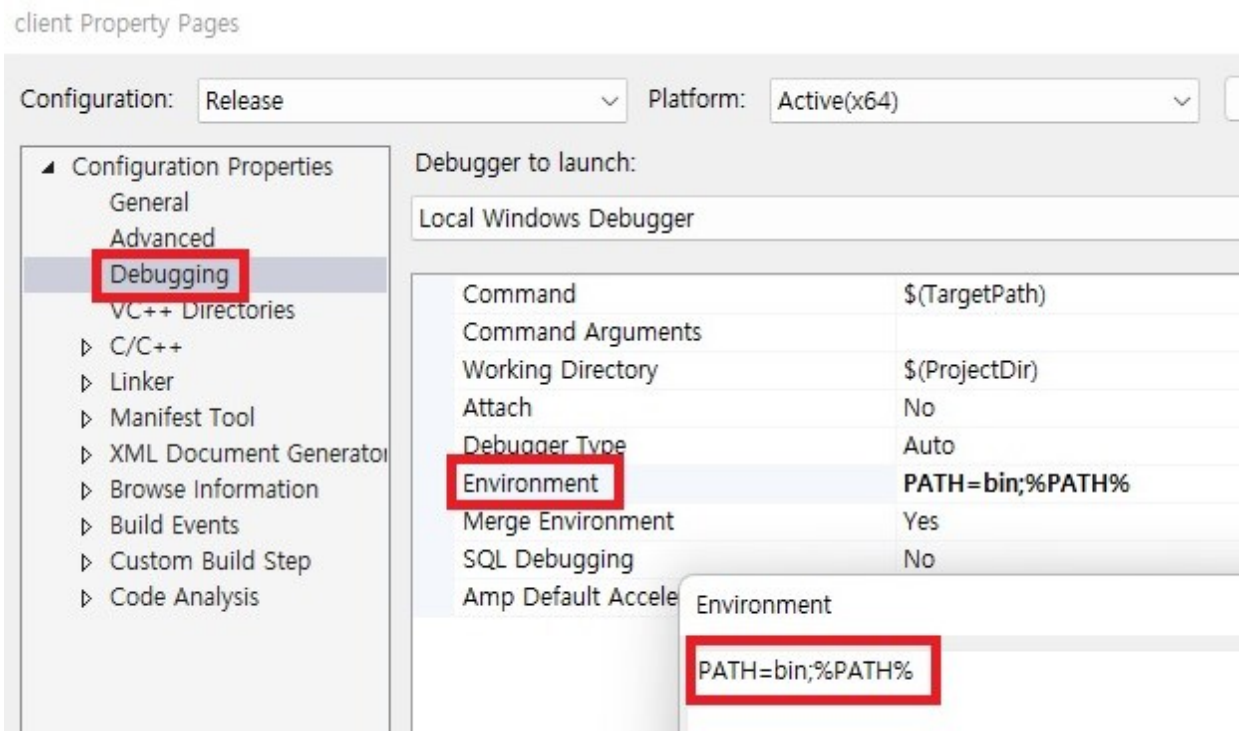
- Use the solution configuration "Release" if you are using the Nvidia GPU to infer the model, or use the "CPU" if you are using a CPU or Intel internal GPU
 - Debug configuration is not supported
- Fix the preprocessor of the CPU solution configuration
 - Path: C/C++ → Preprocessor → Preprocessor definition
 - Enter "_CPU_INFER" preprocessing statement



- From 4.C, apply to both Release and CPU configurations

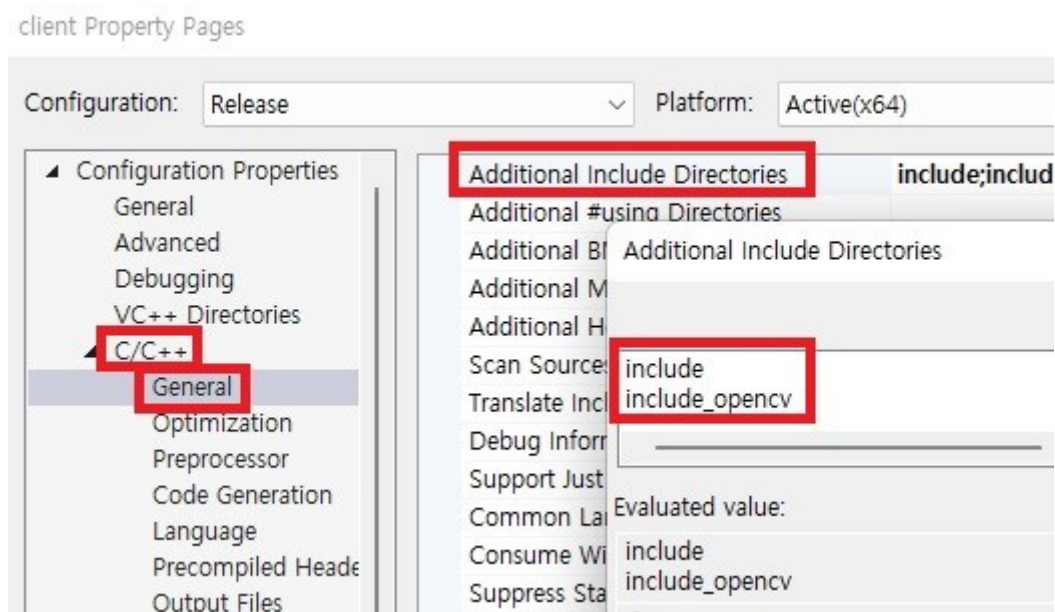
C. Modify the local PATH variable (the system PATH variable is not affected)

- Path: Properties → Debugging → Environment
- Enter "bin" directory to the PATH variable
 - Example: `PATH=bin;%PATH%`



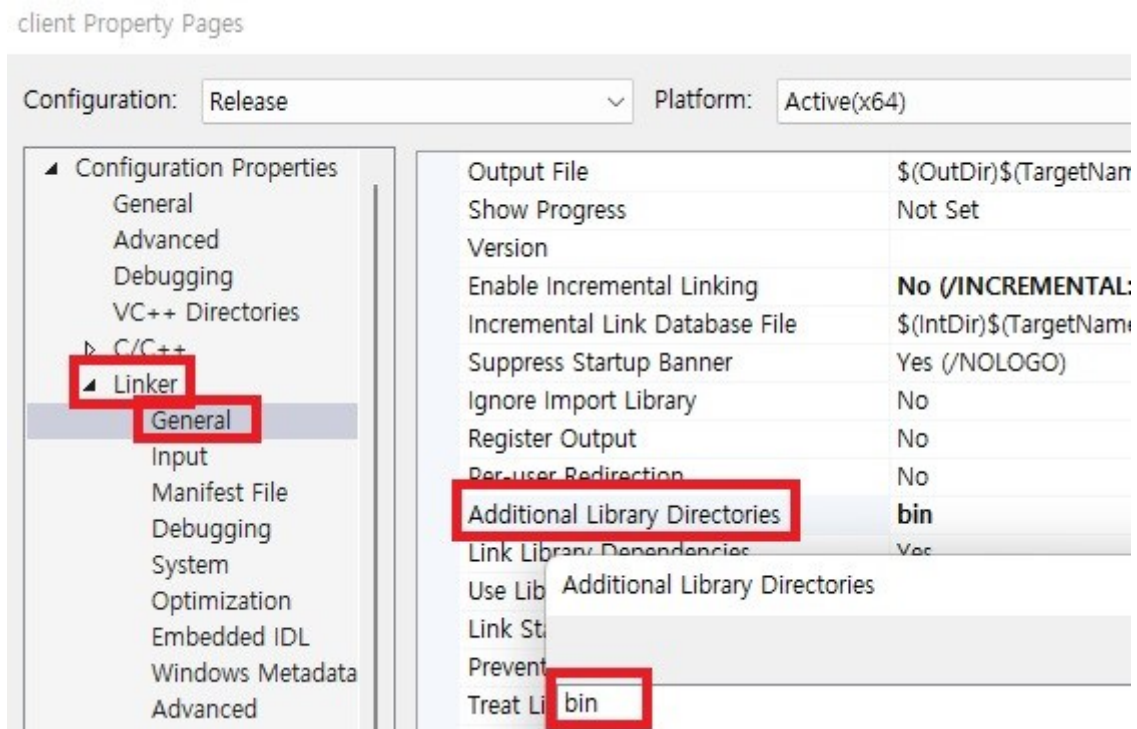
D. Modify Additional include Directories

- Path: Properties → C/C++ → General → Additional include Directories
- Enter "include" and "include_opencv"



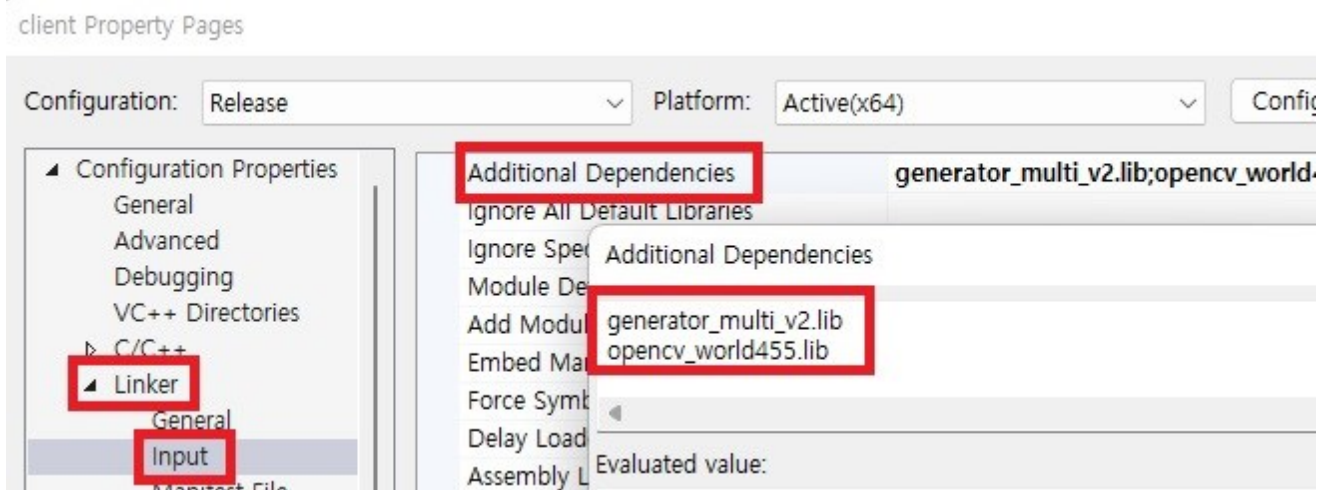
E. Modify Additional Library Directories

- Path: Properties → Linker → General → Additional Library Directories
- Enter "bin"



F. Modify Additional Dependencies

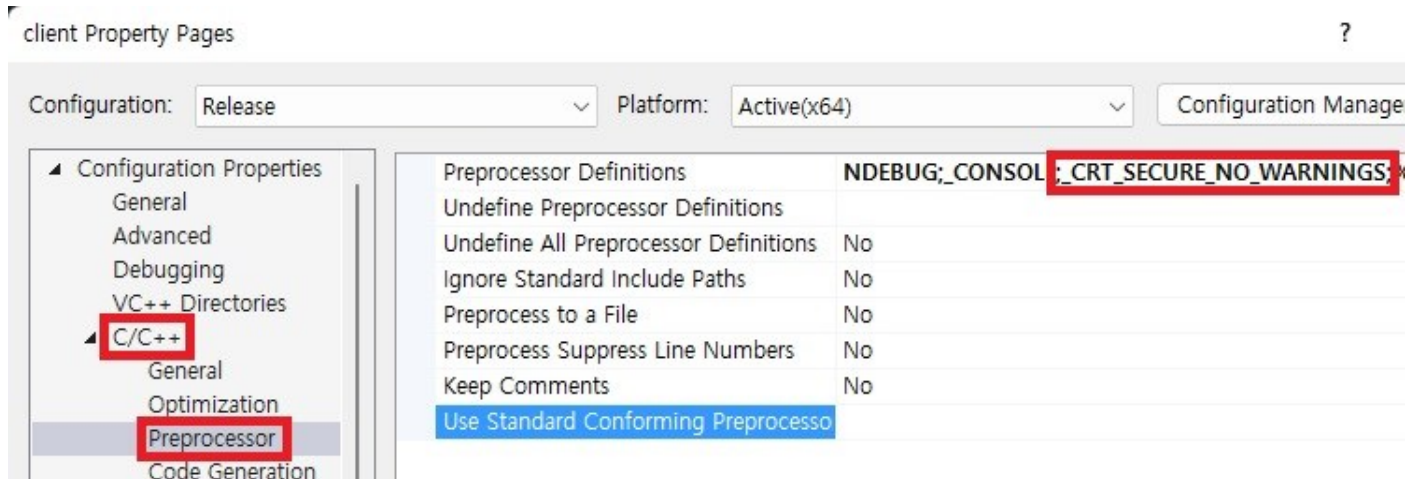
- Path: Properties → Linker → General → Additional Dependencies
- Enter “generator_multi_b.lib” and “opencv_world455.lib”



G. Disable security error

- Disable forced MS security functions usage
 - Error Message: error C4996: 'localtime': This function or variable may be unsafe.
- Path: Properties → C/C++ → preprocessor → Preprocessor Definitions

- Enter “_CRT_SECURE_NO_WARNINGS” to Preprocessor Definitions



5. Install required directories

- Download and unzip the following zip file. Then, copy and paste bin, inputs, and videos directories to the solution directory (the directory including the .sln file):
 - Please refer to the link in README.md of the github repository.

6. Set Release or CPU mode. Then, run the solution.

Part II. iNet Solution Developer Guide

1. Solution Introduction

- AIPro iNet Solution proceeds in four major steps, Initialization, Execution-1, Execution-2, and Destruction. The functions and details corresponding to each step are as follows:

Step	API Function	Content
Initialization	initModel()	<ul style="list-style-type: none"> - Initialize models and internal memory required to run the solution
Execution-1	runModel()	<ul style="list-style-type: none"> - Receive a batch of frames and perform inference - Responsible for object detection, tracking, counting (Line & Zone), PAR (Pedestrian attribute recognition), Pose estimation, and Action recognition. - Fill out detected object boxes, tracking IDs, counting results, PAR info, detected Skeleton information, and action IDs to the DetBox object
Execution-2	runModelFD()	<ul style="list-style-type: none"> - Receive a batch of frames and perform inference - Responsible for detecting Fire and Smoke objects - Fill out detected FD objects the FireBox object
Destruction	destroyModel()	<ul style="list-style-type: none"> - Destroy models and free memory

- The initialization and destruction functions are called once at the start and end of the program, respectively. Execution proceeds by repeatedly calling runModel() and runModelFD() for each batch of frames



<Figure> Flowchart of iNet Solution

2. Program development using the iNet solution

- Basically, the iNet solution parses the config.json file to create a Config object (cfg) and uses it to operate the entire solution. In order to develop a program using the

solution, the developer should modify the `parseConfigAPI()` function depending on each application.

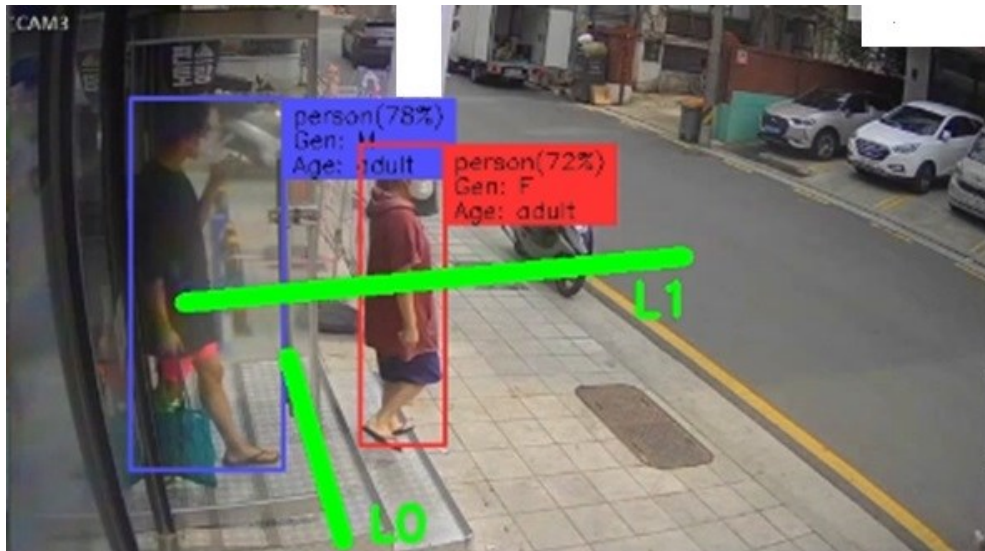
- It is recommended not to modify constant values in `parseConfigAPI()`
- After creating a `cfg` object that fits the application using both the json file and data extracted during application operation, initialization, execution, and destruction steps should be performed in the same way as the example code

3. Counting Information

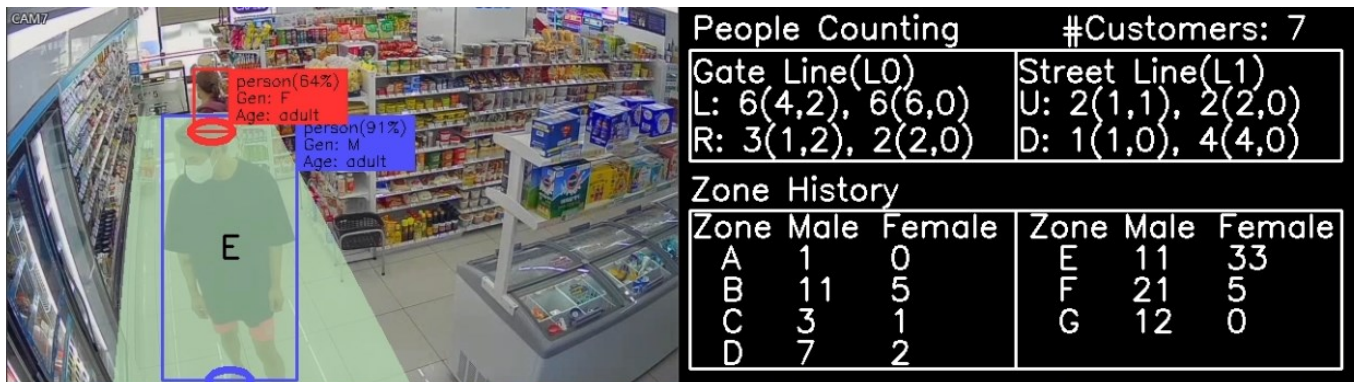
- By default, count the number of people passing through the `CntLine` and the number of people staying inside the `Zone`. When counting the number of people, count a total of six cases, considering gender (male and female) and age group (children, adult, and the elder)
 - Each counting information is stored in a 2x3 array in a [male/female][child/adult/elderly] manner and is output in the following format:

`M_Total(M_Child, M_Adult, M_Elder) F_Total(F_Child, F_Adult, F_Elder)`

- Each `CntLine` object counts the number of people passing in the Up/Down or Left/Right direction, and each `Zone` stores the number of people currently located inside and the number of people present (counting once a second)
 - `CntLine` class includes `TotalUL[2][3]` and `TotalDR[2][3]` arrays for counting
 - `Zone` class includes `curPeople[2][3]` and `hitMap[2][3]` arrays for hitmap
-



<Figure> Line Setting Example (L0 and L1)



<Figure> Line Counting & Zone Hitmap Example

4. API Functions

bool initModel(**Config** &cfg, **ODRecord** &odRcd, **FDRecord** &fdRcd)

Initialize model

- param cfg configuration struct
- param odRcd object detection record struct
- param fdRcd fire detection record struct
- return initialization result(true: success, false: fail)

bool runModel(**vector**<**vector**<**DetBox**>> &dboxesMul, **vector**<**Mat**> &frames, **vector**<**int**> &vchIDs, **vector**<**uint**> &frameCnts, **float** odScoreTh, **float** actScoreTh)

Run OD, PAR, Pose, and Action models for a frame batch

- param dboxesMul return detected dboxes of all video channels(vchIDs)
- param frames batch of frames
- param vchIDs vchIDs of batched frames
- param frameCnts frameCnts of batched frames

- param odScoreTh threshold for filtering low confident object detections
- param actScoreTh threshold for filtering low confident action recognitions
- return runModel result(true: success, false: fail)

```
bool runModelFD(vector< vector<FireBox>> &fbboxesMul, vector<Mat> &frames,  
vector<int> &vchIDs, vector<uint> &frameCnts, float fdScoreTh)
```

Run FD models for a frame batch

- param fbboxesMul return detected fbboxes of all video channels(vchIDs)
- param frames batch of frames
- param vchIDs vchIDs of batched frames
- param frameCnts frameCnts of batched frames
- param fdScoreTh threshold for filtering low confident detections
- return flag for the running result(true: success, false: fail)

```
bool destroyModel()
```

Destroy model

- param None
- return flag for destruction result(true: success, false: fail)

```
bool resetCntLineAndZone(ODRecord &odRcd)
```

Reset CntLine and Zone configuration

- param odRcd record struct
- return flag for reset(true: success, false: fail)

```
bool resetCntLineAndZoneRecord()
```

Reset CntLine and Zone record

- param None
- return flag for reset(true: success, false: fail)

```
bool resetFD (FDRecord &fdRcd)
```

Reset FD record

- param fdRcd record struct
- return flag for reset(true: success, false: fail)

```
bool resetFDRecord()
```

Reset fd record

- param None
- return flag for reset(true: success, false: fail)

5. Configuration of config.json

Name	Item	Value
global	apikey	Solution key (must use "aiprotect")
	frame_limit	Number of frames to be processed
	input_files	Input video files with path
	output_files	output videos files with path
od	score_th	Score value for object detection
fd	enable	FD inference On/Off
	score_th	Score value for fire detection
par	enable	PAR inference On/Off
pose	enable	Pose inference On/Off
	score_th	Score value for drawing Skeleton
act	enable	Action inference On/Off
	score_th	Score value for Action recognition
line	param	<p>Enter Counting Line information as follows:</p> <p>[line_id vchID x1 y1 x2 y2]</p> <ul style="list-style-type: none"> - line_id: unique ID - vchID: Video Channel ID - x1, y1: Point 1 - x2, y2: Point 2 <div data-bbox="715 1503 1345 1798" data-label="Diagram"> <p>Input video</p> <p>(x₁, y₁) (x₂, y₂)</p> </div> <p>(Counting example) UP: Male 3(1/2/0), Female 2(0/1/1) Down: Male 5(2/2/1), Female 1(0/1/0)</p>

zone	param	<p>Enter Zone information as follows:</p> <p>[zone_id vchID isRestricted x1 y1 x2 y2 x3 y3 x4 y4]</p> <ul style="list-style-type: none"> - zone_id: unique ID - vchID: Video Channel ID - isRestricted: flag for specifying a restricted area (internally not used) <ul style="list-style-type: none"> - x1 y1 x2 y2 x3 y3 x4 y4: four vertex coordinates(entered in consecutive directions)
-------------	-------	---

6. Person Attribute Recognition

- In the runModel() function, enter attributes of the PedAtts struct and additional member variables of the DetBox object

Attribute	Content
Gender	<p>Male/Female recognition (accuracy: about 93%)</p> <ul style="list-style-type: none"> - DetBox-PedAtts-atts[0]: 0:Male, 1:Female
Age	<p>Child/Adult/Elder recognition (accuracy: about 85%)</p> <ul style="list-style-type: none"> - DetBox-PedAtts-atts[1]: confidence to be child - DetBox-PedAtts-atts[2]: confidence to be adult - DetBox-PedAtts-atts[3]: confidence to be elder

All attributes	<pre> #define NUM_ATTRIBUTES 30 /// number of attributes #define ATT_GENDER 0 /// gender should be the #define ATT_AGE_CHILD 1 #define ATT_AGE_ADULT 2 #define ATT_AGE_ELDER 3 #define ATT_HAIR_LEN_SHORT 4 #define ATT_HAIR_LEN_LONG 5 #define ATT_UBODY_LEN_SHORT 6 #define ATT_UBODY_COL_BLACK 7 #define ATT_UBODY_COL_BLUE 8 #define ATT_UBODY_COL_BROWN 9 #define ATT_UBODY_COL_GREEN 10 #define ATT_UBODY_COL_GRAY 11 #define ATT_UBODY_COL_PINK 12 #define ATT_UBODY_COL_PURPLE 13 #define ATT_UBODY_COL_RED 14 #define ATT_UBODY_COL_WHITE 15 #define ATT_UBODY_COL_YELLOW 16 #define ATT_UBODY_COL_OTHER 17 #define ATT_LBODY_LEN_SHORT 18 #define ATT_LBODY_COL_BLACK 19 #define ATT_LBODY_COL_BLUE 20 #define ATT_LBODY_COL_BROWN 21 #define ATT_LBODY_COL_GRAY 22 #define ATT_LBODY_COL_WHITE 23 #define ATT_LBODY_COL_OTHER 24 #define ATT_LBODY_TYPE_TROUSER_SHORT 25 #define ATT_LBODY_TYPE_SKIRT_DRESS 26 #define ATT_BACKPACK 27 #define ATT_BAG 28 #define ATT_HAT 29 </pre>
Movement	<p>Motion activity (for detecting fainting, falling, sleep, etc)</p> <ul style="list-style-type: none"> - DetBox-distVar: box center variation after temporal pooling
Time to appear	<p>For calculating the time spent after appear</p> <ul style="list-style-type: none"> - DetBox-inTime: time when this object is detected
Previous position	<p>For estimating movement path</p> <ul style="list-style-type: none"> - DetBox-(rxP, ryP): reference position in the previous frame

7. Action Recognition

- In the runModelAck() function, enter the action information of the DetBox object
 - DetBox-actID: action ID in actIDMapping
 - DetBox-actConf: confidence of the current act

■ Action ID Mapping

17 actions (ID: Label)			
0: Hand on mouth	1: Pick up	2: Throw	3: Sit down
4: Stand up	5: Clapping	6: Reading/writing	7: Hand wave
8: Kick	9: Cross hands	10: Staggering	11: Fall down
12: Punch/Slap	13: Push	14: Walk	15: Squat down
16: Run			

8. Fire Detection

- Enter information about FireBox objects detected by the runModelFD() function into the fboxesMul object
 - Fire and Smoke Object Detection (2 class IDs)
- Enter the probability of fire and smoke in the fdRcd object, respectively

9. Average Inference Time

- Measure time delays inside runModel()
 - Part 0: OD + Tracking + PAR
 - Part 1: Pose Estimation + Action Recognition
 - Input Video Resolution: FHD, Model Input Resolution: 960x544, GPU: 2080Ti, CPU: i9-10900X@3.70GHz, Batch Size: 1 frame

OD+Track+PAR: 18ms/frame, POSE+ACT: 18ms/frame


```
C# Microsoft Visual Studio Debug Console
Frame 98>      OD+Track+PAR: 20ms      POSE+ACT: 20ms
Frame 98>      OD+Track+PAR: 16ms      POSE+ACT: 16ms
Frame 99>      OD+Track+PAR: 18ms      POSE+ACT: 18ms
Frame 99>      OD+Track+PAR: 16ms      POSE+ACT: 16ms
Frame 99>      OD+Track+PAR: 17ms      POSE+ACT: 17ms
Frame 99>      OD+Track+PAR: 14ms      POSE+ACT: 14ms
Frame 100>     OD+Track+PAR: 20ms      POSE+ACT: 20ms
Frame 100>     OD+Track+PAR: 17ms      POSE+ACT: 17ms
Frame 100>     OD+Track+PAR: 17ms      POSE+ACT: 17ms
Frame 100>     OD+Track+PAR: 16ms      POSE+ACT: 16ms

Average Inference Time> OD+Track+PAR: 18ms      POSE+ACT: 18ms

Output file(s):
    videos/in0_o.mp4
    videos/in1_o.mp4
    videos/in2_o.mp4
    videos/in3_o.mp4

Terminate program!
```