AIPro Inc. Technical Documentation (Confidential)

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, Crowd Counting
Program Language	C/C++
OS	Windows 11
Environment	Visual Studio 2022 CUDA 12.1.0 cuDNN 8.9.6
Demo Dependency	OpenCV-4.9.0(included), TensorRT-8.6.1.6 (included), Openvino(included)
GPU	RTX 30xx

Date: Jan. 2024

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

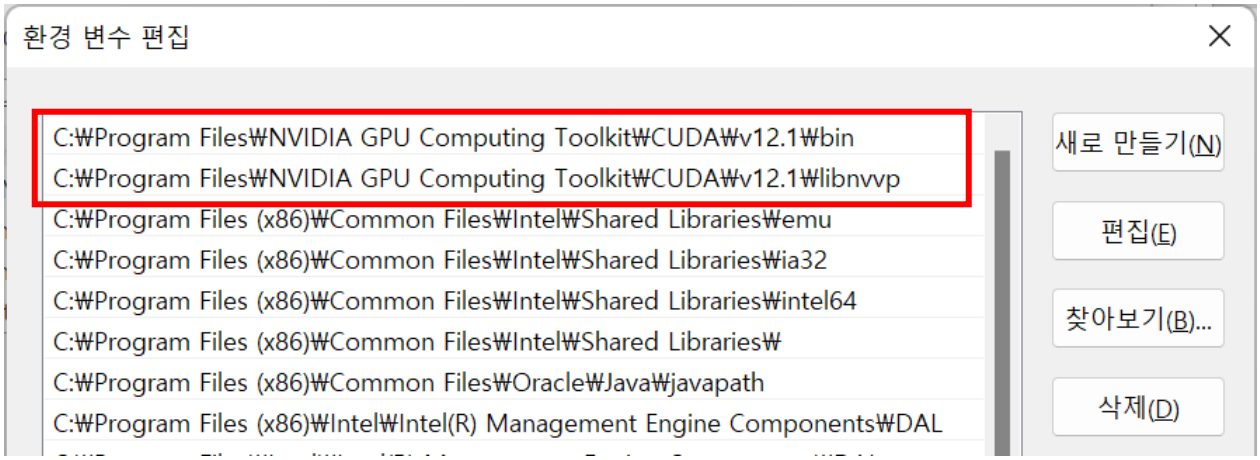
2. CUDA Installation

A. Install CUDA Toolkit 12.1.0

- Link: https://developer.nvidia.com/cuda-toolkit-archive
- File name: cuda_12.1.0_531.14_windows.exe
- Use default path

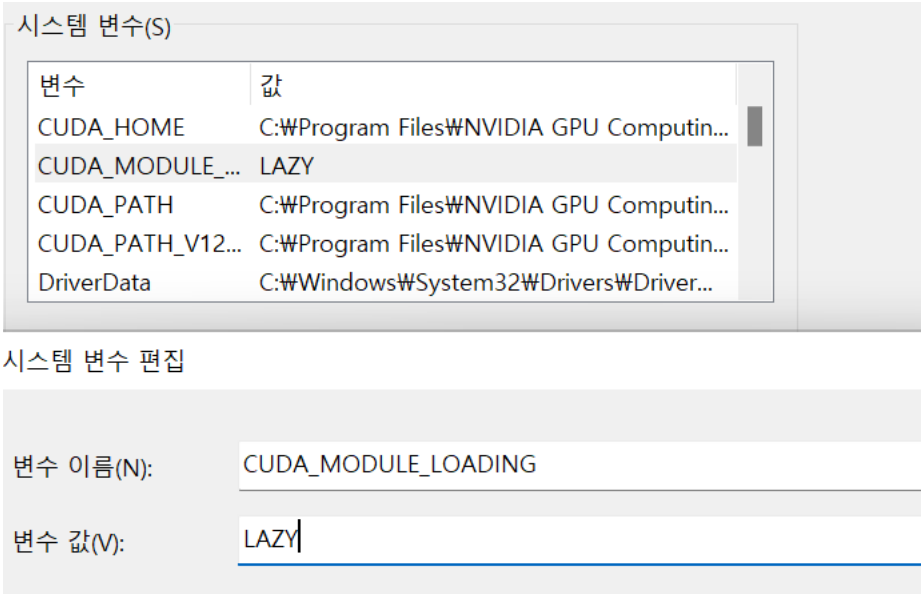
B. Add the followings to PATH

- C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1\bin
- C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1\libnvvp



C. Add Lazy loading environment variable

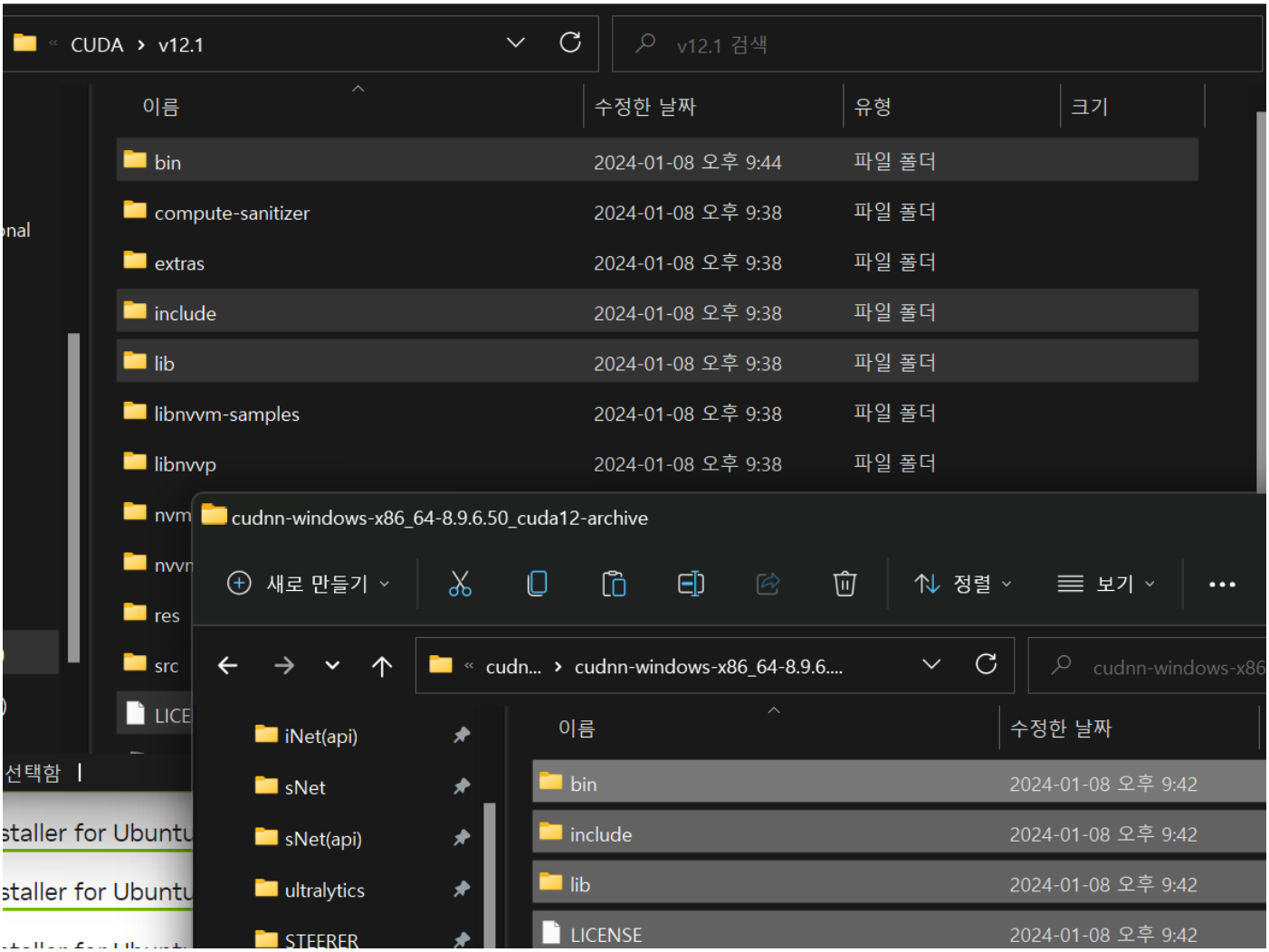
- Name: CUDA_MODULE_LOADING, Value: LAZY
- Reboot required after adding
- Link: https://docs.nvidia.com/cuda/cuda-c-programming-guide/#lazy-loading



3. cuDNN Installation

A. Install cuDNN 8.9.6

- Link: https://developer.nvidia.com/rdp/cudnn-download
- File name: cudnn-windows-x86_64-8.9.6.50_cuda12-archive.zip
  - 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\v12.1"



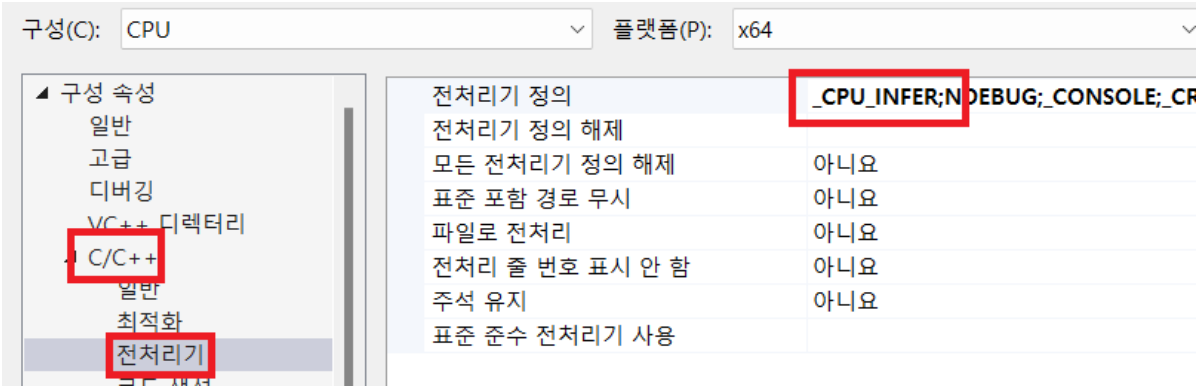
4. Visual Studio 2022 Setting

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

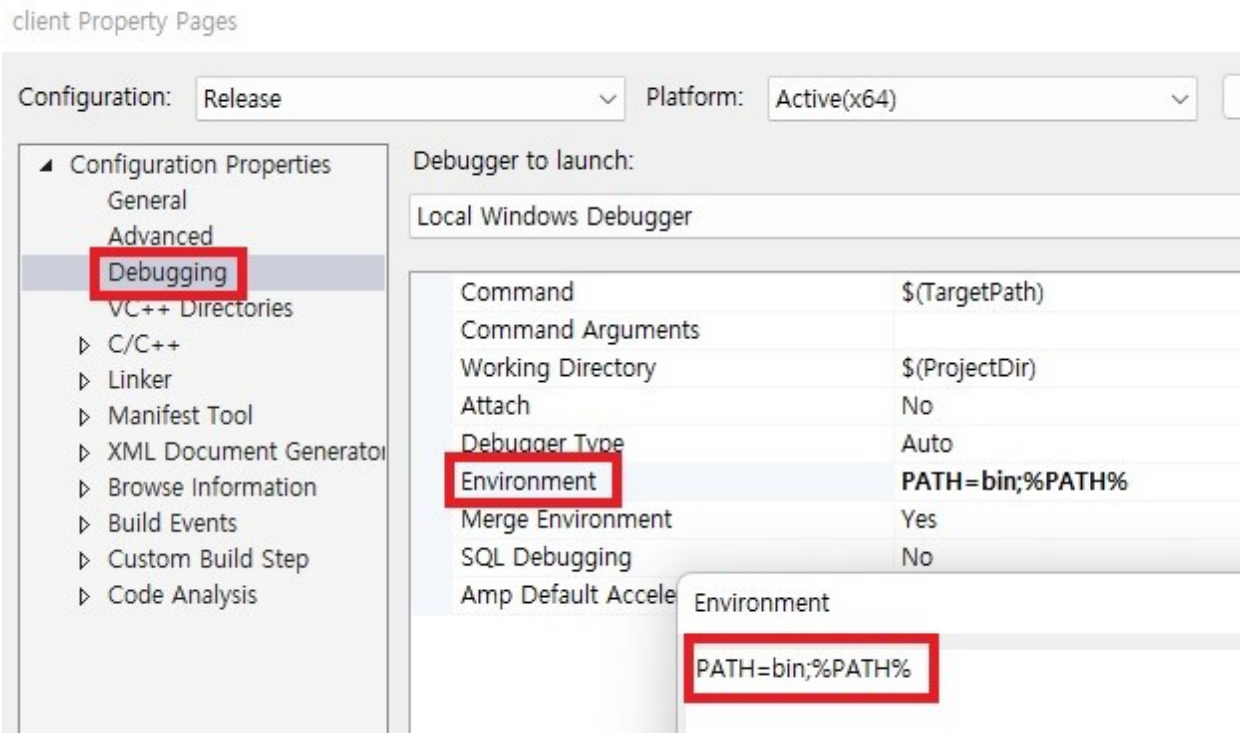
- Use the solution configuration "Release" if you are using the Nvidia GPU to infer the model, or use the "CPU" if you are using a CPU or Intel internal GPU
  - Debug configuration is not supported
- Fix the preprocessor of the CPU solution configuration
  - Path: C/C++ → Preprocessor → Preprocessor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



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

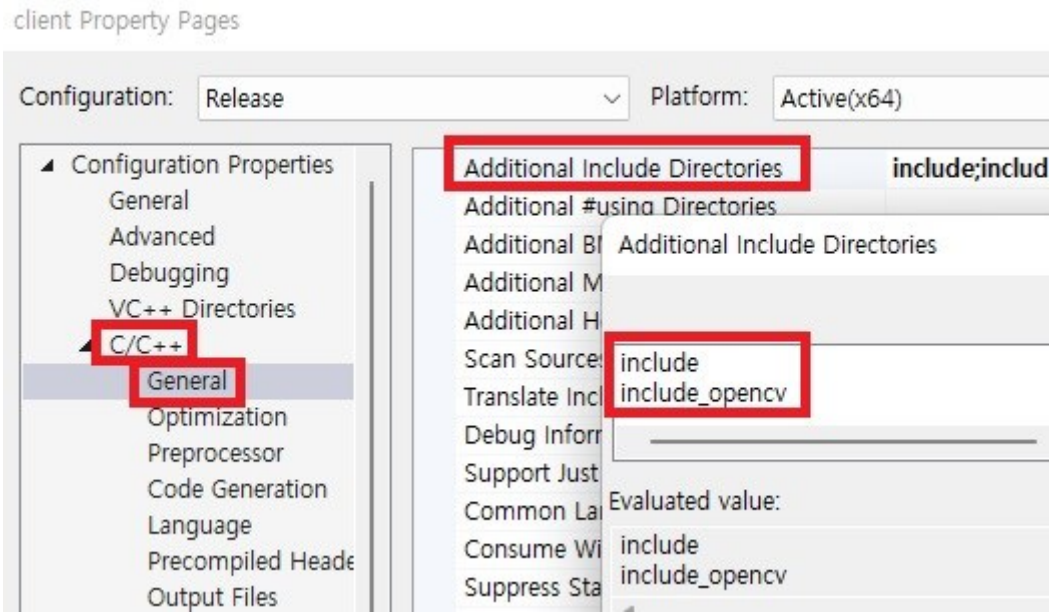
C. Modify the local PATH variable (the system PATH variable is not affected)

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



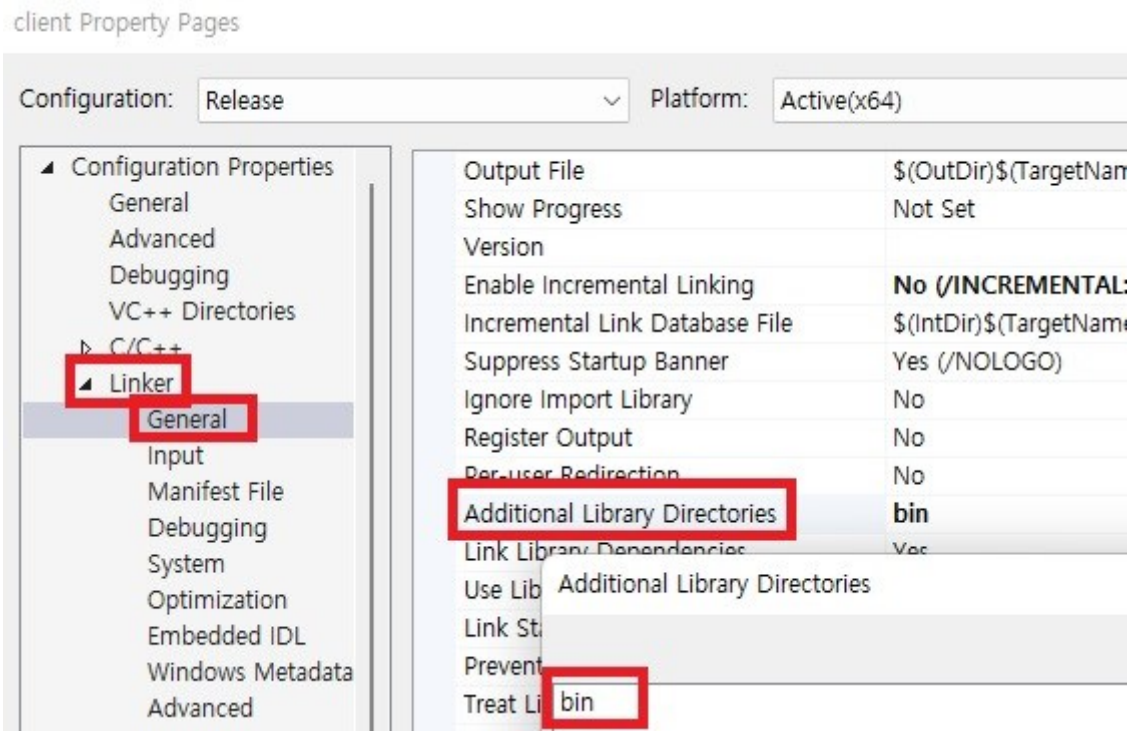
D. Modify Additional include Directories

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



E. Modify Additional Library Directories

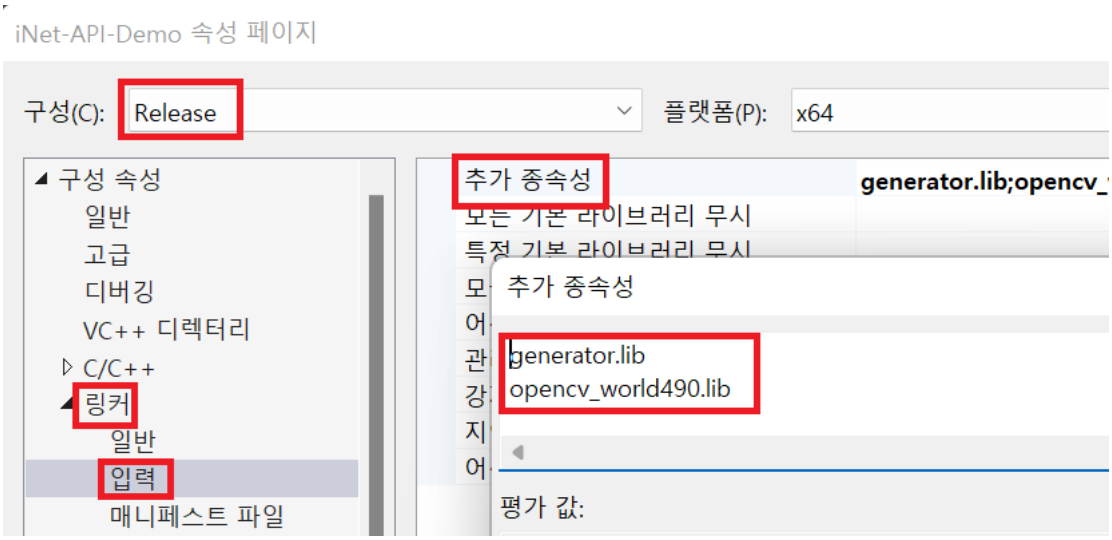
- Path: Properties → Linker → General → Additional Library Directories
- Enter “bin”

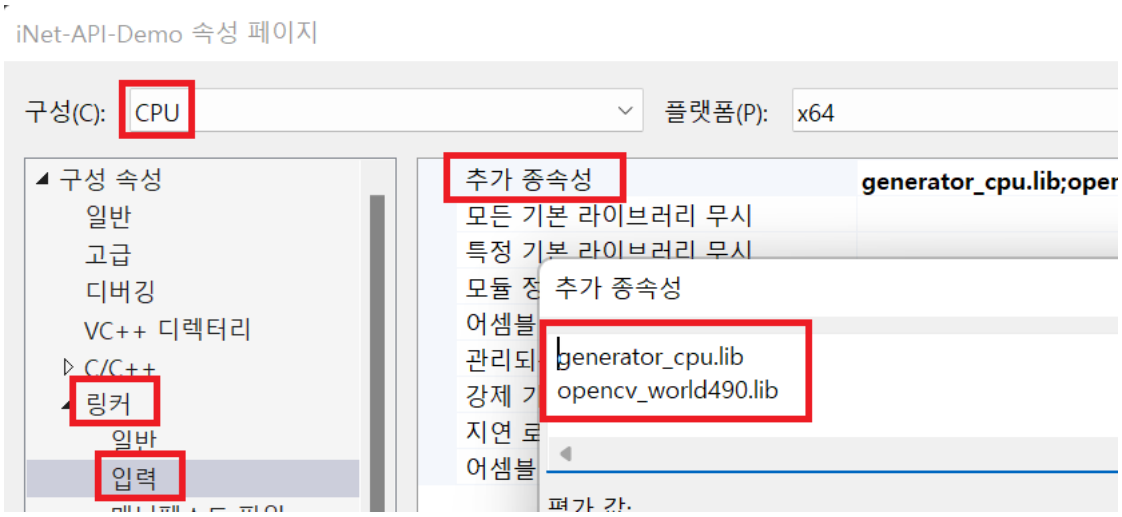


F. Modify Additional Dependencies

- Path: Properties → Linker → General → Additional Dependencies
- Edit Additional Dependencies

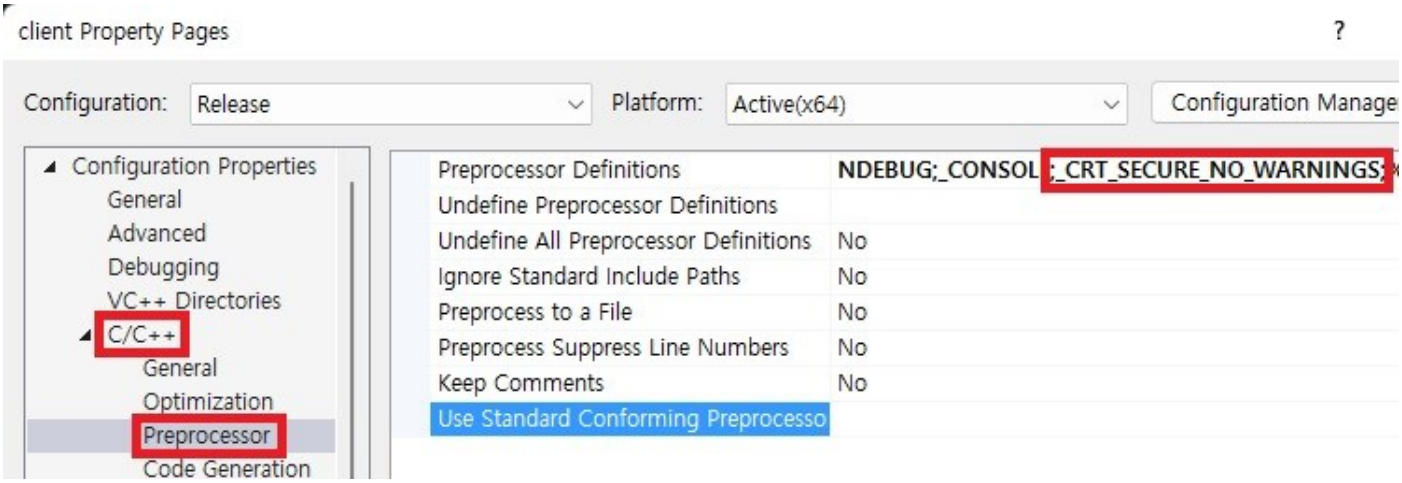
- Release configuration: “generator.lib”, “opencv_world490.lib”
- CPU configuration: “generator_cpu.lib”, “opencv_world490.lib”





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

A. Download and upzip the followings 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 configuration. Then, run the solution.

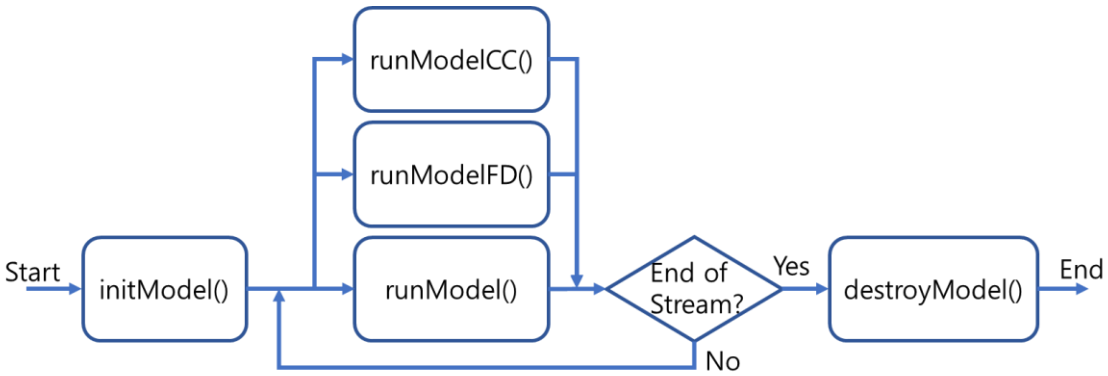
Part II. iNet Solution Developer Guide

1. Solution Introduction

■ AIPro iNet Solution proceeds in four major steps, Initialization, Execution-1, Execution-2, Execution-3, and Destruction. The functions and details corresponding to each step are as follows:

Step	API Function	Content
Initialization	initModel()	- Initialize models and internal memory required to run the solution
Execution-1	runModel()	- Receive a frame and perform inference - Responsible for object detection, tracking, counting (Line & Zone), and PAR (Pedestrian attribute recognition) - Fill out detected object boxes, tracking IDs, counting results, and PAR info to the DetBox object
Execution-2	runModelFD()	- Receive a frame and perform inference - Responsible for detecting Fire and Smoke objects - Fill out detected FD objects to the FireBox object
Execution-3	runModelCC()	- Receive a frame and perform inference - Responsible for crowd counting - Fill out detected density map and the number of people
Destruction	destroyModel()	- 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(), runModelFD, and runModelCC() for each frame



<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()
  - Depending on "parsing mode" of config.json, initialization is performed using config.json in .₩inputs or cam.json and init.txt in the C:₩aipro₩data₩config path
- 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

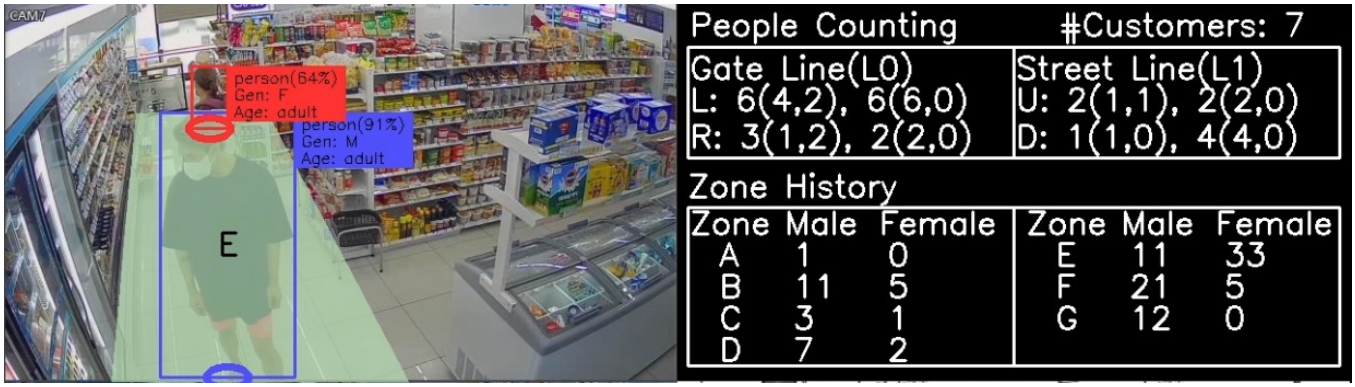
3. Counting Information

- 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, CCRecord &ccRcd)
Initialize model - param cfg configuration struct - param odRcd object detection record struct - param fdRcd fire detection record struct - param ccRcd crowd counting record struct - return initialization result(true: success, false: fail)

bool runModel(vector<DetBox> &dboxes, Mat &frame, int vchID, uint frameCnt, float odScoreTh, float actScoreTh)
Run OD and PAR models for a frame batch - param dboxes return detected dboxes of the vchIDl video channel - param frame input frame - param vchID vchID of the input frame - param frameCnt frameCnt of the input frame - 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(std::vector<FireBox> &fboxes, cv::Mat &frame, int vchID, uint &frameCnt, float fdScoreTh)
Run FD models for a frame batch - param fboxes return detected fboxes of the vchID channel - param frame input frame - param vchID vchID of the input frame - param frameCnt frameCnt of the input frame - param fdScoreTh threshold for filtering low confident detections - return flag for the running result(true: success, false: fail)

bool runModelCC(cv::Mat &density, cv::Mat &frame, int vchID, float ccScoreTh)
Run FD models for a frame batch - param density return the density of people - param frame input frame - param vchID vchID of the input frame

- param ccScoreTh 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

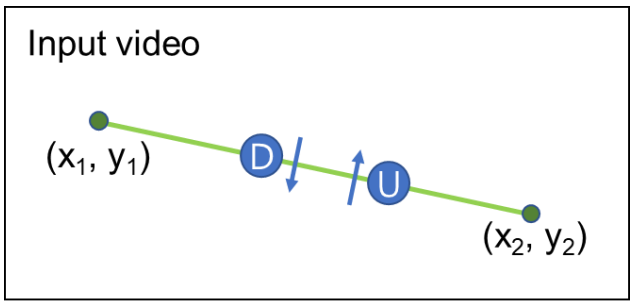
- param fdRcd record struct
- return flag for reset(true: success, false: fail)

bool resetFDRecord()

Reset fd record

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

5. Configuration of config.json

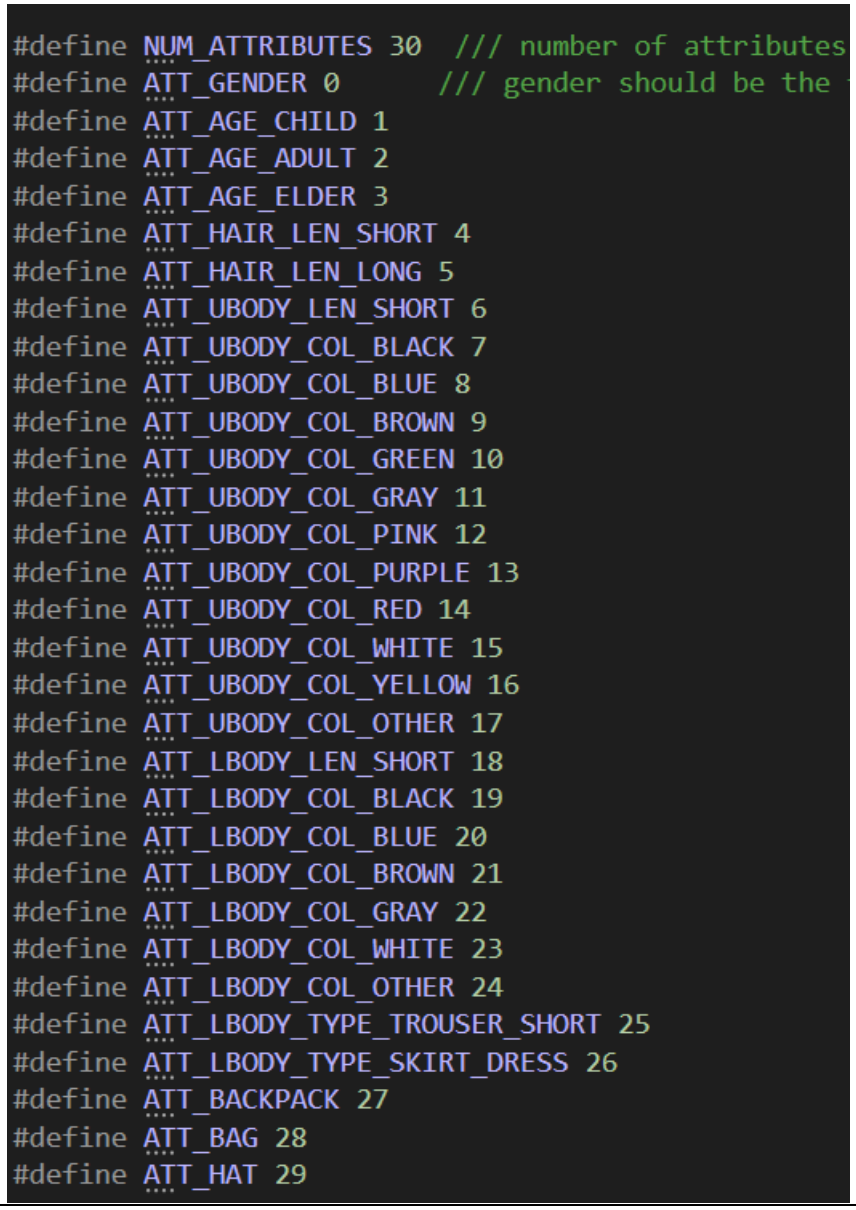
Name	Item	Value
global	apikey	Solution key (must use “aiprotest”)
	frame_limit	Number of frames to be processed
	recording	Recording On/Off
	debug_mode	Including debug Information in output frames
	parsing_mode	Usage of cam.json and init.txt
	input_files	Input video files with path
	output_files	Output 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	Enter Counting Line information as follows: [line_id vchID x1 y1 x2 y2] - line_id: unique ID - vchID: Video Channel ID - x1, y1: Point 1 - x2, y2: Point 2  (Counting example) UP: Male 3(1/2/0), Female 2(0/1/1) Down: Male 5(2/2/1), Female 1(0/1/0)
zone	param	Enter Zone information as follows: [zone_id vchID isRestricted x1 y1 x2 y2 x3 y3 x4 y4] - zone_id: unique ID

		- vchID: Video Channel ID - isRestricted: flag for specifying a restricted area (internally not used) x1 y1 x2 y2 x3 y3 x4 y4: four vertex coordinates(entered in consecutive directions)
cc_zone	param	Add [ccLevelTh1, ccLevelTh2, ccLevelTh3] to zone's param - LevelTh1: Leve1 threshold (Level1 if exceeds) - LevelTh2: Leve1 threshold (Level2 if exceeds) - LevelTh3: Leve1 threshold (Level3 if exceeds)

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	Male/Female recognition (accuracy: about 93%) - DetBox-PedAtts-atts[0]: 0:Male, 1:Female
Age	Child/Adult/Elder recognition (accuracy: about 85%) - 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	
Movement	Motion activity (for detecting fainting, falling, sleep, etc) - DetBox-distVar: box center variation after temporal pooling
Time to appear	For calculating the time spent after appear - DetBox-inTime: time when this object is detected
Previous position	For estimating movement path - DetBox-(rxP, ryP): reference position in the previous frame

7. 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

8. Average Inference Time

- Measure time delays inside runModel()
  - Part 0(OD + Tracking + PAR): 14~16ms
  - Part 1(FD): 20~25ms
  - Part 2(CC): 70~80ms
  - Input Video Resolution: FHD, Model Input Resolution: 960x544, GPU: 2080Ti, CPU: i9-10900X@3.70GHz, Batch Size: 1 frame