AIPro Inc. Technical Documentation (Confidential)

AIPro sNet Solution Demo Guide

(API Version)

Part I. Demo Program Installation

Part II. sNet Solution Guide

System Requirement & Dependency

Category	Content
Model	Super Resolution
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

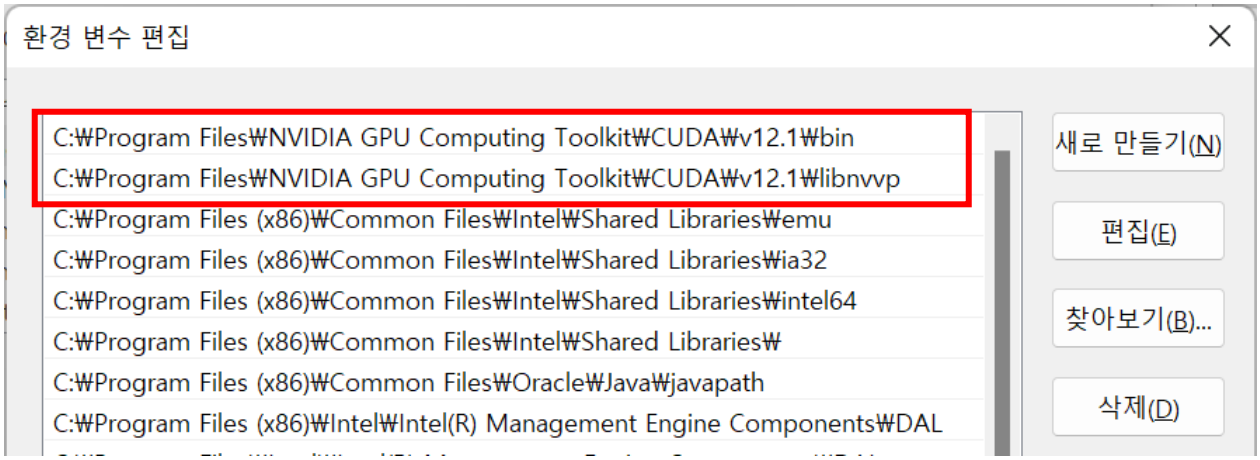
2. CUDA Installation

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

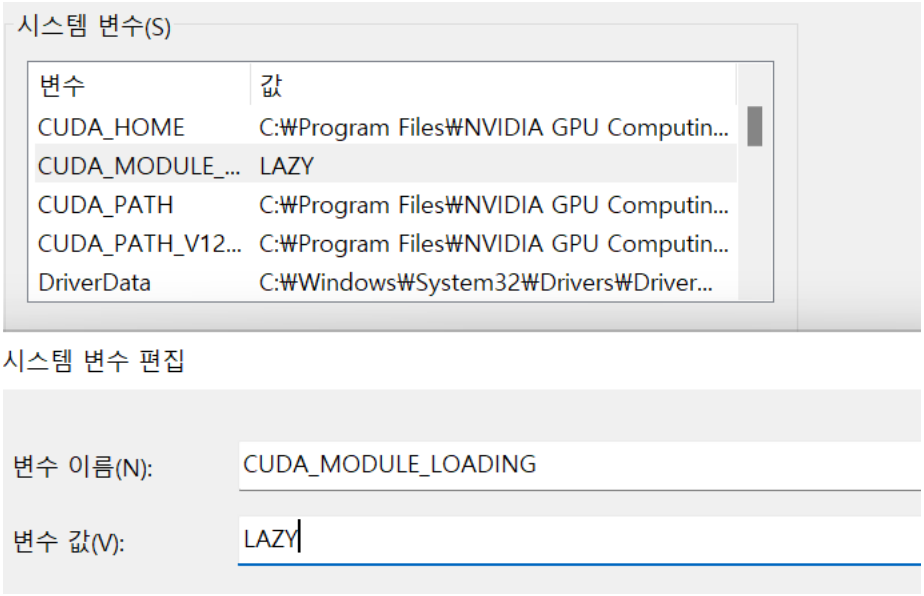
B. Add the followings to PATH

- 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

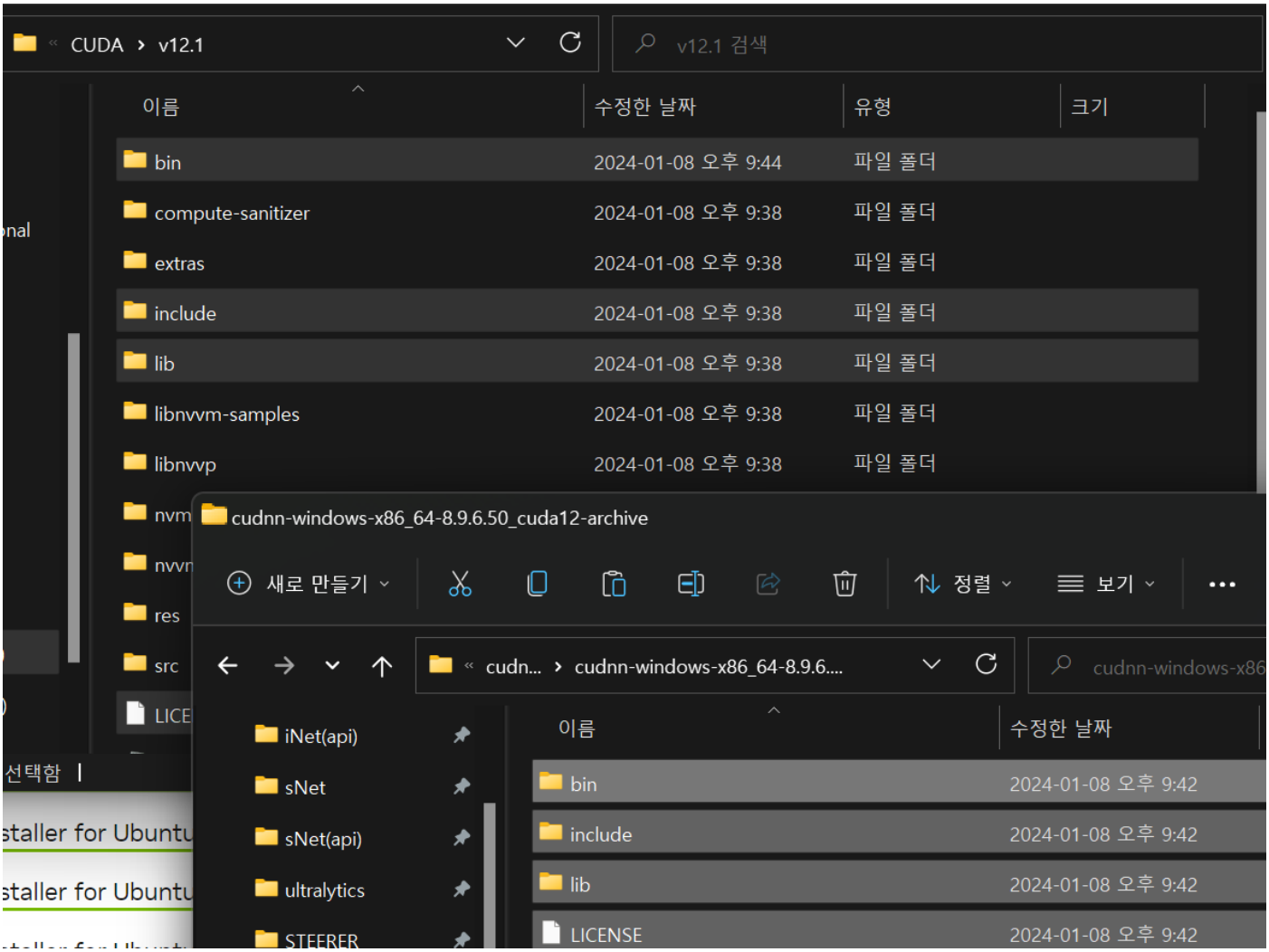
- Name: CUDA_MODULE_LOADING, Value: LAZY
- Reboot required after adding
- Link: https://docs.nvidia.com/cuda/cuda-c-programming-guide/#lazy-loading



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 sNet-API-Demo repository

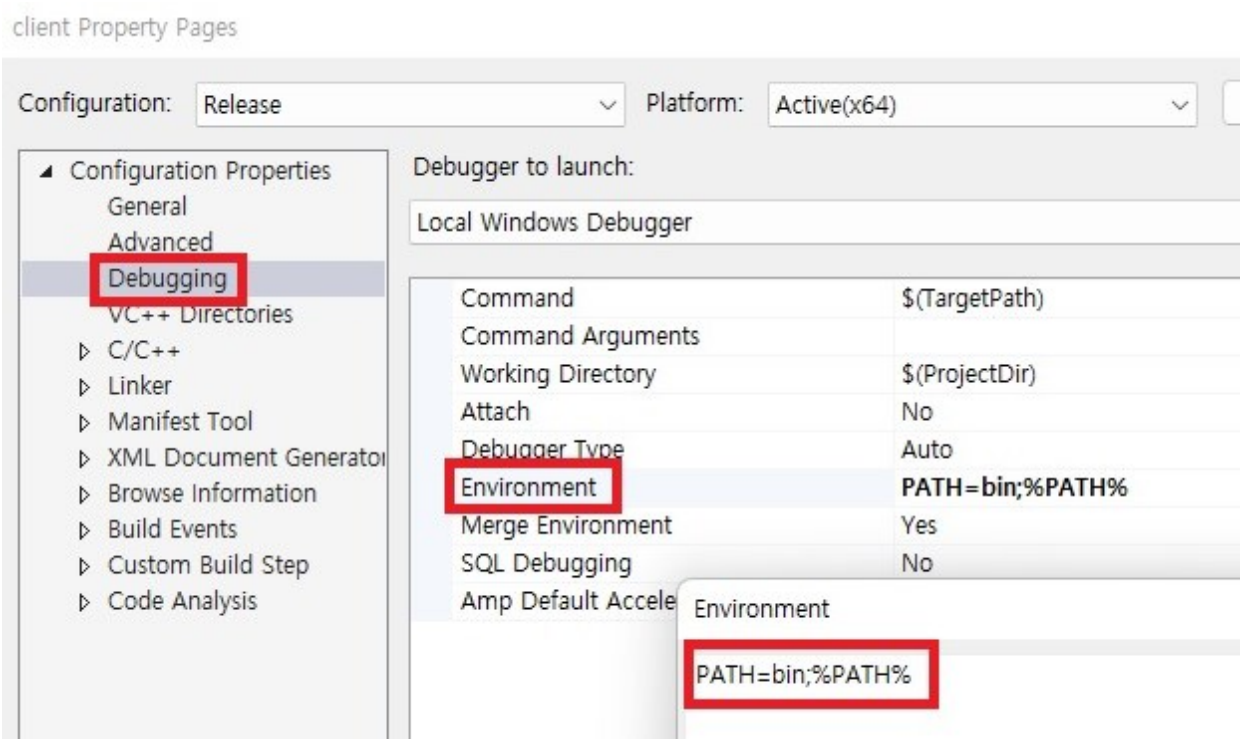
- Repository Address: https://github.com/AIProCo/sNet-API-Demo

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

- Set Configurations to “Release” and Platforms to “x64”
- Debug configuration is not supported

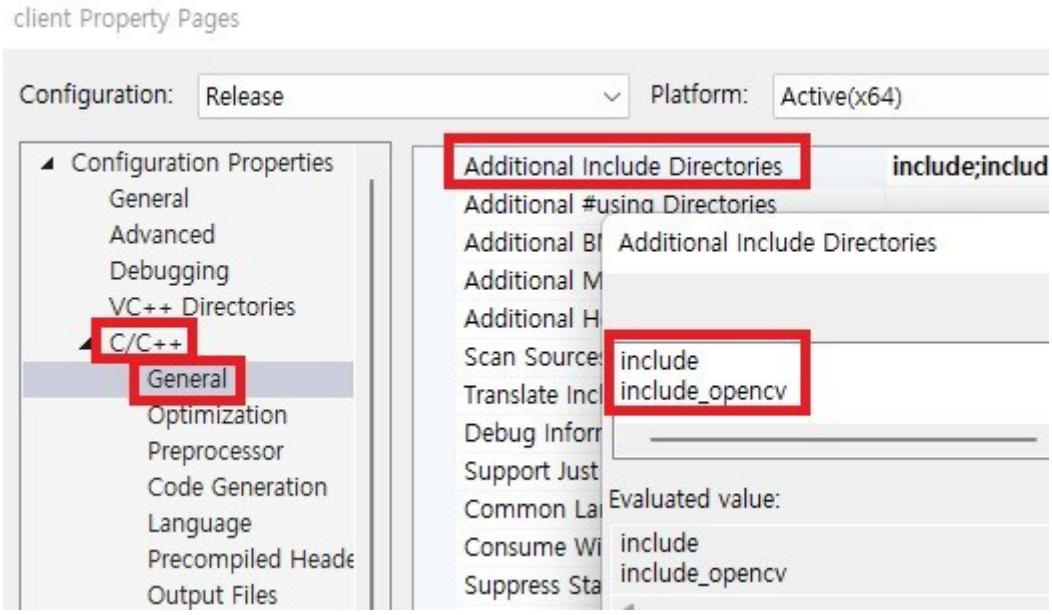
C. Modify the local PATH variable (the system PATH variable is not affected)

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



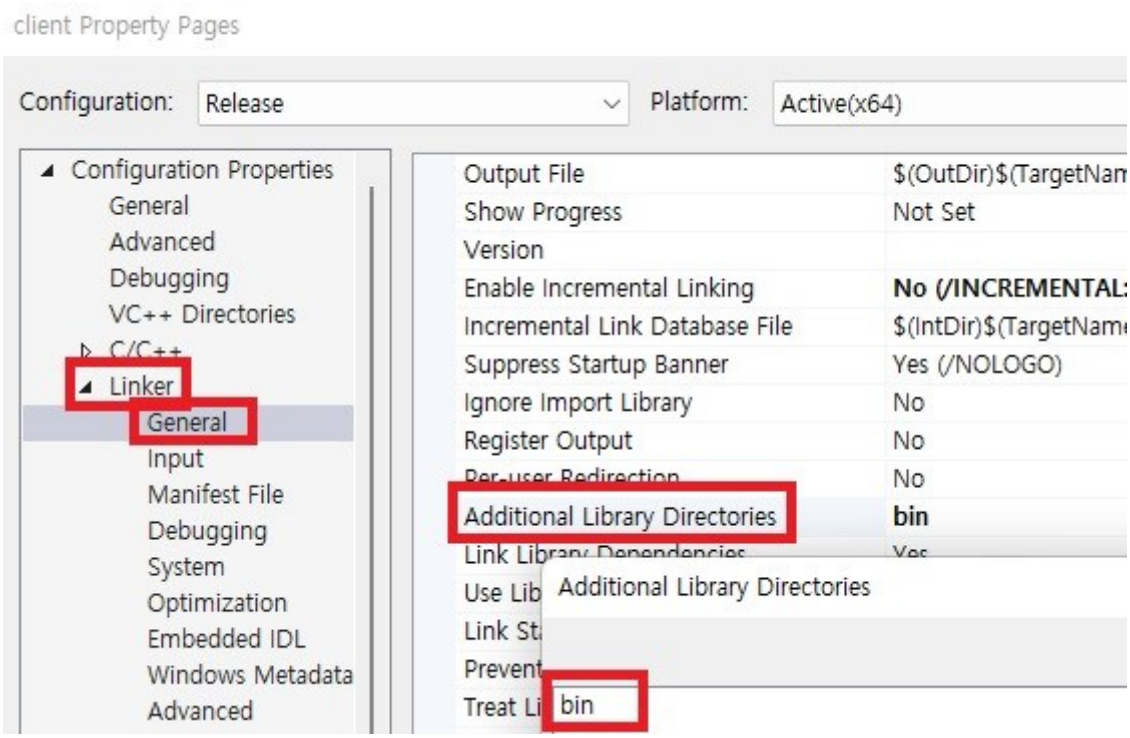
D. Modify Additional include Directories

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



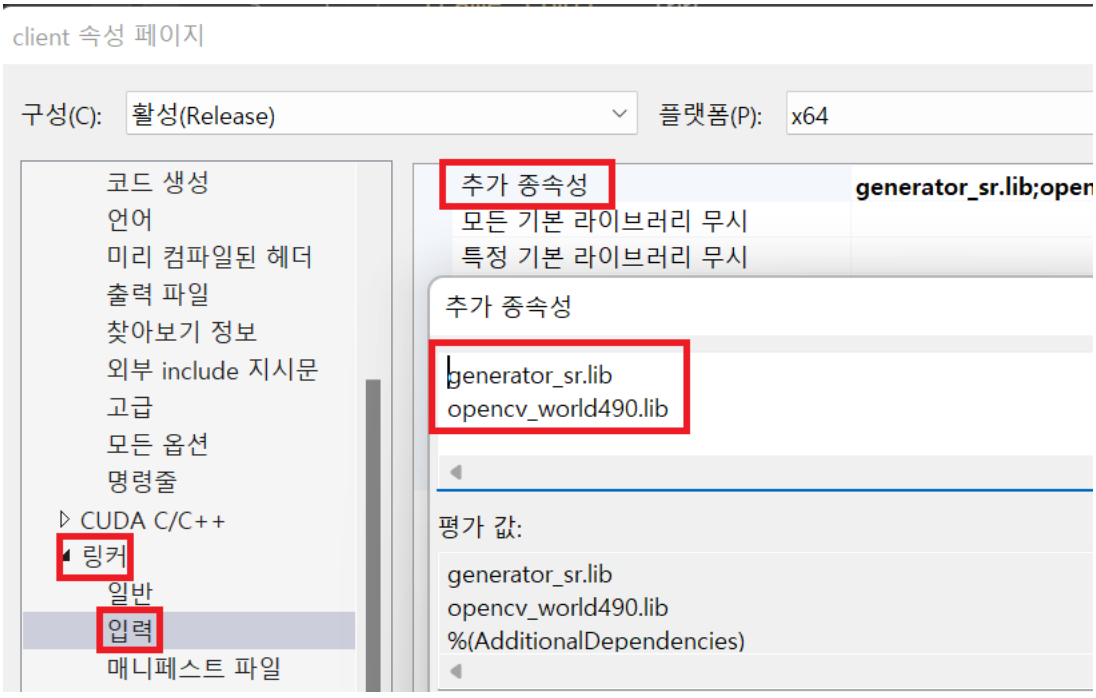
E. Modify Additional Library Directories

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



F. Modify Additional Dependencies

- Path: Properties → Linker → General → Additional Dependencies
- Enter “generator_sr.lib” and “opencv_world490.lib”

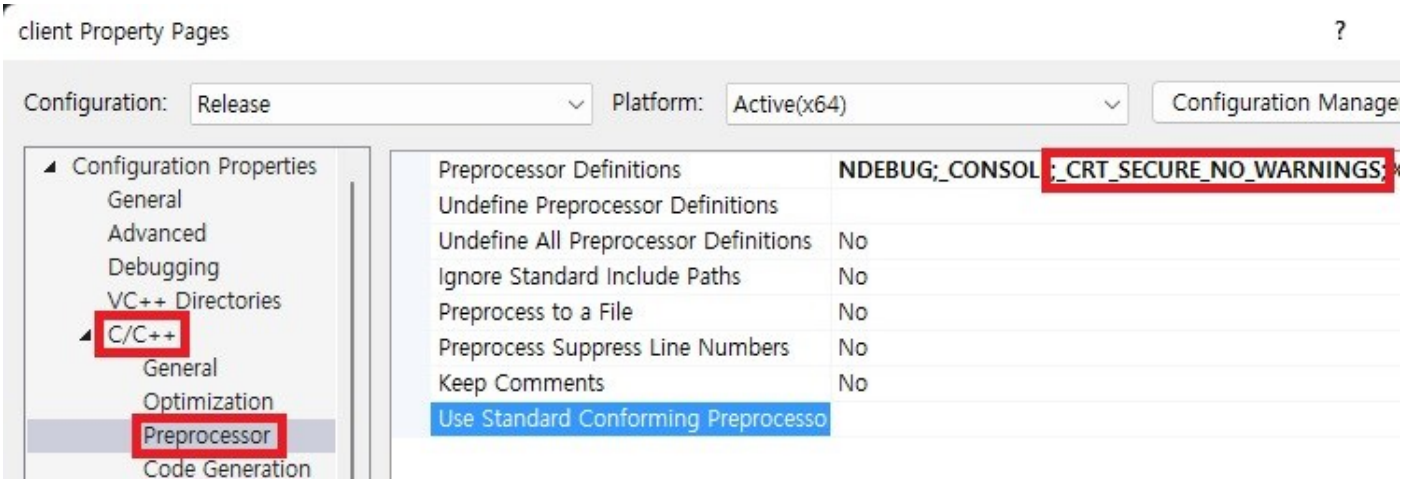


G. Disable security error

- 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 links of the github repository.

6. Set Release mode and x64 platform. Then, run the solution.

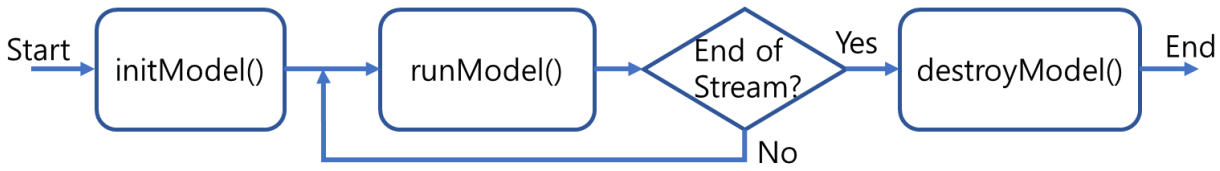
Part II. sNet Solution Developer Guide

1. Solution Introduction

■ AIPro sNet Solution proceeds in three major steps, Initialization, Execution, 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	runModel()	- Receive a frame and perform SR inference - Fill out an upsampled frame to srFrame
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() for each batch of frames



<Figure> Flowchart of sNet Solution

2. Program Development Using sNet

■ Basically, the sNet 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. API Functions

bool initModel(std::string _srModelFileX2, std::string _srModelFileX1_5)
Initialize model - param _srModelFileX2 x2 SR model file path - param _srModelFileX1_5 x1.5 SR model file path - return initialization result(true: success, false: fail)

bool runModel(vector<Mat> &frames, vector<Mat> &srFrames, vector<int> &vchIDs)
Run the SR model for a frame batch - param frames batch of frames - param srFrames batch of output SR frames - param vchIDs vchIDs of batched frames - return run result(true: success, false: fail)

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

4. Configuration of config.json

Name	Item	Value
global	apikey	Solution key (must use “aiprotest”)
	frame_limit	Number of frames to be processed
	input_files	Input video files with path
	output_files	Output SR video files with path
	filter_enable	Enable to output upsampled videos using a conventional filter - Upsample an input frame using the bilinear filter and save the result frame - Create a file by adding "_filter" to the input file name
	filter_file_append	Sting to be appended to the filter filenames
sr	scale_factor	Scale factors for SR conversion - Support 1.5 and 2.0 - Example: [2, 1.5]

- The resolution of the SR output image is automatically calculated according to input resolution and scale factor.
  - Input resolution: 1280x720, scale_factor: 1.5 → Output resolution: 1920x1080
  - Input resolution: 1920x1080, scale_factor: 2 → Output resolution: 3840x2160
- This demo supports all input resolutions smaller than 1920x1080

5. Average Inference Time

- Measure time delays of the runModel() function
  - GPU: RTX-3090, CPU: i9-10900X@3.70GHz
  - qHD2FHD: 28ms/frame
  - HD2FHD: 45ms/frame