Technical Report: Detectron2 Installation on Windows

Date: February 10, 2026 Environment: Windows 11 (x64) Target Library: Detectron2 (Facebook Research) Hardware: NVIDIA GPU (CUDA-enabled)

1. Executive Summary & Final Configuration

The installation initially failed due to incompatibilities between the bleeding-edge Visual Studio 2026 (Preview) compiler and the stable CUDA 12.1 toolkit. By downgrading the compiler environment to Visual Studio 2022, installing the correct CUDA Toolkit, and patching specific system header files to bypass version checks, a fully GPU-accelerated build was achieved.

Final Successful Software Stack:

- **OS:** Windows 11 x64
- **Python:** 3.10 (via Conda environment det2)
- **Compiler:** Microsoft Visual Studio 2022 Community (v17.14) – cl.exe v19.44
- **CUDA Toolkit:** Version 12.1 – nvcc.exe v12.1
- **Deep Learning Framework:** PyTorch 2.1.x + cu121
- **Build System:** Ninja 1.11+

2. Troubleshooting Log: Issues & Resolutions

This section documents the specific obstacles encountered in your system and how they were diagnosed and resolved.

Issue A: Compiler Not Found ([WinError 2])

Symptom: The build failed immediately with cl.exe: No such file or directory.

Diagnosis: The command was run in a standard PowerShell terminal. Standard terminals do not have access to C++ build tools.

Resolution: Switched to the "x64 Native Tools Command Prompt", which pre-loads the compiler PATH.

Issue B: Missing CUDA Compiler (nvcc not found)

Symptom: Running nvcc --version returned "not recognized".

Diagnosis: While NVIDIA Drivers were present (allowing games/display to work), the CUDA Toolkit (required for compiling code) was missing.

Resolution: Installed CUDA Toolkit 12.1 explicitly from the NVIDIA Developer archive.

Issue C: Compiler Incompatibility (Visual Studio 2026)

Symptom: Build failed with fatal error C1189: #error: -- unsupported Microsoft Visual Studio version!.

Diagnosis: Your system originally had Visual Studio 2026 Preview. PyTorch and Detectron2 are not yet compatible with this unreleased compiler.

Resolution: Installed the stable Visual Studio 2022 Community Edition and switched to its specific command prompt.

Issue D: The "Time Bomb" (Error STL1002)

Symptom: Even with VS 2022, the build failed with: error STL1002: Unexpected compiler version, expected CUDA 12.4 or newer.

Diagnosis: The latest update to VS 2022 (v17.14) introduced a strict check that rejects CUDA 12.1 because it is "too old."

Resolution:

- **Code Patch:** Manually edited the system file yvals_core.h to comment out the error trigger.
- **Flag Override:** Set the environment variable NVCC_APPEND_FLAGS=-allow-unsupported-compiler to force the build to proceed.

3. Standard Operating Procedure (Step-by-Step Guide)

Use this checklist to replicate the installation on another Windows machine.

Phase 1: Prerequisites (Install First)

Microsoft Visual Studio 2022 Community:

- **Crucial:** During installation, select the "Desktop development with C++" workload.
- Ensure "Windows 11 SDK" is checked.

NVIDIA CUDA Toolkit 12.1:

- Download version 12.1 specifically (to match stable PyTorch).
- Verify install by opening a terminal and running: nvcc --version

Anaconda (or Miniconda): For Python environment management.

Phase 2: Configuration & Patching

Before starting the installation, you must "defuse" the compiler version check.

- **Navigate to:** C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\<version>\include
- Locate yvals_core.h.
- Right-click -> Properties -> Security -> Give your user Full Control.
- Open in Notepad and find STL1002.
- **Comment out the error line:**

```
C++
// _EMIT_STL_ERROR(STL1002, "Unexpected compiler version...");
```

- Save and close.

Phase 3: The Installation Ritual

Note: This MUST be done in the "x64 Native Tools Command Prompt for VS 2022" (Run as Administrator).

Set up the Environment:

```
DOS
call D:\Conda\Scripts\activate.bat det2
set DISTUTILS_USE_SDK=1
set MSSdk=1
set NVCC_APPEND_FLAGS=-allow-unsupported-compiler
```

Install Base Libraries:

```
DOS
pip install torch torchvision torchaudio --index-url https://
download.pytorch.org/whl/cu121
pip install ninja opencv-python
pip install "git+https://github.com/philferriere/
cocoapi.git#subdirectory=PythonAPI" --no-build-isolation
```

Install Detectron2:

```
DOS
pip install "git+https://github.com/facebookresearch/
detectron2.git" --no-build-isolation --no-cache-dir
```

Phase 4: Verification

Create a python script (test.py) with the following to confirm GPU acceleration:

```
Python
import torch, detectron2
from detectron2.utils.logger import setup_logger
setup_logger()

print(f"Detectron2 Version: {detectron2.__version__}")
print(f"CUDA Available: {torch.cuda.is_available()}")
print(f"Device Name: {torch.cuda.get_device_name(0)}")
```

4. Conclusion

The installation is now fully operational. The inference test performed on input.jpg successfully utilized the GPU (Target Device: cuda) and generated segmentation masks (output.jpg), confirming that the C++/CUDA custom layers were compiled correctly.

Recommendation: Do not update Visual Studio or CUDA on this machine unless necessary, as it may revert the yvals_core.h patch or break compatibility again.