PlantVillage SSL - Installation Guide

Warre Snaet | Howest MCT | Research Project 2025-2026

---

1. System Requirements

Target Hardware

Device	Specification
Primary Target	Edge devices (embedded GPUs, 8GB+)
Alternative	Linux PC with NVIDIA GPU (4GB+ VRAM)

Operating System

- **Ubuntu 20.04 LTS** or **Ubuntu 22.04 LTS**

Minimum Requirements

Resource	Minimum	Recommended
RAM	8 GB	16 GB
Storage	10 GB	50 GB
GPU VRAM	4 GB	8 GB
Internet	Required for initial setup	Not needed at runtime

---

2. Install System Dependencies

PROFI

Open a terminal and install the required build tools and libraries:

```
# Update package list
sudo apt update && sudo apt upgrade -y

# Install build essentials
sudo apt install -y \
    build-essential \
    cmake \
    libssl-dev \
    pkg-config \
    libclang-dev \
    curl \
    git \
    wget
```

3. Install Rust

The core machine learning engine is written in Rust.

```
# Install Rust via rustup
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh -s -- -y

# Configure the current shell
source "$HOME/.cargo/env"

# Verify installation
rustc --version
# Expected: rustc 1.7x.x or higher

cargo --version
# Expected: cargo 1.7x.x
```

4. Install CUDA (GPU Support)

For GPU acceleration on NVIDIA hardware.

Desktop Linux

1. **Install NVIDIA Driver** (if not installed):

   ```
   sudo apt install nvidia-driver-535  # Or latest available
   ```

2. **Install CUDA Toolkit**:

   ```
   sudo apt install nvidia-cuda-toolkit
   ```

3. **Configure Environment Variables**:

   ```
   # Add to ~/.bashrc
   echo 'export PATH=/usr/local/cuda/bin:$PATH' >> ~/.bashrc
   echo 'export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH' >>
   ~/.bashrc
   source ~/.bashrc
   ```

4. **Verify Installation**:

PROFI

```
nvcc --version
# Expected: Cuda compilation tools, release 11.x or 12.x

nvidia-smi
# Should show GPU information
```

Platform Notes

For some embedded platforms the vendor provides a pre-installed SDK that bundles CUDA and related components. Verify your platform's SDK documentation for exact CUDA versions.

---

5. Install GUI Dependencies (Tauri)

The graphical interface requires GTK and WebKit libraries.

```
# Install Tauri dependencies
sudo apt install -y \
    libwebkit2gtk-4.0-dev \
    libwebkit2gtk-4.1-dev \
    libgtk-3-dev \
    libayatana-appindicator3-dev \
    librsvg2-dev \
    file
```

Install Bun (JavaScript Runtime)

Bun is used for the frontend build system (faster than Node.js).

PROFI

```
# Install Bun
curl -fsSL https://bun.sh/install | bash

# Reload shell configuration
source ~/.bashrc

# Verify installation
bun --version
# Expected: 1.x.x
```

---

6. Clone the Repository

```
# Navigate to your preferred directory
cd ~/Documents
```

```
# Clone the repository
git clone https://github.com/YourUsername/Research_Project_Rust_Semi-
Supervised_Learning.git

# Or if you have the source ZIP:
unzip WS_SourceCode.zip -d Research_Project
cd Research_Project
```

---

7. Download the Dataset

The application uses the PlantVillage / New Plant Diseases Dataset from Kaggle.

Option A: Using the Provided Script

```
cd plantvillage_ssl
./scripts/download_dataset.sh
```

Option B: Manual Download via Kaggle CLI

```
# Install Kaggle CLI (if not installed)
pip install kaggle

# Configure Kaggle API credentials
# Download kaggle.json from https://www.kaggle.com/account
mkdir -p ~/.kaggle
mv ~/Downloads/kaggle.json ~/.kaggle/
chmod 600 ~/.kaggle/kaggle.json

# Download the dataset
cd plantvillage_ssl
kaggle datasets download -d vipoooool/new-plant-diseases-dataset
unzip new-plant-diseases-dataset.zip -d data/plantvillage/
```

PROFI

Expected Structure After Download

```
plantvillage_ssl/data/plantvillage/
├── train/
│   ├── Apple___Apple_scab/           (~2000 images)
│   ├── Apple___Black_rot/
│   ├── Tomato___Early_blight/
│   └── ... (38 class folders)
└── valid/
```

```
    ├── Apple___Apple_scab/          (~500 images)
    └── ... (38 class folders)
```

---

8. Build the Project

Build the Rust Backend

```
cd plantvillage_ssl

# Build with CUDA support (recommended)
cargo build --release --features cuda

# Alternative: CPU-only build
cargo build --release --features cpu
```

Note: The first build takes 5-15 minutes to compile all dependencies.

Build the GUI Frontend

```
cd plantvillage_ssl/gui

# Install JavaScript dependencies
bun install

# Verify build (optional)
bun run check
```

PROFI

9. Run the Application

Option A: Launch GUI in Development Mode

```
cd plantvillage_ssl/gui
bun run tauri:dev
```

Option B: Build Production Application

```
cd plantvillage_ssl/gui
bun run tauri:build

# The built application is at:
# src-tauri/target/release/plantvillage-ssl
```

Option C: CLI Only (No GUI)

```
cd plantvillage_ssl

# View available commands
./target/release/plantvillage_ssl --help

# Train a model
./target/release/plantvillage_ssl train \
    --epochs 30 \
    --cuda \
    --labeled-ratio 0.2

# Run inference
./target/release/plantvillage_ssl infer \
    --model-path best_model.mpk \
    --image-path /path/to/leaf.jpg
```

---

10. Configuration

Environment Variables

Create a `.env` file or set these in your shell:

```
# Enable CUDA (optional, enabled by default)
export CUDA_VISIBLE_DEVICES=0

# Set data directory (optional)
export PLANTVILLAGE_DATA_DIR=/path/to/plantvillage

# Enable debug logging (optional)
export RUST_LOG=info
```

PROFI

Pipeline Configuration

The main configuration is in `pipeline_config.yaml`:

```
# Key settings
labeled_ratio: 0.2             # 20% labeled, 80% unlabeled
confidence_threshold: 0.9      # Pseudo-label threshold
retrain_threshold: 200         # Images before retraining
epochs: 30                     # Training epochs
batch_size: 32                 # Batch size (reduce if OOM)
```

11. Verification

Run these commands to verify the installation:

```
# 1. Check Rust backend compiles
cd plantvillage_ssl
cargo check --release

# 2. Run tests
cargo test

# 3. Check dataset
./target/release/plantvillage_ssl stats --data-dir data/plantvillage

# 4. Run inference benchmark
./target/release/plantvillage_ssl benchmark --model-path best_model.mpk

# 5. Start GUI (final test)
cd gui && bun run tauri:dev
```

12. Troubleshooting

Common Issues & Solutions

Error	Cause	Solution
"CUDA not found"	CUDA path not set	Add to PATH: `export PATH=/usr/local/cuda/bin:$PATH`
"Linker error: ld"	Missing build tools	Run: `sudo apt install build-essential pkg-config`
"webkit2gtk not found"	Missing Tauri deps	Run step 5 again
"Out of memory"	Insufficient VRAM	Reduce batch size or use `--features cpu`
Compile timeout	Slow machine	Use `cargo build --release -j 2` to limit parallelism

PROFI

Platform-Specific Issues

```
# If "nvcc not found":
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
```

```
# If WebKit2GTK issues:
sudo apt install libwebkit2gtk-4.0-dev --fix-missing
```

Clean Rebuild

If you encounter strange build errors:

```
cd plantvillage_ssl
cargo clean
cargo build --release --features cuda
```

---

13. Directory Structure

After successful installation, your directory should look like:

```
Research_Project/
├── Source/
│   ├── plantvillage_ssl/          # Main SSL application
│   │   ├── src/                   # Rust source code
│   │   ├── gui/                   # Tauri GUI application
│   │   ├── data/plantvillage/     # Dataset
│   │   ├── output/                # Training outputs
│   │   ├── scripts/               # Utility scripts
│   │   ├── best_model.mpk         # Pre-trained model
│   │   └── target/release/        # Compiled binaries
│   ├── incremental_learning/      # Incremental learning workspace
│   ├── pytorch_reference/         # Python baseline
│   ├── benchmarks/                # Benchmark scripts
│   └── research/                  # Documentation
└── Gastsessies/                   # Guest lecture notes
```

PROFI

---

14. Next Steps

After successful installation:

1. **Read the User Manual** (`WS_UserManual.md`) for operation instructions
2. **Train your first model** using the provided dataset
3. **Run benchmarks** to verify performance on your hardware
4. **Try the Simulation** to see semi-supervised learning in action

---

15. Quick Reference

Essential Commands

```
# Start GUI
cd plantvillage_ssl/gui && bun run tauri:dev

# Train model (CLI)
cd plantvillage_ssl && ./target/release/plantvillage_ssl train --cuda --epochs
30

# Run inference
./target/release/plantvillage_ssl infer --model best_model.mpk --image leaf.jpg

# Benchmark
./target/release/plantvillage_ssl benchmark --model best_model.mpk
```

Contact: Warre Snaet | Howest MCT

Document Version: 1.0 | January 2026