Introduction to Deep Learning
Troubleshooting Guide

MSc Computer Science

Week 1

Contents

1 Introduction

This guide covers common problems encountered during environment setup and how to fix them. Problems are organized by operating system and category.

2 General Issues

2.1 "command not found" or "is not recognized"

Problem

When trying to run `python` or `pip`, you get:

- Windows: "'python' is not recognized as an internal or external command"
- Linux/macOS: "python: command not found"

Solution

Windows:

1. Python is not in your PATH
2. Reinstall Python and check "Add Python to PATH"
3. Or manually add Python to PATH:
   - Search for "Environment Variables" in Start Menu
   - Click "Environment Variables"
   - Under "System variables", find "Path", click "Edit"
   - Click "New" and add: `C:\Users\YourUsername\AppData\Local\Programs\Python\Python311`
   - Also add: `C:\Users\YourUsername\AppData\Local\Programs\Python\Python311\Scripts`
   - Click OK, close all terminals, open a new one

Linux/macOS:

1. Try `python3` instead of `python`
2. If still not found, reinstall Python (see setup guide)
3. Check installation:

```
which python3
```

2.2 Virtual Environment Not Activating

Problem

After running the activation command, you don't see `(deep_learning_env)` in your prompt.

✔ Solution

Windows (PowerShell):

1. PowerShell may block scripts by default
2. Run PowerShell as Administrator
3. Execute:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope
    CurrentUser
```

4. Try activating again:

```
deep_learning_env\Scripts\Activate.ps1
```

Windows (Command Prompt):

1. Make sure you're using `.bat` not `.ps1`:

```
deep_learning_env\Scripts\activate.bat
```

Linux/macOS:

1. Make sure you use `source`:

```
source deep_learning_env/bin/activate
```

2. Check if venv was created successfully:

```
ls deep_learning_env/bin/
```

3. You should see `activate` file

2.3 Permission Denied Errors

Problem

Getting "Permission denied" when trying to create virtual environment or install packages.

✔ Solution

Linux/macOS:

1. DON'T use `sudo` with pip or venv
2. Make sure you own the directory:

```
cd ~/Documents
mkdir DeepLearning
cd DeepLearning
```

3. If you accidentally used sudo, fix permissions:

```
sudo chown -R $USER:$USER ~/Documents/DeepLearning
```

Windows:

1. Don't create the folder in `C:\Program Files`
2. Use your user directory: `C:\Users\YourUsername\Documents`

3 Package Installation Issues

3.1 "No module named 'pip"

Problem

Error when trying to use pip: `No module named 'pip'`

✔ Solution

Windows:

```
python -m ensurepip --upgrade
```

Linux:

```
sudo apt install python3-pip
```

macOS:

```
python3 -m ensurepip --upgrade
```

3.2 PyTorch Installation Fails or Takes Forever

Problem

`pip install torch` is extremely slow or fails with timeout errors.

✔ Solution

1. Use PyTorch's own package index (faster):

```
pip install torch torchvision torchaudio --index-url https://
    download.pytorch.org/whl/cpu
```

2. If still slow, try increasing timeout:

```
pip install --default-timeout=1000 torch torchvision
    torchaudio
```

3. Or download wheel file manually:

   - Go to `https://download.pytorch.org/whl/torch_stable.html`
   - Download appropriate `.whl` file for your system
   - Install with:

```
pip install path/to/downloaded/file.whl
```

3.3 SSL Certificate Errors

Problem

`SSL: CERTIFICATE_VERIFY_FAILED` when installing packages.

✔ Solution

Quick fix (not recommended for production):

```
pip install --trusted-host pypi.org --trusted-host pypi.python.org
    --trusted-host files.pythonhosted.org -r requirements.txt
```

Proper fix:

1. Update certificates:

   - Windows: Run `Install Certificates.command` in Python installation directory
   - macOS:

```
/Applications/Python\ 3.11/Install\ Certificates.command
```

   - Linux:

```
sudo apt install ca-certificates
```

3.4 Version Conflicts

Problem

"Could not find a version that satisfies the requirement" or dependency conflicts.

✔ Solution

1. Make sure virtual environment is activated
2. Update pip:

```
pip install --upgrade pip
```

3. Try installing packages one by one instead of from requirements.txt:

```
pip install torch
pip install jupyter
pip install matplotlib
pip install numpy
```

4. If still failing, create a new virtual environment:

```
deactivate
rm -rf deep_learning_env  # or delete folder on Windows
python3 -m venv deep_learning_env
source deep_learning_env/bin/activate
pip install --upgrade pip
pip install -r requirements.txt
```

4 Jupyter Notebook Issues

4.1 Jupyter Notebook Won't Start

🐞 Problem

`jupyter notebook` command fails or browser doesn't open.

✔ Solution

1. Make sure Jupyter is installed:

```
pip install jupyter notebook
```

2. Try specifying a port:

```
jupyter notebook --port=8889
```

3. Check if port 8888 is already in use:
   - Windows:

```
netstat -ano | findstr :8888
```

   - Linux/macOS:

```
lsof -i :8888
```

4. If browser doesn't open, manually copy the URL from terminal (starts with `http://localhost:8888/...`)

4.2 Kernel Won't Connect or Keeps Dying

Problem

Jupyter kernel fails to start or crashes when running cells.

Solution

1. Install/reinstall ipykernel:

```
pip install --upgrade ipykernel
python -m ipykernel install --user --name=deep_learning_env
```

2. Select correct kernel in Jupyter:
   - In notebook: Kernel → Change kernel → deep_learning_env
3. Check for memory issues (especially on older computers):
   - Close other applications
   - Try smaller batch sizes in code
4. Restart Jupyter completely:

```
# In terminal where Jupyter is running: Ctrl+C, Ctrl+C
jupyter notebook
```

4.3 Import Errors in Jupyter

Problem

`ModuleNotFoundError: No module named 'torch'` even though you installed it.

Solution

Jupyter is using wrong Python environment!

1. Check which Python Jupyter is using:

```
import sys
print(sys.executable)
```

   Should point to your `deep_learning_env`

2. Make sure virtual environment was activated before starting Jupyter

3. Install Jupyter inside the virtual environment:

```
# Activate environment first!
source deep_learning_env/bin/activate  # Linux/macOS
# or: deep_learning_env\Scripts\activate  # Windows

pip install jupyter
jupyter notebook
```

4. Register the environment as a kernel:

```
python -m ipykernel install --user --name=deep_learning_env --
    display-name="Python (Deep Learning)"
```

   Then in Jupyter: Kernel → Change kernel → Python (Deep Learning)

5 Platform-Specific Issues

5.1 Windows Specific

5.1.1 Long Path Issues

Problem

Errors about path names being too long.

Solution

1. Enable long paths in Windows:
   - Open Registry Editor (Win+R, type `regedit`)
   - Navigate to: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem`
   - Set `LongPathsEnabled` to 1

2. Or create project in shorter path:

```
cd C:\DL
```

5.1.2 Antivirus Blocking Installation

Problem

Installation hangs or fails with cryptic errors.

Solution

1. Temporarily disable antivirus during installation
2. Add Python and pip to antivirus exceptions
3. Use Windows Defender instead of third-party antivirus if possible

5.2 macOS Specific

5.2.1 "xcrun: error" on macOS

Problem

```
xcrun: error: invalid active developer path
```

Solution

Install Xcode Command Line Tools:

```
xcode-select --install
```

5.2.2 M1/M2/M3 Mac Issues

Problem

Architecture errors or package incompatibilities on Apple Silicon.

Solution

1. Make sure you're using ARM version of Python, not x86:

   ```
   python3 -c "import platform; print(platform.machine())"
   ```

   Should print `arm64`, not `x86_64`

2. If you installed Python via Homebrew:

   ```
   arch -arm64 brew install python@3.11
   ```

3. Some packages may need Rosetta 2:

   ```
   softwareupdate --install-rosetta
   ```

5.3 Linux Specific

5.3.1 "python3-venv" not found

Problem

`The virtual environment was not created successfully because ensurepip is not available`

Solution

Install venv module:

```
# Ubuntu/Debian
sudo apt install python3.11-venv

# Fedora
sudo dnf install python3-virtualenv

# Arch
sudo pacman -S python-virtualenv
```

5.3.2 "externally-managed-environment"

Problem

Error: "externally-managed-environment" when trying to install packages (common on Ubuntu 23.04+).

Solution

This is why we use virtual environments! Make sure you:

1. Created a virtual environment
2. Activated it before installing packages
3. See `(deep_learning_env)` in your prompt

If you really need to install globally (not recommended):

```
pip install --break-system-packages package_name
```

But use virtual environments instead!

6 Performance Issues

6.1 Installation is Extremely Slow

Problem

Package installation takes hours or appears stuck.

✔ Solution

1. Check your internet connection
2. Use PyTorch's CDN (faster):

```
pip install torch torchvision torchaudio --index-url https://
    download.pytorch.org/whl/cpu
```

3. Install packages one at a time to identify the slow one
4. Try a different mirror/index:

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple torch
```

6.2 Jupyter/Code Runs Very Slowly

Problem

Code execution is extremely slow, especially tensor operations.

✔ Solution

1. Check if PyTorch is using CPU (expected without GPU):

```
import torch
print(torch.cuda.is_available())  # False is normal without
    NVIDIA GPU
```

2. Reduce batch sizes or data size
3. For M1/M2/M3 Macs, make sure MPS is available:

```
print(torch.backends.mps.is_available())
```

4. Close other applications
5. Check Task Manager/Activity Monitor for CPU/memory usage

7 Still Stuck?

If you've tried everything and still have issues:

1. **Document your error:**
   - Copy the full error message
   - Note your OS and Python version
   - What you were trying to do
   - What you've already tried
2. **Get help:**
   - Post in course forum with your documentation

- Come to office hours
- Email instructor with details

3. **Temporary workaround:**
   - Use GitHub Codespaces (see backup guide)
   - Continue with exercises while we troubleshoot
   - Get local setup working for next week

8 Prevention Tips

To avoid issues in the future:

1. Always activate virtual environment before installing packages
2. Keep virtual environment in project folder (easier to manage)
3. Don't use `sudo` with pip
4. Update pip regularly: `pip install --upgrade pip`
5. Use `requirements.txt` to track dependencies
6. Back up your environment:

```
pip freeze > requirements_backup.txt
```