Introduction to Deep Learning
Local Development Environment Setup Guide
Linux

MSc Artificial Intelligence

Contents

1 Overview

This guide will help you set up a local Python development environment on **Linux** for the Deep Learning course. You will:

- Install Python 3.10 or later
- Create a virtual environment
- Install PyTorch and required packages
- Verify your installation with Jupyter Notebook

Important

Please complete this setup **before** the first exercise session. If you encounter problems, consult the troubleshooting guide or use GitHub Codespaces as a temporary backup.

Note

Estimated time: 20-30 minutes depending on your internet speed and distribution.
This guide covers Ubuntu/Debian. For other distributions (Fedora, Arch, etc.), adjust package manager commands accordingly.

2 Step 1: Check Existing Python Installation

Most Linux distributions come with Python pre-installed.

2.1 Open Terminal

Press `Ctrl + Alt + T` or search for "Terminal" in your applications.

2.2 Check Python Version

In terminal, run:

```
python3 --version
```

If you see `Python 3.10.x` or higher:

- Great! Skip to Step 3 (Setting Up Virtual Environment)
- But first, check if venv is installed (see Step 2.3)

If you see an older version or error:

- Continue to Step 2 to install Python

3 Step 2: Install Python

3.1 Ubuntu/Debian

3.1.1 Update Package List

```
sudo apt update
```

3.1.2 Install Python 3.11

```
sudo apt install python3.11 python3.11-venv python3-pip
```

Note

If Python 3.11 is not available in your repositories, you may need to add the deadsnakes PPA:

```
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt update
sudo apt install python3.11 python3.11-venv
```

3.1.3 Verify Installation

```
python3.11 --version
```

3.2 Fedora

```
sudo dnf install python3.11
```

3.3 Arch Linux

```
sudo pacman -S python python-pip
```

3.4 Check venv Module

The `venv` module is required for creating virtual environments:

```
python3 -m venv --help
```

If you get an error:

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

# Fedora
sudo dnf install python3-virtualenv

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

4 Step 3: Create Project Folder

4.1 Navigate to Home Directory

```
cd ~
```

or

```
cd ~/Documents
```

4.2 Create and Navigate to Project Folder

```
mkdir DeepLearning
cd DeepLearning
```

Tip

You can verify your current location with:

```
pwd
```

Should show: `/home/yourusername/DeepLearning`

5 Step 4: Create Virtual Environment

A virtual environment keeps your project dependencies isolated.

5.1 Create the Virtual Environment

In your `DeepLearning` folder:

```
python3 -m venv deep_learning_env
```

This creates a new folder called `deep_learning_env` containing:

- Python interpreter
- pip package manager
- Space for installed packages

Wait 10-30 seconds for creation to complete.

5.2 Activate the Virtual Environment

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

5.3 Verify Activation

After activation, you should see:

```
(deep_learning_env) username@hostname:~/DeepLearning$
```

The `(deep_learning_env)` prefix indicates the virtual environment is active!

Tip

To deactivate later, simply type:

```
deactivate
```

Important

Important: You must use `source` to activate. Just running `deep_learning_env/bin/activate` without `source` will not work!

6 Step 5: Install Required Packages

Important

Make sure your virtual environment is activated! You should see `(deep_learning_env)` in your prompt.

6.1 Upgrade pip

First, upgrade pip to the latest version:

```
python3 -m pip install --upgrade pip
```

6.2 Download requirements.txt

1. Download `requirements.txt` from the course repository
2. Save it to your `DeepLearning` folder
3. Verify it's there:

   ```
   ls requirements.txt
   ```

Tip

You can download directly from terminal if you have the URL:

```
wget https://url-to-requirements.txt
# or
curl -O https://url-to-requirements.txt
```

6.3 Install All Packages

```
pip install -r requirements.txt
```

Note

This will take 5-10 minutes as PyTorch is a large package (700MB). You'll see progress bars for each package being downloaded and installed.

6.4 Alternative: Manual Installation

If you don't have `requirements.txt`, install packages individually:

```
pip install torch torchvision torchaudio
pip install jupyter notebook
pip install matplotlib numpy pandas
pip install ipywidgets
```

Important

Never use sudo with pip! This can break your system Python. Always use virtual environments.

7 Step 6: Verify Installation

7.1 Test Python and PyTorch

1. Start Python interpreter:

```
python3
```

2. In the Python prompt (`>>>`), run:

```
import torch
print(f"PyTorch version: {torch.__version__}")
print(f"CUDA available: {torch.cuda.is_available()}")

# Create a test tensor
x = torch.tensor([1, 2, 3])
print(f"Test tensor: {x}")
```

3. You should see:

```
PyTorch version: 2.x.x
CUDA available: False
Test tensor: tensor([1, 2, 3])
```

4. Exit Python:

```
exit()
```

Note

`CUDA available: False` is normal if you don't have an NVIDIA GPU. PyTorch will use your CPU, which is fine for this course.

7.2 Launch Jupyter Notebook

1. Make sure your virtual environment is still activated
2. Run:

```
jupyter notebook
```

3. A browser window should open automatically showing the Jupyter interface
4. You should see your `DeepLearning` folder contents

Tip

If the browser doesn't open automatically:

- Look for a URL in the terminal output
- It will look like: `http://localhost:8888/?token=...`
- Copy this URL and paste it into your browser

7.3 Create and Test a Notebook

1. In Jupyter, click "New" → "Python 3 (ipykernel)"
2. In the first cell, type:

```
import torch
import matplotlib.pyplot as plt
import numpy as np

print("Setup successful!")
print(f"PyTorch version: {torch.__version__}")

# Simple test
x = torch.randn(5)
print(f"Random tensor: {x}")
```

3. Press `Shift + Enter` to run the cell
4. If you see "Setup successful!" and no errors, everything is working!
5. Close the notebook (File → Close and Halt)
6. Stop Jupyter by pressing `Ctrl + C` twice in the terminal

8 Step 7: Daily Workflow

Every time you want to work on the course:

8.1 Starting Your Work Session

1. Open Terminal (`Ctrl + Alt + T`)
2. Navigate to your project folder:

```
cd ~/DeepLearning
```

3. Activate virtual environment:

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

4. Start Jupyter:

```
jupyter notebook
```

5. Work in your notebooks

8.2 Ending Your Work Session

1. Save your notebooks (`Ctrl + S`)
2. Close notebooks in Jupyter (File → Close and Halt)
3. Stop Jupyter: `Ctrl + C` twice in terminal
4. Deactivate virtual environment:

```
deactivate
```

5. Close terminal

Tip

Create a shell alias to make activation easier:

```
# Add to ~/.bashrc or ~/.zshrc
alias dlenv='cd ~/DeepLearning && source deep_learning_env/bin/
    activate'
```

Then you can just type `dlenv` to navigate and activate!

9 Optional: IDE Setup

While Jupyter Notebooks are great for exercises, you may want a full IDE for projects.

9.1 VS Code (Recommended)

9.1.1 Installation

Ubuntu/Debian:

```
# Download .deb package
wget -O vscode.deb 'https://code.visualstudio.com/sha/download?build=
    stable&os=linux-deb-x64'

# Install
sudo apt install ./vscode.deb
```

Fedora:

```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
sudo dnf install code
```

Arch:

```
yay -S visual-studio-code-bin
```

9.1.2 Setup for Python

1. Launch VS Code: `code`
2. Click Extensions icon (left sidebar) or press `Ctrl + Shift + X`
3. Search for and install:
   - "Python" by Microsoft
   - "Jupyter" by Microsoft
4. Open your DeepLearning folder: File → Open Folder
5. Select Python interpreter:
   - Press `Ctrl + Shift + P`
   - Type "Python: Select Interpreter"
   - Choose the one in `deep_learning_env/bin/python`
6. You can now open and run .ipynb files directly in VS Code!

9.2 PyCharm Community (Alternative)

9.2.1 Installation

Ubuntu (Snap):

```
sudo snap install pycharm-community --classic
```

Manual Download:

1. Go to `https://www.jetbrains.com/pycharm/download/#section=linux`
2. Download Community Edition
3. Extract and run

9.2.2 Configuration

1. Open PyCharm
2. Open your DeepLearning folder
3. Configure interpreter:
   - File → Settings → Project → Python Interpreter
   - Click gear icon → Add
   - Choose "Existing environment"
   - Navigate to: `/DeepLearning/deep_learning_env/bin/python`

10 Optional: GPU Support

> **Note**
>
> GPU support is **not required** for this course, but can speed up training significantly from Week 4 onwards. Skip this section if you don't have an NVIDIA GPU.

10.1 Check if You Have NVIDIA GPU

```
lspci | grep -i nvidia
```

If you see output listing an NVIDIA GPU, you can set up CUDA. Otherwise, skip this section.

10.2 Install NVIDIA Drivers

Ubuntu:

```
# Check recommended driver
ubuntu-drivers devices

# Install recommended driver
sudo ubuntu-drivers autoinstall

# Or install specific version
sudo apt install nvidia-driver-535

# Reboot
sudo reboot
```

Fedora:

```
sudo dnf install akmod-nvidia
sudo reboot
```

Arch:

```
sudo pacman -S nvidia nvidia-utils
sudo reboot
```

10.3 Verify Driver Installation

After reboot:

```
nvidia-smi
```

Should show your GPU information.

10.4 Install CUDA Toolkit

Ubuntu:

```
# Add CUDA repository
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204
   /x86_64/cuda-keyring_1.1-1_all.deb
sudo dpkg -i cuda-keyring_1.1-1_all.deb
sudo apt update

# Install CUDA
sudo apt install cuda-toolkit-11-8
```

Other distributions: See `https://developer.nvidia.com/cuda-downloads`

10.5 Reinstall PyTorch with CUDA

1. Activate your virtual environment
2. Uninstall current PyTorch:

   ```
   pip uninstall torch torchvision torchaudio
   ```

3. Install PyTorch with CUDA 11.8:

   ```
   pip install torch torchvision torchaudio --index-url https://
      download.pytorch.org/whl/cu118
   ```

4. Or for CUDA 12.1:

   ```
   pip install torch torchvision torchaudio --index-url https://
      download.pytorch.org/whl/cu121
   ```

10.6 Verify GPU Support

```
import torch

print(f"CUDA available: {torch.cuda.is_available()}")

if torch.cuda.is_available():
```

```
    print(f"GPU: {torch.cuda.get_device_name(0)}")
    print(f"CUDA version: {torch.version.cuda}")

    # Test GPU computation
    x = torch.randn(3, 3).cuda()
    print(f"Tensor on GPU: {x.device}")
```

Should print:

```
CUDA available: True
GPU: NVIDIA GeForce RTX 3060
CUDA version: 11.8
Tensor on GPU: cuda:0
```

Important

GPU setup can be tricky. If you have issues, stick with CPU for now. We can revisit GPU setup in Week 4 when it becomes more beneficial.

11 Troubleshooting Common Issues

11.1 Python Not Found

Problem

Error: `python3: command not found`

Solution

1. Install Python (see Step 2)
2. Check if it's installed but not in PATH:

```
which python3
whereis python3
```

3. Try with version number:

```
python3.11 --version
```

11.2 venv Module Not Found

Problem

Error: `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
```

11.3 Virtual Environment Won't Activate

Problem

After running activation, no `(deep_learning_env)` prefix appears.

✔ Solution

1. Make sure you use `source`:

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

2. Check if venv was created successfully:

   ```
   ls deep_learning_env/bin/
   ```

   Should see `activate` file

3. Try recreating:

   ```
   rm -rf deep_learning_env
   python3 -m venv deep_learning_env
   source deep_learning_env/bin/activate
   ```

11.4 Permission Denied

Problem

Getting "Permission denied" errors when creating venv or installing packages.

✔ Solution

NEVER use sudo with pip or venv!

1. Make sure you own the directory:

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

2. If you accidentally used sudo, fix permissions:

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

3. Delete venv and recreate without sudo:

```
rm -rf deep_learning_env
python3 -m venv deep_learning_env
```

11.5 Externally-Managed-Environment Error

Problem

Error: `externally-managed-environment` when installing packages (Ubuntu 23.04+).

✔ Solution

This is exactly why we use virtual environments!

1. Make sure you created a virtual environment
2. Make sure it's activated (see `(deep_learning_env)` prefix)
3. If activated and still getting error, recreate venv:

```
deactivate
rm -rf deep_learning_env
python3 -m venv deep_learning_env
source deep_learning_env/bin/activate
pip install -r requirements.txt
```

Do NOT use `--break-system-packages`! Use virtual environments instead.

11.6 Package Installation Fails

Problem

`pip install` fails with timeout or connection errors.

✔ Solution

Try PyTorch CDN (faster):

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

Increase timeout:

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

Check internet connection:

```
ping pypi.org
```

11.7 Jupyter Notebook Won't Start

Problem

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

✔ Solution

1. Verify Jupyter is installed:

   ```
   pip install jupyter notebook
   ```

2. Try different port:

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

3. Check if port 8888 is in use:

   ```
   lsof -i :8888
   ```

4. Manually copy URL from terminal to browser

11.8 Module Not Found in Jupyter

Problem

`ModuleNotFoundError: No module named 'torch'` in Jupyter.

✔ Solution

Jupyter is using wrong Python environment!

1. Activate virtual environment BEFORE starting Jupyter
2. Check Python path in notebook:

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

   Should point to `deep_learning_env`

3. Register environment as kernel:

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

   Then: Kernel → Change kernel → deep_learning_env

11.9 SSL Certificate Errors

Problem

`SSL: CERTIFICATE_VERIFY_FAILED` when installing packages.

✔ Solution

Update certificates:

```
# Ubuntu/Debian
sudo apt install ca-certificates
sudo update-ca-certificates

# Fedora
sudo dnf install ca-certificates
```

Temporary workaround (not recommended):

```
pip install --trusted-host pypi.org --trusted-host files.
    pythonhosted.org torch
```

12 Getting Help

If you've tried the troubleshooting steps and still have issues:

1. Document your error:
   - Copy full error message
   - Note your Linux distribution and version
   - Note Python version (`python3 --version`)
   - What you were trying to do
   - What you've already tried
2. Get help:

- Post in course forum with documentation
- Email instructor with details
- Come to office hours

3. Temporary workaround:
   - Use GitHub Codespaces (see separate guide)
   - Continue with exercises while troubleshooting local setup

13 Next Steps

1. Download Week 1 exercise notebooks from course repository
2. Place them in your `DeepLearning` folder
3. Activate virtual environment
4. Start Jupyter Notebook
5. Open `week1_exercises_starter.ipynb`
6. You're ready to start coding!

Tip

Create a habit: Always activate your virtual environment before working on course materials!

14 Quick Reference

14.1 Essential Commands

```
# Navigate to project
cd ~/DeepLearning

# Activate virtual environment
source deep_learning_env/bin/activate

# Start Jupyter
jupyter notebook

# Stop Jupyter
# Press Ctrl+C twice

# Deactivate virtual environment
deactivate
```

14.2 Troubleshooting Quick Fixes

- **python3: command not found**: Install Python (Step 2)
- **venv module not found**: Install `python3-venv` package
- **Virtual env won't activate**: Use `source` command

- **Module not found in Jupyter**: Activate venv before starting Jupyter
- **Permission denied**: Don't use `sudo`, check folder ownership
- **externally-managed-environment error**: Use virtual environment!