Introduction to Deep Learning
Local Development Environment Setup Guide
macOS

MSc Artificial Intelligence

Contents

1 Overview

This guide will help you set up a local Python development environment on **macOS** 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.

ℹ Note

Estimated time: 30-45 minutes depending on your internet speed.
This guide covers both Intel and Apple Silicon (M1/M2/M3) Macs.

2 Step 1: Check Existing Python Installation

macOS usually comes with Python, but it's often an older version.

2.1 Open Terminal

Press `Cmd + Space`, type "Terminal", and press Enter.

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)

If you see an older version or error:

- Continue to Step 2 to install Python

Tip

Don't use `python` (without the 3) - this often points to Python 2.7 on macOS.

3 Step 2: Install Python

3.1 Option 1: Using Homebrew (Recommended)

Homebrew is the most popular package manager for macOS.

3.1.1 Install Homebrew (if not already installed)

Check if Homebrew is installed:

```
brew --version
```

If not installed, install it:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/
    install/HEAD/install.sh)"
```

Follow the instructions to add Homebrew to your PATH.

3.1.2 Install Python

```
brew install python@3.11
```

Wait 2-5 minutes for installation to complete.

3.1.3 Verify Installation

```
python3 --version
```

Should show Python 3.11.x

3.2 Option 2: Official Python Installer

If you prefer not to use Homebrew:

1. Go to `https://www.python.org/downloads/macos/`
2. Click "Download Python 3.11.x" (latest version)
3. Open the downloaded `.pkg` file
4. Follow the installation wizard
5. Use default options and click "Install"
6. Enter your password when prompted
7. Click "Close" when finished

3.2.1 Install Certificates (Important!)

After installing Python, you need to install SSL certificates:

```
# Replace 3.11 with your Python version
/Applications/Python\ 3.11/Install\ Certificates.command
```

This prevents SSL errors when downloading packages.

3.3 Verify Installation

Open a new terminal and run:

```
python3 --version
pip3 --version
```

Both should work without errors.

4 Step 3: Install Xcode Command Line Tools

These are required for some Python packages.

4.1 Install Command Line Tools

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

- Click "Install" in the popup window
- Accept the license agreement
- Wait 5-10 minutes for installation

Note

If you get "command line tools are already installed", that's fine - skip this step.

5 Step 4: Create Project Folder

5.1 Navigate to Documents

```
cd ~/Documents
```

5.2 Create and Navigate to Project Folder

```
mkdir DeepLearning
cd DeepLearning
```

Tip

You can verify your current location with:

```
pwd
```

Should show: `/Users/yourusername/Documents/DeepLearning`

6 Step 5: Create Virtual Environment

A virtual environment keeps your project dependencies isolated.

6.1 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.

6.2 Activate the Virtual Environment

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

6.3 Verify Activation

After activation, you should see:

```
(deep_learning_env) computername:DeepLearning username$
```

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!

7 Step 6: Install Required Packages

Important

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

7.1 Upgrade pip

First, upgrade pip to the latest version:

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

7.2 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:

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

7.3 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.

7.4 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
```

Note

For Apple Silicon Macs (M1/M2/M3): PyTorch will automatically install the ARM64 version optimized for your processor. No special steps needed!

8 Step 7: Verify Installation

8.1 Test Python and PyTorch

1. Start Python interpreter:

   ```
   python3
   ```

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

   ```
   import torch
   import platform

   print(f"PyTorch version: {torch.__version__}")
   print(f"Python architecture: {platform.machine()}")
   print(f"CUDA available: {torch.cuda.is_available()}")
   print(f"MPS available: {torch.backends.mps.is_available()}")

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

3. You should see output similar to:

   ```
   PyTorch version: 2.x.x
   Python architecture: arm64  # or x86_64 for Intel Macs
   CUDA available: False
   MPS available: True  # or False for Intel Macs
   Test tensor: tensor([1, 2, 3])
   ```

4. Exit Python:

```
exit()
```

Note

For M1/M2/M3 Macs: `MPS available: True` means you can use Apple's Metal Performance Shaders for GPU acceleration (optional).
For Intel Macs: Both CUDA and MPS will be False - you'll use CPU, which is fine for this course.

8.2 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

8.3 Create and Test a Notebook

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

   ```python
   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

9 Step 8: Daily Workflow

Every time you want to work on the course:

9.1 Starting Your Work Session

1. Open Terminal (`Cmd + Space` → "Terminal")
2. Navigate to your project folder:
   ```
   cd ~/Documents/DeepLearning
   ```
3. Activate virtual environment:
   ```
   source deep_learning_env/bin/activate
   ```
4. Start Jupyter:
   ```
   jupyter notebook
   ```
5. Work in your notebooks

9.2 Ending Your Work Session

1. Save your notebooks (`Cmd + 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 (`Cmd + Q`)

Tip

Create a shell alias to make activation easier:

```
# Add to ~/.zshrc (or ~/.bash_profile for older macOS)
alias dlenv='cd ~/Documents/DeepLearning && source
    deep_learning_env/bin/activate'
```

Then you can just type `dlenv` to navigate and activate!
To reload your shell config: `source  /.zshrc`

10 Optional: IDE Setup

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

10.1 VS Code (Recommended)

10.1.1 Installation

1. Go to `https://code.visualstudio.com/`
2. Click "Download Mac Universal" (works for both Intel and Apple Silicon)
3. Open the downloaded `.zip` file
4. Drag `Visual Studio Code.app` to your Applications folder
5. Launch VS Code from Applications

10.1.2 Setup for Python

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

10.2 PyCharm Community (Alternative)

1. Go to `https://www.jetbrains.com/pycharm/download/#section=mac`
2. Download Community Edition (free)
3. Open the `.dmg` file
4. Drag PyCharm to Applications
5. Launch PyCharm
6. Open your DeepLearning folder
7. Configure interpreter:
   - PyCharm → Settings → Project → Python Interpreter
   - Click gear icon → Add
   - Choose "Existing environment"
   - Navigate to: `/Documents/DeepLearning/deep_learning_env/bin/python`

11 Optional: Apple Silicon GPU Support (M1/M2/M3)

Note

Apple Silicon Macs can use Metal Performance Shaders (MPS) for GPU acceleration. This is optional and not required for Week 1, but can speed up training from Week 4 onwards.

11.1 Check MPS Availability

MPS should already be available if you have an M1/M2/M3 Mac:

```python
import torch

print(f"MPS available: {torch.backends.mps.is_available()}")
print(f"MPS built: {torch.backends.mps.is_built()}")
```

Should print:

```
MPS available: True
MPS built: True
```

11.2 Using MPS in Your Code

```python
import torch

# Set device to MPS if available, otherwise CPU
device = torch.device("mps" if torch.backends.mps.is_available() else "
    cpu")
print(f"Using device: {device}")

# Create tensor on MPS
x = torch.randn(3, 3).to(device)
print(f"Tensor device: {x.device}")

# Or create directly on MPS
y = torch.randn(3, 3, device=device)
```

Note

You'll learn more about device management in later weeks. For now, just knowing it's available is enough!

11.3 MPS Limitations

Some operations may not be supported on MPS yet. If you encounter errors, fall back to CPU:

```python
try:
    x = x.to('mps')
except:
    print("Operation not supported on MPS, using CPU")
    x = x.to('cpu')
```

12 Troubleshooting: Architecture Issues (M1/M2/M3)

12.1 Check Python Architecture

Make sure you're using ARM64 Python, not x86_64 (Rosetta):

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

Should print: `arm64`
If it prints `x86_64`, you installed x86 Python. Reinstall using Homebrew:

```
# Uninstall old Python
brew uninstall python@3.11

# Make sure Homebrew is ARM64
arch -arm64 brew install python@3.11
```

12.2 Rosetta 2

Some packages may still require Rosetta 2. If prompted, install it:

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

13 Troubleshooting Common Issues

13.1 Python Not Found

Problem

Error: `python3: command not found`

Solution

1. Install Python (see Step 2)
2. Check if Homebrew installed it:

```
which python3
/usr/local/bin/python3 --version
```

3. Try with version number:

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

13.2 xcrun Error

Problem

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

✓ Solution

Install Xcode Command Line Tools:

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

Click "Install" and wait for completion.

13.3 SSL Certificate Errors

Problem

`SSL: CERTIFICATE_VERIFY_FAILED` when installing packages.

✓ Solution

Run the certificate installer that comes with Python:

```
# Replace 3.11 with your version
/Applications/Python\ 3.11/Install\ Certificates.command
```

Or if installed via Homebrew:

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

13.4 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:

   ```
   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
   ```

13.5 Architecture Issues (Apple Silicon)

Problem

Package installation fails with architecture errors on M1/M2/M3 Mac.

✓ Solution

Check Python architecture:

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

Should print `arm64`, not `x86_64`.
If showing x86_64:

1. You have x86 Python (running under Rosetta)
2. Uninstall and reinstall ARM64 version:

   ```
   # If using Homebrew
   brew uninstall python@3.11
   arch -arm64 brew install python@3.11
   ```

3. Or download official ARM64 installer from python.org

Install Rosetta 2 if needed:

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

13.6 Homebrew Issues

Problem

Homebrew commands not working or packages not found.

✓ Solution

Add Homebrew to PATH:
For Apple Silicon (M1/M2/M3):

```
echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> ~/.zshrc
source ~/.zshrc
```

For Intel Macs:

```
echo 'eval "$(/usr/local/bin/brew shellenv)"' >> ~/.zshrc
source ~/.zshrc
```

Update Homebrew:

```
brew update
brew upgrade
```

13.7 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 connection:

```
ping pypi.org
```

13.8 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

13.9 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

13.10 MPS Not Working (Apple Silicon)

Problem

`torch.backends.mps.is_available()` returns False on M1/M2/M3 Mac.

✓ Solution

1. Make sure you have ARM64 Python (not x86_64)
2. Check macOS version (need macOS 12.3+):

   ```
   sw_vers
   ```

3. Update PyTorch to latest version:

   ```
   pip install --upgrade torch torchvision torchaudio
   ```

4. Some operations may not be MPS-compatible yet - use CPU as fallback

13.11 Permission Issues

Problem

Getting permission errors when creating files or installing packages.

✓ Solution

1. Make sure you're working in your home directory (`~/Documents`)
2. Never use `sudo` with pip
3. If you used sudo accidentally, fix ownership:

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

14 Getting Help

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

1. Document your error:
   - Copy full error message
   - Note your macOS version (`sw_vers`)
   - Note your Mac type (Intel or M1/M2/M3)
   - Note Python version and architecture
   - 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

15 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!

16 Quick Reference

16.1 Essential Commands

```
# Navigate to project
cd ~/Documents/DeepLearning

# Activate virtual environment
source deep_learning_env/bin/activate
```

```
# Start Jupyter
jupyter notebook

# Stop Jupyter
# Press Ctrl+C twice

# Deactivate virtual environment
deactivate
```

16.2 Troubleshooting Quick Fixes

- **python3: command not found**: Install Python (Step 2)
- **SSL certificate errors**: Run Install Certificates.command
- **xcrun: error**: Install Xcode Command Line Tools (Step 3)
- **Virtual env won't activate**: Use `source` command
- **Module not found in Jupyter**: Activate venv before starting Jupyter
- **Architecture is x86_64 on M1/M2/M3**: Reinstall ARM64 Python

16.3 Default Shell Note

Note

macOS Catalina (10.15) and later use `zsh` as the default shell. Configuration file: `/.zshrc`
Older versions use `bash`. Configuration file: `/.bash_profile`