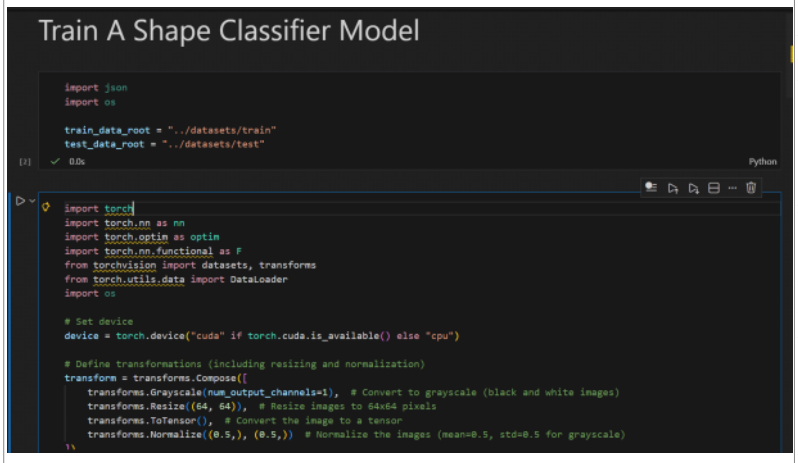
Debugging

Tuesday, November 18, 2025 7:49 PM

problem	solving

	**From official documentation**
	VS Code + venv + Jupyter (Windows) – Quick Cheat Sheet
	0. Mental model
	Always keep **three things pointing to the same place**:
	1. The **project folder** in VS Code
	2. The **virtual environment** (`.venv`)
	3. The **Jupyter kernel** used by the notebook
	If all three match, `import torch` etc. will work.
	1. Open the correct folder in VS Code
	1. In VS Code: **File → Open Folder…**
	2. Choose your project root, for example:
	`D:\MasterSRC\in0utCampusPractice\silverpond_enter_project\intern-skills-silverpond`
	Everything below assumes your terminal path looks like this.
	2. Create and activate the virtual environment
	Open a **Terminal** in VS Code (PowerShell is fine):
	`# 1) Create venv in project folder (run only once)`
	`python -m venv .venv`
	`# 2) Activate venv (every new terminal)`
	`.\.venv\Scripts\Activate.ps1`
	You should see `(.venv)` at the start of the prompt.
	If not, the venv is not active.
	3. Install project dependencies
	Still in the same terminal (with `(.venv)`):
	`pip install -r requirements-cpu.txt`
	If you just need `torch` / `torchvision` manually, you can also do:
	`pip install torch torchvision`
	(But for Silverpond task, the provided `requirements-cpu.txt` is better.)
	4. Install and register an ipykernel (once per venv)
	This step makes Jupyter see your venv as a **named kernel**:
	`pip install ipykernel`
	`python -m ipykernel install --user`
	`--name silverpond-venv`
	`--display-name "Python (silverpond .venv)"`
	• `--name` is the internal kernel name.
	• `--display-name` is what you see in the Jupyter kernel menu.
	You only need to do this **once** for this venv.
	5. Select the right kernel in the notebook
	1. Open `notebooks/shape_classifier.ipynb` in VS Code.
	2. Top-right corner, click on the **kernel name** (e.g. "Python 3.11.x").
	3. Choose:
	`Python (silverpond .venv)`
	If you see this name, it means VS Code found the kernel you registered.
	6. Sanity check inside the notebook
	In a new cell, run:
	`import sys, torch`
	`print(sys.executable)`
	`print("torch version:", torch.__version__)`
	Expected result:
	• sys.executable ends with something like:
	`...\intern-skills-silverpond\.venv\Scripts\python.exe`
	• torch version: prints a valid version (e.g. `2.9.1+cpu`), **no error**.
	If both are true, your environment is **correctly wired**:
	• Notebook → venv Python
	• torch (and other libs) installed in that venv
	7. Run the workflow
	Now you can safely:
	1. Run the cell that generates the dataset
	(`python generate_classification_dataset.py` or notebook cell)
	2. Run training cells (`train_model(...)`)
	3. Run evaluation (`test(model, test_loader)`)
	4. Run visualization (`show_prediction(...)`)
	If later you open a new VS Code session:
	1. Open the **same project folder**.
	2. Activate venv again:
	`.\.venv\Scripts\Activate.ps1`
	3. In the notebook, check that kernel is still **Python (silverpond .venv)**.
	8. Optional: clean notebook outputs before commit (nbstripout)
	In the venv:
	`pip install nbstripout`
	`nbstripout notebooks/shape_classifier.ipynb`
	This removes big cell outputs so your git diff is clean.
	"What went wrong before?" (Very short explanation)
	Earlier problems were caused by **environment mismatch**:
	• VS Code Jupyter kernel was using **global Python** (`D:\python\python.exe`)
	• You installed `torch` into **.venv**, not into global Python
	• So notebook couldn't find `torch`, even though terminal inside `.venv` worked
	Now all three are aligned on `.venv`, so things are stable.
	Useful official docs (for reference)
	They are standard docs people use for this setup.
	Visual Studio Code – Python environments
	https://code.visualstudio.com/docs/python/environments
	Visual Studio Code – Jupyter Notebooks
	https://code.visualstudio.com/docs/datascience/jupyter-notebooks
	Python – venv module (official docs)
	https://docs.python.org/3/library/venv.html
	Jupyter – Using different Python environments & kernels
	https://ipython.readthedocs.io/en/stable/install/kernel_install.html

Since tensors do not have an append method, the workaround is to initialize them as a Python list all_preds = [].	First, collect the data using a Python list, and then merge it all at once using torch.cat or torch.stack. **Useful Helper docs (for reference)** Stackoverflow https://www.bing.com/ck/a?!&&p=f5d753484c4b8c7789e34f4b981a8d0a115859 2e441d96911cd997f168cf9081JmltdHM9MTc2MzUxMDQwMA&ptn=3&ver=2&hsh=4&fclid=2fe3208d-c423-67bb-299e-36b3c533660a&psq=AttributeError%3a+%27Tensor%27+object+has+no+attribute+%27append%27+pytorch%22&u=a1aHR0cHM6Ly9zdGFja292ZXJmbG93LmNvbS9xdWVzdGlvbnMvNTUxNTM5MTUvYXR0cmlidXRlZXJyb3ItdGVuc29yLW9iamVjdC1oYXMtbm8tYXR0cmlidXRlLWFwcGVuZA

finish	git status git add notebooks/shape_classifier.ipynb git commit -m "Strip notebook outputs" git push origin master