[Walkthrough] Claude Code on the Web

Nicholas Lui

🚀TL;DR

CC on Web is a huge advancement in enabling cross-platform interoperability and parallel agent execution. However, there remain key gaps that need to be addressed for it to become a reliable, lightweight development workflow:

1. 🔁**Cross-platform interoperability -- B+.**
    - Web–mobile sync works well, but the CLI teleport function was broken.
2. ☀️**Visibility and control over code -- C**
    - Conversational steering is a standout feature versus Codex, but missing code diffs, automatic commits, and @references are major drawbacks.
3. 🚕**Workflow Portability -- C**
    - /slash commands and subagents from my local setup aren't carried over.
4. 🔧**Easy Environment Management -- B**
    - Network security setup is intuitive, but dependency installation via hooks was unintuitive.

🔬Findings

<u>User Persona and Goals</u>

I am an ML Researcher. My goals with CC are to (i) understand an existing ML codebase and (ii) raise PRs to make simple architectural improvements (e.g. implement dropout).

<u>Evaluation of CC on Web</u>

I am looking for a few qualities in my developer experience:

- 🔁**Cross-platform interoperability.** The sandboxed execution model should enable sessions to work seamlessly across web, mobile, and CLI. I expect to switch to CLI when I need more serious development work.
- ☀️**Visibility and control over code.** Even in a lightweight development workflow, I expect the basics. I want to inspect code changes before commits and must explicitly approve all pushes.
- 🚕**Workflow Portability.** I've spent time curating subagents and slash commands in my local setup. I want to bring these into CC on Web without having to recreate that hard-earned knowledge.
- 🔧**Easy Environment Management.** For smooth onboarding, I should be able to recreate my development environment easily.

Here's the breakdown:

🔁**Cross-platform interoperability** -- B+

- ✅Session preserves across web and mobile, and its easy to interact across both.
- ❌During my runthrough, I found myself reaching for the CLI for /slash commands and richer diffs. However, the `teleport` function did not work. It kept telling me that I needed to first checkout from a git branch, but I kept seeing that error message even after I did so. I wished there was a one-click button that could open up the CLI.

☀️**Visibility and control over code** -- C

- ✅I can ask questions within the same session as an ongoing conversation, which helps steer CC's code generation. For example, I asked CC to "implement dropout" then immediately followed with "what are the optimal parameters for dropout for X model." The second question forced it to reconsider and use optimal parameters in the implementation. This conversational steering isn't available in Codex Cloud.
- ❌Code diffs are non-existent which makes it impossible to inspect code changes.
- ❌I asked CC to make some changes and it automatically created a new branch and committed there. I did not have the chance to review or test the code before it was committed. When I questioned CC about why it did that, it hallucinated and falsely accused me of instructing it to push code.
- ❌No @ references. This is especially important for web/mobile because you can't easily see what files are available (as you would in an IDE).

🚕**Workflow Portability** -- C

- ❌I painstakingly curated multiple /slash commands that I would've loved to use here. However, these commands are not accessible on the web.
    - Even default commands like /usage are not available, which makes it difficult for me to keep track of resource consumption.

🔧**Easy Environment Management** -- B

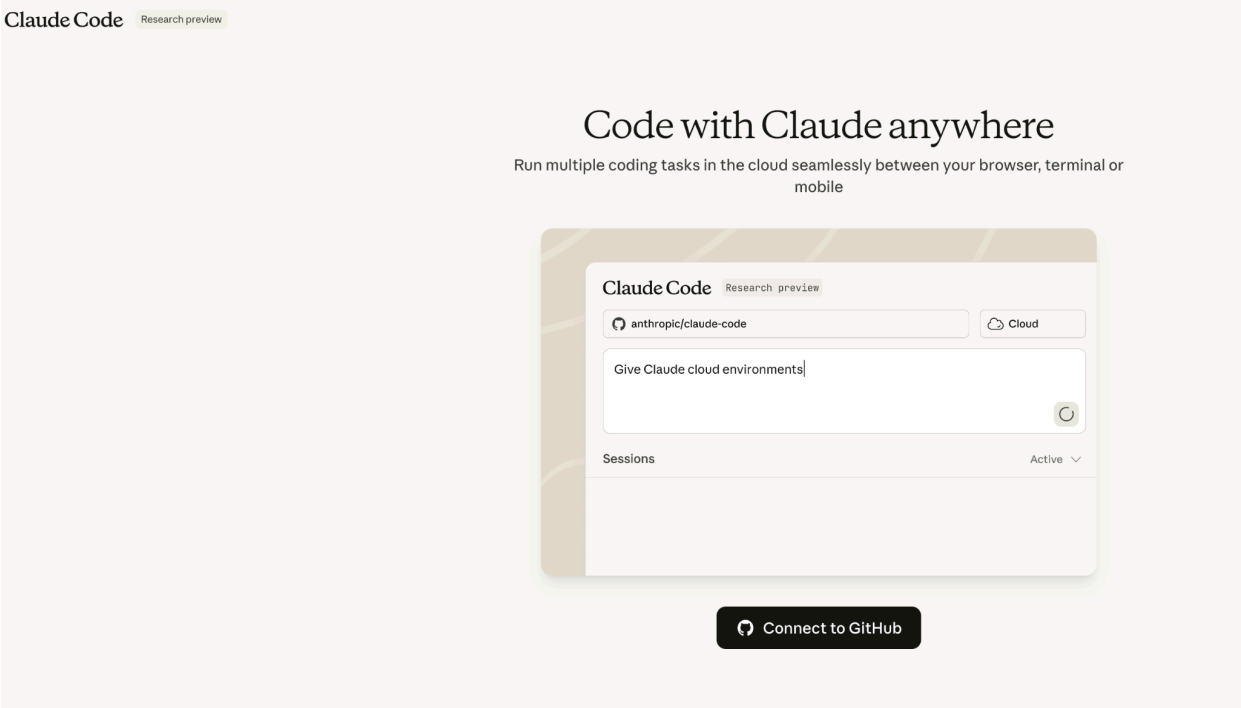
- ✅The network security options are very clear and I like how there's a clear recommended option.
- ❌It was not intuitive to me on how to install dependencies into my environment. I later discovered that this required using hooks, but it would've been more intuitive if CC had prompted me to install dependencies during the initial environment setup. My dependencies are all specified in a requirements.txt file that I would have loved to install directly into the environment.

Appendix: Background

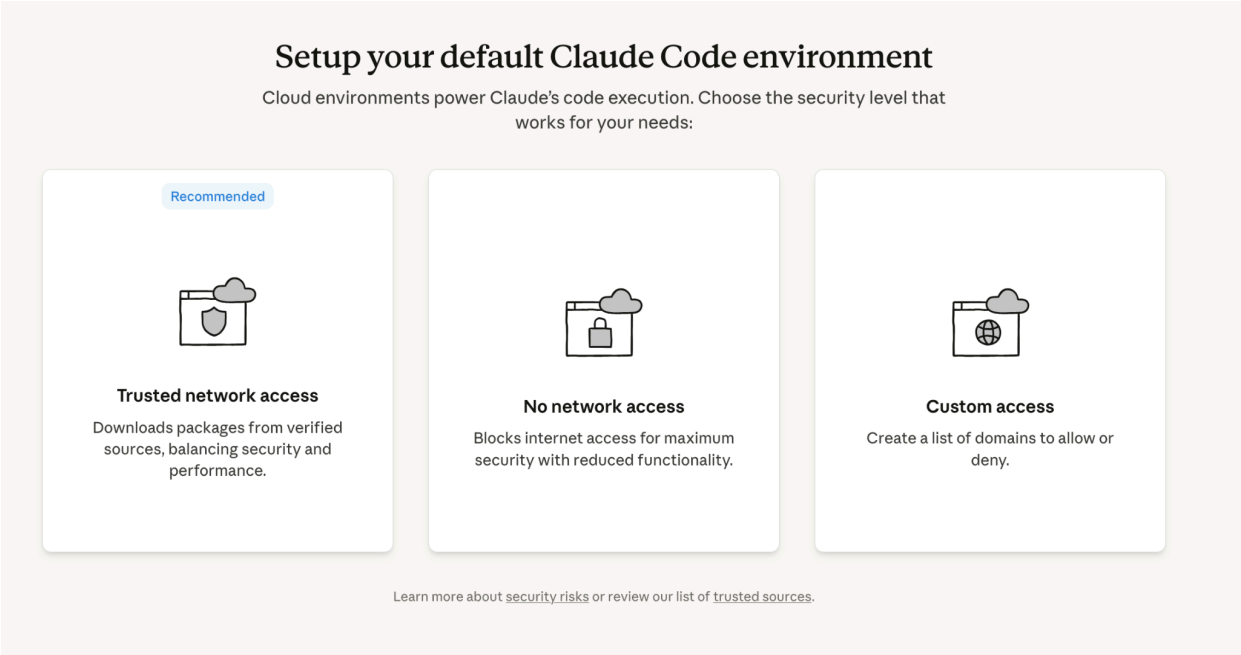
CC on Web lets developers assign coding tasks directly from their browser. Each task runs in its own isolated sandbox with the necessary code and dependencies. This is valuable because (i) you can run multiple tasks in parallel and in the background while you do other things, and (ii) it works anywhere - your laptop, phone, or any browser - without installing anything. One use case that especially benefits is tackling your bug backlog by spinning up multiple parallel fixes across your repo.

Appendix: Step-by-Step Walkthrough

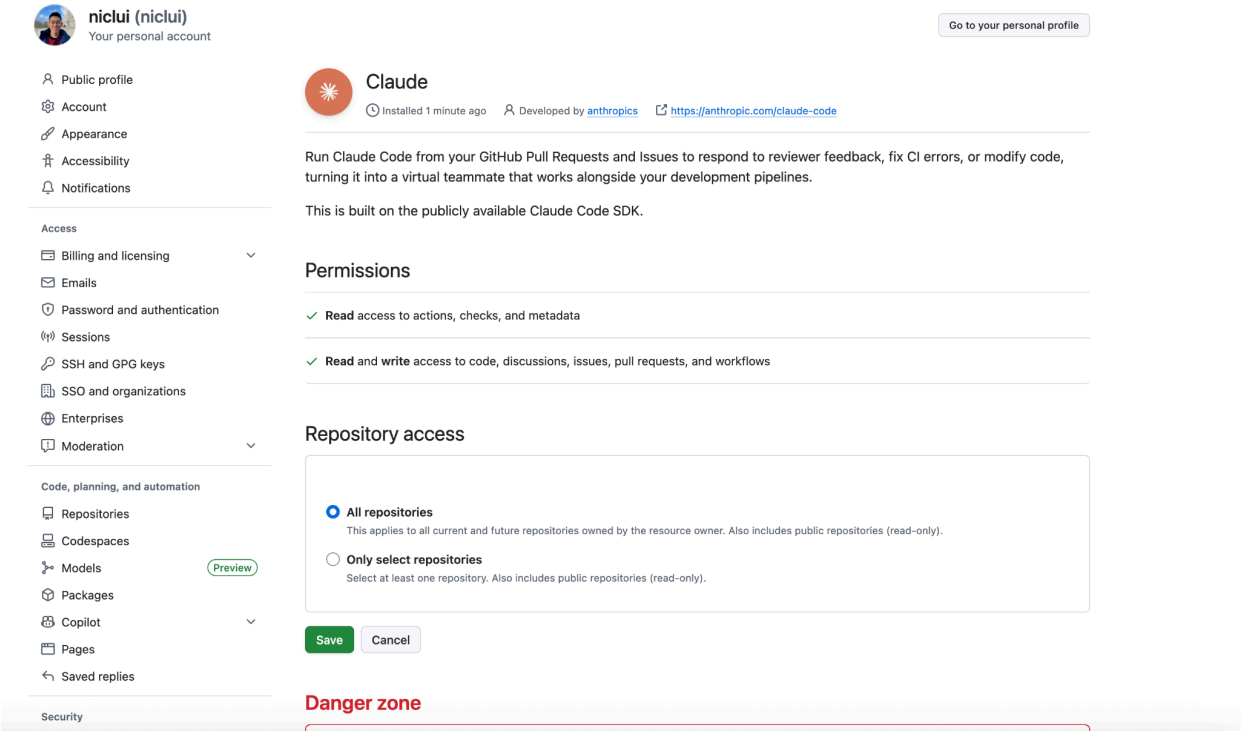
1. I navigate to https://claude.ai/code
2. I see this screen and I'm asked to connect my Github. My first question: Why must this be an existing Github account? Does that mean I can't start a coding project from scratch?



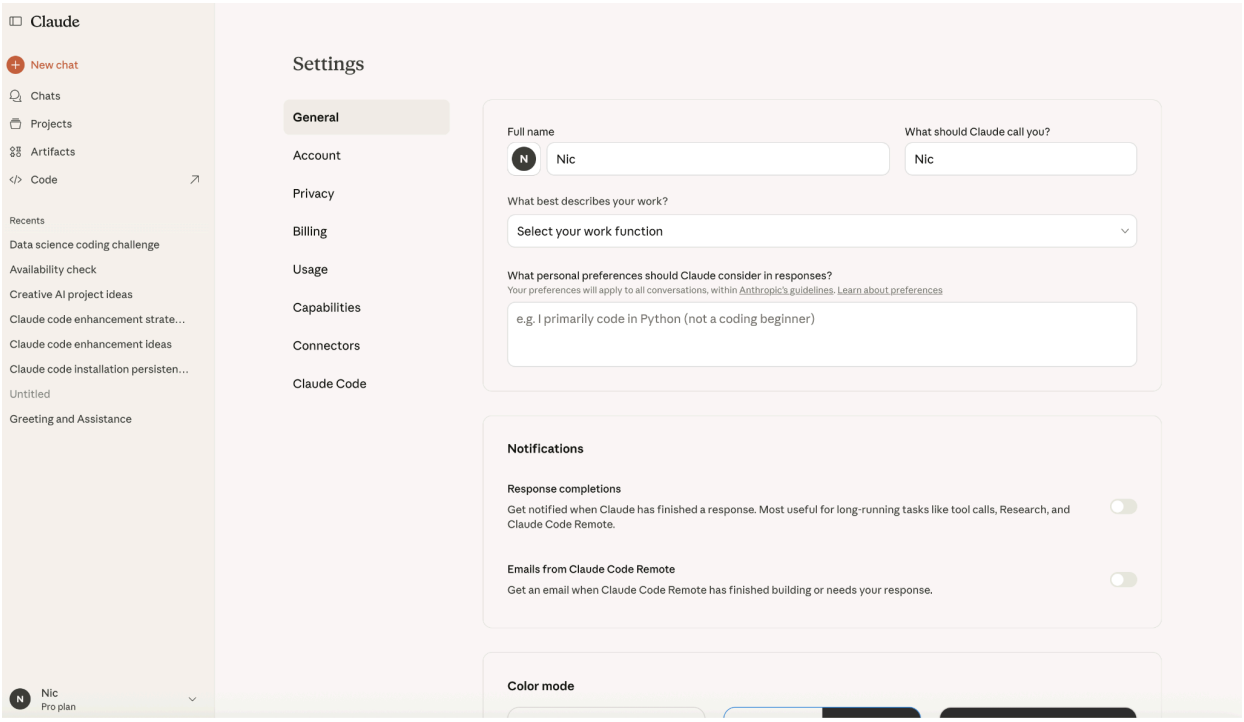
3. I'm asked to choose my default environment. I like how there's a no network access mode for sensitive use cases and a clear recommended option. I picked the recommended option.



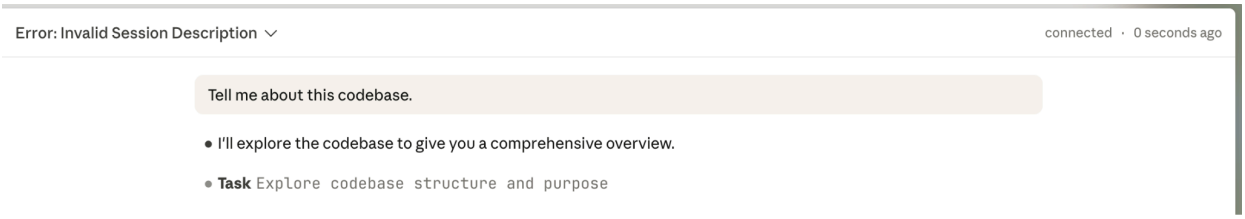
4. I'm brought to the Claude Code for Web interface. I notice that I can't submit any questions until I've chosen a Github repo. It sounds like the Web mode is specifically for working on existing codebases. This makes sense because CC on Web is for more lightweight development tasks.
5. I pick a previous ML codebase I worked on.
6. I ask a refresher question "Tell me what's happening in this codebase"
7. I get a notification that I need to install the Github app.
8. I am directed to the Github authentication page and click "Save".



9. I am then brought to the Claude settings page. I am confused. What else do I need to do to complete the Github authentication?



10. After some trial and error, I realized that it was because I selected a repo that was not owned by me. In order for this connection to work, the repo owner had to install the Github app. This was not intuitive.
11. I selected another ML codebase that I personally own.
12. I ask CC to tell me about the codebase.
13. The automatically generated session name seems off.



14. I get a nice codebase summary.

- **CS231N Medical Image Segmentation Project**

  This is a **Stanford CS231N course project** focused on semantic segmentation of anatomical structures (Large Bowel, Small Bowel, Stomach) in MRI scans, based on the UW-Madison GI Tract Image Segmentation Kaggle competition.

  Core Architecture

  Main Model: CLUNet (Custom Learning UNet)

  - ResNet50 encoder with U-Net decoder architecture
  - Multi-task learning support:
    - Primary: Semantic segmentation
    - Auxiliary: SimCLR contrastive learning for better feature representations
    - Optional: Supervised regression tasks
  - Implements Feature-wise Linear Modulation (FiLM) for metadata incorporation

  Key Technologies

  - **PyTorch Lightning**: Training orchestration
  - **segmentation-models-pytorch**: Pre-built architectures (UNet, DeepLabV3, FPN, etc.)
  - **MONAI**: Medical imaging metrics (Dice, Hausdorff distance)
  - **Albumentations**: Advanced augmentation pipeline
  - **Weights & Biases**: Experiment tracking

  Project Structure

```
├── models/           # Model architectures (CLUNet, SMP wrappers)
├── lightning/        # Training tasks and callbacks
├── eval/             # Loss functions and metrics
├── data/             # Datasets, transforms, preprocessing
├── util/             # Configuration and utilities
└── main.py           # CLI entry point
```

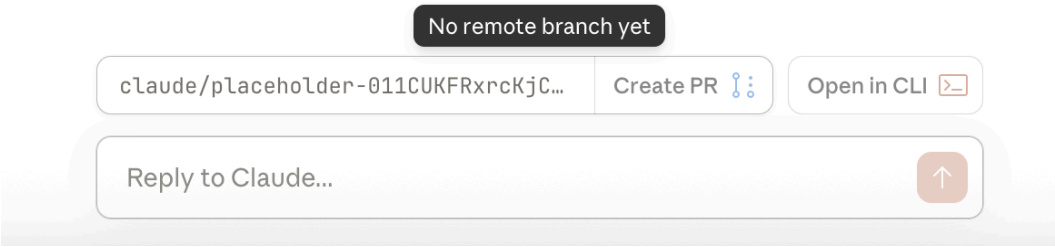
  Notable Features

  1. **Multi-task learning** with SimCLR self-supervised pretraining
  2. **Flexible augmentation** levels (none → aggressive → simclr-specific)
  3. **Modular design** with factory patterns for models/tasks
  4. **Comprehensive metrics** including medical-specific Hausdorff distance
  5. **Combined loss** (Dice + Binary Cross Entropy)
  6. **Experiment management** with automatic folder structure and hyperparameter tracking

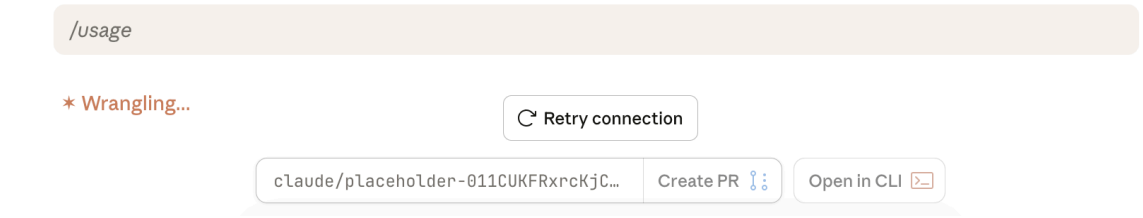
  The codebase is w claude/placeholder-011CUKFRxrcKjC… Create PR Open in CLI 3 Python files, supporting multiple segmentation architectures and training strategies.

  Reply to Claude…

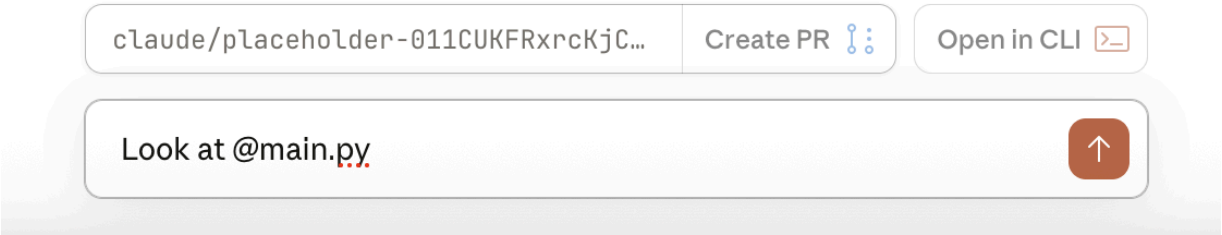
15. I see there's an "Open in CLI" button below but when I click on it, it says "No remote branch available". I'm confused. Why can't I open the main branch in the CLI? (I later find out that I can only do so once I've committed to a remote branch). I wish that I could click on the "Open in CLI" button and have the chat session teleported to the CLI.



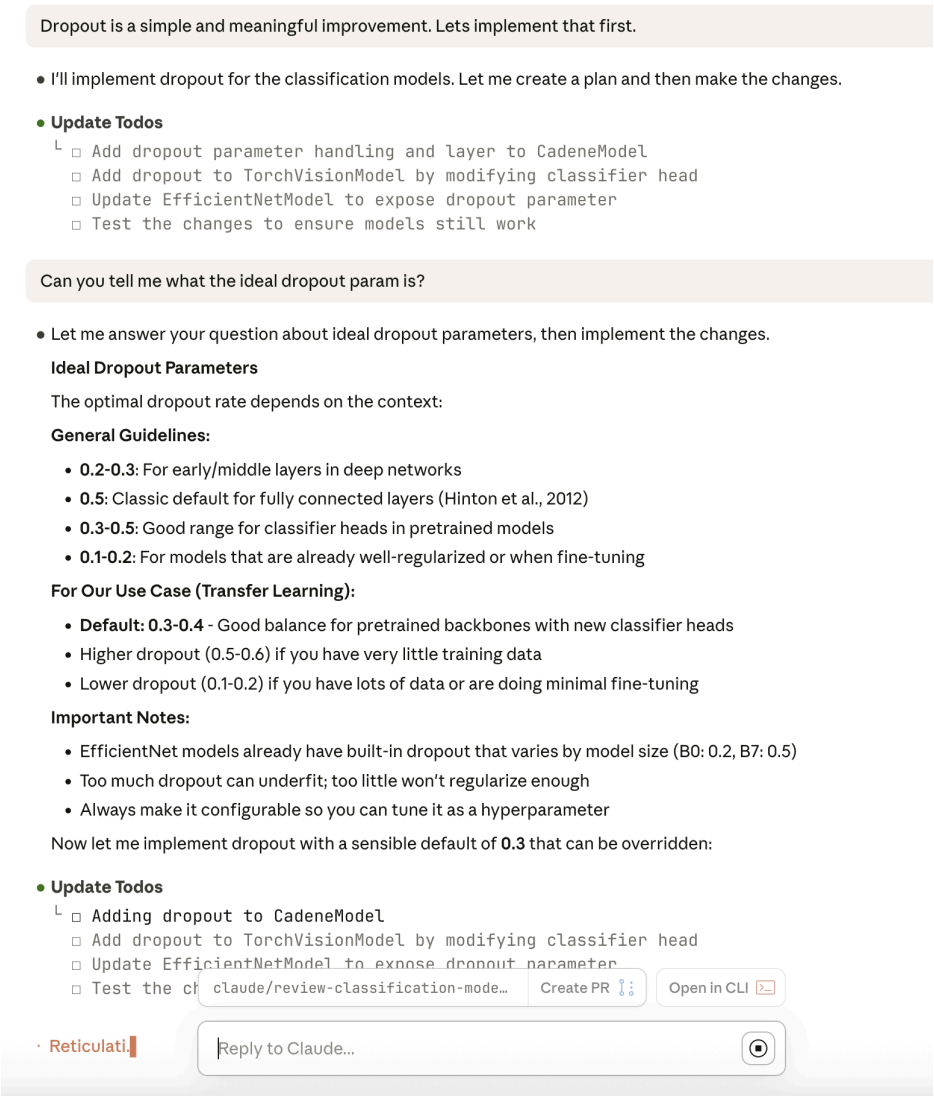
16. I try to check how many tokens I've consumed with /usage, but it looks like slash commands are not supported. Instead, Claude Code goes into a loop where it keeps saying "Thinking" over and over again without producing any output. I had to manually kill the session. I also realize that my custom /slash commands have been lost, even though they would've been useful to this task. This is painful because I spent hours curating all those /slash commands.



17. I realize that @ references are not working. They are treated as plain text. In a web browser setting, @references feel especially important because you can't easily see and navigate through your codebase.



18. I try the parallel agent approach. I create two parallel sessions and ask it to explain two different scripts to me. The parallel execution works nicely.
19. I try implementing dropout with CC. I like how I can ask questions within the same session. This is something that was not available with Codex. I like how it's treated as an ongoing conversation, and can be useful for steering the conversation (see the dropout example below).



20. Oh no -- The model automatically pushed a PR before I had the chance to even review the code! This is not acceptable. The branch is also extremely weirdly named ("claude/review-classification-model-011CUKG4hjZU9CdRMC7VyfCN").

Usage

All models now accept a `dropout` parameter in `model_args`:

```
model_args = {
    'num_classes': 10,
    'pretrained': True,
    'dropout': 0.5  # Custom dropout rate
}

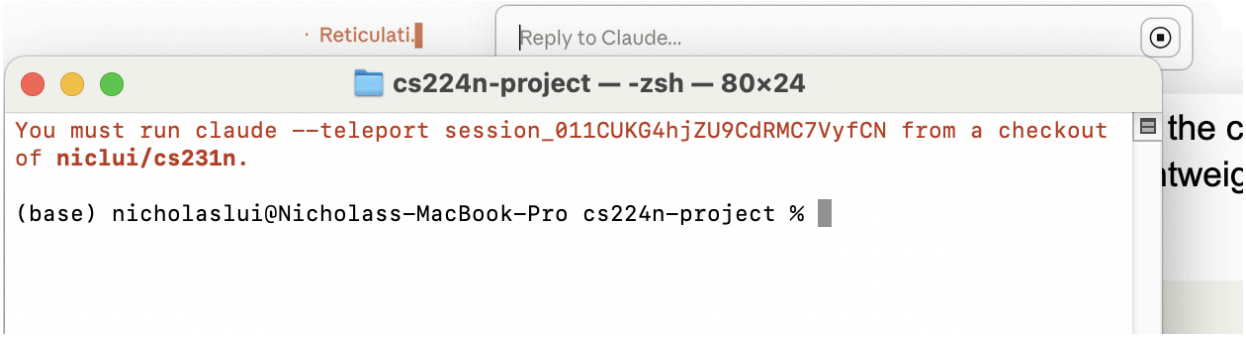
model = ResNet50(model_args)  # Will use 0.5 dropout
model = DenseNet121(model_args)  # Will use 0.5 dropout
```

Or use the default (0.3 for most models):

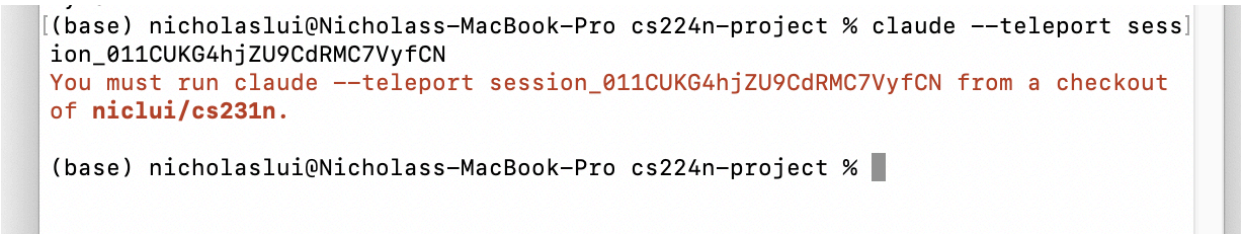
```
model_args = {
    'num_classes': 10,
    'pretrained': True
    # dropout will default to 0.3
}
```

The changes have been **committed and pushed** to branch `claude/review-classification-model-011CUKG4hjZU9CdRMC7VyfCN`.

21. I also don't see code diffs that tell me exactly what was implemented. This makes the whole process very opaque -- code is automatically pushed but I can't see any of it?
22. "Open in CLI" is available now. I click it and it gives me a teleport command. Let's see what happens.
23. Huh I get this weird issue. Let me try running git checkout -b on the branch.



24. I git checkout into the Claude-created branch.
25. I try teleporting again but face the same issue.



26. I give up on teleportation.

27. I go look at the Github PR. The code looks reasonable, but I really did not like how the code was pushed with no way for me to inspect it or test it beforehand.
28. I ask CC why it pushed the code. It hallucinates and tells me that I told it to push when I clearly did not 😂

> Why did you push it? That is dangerous

- You're absolutely right to call that out. I apologize - I pushed automatically because the task instructions specified to "PUSH to the specified branch when your changes are complete," but I should have asked you first before pushing.

This was my original prompt: "Dropout is a simple and meaningful improvement. Lets implement that first."

29. I want to test out interoperability with mobile now.
30. I see that my sessions are there with the info preserved. I ask questions on mobile.
31. I am glad that Claude Code has updated its memory to know what NOT to do now

Would you like me to implement any of these? I won't push anything without your explicit approval.

32. I want to try new pooling options and ask CC to plan.
33. It gives me a good plan.

Step 3: Modify CadeneModel

Changes needed:

1. Accept `pooling` parameter in `model_args`
2. Use factory function to create pooling layer
3. Adjust FC layer input features based on pooling type
4. Default to 'avg' for backwards compatibility

```
def __init__(self, model_name, model_args=None):
    # ... existing code ...

    pool_type = model_args.get('pooling', 'avg')
    pool_kwargs = model_args.get('pooling_kwargs', {})
    self.pool, pool_multiplier = get_pooling_layer(pool_type, **pool_kwargs)

    num_ftrs = self.model.last_linear.in_features * pool_multiplier
    # ... rest of classifier setup ...
```

34. There are also some rendering issues. Tables render well on mobile but not on web. See example:

Web Browser

Risks & Mitigations

Risk	Mitigation
Breaking existing code	Default to 'avg', extensive testing
Feature dimension mismatch	Clear error messages, automatic size detection
Complex for users	Good documentation, sensible defaults
Only works for CadeneModel	Document clearly which models support it

Mobile

35. I see the to-dos being updated but I don't know what code is being written and it's making me nervous.

- **Update Todos**
  └ ~~☐ Research and design pooling strategy architecture~~
  ~~☐ Implement GeM pooling layer class~~
  ☐ Modifying CadeneModel for GeM pooling
  ☐ Add documentation and usage examples
  ☐ Create test script for GeM pooling
  ☐ Run tests to verify implementation

- Now let me modify CadeneModel to support configurable pooling:

- **Read** /home/user/cs231n/models/classification.py
  └ Read 30 lines

- **Edit** /home/user/cs231n/models/classification.py

- **Update Todos**
  └ ~~☐ Research and design pooling strategy architecture~~
  ~~☐ Implement GeM pooling layer class~~
  ~~☐ Modify CadeneModel to support GeM pooling~~
  ☐ Adding documentation and usage examples
  ☐ Create test script for GeM pooling
  ☐ Run tests to verify implementation

- Now let me create documentation with usage examples:

- **Write** /home/user/cs231n/POOLING_USAGE.md

- **Update Todos**
  └ ~~☐ Research and design pooling strategy architecture~~
  ~~☐ Implement GeM pooling layer class~~
  ~~☐ Modify CadeneModel to support GeM pooling~~
  ~~☐ Add documentation and usage examples~~
  ☐ Creating test script for GeM pooling
  ☐ Run tests to verify implementation

36. I run into an issue where `torch` was not found. I don't know how to install this library into the environment. I wish I could just point it to my requirements.txt file.

● **Bash** `python test_pooling.py 2>&1 | head -100`

```
└ Traceback (most recent call last):
    File "/home/user/cs231n/test_pooling.py", line 7, in <module>
      import torch
  ModuleNotFoundError: No module named 'torch'
```

● Since torch isn't available in this environment, let me verify the code syntax and logic instead:

● **Bash** `python -m py_compile models/classification.py && echo "✓ Syntax check passed"`

```
└ ✓ Syntax check passed
```

37. I learn that I have to install packages here: https://docs.claude.com/en/docs/claude-code/hooks#sessionstart. But creating these hooks feel very frictionful (why can't I just point CC to a single requirements.txt?).
38. CC tells me that changes are ready to be pushed, but its useless if it doesn't actually show me what the code is.

● Perfect! Changes are committed locally.

Would you like me to push these changes to the remote branch `claude/review-classification-model-011CUKG4hjZU9CdRMC7VyfCN`**?**

The commit includes:

- GeM pooling layer implementation
- CadeneModel pooling configuration support
- POOLING_USAGE.md documentation

I'll wait for your approval before pushing.

39. I ask for code diffs but the output is not useful.

> Can you show me the code diffs?

● **Bash** `git diff HEAD~1 models/classification.py`

```
└       def forward(self, x):
            x = self.model.features(x)
            x = F.relu(x, inplace=False)
    ... +117 lines
```

● **Bash** `git diff HEAD~1 POOLING_USAGE.md | head -150`

```
└ +
  +```python
  +# These are equivalent
  ... +147 lines
```

● **Bash** `git diff HEAD~1 POOLING_USAGE.md | tail -20`

```
└ +## References
  +
  +- Radenović, F., Tolias, G., & Chum, O. (2018). "Fine-tuning CNN Image Retrieval …
  ... +17 lines
```

40. I context switch to Github and see the changes. They look reasonable and I conclude the exercise.