Workshop 2026-01-28

Záznam workshopu: SpecKit-2026-01-28.mp4
Summary workshopu: SpecKit.pdf

Instalace

Nainstalované Gemini-CLI:

https://geminicli.com/docs/get-started/installation/

Jen pozor, je třeba mít nainstalované NPM, součást Node.js: https://nodejs.org/en/download, na Mac OS stačí `brew install node`

Dále je potřeba mít Python package manager UV:

```
pip install uv
```

Začínáme

Vytvoření GIT repozitáře, já jsem použil Github: https://github.com/honzas83/nnvisu

Naklonujeme do adresáře:

```
git clone git@github.com:honzas83/nnvisu.git
```

A inicializujeme speckit

```
uvx --from git+https://github.com/github/spec-kit.git specify init --here
```

Stránky projektu a dokumentace zde: https://github.com/github/spec-kit

Speckit nainstalován

Máme novou sadu slash-příkazů v Gemini:

1. You're already in the project directory!
2. Start using slash commands with your AI agent:
   2.1 /speckit.constitution - Establish project principles
   2.2 /speckit.specify - Create baseline specification
   2.3 /speckit.plan - Create implementation plan
   2.4 /speckit.tasks - Generate actionable tasks

2.5 /speckit.implement - Execute implementation

Optional commands that you can use for your specs (improve quality & confidence)

○ /speckit.clarify (optional) - Ask structured questions to de-risk ambiguous areas before planning (run before /speckit.plan if used)
○ /speckit.analyze (optional) - Cross-artifact consistency & alignment report (after /speckit.tasks, before /speckit.implement)
○ /speckit.checklist (optional) - Generate quality checklists to validate requirements completeness, clarity, and consistency (after /speckit.plan)

Projekt

Cíl

Vizualizace trénovacího procesu neuronové sítě
Aktuálně nasazeno jako: https://dialogs.kky.zcu.cz/nnvisu/
GitHub repozitář: https://github.com/honzas83/nnvisu

Ústava

/speckit.constitution Create principles focused on code quality, testing standards, user experience consistency, and performance requirements. Use Python, Tornado, Websockets for realtime communication, ruff and mypy for quality assurance.

Můžeme se podívat do šablon .specify/templates/

Summary of Changes:
* Principles Established:
1. High Assurance Code Quality: Enforced via ruff and mypy (strict).
2. Realtime-First Architecture: Built on Tornado and Websockets with a non-blocking async mandate.
3. Rigorous Testing Standards: Test-first approach with focus on stability.
4. User Experience Consistency: Predictable, responsive UI with immediate feedback.
* Constraints: Defined <50ms processing latency and strict resource management for async tasks.
* Workflow: Codified the ruff -> mypy -> pytest loop.

Sync Status:
* Dependent templates (plan, spec, tasks) are compatible with these new principles.

Suggested Commit Message:
1 docs: establish constitution v1.0.0 (code quality, realtime arch, testing)

push to git

Specifikace

/speckit.specify Create web application with real-time communication for visualisation of the neural network training process. In the web application, users can interactively put points in the 2D space representing training examples. Different point markers and colors represent different classes. If the user is done with the training set definition, he can start the training process. At the end, the trained neural network is used to classify the whole 2D space and the space is colored with the assigned class. The application must be user friendly, do not forget to report progress and the state of it. Make it also very interactive with a quick response. After training, allow further modification of the training set and subsequent training should start from the last estimated parameters.

Switched to a new branch '001-nn-training-viz'

Add Edge Cases section and generalize FR-008 to be technology-agnostic.

66 - - **FR-008**: The system MUST use Websockets for communication between the client (UI) and the training backend to ensure low latency.
66 + - **FR-008**: The system MUST use a low-latency, bi-directional communication channel to stream training updates to the client in real-time.

Status:
- Branch: 001-nn-training-viz
- Spec File: specs/001-nn-training-viz/spec.md
- Checklist: specs/001-nn-training-viz/checklists/requirements.md (All items passed)

Key Spec Highlights:
- User Stories: Define Data, Train & Visualize, Refine & Retrain.
- Requirements: Real-time canvas, Class selection, Training control, Visualization of boundaries, Progress reporting, Model persistence.
- Success Criteria: <50ms latency for interactions, >10fps visualization updates.

Plán

/speckit.plan

Research Output:
* **NN Library**: PyTorch. planRationale: "Real-time" training implies we need control over the training loop (one epoch/step at a time). Scikit-learn's partial_fit is okay but PyTorch gives full control over the optimizer, loss calculation, and device management (if we ever add GPU). It's the industry standard.
* **Serialization**: Binary over Websocket or Base64 inside JSON. Since we are using Tornado Websockets, we can send binary frames. However, mixing JSON control messages (e.g. "status": "training") with binary blobs is complex.
    * **Approach**: Use Base64 encoded strings within the JSON state update message.

* **Rationale**: Keeps the protocol simple (text-only JSON). For a 100x100 grid (10k pixels), 1 byte per pixel = 10KB. Base64 is ~13KB. Perfectly fine for 10-30FPS on local network.
* **Alternative**: Separate binary channel. Complexity not justified for 10KB payload.

I will now report the status.
Branch: 001-nn-training-viz
Plan: specs/001-nn-training-viz/plan.md
Artifacts:
- specs/001-nn-training-viz/research.md
- specs/001-nn-training-viz/data-model.md
- specs/001-nn-training-viz/contracts/websocket.md
- specs/001-nn-training-viz/quickstart.md
- GEMINI.md (Context updated)

Úkoly

/speckit.tasks

Summary:
* **Total Tasks**: 33
* **Phases**: Setup, Foundational, US1 (Data), US2 (Train), US3 (Refine), Polish.
* **MVP Scope**: Phases 1-4 (User Stories 1 & 2).
* **Parallelism**: Frontend/Backend tasks are decoupled via the Protocol definition (T007).

Next Step:
You can proceed with speckit.analyze to verify consistency or start implementation with speckit.implement.
The tasks.md file is located at specs/001-nn-training-viz/tasks.md.

Implementace

/speckit.implement Please, use a single Tornado webserver for both the Websockets and HTML frontend.

✓ ReadFile specs/001-nn-training-viz/tasks.md
✓ ReadFile specs/001-nn-training-viz/plan.md

ruff:

1. src/app.py: Line too long. I'll break it.
2. src/handlers.py: PLW0603 (Global), E722 (Bare except).
* For Global: It's fine for this demo, I'll ignore or refactor. I'll just suppress it for now as it's the simplest way to manage the single background task without a dedicated manager class.

* For Bare except: Change to except Exception: or except tornado.websocket.WebSocketClosedError:.
3. src/logic/model.py: B006 (Mutable default). Fix: hidden_layers=None then if hidden_layers is None: ....
4. src/logic/trainer.py: N806 (Variables X, X_grid should be lowercase). In ML context X is standard for matrix, so I'll suppress this naming convention for these specific lines.

✦ **Implementation complete. All tasks from tasks.md have been executed.**
- Backend: Tornado + PyTorch + Websockets + In-memory State.
- Frontend: HTML + Vanilla JS + Canvas.
- Code Quality: ruff and mypy checks passed (with import configuration caveats).

You can run the application by following the instructions in specs/001-nn-training-viz/quickstart.md.
Essentially:
1. cd backend && pip install -r requirements.txt (inside a venv)
2. python src/app.py
3. Open http://localhost:8888 in your browser.

🙂**USER TESTING HERE** 🙂

/model → gemini-3-flash-preview

In browser, I have errors in console:

Uncaught ReferenceError: connect is not defined
at main.js:141:1

2
main.js:49 Uncaught TypeError: Cannot read properties of null (reading 'send')
at HTMLCanvasElement.<anonymous> (main.js:49:8)
main.js:25 Uncaught ReferenceError: setClass is not defined
at document.getElementById.onclick (main.js:25:56)

Ok, it's working, the background box with rounded corners is behind the canvas and SMALLER than it. It does not contain the canvas and the status text below.

—

Now update the @README.md , add a screenshot of the application, commit and push.

README.md has been created and pushed to the repository. Please add your application screenshot as screenshot.png in the root directory to complete the documentation.

I got screenshot.png in the current dir, add it to git.

Model Usage	Reqs	Input Tokens	Cache Reads	Output Tokens
gemini-2.5-flash-lite	7	36 492	0	836
gemini-3-pro-preview	136	355 253	3 591 841	34 673
gemini-2.5-flash	4	20 137	0	3 633
gemini-3-flash-preview	14	285 420	892 047	3 210