Powering Cognitive Radio with ~~LLMs~~ **AI**

Paul David
paul.david@simplirf.com

GRCon 2025 - September 11th, 2025

Agenda

- LLMs and how to fine-tune them for GNU Radio
- GNU Radio LLM Project to build an interface for generation and control
- Beyond backpropagation: Hebbian learning and cellular automata

Motivation and Journey

- Started with a simple question:
  How can we use LLMs (and DL in general) to help build and control radio systems?
- They work well as a natural language interface for generating flowgraphs and high level control
- But they fall short for low-level and latency sensitive tasks
- The surprising effectiveness of LLMs and signs of emergent phenomena sparked new ideas
- So the question became: What other models might exist?

Quick overview of Large Language Models (LLMs)

- Typically transformer-based architectures which are deep feedforward neural networks
- Trained on a vast amount of data using backpropagation
- Great for generation and following natural language instructions (they are language models after all)
- Each token forms a query to compare with every other token's key, and uses those matches to weight the values
- Where do they fit best in communications and GNU Radio?

Tuning an LLM with LoRA

- LoRA (Low Rank Adaptation) is a technique for fine-tuning LLMs
- Typical deep learning linear layer:

$$\mathbf{y} = \mathbf{W}_0\mathbf{x} + \mathbf{b}$$

- LoRA modifies this by adding low-rank matrices:

$$\mathbf{y} = \mathbf{W}_0\mathbf{x} + \mathbf{B}\mathbf{A}\mathbf{x}$$

- $\mathbf{W}_0$ is kept frozen and only $\mathbf{B}$ and $\mathbf{A}$ are updated
- Reduces trainable parameters substantially

Tuning an LLM with QLoRA

- QLoRA (Quantized LoRA) extends LoRA by quantizing the base model
- Reduces memory footprint and speeds up training
- Quantize the base model to 4-bit normalized `FP`:

$$\mathbf{W}_0^{(q)} = Q(\mathbf{W}_0)$$

- We do need to keep a higher precision for the low-rank matrices, typically using `FP16`

GNU Radio LLM Project

- A simple LLM interface for generating and controlling flowgraphs
- Provides a pipeline for tuning LLM base models on this task
- Adds several dataset tools and serves as a proof of concept

Natural Language as an Interface to GNU Radio

- We can use prompts to guide the LLM in generating flowgraphs:
  Create a new flowgraph with ID derp.
  Add an analog source block to the flowgraph.
- We can also give the LLM high level instructions on using flowgraphs:
  Set the center frequency to 90 MHz.
  Set the RF gain to 20 dB.

GNU Radio LLM Example

Radio CLI for inference

🔧 Flowgraph loaded.
✔ **Flowgraph successfully built!**

Blocks

ID	Name
variable_qtgui_range	center_freq
variable	samp_rate
analog_sig_source_x	analog_sig_source_x_0
throttle	throttle_0
blocks_null_sink	blocks_null_sink_0

Connections

Source ID	Destination ID
analog_sig_source_x_0	throttle_0

» Connect the throttle to the null sink.

The Hard Part: Acquiring Training Data

- A decent amount of training data is needed to tune an LLM
- How do we get more data for communication flowgraphs and control?
- Hooking GNU Radio Companion! Created a GRC data logger for capturing user interactions

Generating Training Data from GRC

- Two loggers: one tracks flowgraph changes and the other actions on an executed flowgraph during runtime
- A second tool transforms traces into prompt-context-completion tuples

Future Work

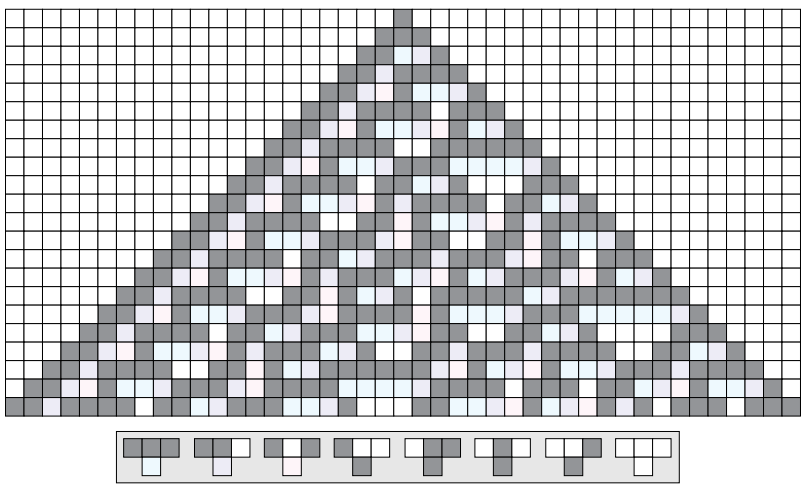
- Expand trace coverage and prompt diversity
- Look at Vision Transformers and multi-modal models
- Investigate other fine-tuning techniques
- GitHub repository:
  `https://github.com/SimpliRF/gnuradio_llm`

Hebbian Cellular Automata

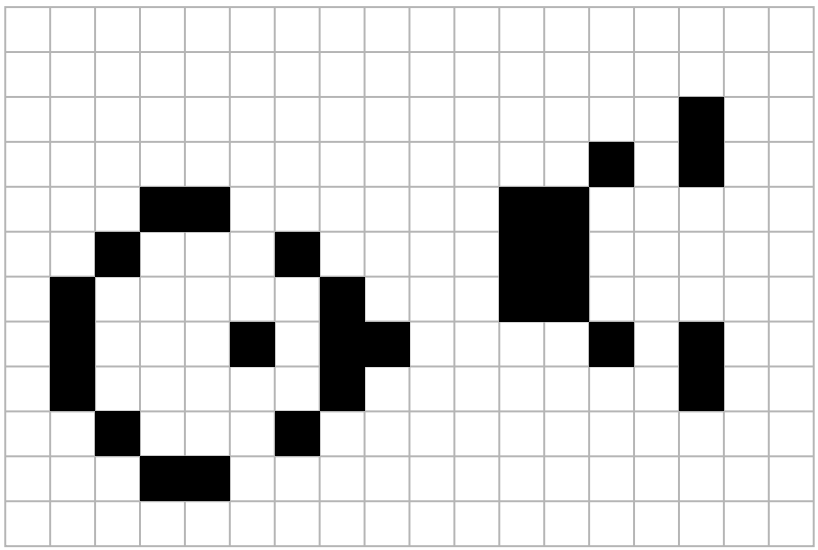
- A novel model inspired by neuroscience, communication theory, and dynamical systems
- Combines principles from Hebbian learning: *Cells that fire together wire together*
- Idea originated from concerns about latency and efficiency
- This is relevant to us if we want to build adaptive communication systems where we may lack training data

Cellular Automata

- Originally discovered at Los Alamos by Stanislaw Ulam and John Von Neumann in the 1940s for modeling self-replicating machines
- Grid of cells, each following local update rules and interacting locally with neighbors



1D Cellular Automata: Rule 30
Stolen from Wikipedia



2D Cellular Automata:
Conway's Game of Life

Modeling a Neural Network in a Cellular Automata

- Each cell contains a state vector $\mathbf{s}_i$
- Each state contains activations, incoming weights, traces, modulation value, and a modulation counter:
  $a_i \in \{-1, +1\}$, $w_{ij} \in \{-1, 0, +1\}$, $M_i \in \{-2, -1, 0\}$, $R_i \in \mathbb{N}$
- A trace on an edge e_{ij} is a raw correlation statistic:

$$C_{ij} = \sum_{n=T-W}^{T-1} a_i[n] \cdot a_j[n]$$

Introducing Feedback and Modulation

- Question: How can we avoid backpropagation? Is it possible?
- One alternative is Hebbian learning, which uses a 3 factor learning rule:

$$\Delta w_{ij} \propto M_i \cdot C_{ij}$$

- Hebbian learning is usually not stable, but with modulation or feedback, it can be made to encode task goals
- Idea to use feedback originated from wireless cellular networks

Modulation Diffusion and Direction

- There was initially a stability issue even with feedback
- Turns out that the gradient/direction of the modulation signal M_i is important for information flow
- A direction indicator is needed:

$$D_{ij} = \begin{cases} +1 & \text{if } R_i \geq R_j \\ -1 & \text{if } R_i < R_j \end{cases}$$

where R_i is the modulation counter

Modulation Diffusion and Direction

Modulation diffusion from source cell

4	3	3	3	3
4	3	2	2	2
4	3	2	1	1
4	3	2	1	S

Weight Update

- The weight update:

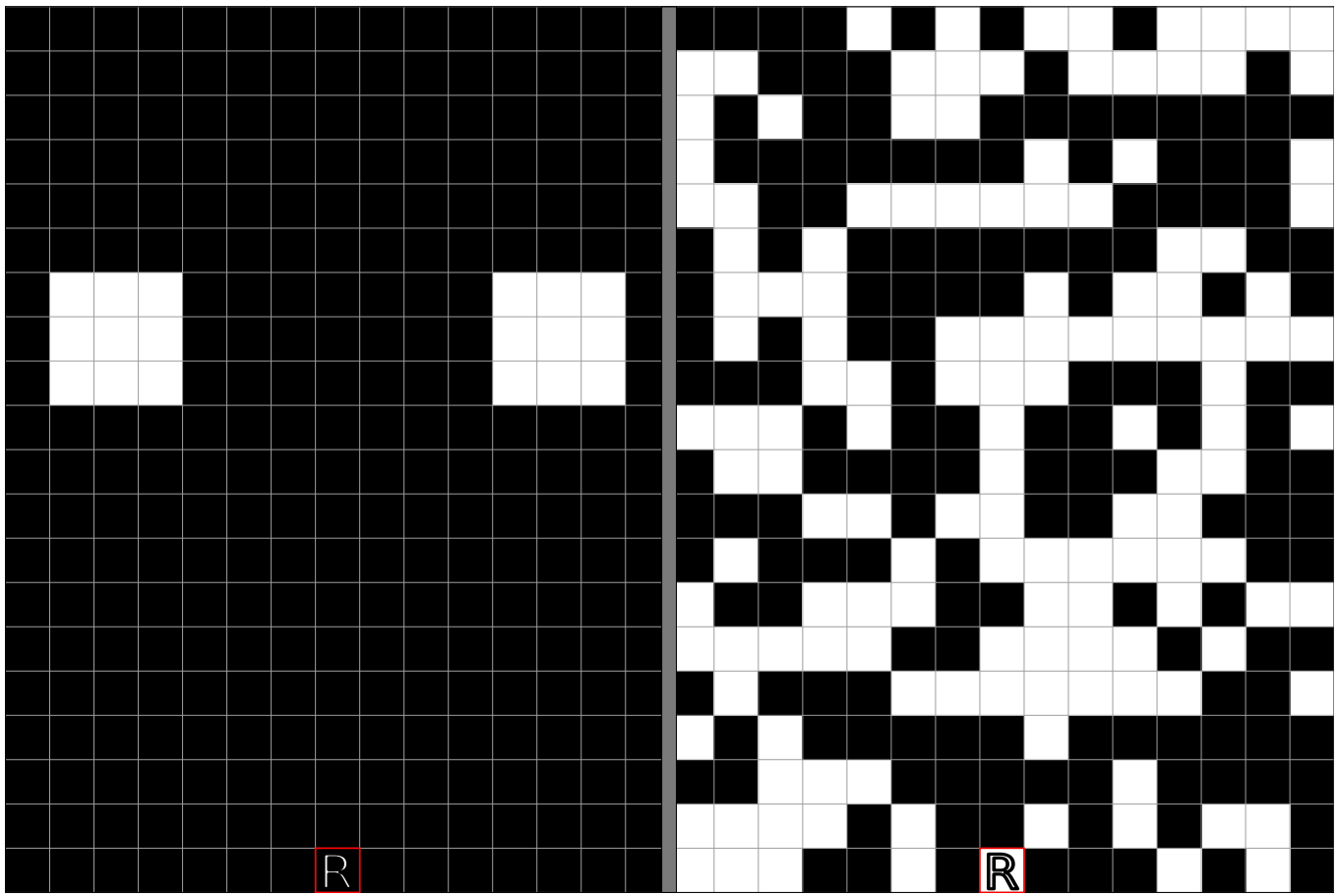
$$\Delta w_{ij} = M_i \cdot f_\theta\left(C_{ij}\right) \cdot D_{ij}$$

- Applied according to:

$$w_{ij} \leftarrow \text{clip}(w_{ij} + \Delta w_{ij}, -1, +1)$$

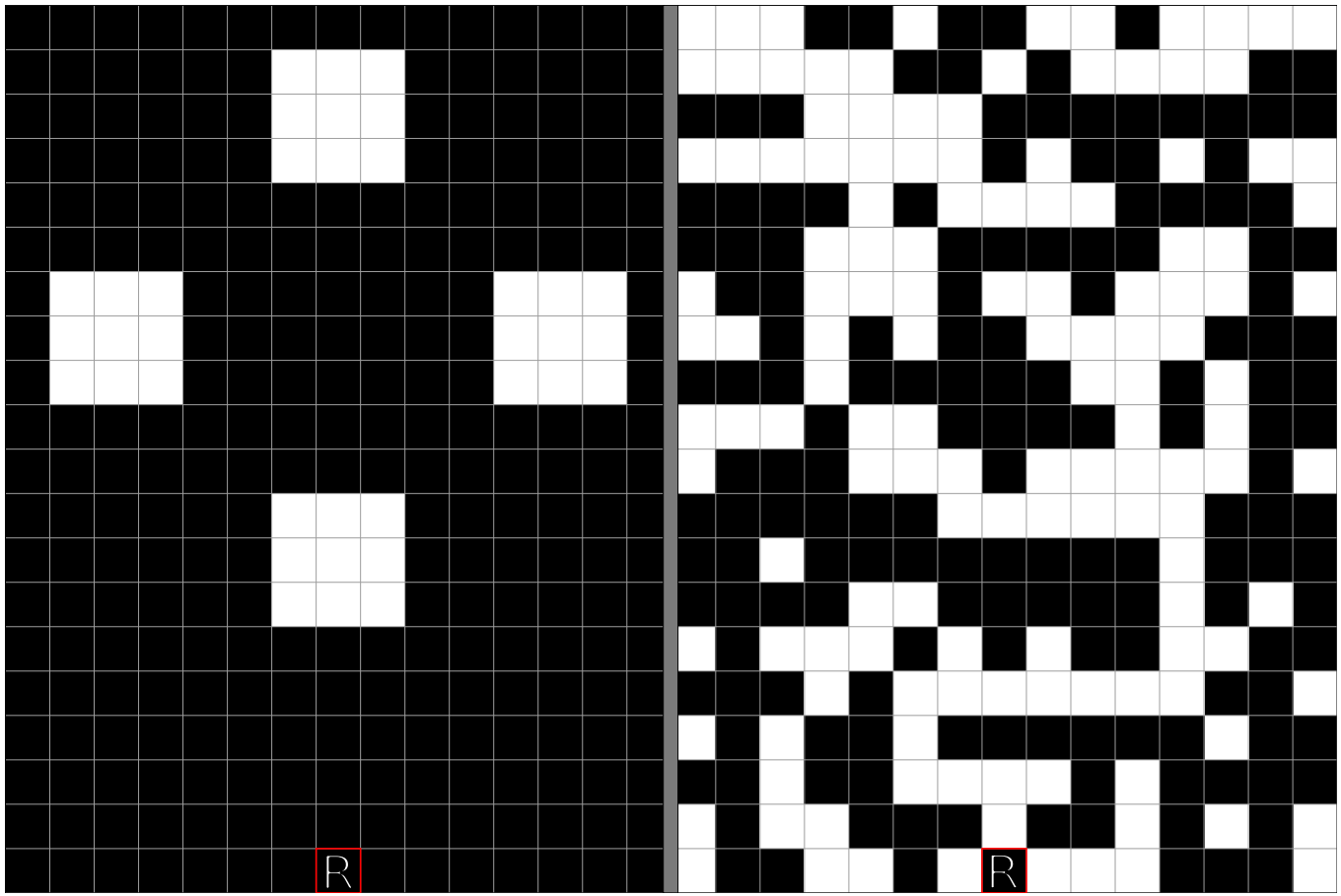
Simple HCA Example for Mod Classification

Clean BPSK constellation



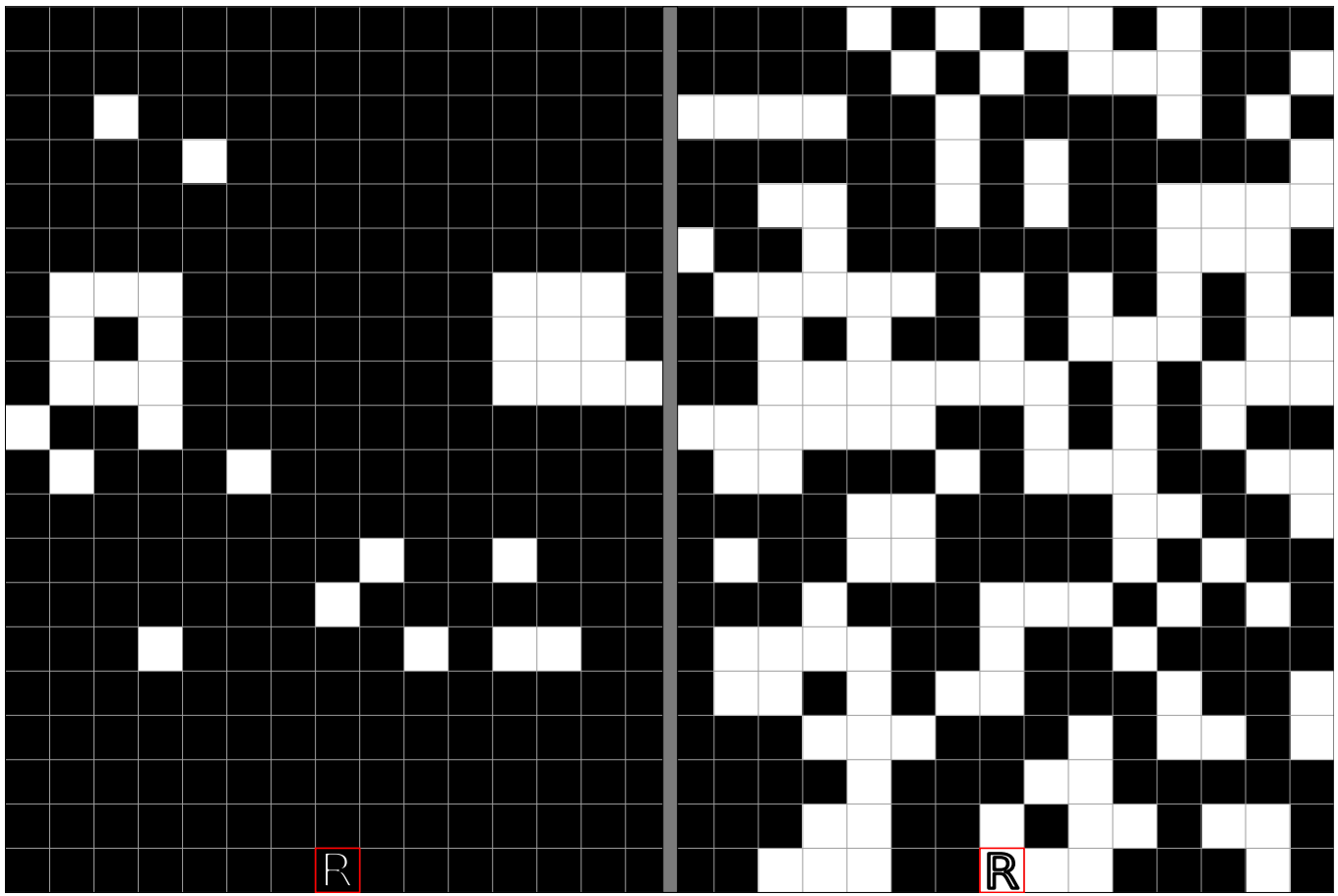
Simple HCA Example for Mod Classification

Clean QPSK constellation



Simple HCA Example for Mod Classification

Noisy BPSK constellation



Connections to Other Energy Models

- HCA is a member of the energy family of models
- Other examples include Hopfield networks and Boltzmann machines
- Unique in that this one includes Hebbian learning, but these others also avoid backpropagation
- This model does not have a global energy function like the others

Preliminary Results

- Still early in development and need to evaluate on benchmarks
- Initial results show extremely sample efficient learning from ideal cases
- Only took 20 or so training iterations at 30 ticks to train a PSK classifier from ideal examples
- Also need to put this into a GNU Radio block or OOT module

Conclusion

- LLMs are very good hammers but not everything is a nail
- The HCA introduced is an example of a novel model
- New approaches to AI are needed for increased efficiency,
- Ideas from communications, biology, and physics can inform efforts
- Questions?