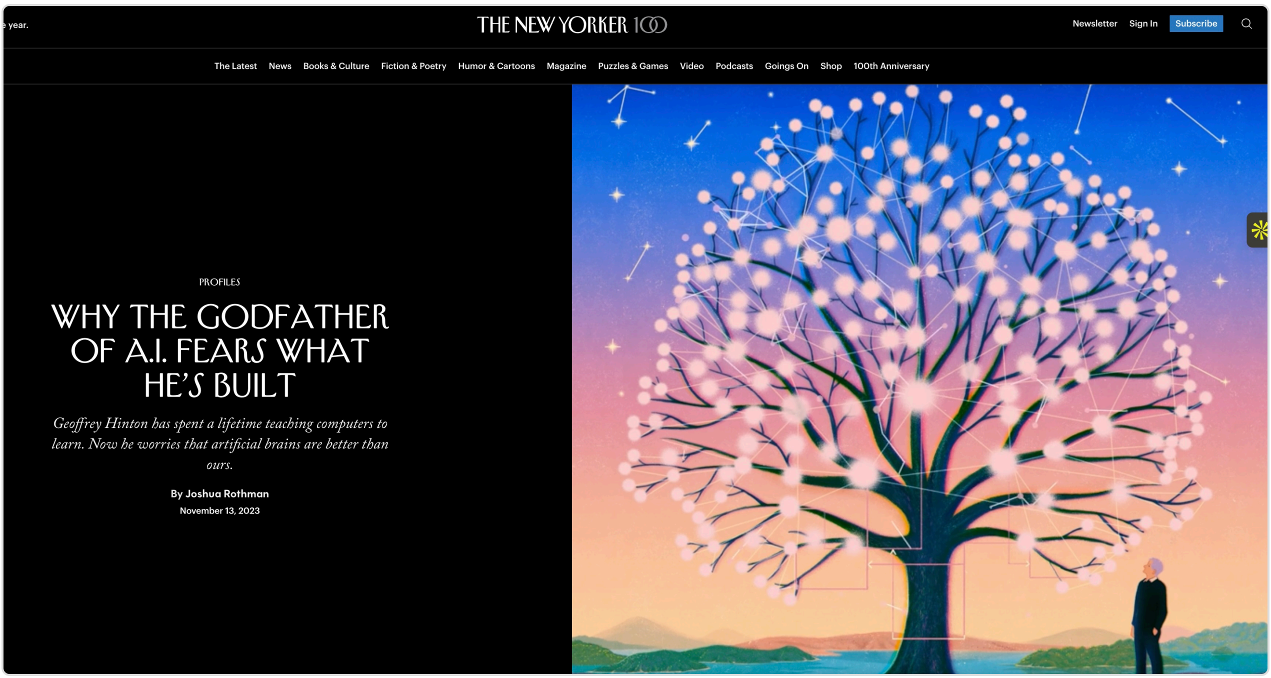
Neural Network Interpretability

Understanding Deep Learning Through Polytopes

Burton Alexander, Charlie Cruz, Michael Khalfin

Motivation

Neural networks are famously known to be **black boxes**



To address this we focused on a case study: simple feedforward neural networks.

Related Work

Interpretability is Complex

> **The Mythos of Model Interpretability** (Lipton '17)
>
> - Many competing definitions for interpretability
> - Just because a model is linear, does not make it interpretable...

Interpreting Optimization Problems

Nevertheless in the literature they have succeeded in interpreting the relationships between variables in LPs and MIPs such as in...

- **The Voice of Optimization** (Bertsimas, Stellato '20)
- **Machines Explaining Linear Programs** (Steinmann et al. '22)

Previous Work on Neural Network Interpretability

Out of scope!

Project Overview

This project demonstrates how we can:

- **Build** a tiny MNIST digit classifier (49 → 3 → 3 → 10 neurons)
- **Encode** network behavior using polytope representations
- **Understand** what each neuron learns by building visualizations
- **Interpret** system-level behavior through optimization

Key Innovation

Whereas it is fairly common to use linear programming to formally verify properties about network behavior, we show it is also a powerful tool for **discovering** properties.

Network Architecture

We use a deliberately small network for our proof of concept:

Architecture: GELU-GELU-Linear

- **Input:** 49 neurons (7×7 downsampled MNIST images)
- **Hidden Layer 1:** 3 neurons with GELU activation
- **Hidden Layer 2:** 3 neurons with GELU activation
- **Output:** 10 neurons (digit classes 0-9)
- **Post-processing:** Softmax for probabilities

Why GELU?

ReLU (Rectified Linear Unit) would have been a simpler choice, as it is piecewise, whereas GELU (Gaussian Error Linear Unit) is a smooth activation function. However, our architecture is activation-function agnostic as we can tightly approximate any function with linear envelopes.

```
GELU(x) = x · Φ(x)

Where Φ(x) is the CDF of the standard normal distribution

Approximation: GELU(x) ≈ 0.5x(1 + tanh(√(2/π) · (x + 0.044715x³)))
```

Mathematical Formulation

Forward Pass

x_0 = input (49-dimensional, 7×7 flattened)

$a_1 = W_1 \cdot x_0 + b_1$ (shape: 3)
$z_1 = \mathrm{GELU}(a_1)$

$a_2 = W_2 \cdot z_1 + b_2$ (shape: 3)
$z_2 = \mathrm{GELU}(a_2)$

$a_3 = W_3 \cdot z_2 + b_3$ (shape: 10, output logits)

Prediction = $\mathrm{argmax}(a_3)$

Affine Transformations

$a_\ell = W_\ell \cdot z_{\ell-1} + b_\ell$

Where:
- W_ℓ is the weight matrix for layer ℓ
- b_ℓ is the bias vector
- $z_{\ell-1}$ is the output from the previous layer

The Polytope Representation

Definition

A **polytope** is a geometric region defined by linear inequalities. For neural network verification, we construct a polytope that *over-approximates* all possible network behaviors for inputs in a given region.

Variables in our polytope:

- $x_0[i]$ for i = 0..48: Input pixels
- $a_1[j], z_1[j]$ for j = 0..2: Pre/post-activation for hidden layer 1
- $a_2[k], z_2[k]$ for k = 0..2: Pre/post-activation for hidden layer 2
- $a_3[m]$ for m = 0..9: Output logits

Constraints in our polytope:

1. **Input box:** $x_0[i] \in [x_0[i] - \varepsilon, x_0[i] + \varepsilon] \cap [0, 1]$ for all i
2. **Affine relations:** $a_\ell = W_\ell \cdot z_{\ell-1} + b_\ell$ (equality constraints)
3. **GELU envelopes:** Linear lower/upper bounds on $z = \mathrm{GELU}(a)$

Encoding GELU with Linear Constraints

The key insight: we can replace the nonlinear GELU activation with tight linear envelopes!

```
For each neuron with pre-activation a and post-activation z:

Lower envelope: z ≥ αₗ · a + βₗ
Upper envelope: z ≤ αᵤ · a + βᵤ

Where αₗ, βₗ, αᵤ, βᵤ are computed using:
• Interval bounds [L, U] for a (from IBP)
• Tight linear envelopes that bound GELU over [L, U]
```

Why This Works

By using linear constraints instead of the actual nonlinear GELU function, we create a polytope that:

- Contains all possible network behaviors in the input region
- Can be analyzed using efficient linear programming solvers
- Provides formal verification guarantees

Interactive Network Visualization

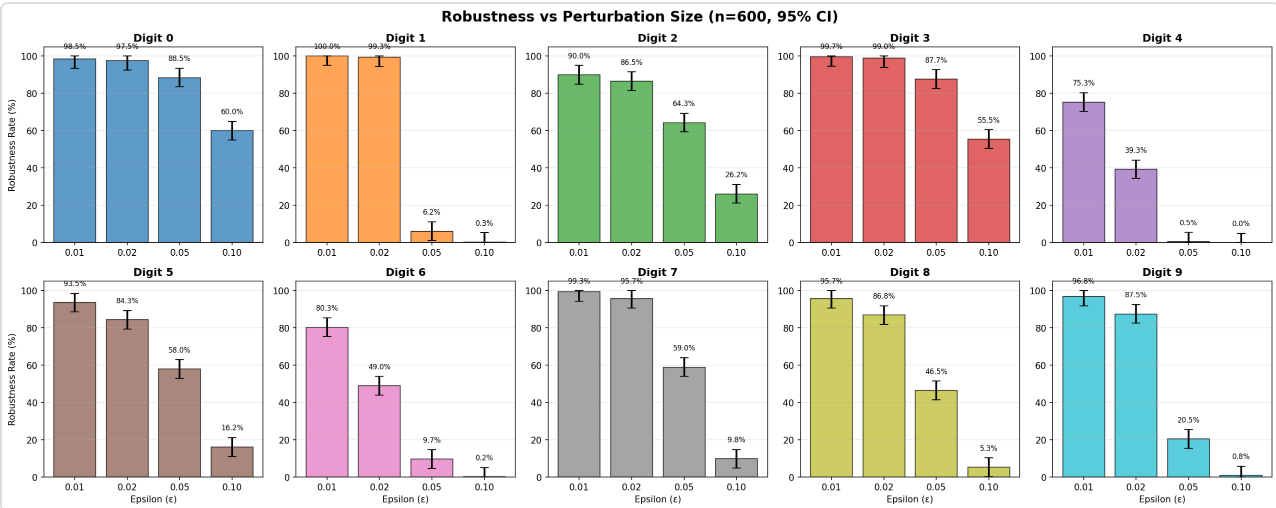
💻 Interactive Demo Available Online

Visit the web version to explore the interactive neural network visualization with:

- Live network activation display
- Hover tooltips showing neuron details
- Clickable neurons to see learned patterns
- Navigate through MNIST samples

Key Finding: Linear Classifier

Using the polytope representation, we can verify how robust the network is to input perturbations:



Key Findings

- The LP maintains high accuracy for small perturbations (ε ≤ 0.02)
- Different digits show varying sensitivity to perturbations
- Digit 1 remains highly robust even at ε = 0.02
- Digit 4 degrades more quickly with larger perturbations
- The LP itself appears to be a good classifier for MNIST

Understanding What Neurons Learn

Each hidden neuron learns interpretable patterns that combine to form digit classifiers.

Example: How the network recognizes Digit 0

Digit 0 ∝ (++ Frame) - (- Spine) - (- Belt)

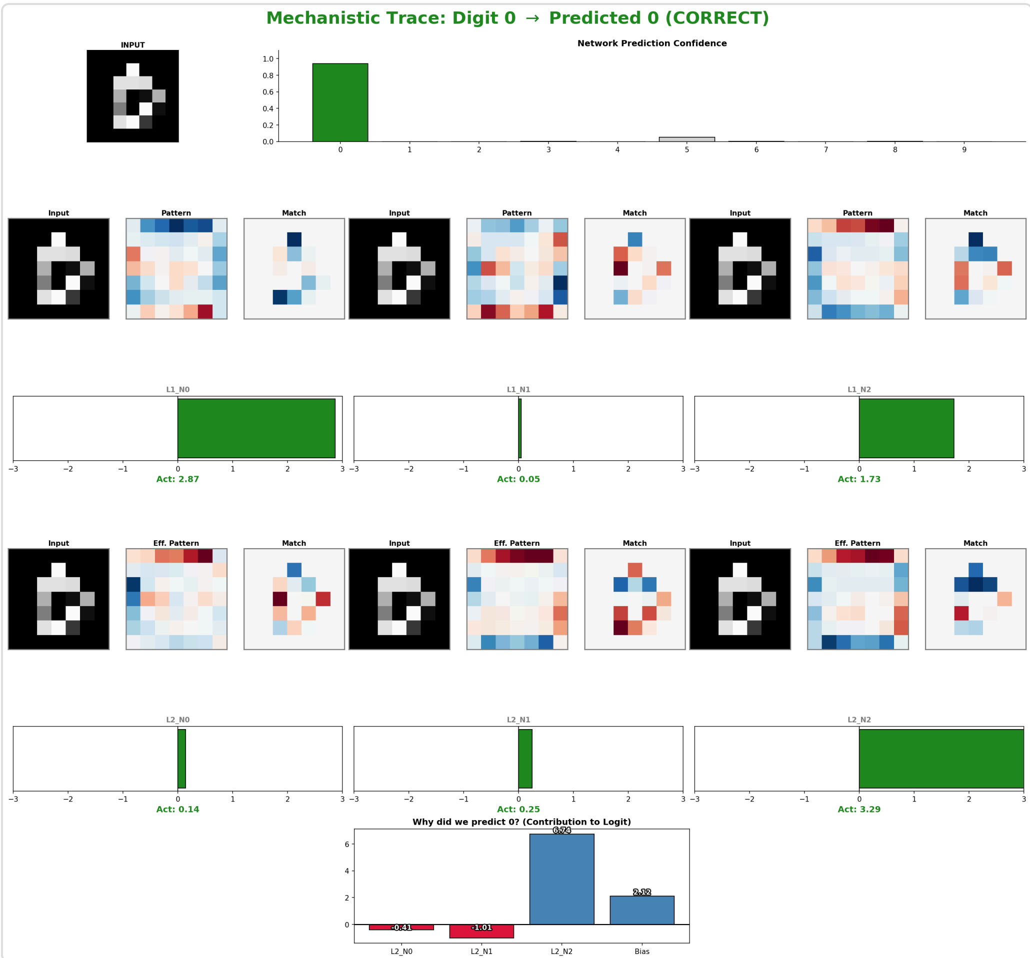
Must act like a container   Must have empty center   Must have empty middle

Layer 1 Neurons:

- **Frame:** Detects outer boundary/container structure
- **Spine:** Detects vertical centerline activation
- **Belt:** Detects horizontal middle activation

The network learns that digit 0 should strongly activate the "Frame" detector while avoiding activation of "Spine" and "Belt" detectors (which would indicate filled regions).

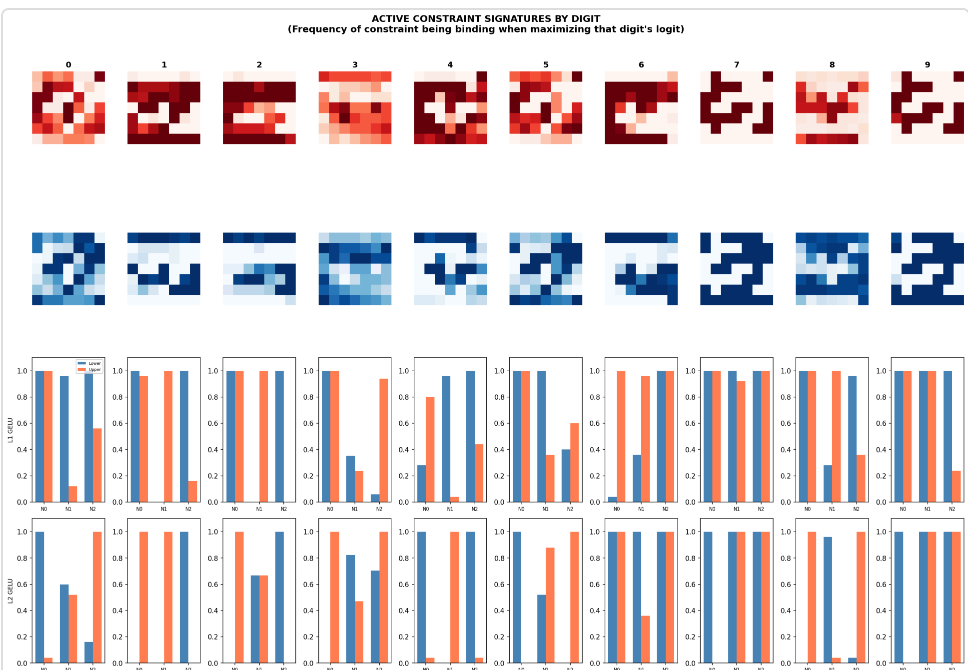
Mechanistic Trace for Digit 0



The dashboard shows how Layer 1 neurons detect basic patterns (Frame, Spine, Belt), and Layer 2 neurons combine them with learned weights to produce the final digit 0 logit.

Constraint Signatures

- For each digit, we take a real image, allow an ε-ball of perturbations, and use an LP to maximize that digit's logit inside the polytope.
- At the optimum we record which inequalities are tight; this gives a "constraint signature" for that digit.
- Red heatmaps: pixels that frequently hit their lower bound (the network prefers less ink there).
- Blue heatmaps: pixels that frequently hit their upper bound (the network prefers more ink there).
- Bar plots: how often each GELU envelope in layers 1 and 2 is active for that digit.
- Experiment: Each digit ends up with a distinctive pattern that we can read as a rule about where the network expects ink or blank space.



System-Level Behavior

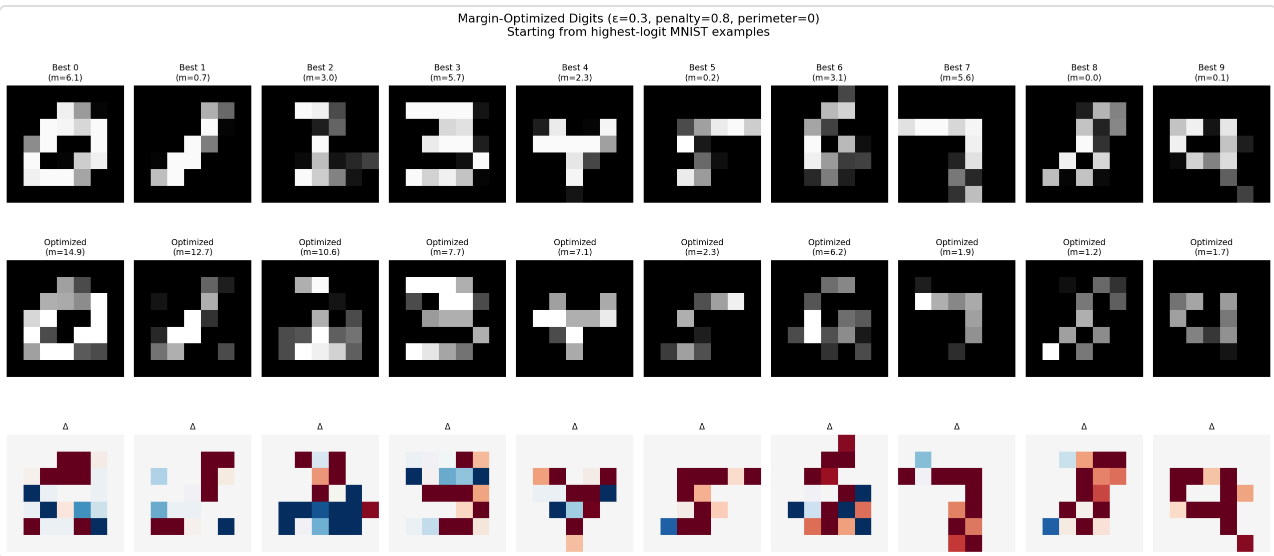
Formulation

$$
\begin{aligned}
\mathit{max} \quad & a_3[\text{true class}] - t \\
\mathit{s.t.} \quad & t \geq a_3[k] \quad \forall\, k \neq \text{true class} \\
& \text{all polytope constraints hold}
\end{aligned}
$$

Seeks to maximize distance between correct logit and largest other logit

Starts from best performing version of each digit and can deviate by ε amount

Uses ℓ_1 regularization



Limitations & Next Steps

Limitations & Speculation

We don't know whether our methods will scale to larger networks!

- Too many hidden neurons, possibly representing multiple concepts
- For larger networks, the system-level analysis may be more useful!
- Increased computational cost for LP solving

Next Steps

- Test other activation functions
- Beyond feedforward: CNNs, RNNs, or Transformer models?
- Neural network training as a way to create linear classification models
- How can we use interpretability results?

Thank You!

Questions?

Key Takeaways

- Polytopes enable formal verification of neural networks
- Small networks can be fully interpretable
- Linear programming provides both verification and insights
- Mechanistic interpretability reveals how features compose