Width is Less Important than Depth in ReLU Networks

Key ideas + one theorem intuition + experiment (5 min)

Ilyas Madah - Moghit Yebari

February 1, 2026

Paper

"Width is Less Important than Depth in ReLU Neural Networks" (Vardi, Yehudai, Shamir, 2022).

Takeaway

Flattening depth into width is expensive, but **trading width for depth is cheap** (under mild assumptions).

Context: why compare width vs depth?

- ReLU nets are universal approximators if width is large enough, even at depth 2.
- But: empirical success suggests **depth** may be the main driver of expressive power.
- Question (Lu et al. 2017 style): are width and depth incomparable, or can depth compensate for width?

This paper's message

For many settings, **width beyond** $O(d)$ **is not essential**: any target ReLU net can be approximated by a **narrower** net (width $\approx O(d)$) with additional depth and only polynomial overhead.

Why depth $\to$ width is *expensive*

Linear-region argument in 1D

A depth-2 ReLU network in 1D with width m has at most $m + 1$ linear pieces.

- Consider an iterated "triangle wave" function $g^{\circ L}$ (Telgarsky-type construction).
- It has 2^L linear regions on $[0, 1]$.
- Therefore any depth-2 exact representation needs:

$$m \geq 2^L - 1 \qquad \text{(exponential in depth).}$$

Interpretation

If you cap depth (say $L = 2$), matching deep functions can require **exponentially large width**.

Why width $\to$ depth is *cheap*: the paper's construction

Core trick: keep nonnegative "state"

Coordinate-wise: $x = \sigma(x) - \sigma(-x)$, so we store $\sigma(x)$ and $\sigma(-x)$.

- Goal: simulate a wide, shallow net

$$f(x) = \sum_{i=1}^{n} u_i\, \sigma(\langle w_i, x\rangle + b_i) + b_{\text{out}}.$$

- Build a **narrow deep** net of width $2d + 3$ that:
  1. computes each hidden unit sequentially into a scratch coordinate,
  2. accumulates contributions into two nonnegative sums S^+, S^-,
  3. outputs $S^+ - S^- + b_{\text{out}}$.

Cost

Width becomes constant $(2d + 3)$ and depth grows linearly $(\approx 2n + 2)$. This is the "cheap" direction.

Experiment (report result): overfitting + conversion check

Setup (binary classification)

`make_moons`, $N = 600$; train=32 points; label noise $p_{\text{flip}} = 0.4$ on train only.

Model	Depth	Width	Train acc	Test acc
Shallow (trained)	2	32	0.9688	0.6021
Deep (converted)	66	7	0.9688	0.6021
Deep (scratch)	66	7	1.0000	0.4806

- Conversion correctness: max output diff $\approx 10^{-13}$ (numerically identical).
- The converted deep net **inherits the same overfitting** (same function).
- Training the same deep architecture from scratch can overfit **even more** (optimization/inductive bias differ).

Final takeaway

Depth provides expressive efficiency; width can be traded for depth at moderate cost, but not vice versa.