nGPT scaling: flat across the width × depth grid

> Width × depth sweep of our simplified nGPT to check that it stays well-behaved as it grows. Converged loss improves with width and is flat across depth, and no condition spikes or stalls. The architecture seems safe to build the color-mixing experiments on.

Before we build the color-mixing experiments on top of this transformer, we want to know that it holds its shape as it grows. This report trains the model at a range of sizes and checks that none of them misbehave.

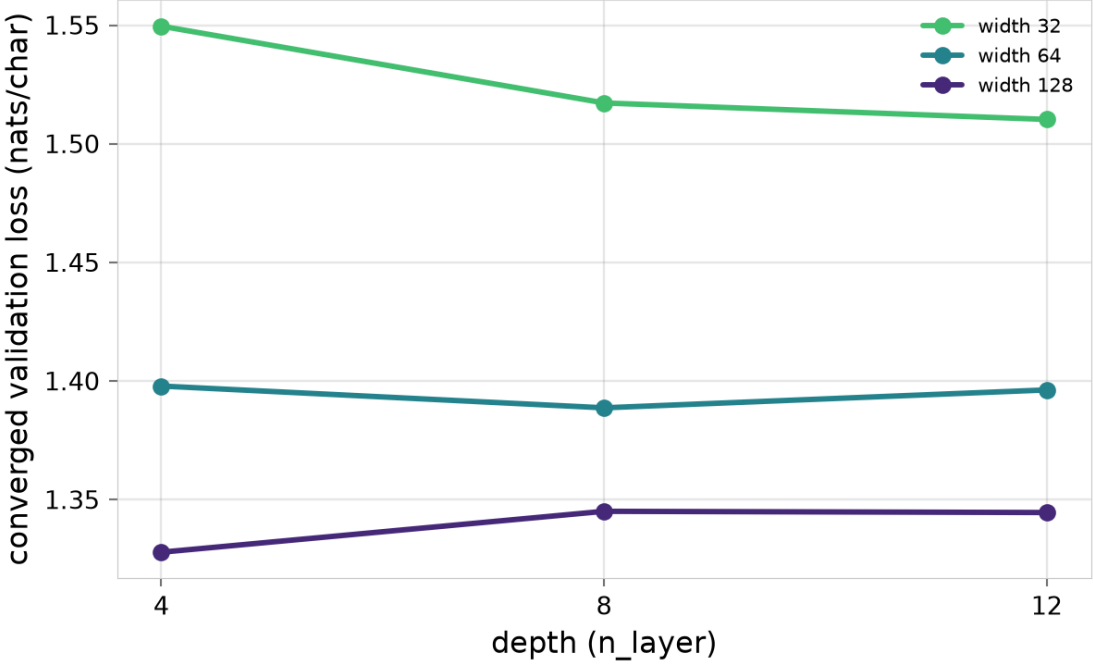
The model is a simplified version of *nGPT*. The idea behind nGPT is to keep the running state of the model (the *residual stream*, the vector that each layer reads from and writes back to) on the surface of a hypersphere, by normalizing it after every step. We keep that residual update, `h ← Norm(h + α·(Norm(sub(h)) − h))`, which moves the state `h` a fraction `α` of the way toward the normalized output of a sub-module and then renormalizes. We simplify two pieces of it. Each sub-module gets a single learned *gain* (one number that scales its output) in place of the nGPT per-channel *eigen learning rates*, and the residual step `α` is fixed at 1/n_layer instead of being learned, which is about where the learned version settled anyway.

For the claim that SCA (the concept-anchoring method this project studies) carries over to language models, this pared-down architecture needs to stay well-behaved as it scales. So the experiment trains it across a grid: three widths (how many numbers are in that state vector) crossed with three depths (how many layers), {32, 64, 128} × {4, 8, 12}, with everything else held fixed: batch size 16, peak learning rate 10^{-2}, 100 epochs, and *Pride and Prejudice* for the training text. Two outcomes would concern us. One is a **depth penalty**, where adding layers at a fixed width makes the model worse. The other is an instability that shows up only at large width, where a run spikes or fails to train. We hope to see neither.

The architecture scales cleanly. We score each run by its converged loss: the average model error at predicting the next character once training has settled, measured in *nats per character* (natural-log units, where lower is better). That loss never rises as we add layers. At each width, the three depths land within 0.04 nats/char of one another, well inside the ±0.08 of noise we see between epochs, so the depth axis is flat. Width behaves the way added capacity should: loss falls monotonically from 1.55 at width 32 to 1.34 at width 128 with 12 layers. No condition spikes or stalls, and the same learning rate (10^{-2}) works everywhere. So fixing the residual step to a scalar constant seems to be enough.

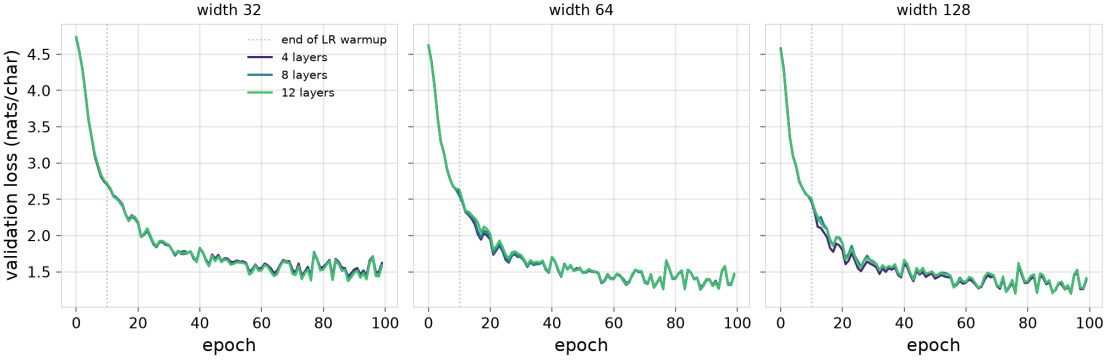
Converged loss versus depth

This chart shows converged validation loss (measured on held-out text and averaged over the last 10 epochs) against depth, with one line per width. Read each line from left to right: if adding layers hurt, the line would slope up. Instead each one is nearly horizontal, so extra depth costs nothing at any width. The lines also stack in width order, so a wider model is uniformly better, and there is no far corner, wide and deep together, where the loss turns back up.



Training curves

The same runs, now shown across training: one panel per width, one line per depth, loss on the vertical axis and epoch on the horizontal. Every run descends through the learning-rate warmup (the opening stretch of training, where the step size ramps up from small) and settles onto a plateau. Within a panel, the depth lines sit on top of one another rather than fanning apart, and the wider panels settle lower.



Findings

Across the whole grid, the simplified nGPT trains flat across depth and keeps improving with width, and no condition destabilizes (1.34 nats/char at the deepest, widest corner). That is what we were hoping for. The architecture that SCA will anchor concepts in scales without a depth penalty, so if a later experiment runs into trouble, the simplified architecture is unlikely to be the reason.

The grid tops out at width 128 with 12 layers on an L4, which is probably fine for M2 (this milestone). For M3 (a future milestone), we should confirm that the fixed scalar residual step still holds at a genuinely larger size: wider and deeper, on a bigger GPU with a bigger batch.