

Normalization or Activation? What to Tune as a Network Grows (Residual CNNs, CIFAR-10)

Which parameter is most powerful to tweak, and why does mixing activations or normalizations sometimes help?

VERSION 2.0 · EXPLORATORY RESEARCH REPORT · 10 OCTOBER 2026

Question: as a residual CNN gets deeper, how much does tuning its activation functions or its normalization layers help, are the two equally useful, and when is a mixture of different choices across layers better than one choice everywhere?

Abstract

We grow tiny residual CNNs from 2 to 16 blocks (16 channels each) and fit a fixed set of 10,000 CIFAR-10 images downsampled to 16 x 16. Each block chooses one of four activations (LeakyReLU, GELU, Tanh, ELU) and one of three normalizations (none, LayerNorm, BatchNorm); kernel size is fixed, so every configuration has nearly the same parameter count. Starting from a LeakyReLU + LayerNorm default, we run six searches: normalization only, activation only, and both, each applied uniformly (one choice for every block) or mixed (chosen per block), with up to 24 trials. Searches select on single-seed training loss and the winners are re-trained with fresh seeds. We repeat each search 4 times and record 771 search fits and 312 fresh-seed fits. Four results stand out. (1) Which knob matters most depends on depth. Among the twelve uniform recipes, normalization explained 65% of the variance in training loss at 2 blocks but only 17% at 16 blocks, while activation went from 15% to 59%. In the full searches, normalization-only tuning led slightly at 2-4 blocks and activation-only tuning led at 8-16 blocks (at 16 blocks 18.8% versus 13.6% lower training loss than the default). (2) The default (LeakyReLU + LayerNorm) did not exploit depth at all (training loss 1.25 at 2 blocks, 1.24 at 16) while the tuned networks did (1.15 falling to 0.86), so the benefit of tuning grew from 8% at 2 blocks to 31% at 16 blocks, and tuning both knobs together beat either alone. (3) A per-layer mixture beat the best uniform recipe only when both knobs were searched together and only in deep networks (+2.4 and +5.5 percentage points at 8 and 16 blocks, better in 3 of 4 and 4 of 4 repetitions); random mixtures within a single knob did worse than a uniform choice. (4) Mixing helps only when it is placed correctly. In deliberate two-recipe mixtures, swapping the order of the same two recipes changed the loss by 7-10% on average, with a consistent preference for no normalization or LayerNorm early and BatchNorm late (for example ELU without normalization followed by GELU + BatchNorm was 17% better than the reverse at 16 blocks, in 4 of 4 repetitions). Well-placed mixtures beat the better of their two ingredients by 3-6%, but only 29-50% of deliberate mixtures beat it at all, so mixing is not a free benefit and position is the main reason it sometimes helps. These findings describe a small, fixed-width experiment on one dataset and do not establish a universal rule.

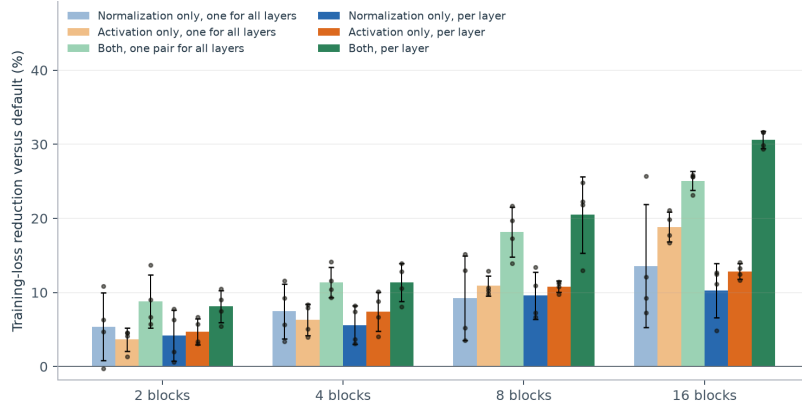
What we found

- The most powerful knob changes with depth. Of the variance in training loss among the 12 uniform recipes, normalization explained 65%, 33%, 21% and 17% at 2, 4, 8 and 16 blocks, and activation 15%, 31%, 44% and 59%. The loss range between the best and worst choice was 0.13 (normalization) versus 0.07 (activation) at 2 blocks, and 0.12 versus 0.24 at 16 blocks.
- In full searches, tuning only the normalization reduced fresh-seed training loss by 5.4%, 7.5%, 9.2% and 13.6% at 2, 4, 8 and 16 blocks, and tuning only the activation by 3.6%, 6.3%, 10.9% and 18.8% (one choice for all blocks; 3 and 4 candidates). The ordering at 2-4 blocks is within the noise of four repetitions; the activation advantage at 16 blocks (18.8% versus 13.6%) is the clearer result, and activation-only results were also much more reproducible across repetitions (SD 1.4-2.1 points versus 3.7-8.3).

- Tuning both knobs together was best: 8.8%, 11.4%, 18.1% and 25.1% for the 12 uniform recipes, and 8.1%, 11.4%, 20.5% and 30.6% for a per-layer search with up to 24 trials. The two effects are sub-additive (13.6% + 18.8% would be 32% at 16 blocks, versus 25.1% observed).
- The default does not use depth; tuned networks do. The default's fresh-seed training loss was 1.25, 1.22, 1.23 and 1.24 at 2, 4, 8 and 16 blocks, while the best per-layer recipe reached 1.15, 1.08, 0.98 and 0.86. The default activation (LeakyReLU) was the weakest activation at depth: switching only the activation to Tanh, ELU or GELU (with LayerNorm, BatchNorm or none) gave 9-26% lower loss at 16 blocks (18-26% in all but the LayerNorm variants of GELU and ELU), while LeakyReLU gained at most 12% and only with BatchNorm.
- Normalization is partly a substitute for what the activation lacks. At 16 blocks LeakyReLU needed BatchNorm (+12% with BatchNorm versus -2% with none), but Tanh gained 22-25% whatever the normalization, and ELU was best with no normalization at all (+26% with none, +18% with BatchNorm, +11% with LayerNorm). The activation-by-normalization interaction accounted for 10-19% of the variance. In trained 16-block networks, LeakyReLU and GELU without normalization had 86-89% negative pre-activations and weak early blocks, whereas ELU and Tanh did not.
- Mixing across layers helped only in one situation: when both knobs were searched together and the network was deep. The joint per-layer search beat the best uniform recipe by 2.4 and 5.5 percentage points at 8 and 16 blocks (fresh seeds; better in 3 of 4 and 4 of 4 repetitions) and also at equal trial counts (28.2% versus 25.4% with 8 trials at 16 blocks), and tied it at 2-4 blocks. Random mixtures within one knob did worse than the uniform choice (for example 12.9% versus 18.8% for activations at 16 blocks).
- Position matters, in a consistent pattern. The regression model shows that at 8-16 blocks bounded or smooth activations help most in the early blocks (loss change of -0.06 to -0.12 with Tanh, GELU or ELU in the early third, 0 to -0.04 in the late third), no normalization in the early blocks is better than LayerNorm (-0.11 at 16 blocks), and BatchNorm helps most in the late blocks at 16 blocks (-0.07) and in the middle blocks at 8 blocks (-0.07). A position-aware model predicted held-out loss only slightly better than a position-agnostic one, so this is a tendency and not a strong effect.
- Normalization shows diminishing returns. Among configurations that vary only the normalization, at 16 blocks going from no BatchNorm to BatchNorm in up to a third of the blocks lowered the loss by 0.12, and the remaining step to BatchNorm everywhere by only a further 0.04. Controlling for composition, each additional distinct normalization type was associated with a loss 0.02-0.04 lower (bootstrap intervals excluding zero at all depths).
- Gains in fit carried over to held-out accuracy in this underfit regime: test accuracy of the best per-layer search rose by 1.8, 2.1, 4.2 and 5.7 percentage points above the default (about 52%) at 2, 4, 8 and 16 blocks, whereas the default did not improve with depth.
- Placement decides whether a mixture helps. In a deliberate test, we built mixtures of the four best uniform recipes at each depth, with recipe A in the first half of the blocks and B in the second half, or alternating. Swapping the order of the same two recipes changed the training loss by 6.8%, 7.3%, 10.4% and 9.8% on average (mean absolute difference) at 2, 4, 8 and 16 blocks, more than the typical gain from mixing. The direction was consistent: putting the BatchNorm recipe in the second half was better, for example ELU without normalization then GELU + BatchNorm beat the reverse order by 16.6% (4 of 4 repetitions) and Tanh + LayerNorm then Tanh + BatchNorm beat its reverse by 13.3% (4 of 4). Mixtures beat the better of their two ingredients in only 29-50% of cases (median -4.3% to +0.1%) but beat their average in 46-75%, so a mixture usually lands between its ingredients unless it is placed well; the best-placed mixtures were 3-6% better than the better ingredient.

Normalization versus activation: which parameter is most powerful to tweak?

Which knob helps most? Full search of each arm



Per-layer search over trials (16 blocks)

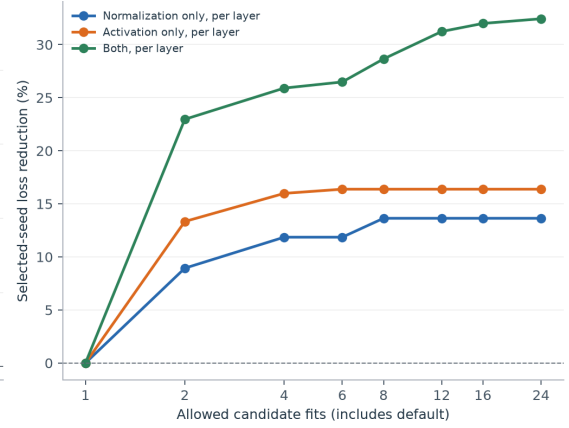


Figure 1. Which knob is most powerful? Left: fresh-seed reduction in training cross-entropy versus the default, for each search arm run to completion, at each depth (bars: mean over repetitions; dots: individual repetitions). Blue arms vary only normalization, orange only activation, green both; light shades are uniform (one choice for every block), dark shades are mixed per block. Right: single-seed selected loss reduction as the number of allowed trials grows at the deepest network; this single-seed view is optimistic because selection noise is included.

1. The experiment

Growth means separately training residual CNNs with 2, 4, 8 and 16 blocks from scratch. A fixed stem convolution (LeakyReLU, max-pool) feeds D residual blocks. Each block computes $x + \text{conv}_{3 \times 3}(x) \rightarrow \text{normalization} \rightarrow \text{activation}$, scaled by $1/\sqrt{D}$, and a second max-pool follows the first half of the blocks. A pooled linear classifier predicts ten classes. Residual connections keep deep plain stacks trainable, so any change with depth is not simply a failure to optimize a plain stack. The objective is to fit a fixed 10,000-image training set, as in version 1; held-out accuracy is a secondary diagnostic. A first MLP study and a CNN study without residual connections (v1) saw depth stall or hurt; they are described in the limitations.

Quantity	Executed setting
Data	CIFAR-10, 2×2 average pooling to 16×16 ; fixed 10,000 training images (1,000 per class); official 10,000 test images used only for the secondary accuracy
Depths	2, 4, 8, 16 residual blocks; 16 channels; 5,802 to 38,730 parameters (all configurations of a depth within 1% of each other)
Choices per block	Activation LeakyReLU / GELU / Tanh / ELU $\times$ normalization none / LayerNorm / BatchNorm = 12; kernel size fixed at 3
Default	LeakyReLU + LayerNorm in every block
Training	Adam, learning rate 0.003, batch 128, 10 epochs, final-epoch selection on training cross-entropy
Replication	4 independent search repetitions; 2 fresh training seeds per selected configuration
Compute	6 CPU workers, one PyTorch thread each; per-fit process CPU time recorded

The six search arms

Arm	What varies	Candidates
Normalization only, uniform	One normalization for every block; activation fixed to LeakyReLU	3
Activation only, uniform	One activation for every block; normalization fixed to LayerNorm	4
Both, uniform	One (activation, normalization) pair for every block	12
Normalization only, per layer	Normalization chosen per block (at least two distinct); activation LeakyReLU	up to 8
Activation only, per layer	Activation chosen per block (at least two distinct); normalization LayerNorm	up to 8
Both, per layer	Pair chosen per block (at least two distinct)	up to 24

Every arm starts with the default and larger budgets are prefixes of the same pre-generated, randomly ordered sequence. The six arms share fits wherever they overlap and each is charged the cost it would incur alone. A single training seed per repetition is shared by all candidates in that repetition, and the selected configuration is re-trained with fresh seeds before it is reported. Test data are never touched during search. The uniform arms are small exhaustive searches; the mixed arms sample uniformly from large spaces, so the comparison is between realistic search designs rather than a pure intervention on dimensionality.

2. Hypotheses and what we found

Hypothesis	Evidence sought	Version 2.0 assessment
H1. Normalization is the more powerful knob to tune.	Normalization-only tuning gains more than activation-only tuning, and explains more loss variance.	Only for shallow networks. Normalization explained 65% of the variance at 2 blocks but 17% at 16 blocks; activation explained 15% and 59%. In full searches normalization-only led slightly at 2-4 blocks (not clearly beyond noise) and activation-only led at 8-16 blocks.
H2. Tuning both knobs beats tuning either alone.	Joint search gains more than the better of the single-knob searches.	Supported. At 16 blocks 25.1% (uniform) and 30.6% (per layer) versus 18.8% (activation) and 13.6% (normalization); the gains are sub-additive.
H3. The benefit of tuning grows with depth.	The improvement over the default grows with depth.	Supported. Because the default does not exploit depth, the best-versus-default reduction grew from 8% at 2 blocks to 31% at 16 blocks.
H4. Per-layer choices beat one choice for all layers.	Mixed searches beat uniform searches at equal or larger budgets.	Only when both knobs are searched jointly and the network is deep (8-16 blocks). Within one knob, random mixtures did worse than the uniform choice, and at 2-4 blocks the two joint searches tied.

3. Normalization or activation?

Blocks	Default loss	Norm only (uniform)	Act only (uniform)	Both (uniform)	Norm only (per layer)	Act only (per layer)	Both (per layer)
2	1.247	+5.4 ± 4.6	+3.6 ± 1.6	+8.8 ± 3.6	+4.2 ± 3.4	+4.7 ± 1.7	+8.1 ± 2.2
4	1.218	+7.5 ± 3.7	+6.3 ± 2.1	+11.4 ± 2.1	+5.6 ± 2.6	+7.4 ± 2.6	+11.4 ± 2.6
8	1.230	+9.2 ± 5.7	+10.9 ± 1.4	+18.1 ± 3.4	+9.6 ± 3.2	+10.8 ± 0.7	+20.5 ± 5.2
16	1.240	+13.6 ± 8.3	+18.8 ± 2.0	+25.1 ± 1.3	+10.3 ± 3.7	+12.9 ± 1.1	+30.6 ± 1.2

Cells show the percentage reduction in fresh-seed training cross-entropy relative to the same-depth default (mean ± descriptive SD across search repetitions). Default loss is the fresh-seed training cross-entropy of the default.

Figure 1 and the table show the central result. At 2 blocks the normalization is the lever (a uniform BatchNorm network was 8-11% below the default loss for every activation, and the variance split puts 65% of the differences on normalization). By 16 blocks the activation is the lever (Tanh, ELU or GELU instead of LeakyReLU gains 18-26%, and 59% of the variance is the activation). The crossover lies between 4 and 8 blocks, where the two shares are about equal.

Our diagnostics of trained 16-block networks (section 4) suggest a centering explanation. Without normalization, LeakyReLU and GELU spend most of their time with negative pre-activations, where they pass little signal (89% and 86% of pre-activations were negative in the early blocks), and the early blocks add little to the residual stream. Normalization re-centers the pre-activations (52-58% negative) and restores the blocks' contribution. ELU and Tanh stay informative for negative inputs, so without normalization their early blocks contribute as much as the normalized ones. We first suspected that the activation matters because unbounded activations let the signal grow, but the measurements did not support that: the stream at the last block was smaller without normalization than with it for LeakyReLU (2.4 versus 2.9-3.1).

The default recipe is therefore a poor choice at depth in this architecture family. That makes tuning gains look large (25-31% at 16 blocks); readers should treat the percentages as relative to this default, and compare absolute losses (Figure 11) when judging the practical size of the effect.

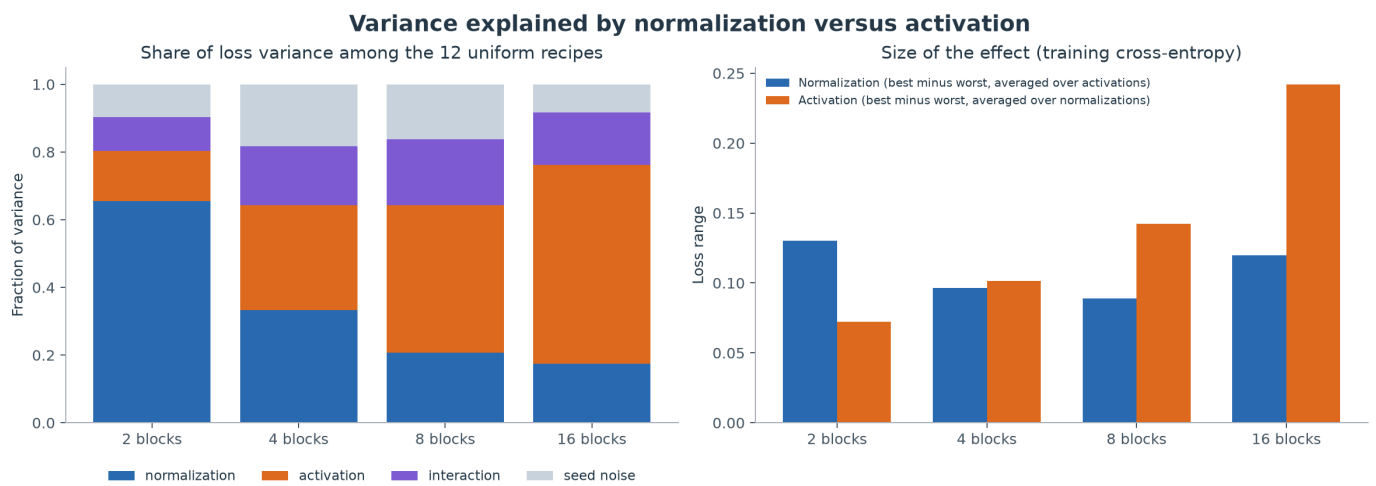


Figure 2. How much of the loss differences among the twelve uniform recipes is due to normalization, activation, their interaction, or seed noise (two-way decomposition, repetitions as replicates; the shared-seed repetition effect is removed first). Right: the size of the effect, as the best-minus-worst average loss across normalizations or activations.

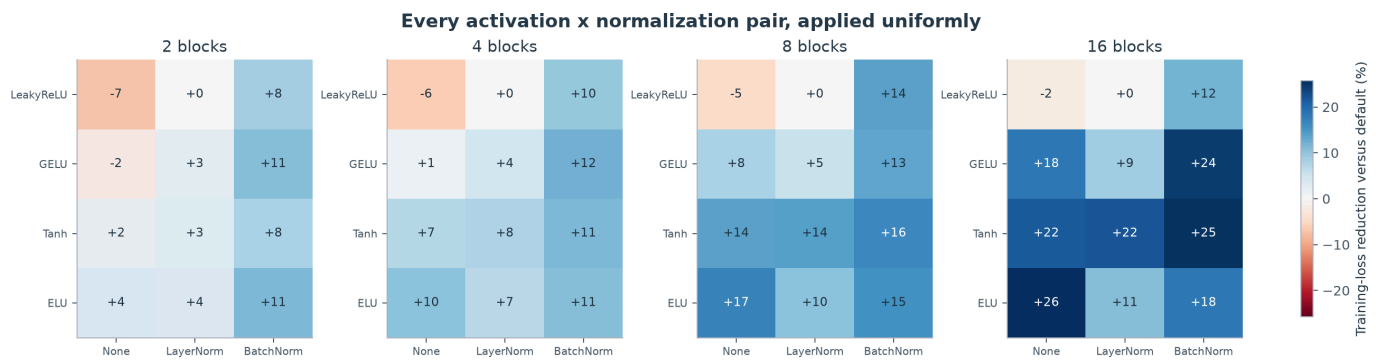


Figure 3. Every activation × normalization pair applied uniformly, as percentage reduction in single-seed training loss relative to the default (blue = better, red = worse), averaged over repetitions.

4. Why would a mixture of choices help?

We test five candidate explanations. They are not mutually exclusive and the experiment can support or contradict a theory but cannot prove it.

Theory	Prediction	What the data say
T1. Complementarity. Different activations or normalizations do different jobs, so mixing them gives a network the benefits of both.	Mixtures beat their own best ingredient often, and more distinct choices help after accounting for composition.	Partly. Deliberate two-recipe mixtures beat the average of their two ingredients in 56-75% of cases but beat the better ingredient in only 29-50%, with a median gain against the better ingredient of -0.6% to -4.3% for split mixtures. Random per-layer mixtures beat their best ingredient in 22-45%. Only well-placed mixtures beat the better ingredient, by 3-6% at best. Complementarity exists but is not generic.

Theory	Prediction	What the data say
T2. Position-specific optimum. Early and late blocks want different things, so placing the right recipe in the right place beats any uniform choice.	Position effects differ between early and late blocks; swapping the order of two recipes changes the loss.	Supported, and the strongest evidence in this study. Swapping the order of two recipes changed the loss by 6.8-10.4% on average (mean absolute difference), more than the typical gain from mixing, and in a consistent direction: the BatchNorm recipe belongs in the second half. At 16 blocks ELU + none then GELU + BatchNorm beat the reverse by 16.6% (4 of 4 repetitions), Tanh + LayerNorm then Tanh + BatchNorm beat the reverse by 13.3% (4 of 4), and Tanh + LayerNorm then GELU + BatchNorm beat the reverse by 10.7% (4 of 4). The position-group regression showed the same pattern (BatchNorm in the late blocks -0.07, no normalization in the early blocks -0.11 at 16 blocks).
T3. Diminishing returns. A few normalization layers capture most of the benefit, so a mixture can drop normalization in some blocks at little cost and spend them elsewhere.	Loss versus the fraction of blocks with a normalization is concave; more distinct normalizations help beyond composition.	Supported for normalization. At 16 blocks adding BatchNorm to up to a third of the blocks lowered the loss by 0.12, but the remaining blocks only by a further 0.04. The diversity coefficient for normalization was negative (lower loss) at all depths (-0.017 to -0.038 per extra distinct normalization). Activations showed a weaker effect (-0.008 to -0.009).
T4. Artifact of search. Mixtures only win because the mixed arm tries more candidates or because selection picks lucky seeds.	The advantage vanishes at equal trial counts and after retraining with fresh seeds.	Not the whole story. The advantage survived fresh-seed retraining and was present at equal trial counts (at 16 blocks 28.2% versus 25.4% at 8 trials, 26.7% versus 19.2% at 4 trials), and the joint mixed search was better in 4 of 4 repetitions. At 2-4 blocks there was no advantage.
T5. Substitution. Bounded or smooth activations and normalization both stabilize deep residual stacks, so each makes the other less necessary and mixing lets a network use only what it needs.	The benefit of normalization shrinks for Tanh and ELU; the activation-by-normalization interaction is non-trivial.	Supported. At 16 blocks normalization added +14 points for LeakyReLU (BatchNorm versus none), about 3 for Tanh, and was harmful for ELU (none +26%, BatchNorm +18%, LayerNorm +11%). The interaction explained 10-19% of the variance. The diagnostics of trained 16-block networks agree: without normalization, 89% and 86% of the LeakyReLU and GELU pre-activations were negative in the early blocks against 52-58% with it, and their early blocks added only 0.15 and 0.11 of the residual stream, while ELU and Tanh without normalization had healthy early blocks (0.31 and 0.30).

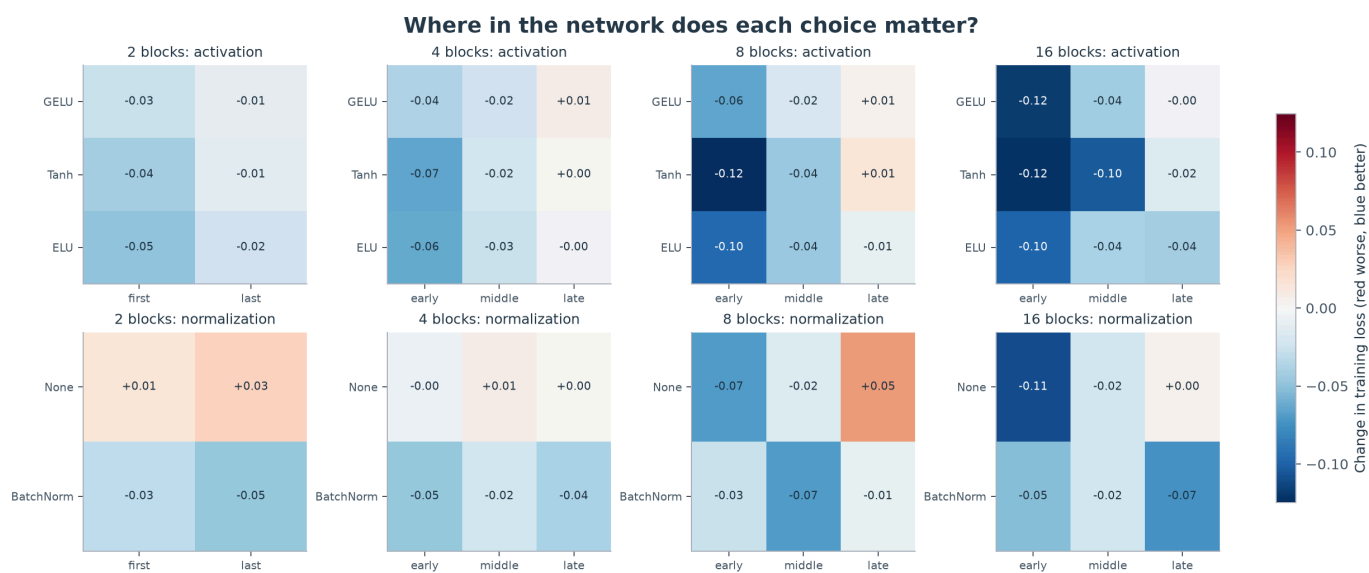


Figure 4. Position-specific effect of each choice. For each depth, a ridge regression of training loss on the fraction of blocks in the early/middle/late group using each choice (with repetition effects) gives the change in training loss when a whole group uses that choice instead of the default. Red = worse, blue = better.

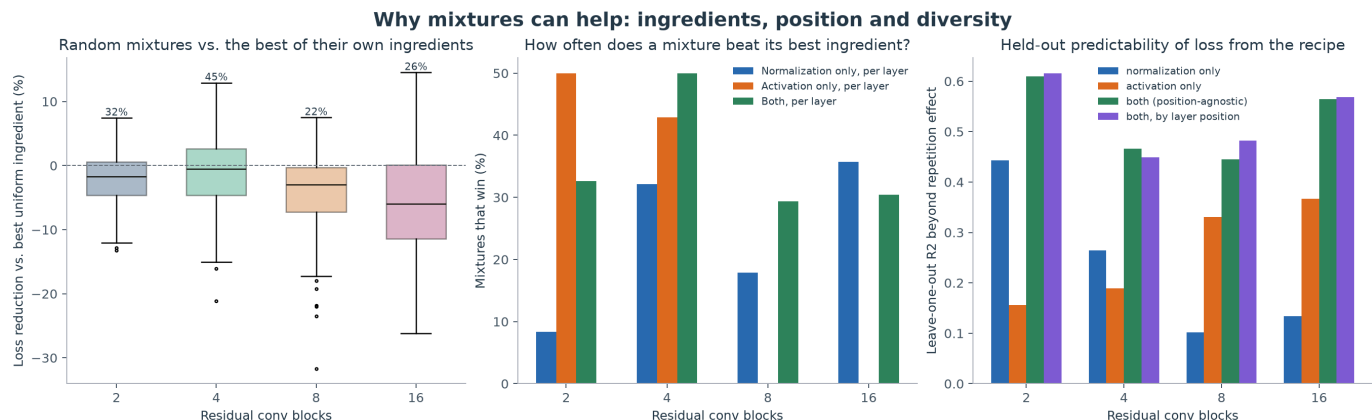


Figure 5. Left: for each random mixed configuration, the percentage by which its training loss beats the best uniform recipe built from the same ingredients (the percentage of mixtures that win is printed above each box). Middle: how often mixtures beat their best ingredient, by arm. Right: leave-one-out predictability of training loss from the recipe (beyond the repetition effect), using normalization only, activation only, both, and both by layer position.

Random mixtures are a weak test, and most do not help. Pooled over the three mixed arms, only 22-45% of random per-layer configurations beat the best uniform recipe built from their own ingredients (for the joint arm 33% and 50% at 2 and 4 blocks and 29% and 30% at 8 and 16 blocks), and the median mixture was worse than its best ingredient. Activation-only mixtures never beat their best ingredient at 8 or 16 blocks, because a random per-block activation includes LeakyReLU, the weakest choice, in some blocks. A mixture is only as good as its composition: a leave-one-out linear model on the fractions of blocks using each choice explained 46-58% of the variance in loss for both knobs together, and adding layer position raised this by at most 0.04.

So the claim that mixtures help needs care. Mixing is not a free benefit; what the experiments show is that when both knobs are free and the network is deep, a search that can assign different recipes to different blocks reaches lower loss than a search over one recipe for all blocks, and that normalization has diminishing returns. The deliberate mixtures below test the position explanation directly.

Deliberate mixtures: a follow-up test

Random mixtures mostly sample mediocre combinations, so they are a weak test of whether mixing can help. After the main search we therefore built mixtures on purpose. At each depth we took the four uniform recipes with the lowest mean training loss in the search (an ingredient choice made from search data only), and built (a) split mixtures, with recipe A in the first half of the blocks and recipe B in the second half, for every ordered pair, and (b) alternating mixtures A, B, A, B for every unordered pair. Every mixture and every uniform ingredient was re-trained with a new seed in each repetition, so comparisons are paired and free of the selection bias of the main search.

Blocks	Split: beats better ingredient / median gain	Alternating: beats better ingredient / median gain	Order effect: mean absolute loss difference between A→B and B→A
2	35% / -1.9%	n/a	6.8%
4	35% / -1.8%	29% / -4.0%	7.3%
8	44% / -0.6%	50% / +0.1%	10.4%
16	29% / -4.3%	38% / -0.9%	9.7%

Deliberate mixtures: do they help, and does placement matter?

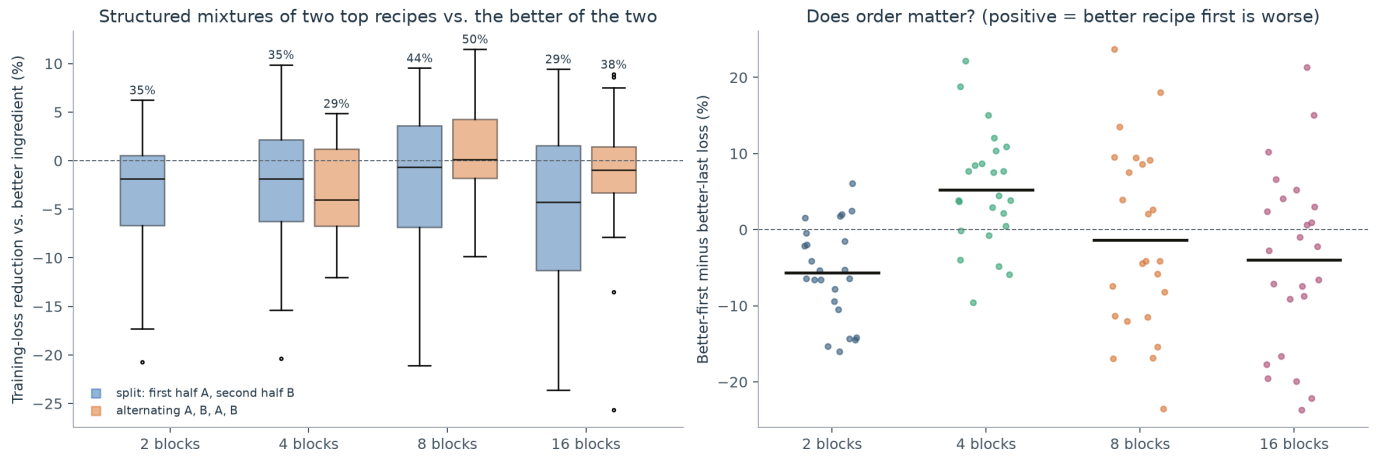


Figure 6. Deliberate mixtures of two top recipes. Left: training-loss reduction versus the better of the two uniform ingredients, paired within repetition (the percentage of mixtures that win is printed above each box). Right: for the same pair of recipes, the loss when the better uniform recipe is in the first half minus the loss when it is in the second half, in percent of their mean (positive means that putting the better recipe first is worse). The sign depends on which recipes are paired; the table above reports the size of the effect.

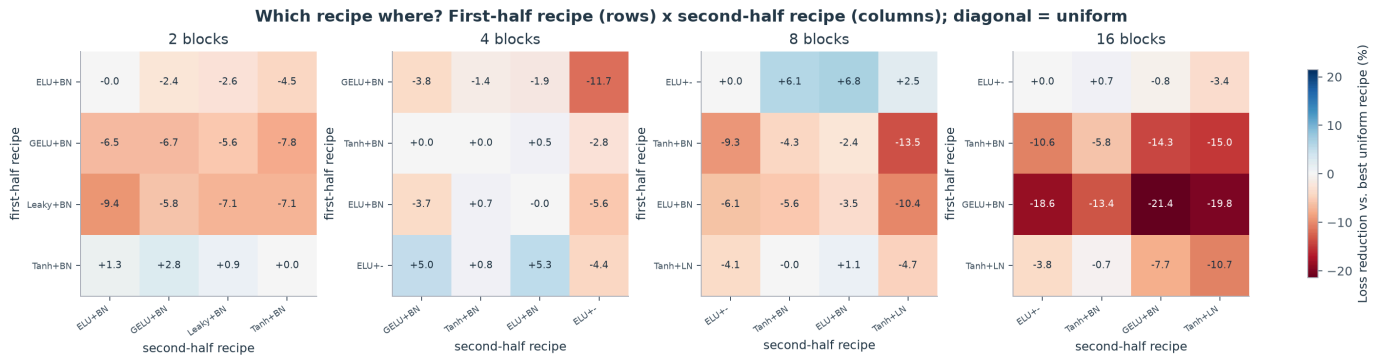


Figure 7. Training-loss reduction versus the best uniform recipe when recipe A (row) is used in the first half of the blocks and recipe B (column) in the second half; the diagonal shows the uniform recipes. Cells are means over search repetitions.

Result 1: mixing a pair of good recipes did not usually beat the better one. Split mixtures beat the better of their two ingredients in 35%, 35%, 44% and 29% of cases at 2, 4, 8 and 16 blocks, and alternating mixtures in 29%, 50% and 38% at 4, 8 and 16 blocks, with median gains against the better ingredient between -4.3% and +0.1%. They did beat the average of the two ingredients in 46-75% of cases (alternating mixtures at 16 blocks beat it in 75%, by 4.2% on average), so a typical mixture lands between its ingredients.

Result 2: order matters, much more than the mixing itself. Averaged over all pairs, the loss differed by 6.8%, 7.3%, 10.4% and 9.8% (mean absolute difference, at 2, 4, 8 and 16 blocks) between putting recipe A in the first half and B in the second half and the reverse. The direction was systematic. At 16 blocks, ELU without normalization followed by GELU + BatchNorm was 16.6% better than the reverse (4 of 4 repetitions: 19.9%, 6.6%, 22.1%, 17.7%), ELU without normalization followed by Tanh + BatchNorm was 10.1% better than the reverse (3 of 4 repetitions), Tanh + LayerNorm followed by Tanh + BatchNorm was 13.3% better than the reverse (4 of 4), and Tanh + LayerNorm followed by GELU + BatchNorm was 10.7% better than the reverse (4 of 4). In every one of these pairs the recipe with BatchNorm did better in the second half. Figure 7 shows the whole picture: at 4-16 blocks, rows that start with a BatchNorm recipe are mostly worse than the best uniform recipe.

Result 3: the best-placed mixtures were 3-6% better than the better ingredient. The largest paired gains against the better of the two ingredients were ELU without normalization followed by GELU + BatchNorm at 4 blocks (+4.9%), ELU without normalization followed by ELU + BatchNorm at 8 blocks (+6.3%) or Tanh + BatchNorm (+6.0%), Tanh + LayerNorm followed by Tanh + BatchNorm at 16 blocks (+4.6%), and an alternating ELU + none and GELU + BatchNorm mixture at 16 blocks (+4.3%). The worst mixtures, usually the reversed orders, were 10-19% worse than the better ingredient.

Two caveats. First, BatchNorm recipes were much noisier across seeds than recipes without it: the standard deviation of fresh-seed training loss over repetitions was 0.04-0.22 for the BatchNorm recipes among the top four (for example GELU + BatchNorm at 16 blocks, 1.139 with SD 0.221) against 0.02-0.05 for ELU without normalization, and the top-four recipes chosen from the search looked 0.02-0.21 better in the search than on fresh seeds (a selection effect that was largest for BatchNorm). Order effects are paired within a repetition, so they are not caused by this noise, but they may partly reflect how BatchNorm's running statistics behave at different positions. Second, we built mixtures from only two recipes and a single split point; richer structures could behave differently.

What the trained networks look like inside

To check the mechanisms behind the theories we trained the main uniform recipes at 16 blocks (two seeds each, protocol settings, training set only) and measured, for every block, the size of the block's output relative to the residual stream, the fraction of its pre-activations in the activation's weak region, and the scale of the stream. These are descriptive measurements of a few trained networks, not tests of causes.

Recipe	Last-block stream RMS	Block output / stream, early blocks	Block output / stream, late blocks	Weak-region fraction, early	Weak-region fraction, late
LeakyReLU, none	2.44	0.15	0.16	0.89	0.79
LeakyReLU, LayerNorm	2.88	0.26	0.07	0.58	0.57
LeakyReLU, BatchNorm	3.15	0.33	0.10	0.54	0.52
GELU, none	2.21	0.11	0.22	0.86	0.63
GELU, BatchNorm	2.68	0.32	0.11	0.54	0.53
Tanh, none	1.49	0.30	0.16	0.27	0.47
Tanh, BatchNorm	1.56	0.35	0.15	0.21	0.33
ELU, none	2.29	0.31	0.23	0.39	0.52
ELU, LayerNorm	2.10	0.34	0.12	0.21	0.20
ELU, BatchNorm	2.33	0.39	0.13	0.21	0.22

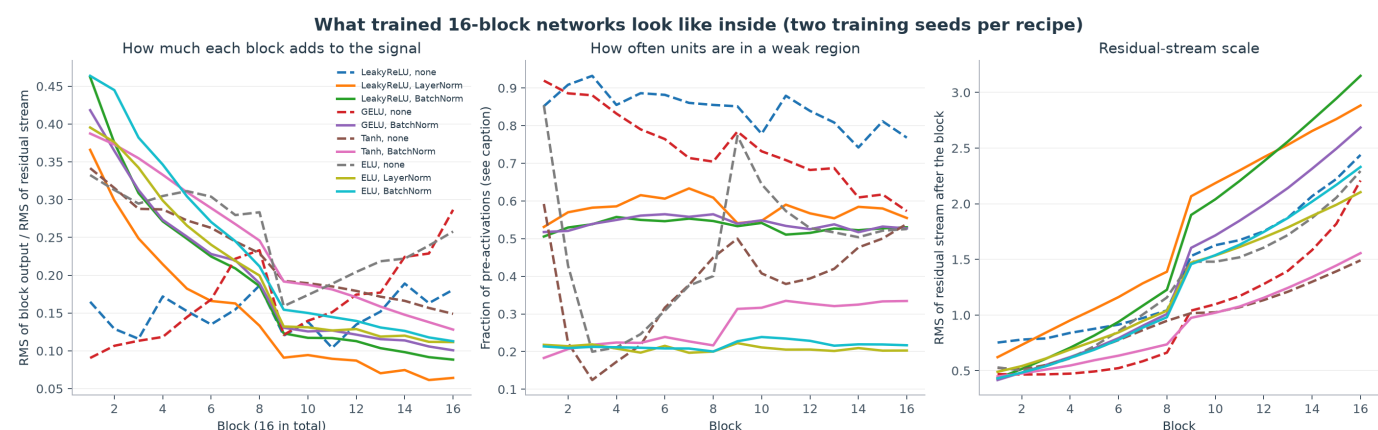


Figure 8. Inside trained 16-block networks. Left: RMS of each block's (scaled) output divided by the RMS of the residual stream it is added to. Middle: fraction of the block's pre-activations in the weak region of the activation (negative for LeakyReLU and GELU; below -1 for ELU; $|\tanh|$ above 0.9 for Tanh); the definitions differ by activation, so compare normalizations within an activation rather than across activations. Right: RMS of the residual stream after each block. Dashed lines are recipes without normalization. Averages over two training seeds, on 1,000 random training images.

Without normalization, LeakyReLU and GELU spent most of their time in the weak region. In the early blocks 89% and 86% of their pre-activations were negative (79% and 63% in the late blocks), against 52-58% for the same activations with LayerNorm or BatchNorm, and the early blocks added little to the signal: the block output was about 0.15 and 0.11 of the residual stream, against 0.26-0.33 with normalization. ELU and Tanh without normalization did not have this problem: their early blocks contributed 0.31 and 0.30 of the stream, about as much as the normalized versions (0.33-0.39), and only 39% (ELU, below -1) and 27% (Tanh, $|\tanh|$ above 0.9) of their units were in the weak region in the early blocks.

This fits a centering explanation. Normalization keeps pre-activations spread around zero. That matters for activations that discard most of the negative half (LeakyReLU, GELU) and matters little for activations that keep negative values informative (ELU, Tanh). It matches the grid of uniform recipes, where LeakyReLU gained 12% with BatchNorm and lost 2% without it at 16 blocks, while Tanh and ELU gained 22-26% without any normalization.

The signal-growth idea we had suspected did not hold. The residual stream after the last block was 2.4 for LeakyReLU without normalization and 2.9-3.1 with LayerNorm or BatchNorm, so normalization made the stream larger, not smaller. Tanh had the smallest stream (1.5), which fits its bounded output. ELU with LayerNorm was the worst ELU variant (training loss 1.12 on the diagnostic subset versus 0.90 without normalization); we did not test why. The jump in stream scale after block 8 is the second max-pool. These are measurements on two seeds per recipe and do not establish causes.

5. How many attempts, and how much compute?

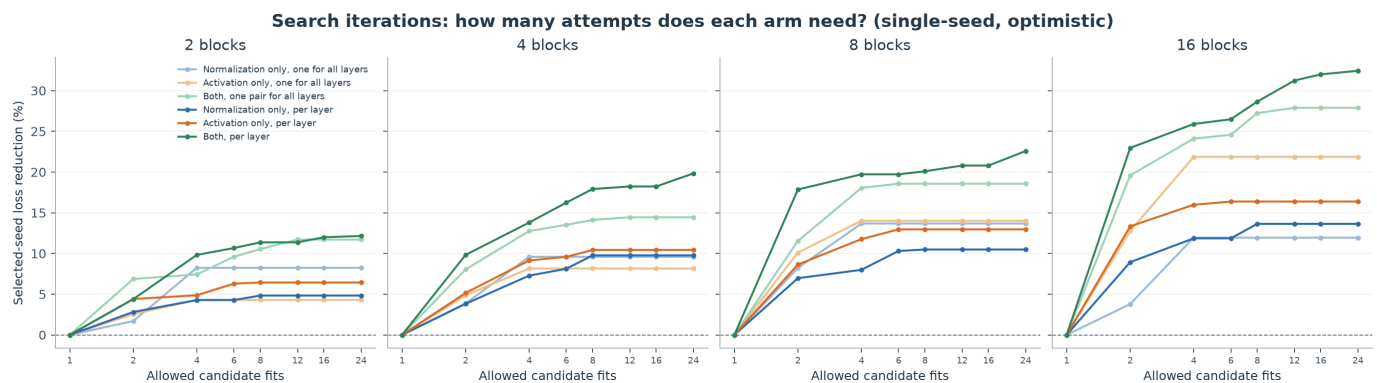


Figure 9. Single-seed selected loss reduction versus allowed candidate fits for every arm and depth. Because selection and measurement share a seed this curve is optimistic; fresh-seed values at selected budgets are in the table above and in measurements.csv.

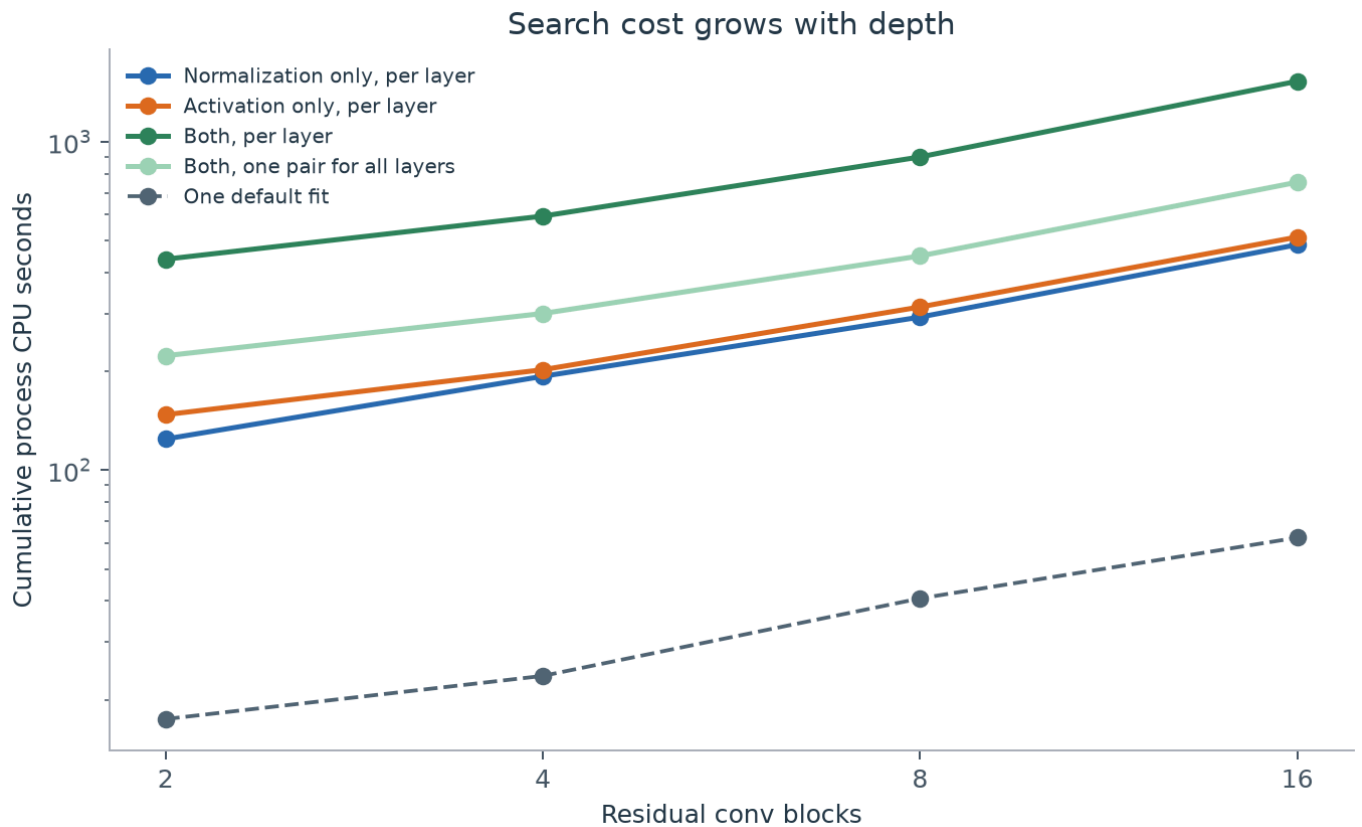


Figure 10. Cumulative process CPU seconds for each arm run to completion. This is additive compute across workers under shared machine load, not wall-clock latency.

Figure 9 shows how many attempts each arm needs (single-seed values, which are optimistic). The activation-only and normalization-only curves flatten after 4-8 trials because their spaces are small (3 or 4 uniform recipes, up to 8 mixed). The joint per-layer curve at 16 blocks keeps rising through 24 trials (23% after 2 trials, 26% after 4, 29% after 8 and 32% after 24), so the largest space is also the one where more attempts still help.

Figure 10 puts this on a compute axis. Running each arm to completion cost, at 16 blocks, about 179 CPU-seconds for the 3 normalization recipes, 262 for the 4 activation recipes, 754 for all 12 uniform recipes and 1,531 for the 24-trial joint per-layer search, against 18, 24, 41 and 62 CPU-seconds for one default fit at 2, 4, 8 and 16 blocks. In cost-effectiveness terms the 4-recipe activation search (262 s, 18.8%) is the cheapest large win at depth, the 12 uniform recipes (754 s, 25.1%) capture most of the best result, and the joint per-layer search buys a further 5.5 points for twice the cost.

Single-seed selection is optimistic, so every figure reports retrained fresh-seed values. Without that step the apparent gains would be larger and the comparisons between arms less reliable.

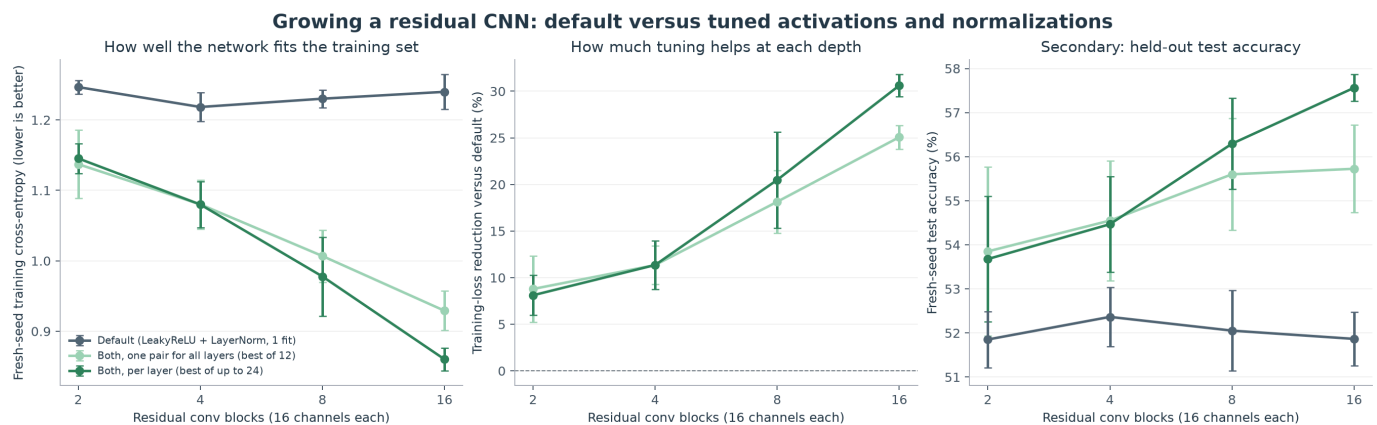


Figure 11. Growing the network. Left: fresh-seed training cross-entropy of the default, the best of all twelve uniform recipes, and the best per-layer mixture. Middle: percentage reduction versus the default. Right: held-out test accuracy, a secondary diagnostic never used for selection. Error bars are ± 1 descriptive SD over search repetitions.

6. What surprised us

1. The default did not use depth at all. LeakyReLU + LayerNorm gave the same training loss at 2 and 16 blocks, while Tanh, ELU or GELU in the same residual architecture improved steadily with depth. Changing only the activation was enough to unlock depth.
2. The best normalization at depth was often none. ELU with no normalization was the best uniform recipe at 16 blocks in the single-seed grid (+26%), better than ELU with BatchNorm or LayerNorm, so normalization is not a universal good; its value depends on the activation.
3. The more powerful knob flips with depth. At 2 blocks normalization explained two thirds of the differences; at 16 blocks activation explained three fifths. A single answer to 'which parameter is most powerful to tweak' does not exist; it depends on how deep the network is.
4. Mixing helps less than hoped, and in a narrow way. Random mixtures within one knob lost to the best uniform choice, and even the joint mixtures beat their own best ingredient only 29-50% of the time. The joint per-layer search nevertheless won at 8-16 blocks, which suggests that its advantage comes from reaching good compositions and placements that a uniform search cannot express, and not from the act of mixing.
5. Activation-only tuning was much more reproducible than normalization-only tuning (SD 1.4-2.1 versus 3.7-8.3 points). The best activation was the same across repetitions; the best normalization varied, possibly in part because BatchNorm's training-time and evaluation-time statistics add noise.
6. Unlike the earlier plain-CNN study, gains in fit carried over to held-out accuracy (+1.8 to +5.7 points), because this task is underfit (default accuracy about 52%) and fitting and generalizing are almost the same thing in that regime.
7. The first mechanism we suspected was wrong. We expected normalization to matter because unbounded activations let the residual stream grow; the stream was in fact smaller without normalization. What the measurements support is a centering effect: LeakyReLU and GELU without normalization sit mostly in their weak negative region, so their early blocks add little, while ELU and Tanh do not.

7. What to investigate next

- Test the mechanisms directly: measure the signal (residual stream scale, saturation, gradient norms) during training for the winning recipes, and ablate individual blocks (set one block's recipe to its alternative) to separate position effects from composition effects.
- Search with a smarter algorithm over placement: successive halving or evolutionary search over per-block recipes could find the position-structured mixtures that random search only finds by chance.
- Extend to more activations (SiLU, Softplus, learnable activations) and normalizations (GroupNorm with more groups, RMSNorm), and to wider networks, to see whether the crossover depth moves.
- Tune the learning rate and activation/normalization jointly: the optimum learning rate probably depends on the recipe, and the pilot hinted at that.
- Repeat for generalization: select on validation loss rather than training loss and on larger training sets, where the best recipe may differ.
- Use more repetitions (we have 4) and more datasets; the normalization-only results in particular have large run-to-run spread.

8. Limitations

- One dataset (CIFAR-10 subsampled and downsampled), one training set, narrow residual CNNs, and four search repetitions per cell. Bands are descriptive run-to-run variation, not confidence intervals for a broad population of tasks.
- The objective is fit to a fixed training set within a short 10-epoch budget, not generalization. Held-out accuracy is secondary and was never used for selection; tuning for generalization could rank choices differently.

- The default (LeakyReLU + LayerNorm) did not improve with depth in this architecture, which makes tuning gains look large. A different default would change the percentages.
- Learning rate, optimizer, initialization, residual scaling ($1/\sqrt{D}$) and training length are fixed; the activation and normalization effects interact with them. An earlier version of this study found different depth behavior in plain stacks, so conclusions are specific to this architecture family.
- Mixed configurations are sampled randomly; a better search algorithm could find better mixtures, so failure of random mixtures to beat uniform recipes does not mean better mixtures do not exist.
- BatchNorm is evaluated with running statistics; its effects mix normalization with batch-size and train/eval-mode differences. The regression analyses are descriptive: they use selection-influenced fits and cannot establish causes.
- The deliberate-mixture test and the signal diagnostics were designed after the main results were known: the position hypothesis came from the regression analysis of the main search, and the ingredients of the mixtures were chosen from search data. They use fresh seeds, but they are post hoc and exploratory, not confirmatory.
- The study is exploratory. Settings for the main search were fixed before it ran, but it is not an externally preregistered confirmatory study. CPU costs are local measurements.

9. Relation to earlier work

Random search is a standard baseline and its effectiveness depends on how many hyperparameters materially affect performance [1]. Joint architecture and hyperparameter search is an established research area [2]. Normalization layers are known to change the optimization landscape and its conditioning in deep networks [3, 4], and the choice of activation function interacts with initialization and signal propagation [5]. Residual connections with downscaled branches are a standard route to trainable deep networks [6]. Repeated selection on a noisy criterion can overfit it [7], motivating our fresh-seed assessment of every reported winner. CIFAR-10 supplies the image classification task [8].

[1] Bergstra, J. & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *JMLR* 13, 281–305.

[2] Bansal, A. et al. (2022). JAHS-Bench-201: A Foundation For Research On Joint Architecture And Hyperparameter Search. *NeurIPS Datasets and Benchmarks*.

[3] Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ICML*.

[4] Ba, J. L., Kiros, J. R. & Hinton, G. E. (2016). Layer Normalization.

[5] Hendrycks, D. & Gimpel, K. (2016). Gaussian Error Linear Units (GELUs).

[6] He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep Residual Learning for Image Recognition. *CVPR*.

[7] Cawley, G. C. & Talbot, N. L. C. (2010). On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *JMLR* 11, 2079–2107.

[8] Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images (CIFAR-10).

Reproducibility and accounting

Recorded fits: 771 main-search candidates, 312 fresh-seed assessments, 328 deliberate-mixture fits and 20 diagnostic trainings. Numerical failures in the main search: 0. Recorded main-search CPU cost: 7.8 CPU-hours; all recorded CPU including the follow-up stages: 14.7 CPU-hours, excluding pilots. CPU hours add across workers and are not elapsed clock hours.

The reproduction scripts check the data checksum, a balanced fixed training set, test-free search records, deterministic training, equal model size across choices, unique candidate sequences, and that every reported incumbent is the one that selection on training loss would pick. Raw JSONL records, frozen selections, protocol settings and tidy CSV tables are provided in the download bundle.