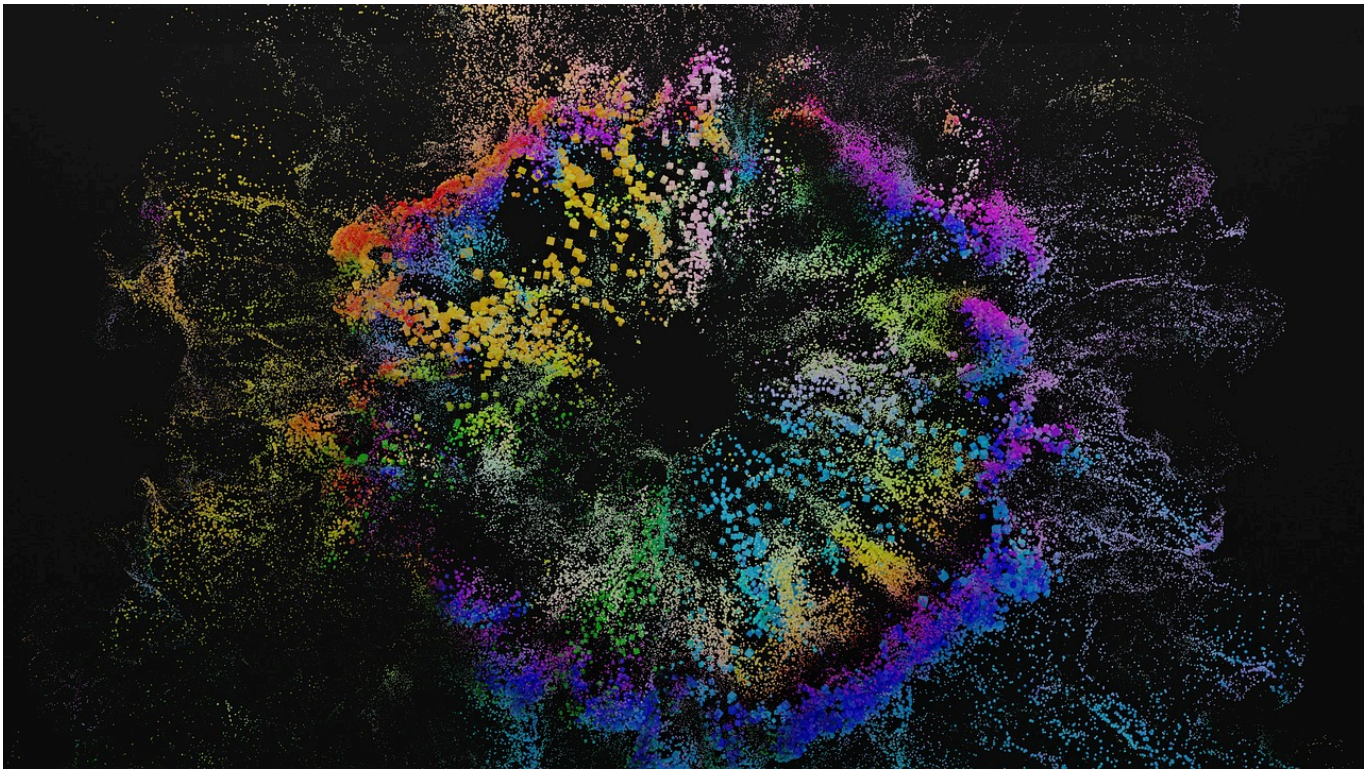
AI APPLICATIONS

I Measured Neural Network Training Every 5 Steps for 10,000 Iterations

What high-resolution training dynamics taught me about feature formation

Javier Marin

Nov 13, 2025 · 9 min read



I thought I understood how neural networks learned. Train them, watch the loss go down, save checkpoints

every epoch. Standard workflow. Then I measured training dynamics at 5-step intervals instead of epoch-level, and everything I thought I knew fell apart.

The question that started this journey: Does a neural network's capacity expand during training, or is it fixed from initialization? Until 2019, we all assumed the answer was obvious—parameters are fixed, so capacity must be fixed too. But Ansuini et al. discovered something that shouldn't be possible: the effective representational dimensionality *increases* during training. Yang et al. confirmed it in 2024.

This changes everything. If learning space expands *while* the network learns, how can we mechanistically understand what it's actually doing?

High-Frequency Training Checkpoints

When we are training a DNN with 10,000 steps, we use to set up chack points every 100 or 200 steps. Measuring at 5-step intervals generates too much records that aren't easy to manage. But this high-frquency checkpoints reveals provide very valuable information about how DNN learns.

High-frequency checkpoints provides information about:

- Whether early training mistakes can be recovered from (they often can’t)
- Why some architectures work and others fail
- When interpretability analysis should happen (spoiler: way earlier than we thought)
- How to design better training approaches

During an applied research project I have measured DNN training at high resolution — every 5 steps instead of every 100 or 500. I used a basic MLP architecture with the same dataset I”m’ve been using last 10 years.

Parameter	Value
Architecture	784-256-128-10 (MLP)
Dataset	MNIST (60k train/10k test)
Optimizer	Adam
Hyperparameters	$\beta_1 = 0.9$, $\beta_2 = 0.999$
Learning rate	0.001, batch size 64
Training duration	8000 steps
Loss function	Cross-entropy

Figure 1. Experimental setupWe detect discrete transitions using z-score analysis with rolling statistics:

The results were surprising. Deep neural networks, even simple architectures, expand their effective parameter space during training. I had assumed this space was predetermined by the architecture itself. Instead, DNNs undergo discrete transitions—small

jumps that increase the effective dimensionality of their learning space.

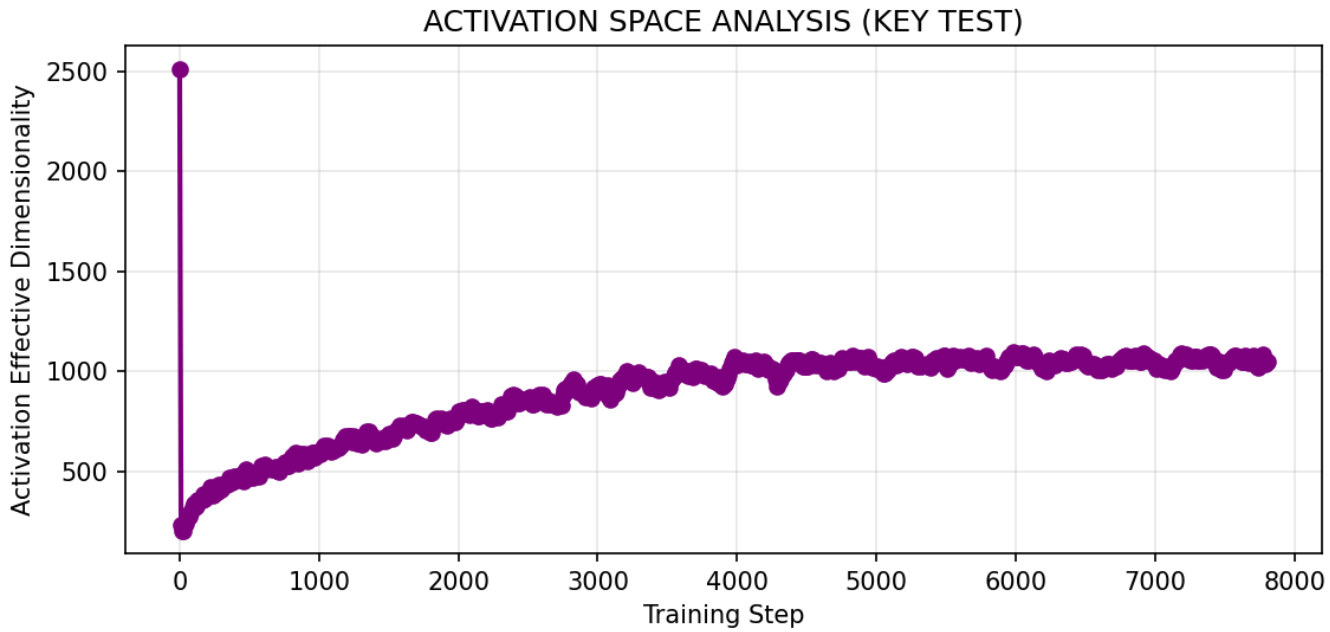


Figure 2: Effective dimensionality of activation patterns during training, measured using stable rank. We see three distinct phases emerge: initial collapse (steps 0-300) where dimensionality drops from 2500 to 500, expansion phase (steps 300-5000) where dimensionality climbs to 1000, and stabilization (steps 5000-8000) where dimensionality plateaus. This suggests steps 0-2000 constitute a qualitatively distinct developmental window. Image by author.

In Figure 2 we can see the monitoring of activation effective dimensionality during training. We see these transitions concentrate in the first 25% of training, and are hidden at larger checkpoint intervals (100-1000 steps). We needed a high-frequency checkpointing (5 steps) to detect most of them. The curve also show an interesting behavior. The initial collapse represents loss landscape restructuring where random initialization gives way to task-aligned structure. Then we see a expansion phase with gradual dimensionality growth. Between 2000-3000 steps, there is a

stabilization that reflects DNN architectural capacity limits.

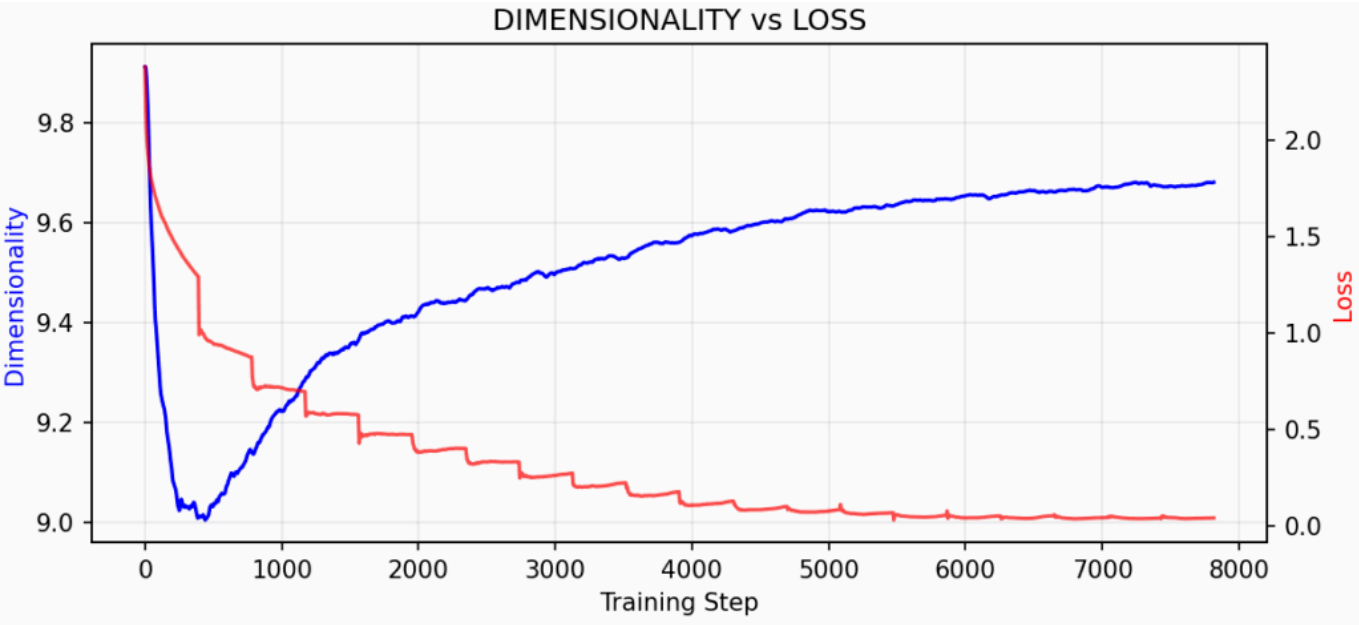


Figure 3: Representational dimensionality (measured using stable rank) shows strong negative correlation with loss (ρ = −0.951) and moderate negative correlation with gradient magnitude (ρ = −0.701). As loss decreases from 2.0 to near zero, dimensionality expands from 9.0 to 9.6. Counterintuitively, improved performance correlates with expanded rather than com- pressed representations. Image by author.

This changes how we should think about DNN training, interpretability, and architecture design.

Exploration vs Expansion

Consider the following two scenarios:

Scenario A: Fixed Capacity (Exploration)	**Scenario B: Expanding Capacity (Innovation)**
Your network starts with a fixed representational capacity. Training explores	Your network starts with minimal capacity. Training *creates* representational

different regions of this pre-determined space. It is like navigating a map that exists from the beginning. Early training just means “haven’t found the good region yet”.	structures. Its like building roads while traveling — each road enables new destinations. Early training establishes what becomes learnable later.

Which is it?

The question matters because if capacity expands, then early training isn’t recoverable. You can’t just “train longer” to fix early mistakes. So, interpretability has a *timeline* where features form in sequence. Understanding this sequence is key. Furehtermore, architecture design seems to be about expansion rate not just final capacity. Finally, critical periods exist. If we miss the window, we miss the capability.

When We Need to Measure High-Frequency Checkpoints

Expansion vs Exploration

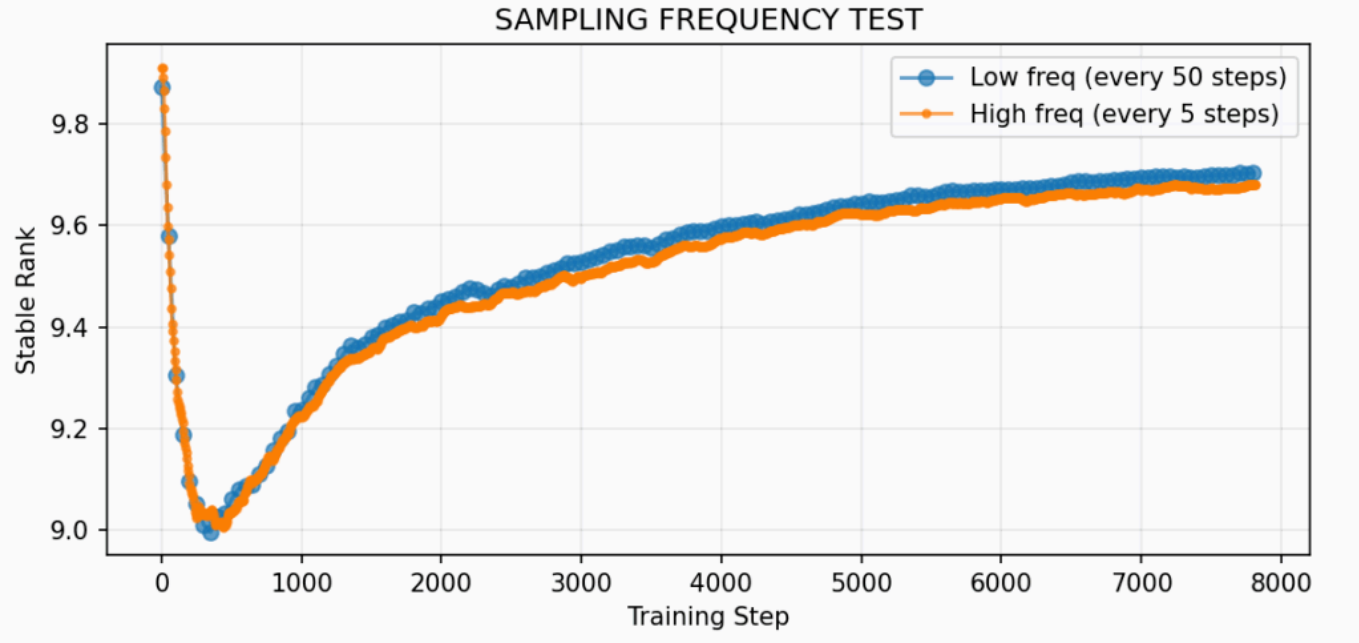


Figure 4: High-frequency sampling vs. Low Frequency sampling in the experiment described in Figure 1. We detect discrete transitions using z-score analysis with rolling statistics. High-frequency sampling captures rapid transitions that coarse-grained measurement misses. This comparison tests whether temporal resolution affects observable dynamics.

As seen in Figures 2 and 3, high-frequency sampling reveals interesting information. We can indentify three different phases:

Phase 1: Collapse (steps 0-300) The network restructures from random initialization. Dimensionality drops sharply as the loss landscape is reshaped around the task. This isn't learning yet, it's preparation for learning.

Phase 2: Expansion (steps 300-5,000)
Dimensionality climbs steadily. This is capacity expansion. The network is building representational structures. Simple features that enable complex features that enable higher-order features.

Phase 3: Stabilization (steps 5,000-8,000) Growth plateaus. Architectural constraints bind. The network refines what it has rather than building new capacity.

This plots reveals expansion, not exploration. The network at step 5,000 can represent functions that were impossible at step 300 because didn't exist.

Capacity Expands, Parameters Don't

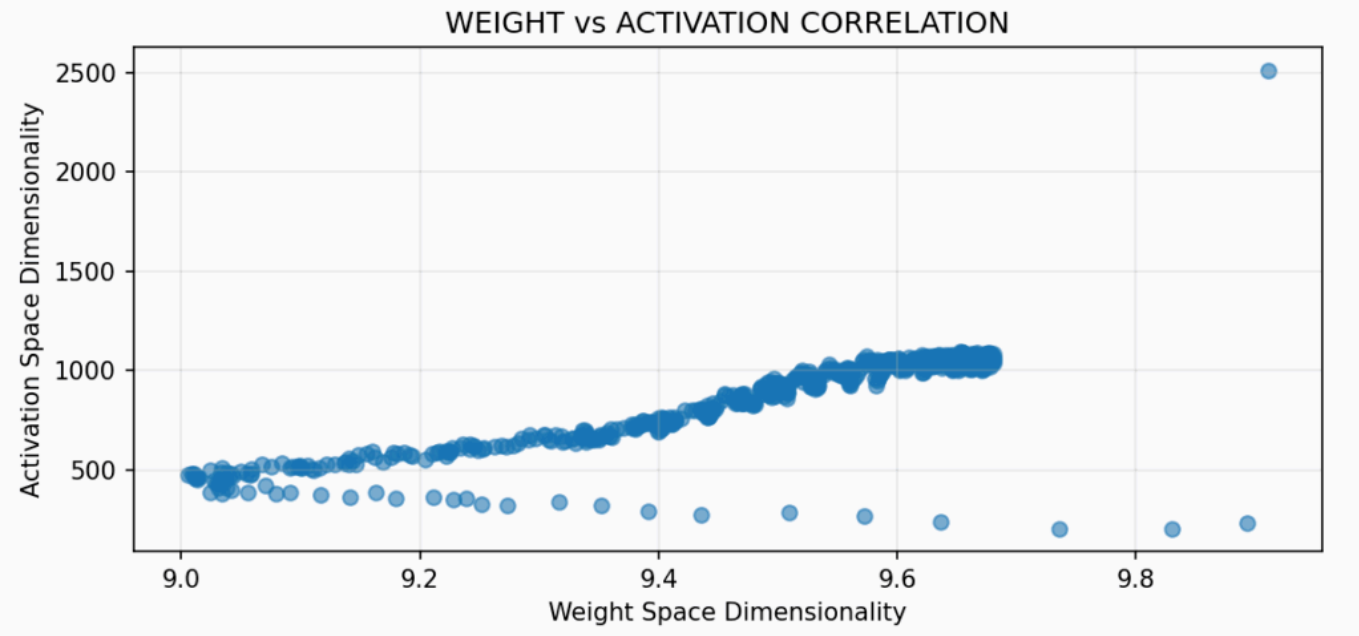


Figure 5: Comparison of activation space to weight space. Weight space dimensionality remains nearly constant (9.72-9.79) with only 1 detected "jump" across 8000 steps. Image by author

The comparison between activation and weight spaces shows that both follows different dynamics with high-frequency sampling. Activation space shows ap. 85 discrete jumps (including Gaussian noise). Weight space shows only 1. The same network with the same training run. It confirms that the network at step 8000 computes functions inaccessible at step 500 despite identical parameter count. This is the clearest evidence for expansion.

> *DNN innovate by generating new parameter space options during training in order to represent complex taks.*

Transitions Are Fast and Early

We have seen how high-frequency sampling shows much more transitions. Low-frequency checkpointing would miss nearly all of them. This transitions concentrate early. Two thirds of all transitions happen in the first 2,000 steps — just 25% of total training time. It means that if we want to understand what features form and when, we need to look during steps 0-2,000, not at convergence. By step 5,000, the story is over.

Expansion Couples to Optimization

If we look again at Figure 3, we see that as loss decreases, dimensionality expands. The network doesn't simplify as it learns. It becomes more complex. Dimensionality correlates strongly with loss ($\rho = -0.951$) and moderately with gradient magnitude ($\rho = -0.701$). This could seem counterintuitive: improved performance correlates with *expanded* rather than *compressed* representations. We might expect networks to find simpler, more compressed

representations as they learn. Instead, they expand into higher-dimensional spaces.

Why?

A possible explanation is that complex tasks require complex representations. The network doesn't find a simpler explanation and builds the representational changes needed to separate classes, recognize patterns, and generalize.

Practical Deployment

We have seen a different way to understand and debug DNN training across any domain.

> *If we know when features form during training, we can analyze them as they crystallize rather than reverse-engineering a black box afterward.*

In real deployment scenarios, we can track representational dimensionality in real-time, detect when expansion phases occur, and run interpretability analyses at each transition point. This tells us precisely when our network is building new representational structures—and when it's finished. The measurement approach is architecture-agnostic: it works whether you're training CNNs for vision, transformers for

language, RL agents for control, or multimodal models for cross-domain tasks.

Example 1: Intervention experiments that map causal dependencies. Disrupt training during specific windows and measure which downstream capabilities are lost. If corrupting data during steps 2,000-5,000 permanently damages texture recognition but the same corruption at step 6,000 has no effect, you've found when texture features crystallize and what they depend on. This works identically for object recognition in vision models, syntactic structure in language models, or state discrimination in RL agents.

Example 2: For production deployment, continuous dimensionality monitoring catches representational problems during training when you can still fix them. If layers stop expanding, you have architectural bottlenecks. If expansion becomes erratic, you have instability. If early layers saturate while late layers fail to expand, you have information flow problems. Standard loss curves won't show these issues until it's too late—dimensionality tracking surfaces them immediately.

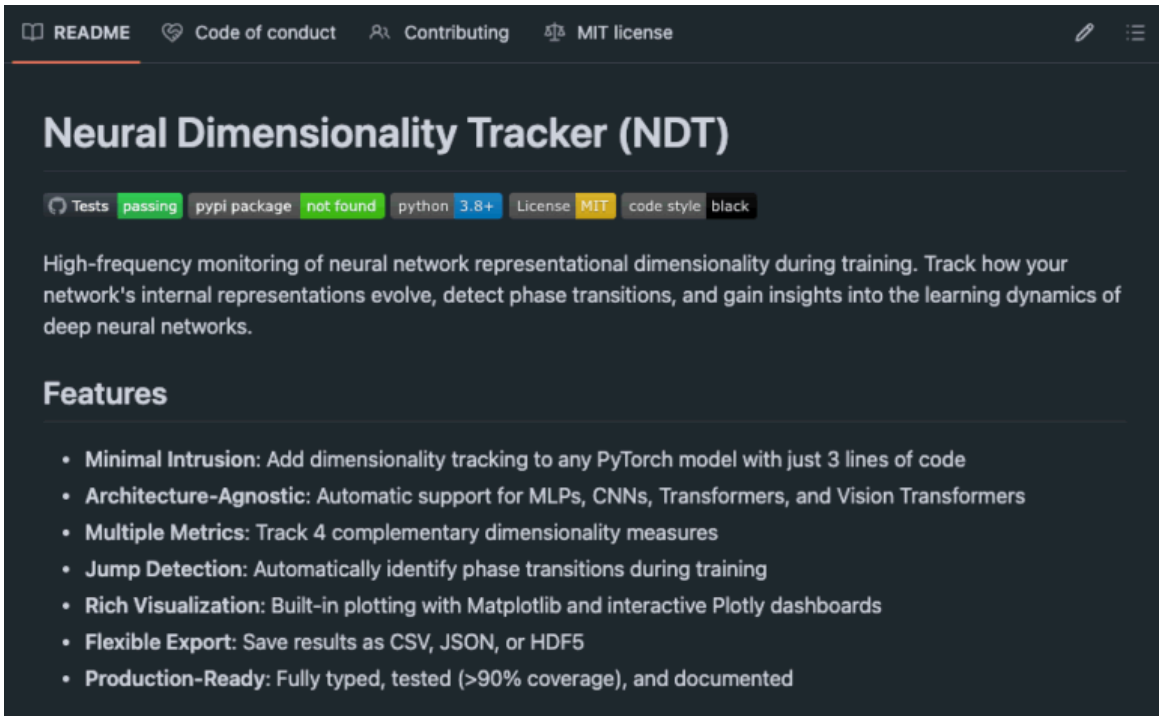
Example 3: The architecture design implications are equally practical. Measure expansion dynamics during the first 5-10% of training across candidate architectures. Select for clean phase transitions and structured bottom-up development. These networks aren't just more performant—they're fundamentally more interpretable because features form in clear sequential layers rather than tangled simultaneity.

What's Next

So we've established that networks expand their representational space during training, that we can measure these transitions at high resolution, and that this opens new approaches to interpretability and intervention. The natural question: can you apply this to your own work?

I'm releasing the complete measurement infrastructure as open source. I included validated implementations for MLPs, CNNs, ResNets, Transformers, and Vision Transformers, with hooks for custom architectures.

Everything runs with three lines added to your training loop.



The Gitgub repository provides experiment templates for the experiments discussed above: feature

formation mapping, intervention protocols, cross-architecture transfer prediction, and production monitoring setups. The measurement methodology is validated. What matters now is what you discover when you apply it to your domain.

Try it: Neural Dimensionality Tracker (NDT)

The code is production-ready. The protocols are documented. The questions are open. I would like to see what you find when you measure your training dynamics at high resolution no mather the context and the architecture.

You can share your results, open issues with your findings, or just ⭐ the repo if this changes how you think about training. Remember, the interpretability timeline exists across all neural architectures.

Javier Marín | LinkedIn | Twitter

References & Further Reading

- Achille, A., Rovere, M., & Soatto, S. (2019). Critical learning periods in deep networks. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=BkeStsCcKQ

- Frankle, J., Dziugaite, G. K., Roy, D. M., & Carbin, M. (2020). Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning* (pp. 3259-3269). PMLR. https://proceedings.mlr.press/v119/frankle20a.html
- Ansuini, A., Laio, A., Macke, J. H., & Zoccolan, D. (2019). Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems* (Vol. 32, pp. 6109-6119). https://proceedings.neurips.cc/paper/2019/hash/cfcce0621b49c983991ead4c3d4d3b6b-Abstract.html
- Yang, J., Zhao, Y., & Zhu, Q. (2024). ε-rank and the staircase phenomenon: New insights into neural network training dynamics. *arXiv preprint arXiv:2412.05144*. https://arxiv.org/abs/2412.05144
- Olah, C., Mordvintsev, A., & Schubert, L. (2017). Feature visualization. *Distill*, *2*(11), e7. https://doi.org/10.23915/distill.00007
- Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., &

Olah, C. (2021). A mathematical framework for transformer circuits. *Transformer Circuits Thread*. https://transformer-circuits.pub/2021/framework/index.html