The Marauder's Map: Bézier Manifolds Reveal Hidden Surfaces for Model Merging and Ensembling

Abhiram Iyer[1, 3]
Mark T. Harnett[†2, 3] **Sarthak Chandra**[†4]

[1]Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA
[2]Department of Brain and Cognitive Sciences, MIT, Cambridge, MA
[3]McGovern Institute for Brain Research, MIT, Cambridge, MA
[4]International Centre for Theoretical Sciences (ICTS-TIFR), Bengaluru, KA, India
{abiyer, harnett}@mit.edu, sarthak.chandra@icts.res.in

Abstract

Neural circuits are famously more robust than their deep network counterparts. This phenomenon is often explained through degeneracy, where structurally distinct configurations support the same function. However, real biological systems exhibit a more flexible form of degeneracy in which specialized circuits continually adapt towards new competencies through localized changes while preserving existing function. Deep learning studies degeneracy through mode connectivity, which connects same-objective networks by low-loss curves, sheets, or volumes. However, the field lacks an analogous counterpart to flexible degeneracy, which demands more than ordinary mode connectivity. Defining such an analogue requires asking whether many independently initialized models trained on distinct tasks can be connected by a high-dimensional region of competent multi-task solutions. To address this issue, we introduce *Nimbus*, which learns a high-dimensional Bézier manifold connecting N specialists, going beyond the one-dimensional paths and two-dimensional sheets of prior mode connectivity work. A naive parameterization would scale combinatorially with N, but *Nimbus* requires learning correction tensors of only 3 interaction orders, showing that the connecting manifold has surprisingly low complexity yet supports multi-task competencies. The resulting manifold from *Nimbus* forms a navigable "Marauder's Map" that connects independently trained specialist models through a shared geometry of competent solutions. Trained across vision and language domains, *Nimbus* produces manifolds that are functionally and locally weight-flat: ensembles sampled from them degrade more gracefully than baselines under input and weight perturbations, with the largest gains in negative log-likelihood loss and expected calibration error. Thus, *Nimbus* supplies a deep learning interpretation of flexible degeneracy, recasting model connectivity from a low-dimensional path between distant solutions into a high-dimensional volume of nearby ones.

1 Introduction

One of the key features of biological neural circuits is their robustness under synaptic noise. This defining characteristic has long been viewed through the lens of biological *degeneracy* [1], in which structurally distinct micro-configurations support the same macro-level function. Analogously, deep learning studies a similar phenomenon but seeks to answer how distinct yet functionally similar

[†]Equal supervision and co-corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

solutions are *connected* in high-dimensional space. Mode connectivity shows that deep networks trained on the same objective can often be linked by low-loss curves or linear paths [2–4], with some work even extending to two-dimensional nonlinear sheets [5] and low-loss volumes [6].

However, *real* biological systems routinely undergo structural change as they learn new competencies. For example, dendritic spines grow and shrink, and can even transition a reservoir of silent synapses into active connections to directly alter synaptic strength. Meanwhile, dendrites themselves undergo broader morphological remodeling, altering their overall length, branching complexity, and local signaling capacity. Together, these elements of structural change implement putative local learning mechanisms such as synaptic clustering or dendritic compartmentalization. These local degrees of freedom (Fig. 1) are the means by which circuits can change without completely overwriting prior behavior [7–12].

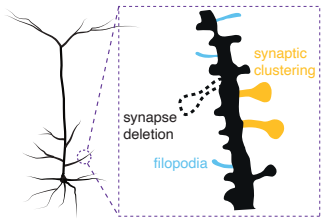


Figure 1: **Local degrees of freedom in biological learning.** Dendrites support synaptic clustering (yellow), deletion (dashed), and filopodia (blue) carrying silent synapses that can become active. These localized mechanisms of change let circuits learn without overwriting existing function.

Flexible degeneracy is a more accurate description of biology, where specialized circuits can gradually adapt towards multiple new competencies through sequences of localized change while still preserving existing functionality. The equivalent question in deep learning is clear. Can a high-dimensional manifold connect *many* independently initialized models that are trained on *distinct* tasks while maintaining uniform task competency and retaining prior functionality?

We study exactly this. Given N independently initialized models with a shared architecture but trained on different tasks, we learn a high-dimensional manifold connecting them, a setting that is stricter than ordinary mode connectivity as endpoints need not solve every task. Moreover, we will demonstrate that this manifold need not have a complex structure, but can instead be parameterized through a small number of tunable controls. We propose *Nimbus*: an N-expert Interpolating Manifold built as a Bezier surface over an Underlying Simplex. The N specialist models anchor the vertices of the simplex, and *Nimbus* learns the $(N-1)$-dimensional manifold that fills its interior, supplying an infinite family of competent multi-task models. Empirically, we will show that *Nimbus* constructs functional manifolds between independently trained models in both vision and language.

Through this lens, *Nimbus* operates under a richer premise in both mode connectivity *and* ordinary model merging. Most merging methods [13–17] output a single model while assuming no access to task data. We instead operate in a distinct regime where task data is available and the goal is to learn the *geometry* of the merge. Crucially, data here is not merely supervision for a single output model, but the foundation for building a high-dimensional manifold of models instead.

Once such a manifold has been constructed, we can also efficiently sample an arbitrarily large number of models for ensembling. Under input and weight perturbations, this ensemble model maintains strong accuracy while improving negative log-likelihood (NLL) loss and expected calibration error (ECE), metrics that gauge predictive quality and uncertainty more reliably than top-K accuracy [18, 19]. Without explicit flatness constraints, the process naturally yields a high-dimensional, functionally and locally weight-flat manifold of specialized models—a structural feature that conveniently aligns with established heuristics for out-of-distribution generalization.

1.1 Related work

Mode connecting paths and surfaces. Mode connectivity asks whether trained networks are isolated minima or points in a connected low-loss set. Ainsworth et al. [20] found that low-loss *linear* paths between differently initialized same-objective models can only exist after neuron asymmetries are resolved via permutation. Garipov et al. [2] and Draxler et al. [3] found *nonlinear* low-loss paths between modes by constructing simple curves, Nguyen [21] gave theoretical conditions for connected sublevel sets, and Zhou et al. [22] studied connectivity at the feature level. Beyond paths, Benton et al. [6] show that same-objective modes can lie in multi-dimensional low-loss simplicial complexes, and Ren et al. [5] construct two-dimensional Bézier surfaces connecting up to four networks. These are the closest geometric precedents to *Nimbus*, but all assume *same-objective* training and connect at most a handful of networks. *Nimbus* drops both assumptions as *many* models are *independently* initialized

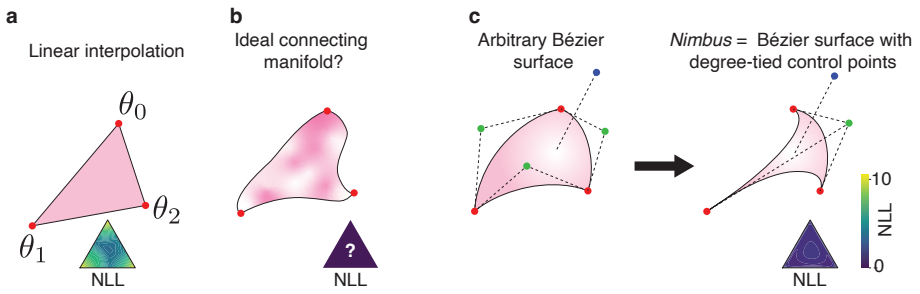


Figure 2: ***Nimbus* constructs a low-loss, high-dimensional volume through a small number of corrections. a.** Constructing multi-task models through linear interpolation of independent task-trained models results in high NLL due to feature differences and neuron asymmetries. **b.** We hypothesize the existence of an ideal connecting manifold that can demonstrate low loss on the joint set of tasks in its entire volume. **c.** (*left*) Could an arbitrary Bézier manifold defined over a simplex satisfy this requirement? Even if it could, a naive parameterization would require combinatorially many correction terms, e.g., the green dots indicate learned quadratic corrections for every edge. (*right*) *Nimbus* demonstrates that such manifolds not only exist, but can be parameterized through only a small number of corrective terms shared per-degree.

and trained on *disjoint* tasks. Non-trivially, *Nimbus* finds a high-dimensional mode-connected region with multi-task capabilities while also having specialist models at the boundaries of this region.

Model merging. Model merging combines trained models without retraining. Approaches include weight averaging in a compatible basin [23], curvature-weighted combination (Fisher Merging [13]), task-vector composition over a shared pre-trained base (Task Arithmetic and TIES Merging [24, 15]), feature-matched merging across specialist networks (ZipIt! [16]), permutation-based merging of same-objective models (Git Re-Basin [20]), and data-less or adaptive variants [14, 17], with recent LLM work absorbing capabilities from homologous specialists [25]. All collapse N specialists into a single model and typically assume a shared pre-trained base or fine-tuning lineage. Feature- and permutation-matching variants further require near-identical tasks, or merge only part of the network and leave per-specialist heads that scale linearly with N. *Nimbus* instead preserves specialists exactly and adds only a constant number of learnable corrections to build a continuum of multi-task models between them, a geometry that can be traversed to tune task mixtures, sampled for ensembling, and stress-tested under perturbations.

Single-model solutions for robustness and multi-task training. A separate line of work improves robustness or mitigates task interference at the level of a single solution. PCGrad addresses interference during joint training by projecting conflicting gradients onto orthogonal subspaces [26]. Deep ensembles remain strong baselines under distribution shifts [27, 19] but require many independent models. Stochastic weight averaging (SWA) averages along an SGD trajectory to find wider optima [28], and Adaptive Sharpness-Aware Minimization (ASAM) explicitly optimizes for weight flatness [29]. *Nimbus* presents a continuum with a closed functional form, so ensembling can operate over an arbitrarily large set of models. Even without explicit functional and weight flatness penalties, its ensemble outperforms these baselines under perturbations, suggesting that the learned manifold is naturally well-conditioned.

Contributions. We make the following key contributions:

- **A high-dimensional manifold connecting independently trained specialists.** Prior mode-connectivity work links models through low-loss paths and surfaces only when they share the same objective. *Nimbus* vastly generalizes this paradigm by establishing an $(N-1)$-dimensional low-loss manifold across N specialists trained with *distinct* initializations on *distinct* training tasks.
- **Connectivity manifold has low intrinsic complexity.** The high-dimensional manifold is parameterized as a linear mixture of the N frozen specialists plus a small number of additive correction tensors that capture how groups of specialists interact. A natural parameterization would assign one correction per face of the simplex and would grow exponentially in N. Non-trivially, *Nimbus* finds

that just 3 interaction orders are sufficient to build competent manifolds, keeping the construction tractable at scale.

- **Functionally and locally weight-flat manifolds.** When *Nimbus* is trained across vision and language domains, sampling from its manifold yields ensembles that degrade more gracefully than baselines under input and weight perturbations, with the largest gains in NLL and ECE, metrics that best capture predictive-distribution quality.
- **A deep learning instantiation of flexible neural degeneracy.** Just as a neural circuit uses localized mechanisms to acquire new competencies in high-dimensional space without catastrophic overwriting (i.e., flexible degeneracy), a specialist can smoothly accumulate new task competencies on *Nimbus*'s manifold while small synaptic perturbations on the same manifold yield functionally equivalent yet differently configured models. Our work fundamentally unifies neural degeneracy with mode connectivity and model merging, consequently recasting degeneracy from low-dimensional surfaces to high-dimensional volumes.

2 How do we construct a Bézier manifold between models?

2.1 Preliminaries

We are given N models $\{f_{\theta_i}\}_{i=1}^N$ that share a single architecture f_θ and have been trained independently on different tasks $\{\mathcal{T}_i\}_{i=1}^N$ with independent arbitrary initializations. Our goal is to construct a single *parameter map* $\theta : \Delta^{N-1} \to \mathbb{R}^{|\theta|}$, where Δ^{N-1} is the $(N-1)$-simplex ($\Delta^{N-1} = \{\boldsymbol{w} \in \mathbb{R}^N : w_i \geq 0, \sum_{i=1}^N w_i = 1\}$), such that $\theta(\boldsymbol{w})$ at any mixing weight $\boldsymbol{w} \in \Delta^{N-1}$ yields a model that performs well on the combination of tasks corresponding to the nonzero entries of $\boldsymbol{w}$. Specifically, this demands that the vertices $\boldsymbol{e}_i$ of the simplex Δ are recovered exactly, i.e., $\theta(\boldsymbol{w} = \boldsymbol{e}_i) = \theta_i$, and that interior $\boldsymbol{w}$ yields models that behave like genuine multi-task models rather than naive parameter averages.

Model soups and linear mode connectivity [23, 4] typically employ a linear combination of weights (i.e., the parameter map $\theta(\boldsymbol{w}) = \sum_i w_i \theta_i$). However, when tasks are disjointly defined and models are free to start from arbitrary initializations, linear combinations are insufficient due to conflicting features and neuron asymmetries (Fig. 2a). Given this practical constraint, we seek to construct parameter maps through nonlinear combinations of model weights to answer whether such a low-loss manifold spanning specialist models can be constructed automatically (Fig. 2b).

2.2 Bézier manifolds

We consider an N-variate, d-degree Bézier surface as our natural starting point. Such a manifold is typically written in the form of:

$$\theta(\boldsymbol{w}) = \textstyle\sum_{|\boldsymbol{\alpha}|=d} b^d_{\boldsymbol{\alpha}}(\boldsymbol{w}) P_{\boldsymbol{\alpha}}, \quad b^d_{\boldsymbol{\alpha}} = \frac{d!}{\alpha_1! \alpha_2! \ldots \alpha_N!} \prod_{i=1}^N w_i^{\alpha_i} \tag{1}$$

where $\boldsymbol{\alpha}$ is a multi-index that iterates over all interaction terms on the $(N-1)$-simplex, $b^d_{\boldsymbol{\alpha}}$ are the Bernstein basis polynomials, and $P_{\boldsymbol{\alpha}}$ are the Bézier (or Bernstein) coefficients (conventionally referred to in mathematical literature as "control points"). We note that when $\boldsymbol{\alpha} = d\boldsymbol{e}_i$ for unit basis vector $\boldsymbol{e}_i \in \mathbb{R}^N$, $P_{\boldsymbol{\alpha}} = \theta_i$ is the original i-th vertex model exactly. We can reinterpret Eq. 1 as a linear combination of vertex models plus additive correction terms applied to each face of the simplex (see Supp. Sec. D for an expanded proof).

$$\theta(\boldsymbol{w}) = \textstyle\sum_{i=1}^N w_i \theta_i + \sum_{|\boldsymbol{\alpha}|=d} b^d_{\boldsymbol{\alpha}}(\boldsymbol{w})\, \tilde{P}_{\boldsymbol{\alpha}}, \qquad \tilde{P}_{\boldsymbol{\alpha}} := P_{\boldsymbol{\alpha}} - \textstyle\sum_{i=1}^N \frac{\alpha_i}{d} \theta_i. \tag{2}$$

Here, $\tilde{P}_{\boldsymbol{\alpha}}$ is the residual Bernstein coefficient after subtracting the affine interpolation of the vertex models. If $N = 3$, for instance, these corrections are added to the three pairwise edges and the single 2-face of the $\{\theta_0, \theta_1, \theta_2\}$ triangle.

2.3 Learnable corrections on a Bézier manifold

Our method falls directly out of Eq. 2 given a specific choice of $b^d_{\boldsymbol{\alpha}}$ and learnable $\tilde{P}_{\boldsymbol{\alpha}}$, where $|\tilde{P}_{\boldsymbol{\alpha}}| = |\theta|$. We use $\tilde{b}^d_{\boldsymbol{\alpha}} = \frac{b^d_{\boldsymbol{\alpha}}(\boldsymbol{w})}{b^d_{\boldsymbol{\alpha}}(\frac{\boldsymbol{\alpha}}{d})} = \prod_i \left(\frac{dw_i}{\alpha_i}\right)^{\alpha_i}$, which is normalized by the maximum value that the Bernstein basis polynomial can take under the function $\boldsymbol{w} \mapsto \prod_i (w_i)^{\alpha_i}$.

We model $\tilde{P}_{\boldsymbol{\alpha}}$ by trying to strike the right balance between maximizing expressivity and minimizing space complexity. If we take Eq. 2 as is, then the number of $\tilde{P}_{\boldsymbol{\alpha}}$ parameters is exorbitantly high and scales as $\mathcal{O}\left(|\theta| \cdot \binom{N+d-1}{d}\right) = \mathcal{O}\left(|\theta| \cdot d^{N-1}\right)$. First, we restrict $\boldsymbol{\alpha} \in \{0,1\}^N$ to shrink the per-degree basis from $\binom{N+d-1}{d}$ to $\binom{N}{d}$, eliminating self- and cross-interaction terms (e.g., w_i^2 and $w_i^2 w_j$) that do not add considerable expressivity beyond the linear term (i.e., $\sum_i w_i \theta_i$) and multilinear terms (e.g., $w_i w_j$). As a consequence, the highest interaction order is capped at N. Second, given this ceiling, we choose which interaction orders we require from $\{2, ..., N\}$ instead of operating with a fixed degree d. Third, we share $\tilde{P}_{\boldsymbol{\alpha}}$ for all interaction terms within a single degree. These decisions not only collapse the parameter count to $\mathcal{O}\left(|\theta| \cdot |\mathcal{D}|\right)$, independent of d, but also impose a permutation symmetry across vertex models at each interaction order (Fig. 2c). This symmetry crucially encodes the prior that the *order* of an interaction matters but the *identity* of which vertices participate at that order does not. Further discussion of these design choices is included in Supp. Sec. C.1.

Taken together, we turn Eq. 2 into:

$$\theta(\boldsymbol{w}) = \sum_{i=1}^{N} w_i \theta_i + \sum_{r \in \mathcal{D}} \sum_{\substack{\boldsymbol{\alpha} \in \{0,1\}^N \\ |\boldsymbol{\alpha}| = r}} \tilde{b}^r_{\boldsymbol{\alpha}}(\boldsymbol{w})\, \tilde{P}_r. \tag{3}$$

Geometrically, this construction iterates through faces of the simplex by dimension: each $r \in \mathcal{D}$ contributes a correction supported on the $(r-1)$-faces of Δ^{N-1}. Therefore, $\mathcal{D}$ ultimately selects which face dimensions carry learnable structure. For an N-vertex Bézier manifold, we found that $\mathcal{D} = \{2, N-1, N\}$ was sufficient to capture the salient interactions between all vertices as this included both the dominant and highest-order interactions. Because $|\mathcal{D}| \leq 3$, the parameter count ultimately reduces to $\mathcal{O}\left(|\theta|\right)$, independent of both d and N. In-depth ablations of each $r \in \mathcal{D}$ are included in Supp. Sec. C.2.

2.4 Algorithm

In practice, we augment the parameter map of (3) with one learnable scalar gate per degree, $g_r \in \mathbb{R}$ for $r \in \mathcal{D}$, that multiplicatively rescales each correction:

$$\theta(\boldsymbol{w}) = \sum_{i=1}^{N} w_i\, \theta_i + \sum_{r \in \mathcal{D}} e^{g_r} \sum_{\substack{\boldsymbol{\alpha} \in \{0,1\}^N \\ |\boldsymbol{\alpha}| = r}} \tilde{b}^r_{\boldsymbol{\alpha}}(\boldsymbol{w})\, \tilde{P}_r. \tag{4}$$

Gates are initialized to 0 (so $e^{g_r} = 1$) and provide a small, cheap handle for calibrating each correction's magnitude once the correction for a degree has been learned. We train *Nimbus* in two phases: Phase 1 fits the correction terms $\{\tilde{P}_r\}$ while Phase 2 freezes these corrections and tunes only the gates. The original $\{\theta_i\}$ always remain frozen. The full pseudocode is outlined in Supp. Sec. B, but we include a summary here:

Phase 1: low-to-high degree hierarchical training. We take inspiration from Kobayashi et al. [30], which proposes fitting optimization problems on a simplex from lower faces to higher faces. We therefore fit our correction terms in increasing order of degree. Additionally, when training a particular correction, the gates of all previous degrees are unfrozen for joint refinement. Each step draws one minibatch per task and samples a small subset of degree-r faces. On each face, we sample a set of $\boldsymbol{w}$ from a $50:50$ mixture of $\mathrm{Dirichlet}(r \cdot \mathbf{1}_r)$ and $\mathrm{Dirichlet}(\mathbf{1}_r)$. The first term puts its mass at each face's center (where the interaction we are correcting has the highest mass), while the second term spreads samples across the rest of the face, including near its lower-dimensional edges. Weighting the $\mathrm{Dirichlet}$ terms equally trains each degree across its entire face, ensuring correction tensors neither overfit the face center nor ignore the boundaries where they must blend smoothly into fixed lower-order edge residuals. On each sampled $\theta\left(\boldsymbol{w}\right)$, we minimize a cross-entropy loss on the combined labels from the tasks on the face, plus a KL-divergence to each task's vertex model evaluated on that task's own data.

Phase 2: gate calibration. We freeze all correction terms and only train the gates $\{g_r\}$. We sample a set of $\boldsymbol{w}$ uniformly across degrees $r \in \mathcal{D}$ and uniformly among all faces within each degree. On the pooled data of all tasks across the sampled faces, we minimize the mean (i) cross-entropy of the averaged logits, (ii) NLL of the averaged probabilities, and (iii) KL-divergence to each task's vertex model evaluated on that task's own data. We ablate these design choices in Supp. Sec. C.1 and conclude that *Nimbus*'s shared-by-degree corrections (covering pairwise and

highest-order interactions), KL distillation losses, and Phase 2 training all contribute to a uniformly low-loss manifold.

3 Experimental results

3.1 Tasks and architectures

We evaluate *Nimbus* across both vision and language modalities.

ConvNeXt. For vision tasks, we train separate, randomly initialized ConvNeXt models [31] on subsets of CIFAR-100 [32], where each task consists of 10 disjoint classes selected under a fixed random permutation across $10N$ total classes ($N \in \{3, 5, 7, 10\}$). All models have a $10N$-node output head. The $|\mathcal{D}| \leq 3$ correction tensors $\tilde{P}_r$ use the same architecture.

ViT-B/32. We additionally consider the ViT-B/32 vision encoder from OpenAI CLIP [33], a large-scale vision transformer (32×32 patch size, 224×224 input image size, and approximately 87.5M parameters) that was jointly pre-trained with a transformer text encoder on roughly 400M image-text pairs. We fine-tune separate, pre-trained ViT-B/32 models via LoRA [34] (rank$= 64$, applied only to the attention projections and MLPs) on $N = 8$ vision tasks discussed in Task Arithmetic [24], which include MNIST (digits), SVHN (digits), GTSRB (traffic signs), DTD (perceptual textures), EuroSAT (satellite imagery), Cars, RESISC45 (satellite imagery), and SUN397 (naturalistic scenes). The $|\mathcal{D}| = 3$ correction tensors $\tilde{P}_r$ use the same architecture and are optimized similarly with LoRA under the same setting. To perform zero-shot classification, we replace the final linear layer with class embeddings generated by CLIP's text encoder using task-specific prompt templates. Given an input image, the vision encoder extracts a feature vector, and the model assigns the class label whose normalized text embedding (512-dim) yields the highest cosine similarity with the image feature.

nanoGPT. For language, we train separate, randomly initialized nanoGPT models [35] on $N = 7$ different pieces of text, which include Early Modern English (Shakespeare, the KJV, Milton, Jonson, and other in-period authors) from Project Gutenberg [36], WikiText-103 [37], TinyStories [38], Python source code (CPython stdlib [39] and CodeParrot [40]), arXiv papers [41], U.S. legal text (Pile of Law [42], English-only subsets), and French/German literature (Project Gutenberg [36]). All domains are tokenized with a shared 2048-token BPE vocabulary (2048-node output head) fit on a concatenation of the corpora. The $|\mathcal{D}| = 3$ correction tensors $\tilde{P}_r$ use the same architecture.

Hyperparameters, experimental settings, and code are given in Supp. Sec. A.1.

3.2 *Nimbus* admits flat functional manifolds

We begin by considering *Nimbus* instantiated as ConvNeXt ($N = 3$, $\mathcal{D} = \{2, 3\}$) and visualize the loss manifold induced by *Nimbus* over Δ^2, where vertices of the triangle are given by the 3 task specialists and mixing weight $\boldsymbol{w} \in \Delta^2$ refers to points in this triangle. Fig. 3a demonstrates that naive linear interpolation across these specialists encounters high-loss barriers, while *Nimbus* is uniformly low in NLL loss despite models starting from different initializations and learning entirely different tasks. As $\boldsymbol{w}$ approaches a particular vertex, we also recover that vertex and its task performance through our parameter map $\theta(\boldsymbol{w})$.

Remarkably, building manifolds through *Nimbus* demonstrates that it suffices to move a small synaptic distance from any vertex towards the centroid of the manifold in order to access an *infinite* set of competent solutions. We can validate this theory by tracing and averaging the loss along N different "rays", each starting from a vertex and moving toward the manifold centroid (see Fig. 3b). Distance along each ray is measured in Euclidean parameter space and normalized by the mean pairwise inter-expert distance. While the multi-task NLL loss near any one expert is ≈ 10.0 (≈ 3.4 is chance), moving roughly half an inter-expert distance along *Nimbus*'s manifold collapses the loss to ≈ 1.0. The average loss across these rays then stays in a broad, flat low-loss basin all the way across the interior. By contrast, the linear weight-average crosses many high-loss barriers, reaching a multi-task NLL of ≈ 4.0 near the interpolation centroid.

Fig. 3c demonstrates functional flatness through uniformly high top-3 and top-5 accuracy and uniformly low ECE, indicating that any convex combination of specialists (plus learned corrections)

yields a competent merged multi-task model. Even when *Nimbus* is optimized over $N = 4$ specialists, the low-loss volume visualized as a Δ^3 tetrahedron is bounded by lower-dimensional surfaces, each of which effectively merges subsets of specialist models (see Fig. 3d). Thus, faces of the tetrahedron correspond to unique sets of $N = 3$ tasks that are also uniformly low in NLL loss. Despite a simplex face count that grows exponentially (here, $\binom{4}{4} + \binom{4}{3} + \binom{4}{2} + \binom{4}{1} = 15$), we conclude that a small number of correction tensors shared per order ($|\mathcal{D}| = |\{2, 3, 4\}| = 3$) suffices to populate the simplex with competent models.

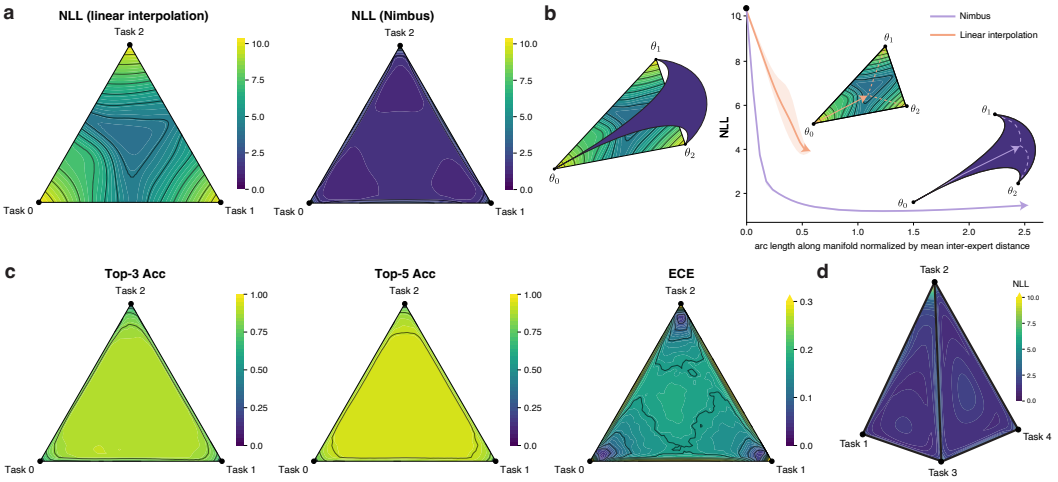


Figure 3: **The Bézier manifold induced by *Nimbus* is flat in loss. a.** Each point in the triangle heatmap corresponds to a mixing weight $\boldsymbol{w} \in \Delta^2$. For ConvNeXt ($N = 3$, $\mathcal{D} = \{2, 3\}$), linear connectivity between 3 disjoint models crosses many high-loss barriers, while *Nimbus* is uniformly low in NLL loss (lower is better). NLL ≈ 3.4 is chance. **b.** We can compare the linear paths towards the vertex average and the paths towards *Nimbus*'s centroid starting from any vertex. After moving roughly half an inter-expert distance, linear interpolation produces poorly functioning models while the path along *Nimbus*'s manifold yields access to an infinite set of highly competent solutions. **c.** Top-3 and top-5 accuracy (higher is better) stay uniformly high across the interior, while ECE (lower is better) stays uniformly low, indicating that any convex combination of specialists (plus learned corrections) yields a competent merged multi-task model. **d.** Even when *Nimbus* is optimized over $N = 4$ specialists, the low-loss Δ^3 tetrahedron is bounded by lower-dimensional surfaces, each of which effectively merges subsets of specialist models. We visualize here two unique sets of $N = 3$ tasks as two faces of the tetrahedron. NLL ≈ 3.7 is chance.

3.3 Ensembling with *Nimbus*

Nimbus's manifold admits a continuum of models, so we can use it for ensembling under both logit and softmax averaging. Under both rules, we fix a sampling budget split approximately equally across $r \in \mathcal{D}$, then uniformly across the faces within each degree, with weights drawn from a Dirichlet over the vertices of each sampled face.

Fig. 4 compares *Nimbus* against a baseline ensemble of $|\mathcal{D}|$ arbitrarily initialized jointly trained models, each trained over all tasks. For simplicity, we report logit ensembling here and include full results (softmax ensembling, raw parameter averaging) in Supp. Sec. F along with further discussion on the differences between the ensembling rules. On logit ensembling, *Nimbus* crucially achieves lower NLL on every task across ConvNeXt ($N = 7$), ViT-B/32 ($N = 8$), and nanoGPT ($N = 7$), and is comparably well-calibrated.

3.4 *Nimbus* demonstrates robustness under input and weight perturbations

Next, we evaluate *Nimbus* under distribution shifts simulated by input and weight perturbations, which probe functional and synaptic weight flatness, respectively.

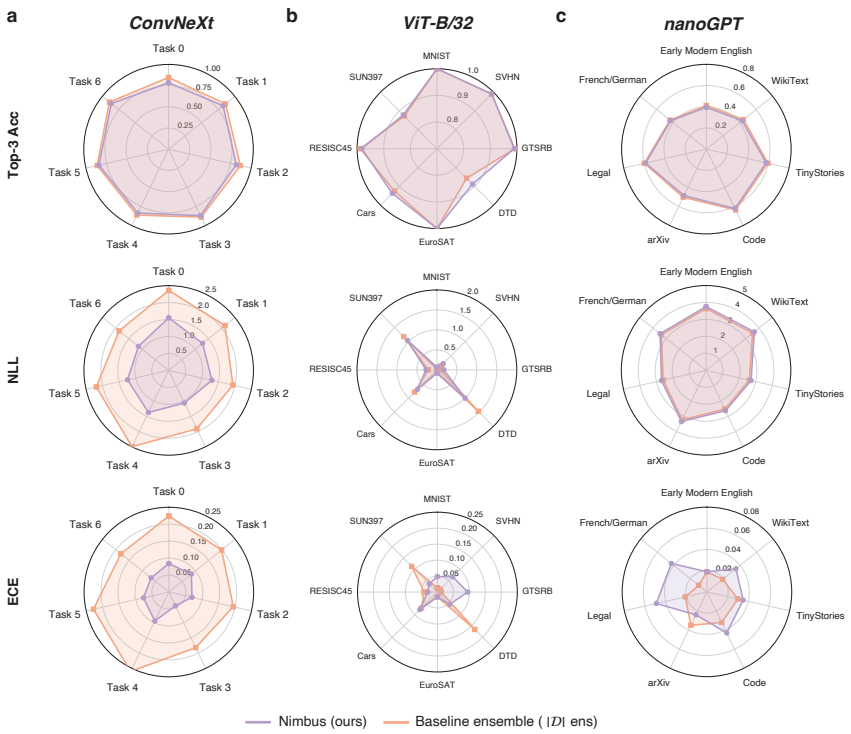


Figure 4: **Per-task metrics for ensembling on *Nimbus*'s manifold versus an ensemble of jointly trained models. a.** ConvNeXt ($N = 7$), **b.** ViT-B/32 ($N = 8$), and **c.** nanoGPT ($N = 7$) per-task metrics such as top-3 accuracy (higher is better), NLL (lower is better), and ECE (lower is better) under logit ensembling for *Nimbus* (purple) versus a baseline ensemble of jointly trained models (orange). *Nimbus* achieves lower NLL and comparable or higher top-3 accuracy than the baseline on every task in all settings and is well-calibrated under logit ensembling.

We compare against three baselines matched to *Nimbus* on both total learnable parameters and total training compute, under progressively more conservative accounting of what a practitioner is responsible for (see Supp. Sec. A.2 for a rigorous treatment of this topic). The first is an ensemble of $|\mathcal{D}|$ models fine-tuned jointly on all tasks, splitting *Nimbus*'s budget evenly among them. The second charges *Nimbus* for the cost of training its N vertices as well, expanding the ensemble to $|\mathcal{D}| + N$ models at the same per-model budget (effectively $\frac{N}{|\mathcal{D}|} \times$ *more* compute). The third assumes the practitioner does not want to ensemble at all and thus trains a single joint model with $2\times$ the compute and $2\times$ the parameters of *Nimbus* (effectively $4\times$ the compute). Additional baselines such as linear interpolation, ASAM, SWA, PCGrad, Fisher Merging, TIES Merging, and Task Arithmetic [29, 28, 26, 13, 15, 24] are included with the full results in Supp. Sec. F; *Nimbus* outperforms each substantially.

Input perturbations. Within the vision modality, we perturb input images with additive Gaussian noise of magnitude ε relative to the ℓ_2-norm of each test image (see Fig. 5a). Within the language modality, we randomly substitute each input token with probability $p \in [0, 1]$ (see Fig. 5b).

Relative to every compute-matched and extra-compute baseline, *Nimbus* instantiated as ConvNeXt and ViT degrades more gracefully, with the largest gaps in NLL and ECE (see Fig. 6a–b). On language modeling, *Nimbus* is comparable to or better than the baselines while achieving superior calibration (see Fig. 6c). *Nimbus* thus produces a more *functionally flat* manifold than baselines, crucially without any explicit optimization for functional flatness.

Weight perturbations. Across all modalities, we perturb model weights with additive Gaussian noise of magnitude ε relative to the ℓ_2-norm of the full model. We again compare *Nimbus* against every compute-matched and extra-compute baseline, and find that on ConvNeXt and nanoGPT, *Nimbus* degrades significantly more gracefully, with the largest gaps in NLL and ECE (see Fig. 7a, c).

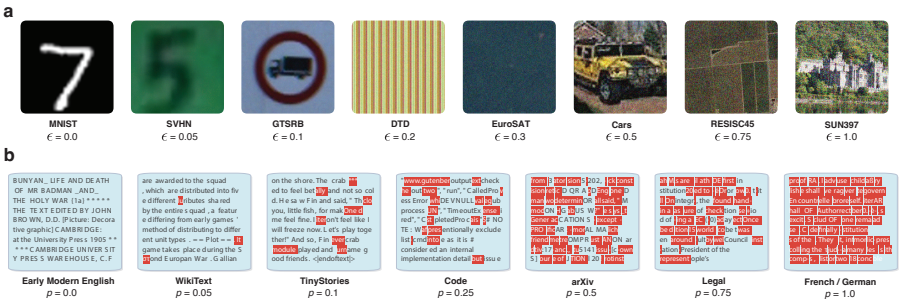


Figure 5: **Input corruption for vision and language experiments. a.** Sample image tasks for ViT-B/32 experiments. Gaussian noise of magnitude $\varepsilon \in \{0.0, 0.05, 0.1, 0.2, 0.3, 0.5, 0.75, 1.0\}$ relative to each image's ℓ_2-norm is added. **b.** Sample text tasks for nanoGPT experiments. Input tokens are independently replaced with a uniform random token from the BPE vocabulary with probability $p \in \{0.0, 0.05, 0.1, 0.25, 0.5, 0.75, 1.0\}$. Corrupted tokens are highlighted in red.

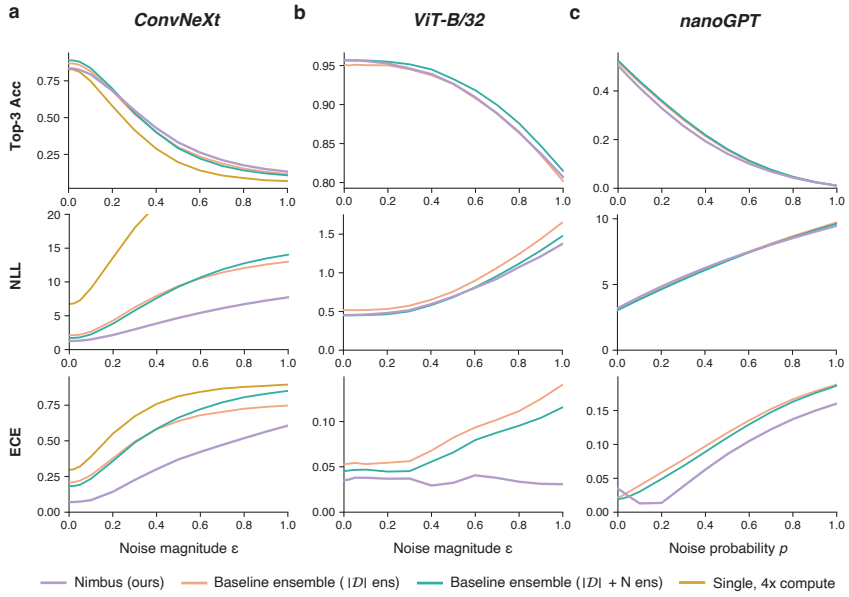


Figure 6: ***Nimbus* is more robust to input perturbations than baselines. a.** ConvNeXt ($N = 7$), **b.** ViT-B/32 ($N = 8$), and **c.** nanoGPT ($N = 7$) robustness to input perturbations under logit ensembling for metrics such as top-3 accuracy (higher is better), NLL (lower is better), and ECE (lower is better). We compare between *Nimbus* (purple) versus a baseline ensemble of $|\mathcal{D}|$ jointly trained models (orange) and $|\mathcal{D}| + N$ jointly trained models (green), and versus a single jointly trained model with $4\times$ the effective compute (gold) as *Nimbus* (only computationally feasible to run for ConvNeXt). *Nimbus* finds a more *functionally flat* manifold than baselines.

Nimbus even outperforms ASAM, which is explicitly optimized for finding weight-flat minima (see Supp. Sec. F). On ViT-B/32, *Nimbus* is comparable to or better than the baselines for varying levels of weight noise (see Fig. 7b).

We include a sensitivity analysis of our input and weight robustness experiments in Supp. Sec. E.

4 Discussion and future work

***Nimbus* as a manifold learning framework for merging and ensembling.** Where prior connectivity work links specialists by one-dimensional paths or two-dimensional sheets between a handful of same-objective networks, *Nimbus* shows that an $(N-1)$-dimensional manifold of uniformly low loss exists between N *disjointly* trained ones, effectively reframing model merging as a traversable

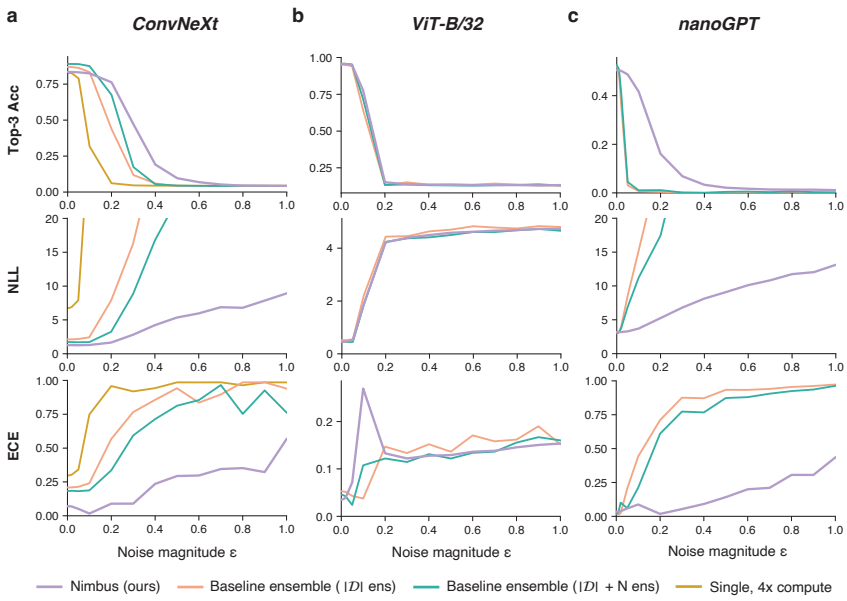


Figure 7: ***Nimbus* is more robust to weight perturbations than baselines. a.** ConvNeXt ($N = 7$), **b.** ViT-B/32 ($N = 8$), and **c.** nanoGPT ($N = 7$) robustness to weight perturbations under logit ensembling for metrics such as top-3 accuracy (higher is better), NLL (lower is better), and ECE (lower is better). Weight perturbations are additive Gaussian noise ε relative to the norm of the model's own weights. We compare between *Nimbus* (purple) versus a baseline ensemble of $|\mathcal{D}|$ jointly trained models (orange) and $|\mathcal{D}| + N$ jointly trained models (green), and versus a single jointly trained model with $4\times$ the effective compute (gold) as *Nimbus* (only computationally feasible to run for ConvNeXt). *Nimbus* finds a more *weight-flat* manifold than baselines.

geometry instead of producing a single solution. As a consequence, the explicit manifolds that *Nimbus* builds are also superior for robustness through arbitrarily large ensembles. As we show, our method naturally extends to post-training of transformer models. Since fine-tuned specialists share a pre-training recipe, the linear sum $\sum_i w_i \theta_i$ naturally lies close to a competent initialization, so the correction terms need only repair residual interference rather than resolve systemic neuron asymmetries. A potential future direction is utilizing only the KL term in Algorithm 1's objective; eliminating the cross-entropy weight would test how much of the manifold's behavior can be carried through distillation alone, extending *Nimbus* to regimes where source labels are scarce or unavailable.

A neural reading of the framework. The vertices $\{\theta_i\}$ map naturally onto specialized neural pathways. Biology realizes flexible degeneracy through localized mechanisms such as dendritic compartmentalization [8, 7], synaptic clustering [9–11], and the activation of silent synapses [12], all of which adapt circuit computation without overwriting existing pathways. *Nimbus* approaches this problem through a deep learning lens. Just as specialized circuits undergo structural changes to accumulate new capabilities, specialist models at the vertices of *Nimbus*'s manifold move away from boundaries to smoothly accumulate new task competencies. Since a continuous family of models lies on the high-dimensional manifold induced by *Nimbus*, a small synaptic perturbation can reach equally competent configurations with a minimal metabolic cost of synaptic change. Our work fundamentally unifies concepts from neural degeneracy with mode connectivity and model merging, yielding a deep learning framework for flexible degeneracy. This flexible degeneracy appears in both vision and language tasks, as *Nimbus* is both functionally and locally weight-flat, with nearby points on the high-dimensional manifold close in parameter space and matched in behavior.

Limitations. While *Nimbus* is quite performant in a within-modality heterogeneous setting (ViT fine-tuning tasks are visually unrelated domains), we have yet to test *Nimbus* in a truly cross-modal setting, such as across language and vision simultaneously. Additionally, we are still searching for a core theoretical explanation of why or when constructing such a *Nimbus* manifold is possible among arbitrary models.

Acknowledgements

AI would like to thank Cynthia Rais, Raul Mojica Soto-Albors, Courtney Yaeger, and Akhilan Boopathy for invaluable help in figure making and for very useful scientific discussions. AI was supported by the MathWorks Graduate Research Fellowship. SC acknowledges support of the Department of Atomic Energy, Government of India, under Project No. RTI4001, the Murty Science Fellowship, and the Centre for Artificial Learning and Intelligence for Biological Research and Education, ICTS-TIFR.

References

[1] Giulio Tononi, Olaf Sporns, and Gerald M Edelman. Measures of degeneracy and redundancy in biological networks. *Proceedings of the National Academy of Sciences*, 96(6):3257–3262, 1999.

[2] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

[3] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

[4] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International conference on machine learning*, pages 3259–3269. PMLR, 2020.

[5] Jie Ren, Pin-Yu Chen, and Ren Wang. Revisiting mode connectivity in neural networks with bezier surface. In *ICLR*, 2025.

[6] Gregory Benton, Wesley Maddox, Sanae Lotfi, and Andrew Gordon Gordon Wilson. Loss surface simplexes for mode connecting volumes and fast ensembling. In *International Conference on Machine Learning*, pages 769–779. PMLR, 2021.

[7] Jeffrey C Magee and Christine Grienberger. Synaptic plasticity forms and functions. *Annual review of neuroscience*, 43(1):95–117, 2020.

[8] Yara Otor, Shay Achvat, Nathan Cermak, Hadas Benisty, Maisan Abboud, Omri Barak, Yitzhak Schiller, Alon Poleg-Polsky, and Jackie Schiller. Dynamic compartmental computations in tuft dendrites of layer 5 neurons during motor behavior. *Science*, 376(6590):267–275, 2022.

[9] Jens P Weber, Bertalan K Andrásfalvy, Marina Polito, Ádám Magó, Balázs B Ujfalussy, and Judit K Makara. Location-dependent synaptic plasticity rules by dendritic spine cooperativity. *Nature communications*, 7(1):11380, 2016.

[10] Nathan G Hedrick, Zhongmin Lu, Eric Bushong, Surbhi Singhi, Peter Nguyen, Yessenia Magaña, Sayyed Jilani, Byung Kook Lim, Mark Ellisman, and Takaki Komiyama. Learning binds new inputs into functional synaptic clusters via spinogenesis. *Nature neuroscience*, 25(6): 726–737, 2022.

[11] Nathan G Hedrick, William J Wright, and Takaki Komiyama. Local and global predictors of synapse elimination during motor learning. *Science Advances*, 10(11):eadk0540, 2024.

[12] Dimitra Vardalaki, Courtney E Yaeger, and Mark T Harnett. Silent synapses in the adult brain. *Annual Review of Neuroscience*, 48, 2025.

[13] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

[14] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.

[15] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

[16] George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. *arXiv preprint arXiv:2305.03053*, 2023.

[17] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*, 2023.

[18] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[19] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

[20] Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

[21] Quynh Nguyen. On connected sublevel sets in deep learning. In *International conference on machine learning*, pages 4790–4799. PMLR, 2019.

[22] Zhanpeng Zhou, Yongyi Yang, Xiaojiang Yang, Junchi Yan, and Wei Hu. Going beyond linear mode connectivity: The layerwise linear feature connectivity. *Advances in neural information processing systems*, 36:60853–60877, 2023.

[23] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[24] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[25] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

[26] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020.

[27] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[28] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[29] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International conference on machine learning*, pages 5905–5914. PMLR, 2021.

[30] Ken Kobayashi, Naoki Hamada, Akiyoshi Sannai, Akinori Tanaka, Kenichi Bannai, and Masashi Sugiyama. Bézier simplex fitting: Describing pareto fronts of simplicial problems with small samples in multi-objective optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2304–2313, 2019.

[31] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[32] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/radford21a.html`.

[34] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[35] Andrej Karpathy. NanoGPT. `https://github.com/karpathy/nanoGPT`, 2022.

[36] Project Gutenberg. Project Gutenberg. `https://www.gutenberg.org`, 2026. Accessed: May 2, 2026.

[37] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[38] Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

[39] Python Software Foundation. Cpython, 2026. URL `https://github.com/python/cpython`. Accessed: 2026-05-02.

[40] CodeParrot (Hugging Face). CodeParrot: A cleaned Python code dataset. `https://huggingface.co/datasets/codeparrot/codeparrot-clean`, 2022. Accessed: 2026-05-02.

[41] Jun He, Liqun Wang, Liu Liu, Jiao Feng, and Hao Wu. Long document classification from local word glimpses via recurrent attention learning. *IEEE Access*, 7:40707–40718, 2019. doi: 10.1109/ACCESS.2019.2907992.

[42] Peter Henderson, Mark S. Krass, Lucia Zheng, Neel Guha, Christopher D. Manning, Dan Jurafsky, and Daniel E. Ho. Pile of law: Learning responsible data filtering from the law and a 256GB open-source legal dataset. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 29217–29234, 2022.

A Experimental details

A.1 Hyperparameters

We include all the relevant hyperparameters from our experiments in Table 1. Our code can be found at `https://github.com/abhi-iyer/nimbus`. All experiments were run on a single NVIDIA RTX 4090 GPU; in total, all experiments took about 10–12 hours to run.

	CIFAR-100	**ViT**	**Language modeling**
Architecture			
Backbone	SmallConvNeXt	CLIP ViT-B/32	nanoGPT
Layers / depth	depths $(1, 1, 3)$	12 blocks	$n_{\text{layer}} = 4$
Width	dims $(16, 48, 192)$	768	$n_{\text{embed}} = 128$
Heads / expansion	expand $(2, 2, 3)$	12 heads	$n_{\text{head}} = 4$
MLP	—	3072	$4\times$
Patch / block size	—	patch 32, image 224	block 256
Activation	SiLU	QuickGELU	GELU
Dropout	—	0	0.1
Vocab / classes	$10N$	zero-shot text heads (512-dim output)	2048
Tokenizer	—	—	byte-level BPE
Vertex training (given as input to Algorithm 1)			
Iters / epochs	100 epochs	16k steps (early-stop)	100k iters (early-stop)
Batch (per task)	256	64	64
Optimizer	AdamW $(0.9, 0.999)$	AdamW $(0.9, 0.95)$	AdamW $(0.9, 0.95)$
Weight decay	0	0	0
LR / schedule	8e-3, OneCycleLR (pct_start 0.2)	1e-4, warmup 100 $\rightarrow$ const	6e-4, warmup 200 $\rightarrow$ const
Vertex form	full model	rank-64 LoRA (on frozen CLIP)	full model
Phase 1 – training correction tensors			
Iters / epochs (per $r \in \mathcal{D}$)	100 epochs	1000 iters	6000 iters
Batch (per task)	256	64	64
Optimizer	AdamW $(0.9, 0.999)$	AdamW $(0.9, 0.95)$	AdamW $(0.9, 0.95)$
LR η_1 (constant)	8e-3	5e-5	8e-3
Weight decay	0	0	0
Correction tensor form	dense	rank-64 LoRA + dense	dense
Phase 2 – gate calibration			
Iters / epochs	50 epochs	300 iters	1000 iters
Batch	256	64	64
Optimizer	AdamW $(0.9, 0.999)$	AdamW $(0.9, 0.95)$	AdamW $(0.9, 0.95)$
LR η_2 (constant)	1e-3	2e-3	2e-3
Samples / step S	5	4	4
Evaluation			
Total $\boldsymbol{w}$ samples	100	100	100
Trainable parameters (M)			
Nimbus	$1.46/2.19/2.21/2.22$	39.4 (126.9 total)	3.25
Baseline ensemble ($\lvert\mathcal{D}\rvert$)	$1.46/2.19/2.21/2.22$	39.4 (126.9 total)	3.25
Baseline ensemble ($\lvert\mathcal{D}\rvert{+}N$)	$3.64/5.85/7.35/9.63$	144.5 (232.0 total)	10.8
Single, $4\times$ compute	$2.88/4.32/4.33/4.34$	—	—
Phase-1 sample-passes (M)			
Nimbus	$12.2/47.3/85.1/162$	15.36	72.6
Baseline ensemble ($\lvert\mathcal{D}\rvert$)	$12.2/47.3/85.1/162$	15.36	72.6
Baseline ensemble ($\lvert\mathcal{D}\rvert{+}N$)	$30.4/126/284/702$	56.3	241.9
Single, $4\times$ compute	$24.3/94.5/170.1/324$	—	—
Tasks and data			

continued on next page

Table 1 (continued)

	CIFAR-100	**ViT**	**Language modeling**
Source	CIFAR-100	8 tasks from Task Arithmetic [24]	7 text domains
$(N,\ \mathcal{D})$ pairs	N=3: $\{2, 3\}$ N=5: $\{2, 4, 5\}$ N=7: $\{2, 6, 7\}$ N=10: $\{2, 9, 10\}$	N=8: $\{2, 7, 8\}$	N=7: $\{2, 6, 7\}$
Per-task budget	10 classes	$\leq$ 50k images	20M tokens (60M chars)
Augmentation	reflect-pad 4, random 32 crop, h-flip (p=0.5)	CLIP preprocess (resize 224, normalize)	none
LoRA rank r	—	64	—
Random seed	42	42	42

Table 1: *Nimbus* hyperparameters, parameters, and compute across all three experiments.

A.2 Compute-matched baselines

As a first-order approximation, *Nimbus* requires:

$$\text{(passes per training step)} \cdot \text{(epochs or iters / degree)} \cdot \text{(batch size)} \sum_{r \in \mathcal{D}} \min\left(\binom{N}{r}, N\right) \cdot r$$

"sample-passes" (training samples $\times$ passes). Following standard FLOP counting estimates, we define the "passes per training step" as 3 (wherein we approximate each forward through a network as 1 "pass" and backward as 2 "passes"). The binomial term measures the number of faces per degree r that we wish to sample from, while the second r term measures the passes per task per face (i.e., a degree-r face joins r task-specialized models).

For example, if our problem setting is ViT-B/32 with $N = 8$, $\mathcal{D} = \{2, 7, 8\}$, iters / degree $= 1000$, and batch size $= 64$, then this reduces to ≈ 15.36M sample-passes. We only include the sample-passes consumed by the degree-by-degree training phase, as the calibration phase is relatively much cheaper. Additionally, *Nimbus* with ViT-B/32 yields approximately 39.4M learnable parameters. Since $|\mathcal{D}| = 3$, an equitable baseline fine-tunes 3 different pre-trained ViT-B/32 base models jointly on all the tasks and ensembles over them. The LoRA rank is likewise 64 (total learnable parameters across baseline models are equal to the total parameter budget of *Nimbus*, which is 39.4M). The number of iterations is picked such that the total number of sample-passes across baseline models is 15.36M (each baseline model receives 5.12M sample-passes). We call this compute-matched baseline the "baseline ensemble" with $|\mathcal{D}|$ member models.

If we suppose that a practitioner employing *Nimbus* must also assume responsibility for the compute in training the vertices, we also include a "baseline ensemble" with $|\mathcal{D}| + N$ member models (each trained with the same number of sample-passes as the original "baseline ensemble") to liberally account for this extra compute (effectively $\frac{N}{|\mathcal{D}|} \times$ *more* compute). And if we further suppose that a practitioner does not want to ensemble at all, we also include a single baseline model that is jointly trained on all the tasks with $2\times$ the sample-passes and $2\times$ the total learnable parameters of *Nimbus* (effectively $4\times$ the compute).

B Pseudocode

Algorithm 1 ***Nimbus***: N-expert Interpolating Manifold: Bezier over an Underlying Simplex

Require: Vertex models $\{\theta_i\}_{i=1}^N$, datasets $\{\mathcal{T}_i\}$, degree set $\mathcal{D} \subseteq \{2, \dots, N\}$, learning rates η_1, η_2, iteration budgets T_1, T_2, calibration sample count S.

Initialize $\tilde{P}_r \leftarrow \mathbf{0}$ and $g_r \leftarrow 0$ for all $r \in \mathcal{D}$.

Phase 1: low-to-high degree hierarchical training

for $r \in \mathcal{D}$ in increasing order **do**
  Make $\tilde{P}_r$ and $\{g_{r'} : r' < r,\ r' \in \mathcal{D}\}$ trainable; freeze the rest (g_r stays at 0).
  for $t = 1, \dots, T_1$ **do**
    Draw $(x_i, y_i) \sim \mathcal{T}_i$ for each $i = 1, \dots, N$.
    Sample $\min(N, \binom{N}{r})$ multi-indices $\{\boldsymbol{\alpha}\}$ of degree r uniformly without replacement.
    for each sampled $\boldsymbol{\alpha}$ with support $F = \{i : \alpha_i = 1\}$ **do**
      Sample $\boldsymbol{w}$ on face F; let $x_F, y_F = \mathrm{concat}_{i \in F}(x_i, y_i)$.
      $\mathcal{L}_{\boldsymbol{\alpha}} \leftarrow \mathrm{CE}\big(f_{\theta(\boldsymbol{w})}(x_F),\, y_F\big) + \mathrm{KL}\big(\sigma(f_{\theta(\boldsymbol{w})}(x_F)) \,\big\|\, \sigma(\mathrm{concat}_{i \in F} f_{\theta_i}(x_i))\big)$.
    end for
    One AdamW step at LR η_1 on $\mathbb{E}_{\boldsymbol{\alpha}}[\mathcal{L}_{\boldsymbol{\alpha}}]$.
  end for
end for

Phase 2: gate calibration

Freeze all $\tilde{P}_r$; make every gate $\{g_r\}_{r \in \mathcal{D}}$ trainable.
for $t = 1, \dots, T_2$ **do**
  Draw $(x_i, y_i) \sim \mathcal{T}_i$ for each i; set $x, y = \mathrm{concat}_i(x_i, y_i)$.
  Sample S multi-indices $\{\boldsymbol{\alpha}_s\}$ from $\bigcup_{r \in \mathcal{D}} \{\boldsymbol{\alpha} \in \{0,1\}^N : |\boldsymbol{\alpha}| = r\}$, equal weight per degree and uniform within degree.
  For each s: sample $\boldsymbol{w}_s$ on face F_s and set $\ell_s \leftarrow f_{\theta(\boldsymbol{w}_s)}(x)$.
  $\mathcal{L}_{\mathrm{NLL}} \leftarrow \frac{1}{2}\big[\mathrm{CE}\big(\frac{1}{S}\sum_s \ell_s,\, y\big) - \frac{1}{|y|}\sum_j \log\big(\frac{1}{S}\sum_s \sigma(\ell_s)\big)_{j,y_j}\big]$.
  for $s = 1, \dots, S$ **do**
    Let $\ell_s^{F_s}$ be the rows of ℓ_s for tasks in F_s.
    $\mathcal{K}_s \leftarrow \mathrm{KL}\big(\sigma(\ell_s^{F_s}) \,\big\|\, \sigma(\mathrm{concat}_{i \in F_s} f_{\theta_i}(x_i))\big)$.
  end for
  One AdamW step at LR η_2 on $\mathcal{L}_{\mathrm{NLL}} + \frac{1}{S}\sum_s \mathcal{K}_s$ (gates only).
end for

return $\{\tilde{P}_r,\, g_r\}_{r \in \mathcal{D}}$.

C Ablations

C.1 Algorithm ablations

Results. Table 2 ablates the main design choices in *Nimbus* for the ConvNeXt, $N = 3$ setting. The full model achieves the lowest mean and worst-case NLL, indicating that both pairwise and highest-order corrections are needed to fill the simplex without leaving high-loss regions. Pairwise-only corrections ($\mathcal{D} = \{2\}$) substantially degrade mean NLL, suggesting that edge-level repairs alone are insufficient for the simplex interior. Conversely, the interior-only correction ($\mathcal{D} = \{3\}$) attains reasonable average NLL but has a very large worst-case NLL, consistent with the absence of learned structure on lower-dimensional faces. Surprisingly, separate per-face corrections (no sharing of correction tensors within a degree) perform worse than the shared per-degree parameterization despite being strictly more expressive. Additionally, Phase 2 crucially improves mean NLL while KL improves worst-case stability.

Why shared correction terms? Without shared correction terms (i.e., unique correction tensors for every face at every degree d), the number of learnable tensors would scale as $\mathcal{O}(d^{N-1})$ and

become computationally prohibitive to optimize for any reasonable choice of N. Additionally, shared correction terms seem to act as a useful regularizer, as per-face corrections are a more ill-posed optimization problem within our hierarchical training recipe. Thus, the shared correction tensors across all faces of a particular degree emerged as a natural choice to operationalize *Nimbus*.

Why does the interaction order appear to matter more than the identities of the participating specialists? Because we are primarily interested in ensembling models along the N-th order Bézier surface (the full surface constructed by all vertex models), we hypothesize that the main direction we wish to deform such a surface is described by the highest-order N-th term (which can be visualized as tugging the surface from its centroid). Working backwards, if we wish to add important error correction terms (corrective tensors whose order is strictly less than N), then the easiest way to do so is to add one corrective tensor per order, in which case such a correction is shared among all faces within an order. We suspect that the interaction order appears to matter more than the identities of the vertices because the Bézier control tensors ($\tilde{P}_r$) are only intended to *correct* the N-th order surface around the local neighborhood of the centroid.

Overall, these ablations support the full *Nimbus* design: shared pairwise and highest-order corrections, stabilized by distillation and lightweight gate calibration, produce the most uniformly low-loss manifold. Robustness ablations follow the same trend across all variants as these no-perturbation ablations and are omitted for succinctness.

Variant	Mean NLL	Worst NLL
Full *Nimbus*	**1.3040**	**1.7548**
$\mathcal{D} = \{2\}$	1.8950	2.6937
$\mathcal{D} = \{3\}$	1.7591	6.2718
Nimbus without degree-sharing	1.5013	3.5682
No KL distillation	1.3680	2.1651
No Phase 2	1.4803	2.0609

Table 2: Ablation study for ConvNeXt, $N = 3$. We report mean and worst-case NLL (lower is better) by sampling over the interior points of the simplex. The full *Nimbus* model uses both pairwise and highest-order corrections, KL distillation, and Phase 2 gate calibration.

C.2 Interaction order ablations

Another important aspect of scaling is to consider how many corrective tensors *Nimbus* requires as we consider an increasing number of specialist models. We try to rigorously identify which corrections are needed for optimal performance of *Nimbus* by ablating the choices for $\mathcal{D}$ for ConvNeXt, $N = 7$.

We uniformly sample over the constructed manifold for various choices of degrees (possible choices being subsets of $\{2, ..., 7\}$) and determine the multi-task loss (see Supp. Fig. 8a). We also investigate how the multi-task NLL appears under logit vs. softmax ensembling for the same choices of degrees (see Supp. Fig. 8b).

We see that $\mathcal{D} = \{2, 6, 7\}$ is clearly the optimal choice in terms of both the quality of the NLL loss distribution (tight spread around the mean) and space complexity (number of learnable parameters that scale with $|\mathcal{D}|$). Additionally, the gap between logit and softmax ensembling reduces significantly under our core hypothesis (lower degrees coupled with a few higher-order degrees are sufficient).

We also showcase *Nimbus*'s gentle NLL loss degradation under both logit and softmax ensembling for each choice of $\mathcal{D}$ (see Supp. Fig. 8c–f). $\mathcal{D} = \{2, 6, 7\}$ is optimal across all robustness curves, as it (1) starts off with the lowest NLL loss, (2) is the most robust against small-magnitude noise, and (3) degrades more gracefully with higher-magnitude noise at a fraction of the total learnable parameters (relative to $\{2, 5, 6, 7\}$ or $\{2, 3, 4, 5, 6, 7\}$).

We ran a similar ablation for ConvNeXt, $N \in \{3, 5, 10\}$ (not included for succinctness) and concluded similarly that $\mathcal{D} = \{2, N - 1, N\}$ is optimal when jointly considering uniformly low NLL loss, space complexity, and robustness.

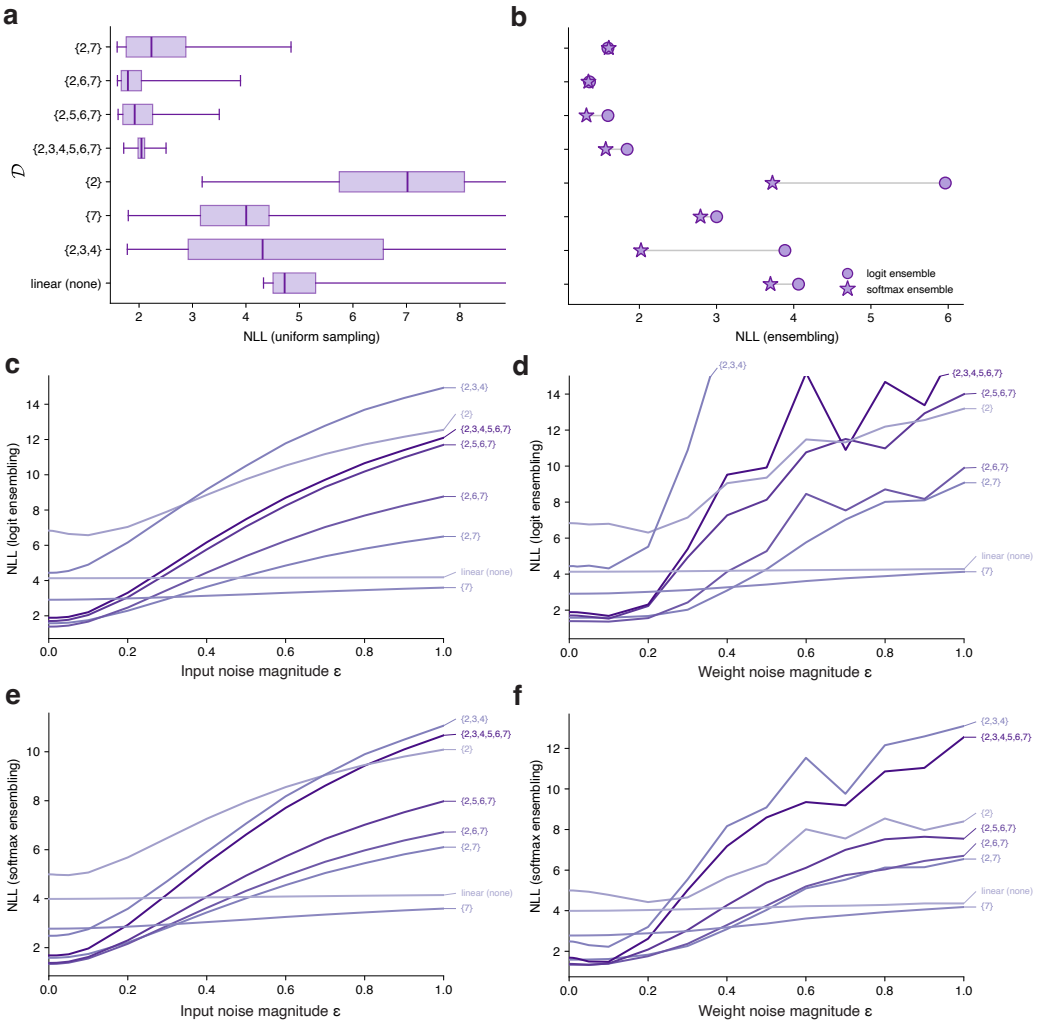


Figure 8: **Ablating the choice of interaction orders in *Nimbus*. a.** Uniformly sampled multi-task NLL loss for various choices of $\mathcal{D}$. **b.** Under the same choices of $\mathcal{D}$, the gap between logit and softmax ensembling. **c, d.** Robustness to input and weight perturbations, respectively, under logit ensembling. **e, f.** Robustness to input and weight perturbations, respectively, under softmax ensembling.

D Proofs

We can rewrite Eq. 1 as a linear combination of the vertex models plus additive correction terms applied to each face of the simplex:

$$
\begin{aligned}
\theta(\boldsymbol{w}) &= \sum_{|\boldsymbol{\alpha}|=d} b^d_{\boldsymbol{\alpha}}(\boldsymbol{w}) \left(\sum_{i=1}^{N} \frac{\alpha_i}{d} \theta_i + P_{\boldsymbol{\alpha}} - \sum_{i=1}^{N} \frac{\alpha_i}{d} \theta_i \right) \\
&= \sum_{|\boldsymbol{\alpha}|=d} b^d_{\boldsymbol{\alpha}}(\boldsymbol{w}) \sum_{i=1}^{N} \frac{\alpha_i}{d} \theta_i + \sum_{|\boldsymbol{\alpha}|=d} b^d_{\boldsymbol{\alpha}}(\boldsymbol{w}) \left[P_{\boldsymbol{\alpha}} - \sum_{i=1}^{N} \frac{\alpha_i}{d} \theta_i \right] \\
&= \sum_{i=1}^{N} \theta_i \underbrace{\sum_{|\boldsymbol{\alpha}|=d} b^d_{\boldsymbol{\alpha}}(\boldsymbol{w}) \frac{\alpha_i}{d}}_{=w_i} + \sum_{|\boldsymbol{\alpha}|=d} b^d_{\boldsymbol{\alpha}}(\boldsymbol{w}) \tilde{P}_{\boldsymbol{\alpha}} \\
&= \sum_{i=1}^{N} w_i \theta_i + \sum_{|\boldsymbol{\alpha}|=d} b^d_{\boldsymbol{\alpha}}(\boldsymbol{w}) \tilde{P}_{\boldsymbol{\alpha}},
\end{aligned}
$$

where $\tilde{P}_{\boldsymbol{\alpha}} := P_{\boldsymbol{\alpha}} - \sum_{i=1}^{N} \frac{\alpha_i}{d} \theta_i$, and the underbraced equality follows from the first-moment identity of the Bernstein basis on the simplex.

E Sensitivity analysis

We showcase *Nimbus*'s robustness to input and weight perturbations across 5 different initial seeds (for ConvNeXt trained on CIFAR-100, $N = 3$ with $\mathcal{D} = \{2, 3\}$) with respect to NLL loss, on logit ensembling for simplicity (see Table 3). Each seed assigns different classes to each of the 3 tasks and initializes the vertices' and *Nimbus*'s weights differently. *Nimbus*'s multi-task loss is very stable across seeds, and the band of variation is particularly tight at low and moderate levels of noise.

ε	**Input Noise NLL**	**Weight Noise NLL**
0.00	0.97 ± 0.03	0.97 ± 0.03
0.01	0.97 ± 0.03	0.97 ± 0.03
0.05	1.00 ± 0.03	0.96 ± 0.03
0.10	1.11 ± 0.04	0.95 ± 0.03
0.20	1.60 ± 0.13	0.96 ± 0.03
0.40	3.06 ± 0.31	1.77 ± 0.17
0.60	4.60 ± 0.42	3.98 ± 0.26
0.80	5.87 ± 0.55	6.05 ± 0.33
1.00	6.89 ± 0.64	7.34 ± 0.59

Table 3: Robustness of NLL (uniformly sampled on *Nimbus*) under input and weight perturbations over 5 random seeds.

F Full experiments

We include here the complete set of experiments conducted with *Nimbus*. Experimental setups include ConvNeXt ($N \in \{3, 5, 7, 10\}$), ViT-B/32 ($N = 8$), and nanoGPT ($N = 7$), each evaluated on logit ensembling, softmax ensembling, and raw parameter averaging over the ensemble (sampled uniformly on *Nimbus*'s constructed manifold). For each experiment, we also include comparisons between *Nimbus* and all other baselines.

We note that *Nimbus* is more suited to logit ensembling than softmax ensembling, as the former preserves the diversity of its member models while the latter compresses output confidence by removing this diversity via normalization. We include both for completeness but note that logit ensembling should be preferred when using *Nimbus* as it typically also yields significantly lower ECE compared to softmax ensembling. Even though these ensembling rules may produce different results (e.g., different ECE), the NLL is usually roughly unchanged, indicating that the two ensembling

rules differ in the shape of *Nimbus*'s predictions rather than their quality (which is most holistically determined from NLL loss). Additionally, we point out that softmax averaging is susceptible to spreading more mass to non-target classes, thus sometimes depressing top-1 confidence below accuracy.

We also ensemble by raw parameter averaging on *Nimbus*'s manifold, which generally matches logit and softmax ensembling in trend across the same metrics. The same operation applied to the baseline ensembles would collapse; since models learn different tasks and start from different initializations, their raw parameter average will inevitably cross high-loss barriers. This is true even if one were to resolve neuron asymmetries. *Nimbus* is immune to these pathologies by construction.

F.1 Per-task performance

Supp. Fig. 9, Fig. 10, and Fig. 11 show the per-modality, per-task performance of *Nimbus*.

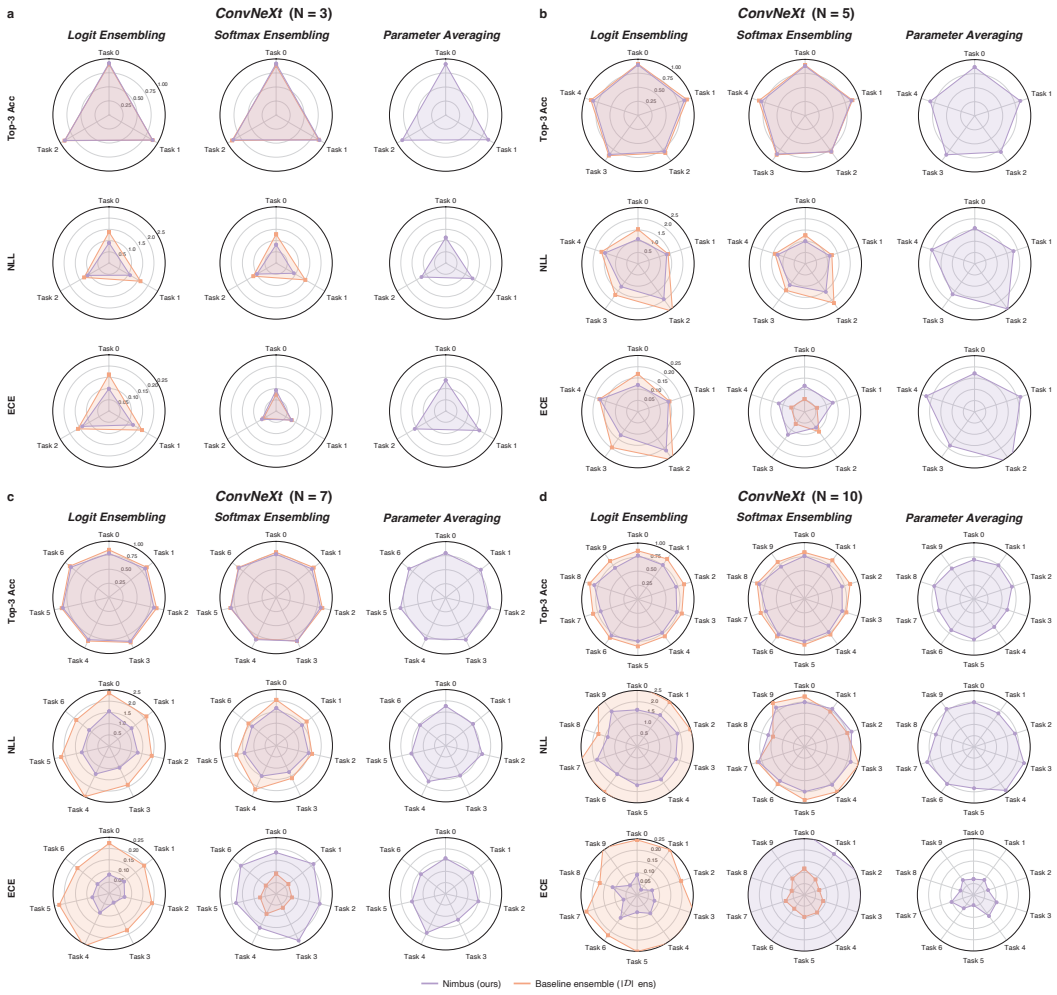


Figure 9: **Per-task performance for ConvNeXt.**

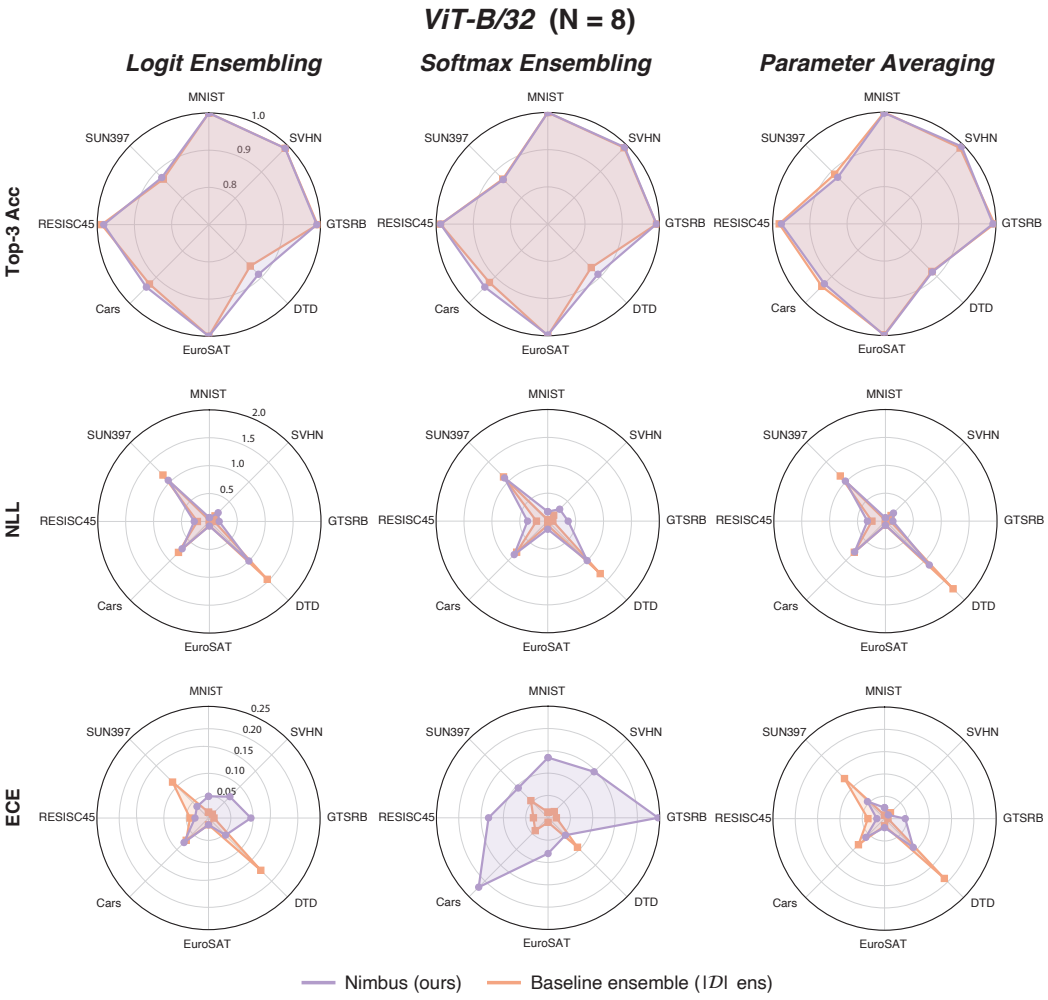


Figure 10: **Per-task performance for ViT-B/32.**

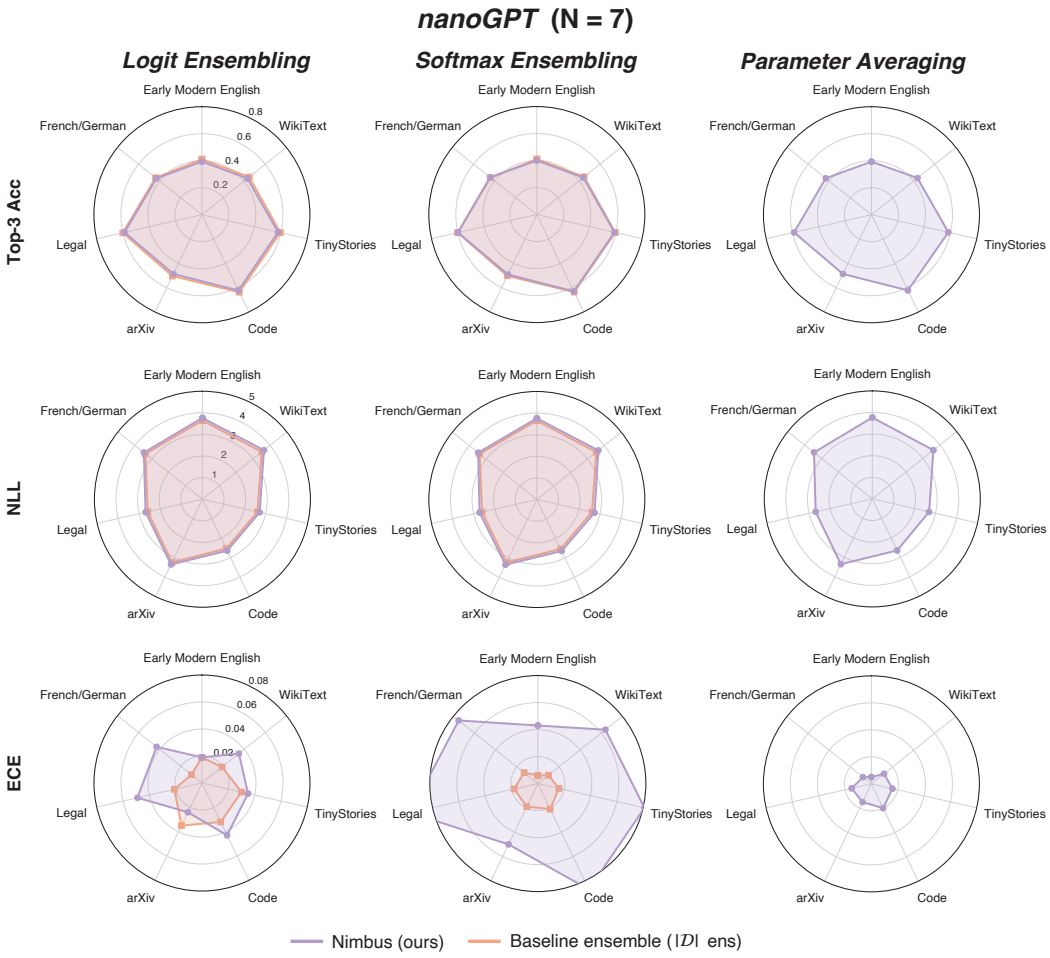


Figure 11: **Per-task performance for nanoGPT.**

F.2 Robustness to input perturbations

Supp. Fig. 12, Fig. 13, and Fig. 14 show *Nimbus*'s robustness to input perturbations, per-modality.

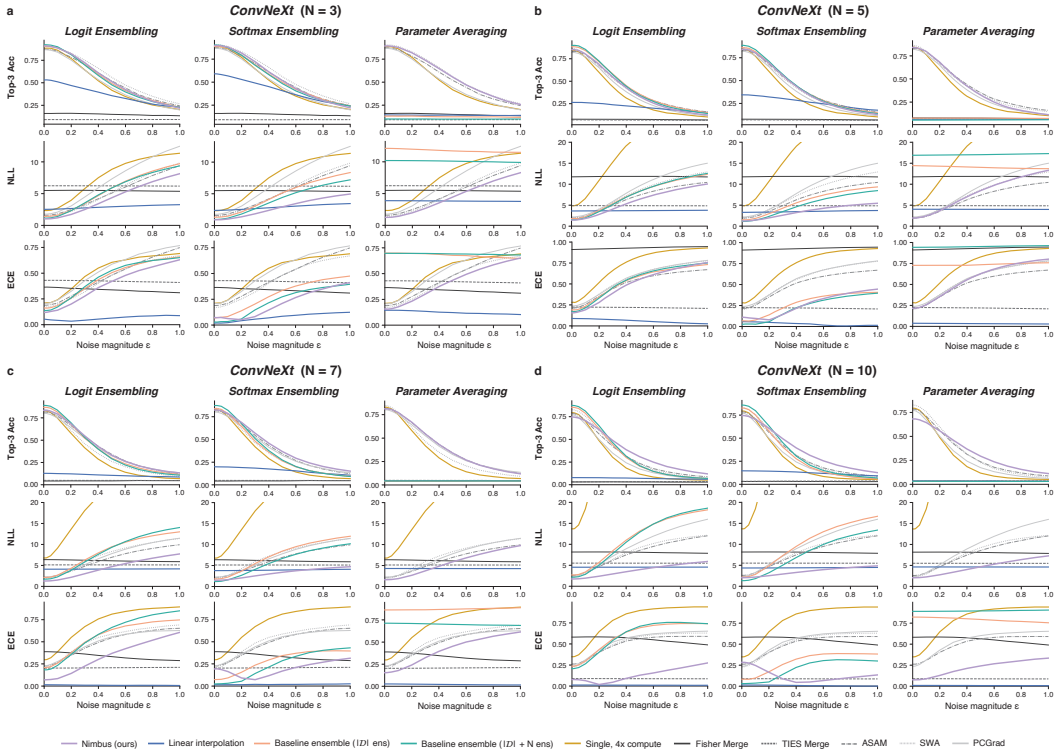


Figure 12: **Input perturbation robustness for ConvNeXt.**

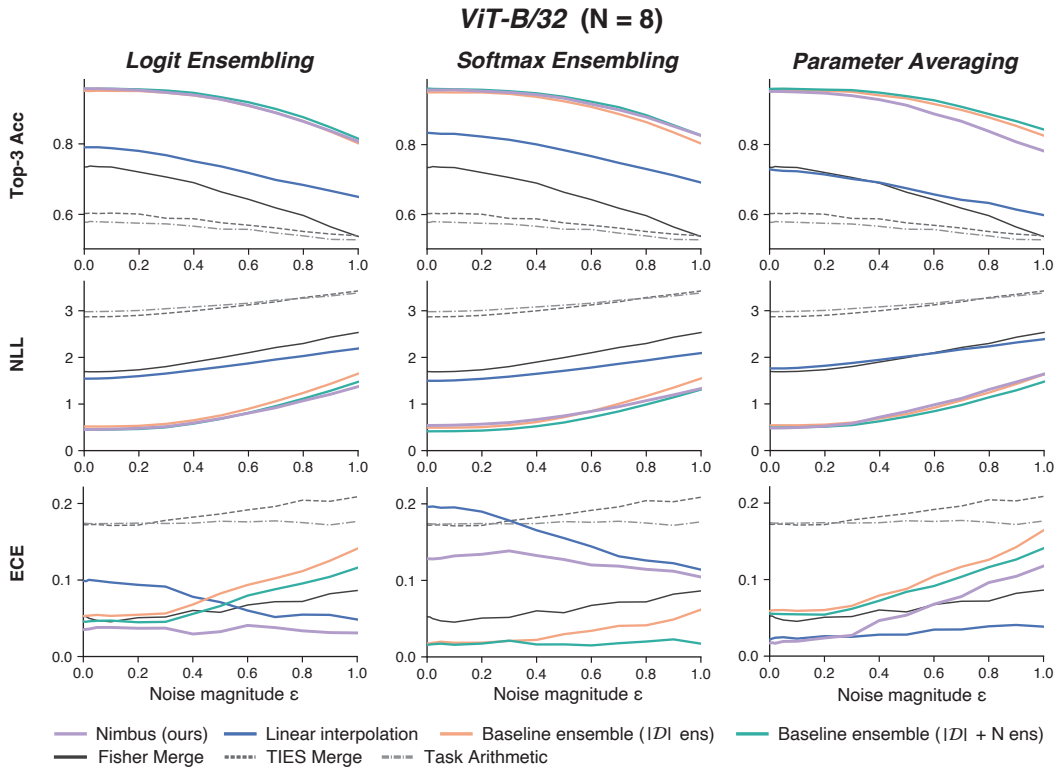


Figure 13: **Input perturbation robustness for ViT-B/32.**

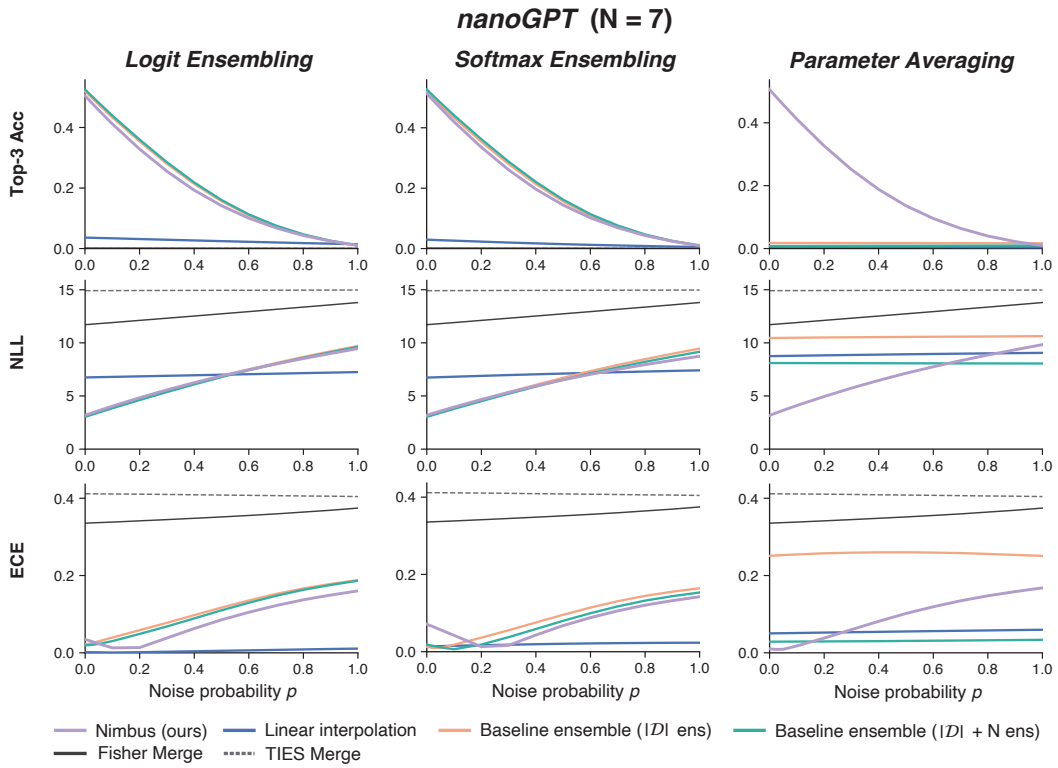


Figure 14: **Input perturbation robustness for nanoGPT.**

F.3 Robustness to weight perturbations

Supp. Fig. 15, Fig. 16, and Fig. 17 show *Nimbus*'s robustness to weight perturbations, per-modality.

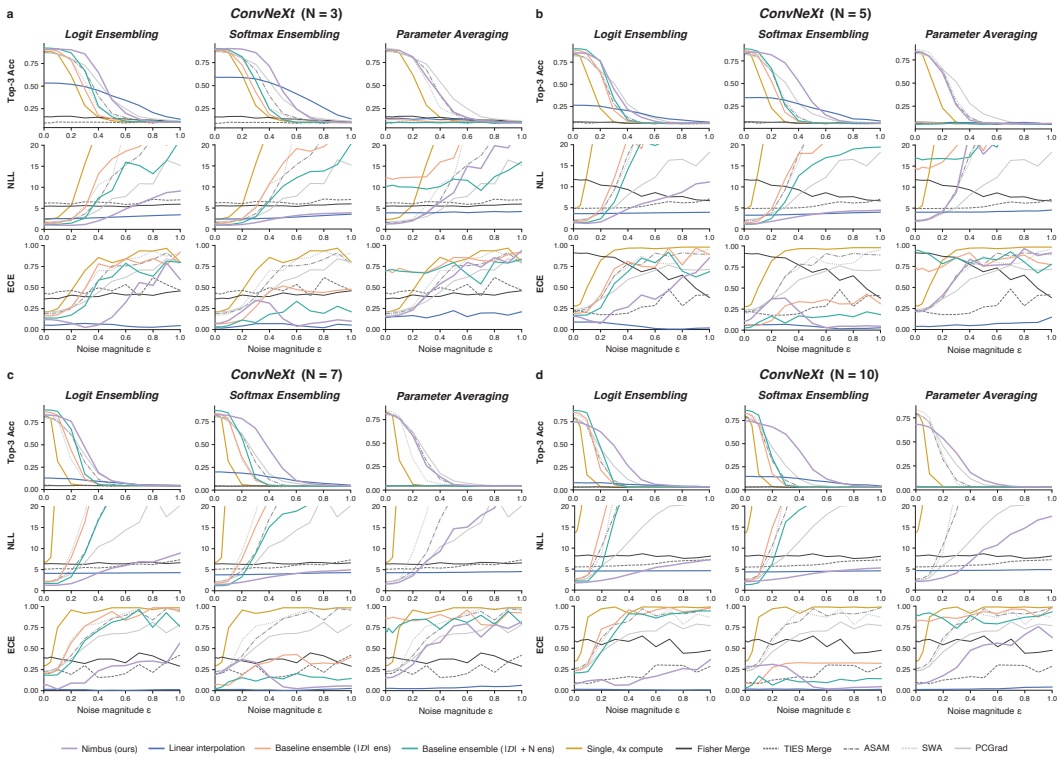


Figure 15: **Weight perturbation robustness for ConvNeXt.**

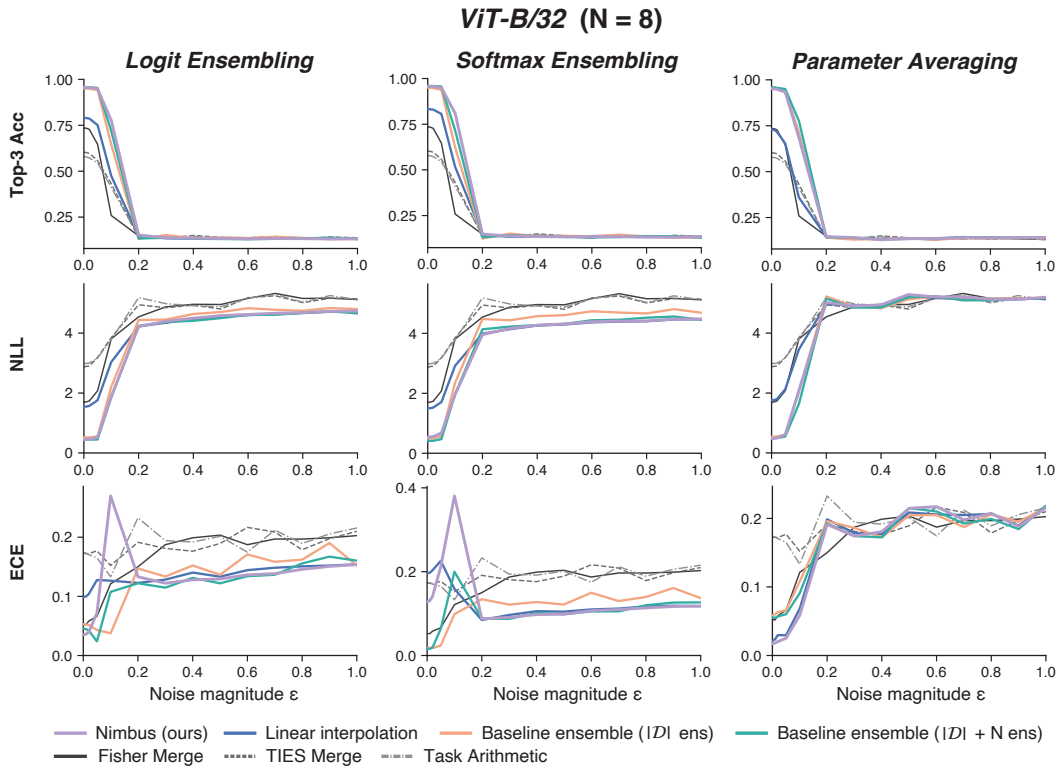


Figure 16: **Weight perturbation robustness for ViT-B/32.**

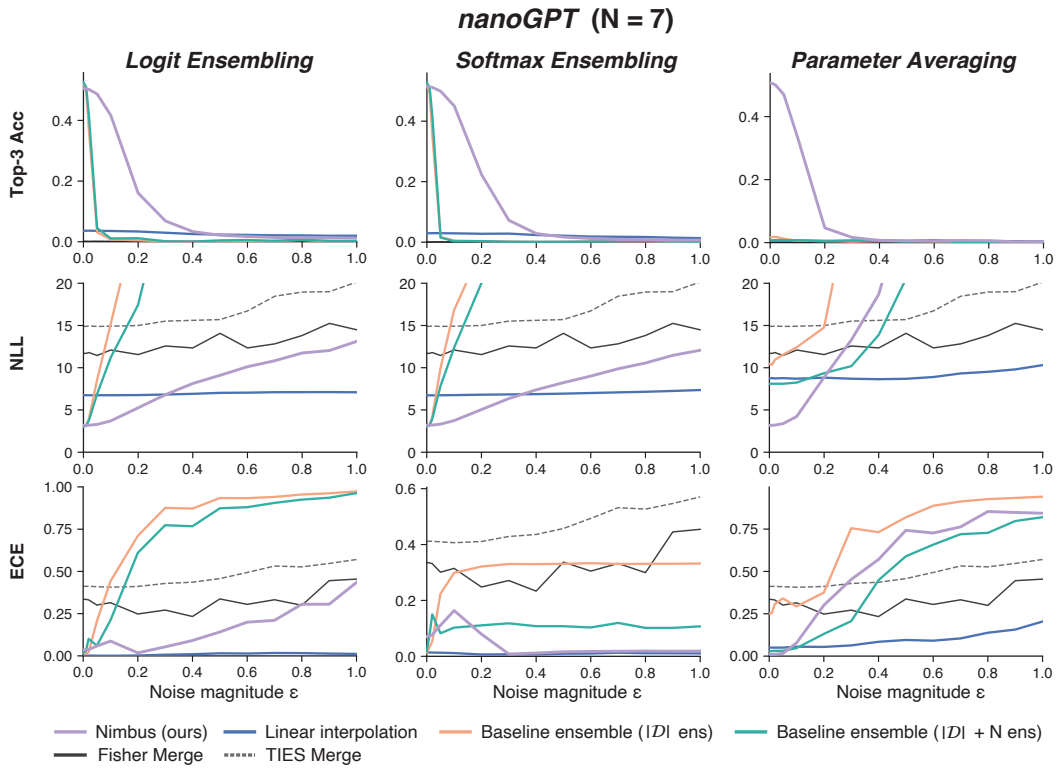


Figure 17: **Weight perturbation robustness for nanoGPT.**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, the main claims of the paper are considered and extensively analyzed in every section after the introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, limitations to this paper are included in the Limitations section of the Discussion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Yes, theoretical motivations and the mathematical framework for our method is clearly discussed in Section 2 and Appendix. Assumptions to our framework are also listed in Section 2.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Code, data, and the hyperparameters are all provided for our experiments in the Supplementary Section. All experiments are seed controlled for reproducibility, and the seed(s) are provided as well as listed in the code.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes, all data and code are provided (see the Supplementary Section). We also provide clear instructions on how to reproduce our data and even the exact figures included in the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, all hyperparameters (data splits, training hyperparams, optimizer type, choice of network, etc.) for training and testing are provided in the supplementary section. The code we provide full repreoduces our training and testing runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, across 5 different seeds, we provide a sensitivity analysis of our method in the Supplementary Section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, we mention the training details in the Appendix section. We mention how long it takes to reproduce all the experiments, as well.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, we have taken into account NeurIPS's Code of Ethics. We abide by them fully.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work does not have any immediate societal impact. We built a neuroscience-inspired model for model merging and ensembling.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our models pose no such risks. Even though part of our experiments train language models, they are small and trained on small datasets. None of our code has potential for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We implemented baselines described by papers. We have cited them in this paper properly.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We don't release new assets (i.e., datasets) of any sort. We do provide code for training our models from scratch, as well as clear instructions on how to run this code. This is all clearly documented and provided in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing data nor does it involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: This paper did not crowdsource experiments nor collect or train on real-world or human data.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research did not involve LLMs or any output generated by LLMs. The main/core methodology was novel and LLM-free.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.