Revisiting Model Stitching in the Foundation Model Era

Zheda Mai[1,3*] Ke Zhang[3] Fu-En Wang[3] Zixiao Ken Wang[3]
Albert Y. C. Chen[3] Lu Xia[3] Min Sun[3] Wei-Lun Chao[2] Cheng-Hao Kuo[3]
[1]The Ohio State University [2]Boston University [3]Amazon
https://zheda-mai.github.io/Model-Stitch mai.145@osu.edu

Abstract

Model stitching, connecting early layers of one model (source) to later layers of another (target) via a light stitch layer, has served as a probe of representational compatibility. Prior work finds that models trained on the same dataset remain stitchable (negligible accuracy drop) despite different initializations or objectives. We revisit stitching for Vision Foundation Models (VFMs) that vary in objectives, data, and modality (e.g., *CLIP, DINOv2, SigLIP2) and ask:* **Are heterogeneous VFMs stitchable?** *We introduce a systematic protocol spanning stitch positions, stitch layer families, training losses, and downstream tasks. Three findings emerge. (1) Stitch layer training matters: conventional approaches that match the intermediate features at the stitch position or optimize the task loss end-to-end struggle to retain accuracy, especially at shallow stitch positions. (2) With a simple feature-matching loss at the target model's penultimate layer, heterogeneous VFMs become reliably stitchable across vision tasks. (3) For deep stitch positions, the stitched model can significantly* ***surpass*** *either constituent model with a small inference overhead (for the stitch layer). Building on these findings, we further propose the VFM Stitch Tree (VST), which shares early layers across VFMs while retaining their later layers, yielding a controllable accuracy-latency trade-off for multimodal LLMs that often leverage multiple VFMs. Taken together, our study elevates stitching from a diagnostic probe to a practical recipe for integrating complementary VFM strengths and pinpointing where their representations align or diverge.*

1. Introduction

The last decade has seen a shift from crafting bespoke model architectures to pre-training vision foundation models (VFMs)—typically Transformers—on massive, heterogeneous data with diverse objectives [4, 39, 49]. VFMs now serve as default backbones for many tasks thanks to strong generalization and transferability: users can often fine-tune with little (or no) task-specific data and reach accuracy that was out of reach only a few years ago [33, 35].

A natural question follows. Although VFMs achieve different scores on downstream, capability-probing tasks [12, 45], are they fundamentally different "end-to-end," or are their internal representations similar or compatible up to simple transformations? The answer matters for strategy: should we keep training new models (costly), or invest in managing and integrating existing ones?

We study this through model stitching: connecting the early layers of a source model to the later layers of a target

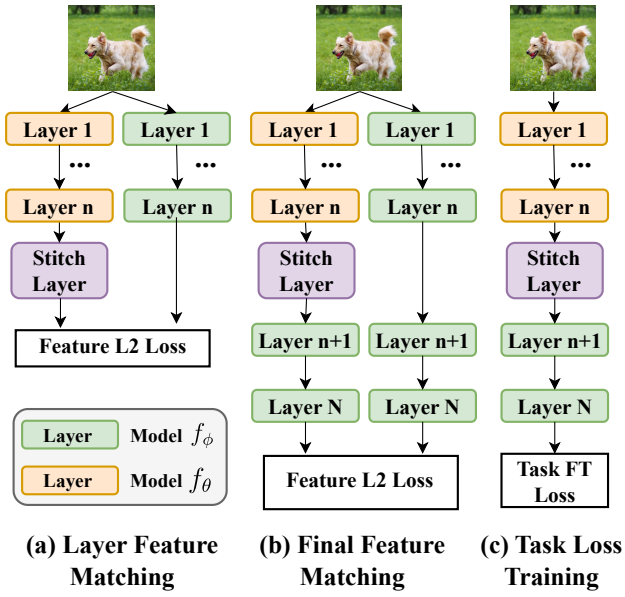

Figure 1. Model stitching training strategies: (a) Layer feature matching trains the stitch layer to match features between the source and target models at the stitch position. (b) Final feature matching trains the stitch layer so that the stitched model matches the target model's final features. (c) Task loss training optimizes the stitch layer with downstream task objective. Across all strategies, the stitch layer is the only trainable component; the source and target models are kept frozen.

*Work done during internship at Amazon.

model via a light stitch layer [3, 9] (Figure 1). Two models are stitchable if, with all original weights frozen and only the stitch layer trained, the stitched model matches the target model's accuracy (*i.e.*, negligible drop). Existing results showed that small models (*e.g.*, ResNet-18 [15]) trained on the same dataset (*e.g.*, CIFAR-10 [28]) are stitchable despite different initializations or objectives. We extend this question to VFMs, large Transformers trained with different datasets (*e.g.*, LAION [21], WebLI [49], LVD-142M [37]), objectives (contrastive vs. reconstruction), and modalities (vision-language vs. pure vision)—including CLIP [39], DINOv2 [37], DINOv3 [40] and SigLIP 2 [49].

Our contribution. We develop a systematic protocol covering the stitch positions, stitch layer designs, training losses, and multiple downstream tasks, and conduct a comprehensive analysis. Key insights are as follows.

Naive stitching fails. We revisit two common strategies to learn the stitch layer [3, 42]: 1) Layer Feature Matching—train the stitch layer to match features between the source and target models at the stitch position; and 2) Task-Loss Training—optimize only the downstream objective (*e.g.*, cross-entropy). In the VFM setting, both struggle, sometimes even much worse than either constituent model, and failures are more pronounced at shallow stitch positions.

Tailored training matters. A closer look at the failures suggests that how to train the stitch layer matters. On the one hand, low feature matching error at the stitch position does not imply aligned final representations, particularly for shallow stitches. On the other hand, Task-Loss Training faces an intrinsic optimization challenge at shallow stitch positions. As all target model's layers after the stitch positions are frozen, gradients originating from the prediction head with weak supervision (*e.g.*, via the final pooled token) must traverse these frozen layers to adjust only the stitch layer, making the loss landscape poorly conditioned, especially when the stitch layer is randomly initialized. We therefore propose a simple two-stage recipe: (i) pre-train the stitch layer to match the target model's final output features (Final Feature Matching), and then (ii) fine-tune with the downstream task loss. The pre-training largely reduces output-feature discrepancy, and the subsequent fine-tuning turns the good initialization into a strong stitched model accuracy—often matching or surpassing linear probes of either VFM across stitch positions.

Stitching integrates complementary strengths. To ensure gains are not merely from the adding stitch layer capacity, we insert the same trainable module into the source-only and target-only models. The stitched model consistently outperforms these self-stitched controls across stitch positions (Figure 5), indicating that stitching VFMs is not only feasible but also has the potential to fuse complementarity between VFMs. We verify this across datasets and tasks—classification and semantic segmentation.

From probe to system: VFM Stitch Tree (VST). Modern multimodal systems (*e.g.* Multimodal LLMs and Vision-Language Models) increasingly deploy multiple VFMs to capture complementary visual cues but incurs linear compute/memory overhead. Leveraging stitchability, we propose the VFM Stitch Tree (VST): share early layers across VFMs and retain specialized deep layers, enabling a controllable accuracy-latency trade-off. Taking a LLaVA model [31] with two VFMs as an example, VST with 22 shared layers and 1 specialized layer (4.3% extra resources) achieves 45% gain of the full two-VFM (requires 100% extra resources). With 14 shared + 9 specialized layers (40% extra), VST achieves 84% of the gain. Thus, VST offers a compute-aware knob for integrating complementary VFMs in multimodal systems.

Summary. Our paper has the following key contributions.

- We revisit model stitching for VFMs and show that with appropriate training, they are *reliably stitchable*. Across vision tasks, stitched models consistently improve upon baseline performance, suggesting complementary knowledge transfer between VFMs.
- Based on the findings, we propose VFM Stitch Tree, offering a controllable performance–efficiency trade-off for multimodal LLMs that employ multiple VFMs.
- To our knowledge, this is the first systematic study of model stitching for VFMs, aiming to advance this technique from a pure diagnostic probe toward a practical recipe and open a path for future work to integrate complementary VFM strengths.

2. Related Work

2.1. Model Stitching and Representational Analysis

Foundations of Model Stitching. Model stitching was introduced by [29] as a tool for analyzing representations through the lens of equivariance and equivalence. The key idea is to connect intermediate layers from two models using a trainable stitch layer and measure whether the stitched model remains functional. Subsequent work [3] showed that models trained under similar settings are often stitchable with negligible performance degradation, suggesting a certain degree of representational compatibility. A critical refinement in this field is the distinction between representational similarity (*e.g.*, CKA [26]) and functional similarity (task performance compatibility) [9]. More recently, [42] questioned whether stitching success should be interpreted as evidence of semantic alignment, arguing instead that it may largely reflect representational clustering. [25] provided a broader survey of representation comparison methods, highlighting the difficulty of drawing robust conclusions about neural representations.

Model Stitching Training. Recent work has explored different objectives for learning the stitch layer. [8] com-

pared layer feature matching, which trains the stitch layer to match source and target features at the stitch position, with task loss matching, which optimizes the stitch layer directly for downstream performance. They found that task loss matching can produce out-of-distribution intermediate representations that improve task-specific accuracy, whereas layer feature matching is often more reliable for representation analysis. Recently, Functional Latent Alignment [1] was proposed to encourage alignment not only at the stitch position but also across subsequent layers.

Application of Model Stitching. Beyond representation analysis, model stitching has also been used for efficient model design and deployment. For example, [38] and its concurrent work [48] use stitching to construct flexible model families that support resource-adaptive inference by recombining components from different networks.

Despite this progress, existing work largely focuses on relatively small models trained from scratch or ImageNet-scale pretraining. In contrast, we explore model stitching between independently trained large VFMs with substantial heterogeneity in pretraining data (*e.g.*, LAION [21], LVD-142M [37]), learning objectives (contrastive versus reconstruction), and modality (vision-language versus pure vision). Furthermore, we shift the paradigm from passive analysis to active knowledge fusion, demonstrating that stitching can bridge complementary specializations to surpass the performance of individual constituent models.

2.2. Vision Foundation Models

A wide range of architectures, learning strategies and data have been explored to develop vision foundation models (VFMs). Self-supervised models such as DINOv2 [37] learn strong visual representations without explicit labels, while vision-language models such as CLIP [39] and SigLIP [46, 49] use contrastive objectives to align images and text. Efforts to scale both model size and data [10, 43] have further pushed the boundaries of zero-shot generalization and robust representation learning. As a result of these differences in architecture, supervision, and training data, different VFMs often exhibit distinct strengths and weaknesses across visual abilities, such as fine-grained recognition, depth estimation, and semantic grounding [34, 45]. This diversity motivates us to ask *whether fundamentally different VFMs can be stitched together, and if so, how their complementary specializations can be leveraged efficiently.* Detailed descriptions of the VFMs considered in this work are provided in Appendix A.

3. Model Stitching for VFMs

3.1. Problem Formulation

Let $f = f^N \circ \cdots \circ f^1$ be a VFM with N Transformer layers. For an image x, we define R_n as the function to map the input x to the intermediate features $\boldsymbol{A}^n \in \mathbb{R}^{d\times k}$ at layer n: $R^n(x) = f^n \circ \cdots \circ f^1(x) = \boldsymbol{A}^n$, where d and k denote the token dimension and number of token. Similarly, we define the function T^N that map the $\boldsymbol{A}^n$ to the output feature $\boldsymbol{A}^N$: $T^N(\boldsymbol{A}^n) = f^N \circ \cdots \circ f^n(\boldsymbol{A}^n) = \boldsymbol{A}^N$.

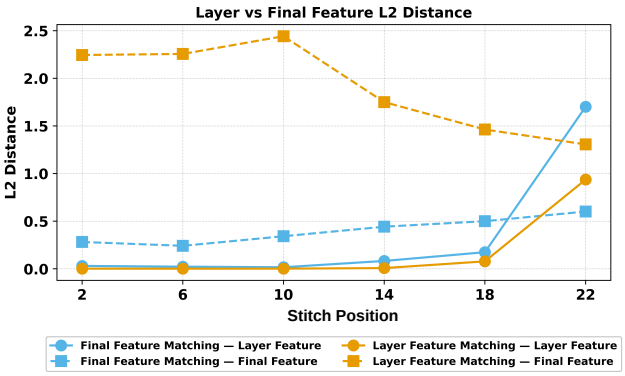

Figure 2. Feature distance of Layer Feature Matching (LFM) and Final Feature Matching (FFM) on SigLIP→DINOv2. We measure the ℓ_2 distance between the stitched model's features and those of the target model when stitching at different positions. -●- indicates the *layer* feature distance and -■- indicates the *final* feature distance. While LFM minimizes the layer feature distance, it leaves a relatively large final feature distance. In contrast, FFM achieves substantially smaller final-feature distance.

Given a source and target VFM with parameters θ and ϕ respectively, we can stitch them at layer n by matching the source features A^n_θ to the target feature A^n_ϕ through a trainable stitching layer $S : \mathbb{R}^{d_\theta \times k} \to \mathbb{R}^{d_\phi \times k}$. Thus, the stitched VFM can be represented by:

$$F(x) = T^N_\phi \circ S \circ R^n_\theta(x)$$

We want to emphasize that all the parameters in T^N_ϕ and R^n_θ are **frozen**; only the stitch layer S is trainable. Following [3, 9], for a given task, the source and target models are stitchable if the stitched model F has limited performance degradation compared to the target model f_ϕ.

3.2. How to Train the Stitch Layer?

To answer whether heterogeneous VFMs are stitchable, we first explore the fundamental question: *Are existing stitching approaches still effective for VFMs?* Two training strategies are most common (see Figure 1): (1) Layer Feature Matching [9] and (2) Task Loss Training [3]. In the following sections, we aim to conduct controlled experiments to evaluate their compatibility with VFMs.

Experiment Setup. We start with two commonly used VFMs: DINOv2-L [37] trained with only vision data (LVD-142M) in a self-supervised manner, and SigLIP2-L [46] trained on vision-language data (WebLI) [6] with both vision self-supervised loss and language supervised loss.

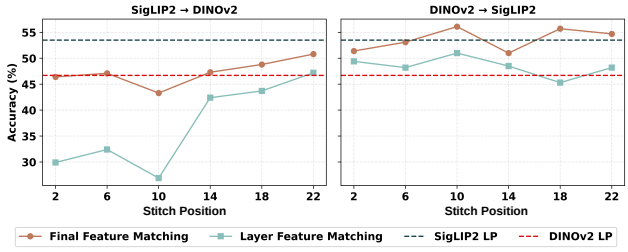

Figure 3. Final Feature Matching consistently shows better accuracy than Layer Feature Matching. In the DINOv2→SigLIP2 case, the stitched model can even exceed the performance of both constituent models.

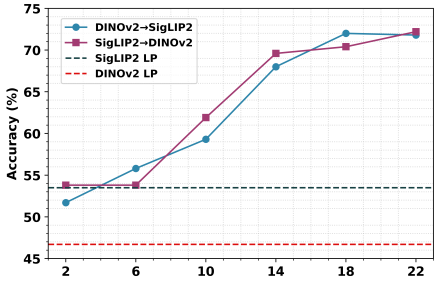

Figure 4. Our two-stage training approach (Final Feature Matching + Task Loss Training) allows stitched models to consistently outperform linear-probing of both constituent models.

Both VFMs have 24 layers. In the following control studies, we use a two-layer perceptron with ReLU (MLP) as the stitch layer, as it's a common feature projector for self-supervised learning [5] and multimodal LLMs (*e.g.* LLaVA-1.5 [32]). We focus first on image classification with fMoW [7], a challenging and commonly used fine-tuning dataset for VFM evaluation [11]. More setup details are provided in Appendix. Detailed investigation of stitch layer families, datasets, VFM, and evaluation tasks will be discussed in the latter sections.

3.2.1. Layer Feature Matching (LFM)

The intuition behind LFM is that the target features should be easily matched from the source features if they are similar enough. Given unlabeled training images $\mathcal{D} = \{x_i\}_{i=1}^M$, LFM trains S by minimizing the feature discrepancy at layer n:

$$\mathcal{L}_{LFM} = \frac{1}{M}\sum_{i=1}^{M} \|S(R_\theta^n(x_i)) - R_\phi^n(x_i)\|_2^2$$

Note that LFM training is label-free, and if we pre-extract the source and target features, the training can be done without VFM inference.

Observations. To assess how well the target features can be transformed from source features across stitch positions $n \in [2, 6, 10, 14, 18, 22]$, we measure on the validation set the mean ℓ_2 *layer* feature distance $\|S(R_\theta^n(x)) - R_\phi^n(x)\|_2$ (● in Figure 2). As expected, because LFM optimizes this directly, these distances are very small (order of 10^{-3}). However, small layer feature discrepancies do *not* guarantee similarity at the final features. On the contrary, we observe substantially larger *final* feature distances $\|T_\phi^N(S(R_\theta^n(x))) - R_\phi^N(x)\|_2$ (■ in Figure 2), especially for shallow stitches. Intuitively, a small mismatch could accumulate and possibly be amplified by the frozen target layers, resulting in a pronounced final feature difference.

Remedy. Motivated by the observations above and inspired by feature distillation, we propose **Final Feature Matching** (FFM), which trains the stitch layer to directly match the final features at layer N between the stitch model F and target model f_ϕ:

$$\mathcal{L}_{FFM} = \frac{1}{M}\sum_{i=1}^{M} \|T_\phi^N(S(R_\theta^n(x_i))) - T_\phi^N(R_\phi^n(x_i))\|_2^2$$

As shown in Figure 2, compared with LFM, FFM significantly reduces the *final* feature distances (■) at shallow stitch positions—precisely where difference accumulation is most pronounced. More interestingly, although FFM directly matches the final features, it still retains similarly low layer feature distances (●) as LFM, suggesting that the supervision at the final layer can induce implicit local alignment at the stitch positions. The better final feature alignment translates into better linear probing accuracy (Figure 3): stitch models trained with FFM consistently outperform their LFM counterparts across stitch positions. Notably, in the DINOv2→SigLIP2 case, the stitched model can exceed the performance of both constituent models under linear probing. Crucially, this improvement is achieved without using any labels; the stitch layer is trained only to match the final representations in FFM.

3.2.2. Task Loss Training (TLT)

Task Loss Training (TLT) trains the stitching layer directly with downstream task loss using the labeled training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^M$, where ℓ is the task-specific loss (e.g., cross-entropy for classification):

$$\mathcal{L}_{task} = \frac{1}{M}\sum_{i=1}^{M} \ell(F(x_i), y_i) \quad (1)$$

Observations. Existing studies show that generally TLT outperforms LFM across stitch positions, as it directly optimizes the task objective. We also observe this pattern at deep stitch positions. However, in contrast to existing observations, we find that TLT fails sharply for **shallow** stitches. Taking DINOv2→SigLIP2 stitching at layer 2 as an example, TLT attains only 25.1%, markedly below the linear probing for DINOv2 (46.7%), SigLIP2 (53.5%), LFM (49.4%) and FFM (51.4%).

Pre-train	DINOv2 → SigLIP2					
Stitch Position	2	6	10	14	18	22
No	25.1	39.4	52.6	62.3	68.6	68.6
LFM	33.1	57.8	59.3	67.3	68.4	70.3
FFM	**51.7**	**59.8**	**61.1**	**68.0**	**72.0**	**71.8**
Pre-train	SigLIP2 → DINOv2					
Stitch Position	2	6	10	14	18	22
No	38.7	50.7	58.3	64.4	70.4	70.1
LFM	35.8	51.8	59.2	69.4	70.2	72.1
FFM	**53.8**	**53.8**	**61.9**	**69.6**	**70.4**	**72.2**

Table 1. Initializing the stitch layer with FFM substantially improves over naive TLT, especially at shallow stitch positions where naive TLT severely underperforms. FFM also provides consistent gains at deeper stitch positions. Compared with LFM initialization, FFM offers a clear advantage at shallow stitch positions, while their performance becomes similar at deeper stitch positions (Numbers are accuracy on fMoW).

Remedy. When stitching shallowly, all post-stitch target layers are frozen; thus, gradients from the prediction head must backpropagate through a long, fixed transformation to update *only* the stitch layer. This may attenuate update directions and cause optimization challenges, especially under random initialization and weak supervision (via pooled-token). To place the stitch layer in a better loss landscape, we adopt a simple two-stage procedure:

(i) **Initialize** the stitch layer by pre-training with FFM
(ii) **Fine-tune** the stitch layer with TLT

As summarized in Table 1, FFM initialization not only rescues naïve TLT at shallow stitches where TLT alone underperforms, but also yields consistent gains at deeper stitch positions. In our experiments, it also significantly surpasses the linear-probing performance of both constituent models, as shown in Figure 4. We also observe that shallow layers consistently underperform. We hypothesize that early layers may encode pretraining-specific features (e.g., low-level visual features for DINOv2, text-aligned features for SigLIP2) while deep layers may develop more transferable, task-relevant representations, leading to easier stitching than shallow layers.

Our approach also resonates with a recent work [8] which argues that TLT can create out-of-distribution representations that optimize for task performance at the cost of representational fidelity. Our FFM initialization mitigates this risk by first preserving representational fidelity, then applying task-level adaptation for functionality.

4. Where do Improvements Come From?

Why do our results differ from prior works? With our two-stage training, the stitched model can substantially exceed linear probing for both the source and target models(Figure 4). One might be tempted to conclude that VFMs are broadly stitchable, given that stitched models typically have around 0-10% accuracy drop compared to the target model in prior works [3]. However, in earlier studies, the source, target, and stitch layer were all trained/evaluated on the *same* dataset (e.g., CIFAR-10). In contrast, VFMs are pretrained on massive, diverse data and evaluated via fine-tuning on downstream data. Under this regime, improvements could arise simply from *task adaptation in the stitch layer*. We therefore design baselines that disentangle stitch layer capacity.

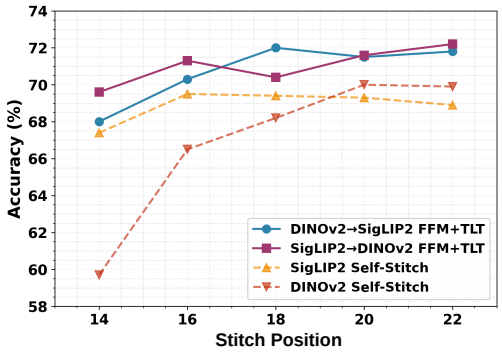

Figure 5. Stitched Model vs Self-Stitch. Both DINOv2→SigLIP and SigLIP→DINOv2 (solid lines) consistently outperform their respective self-stitch baselines (dashed lines), demonstrating genuine knowledge fusion gains.

4.1. Self-Stitch Baseline

To rigorously test whether gains only stem from added capacity, we introduce a Self-Stitch baseline: inserting the identical stitch layer into the source-only and target-only models at the same stitch positions. For example, for SigLIP2→DINOv2 stitched at layer 22, the baselines are SigLIP2→SigLIP2 and DINOv2→DINOv2 with an identical stitch module at layer 22. Self-stitch controls share the *same* trainable module, stitch positions, training loss, and downstream data. Matching self-stitch performance therefore indicates *genuine VFM stitchability*.

4.2. Stitching integrates complementary strengths

Figure 5 compares the cross-VFM stitched model against their self-stitched baselines. Surprisingly, the stitched model consistently outperforms the self-stitched controls across stitch positions, indicating that improvements are *not* explained solely by stitch layer capacity or downstream fine-tuning. Instead, the results suggest that stitching heterogeneous VFMs is not only feasible but also has the potential to *fuse complementarity* between them. To better understand how knowledge fusion affects prediction behavior, we provide prediction analysis for the stitched model and self-stitch baseline in Appendix B.

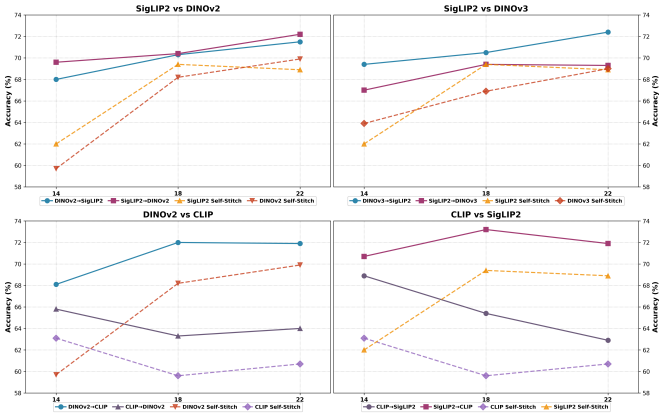

Figure 6. Evaluation of DINOv2, DINOv3, SigLIP2 and CLIP. The stitched models generally outperform both corresponding self-stitch baselines across stitched layers, except for using CLIP as the source model (discussion in Sec. 5.2).

5. Generality of Model Stitching

To stay focused on exploring *how to train the stitch layer* in previous sections, we adopt a fixed experiment setup. We assess the *consistency* of our findings across datasets, tasks, stitch-layer architectures, and VFMs.

5.1. Validation Across Datasets & Tasks

In addition to fMoW [7] (satellite images), we evaluate on two widely used fine-grained classification datasets: iNat-Subset [47] (animal species) and FGVC-Aircraft [36]. Beyond classification, we evaluate semantic segmentation on ADE20K [50, 51] with a linear decoder [23]. To simplify feature matching and avoid confounding factors, no data augmentation is used during training.

As shown in Table 2, stitched models consistently surpass the corresponding self-stitch baselines across datasets in classification (+0.7% to +5.5%), indicating that the observed knowledge fusion gain is *not* a dataset-specific artifact but a generalizable observation. In ADE20K, we likewise observe steady improvements over self-stitching (+0.5 to +0.7 mIoU), demonstrating that complementary knowledge fusion extends to dense prediction. We hypothesize that DINOv2 contributes robust perceptual structure while SigLIP2 provides stronger semantic alignment; their combination, mediated by the stitch layer, yields modest yet reliable improvements. More dataset details, experiment setup and results are provided in the Appendix.

5.2. Validation Across More VFMs

Beyond the DINOv2 and SigLIP2 pairs, we further include the widely used CLIP and the recently released DINOv3 to verify the generality of our findings. As shown in Figure 6, the stitched models generally outperform both corresponding self-stitch baselines across stitched layers, except for using CLIP as the source model. In this case, CLIP is a weaker VFM in our setting (as evidenced by lower linear-probing and self-stitch performance), so the stitched model can improve over CLIP itself but still fails to match the stronger target model. Viewing the source as an encoder, this suggests that a weak encoder may discard task-critical information that the stronger target network cannot recover. Conversely, when CLIP serves as the target, the stitched model achieves strong performance, indicating that as long as a strong source model preserves rich intermediate representations, even a relatively weak target can still transform these features into high-quality final predictions. These results are reminiscent of observations in encoder–decoder architectures for detection and segmentation, where upgrading only the encoder/backbone (e.g., from ResNet-50 to ResNet-101) reliably improves performance, while substituting a strong backbone with a weaker one consistently degrades accuracy even under an unchanged decoder [16, 22, 30].

Stitch Position		2	6	10	14	18	22
D→S	Linear	26.1	54.3	**59.5**	66.5	69.1	69.6
	MLP	**51.7**	**55.8**	59.3	**68.0**	**72.0**	**71.8**
	LoRA	49.1	49.4	57.4	61.7	67.7	67.3
S→D	Linear	50.3	**56.4**	60.0	65.7	69.6	71.9
	MLP	**53.8**	53.8	61.9	**69.6**	**70.4**	**72.2**
	LoRA	48.3	56.2	**62.4**	65.3	66.2	65.0

Table 3. Stitch layer comparison: MLP consistently outperforms Linear and the LoRA option across stitching directions (D→S, S→D) where D and S denote DINOv2 and SigLIP2, respectively.

5.3. Across Stitch Layer Families

Beyond the default MLP, we explore two different stitch layers: (1) a linear layer and (2) source model's n layer with LoRA [17], mapping $(n-1)$-layer features of the source to the n-layer features of the target. Concretely, (2) can be represented as $F_{LoRA}(x) = T_\phi^N \circ f^n_{(\theta,LoRA)} \circ R_\theta^{n-1}(x)$ where only LoRA parameters within $f^n_{\theta,\text{LoRA}}$ [18, 35] are trainable. From a capacity perspective, the linear layer processes tokens independently and is the least expressive; the MLP adds nonlinearity while still operating per token; the LoRA option allows inter-token interactions and is the most expressive. As reported in Table 3, the MLP generally attains the strongest performance on the benchmarks considered, with linear trailing—as expected given its lower capacity. Somewhat counterintuitively, the LoRA-based option often underperforms the MLP despite its higher expressiveness. A plausible interpretation is that stitching may benefit from *controlled* mismatch that enables complementary information to be fused. If the stitch layer perfectly reproduces the target's intermediate features, $S(R^n_\theta(x)) = R^n_\phi(x)$, the stitched model would be expected to match the target's self-

	Classification			Segmentation
	fMoW	**iNaturalist**	**Aircraft**	**ADE20K**
Stitch Position	**6 / 14 / 22**	**6 / 14 / 22**	**6 / 14 / 22**	**14 / 22**
DINOv2→DINOv2	41.5 / 59.7 / 69.9	56.9 / 81.5 / 91.2	37.8 / 79.3 / 91.2	35.4 / 50.9
SigLIP2→SigLIP2	50.5 / 62.0 / 68.9	71.2 / 88.5 / 87.3	67.9 / 88.1 / 89.3	44.5 / 50.5
DINOv2→SigLIP2	**59.8** / 68.0 / 71.8	75.9 / 89.1 / **92.8**	77.8 / 87.6 / **92.4**	44.9 / 51.2
SigLIP2→DINOv2	53.8 / **69.6** / **72.2**	**86.3** / **88.9** / 91.9	80.7 / 89.0 / 91.0	**49.0** / **51.4**

Table 2. Comprehensive results across all datasets and tasks. Classification results in accuracy (%), segmentation in mIoU (%). All results use the two-stage approach (Final Feature Matching + Task Loss Training).

stitch, leaving little room for cross-model complementarity.

5.4. Does Stitching Require Task-Specific Data?

Our previous results show that FFM is an effective stitching objective. However, these findings rely on task-specific training data (*e.g.* use fMoW training images to learn the stitch layer for evaluation on fMoW). We therefore ask whether stitching must be task-specific, or whether it can instead be learned from general task-agnostic data. To study this, we train stitch layers using the task-agnostic LLaVA-1.5 data (1.1M images) [32], and evaluate the resulting stitched models on downstream tasks via linear probing, without further TLT.

	fMoW		iNaturalist	
	T-Spec.	T-Agn.	T-Spec.	T-Agn.
D→S	54.7	53.8	77.3	77.45
S→D	50.8	50.0	75.5	77.7

Table 4. Comparison between task-specific (T-Spec.) and task-agnostic (T-Agn.) stitching. Results are reported for layer 22 (earlier analysis identifies as a strong stitch position).

As shown in Table 4, task-agnostic FFM achieves comparable, and in some cases better, accuracy compared to task-specific training. On fMoW, task-agnostic training matches task-specific performance, while on iNat, it surprisingly yields a performance gain. We hypothesize that iNat aligns more closely with the visual distribution of LLaVA-1.5 than fMoW, allowing the significantly larger scale of the LLaVA-1.5 data to facilitate more robust feature matching. These results suggest that stitch layers can be pre-trained in a task-agnostic manner and reused across diverse downstream applications, obviating the need for per-task retraining. More broadly, this indicates that stitching captures a fundamental alignment between the representation spaces of two VFMs rather than a narrow task-specific mapping. This opens the possibility for **Hybrid VFMs**: stitching can be used for synthesizing a better VFM by fusing the complementary strengths of distinct VFMs in a task-agnostic way. Such hybrid models can then serve as more versatile backbones for a wide array of downstream tasks.

Key Findings

- Heterogeneous VFMs trained with different data, objectives, and modalities remain stitchable across diverse vision tasks and datasets.
- Stitched VFMs consistently outperform self-stitch baselines, indicating that the gains arise not merely from the added capacity of the stitch layer, but also from the complementary strengths of different VFMs.

6. From Insights To Applications

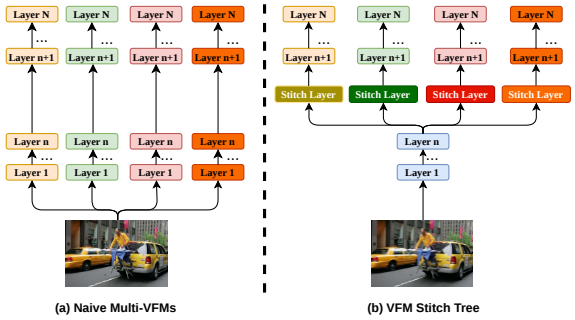

Figure 7. VFM Stitch Tree (VST) serves as a simple and efficient alternative to naively running all VFMs end-to-end.

6.1. The Efficiency Challenge For Multi-VFMs

Modern multimodal systems increasingly deploy multiple VFMs to capture complementary visual information. For example, MoF-LLaVA [45] combines CLIP and DINOv2 to preserve instruction following while improving visual grounding; OpenVLA [24] leverages SigLIP and DINOv2 to fuse low-level spatial structure with higher-level semantics; Cambrian-1 [44] shows that assembling four VFMs with distinct strengths can lift MLLM performance across diverse benchmarks. However, there is no free lunch: with k VFMs, one must load all of them into GPU memory ($k\times$) and process each input k times, yielding roughly $k\times$ latency. **Question:** Can we design an architecture that retains the benefits of multiple VFMs *without* incurring linear

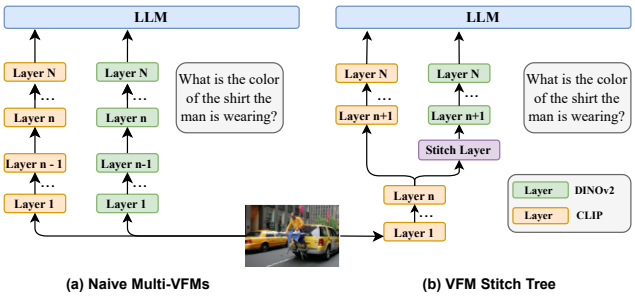

Figure 8. VFM Stitch Tree (VST) can be easily applied in various multimodal systems. This is an example of VST in a MoF-LLaVA with CLIP and DINOv2.

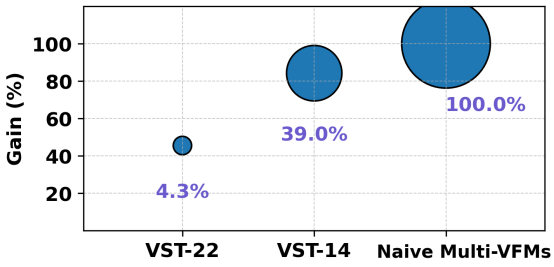

Figure 9. VST recovers 45% and 84% of the gain achieved by the naive multi-VFM baseline (100% extra resources), with only 4.3% and 39% extra resources. Details are in Appendix.

compute and memory costs?

6.2. VFM Stitch Tree (VST)

Motivated by the stitchability of VFMs, we propose the *VFM Stitch Tree (VST)*: shares common shallow layers across VFMs while *retains* model-specific deep layers, connected via stitch layers (Figure 7). VST serves as a simple and efficient alternative to naively running all VFMs end-to-end. Consider Cambrian-1 with four VFMs. A VST stitched at layer 14 reduces GPU memory and computation relative to running all four full VFMs (Tab. 5).

Method	Param (M)	GFLOPs	Lat. (ms)	Mem (MB)
Naive	1581.4	878.8	96.2	6075
VST-6	1219.0	701.7	71.4	4502
VST-14	815.9	419.5	43.9	3065
VST-22	412.9	217.4	20.6	1527

Table 5. Efficiency metrics for different stitching configurations. Latency is measured on a single NVIDIA A100 GPU with a batch size of 1. VST significantly reduces computational overhead compared to the Naive Multi-VFM baseline.

We instantiate VST within a MoF-LLaVA (CLIP + DINOv2) with a Qwen-3B LLM [20]. The model is trained following the standard LLaVA-1.5 [32] training recipe and datasets. We evaluate VST stitched at layer 14 and 22, as demonstrated in Figure 8. Relative to a single-VFM LLaVA baseline, the naïve two-VFM MoF-LLaVA doubles the VFM cost (100% extra). In contrast, our VST variants require only 39% (VST-14) and 4.3% (VST-22) additional resources. In this early exploration, we evaluate the MLLM on VQAv2 [14] and MME-Perception and MME-Cognition [13]. As shown in Figure 9, VST-22, with just *one* additional specialized layer (4.3% overhead), recovers 45% of the two-VFM performance gain; when more budget is available, VST-14 reaches 84% of that gain at 39% overhead. We provide experiment setup, detailed results and resource calculation in the Appendix.

Accuracy–efficiency knob. Without VST, users face a binary choice: deploy an entire additional VFM (higher performance, 100% extra backbone cost) or none (no gain). VST introduces a *compute-aware knob* that interpolates between these extremes, enabling controllable performance–efficiency trade-offs under real deployment constraints. We view this as an initial step toward resource-conscious multi-VFM design; broader evaluations are provided in the Appendix.

We want to highlight that our findings on VFM stitchability open a fertile space of research questions about knowledge fusion and also indicate potential for practical utility. VST is just *one* instantiation of how these insights can be applied. The central takeaway is that heterogeneous VFMs can be stitched and fused to exploit complementary knowledge, which has potential applications across different domains and use cases.

7. Conclusion

We revisit model stitching under the foundation model regime and find that, contrary to the common assumption that heterogeneity hinders compatibility, VFMs trained with different data, objectives, and modalities can in fact be stitched effectively. Our study shows that the key lies in how the stitch layer is trained: while conventional layer feature matching and naive end-to-end task optimization often fail, especially for shallow stitches, a simple two-stage strategy based on final feature matching followed by task fine-tuning yields strong and consistent results. Importantly, stitched models not only remain functional but often surpass corresponding self-stitch baselines, suggesting that stitching can actively fuse complementary knowledge across VFMs rather than merely preserve target behavior. We further demonstrate that these findings generalize across tasks, datasets, stitch-layer designs, and model families, and that stitch layers can even be pretrained in a task-agnostic manner. Finally, we show how these insights enable VFM Stitch Tree, a practical architecture that reduces the cost of multi-VFM systems while preserving much of their benefit. Taken together, our work advances model stitching from a representational probe to a practical paradigm for integrating and scaling heterogeneous foundation models.

References

[1] Ioannis Athanasiadis, Anmar Karmush, and Michael Felsberg. Functional similarity by functional latent alignment. In *Greeks in AI Symposium 2025*, 2025. 3

[2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. In *arXiv preprint arXiv:2309.16609*, 2023. 2

[3] Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2, 3, 5

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660, 2021. 1

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020. 4

[6] Xi Chen, Xiao Wang, Soravit Changpinyo, Anthony J Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022. 3

[7] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 4, 6, 1

[8] Katherine M Collins, Umang Bhatt, and Adrian Weller. How not to stitch representations to measure similarity: Task loss matching versus direct matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. 2, 5

[9] Adrián Csiszárik, Péter Kőrösi-Szabó, Ákos K. Matszangosz, Gergely Papp, and Dániel Varga. Similarity and matching of neural network representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5656–5668, 2021. 2, 3

[10] Alex Fang, Albin Madappally Jose, Amit Jain, Ludwig Schmidt, Alexander Toshev, and Vaishaal Shankar. Data filtering networks. *arXiv preprint arXiv:2309.17425*, 2023. 3

[11] Enrico Fini, Mustafa Shukor, Xiujun Li, Philipp Dufter, Michal Klein, David Haldimann, Sai Aitharaju, Victor G Turrisi da Costa, Louis Béthune, Zhe Gan, et al. Multimodal autoregressive pre-training of large vision encoders. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9641–9654, 2025. 4

[12] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. In *arXiv preprint arXiv:2306.13394*, 2023. 1, 2

[13] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. 8

[14] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017. 8, 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2

[16] Xinyu Hou, Peng Chen, and Haishuo Gu. Lm-deeplabv3+: A lightweight image segmentation algorithm based on multi-scale feature interaction. *Applied Sciences*, 14(4):1558, 2024. 6

[17] 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 2021. *arXiv preprint arXiv:2106.09685*, 10, 2021. 6

[18] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022. 6

[19] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019. 2

[20] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. 8

[21] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, et al. Openclip. https://github.com/mlfoundations/open_clip, 2021. 2, 3

[22] Christoph Kamann and Carsten Rother. Benchmarking the robustness of semantic segmentation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 6

[23] Tommie Kerssies, Daan De Geus, and Gijs Dubbelman. How to benchmark vision foundation models for semantic segmentation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1162–1171, 2024. 6, 2

[24] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 7

[25] Max Klabunde, Tobias Schubert, and Sebastian Lapuschkin. Similarity of neural network models revisited: Measuring functional similarity. In *International Conference on Machine Learning (ICML)*, 2024. 2

[26] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network represen-

tations revisited. In *International conference on machine learning*, pages 3519–3529. PMlR, 2019. 2

[27] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017. 2

[28] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2

[29] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[30] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. 6

[31] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 2

[32] Haotian Liu, Chunyuan Li, Yong Jae Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2024. 4, 7, 8, 2

[33] Zheda Mai, Arpita Chowdhury, Ping Zhang, Cheng-Hao Tu, Hong-You Chen, Vardaan Pahuja, Tanya Berger-Wolf, Song Gao, Charles Stewart, Yu Su, et al. Fine-tuning is fine, if calibrated. *Advances in neural information processing systems*, 37:136084–136119, 2024. 1

[34] Zheda Mai, Arpita Chowdhury, Zihe Wang, Sooyoung Jeon, Lemeng Wang, Jiacheng Hou, and Wei-Lun Chao. Ava-bench: Atomic visual ability benchmark for vision foundation models. *arXiv preprint arXiv:2506.09082*, 2025. 3

[35] Zheda Mai, Ping Zhang, Cheng-Hao Tu, Hong-You Chen, Quang-Huy Nguyen, Li Zhang, and Wei-Lun Chao. Lessons and insights from a unifying study of parameter-efficient fine-tuning (peft) in visual recognition. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14845–14857, 2025. 1, 6

[36] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. http://www.robots.ox.ac.uk/~vgg/data/fgvc-aircraft/, 2013. arXiv preprint arXiv:1306.5151. 6, 1

[37] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 2, 3, 1

[38] Zizheng Pan, Bohan Zhuang, Haoyu He, Jing Liu, and Jianfei Cai. Stitchable neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16041–16050, 2023. 3

[39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. 1, 2, 3

[40] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025. 2, 1

[41] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019. 2

[42] Damian Smith, Harvey Mannering, and Antonia Marcu. Functional alignment can mislead: Examining model stitching. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. 2

[43] Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. Eva-clip: Improved training techniques for clip at scale. *arXiv preprint arXiv:2303.15389*, 2023. 3

[44] Peter Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Adithya Jairam Vedagiri IYER, Sai Charitha Akula, Shusheng Yang, Jihan Yang, Manoj Middepogu, Ziteng Wang, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356, 2024. 7, 4

[45] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. *arXiv preprint arXiv:2401.06209*, 2024. 1, 3, 7, 2

[46] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025. 3, 1, 4

[47] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8769–8778, 2018. 6, 1

[48] Xingyi Yang, Daquan Zhou, Songhua Liu, Jingwen Ye, and Xinchao Wang. Deep model reassembly. *Advances in neural information processing systems*, 35:25739–25753, 2022. 3

[49] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language-image pre-training. *arXiv preprint arXiv:2303.15343*, 2023. 1, 2, 3

[50] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Vision (CVPR)*, 2017. 6, 1

[51] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic under-

standing of scenes through the ade20k dataset. In *International Journal of Computer Vision*, 2019. 6, 1

[52] Baichuan Zhou, Ying Hu, Xi Weng, Junlong Jia, Jie Luo, Xien Liu, Ji Wu, and Lei Huang. Tinyllava: A framework of small-scale large multimodal models. *arXiv preprint arXiv:2402.14289*, 2024. 2

Revisiting Model Stitching in the Foundation Model Era

Supplementary Material

We provide details omitted in the paper.

- Sec. A: Experiment and dataset details
- Sec. B: Detailed results

A. Experiment and Dataset Details

A.1. Vision Foundation Models (VFMs)

We use the following VFMs in our experiments.

- **DINOv2-L** [37]: A ViT-L/14 encoder trained with self-supervised DINOv2 on a curated collection of ∼142M internet images, using a teacher–student distillation objective that encourages consistency across multi-crop image views without labels. In our experiments, we feed 336×336 inputs with patch size 14.
- **CLIP-L** [39]: A ViT-L/14 vision encoder from CLIP, trained jointly with a text encoder on ∼400M image–text pairs from the web using a contrastive objective that pulls matched image–caption pairs together and pushes mismatched pairs apart in a shared embedding space. We use the vision tower with 336×336 inputs and patch size 14.
- **SigLIP2-L** [46]: A ViT-L/16 multilingual vision–language encoder trained on a diverse mixture of web image–text data with a sigmoid-based image–text matching loss, augmented with captioning-style pretraining, self-distillation, masked prediction, and online data curation to improve dense features and localization. We use 384×384 inputs and patch size 16.
- **DINOv3-L** [40]: A ViT-L/16 self-supervised encoder from the DINOv3 family, trained at large scale on a multi-domain image collection with a distillation-style objective and Gram anchoring to stabilize dense feature maps, yielding strong frozen representations for both global and dense tasks. In our experiments, we use 384×384 inputs and patch size 16.

A.2. Fine-Grained Classification

A.2.1. Dataset Details

fMoW [7] The Functional Map of the World (fMoW) dataset is a commonly used and challenging dataset for VFM evaluation. It is a collection of high-quality remote-sensing satellite images collected worldwide, annotated with 62 land-use and functional categories (e.g., airfield, port, hospital). We use the fMoW-RGB variant, which provides pan-sharpened RGB crops and associated metadata. Since the original dataset is highly imbalanced, we construct a class-balanced subset with 230 training and 26 test images per class (14,260 train / 1,612 test) so that our study can focus on model stitching itself rather than confounding effects from label imbalance, which would complicate the interpretation of stitching behavior.

iNaturalist-Subset [47] The iNaturalist 2021 dataset family has become a standard testbed for assessing VFM performance on fine-grained, real-world biodiversity recognition. It is a large-scale and heavily imbalanced species classification benchmark that reflects naturally occurring long-tailed distributions. To obtain a controlled yet challenging setting for analyzing stitching, we sample 106 species from three visually similar Lepidoptera families (Sphingidae, Pieridae, Pyralidae) and rebalance the data with 200 training and 50 test images per class (21,200 train / 5,300 test). This balanced construction removes imbalance as an additional variable, enabling a cleaner analysis of how stitching choices impact performance.

FGVC-Aircraft [36] FGVC-Aircraft is widely adopted as a canonical fine-grained classification benchmark for VFMs. The version we use contains 102 aircraft models and approximately 10k images, with each model appearing in around 100 images. We follow the standard split with 6.6k training and 3.3k test images, which requires distinguishing subtle variations in shape, livery, and viewpoint.

A.2.2. Training Details

For all fine-grained classification experiments, we adopt linear probing: a single linear classifier is trained on pooled image features while all VFM parameters remain frozen.

For *layer feature matching*, we first extract intermediate features from both source and target VFMs and train the stitch MLP purely on these features; no additional VFM forward passes are required during this phase. For *final feature matching*, we extract features from the target model and train only the stitch layer.

To isolate the effect of stitching, we do not apply any data augmentation. We use the AdamW optimizer in all experiments, train for up to 100 epochs with early stopping (patience of 5 epochs), and tune the learning rate in $\{0.001, 0.005, 0.01\}$. For layer feature matching, we use a batch size of 256; all other configurations use a batch size of 128. Training is performed with automatic mixed precision using `bfloat16`.

A.2.3. Semantic Segmentation

A.2.4. Dataset Details

ADE20K [50, 51] ADE20K is a scene-centric semantic segmentation dataset with pixel-level annotations for

150 object and stuff categories across diverse indoor and outdoor environments. We adopt the canonical split with 20,210 training images and 3,000 held-out test images. The dense annotations cover scenes, objects, and parts, making ADE20K a challenging benchmark for evaluating dense prediction performance.

A.2.5. Training Details

For semantic segmentation, we train a linear layer to predict per-pixel class logits for each patch token. The linear layer produces a low-resolution logit map (e.g., 24×24 for a model with patch size 14 and 336×336 input), which is then bilinearly upsampled to the original image resolution to obtain the final segmentation map.

For layer feature matching and final feature matching, we reuse the optimization and hyperparameter settings described for classification (optimizer, learning rates, batch sizes, early stopping, and mixed precision). All remaining experimental details for segmentation follow [23].

A.3. VFM Stitch Tree for Multimodal LLM

A.3.1. Dataset Details

We use the LLaVA-1.5 training data prepared by TinyLLaVA.[1] LLaVA-1.5-Pretrain (PT) consists of 558k image–caption pairs [32], while LLaVA-1.5-SFT comprises 665k visual instruction-tuning conversations that combine academic-style VQA [14, 19, 27, 41] samples with instruction-tuning data from LLaVA-Instruct [31]. In our preliminary study, we evaluate on VQA-v2 [14] and MME [12].

A.3.2. Training Details

We jointly use LLaVA-1.5-Pretrain and LLaVA-1.5-SFT to train the stitch layer for one epoch with a learning rate of 0.001 and batch size 64. All remaining hyperparameters follow the TinyLLaVA recipe [52]. We adopt Qwen2.5-3B [2] as the LLM and employ an interleaved mixture-of-features (MoF) [45] strategy when combining visual tokens from CLIP and DINOv2.

B. Detailed and Additional Results

B.1. Self-Stitch

To rigorously test whether gains only stem from added capacity, we introduce a Self-Stitch baseline: inserting the identical stitch layer into the source-only and target-only models at the same stitch positions. Fig. 10 illustrates the difference between model stitching and the self-stitching baseline.

[1]We use the data preparation pipeline from TinyLLaVA: https://tinyllava-factory.readthedocs.io/en/latest/Prepare%20Datasets.html

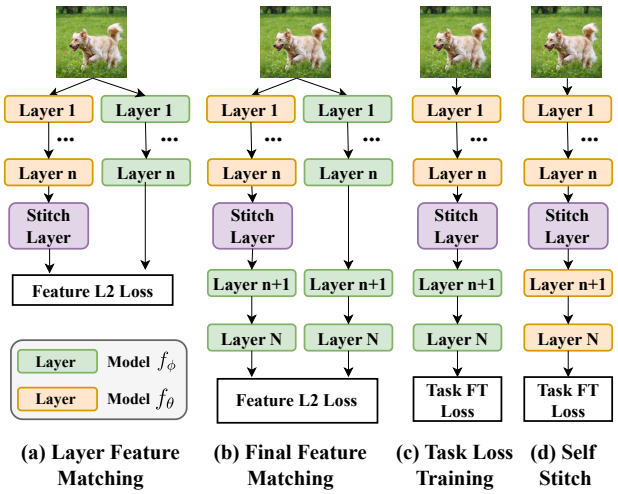

Figure 10. Model stitching training strategies: (a) Layer feature matching trains the stitch layer to match features between the source and target models at the stitch position. (b) Final feature matching trains the stitch layer so that the stitched model matches the target model's final features. (c) Task loss training optimizes the stitch layer with the downstream task objective. (d) Self-Stitch inserts the stitch layer into the source-only and target-only models at the same stitch position. Across all strategies, the stitch layer is the only trainable component; the source and target models are kept frozen.

B.2. Layer and Final Feature Matching

We present a detailed comparison between Layer and Final Feature Matching in Tab. 6. We observe that Final Feature Matching consistently outperforms Layer Feature Matching across all configurations. Notably, in the DINOv2 $\rightarrow$ SigLIP2 setting, the stitched model surpasses the performance of both constituent models. Our results further indicate that stitching a lower-performing source model to a higher-performing target is more effective than the reverse; DINOv2 $\rightarrow$ SigLIP2 consistently achieves higher accuracy than SigLIP2 $\rightarrow$ DINOv2, regardless of the matching strategy employed.

B.3. Stitching Between Different VFMs

To verify the generality of our findings, we extend our analysis beyond DINOv2 and SigLIP2 to include the widely adopted CLIP and the recently released DINOv3. Detailed comparisons between Stitched models, Self-Stitch baselines, and Linear Probing are provided in Tabs. 7 to 11. Consistent with our previous results, stitched models generally outperform their corresponding self-stitch baselines across most layers. The primary exception arises when CLIP is utilized as the source model; we discuss the potential causes for this phenomenon in the main paper.

	DINOv2			SigLIP2		
Linear Probing	46.7			53.5		
Layer	2	6	10	14	18	22
Layer Feature Matching						
DINOv2 → SigLIP2	49.4	48.2	51.0	48.5	45.3	48.2
SigLIP2 → DINOv2	29.9	32.4	26.9	42.4	43.7	47.2
Final Feature Matching						
DINOv2 → SigLIP2	51.4	53.1	56.1	51.0	55.7	54.7
SigLIP2 → DINOv2	46.4	47.1	43.3	47.3	48.8	50.8

Table 6. **Comparison of Layer and Final Feature Matching.** We report linear probing accuracy for different stitching strategies. Orange indicates performance surpassing the SigLIP2 baseline, while Blue indicates performance surpassing the DINOv2 baseline.

	DINOv2		CLIP	
Linear Probing	46.7		46.4	
Layer	6	14	18	22
DINOv2 → DINOv2	41.5	59.7	68.2	69.9
CLIP → CLIP	48.5	63.1	59.6	60.7
CLIP → DINOv2	53.1	65.8	63.3	64.0
DINOv2 → CLIP	**54.1**	**68.1**	**72.0**	**71.9**

Table 7. Stitched Model vs Self-Stitch vs Linear Probing: DINOv2 and CLIP.

	DINOv2		SigLIP2	
Linear Probing	46.7		53.5	
Layer	6	14	18	22
DINOv2 → DINOv2	41.5	59.7	68.2	69.9
SigLIP2 → SigLIP2	50.5	62.0	69.4	68.9
SigLIP2 → DINOv2	53.8	**69.6**	70.4	**72.2**
DINOv2 → SigLIP2	**55.8**	68.0	**72.0**	71.8

Table 8. Stitched Model vs Self-Stitch vs Linear Probing: DINOv2 and SigLIP2.

B.4. Prediction Analysis

To better understand how knowledge fusion affects prediction behavior, we compare each stitched model (DINOv2→SigLIP2 and SigLIP2→DINOv2) against the two self-stitched baselines (DINOv2→DINOv2 and SigLIP2→SigLIP2) on fMoW and iNaturalist. We discard trivial cases where all three models are correct or all three are wrong, and partition the remaining examples into the scenarios in Tab. 12:

1. **Preserve (Cols. 1–2).** At least one self-stitched model is correct and the stitched model is also correct, preserving the correct prediction.
2. **Rescue (Col. 3).** Both self-stitched models are wrong but the stitched model is correct.
3. **Interference (Cols. 4–6).** At least one self-stitched model is correct but the stitched model becomes wrong.

Across both datasets and stitch directions, the total count of *preserve* and *rescue* scenarios clearly exceeds that of *interference* scenarios, indicating that the stitched models benefit more from fusing complementary signals than they suffer from conflicts between the two backbones.

We further ask whose behavior the stitched model tends to follow when the two self-stitched models disagree. On fMoW, where the two self-stitched accuracies are similar, the stitched models are more likely to match the *source* model: for SigLIP2→DINOv2, the number of cases where the stitched prediction agrees with SigLIP2→SigLIP2 substantially exceeds the cases where it agrees with DINOv2→DINOv2, and an analogous trend

	SigLIP2		CLIP	
Linear Probing	53.5		46.4	
Layer	6	14	18	22
SigLIP2 → SigLIP2	50.5	62.0	69.4	68.9
CLIP → CLIP	48.5	63.1	59.6	60.7
CLIP → SigLIP2	48.3	68.9	65.4	62.9
SigLIP2 → CLIP	**59.8**	**70.7**	**73.2**	**71.9**

Table 9. Stitched Model vs Self-Stitch vs Linear Probing: SigLIP2 and CLIP.

	DINOv3		SigLIP2	
Linear Probing	50.0		53.5	
Layer	6	14	18	22
DINOv3 → DINOv3	44.7	63.9	66.9	69.0
SigLIP2 → SigLIP2	50.5	62.0	69.4	68.9
SigLIP2 → DINOv3	**58.6**	67.0	69.4	69.3
DINOv3 → SigLIP2	45.0	**69.4**	**70.5**	**72.4**

Table 10. Stitched Model vs Self-Stitch vs Linear Probing: DINOv3 and SigLIP2.

	DINOv2		DINOv3	
Linear Probing	46.7		50.0	
Layer	6	14	18	22
DINOv2 → DINOv2	41.5	59.7	68.2	69.9
DINOv3 → DINOv3	44.7	63.9	66.9	69.0
DINOv3 → DINOv2	43.8	**67.2**	67.9	**72.0**
DINOv2 → DINOv3	**57.8**	65.0	**69.2**	70.3

Table 11. Stitched Model vs Self-Stitch vs Linear Probing: DINOv2 and DINOv3.

fMoW

	Scenario						Acc.
DINOv2→DINOv2	W	R	W	R	R	W	69.9
SigLIP2→SigLIP2	R	W	W	R	W	R	68.9
SigLIP2→DINOv2	R	R	R	W	W	W	72.2
Count	116	78	47	40	86	33	

	Scenario						Acc.
DINOv2→DINOv2	W	R	W	R	R	W	69.9
SigLIP2→SigLIP2	R	W	W	R	W	R	68.9
DINOv2→SigLIP2	R	R	R	W	W	W	71.8
Count	77	116	54	52	48	72	

iNaturalist

	Scenario						Acc.
DINOv2→DINOv2	W	R	W	R	R	W	91.2
SigLIP2→SigLIP2	R	W	W	R	W	R	87.3
SigLIP2→DINOv2	R	R	R	W	W	W	91.9
Count	138	292	75	68	116	53	

	Scenario						Acc.
DINOv2→DINOv2	W	R	W	R	R	W	91.2
SigLIP2→SigLIP2	R	W	W	R	W	R	87.3
DINOv2→SigLIP2	R	R	R	W	W	W	92.8
Count	127	331	80	49	77	64	

Table 12. Analysis of Predictions: R and W denote Right and Wrong predictions for each model, with counts in the last row and accuracies in the last column for fMoW and iNaturalist. Stitched models fall into complementarity scenarios (stitched is correct while at least one self-stitched baseline is wrong) far more often than interference ones (stitched is wrong while at least one self-stitched baseline is correct). When the two self-stitched models disagree, the stitched model tends to follow the source model on fMoW but follows the stronger model DINOv2 on iNaturalist.

holds for DINOv2→SigLIP2. In contrast, on iNaturalist, where DINOv2→DINOv2 is the stronger self-stitched model, both stitch directions tend to align with the stronger model's predictions regardless of whether it is used as source or target.

Overall, these analyses support our interpretation that stitching primarily acts as a knowledge-fusion mechanism: it preserves or rescues correct predictions from at least one backbone much more often than it disrupts them, and when there is a strong/weak asymmetry, the stitched model gravitates toward the stronger expert's decisions.

B.5. VFM Stitch Tree (VST) for Multimodal LLM (MLLM)

B.5.1. Computation Comparison

Table 13 compares the computational cost of running all VFMs independently (Full) versus our VST method at various stitch positions. We illustrate the efficiency gains using Cambrian-1 with **4** VFMs [44]. Assuming a standard 24-layer ViT-L architecture [46], and noting that MLLMs typically extract features from the second last layer (layer 23) [32], we base our calculations on a 23-layer depth. While the "Full" setting incurs a 300% computational overhead compared to a single VFM, VST-14 significantly reduces this burden. By sharing the first 14 layers and maintaining specialized branches only from layer 15 onwards, VST-14 requires processing only $3 \times (23 - 14) = 27$ additional layers. This results in an overhead of just $27/23 \approx 117\%$, demonstrating a substantial efficiency improvement over the independent baseline.

Number of VFMs	Full	VST-6	VST-14	VST-22
2	100%	74%	39%	4%
3	200%	148%	78%	9%
4	300%	222%	117%	13%
5	400%	296%	156%	17%

Table 13. **Comparison of Additional Computation Cost.** We report the additional increase in computation compared to a single VFM. "Full" denotes running all VFMs independently. "VST-n" denotes a shared backbone up to the stitch position n, where layers 1 through n are shared and subsequent layers are executed independently. For example, VST-14 shares the first 14 layers and maintains specialized branches from layer 15 onwards.

B.6. VST as an Accuracy-Efficiency Knob

We introduce the concept of "Normalized Gain" to quantify the effectiveness of VST relative to running all VFMs independently. Table 14 provides the step-by-step derivation of these values for both VST-22 and VST-14.

We define the Normalized Gain as the ratio between the performance improvement achieved by VST (Δ_{VST}) and the maximum improvement achieved by running all VFMs independently (Δ_{max}):

$$\text{Normalized Gain (\%)} = \frac{\Delta_{\text{VST}}}{\Delta_{\text{max}}} = \frac{\text{Perf}_{\text{VST}} - \text{Perf}_{\text{CLIP}}}{\text{Perf}_{\text{Full}} - \text{Perf}_{\text{CLIP}}} \quad (2)$$

As illustrated in Tab. 14:

- The Orange Row represents the **Denominator** (Δ_{max}): the total performance gap between the single-model baseline and the computationally expensive naive full running

Model Configuration	VQAv2			MME		Efficiency Metrics	
	Yes/No	Number	Other	Percep.	Cogn.	Avg Gain %	Add. Cost
1. Define the Upper Bound (Denominator)							
(A) CLIP Baseline	91.75	58.74	69.00	1418.5	277.1	-	0%
(B) Full (CLIP + DINOv2)	92.72	61.64	70.30	1460.3	311.8	-	100%
(C) Max Gain ($\Delta_{\text{max}} = B - A$)	**0.97**	**2.90**	**1.30**	**41.8**	**34.7**	**(Denom.)**	**-**
2. VST-22: The Lightweight Knob (High Efficiency)							
(D) VST-22 (Ours)	92.12	59.21	69.15	1451.6	305.7	-	**4.3%**
(E) VST Gain ($\Delta_{\text{VST}} = D - A$)	**0.37**	**0.47**	**0.15**	**33.1**	**28.6**	**(Num.)**	**-**
Normalized Gain ($\% = E/C$)	38.1%	16.2%	11.5%	79.1%	82.5%	**45.5%**	**4.3%**
3. VST-14: The Balanced Knob (High Performance)							
(F) VST-14 (Ours)	92.54	60.69	69.88	1474.4	301.8	-	39.0%
(G) VST Gain ($\Delta_{\text{VST}} = F - A$)	**0.79**	**1.95**	**0.88**	**55.9**	**24.7**	**(Num.)**	**-**
Normalized Gain ($\% = G/C$)	81.4%	67.2%	67.7%	133.7%	71.1%	**84.2%**	39.0%

Table 14. **Detailed Calculation of Normalized Gain.** Comparison of the "Lightweight" VST-22 and the "Balanced" VST-14. The Orange Row represents the gain (Δ_{max}) achieved by the computationally expensive Naive ensemble (100% Cost). The Pink Rows represent the actual gain (Δ_{VST}) achieved by our method. The **Normalized Gain** is calculated as $\Delta_{\text{VST}}/\Delta_{\text{max}}$. Note that VST-22 achieves nearly half the total gain (45.5%) with negligible cost (4.3%).

(100% additional cost).

- The Pink Rows represent the **Numerator** (Δ_{VST}): the actual gain realized by our VST method.

The results highlight the versatility of VST as a compute-aware knob. **VST-22** serves as an ultra-lightweight option, recovering **45.5%** of the total possible gain while incurring a negligible **4.3%** increase in backbone cost. Conversely, **VST-14** acts as a balanced high-performance option, recovering **84.2%** of the total gain for only **39%** of the additional cost. This granular control allows practitioners to maximize model utility within specific computational budgets.

Consequently, we view VST as a vital *accuracy–efficiency knob* for architecture design. Without VST, practitioners face a rigid binary choice: either deploy an entire additional VFM (yielding higher performance but incurring **100%** extra backbone cost) or deploy none (yielding no gain). VST transforms this discrete step function into a continuous spectrum. By serving as a compute-aware knob, our method effectively interpolates between these extremes, enabling controllable performance–efficiency trade-offs that can be dynamically tuned to meet strict real-world deployment constraints.