(Almost) Free Modality Stitching of Foundation Models

Jaisidh Singh[1,2,4], **Diganta Misra**[3,4], **Boris Knyazev**[5,6], **Antonio Orvieto**[3,4,7]
[1]University of Tübingen, [2]Zuse School ELIZA [3]ELLIS Institute Tübingen, [4]MPI-IS Tübingen, [5]Samsung – SAIT AI Lab Montréal, [6]Université de Montréal, [7]Tübingen AI Center

Abstract

Foundation multi-modal models are often designed by stitching of multiple existing pretrained uni-modal models: for example, an image classifier with a text model. This *stitching* process is performed by training a *connector* module that aims to align the representation spaces of these uni-modal models towards a multi-modal objective. However, given the complexity of training such connectors on large scale web-based datasets coupled with the ever-increasing number of available pretrained uni-modal models, the task of uni-modal models selection and subsequent connector module training becomes computationally demanding. To address this under-studied critical problem, we propose **Hypernetwork Model Alignment (HYMA)**, a novel all-in-one solution for optimal uni-modal model selection and connector training by leveraging hypernetworks. Specifically, our framework utilizes the parameter prediction capability of a hypernetwork to obtain jointly trained connector modules for $N \times M$ combinations of uni-modal models. In our experiments, HYMA reduces the cost of searching for the best performing uni-modal model pair by $10\times$, while matching the ranking and trained connector performance obtained via grid search across a suite of diverse multi-modal benchmarks.

1 Introduction

Multi-modal foundation models have emerged as a new frontier in the Artificial Intelligence (AI) landscape. Fueled by the increasing need for considering inter-dependency of multiple data modalities in modern tasks, multi-modal foundation models often leverage modality-specific (uni-modal) models as sub-components, which are stitched together via a *connector* module. A prominent class of such models is Vision-Language Models (VLMs) (Radford et al., 2021; Singh et al., 2024; Li et al., 2022; Singh et al., 2022), which comprise image and text encoders that embed image and text concepts into a common contrastively learnt latent space.

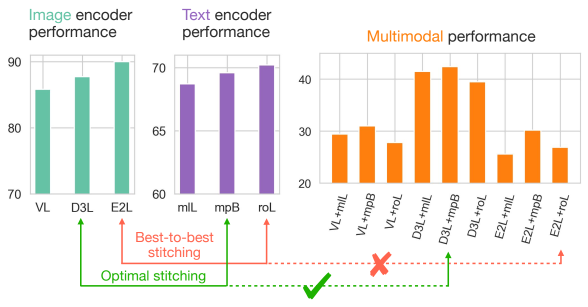
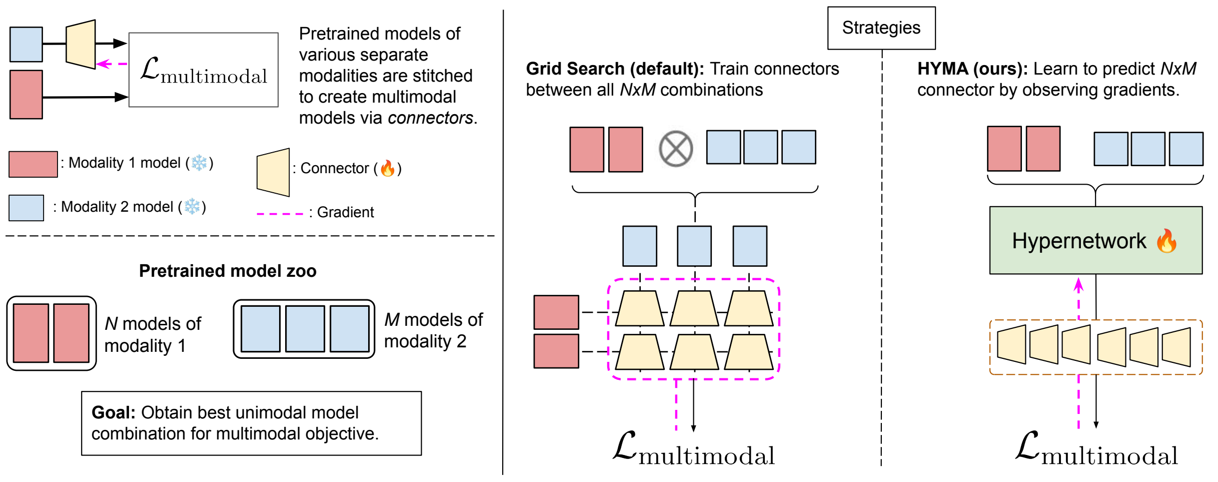


Figure 1: We train connectors between pretrained uni-modal models to show that uni-modal model performance is **not predictive** of multi-modal performance obtained by stitching. Image encoder performance refers to top-1 ImageNet-1K accuracy, text encoder performance refers to semantic search performance across 14 datasets (Reimers and Gurevych, 2019). Multi-modal scores refers to ImageNet-1K top-1 accuracy (classification by matching images to prompts such as "*this is a photo of a* `{class}`").[1]

Connector modules powering VLMs are often constructed as an n-layer multi-layer perceptron (MLP) (Liu et al., 2024), or in some cases even as simple as a linear layer (Merullo et al., 2022), with the purpose of stitching modality-specific models. While some exceptions do arise where these modules are extensively engineered transformer-like architectures (Li et al., 2023), the vast majority consensus on the design of such connector modules has been limited to MLPs (Zhu et al., 2025) due to their efficiency.

While training connector modules for a pair of predetermined uni-modal models is feasible, the picture becomes more complex when considering multiple uni-modal options and aiming to optimize for downstream performance after stitching. Indeed, it is often not the case (see Figure 1) that simply choosing to align best-performing uni-modal models leads to the best multi-modal performance.

[1]All model abbreviations can be found in Appendix B.



$\mathcal{L}_{\text{multimodal}}$
Pretrained models of various separate modalities are stitched to create multimodal models via *connectors*.
: Modality 1 model
: Modality 2 model
: Connector
: Gradient
Pretrained model zoo
N models of modality 1
M models of modality 2
Goal: Obtain best unimodal model combination for multimodal objective.
Strategies
Grid Search (default): Train connectors between all *NxM* combinations
HYMA (ours): Learn to predict *NxM* connector by observing gradients.
Hypernetwork
$\mathcal{L}_{\text{multimodal}}$
$\mathcal{L}_{\text{multimodal}}$

Figure 2: Given multiple options for uni-modal models, pair-wise grid search can be an expensive way to determine the best multi-modal combination. Alternatively, HYMA formulates search as a predictive or generative process.

This trend is further illustrated in Table 1, where uni-modal model parametric capacity fails to serve as a reliable predictor of multi-modal performance. Consequently, the cost of optimal stitching can grow quadratically with the number of available options on both ends. In addition, the availability of extremely large web-scale pretraining datasets, consisting of samples in the order of billions (Schuhmann et al., 2022; Changpinyo et al., 2021; Desai et al., 2021), constitutes a blocker for proper ablation on such design choices.

I (#Params)	T (#Params)	Total #Params	Perf.
EVA2-L (305M)	roberta-L (355M)	**660M** $+c$	26.85
DeiT3-L (304M)	mpnet-B (109M)	413M $+c$	**42.63**

Table 1: **Parametric capacity of unimodal models is not a reliable indicator of multimodal performance**. On the task of multi-modal image classification using the ImageNet-1k dataset, we observe that stitching the highest-capacity models: EVA-2 Large (305M) for the image modality (**I**) and RoBERTa Large (355M) for the text modality (**T**), totaling 660M + c parameters—yields significantly lower performance than a smaller stitched pair: DeiT-3 Large (**I**) (304M) and MPNet-Base (**T**) (109M), totaling just 413M + c parameters. c denotes the parameters contributed by 1-hidden layer MLP connector and **Perf.** denotes the Top-1 accuracy metric.

We highlight and define the problem, which we term **Multi-modal Optimal Pairing and Stitching (M-OPS)**, as:

- **Pairing**: Given a set of N models in modality 1 (e.g., vision) and M models in modality 2 (e.g., text), provide the optimal (best performing) combination pair $(n, m;\ n \in N \mid m \in M)$ to construct a multi-modal model for a target task and/or under target constraints (e.g., parametric size, embedding dimensions).
- **Stitching**: For the selected uni-modal models (n, m), obtain the optimal trained connector f_θ that stitches them to construct the target multi-modal model.

Due to the infeasibility of addressing the **pairing** sub-problem of **M-OPS** via a grid-search approach for a large $N \times M$ pair, we propose a novel alternative approach to tackle both the pairing and stitching steps in a single unified manner that utilizes a HyperNetwork (Ha et al., 2016). The key idea behind our approach is that stitching similar models shares latent semantics, which can be captured by jointly training a network to generate connectors.

We present **Hypernetwork Model Alignment (HYMA)**[2], a method that, given N modality 1 (e.g., image) and M modality 2 (e.g., text) models, leverages a hypernetwork (Ha et al., 2016) that jointly learns to generate connectors for all possible $N \times M$ combinations. Our approach serves both as an indicator for optimal model pair configurations and as a trainer that produces *stitched* multi-modal models performing on par with the best stitched model pair obtained via grid search. In our experiments, where $N \times M$ can be as high as 27 (discussed in Section 5), our method enables an efficiency gain of $10\times$ in obtaining the best stitched model pair compared to grid search.

We highlight our contributions as follows:

[2]https://github.com/jaisidhsingh/hyma

1. We propose **Hypernetwork Model Alignment (Hyma)**, a hypernetwork-based approach for obtaining strong uni-modal model pairs that perform on par with the best stitched model pair obtained via grid search at an order-of-magnitude lower computational cost.

2. Our proposed approach **Hyma** is, to the best of our knowledge, the first to demonstrate the effectiveness of hypernetworks for solving the **M-OPS** problem defined above.

3. We empirically demonstrate the performance and efficiency of **Hyma** on VLMs across various multi-modal benchmarks.

2 Background

In this section, we present the necessary preliminaries for the **M-OPS** problem, along with the general training paradigm of hypernetworks. These formal definitions establish the foundation for our proposed method, Hyma, which we introduce in the following section.

Definition 1 (Hypernetworks for Parameter Prediction)**.** *A hypernetwork (Ha et al., 2016) is a neural network H_ϕ parameterized by ϕ, designed to predict the parameters θ of a target network f_θ based on a conditioning input $\mathbf{c}$. The parameter generation process is defined as:*

$$H_\phi(\mathbf{c}) = \theta.$$

The parameters ϕ of the hypernetwork are optimized indirectly via the performance of the generated network f_θ on a downstream task. Given a task-specific loss $\mathcal{L}_{task}$ evaluated on corresponding data, the optimization objective becomes:

$$\phi^* = \arg\min_\phi \mathcal{L}_{task}(f_{H_\phi(\mathbf{c})}).$$

The trained hypernetwork H_{ϕ^*} can then be used to generate task-adapted parameters θ for f given new conditioning inputs. Optimizing ϕ rather than θ directly can offer advantages in terms of training dynamics, capacity control, and generalization (Chauhan et al., 2024).

For simplicity, assume encoders producing sequences of P features (e.g., number of patches or tokens) living in a D-dimensional space.

Definition 2 (Connector-based multi-modal stitching)**.** *Let $\mathcal{A} : \mathcal{X}_A \to \mathbb{R}^{D_A}$ and $\mathcal{B} : \mathcal{X}_B \to \mathbb{R}^{D_B}$ be pretrained uni-modal encoders for two different modalities with input spaces $\mathcal{X}_A$ and $\mathcal{X}_B$, respectively. The goal is to construct a multi-modal model by learning a connector function $f_\theta : \mathbb{R}^{D_A} \to \mathbb{R}^{D_B}$ that stitches the output of $\mathcal{A}$ to the representation space of $\mathcal{B}$: given input pairs $(\mathbf{u}, \mathbf{v}) \in \mathcal{X}_A \times \mathcal{X}_B$, the connector stitches the modality-A features*

$$\mathbf{x}^a = \mathcal{A}(\mathbf{u}) \in \mathbb{R}^{D_A}$$

to modality-B space via

$$\tilde{\mathbf{x}}^a = f_\theta(\mathbf{x}^a) \in \mathbb{R}^{D_B}$$

The stitched representation $\tilde{\mathbf{x}}^a$ is then combined with $\mathbf{x}^b = \mathcal{B}(\mathbf{v})$ to construct a joint multi-modal representation. The connector parameters θ are optimized while keeping $\mathcal{A}$ and $\mathcal{B}$ frozen. The training objective follows ***contrastive stitching****, that uses a similarity function sim$(\cdot,\cdot)$ and temperature τ to train the connector on the InfoNCE (Oord et al., 2018) loss (quadratic):*

$$\mathcal{L}_{contrastive}(\theta) = -\log \frac{\exp(sim(\tilde{\mathbf{x}}^a, \mathbf{x}^b)/\tau)}{\sum_j \exp(sim(\tilde{\mathbf{x}}^a, \mathbf{x}^b_j)/\tau)}$$

3 Methodology

3.1 Problem formulation

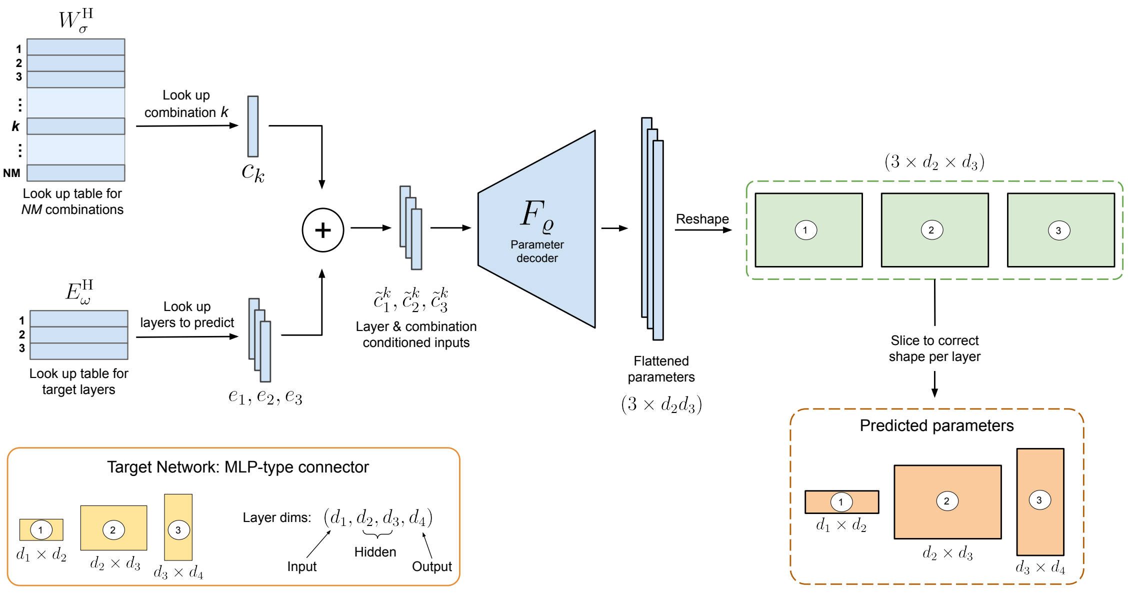
We aim to jointly learn $N \times M$ connectors, where each connector is specified to the hypernetwork via a conditional input $\mathbf{c}^k$. More formally, for the k^{th} model combination, the hypernetwork generates the parameters as $H_\phi(\mathbf{c}^k)$. The resulting connector $f_{H_\phi(\mathbf{c}^k)}$ is then used to compute a task-specific loss. The overall training loss is computed by averaging over all combinations:

$$\mathcal{L}_{\text{HYMA}} = \frac{1}{NM} \sum_{k=1}^{NM} \mathcal{L}_{\text{task}}(f_{H_\phi(\mathbf{c}^k)}). \quad (1)$$

Here, $\mathcal{L}_{\text{task}}$ corresponds to a contrastive InfoNCE loss (for retrieval-style objectives like that in CLIP (Radford et al., 2021)). The trained hypernetwork is denoted by H_{ϕ^*}, where $\phi^* = \arg\min_\phi \mathcal{L}_{\text{HYMA}}$. Following prior work (Rosenfeld et al., 2022; Jia et al., 2024), we restrict connectors to be multi-layer perceptrons (MLPs).

3.2 Hypernetwork architecture

We define the hypernetwork as a function $H_\phi : \mathbb{R}^C \to \mathbb{R}^{D_\theta}$, mapping conditional inputs $\mathbf{c} \in \mathbb{R}^C$ to connector parameters $\theta \in \mathbb{R}^{D_\theta}$. We describe next how $\mathbf{c}$ is constructed and how it is mapped to the parameter space.



W_σ^{H}
Look up table for NM combinations
Look up combination k
c_k
E_ω^{H}
Look up table for target layers
Look up layers to predict
e_1, e_2, e_3
$\tilde{c}_1^k, \tilde{c}_2^k, \tilde{c}_3^k$
Layer & combination conditioned inputs
F_ϱ
Parameter decoder
Flattened parameters
$(3 \times d_2 d_3)$
Reshape
$(3 \times d_2 \times d_3)$
Slice to correct shape per layer
Predicted parameters
$d_1 \times d_2$
$d_2 \times d_3$
$d_3 \times d_4$
Target Network: MLP-type connector
Layer dims: (d_1, d_2, d_3, d_4)
Input
Hidden
Output

Figure 3: A visual walkthrough of our hypernetwork architecture is provided above. We take the example of predicting the parameters of an MLP-type connector with depth= 3 (denotes 2 hidden layers).

Conditional inputs: We use a learnable lookup table of embeddings $\mathbf{W}_\sigma^{\mathrm{H}} \in \mathbb{R}^{NM \times C}$, where $\mathbf{c}^k = \mathbf{W}_\sigma^{\mathrm{H}}[k]$ encodes the k^{th} model pair.

Mapping conditional inputs to parameters: The hypernetwork H_ϕ is implemented using an MLP F_ϱ, which predicts connector parameters layer-wise. Each layer prediction is conditioned on both $\mathbf{c}^k$ and a learnable layer-specific embedding $\mathbf{e}_j = \mathbf{E}_\omega^{\mathrm{H}}[j]$, such that:

$$F_\varrho(\tilde{\mathbf{c}}_j^k) \in \mathbb{R}^{D_{\vartheta^k}}, \quad \text{where} \quad \tilde{\mathbf{c}}_j^k = \mathbf{c}^k + \mathbf{e}_j$$

and ϑ^k denotes the size of the largest layer in the k^{th} connector. The output is then sliced to the appropriate dimension for layer j. This process is repeated for all layers, and the resulting parameters are concatenated to form the complete connector parameter vector $\theta^k \in \mathbb{R}^{D_\theta^k}$. This modular, layer-wise parameterization makes the hypernetwork more tractable and memory-efficient.

3.3 Mini-batching model combinations for scalable hypernetwork training

Jointly training connectors for all $N \times M$ model combinations can become computationally prohibitive. To address this, we follow the strategy of model mini-batching (Knyazev et al., 2023), wherein each training step operates over a batch of B_m model combinations. The modified loss is:

$$\mathcal{L}_{\text{HYMA}} = \frac{1}{B_m} \sum_{k=1}^{B_m} \mathcal{L}_{\text{task}}(f_{H_\phi(\mathbf{c}^k)}). \tag{2}$$

Each training step proceeds as follows:

1. Sample a data batch of size B_d.
2. For each data sample, evaluate $\mathcal{L}_{\text{HYMA}}$ over each of the B_m model combinations.
3. Use the accumulated loss to update hypernetwork parameters ϕ.

This training strategy enables HYMA to scale efficiently without requiring all models or their combinations to be loaded simultaneously. We elaborate on the impact of this choice on our framework in Appendix F.

4 Experiments

4.1 Baselines

To ensure comprehensive evaluation of our proposed method, we compare against the following baselines:

- **Random**: A naive baseline that randomly selects and stitches uni-modal model pairs using the specified connector on the target multimodal dataset. Reported performance is the average over *five* independent trials.
- **UniModal Top-1 (UniT-1)**: Inspired by the observation in Fig. 1, this baseline stitches

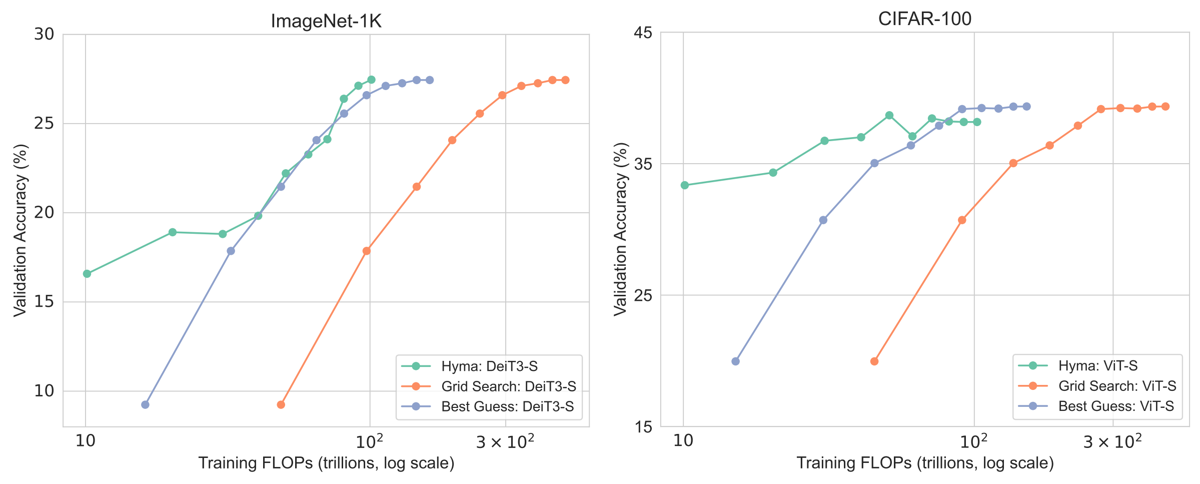


Figure 4: **MLP$_1$** | $N \times M = 3$: We show the trade-off between computational resources (measured in FLOPs) and performance of the best stitched model pairs across all comparative baselines. We find that HYMA is able to predict a highly performance pairing at a significantly reduced FLOP cost in comparison to training on the optimal model pair as well as search over all model pairs for $N \times M = 3$.

the top-performing individual uni-modal models—selected based on their uni-modal benchmark performance—via the target connector. For VLMs, image models are ranked by ImageNet Top-1 accuracy, and text models by their corresponding sentence embedding performance.

- **Ask-LLM**: Since uni-modal model properties such as parameter count and pretraining data can influence multi-modal performance, we define a baseline `Ask-LLM`. Here, a language model is prompted with metadata from the model zoo for both modalities and asked to select the most suitable pair for the target task. The chosen pair is stitched using a connector and evaluated in isolation.

- **AutoPair**: To enable a fair comparison with HYMA's efficiency-focused design, we implement an pairing baseline that iteratively searches a given set of pairs by training for a fixed number of epochs, and then prunes all pairs below the median performance. Specifically, AutoPair optimizes model pair selection and stitching within a FLOPs budget equal to that used by HYMA for the same model zoo. More details are provided in Section 5.3.

- **Oracle (Grid Search)**: This upper-bound baseline performs exhaustive grid search over all model pairs in the zoo, independently training and evaluating each stitched pair. While this provides optimal performance, it is computationally prohibitive.

- **Best Guess**: A hypothetical upper-bound baseline representing the training cost of the model combination that would yield the best multi-modal pair after stitching, assuming the optimal pair was known in advance.

4.2 Models

All model details are provided in Appendix B. To construct our Vision-Language Models (VLMs), we define a model zoo containing $N = 9$ image encoders: `ViT-S`, `DeiT-S`, `DeiT3-S`, `ViT-B`, `DeiT-B`, `DeiT3-B`, `ViT-L`, `DeiT3-L`, `Eva2-L` and $M = 3$ text encoders: `minilm-L`, `mpnet-B`, `roberta-L`. This results in a total of $N \times M = 27$ possible VLM configurations.

4.3 Connector variants

We test HYMA against the aforementioned baselines across three connector configurations:

1. **Linear**: As demonstrated in (Merullo et al., 2022), we construct the connector to be a linear layer parameterized via θ, mapping from the embedding space of the text encoder to the image encoder of a specific pair.

2. **MLP$_1$**: An MLP with one hidden layer of hidden dimension set 1024.

3. **MLP$_2$**: A MLP with two hidden layers, each of dimension 1024.

4.4 Datasets

We employ the LLaVA-CC558K dataset (Jia et al., 2024), which consists of 558,128 high-quality synthetic image-text pairs. Connectors between image and text encoders are trained using the contrastive InfoNCE loss (Oord et al., 2018) for 10 epochs, after which the best-performing checkpoint is selected. Hyperparameters are tuned for performance, stability, and GPU efficiency, detailed in Appendix.

4.5 Evaluation Tasks

Post-training, the resulting VLMs are evaluated on the following four downstream tasks:

- **Multi-modal Image Classification (MIC)**: We compute the zero-shot top-1 image classification accuracies of the VLMs on the ImageNet-1K (Deng et al., 2009) and the CIFAR-100 (Krizhevsky et al., 2009) datasets. The evaluation follows an image-text matching approach, where the text corresponding to each image input takes the form: "THIS IS A PHOTO OF A {CLASS}".

- **Image-Text Matching (ITM)**: Here, we compute the zero-shot recall @ 5 scores of the VLMs on the MSCOCO validation split (Lin et al., 2014) and the Flickr-8K (Hodosh et al., 2013) datasets.

- **Visual Question Answering (VQA)**: We use the validation splits of the OK-VQA (Marino et al., 2019) and the Text-VQA (Singh et al., 2019) datasets. Implementation details for VQA are given in Appendix G.

4.6 Varying multi-modal setting

We also explore our formulation on a different setting than contrastive VLMs, i.e., input-output stitching instead of output-output stitching. Fusing image encoder outputs to LLM inputs is a technique used to develop multi-modal language models (MLLMs) (Achiam et al., 2023; Touvron et al., 2023; Jia et al., 2024; Anthropic), another avenue of multi-modal models that we employ HYMA in. We find that using HYMA for MLLMs does not reflect the ranking observed via full grid search, however, HYMA predicts reliable connectors at a lower cost than grid search, with larger connectors exhibiting the best performance equal to the best setting found via grid search. We provide more details on this in Appendix A.

5 Empirical Results

5.1 MLP_1 | $\mathrm{N} \times \mathrm{M} = 3$

Initially, we stitch $N = 3$ image encoders (`ViT-S`, `DeiT-S`, `DeiT3-S`) with $M = 1$ text encoder (`MiniLM`) using an MLP connector of 1 hidden layer (MLP_1). This yields a total of $N \times M = 3$ possible VLMs, that we construct and evaluate on the image-classification task. For the best performing combination per evaluation benchmark, we show its accuracy in Figure 4 as well as the computational resources, measured in floating point operations (FLOPs) required to obtain the corresponding connectors.

On ImageNet-1K, `DeiT3-S` emerges as the best image encoder to be stitched with `minilm-L`. Further, HYMA and *Grid Search* (and *Best Guess*) exhibit the same final performance, i.e., 27.4 % top-1 accuracy. On the other hand, the most performative image encoder when stitched to `MiniLM` is `ViT-S`. In terms of performance, HYMA exhibits a top-1 accuracy of 38.4 %, nearly matching the performance of baselines that individually train connectors to find the optimal setting, i.e., 39.3 %. Also, HYMA is strongly cost-effective for VLMs, being 4.44× and 1.48× more compute-efficient than *Grid Search* and *Best Guess* respectively.

Dataset	Efficiency @10 ep (×)		Efficiency @ best (×)	
	BG	GS	BG	GS
IN-1K	1.48	4.44	1.48	4.44
CIFAR-100	1.48	4.44	2.96	8.89

Table 2: $N \times M = 3$, MLP_1: HYMA is significantly more compute-efficient than independently stitching model pairs, as shown w.r.t *Best Guess (BG)* and *Grid Search (GS)*.

5.2 Linear, MLP_1, MLP_2 | $\mathrm{N} \times \mathrm{M} = 27$

After demonstrating the efficacy of HYMA on a small search space of $N \times M = 3$ combinations and for MLP_1 scale up the number of combinations in comparison to $N \times M = 27$, and vary the capacity of the connectors in use (Linear, MLP_1, MLP_2). This yields 81 total VLMs. Table 3 shows the performance of HYMA in terms of a search, i.e., how well it matches the true ranking given by full grid search. Performance gain (Δ) is also reported across the *Random, UniT-1, Ask-LLM*, and *Oracle (GS)* baselines for each task and dataset employed.

Multi-modal Image classification: For multi-modal image classification on the ImageNet-1K,

Task	Dataset	Connector	NDCG @ k (↑)			ρ (↑)	$\Delta_{\texttt{Performance}}$ (↑)			
			$k=5$	$k=7$	$k=10$	$N \times M = 27$	Random	UniT-1	Ask-LLM	Oracle (GS)
MIC	IN-1K	Linear	1.0	1.0	0.98	0.97	+6.93	+13.51	+13.51	-4.14
		MLP_1	1.0	0.98	0.96	0.91	+4.78	+11.11	+11.11	-4.47
		MLP_2	0.96	0.93	0.92	0.89	+3.89	+10.34	+10.34	-5.91
	CIFAR-100	Linear	0.88	0.96	0.97	0.97	+6.91	+38.50	+38.50	-3.73
		MLP_1	0.83	0.96	0.97	0.86	+6.31	+35.21	+35.21	-1.85
		MLP_2	0.74	0.93	0.95	0.90	+5.01	+35.48	+35.48	-3.06
ITM	MSCOCO	Linear	0.96	0.95	0.99	0.99	+4.94	+31.62	+33.20	-2.0
		MLP_1	0.92	0.91	0.97	0.99	+3.72	+28.41	+28.41	-3.06
		MLP_2	0.96	0.91	0.97	0.98	+2.22	+27.30	+27.30	-4.03
	Flickr-8K	Linear	0.95	0.99	0.99	0.99	+5.18	+26.68	+7.83	-2.06
		MLP_1	1.0	1.0	0.99	0.99	+3.54	+23.32	+23.32	-2.26
		MLP_2	0.92	0.99	0.96	0.98	+1.92	+21.44	+21.44	-3.25
VQA	OK-VQA	Linear	0.95	0.95	0.98	0.99	+0.81	+7.86	+7.86	-0.43
		MLP_1	0.94	0.90	0.95	0.95	+0.49	+6.63	+6.63	-0.77
		MLP_2	0.99	0.93	0.93	0.97	+0.01	+6.81	+6.81	-1.44
	Text-VQA	Linear	0.94	0.97	0.99	0.97	+1.31	+3.64	+3.64	-0.06
		MLP_1	0.92	0.87	0.90	0.87	+0.72	+2.59	+2.59	-0.32
		MLP_2	0.85	0.87	0.86	0.87	+0.72	+2.28	+2.28	-0.59

Table 3: **HYMA VLM Results**: We report the ranking similarity between HYMA and the Oracle—Grid Search (GS)—using NDCG and Spearman's ρ. Across all three connector configurations, HYMA exhibits a strong correlation with GS rankings. Additionally, we show the performance gain (Δ) of the best connector obtained post stitching via HYMA, compared to four baselines: (a) *Random*: Random pairing and stitching (averaged over five runs), (b) *UniT-1*: Stitching the best unimodal models based on unimodal benchmarks, (c) *Ask-LLM*: Stitching based on model pairs selected via prompting Claude 4 Sonnet (detailed prompt in appendix), and (d) *Oracle*: Full grid search over all possible configurations on the complete model zoo ($N \times M = 27$).

we find that the ranking order of the stitching performed by HYMA reflects that found by full grid search to strong extent. This is indicated by the normalized discounted cumulative gain (NDCG @ k) computed for the top 5 and 7 ranks. Additionally, Spearman's ρ across all $N \times M = 27$ ranks further corroborates this. Notably, both NDCG @ k and Spearman's ρ for CIFAR-100 are lower in value w.r.t ImageNet-1K. In terms of performance gains, HYMA improves upon random selection of encoder pairs to stitch, as well as selecting encoders based on their uni-modal performance. Interestingly, we find that asking a massively pretrained LLM such as Claude 4 Sonnet yields a similar result to *UniT-1*. For Oracle (GS), find that the best stitchings generated by HYMA underperform average of 4.84 % and 2.88 for ImageNet-1K and CIFAR-100 across all connector types. However, this occurs at 10× fewer FLOPs spent.

Image-text matching: For image-text matching, we find higher values of Spearman's ρ, indicating that the stitches predicted by HYMA correlates strongly in performance with those obtained by full grid search on both MSCOCO and Flickr-8K. Similar to image-classification, we find that rank correlation metrics show more positive values for one dataset, Flickr-8K over the other, i.e., MSCOCO. In contrast, for image-text matching, we find that the performance gains (in recall@5) exhibited w.r.t *Ask-LLM* baseline do not match those of *UniT-1* in cases such as Linear connectors. In comparison to *Oracle (GS)*, average reduction in recall@5 is 3.03 for MSCOCO and 2.52 for Flickr-8K across all connectors.

Visual question answering: In visual question answering on both OK-VQA and Text-VQA, Linear connectors exhibit the highest values in terms of NDCG @ k, Spearman's ρ, as well as performance gain. In line with the preceded evaluation tasks, i.e., multi-modal image classification and image-text matching, we find that connectors predicted by HYMA outperform those found by the *Random, UniT-1* and *Ask-LLM* baselines. Most notably, VQA emerges as the task with the least performance gap between HYMA and *Oracle (GS)*, with 0.88 and 0.32 being the difference in the respective recall@5 values across both datasets.

5.3 HYMA vs AutoPair

We conduct a step-wise search-and-prune procedure over 6 image encoders (evenly split across embedding dimensions 768 and 1024) and 2 text encoders (also evenly split across embedding dimensions 768 and 1024). First we initialize a FLOPs

Connector	$\Delta_{\texttt{Performance}}$ (↑)					
	Multi-modal Image Classification		**Image-Text Matching**		**Visual Question Answering**	
	ImageNet-1K	CIFAR-100	MSCOCO	Flickr-8K	OK-VQA	Text-VQA
Linear	+11.28	+10.62	+11.04	+11.14	+2.12	+2.29
MLP_1	+4.50	+7.12	+2.08	+3.69	+0.24	+0.14
MLP_2	+3.25	+6.21	+3.62	+4.55	+0.75	+0.24

Table 4: **HYMA vs AutoPair Results ($N \times M = 12$):** We show the performance gain (Δ) of the best connector (for all connector configurations) obtained post stitching via HYMA, compared to that obtained via *AutoPair*.

budget equal to the total FLOP cost of searching over $N \times M = 12$ pairs with HYMA for 10 epochs. Next, our procedure trains connectors between all 12 pairs for 2 epochs each, after which we rank each connector by its performance on a given task and dataset. After the ranking, we prune all pairs that exhibit performance that is less than or equal to the median performance. This is repeated until we exhaust the budget. If we are left with only one model after iterative pruning, we train it until the budget is exhausted.

As shown in Table 4, stitches obtained by AutoPair exhibit significantly lower performance than those obtained via HYMA, as the budget finishes before the individually trained connectors can reach strong performance.

6 Related Work

Vision language models. CLIP, one of the most popular VLMs, is contrastively pretrained on approximately 400M image-text pairs. Beyond multimodal image classification and image-text retrieval, it has emerged to be applicable for tasks such as open-set attribute recognition (Chen et al., 2023) and object detection (Minderer et al.). Moreover, it inspires modifications to the default InfoNCE recipe, such as image captioning with contrastive pretraining, using sigmoid in place of softmax on the InfoNCE similarity matrix, etc. (Li et al., 2022; Alayrac et al., 2022; Singh et al., 2022; Zhai et al., 2023; Singh et al., 2024). Additionally, datasets oriented towards CLIP-like vision-language pretraining have been released in recent times, including (Schuhmann et al., 2021, 2022; Thomee et al., 2016; Changpinyo et al., 2021; Desai et al., 2021), often of the scale of millions of (image, caption) pairs. As a foundation model, CLIP has been applied in image synthesis (Rombach et al., 2022; Ramesh et al., 2022), and has been extended to modalities such as video (Chai et al., 2023; Wang et al., 2022) and audio (Guzhov et al., 2022). Our work investigates how to efficiently develop multiple CLIP-like models from pretrained uni-modal encoder states.

Hypernetworks in LLMs and multi-modal domains. Hypernetworks (Ha et al., 2016; Schmidhuber, 1992) have been shown useful in improving training efficiency and adaptability in many machine learning pipelines (Chauhan et al., 2024). Several works explored the advantages of hypernetworks for MLLMs and multi-modal models. Specifically, (Zhang et al., 2024) proposes HyperLLaVA that predicts project parameters for MLLMs given task input. Hypernetworks have also been used to predict the parameters of the adapters in parameter efficient fine-tuning of LLMs (Mahabadi et al., 2021; Phang et al., 2023) and VLMs (Zhang et al., 2022). HyperCLIP (Akinwande et al., 2024), trains a hypernetwork to predict the parameters of image encoder layers given the task. Overall, these models improve training efficiency and adaptability of a single combination on new tasks, but require grid search for more pairs. Our work addresses this limitation by training the joint hypernetwork for multiple encoders improving the efficiency and performance significantly.

7 Conclusion

We present a novel investigation of the usage of hypernetworks for the M-OPS problem. **HYMA** is able to subvert expensive grid search across all uni-model model combinations, by learning connector parameters jointly, producing strongly initialised connectors. We demonstrate that HYMA is an efficient solution to the **M-OPS** problem. Also, **HYMA**'s design affords stitching of modalities beyond only image-text: other avenues include, for instance, audio-text. We hope to inspire future work that utilizes hypernetworks for similar problems, where training several small neural networks can be expressed as a generative model that learns the parameters of the target network.

Limitations

Hypernetwork training can be less stable than training a standard connector (i.e., a single MLP). Training instabilities in hypernetworks have been previously studied (Ortiz et al., 2023; Chauhan et al., 2024), and are not unique to the specific design of our framework. However, since H_ϕ acts as a shared generating function across multiple connectors, the interaction of gradients from diverse model combinations—as well as their interplay with B_m—can still lead to instability during training. To stabilize training, we tune the β_2 parameter of the Adam optimizer in accordance with recommendations from the optimization literature (Cattaneo and Shigida, 2025). In practice, we observed that including certain models (for example: the MaxViT family (Tu et al., 2022)) in the $N \times M$ pool led to instability, and thus these models were excluded from our final zoo. This limitation points to the need for a deeper investigation into the training dynamics and architectural properties of similar systems, which could inform strategies to improve both stability and performance of the hypernetwork.

Acknowledgements

The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Diganta Misra. Jaisidh Singh is supported by the Konrad Zuse School of Excellence in Learning and Intelligent Systems (ELIZA) through the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research. This work was enabled by compute resources provided by Max Planck Institute for Intelligent Systems Tübingen & Amazon Science Hub.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Victor Akinwande, Mohammad Sadegh Norouzzadeh, Devin Willmott, Anna Bair, Madan Ravi Ganesh, and J Zico Kolter. 2024. Hyperclip: Adapting vision-language models with hypernetworks. *arXiv preprint arXiv:2412.16777*.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L Menick, Sebastian Borgeaud, and 8 others. 2022. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 23716–23736. Curran Associates, Inc.

Anthropic. Claude 4 sonnet. https://www.anthropic.com/claude.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Jinhe Bi, Yifan Wang, Danqi Yan, Xun Xiao, Artur Hecker, Volker Tresp, and Yunpu Ma. 2025. Prism: Self-pruning intrinsic selection method for training-free multimodal data selection. *Preprint*, arXiv:2502.12119.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Matias D. Cattaneo and Boris Shigida. 2025. Tuning adam(w): Default β2 may be too large. https://mdcattaneo.github.io/papers/Cattaneo-Shigida_2025_TuningAdam.pdf.

Wenhao Chai, Xun Guo, Gaoang Wang, and Yan Lu. 2023. Stablevideo: Text-driven consistency-aware diffusion video editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23040–23050.

Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. 2021. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3558–3568.

Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A Clifton. 2024. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, 57(9):250.

Keyan Chen, Xiaolong Jiang, Yao Hu, Xu Tang, Yan Gao, Jianqi Chen, and Weidi Xie. 2023. Ovarnet: Towards open-vocabulary object attribute recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23518–23527.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE.

Karan Desai, Gaurav Kaul, Zubin Aysola, and Justin Johnson. 2021. Redcaps: Web-curated image-text data created by the people, for the people. *arXiv preprint arXiv:2111.11431*.

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

Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel. 2022. Audioclip: Extending clip to image, text and audio. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 976–980. IEEE.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Junlong Jia, Ying Hu, Xi Weng, Yiming Shi, Miao Li, Xingjian Zhang, Baichuan Zhou, Ziyu Liu, Jie Luo, Lei Huang, and 1 others. 2024. Tinyllava factory: A modularized codebase for small-scale large multimodal models. *arXiv preprint arXiv:2405.11788*.

Boris Knyazev, Doha Hwang, and Simon Lacoste-Julien. 2023. Can we scale transformers to predict parameters of diverse imagenet models? In *International Conference on Machine Learning*, pages 17243–17259. PMLR.

Alex Krizhevsky, Geoffrey Hinton, and 1 others. 2009. Learning multiple layers of features from tiny images.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 36.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*.

Anas Mahmoud, Mostafa Elhoushi, Amro Abbas, Yu Yang, Newsha Ardalani, Hugh Leather, and Ari Morcos. 2024. Sieve: Multimodal dataset pruning using image captioning models. *Preprint*, arXiv:2310.02110.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204.

Jack Merullo, Louis Castricato, Carsten Eickhoff, and Ellie Pavlick. 2022. Linearly mapping from image to text space. *arXiv preprint arXiv:2209.15162*.

M Minderer, A Gritsenko, A Stone, M Neumann, D Weissenborn, A Dosovitskiy, A Mahendran, A Arnab, M Dehghani, Z Shen, and 1 others. Simple open-vocabulary object detection with vision transformers. arxiv 2022. *arXiv preprint arXiv:2205.06230*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Jose Javier Gonzalez Ortiz, John Guttag, and Adrian Dalca. 2023. Magnitude invariant parametrizations improve hypernetwork learning. *arXiv preprint arXiv:2304.07645*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Jason Phang, Yi Mao, Pengcheng He, and Weizhu Chen. 2023. Hypertuning: Toward adapting large language models without back-propagation. In *International Conference on Machine Learning*, pages 27854–27875. PMLR.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents, 2022. *arXiv preprint arXiv:2204.06125*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

Elan Rosenfeld, Preetum Nakkiran, Hadi Pouransari, Oncel Tuzel, and Fartash Faghri. 2022. Ape: Aligning pretrained encoders to quickly learn aligned multimodal representations. *arXiv preprint arXiv:2210.03927*.

Jürgen Schmidhuber. 1992. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, and 1 others. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294.

Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. 2021. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*.

Sheng Shen, Liunian Harold Li, Hao Tan, Mohit Bansal, Anna Rohrbach, Kai-Wei Chang, Zhewei Yao, and Kurt Keutzer. 2021. How much can clip benefit vision-and-language tasks? *arXiv preprint arXiv:2107.06383*.

Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. 2022. Flava: A foundational language and vision alignment model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15638–15650.

Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8317–8326.

Jaisidh Singh, Ishaan Shrivastava, Mayank Vatsa, Richa Singh, and Aparna Bharati. 2024. Learn" no" to say" yes" better: Improving vision-language models via negations. *arXiv preprint arXiv:2403.20312*.

Haoyu Song, Li Dong, Wei-Nan Zhang, Ting Liu, and Furu Wei. 2022. Clip models are few-shot learners: Empirical studies on vqa and visual entailment. *Preprint*, arXiv:2203.07190.

Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. 2016. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, (2):64–73.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. 2022. Maxvit: Multi-axis vision transformer. In *European conference on computer vision*, pages 459–479. Springer.

Junke Wang, Dongdong Chen, Zuxuan Wu, Chong Luo, Luowei Zhou, Yucheng Zhao, Yujia Xie, Ce Liu, Yu-Gang Jiang, and Lu Yuan. 2022. Omnivl: One foundation model for image-language and video-language tasks. In *Advances in Neural Information Processing Systems*, volume 35, pages 5696–5710. Curran Associates, Inc.

Ross Wightman. 2019. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986.

Wenqiao Zhang, Tianwei Lin, Jiang Liu, Fangxun Shu, Haoyuan Li, Lei Zhang, He Wanggui, Hao Zhou, Zheqi Lv, Hao Jiang, and 1 others. 2024. Hyperllava: Dynamic visual and language expert tuning for multimodal large language models. *arXiv preprint arXiv:2403.13447*.

Zhengkun Zhang, Wenya Guo, Xiaojun Meng, Yasheng Wang, Yadao Wang, Xin Jiang, Qun Liu, and Zhenglu Yang. 2022. Hyperpelt: Unified parameter-efficient language model tuning for both language and vision-and-language tasks. *arXiv preprint arXiv:2203.03878*.

Xun Zhu, Zheng Zhang, Xi Chen, Yiming Shi, Miao Li, and Ji Wu. 2025. Connector-s: A survey of connectors in multi-modal large language models. *arXiv preprint arXiv:2502.11453*.

Appendix

A HYMA for Multi-modal Large Language Models (MLLMs)

Another avenue for employing a predictive model for stitching can be MLLMs, which is significantly different from the VLMs case. Not only is the causal language modeling objective different from the contrastive scheme of VLMs, the connector stitches output image encoder representations to LLM input representations. In VLMs, the connector strictly stitches output representations, i.e., features produced by the text encoder are stitched to the space of image encoder features. We are interested in investigating how HYMA responds to this setting via the following experiments.

A.1 MLP_1 | $\text{N} \times \text{M} = 3$

We stitch $N = 1$ image encoder (`ViT-S`) with $M = 3$ LLMs (`GPT-2 (Radford et al., 2019), Pythia-160M (Biderman et al., 2023), Qwen-200M (Bai et al., 2023)`) using a 2-layer MLP (MLP_1) as the connector. Figure 5 shows the performance of HYMA in comparison to the *Grid Search* and *Best Guess* baselines respectively. We report the performance of the best connectors identified by each search method, with the FLOPs incurred via training. We find that HYMA reduces the cost of searching of all combinations, bringing it lower than training only one connector for the $N \times M = 3$ case. The efficiency of HYMA over the two comparative baselines at the final state (rightmost point in each plot) is $3\times$ w.r.t. *Grid Search* and $1.3\times$ w.r.t. *Best Guess*. Further, all search methods yield comparable optimal perplexities, 51.1 for HYMA and 51.0 for *Grid Search* (or *Best Guess*) on MSCOCO. On Flickr-8K, the perplexities are found to be 72.4 and 70.4 for HYMA and *Grid Search* (or *Best Guess*) respectively.

A.2 Linear, MLP_1, MLP_2 | $\text{N} \times \text{M} = 9$

We scale up our experimental setting to now use $N = 3$ image encoders (`Clip-ViT-B`, `DeiT3-B`, `ViT-S`) and $M = 3$ LLMs (`GPT-2`, `Pythia-160M`, `Qwen-200M`). Similar to the case for VLMs, we vary the complexity of the connector from a linear layer, to an MLP with 2 hidden layers. Evaluation is done similarly to the case with $N \times M = 3$ MLLM combinations, i.e., via image captioning on MSCOCO and Flickr-8K. As shown in Table 5. HYMA struggles to match true ranking of model pairs for the MLLM case. Specifically, it performs worse on connectors of lower complexity, and consistently under-performs in terms of validation perplexity. Careful observation shows that for $N \times M = 9$ MLLMs, the ranking of connectors predicted by HYMA follows a trend of uni-modal model performance (the best image encoder (`Clip-ViT-B`) and LLM (`Qwen-200M`) show the best performance. However, independent stitching does not show such behavior. Overall, connectors obtained via independent stitching outperform those obtained from HYMA by a significant margin, and the true ranking diverges notably from that predicted by HYMA. Investigations on disentangling the effects of the causal modeling loss and the change in stitched representation spaces is left as future work.

B Pretrained models

B.1 Image encoders (source: `timm` (Wightman, 2019))

Feature Dim.	Model	Shorthand	Param. count (M)	`timm` specifier
384	`ViT-S`	`VS`	22.05	`vit_small_patch16_224.augreg_in21k_ft_in1k`
	`DeiT-S`	`DS`	22.05	`deit_small_patch16_224.fb_in1k`
	`DeiT-3S`	`D3S`	22.06	`deit3_small_patch16_224.fb_in1k`
768	`ViT-B`	`VB`	86.57	`vit_base_patch16_224.augreg_in21k_ft_in1k`
	`DeiT-B`	`DB`	86.57	`deit_base_patch16_224.fb_in1k`
	`DeiT3-B`	`D3B`	86.88	`deit3_base_patch16_224.fb_in22k_ft_in1k`
	`Clip-ViT-B`	`CVB`	86.86	`vit_base_patch1_clip_224.laion2b_ft_in12k_in1k`
1024	`ViT-L`	`VL`	304.33	`vit_large_patch16_224.augreg_in21k_ft_in1k`
	`Eva2-L`	`E2L`	305.08	`eva02_large_patch14_448.mim_m38m_ft_in22k_in1k`
	`DeiT3-L`	`D3L`	304.37	`deit3_large_patch16_224.fb_in22k_ft_in1k`

Table 6: All pretrained image encoders used in our work are given above, along with their shorthand IDs that may be referred to in the main manuscript.

B.2 Text encoders & LLMs (source: huggingface (Wolf et al., 2020))

Feature Dim.	Model	Shorthand	Param. count(M)	`huggingface` specifier
384	`minilm-L`	`mlL`	33.4	`sentence-transformers/all-MiniLM-L12-v2`
768	`mpnet-B`	`mpB`	109	`sentence-transformer/all-mpnet-base-v2`
1024	`roberta-L`	`roL`	355M	`sentence-transformer/all-roberta-large-v1`
768	`GPT-2`	`g2`	137	`openai-community/gpt2`
	`Pythia-160M`	`py`	213	`EleutherAI/pythia-160m`
	`Qwen-200M`	`qw`	203	`MiniLLM/MiniPLM-Qwen-200M`

Table 7: All pretrained text encoders and LLMs used in our work are given above, along with their shorthand IDs that may be referred to in the main manuscript.

C Designing the Model Zoo

While our empirical analysis suggests that models with larger parametric capacity or higher embedding dimensionality generally perform better after stitching, a natural question arises: why include smaller models in the model zoo at all? We justify their inclusion based on the following:

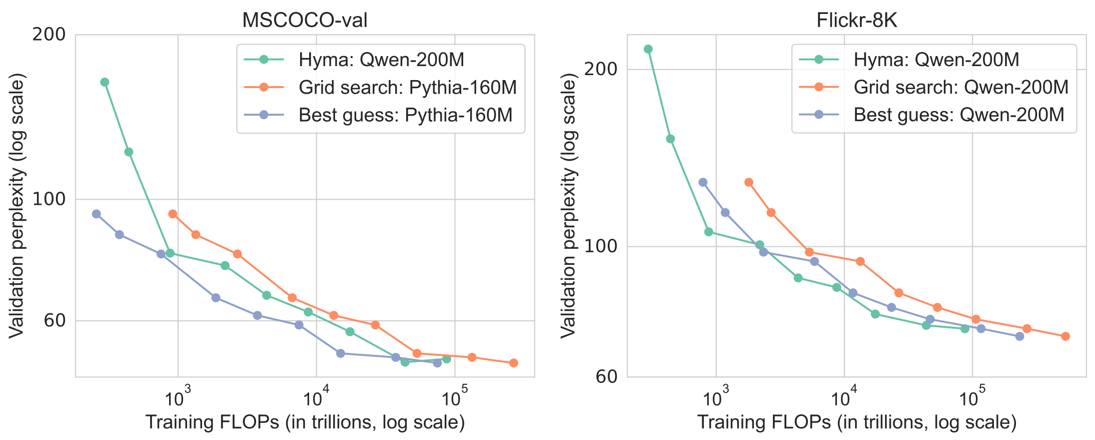


Figure 5: Evaluation of HYMA for MLLMs, on MSCOCO and Flickr-8K ($N = 1, M = 3, B_m = 1$). We report the model combination exhibiting the best final performance for each evaluation benchmark and search method.

Dataset	**Connector**	**NDCG @ k (↑)**			**ρ (↑)**	**$\Delta_{\texttt{Perplexity}}$ (↓)**			
		k=5	k=7	k=9	$N\times M$=9	Rand. (n=5)	UniT-1	Ask-LLM	Oracle (GS)
MSCOCO	Linear	0.16	0.42	0.74	-0.6	+3.68	+6.85	+3.20	+6.85
	MLP$_1$	0.65	0.79	0.89	0.35	+0.65	+2.2	+2.20	+3.5
	MLP$_2$	0.61	0.74	0.85	0.39	+1.13	+3.58	+3.58	+4.01
Flickr-8K	Linear	0.56	0.73	0.85	0.12	+1.65	+5.54	-1.46	+5.54
	MLP$_1$	0.77	0.82	0.90	0.45	-0.1	+1.30	-1.30	+4.5
	MLP$_2$	0.58	0.72	0.83	0.23	-3.57	-0.00	-0.00	-0.00

Table 5: **HYMA MLLM Results**: We report the ranking similarity between HYMA and the Oracle—Grid Search (GS)—using NDCG and Spearman's ρ. Across all three connector configurations, HYMA exhibits strong correlation with GS rankings. Additionally, we show the perplexity difference (Δ) of the best connector obtained post stitching via HYMA, compared to four baselines: (a) *Random*: random pairing and stitching (avg. over 5 runs); (b) *UniT-1*: stitching the best unimodal models; (c) *Ask-LLM*: model pairs picked by Claude 4 Sonnet; and (d) *Oracle*: Full grid search over all $N\times M$=9 configurations.

1. First, including smaller models enables the construction of multi-modal models across a range of parametric capacities, which is crucial for deployment under varying computational or resource constraints. For example, an organization aiming to deploy multi-modal models at multiple scales would incur significantly higher training costs if relying on independent training for each configuration. In contrast, HYMA offers a substantially more cost-effective alternative.

2. Second, our empirical observations indicate that larger models are not always the best-performing choice when stitched into multi-modal pairs. This motivates the inclusion of a diverse set of model configurations in our zoo to better explore the multi-modal design space. By covering a broader range of capacity combinations, HYMA facilitates a more comprehensive and efficient search, supported by observations from Figure 1 and Table 1.

D Training and hyper-parameter details

We tune hyperparameters for each trained model to maximize (i) validation performance, (ii) GPU utilization, and (iii) training stability. Our goal is to demonstrate that hypernetworks can efficiently approach the M-OPS problem that often requires a large amount of computational resources. Hence, we emphasize on the need to have maximum GPU utilization in order to present an efficiency-oriented method. We report the hyperparameters used for training connectors for VLMs along with the configuration for **HYMA**. We use 3 random seeds and report average performance in each experiment.

VLMs. Training individual connectors between VLMs uses hyperparameters that provides the best performance after 10 epochs of training. Our hyperparameter choice is similar to that of (Rosenfeld et al., 2022). Specifically, we use a batch size of 2^{14}, the Adam optimizer, and a learning rate of $1e-2$ subject to a schedule that linearly warms

```
def train_hypernet(hypernet, data_iter, models_iter, optimizer, num_steps):
    hypernet.train()
    for step in range(num_steps):
        # first sample (image, caption) data with batch size B_d
        data_batch = next(data_iter)

        # then subsample the full NxM space of models with batch size B_m
        model_batch = next(models_iter)

        optimizer.zero_grad()

        # input to the hypernetwork are indices or ids of the respective pairs
        vlm_ids_in_full_zoo = get_ids_wrt_full_zoo(model_batch)

        # hypernet outputs parameters of the stitches between the pairs
        generated_params = hypernet(vlm_pair_ids)

        # mapped the data through the stitched model pairs
        # and compute multi-pair multi-modal loss
        loss = hypernet.forward_data_through(
            data_batch,
            generated_params,
            model_batch
        )

        # back-propagate
        loss.backward()
        optimizer.step()
```

Figure 6: PyTorch (Paszke et al., 2019) pseudocode for HYMA training procedure on $N \times M$ models.

up the learning rate from 0 for 50 steps. After that, the learning rate is decayed to 0 following a cosine curve. Training HYMA for VLMs is quite sensitive to hyperparameters, as is to be expected from a complex network that outputs large spaces especially considering how it does so using indirectly (using layer-specific embeddings). The optimal batch size, i.e., that ensures the most stable training is 2^9, and the learning rate is set to $1e-2$ for the Adam optimizer. As mentioned in the main manuscript, the value of the model batch size B_m affect the training strongly, hence we set it to 1 when $N \times M = 3$ and 9 when $N \times M = 27$. For AutoPair, $N \times = 12$ and $B_m = 4$.

MLLMs. For MLLMs, we follow recipes given in (Jia et al., 2024) for training only the connector (referred to as the feature alignment phase of pretraining). Particularly, we use Adam with batch size of 64 for training individual connectors and learning rate $1e-3$. This is subject to a schedule of warmup ratio $3e-2$ following a cosine decay to 0. The batch size training HYMA for MLLMs is 32 and the learning rate is $1e-3$.

Architectural experiments. For VLMs, we tried using a compression of the image encoder features as the conditional input to the hypernetwork, while keeping all other components the same. Only the learnable code-book is replaced by a learnt compression of batch-averaged image encoder features. This configuration, denoted as HYMA$_{EC}$ yielded lower performance than our default methodology HYMA. Specifically for the $N \times M = 3$ case, for multi-modal image classification on ImageNet-1K, we find that the top-1 accuracy of the best model pair given by HYMA is superior to that given by HYMA$_{EC}$ shown in Table 8. Figure 6 provides an example pseudo-code depicting our training setup.

Architecture	IN-1K Top-1 accuracy
HYMA	**27.46**
HYMA$_{EC}$	12.11

Table 8: HYMA performs significantly better downstream in comparison to HYMA$_{EC}$.

E Factors impacting FLOPs

While the numbers of parameters in the model being trained is no doubt a factor that is linearly proportional to the total FLOPs incurred, we note that there are other factors like hyperparameters as well. For loss functions that relate linearly with the batch size, batch size has no effect on the total number of FLOPs incurred after the entire training run, as the model takes fewer update steps on a bigger batch size, but proportionately more on a smaller one. However, for loss functions that scale quadratically with the number of data samples observed, such as the InfoNCE loss (Oord et al., 2018), the value of batch size can significantly affect the FLOP count. This, after the primary design choice of iteratively loading models, which decreases the number of samples shown to a model by $N \times M/B_m$, accounts for why HYMA, that training a large hypernetwork (of an average of $500\times$ more parameters than the connector) is efficient, particularly for VLMs. For the case of MLLMs, the reasons become our design choice of iterative model batches, as well as the fact that certain LLMs are of a larger parametric capacity than others. Hence backpropagating the gradient through them into the connector for a total of $\mathcal{T}$ steps is more expensive than doing so for $\mathcal{T}/(N \times M/B_m)$ steps via HYMA.

F Connection to Data Pruning

Method	Best Model configuration	Perf.
C-GS	DeiT-3S + miniLM-L	24.07
HYMA	DeiT-3S + miniLM-L	**27.46**

Table 9: **HYMA vs. Constrained Grid Search (C-GS).** For the setting $N \times M = 3$, $B_m = 1$, we constrain the total data available to Grid Search to one-third, aligning it with HYMA's data budget. While this constraint results in a comparable reduction in FLOPs relative to full Grid Search, it leads to a notable drop in performance. **Perf.** denotes MIC top-1 accuracy on ImageNet-1K.

While HYMA provides a unified and compute-efficient framework for addressing the M-OPS problem, the primary reduction in FLOPs arises from the dual mini-batching strategy employed during training. This dual mini-batching mechanism results in each model pair configuration being exposed to a smaller subset of data compared to independent stitching, effectively mimicking randomized data pruning in the process of constructing multimodal models from unimodal pairs.

Data pruning and filtering strategies for multimodal training have been extensively explored in prior work (Fang et al., 2023; Bi et al., 2025; Mahmoud et al., 2024), typically focusing on restricting the training data via heuristic-based selection. In contrast, HYMA adopts a randomized approach: the mini-batching process dynamically selects data for each model configuration, and across multiple training steps, both the data and batch assignments are shuffled. This results in a more uniform and implicit allocation of the dataset across the space of possible model configurations, while still maintaining computational efficiency. It is important to note, however, that this data reduction applies only to each model configuration independently; the hypernetwork H_ϕ, which generates the connector weights, is still trained over the entire dataset.

This effect is further evident when comparing HYMA to a constrained version of Oracle (Grid Search) (C-GS). As shown in Table 9, when the total data available to Grid Search is limited to one-third—matching HYMA's data budget—the best-performing model identified by C-GS performs significantly worse than HYMA.

G VQA implemention

We follow a methodology similar to the method Question Irrelevant Prompt (QIP) (Shen et al., 2021; Song et al., 2022) that creates a prompt of "QUESTION: `{question}` ANSWER: `{answer}`" for a given image. This prompt is embedded via the text encoder and the task is to match the image to the correct prompt, as an image-text matching objective.

H Baselines: Ask-LLM (for VLMs)

We prompt Claude 4 Sonnet (Anthropic) to identify the best model pair where we specify the image encoder metadata from `timm` (details of the image encoder from the ImageNet-1K results database such as accuracy, parameters, image size for pretraining). The metadata of the text encoder is obtained via `huggingface` (details of the pretrained text encoder like embedding dimension, parameters). The "`task`" is one among multi-modal image classification, image-text matching, and visual question answering, whereas "`dataset`" is simply the name of the dataset, and "`dataset_metadata`" contains the number of samples, classes, questions, and answers, as needed for the dataset. We specify the type of connector (Linear, MLP_1, MLP_2) via "`depth`".

"You are an oracle which will predict which combination of image and text encoders will perform best on a given task. The task is to predict which (image encoder, text encoder) pair will yield the best CLIP-like VLM from a list of image encoders and text encoders. More details about this: each pair of encoders will be connected via an MLP of number of hidden layers `{depth}` (0 means a linear layer), which will be trained to map text embeddings to the image embedding space such that the InfoNCE loss is minimized.

Your job is NOT TO provide any code or run the experiment. JUST TO PREDICT WHICH PAIR WILL YIELD THE BEST `{task}` `{task_metric}` on `{dataset}` (`{dataset_metadata}`).

Here are the image encoders, along with their metadata: `{image_encoders_with_metadata}`

Here are the text encoders, along with their metadata: `{text_encoders_with_metadata}`

Please provide your answer in (image_encoder, text_encoder) format ONLY. NO OTHER TEXT SHOULD BE PRODUCED BY YOU EXCEPT THE ANSWER IN THE REQUIRED FORMAT."

I Baselines: Ask-LLM (for MLLMs)

"You are an oracle which will predict which combination of image encoder and LLM will perform best on image captioning task. The task is to predict which (image encoder, LLM) pair will yield the best GPT4-like MLLM from a list of image encoders and LLMs. More details about this: each pair will be connected via an MLP of number of hidden layers `{depth}` (0 means a linear layer), which will be trained to map patch-wise image encoder outputs to the input embedding space of LLM such that the causal language modeling loss is minimized.

Your job is NOT TO provide any code or run the experiment. JUST TO PREDICT WHICH PAIR WILL YIELD THE BEST `{task}` `{task_metric}` on `{dataset}` (`{dataset_metadata}`).

Here are the image encoders, along with their metadata: `{image_encoders_with_metadata}`

Here are the LLMs, along with their metadata: `{llms_with_metadata}`

Please provide your answer in (image_encoder, llm) format ONLY. NO OTHER TEXT SHOULD BE PRODUCED BY YOU EXCEPT THE ANSWER IN THE REQUIRED FORMAT."

We specify image encoder details as done for VLMs, but LLMs details are obtained from `huggingface` (parameters, embedding dimension, context length).