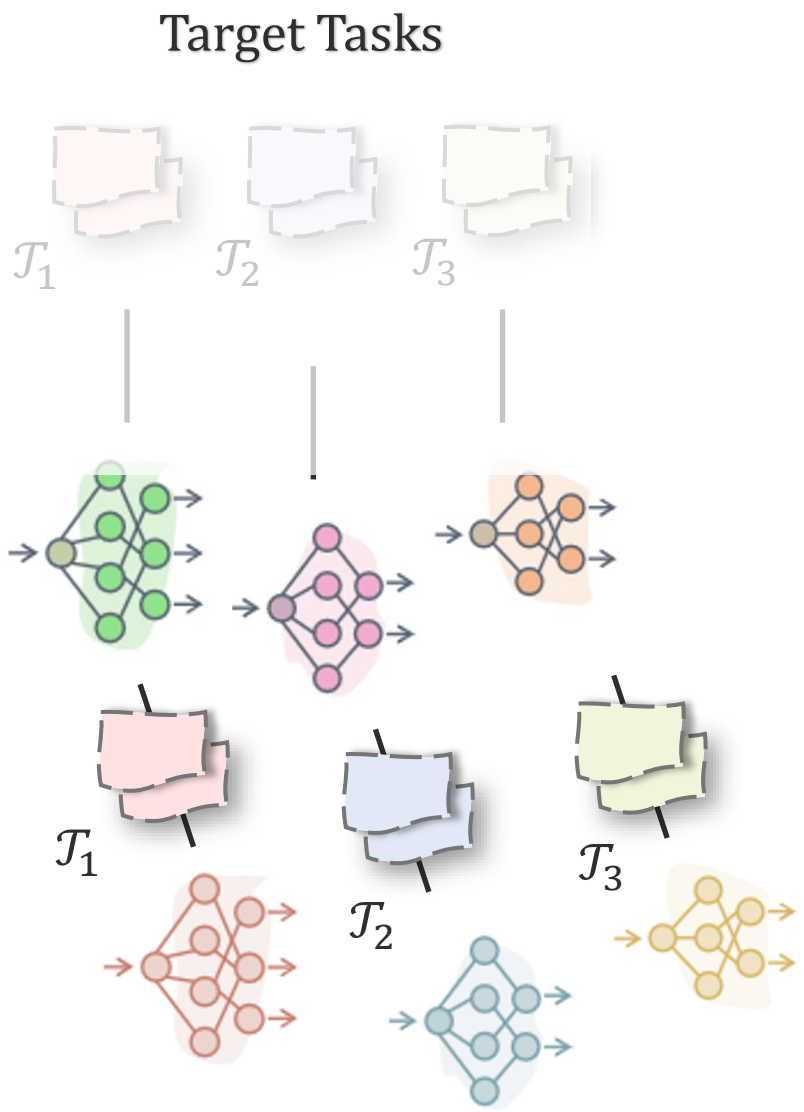
Model Reuse in the LLM Era:

Leveraging Pre-Trained Resources with Classical and Modern Approaches

Da-Wei Zhou 周大蔚
Nanjing University

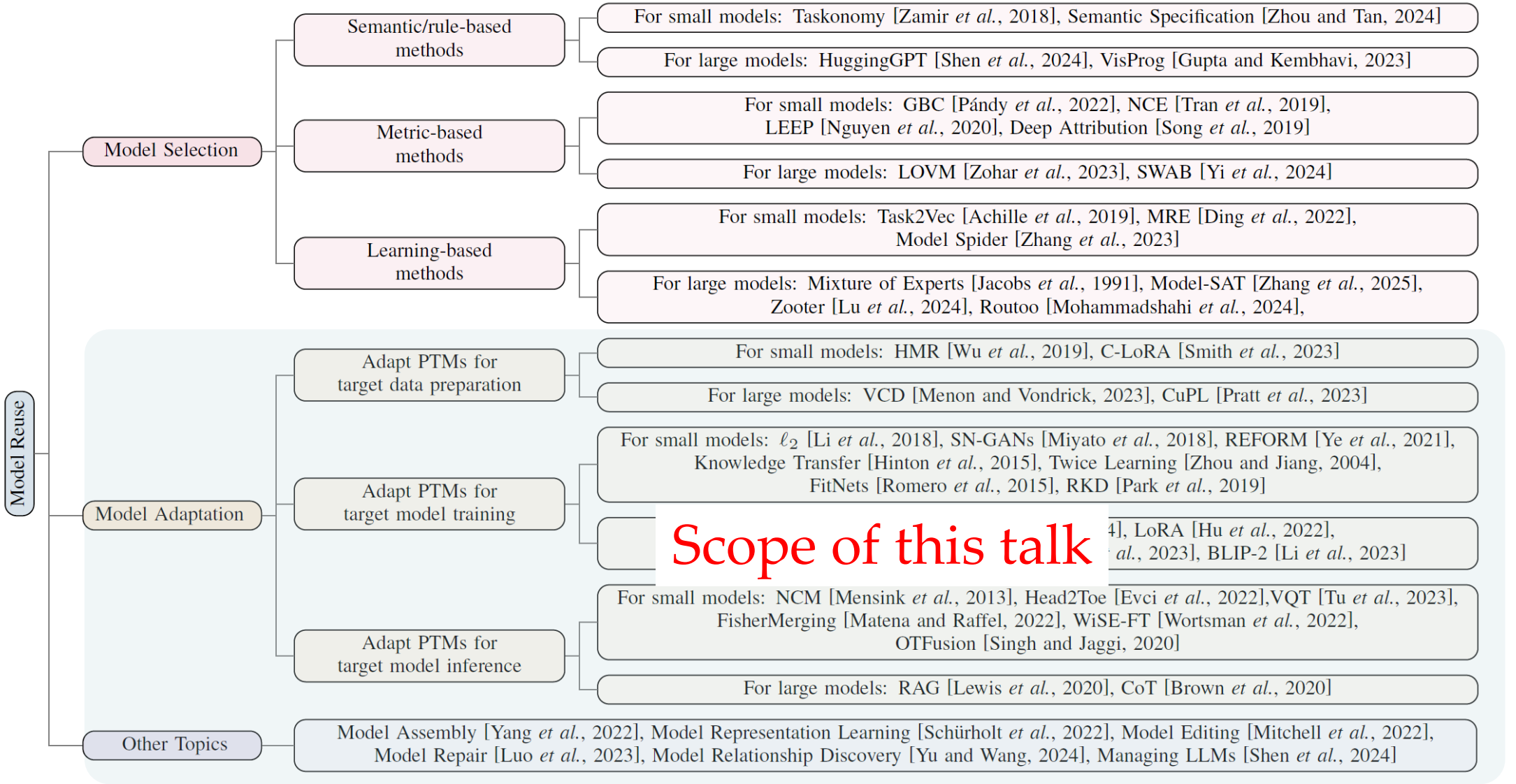
PTM Adaptation



Pre-trained Model
Reuse/Adaptation

Reuse the selected pre-trained models for target tasks

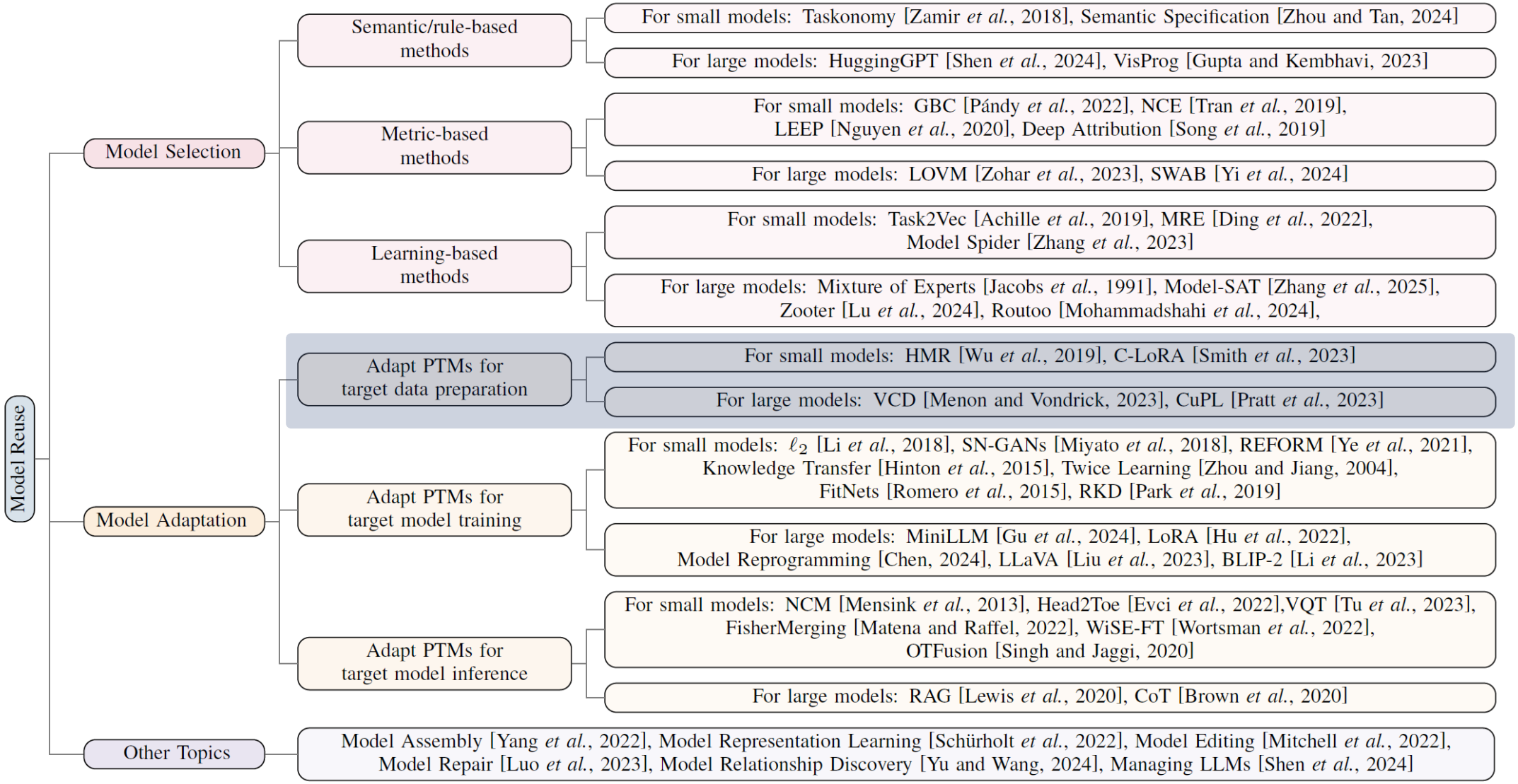
Taxonomy of Model Reuse



Outline

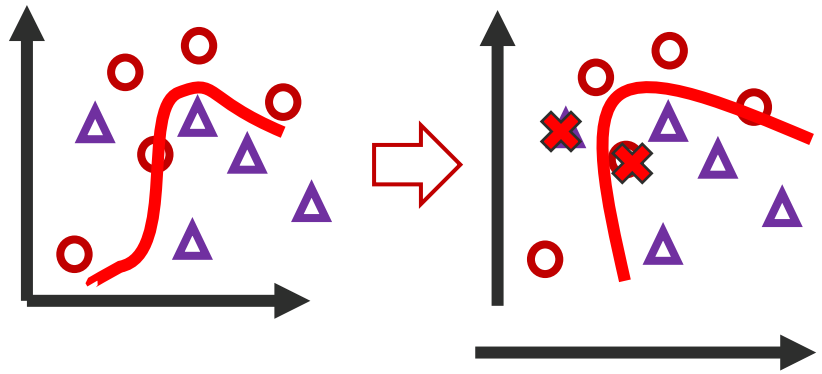
- *Taxonomy of Model Adaptation*
  - *Adapt PTMs for target data preparation*
  - *Adapt PTMs for target model training*
  - *Adapt PTMs for target model inference*
- *Other topics in model reuse*
- *Conclusion*

Taxonomy of Model Reuse

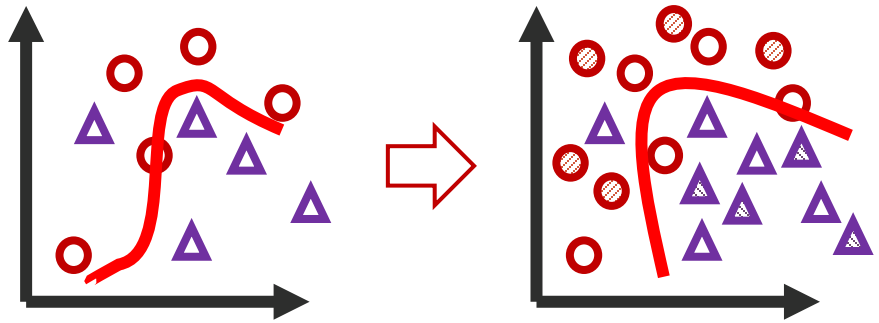


Adapt PTMs for target data preparation

- PTMs serve as experts capable of generating enriched data for the target task, or filtering noisy data for the target task, enabling the target model to achieve better generalization ability.



Filter the existing training set.

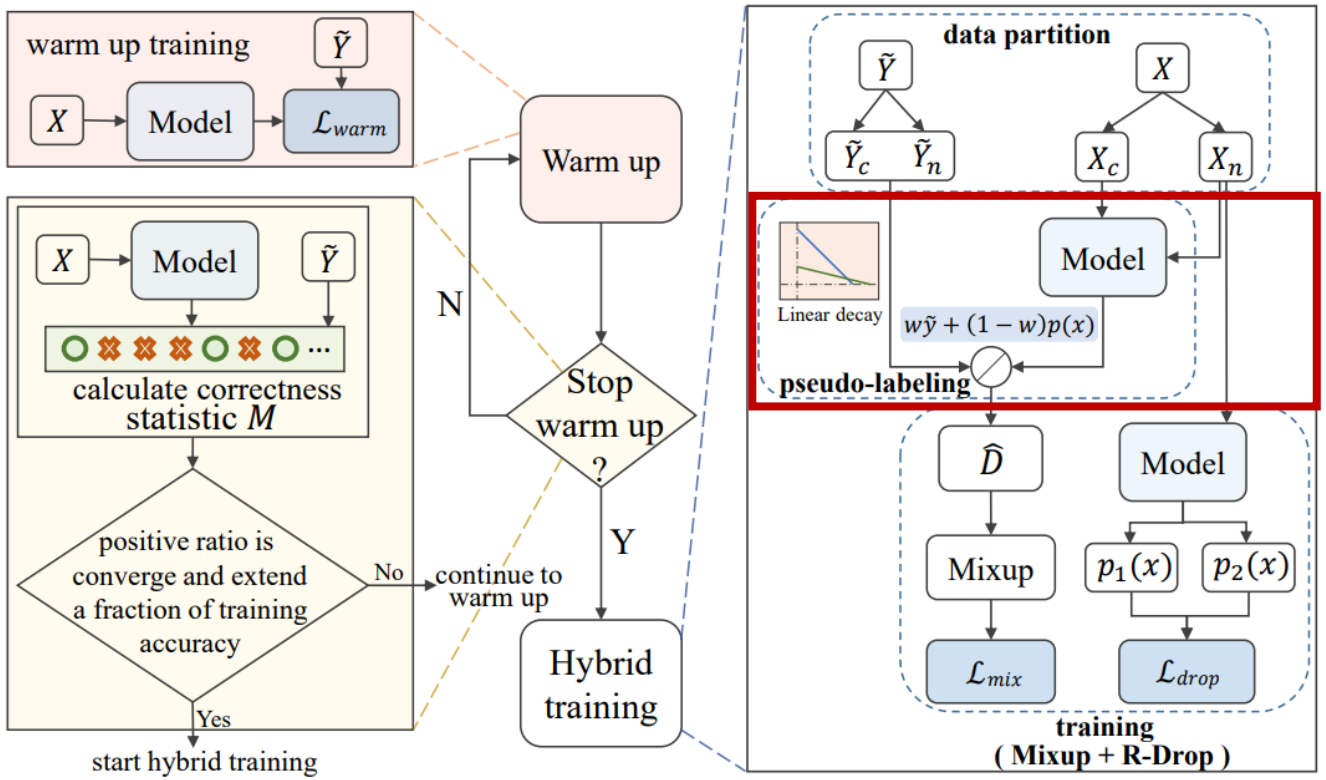


Generate related data.

PTM for Data Denoise

- PTM can rectify the noisy instances x_i or labels y_i in $\mathcal{T}$, based on which we make the target model be trained on a denoised dataset.

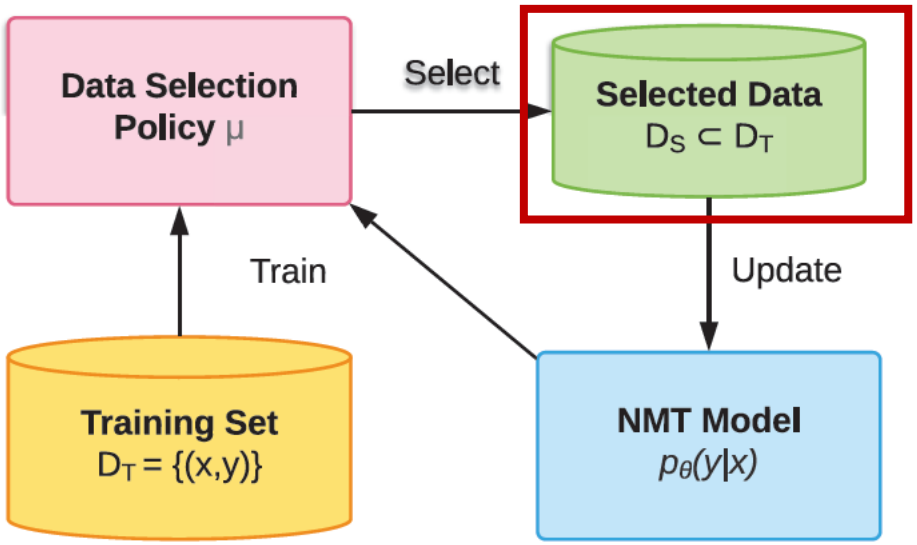
  Mix pseudo label from PTM and the true label [Cheng et al., EMNLP'23].



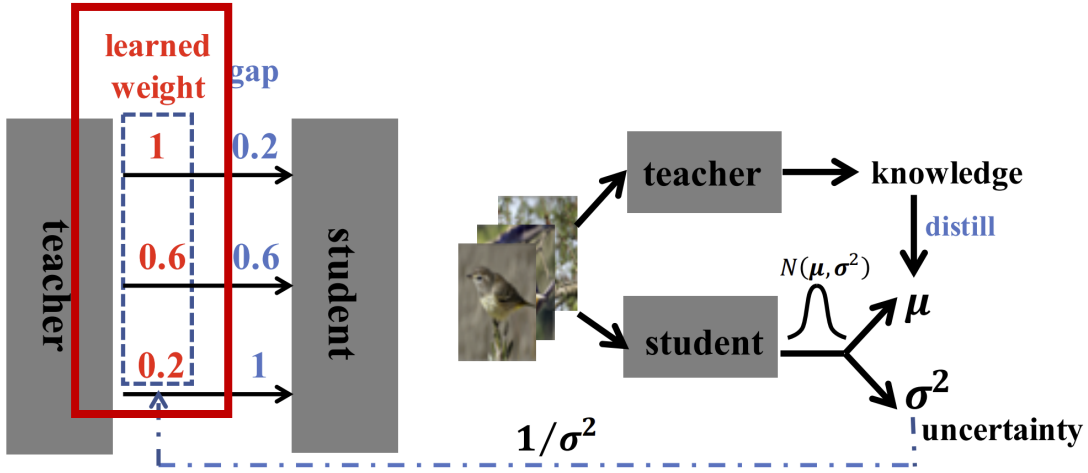
PTM for Data Selection

- PTM can selects the important examples $(\boldsymbol{x}_i, y_i)$, by prioritize the learning on them, the negative effect of noisy data is mitigated.

Utilize pre-trained model to *weigh* different instances in curriculum learning for a faster convergence rate [Zhao et al., AAAI'20].



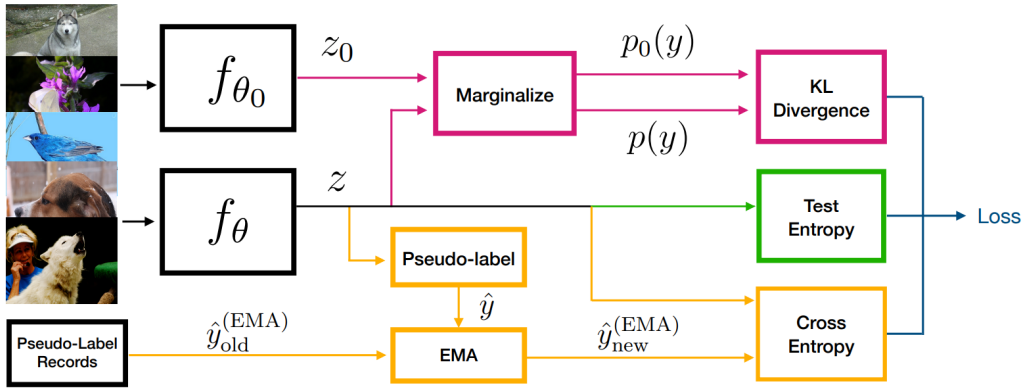
PTM utilizes various criteria (e.g., the prediction uncertainty of the PTM) to set the weight of the target examples [Zhang et al., ECCV'20][Liu et al., AAAI'23].



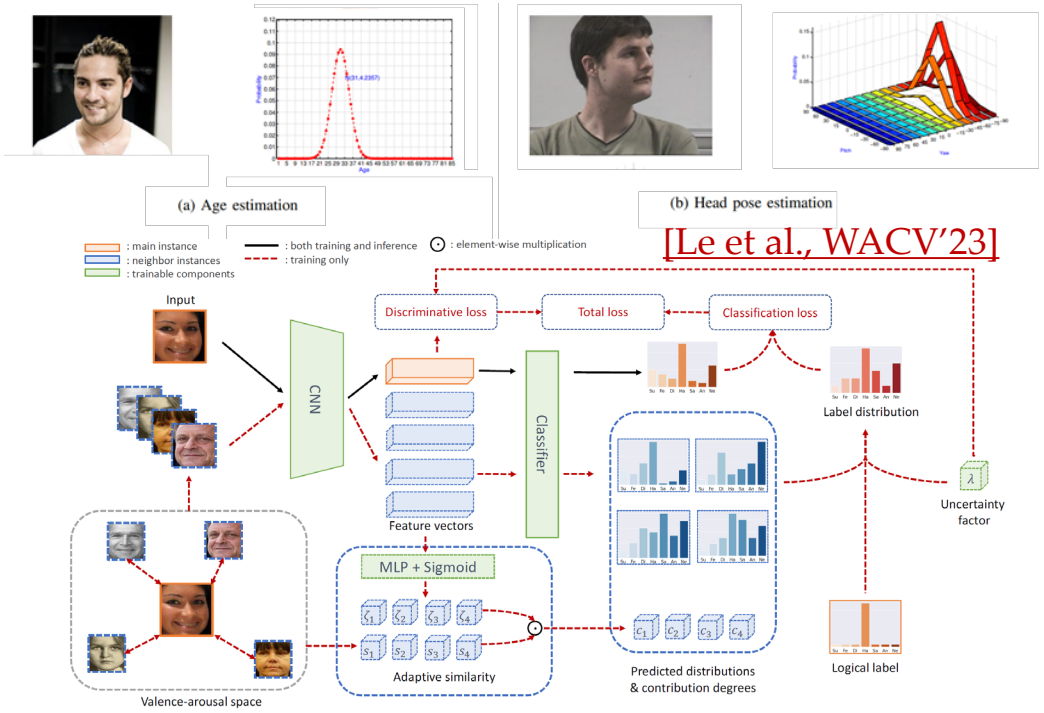
PTM for Richer Labels

- PTM provides better supervision via modified labels. By learning on the enricher supervision, the target model generalizes better.

PTM can provide supervision (pseudo labels) for unlabeled data [Yen et al., CoRR].



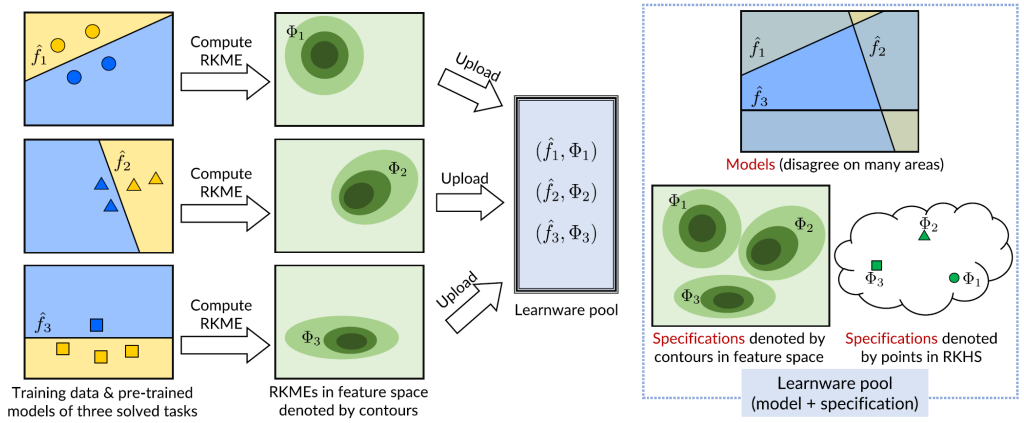
PTM can help obtain pseudo labels, or even the distribution of labels [Geng, TKDE'16][Gao et al., TIP'17].



PTM for Data Augmentation

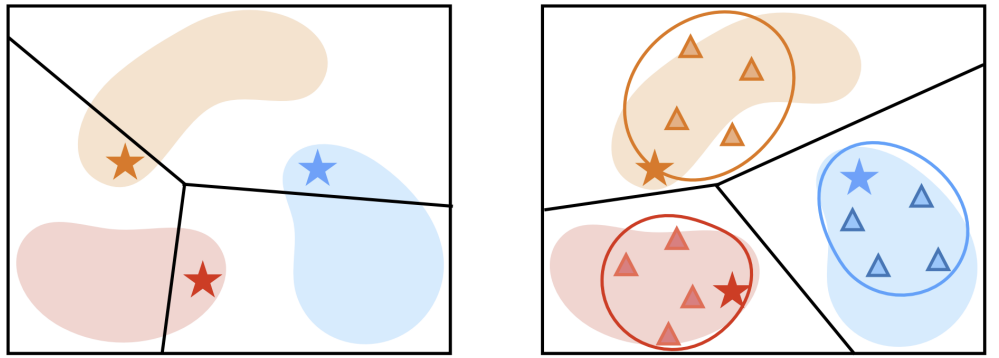
- PTM saves the (sufficient) statistics of the pre-trained data, which is able to generate synthetic data in the reuse phase.

Set PTM as the Reduced Kernel Mean Embedding (RKME) [Wu et al., TKDE'21].



Generate synthetic data with RMKE.

Utilize PTM to synthesize examples with Gaussian augmentation [Yang, Liu, Xu, ICLR'21].



Classifier trained with few-shot features

Classifier trained with features sampled from calibrated distribution

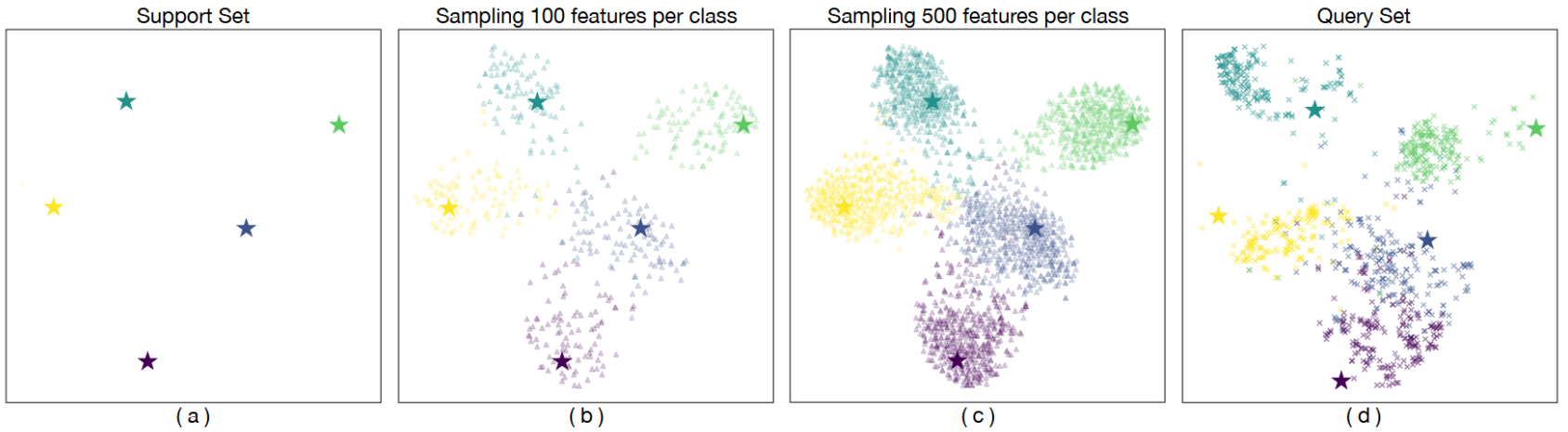
Generate synthetic data given gaussian sufficient statistics.

PTM for Data Augmentation

- PTM saves the (sufficient) statistics of the pre-trained data, which is able to generate synthetic data in the reuse phase.

	Arctic fox	
	mean sim	var sim
white wolf	97%	97%
malamute	85%	78%
lion	81%	70%
meerkat	78%	70%
jellyfish	46%	26%
orange	40%	19%
beer bottle	34%	11%

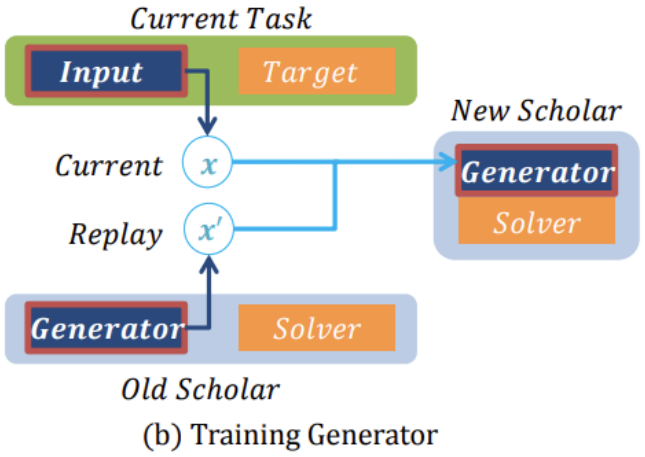
Similar classes share similar statistical information



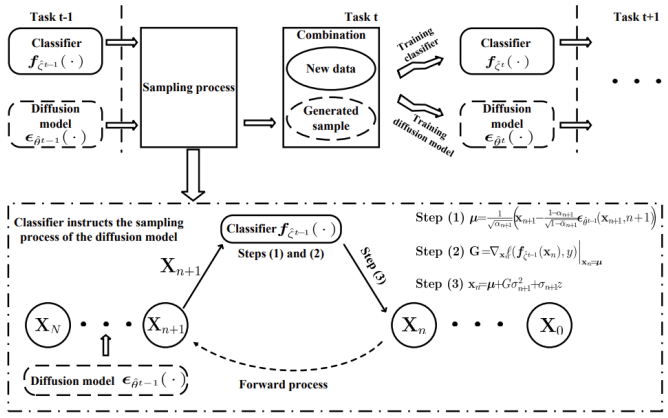
Calibrate and generate samples for few-shot classes

PTM for Data Augmentation

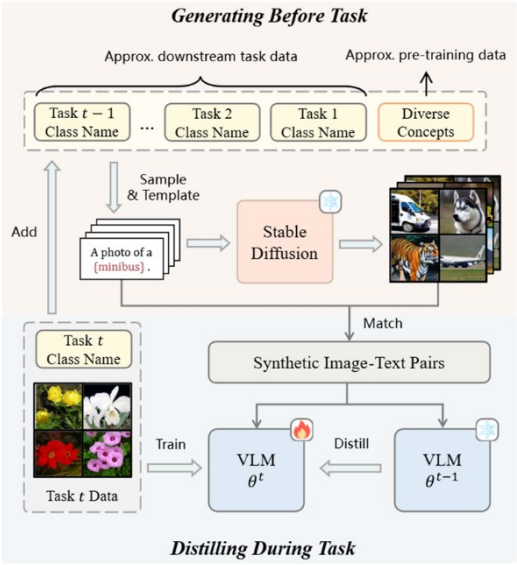
- PTM saves the (sufficient) statistics of the pre-trained data, which is able to generate synthetic data in the reuse phase.
- Utilizing GAN/Stable Diffusion to serve as data generator to generate source instances.



[Shin et al., NIPS'17]



[Gao et al., ICML'23]



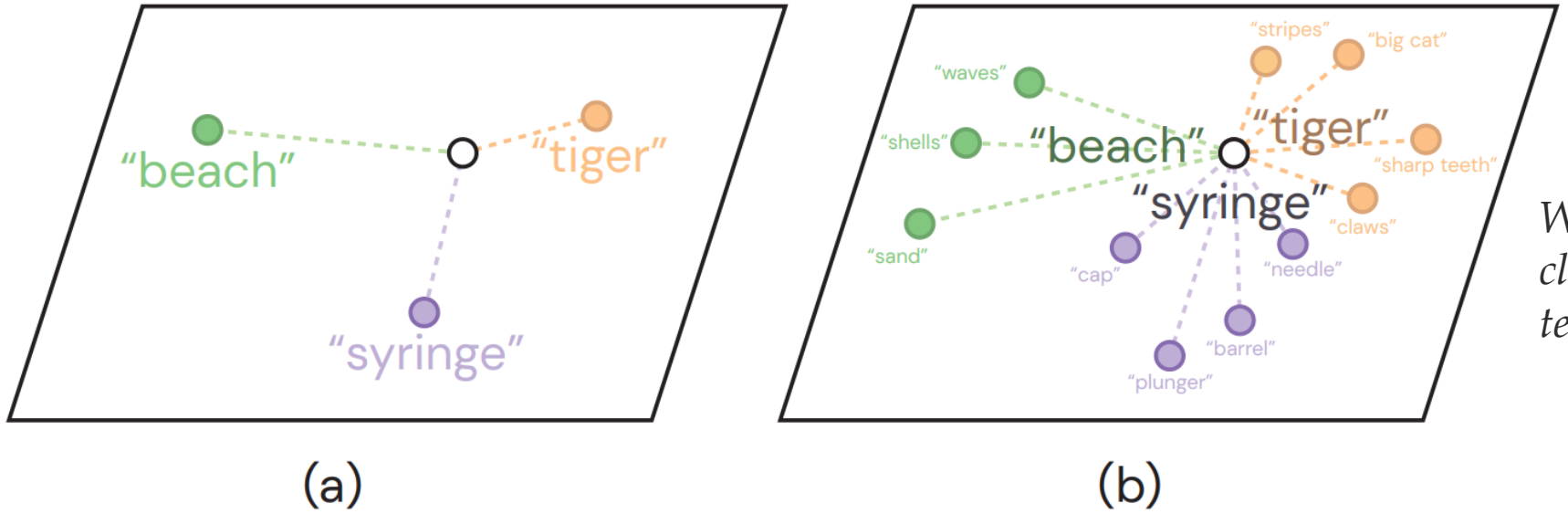
[Wu et al., CVPR'25]

*There exists a **gap** between the target data and generated data, which should be carefully considered in model reuse.*

PTM for Expert Knowledge

- Utilizing PTM to obtain auxiliary meta-data (a.k.a. *privileged information*) [Vapnik, Vashist, NN'09] [Vapnik, Izmailov, JMLR'15].

- Take advantage of ChatGPT to provides expert knowledge, e.g. utilizing ChatGPT to automatically generate descriptors given class names. Then perform image recognition by comparing to the category descriptors based on CLIP [Menon et al., ICLR'23].



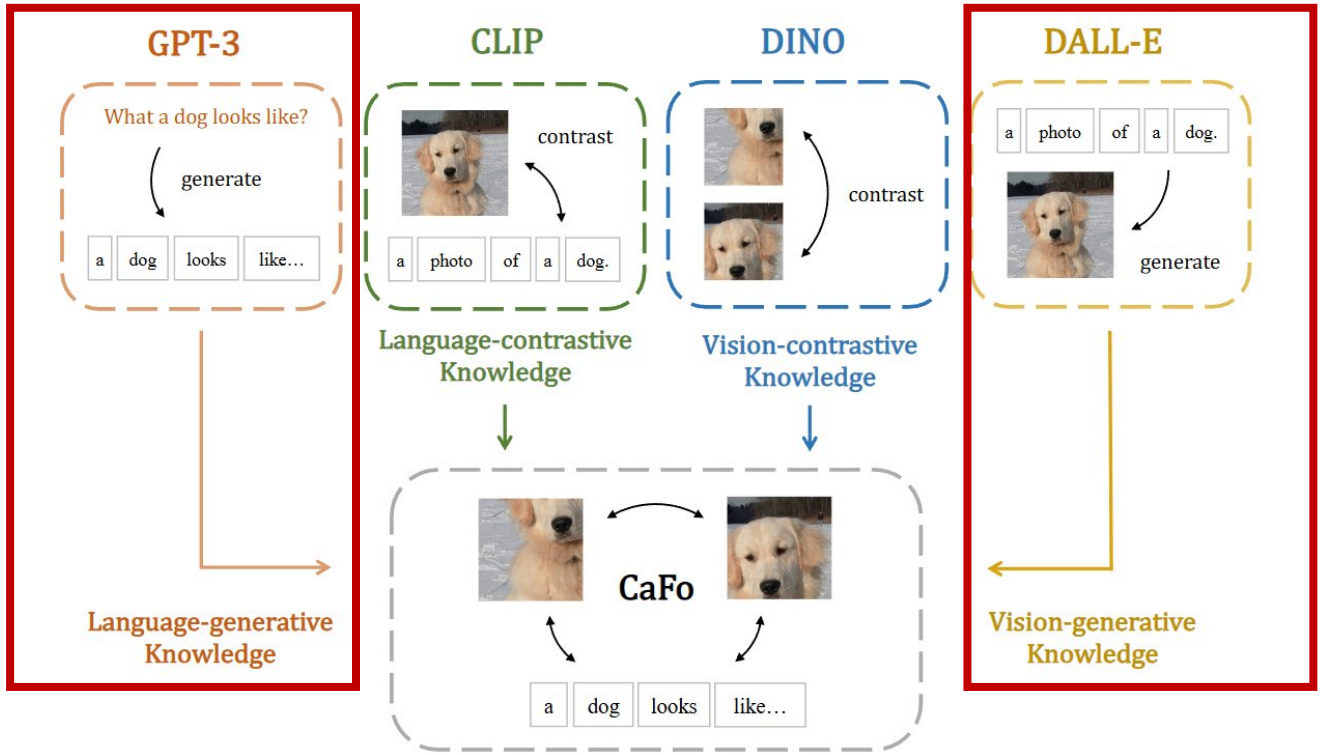
(a) (b)

We can obtain multiple zero-shot classifiers based on the generated texts, which helps the final prediction.

PTM for Expert Knowledge

- Uses GPT-3 to generate richer prompts for CLIP, DALL-E to synthesize additional training images, and a lightweight cache model to adaptively fuse CLIP + DINO predictions, achieving strong few-shot performance without extra manual data collection. [Zhang et al., CVPR'23]

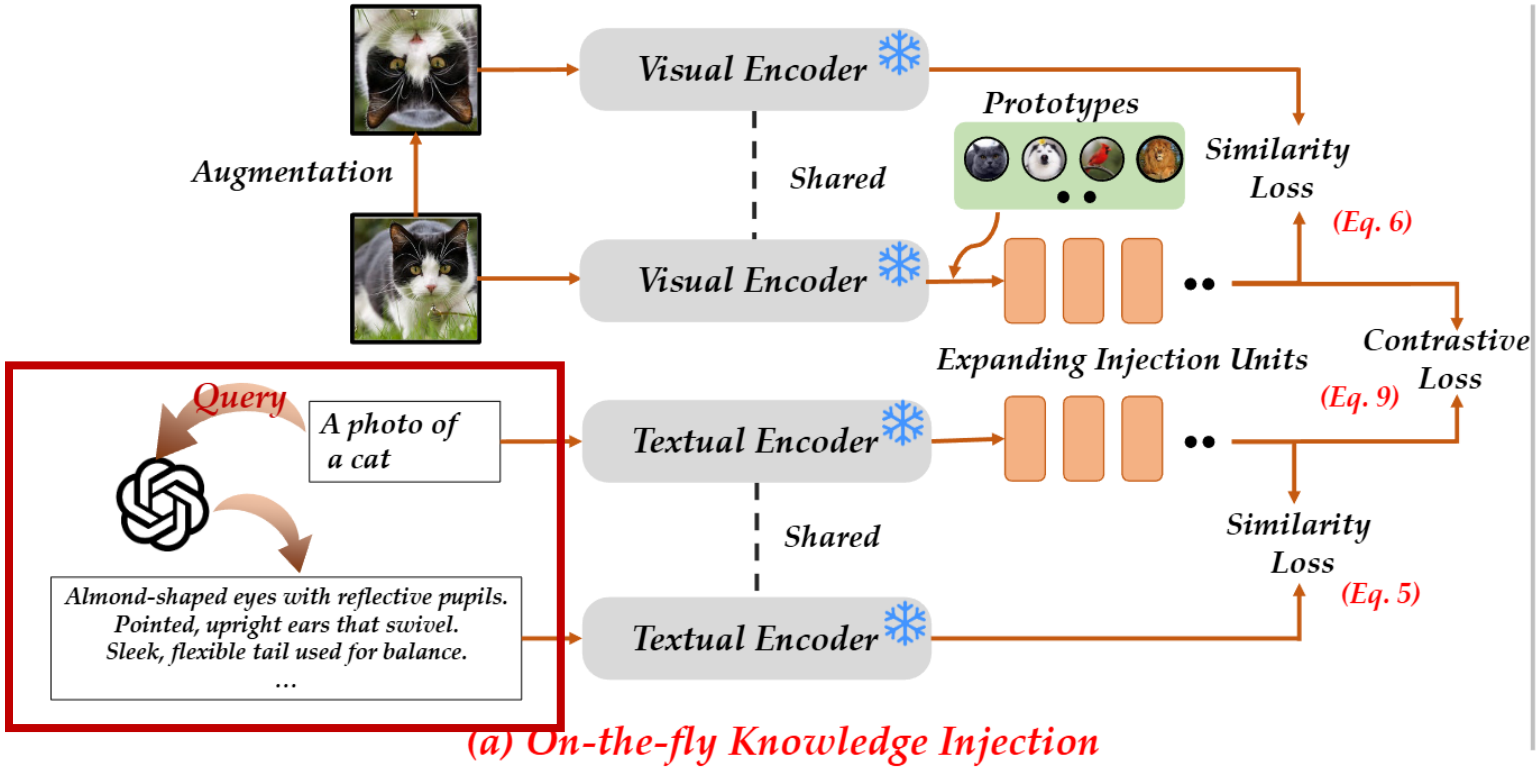
For expert knowledge



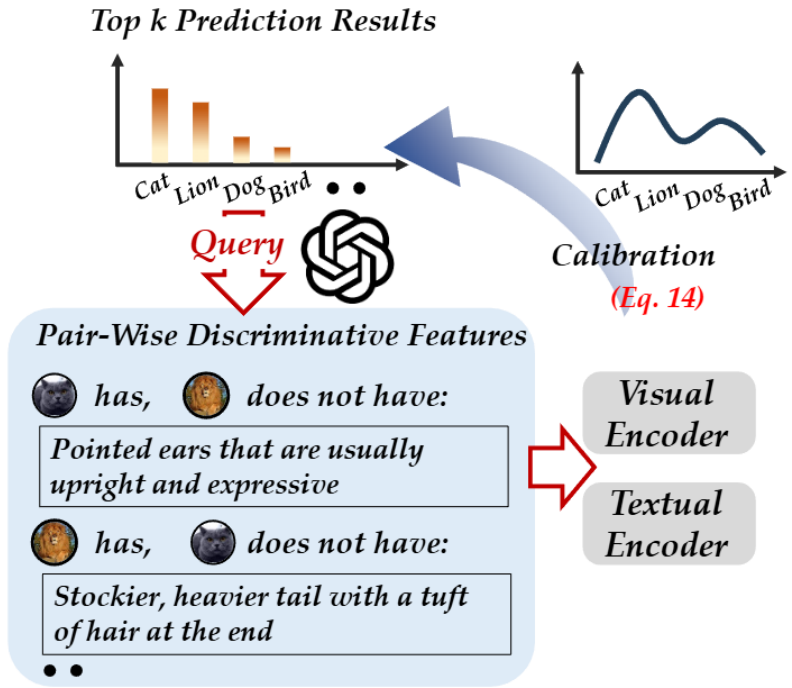
For data augmentation

PTM for Expert Knowledge Injection

- It uses a dual-branch injection tuning pipeline—augmenting visual features and using GPT-4–rewritten discriminative textual descriptors—plus an inference-time re-ranking step. [Zhou et al., ICCV'25]



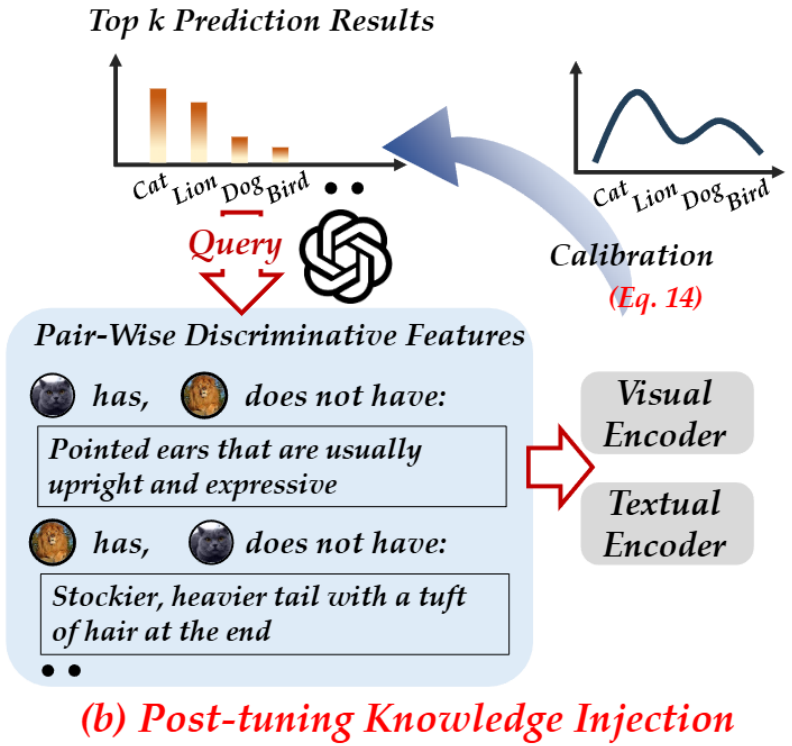
(a) On-the-fly Knowledge Injection



(b) Post-tuning Knowledge Injection

PTM for Expert Knowledge

- It uses a dual-branch injection tuning pipeline—augmenting visual features and using GPT-4–rewritten discriminative textual descriptors—plus an inference-time re-ranking step. [Zhou et al., ICCV'25]



(b) Post-tuning Knowledge Injection

Selecting top-K predictions

Generating pair-wise discriminative features among top-K classes

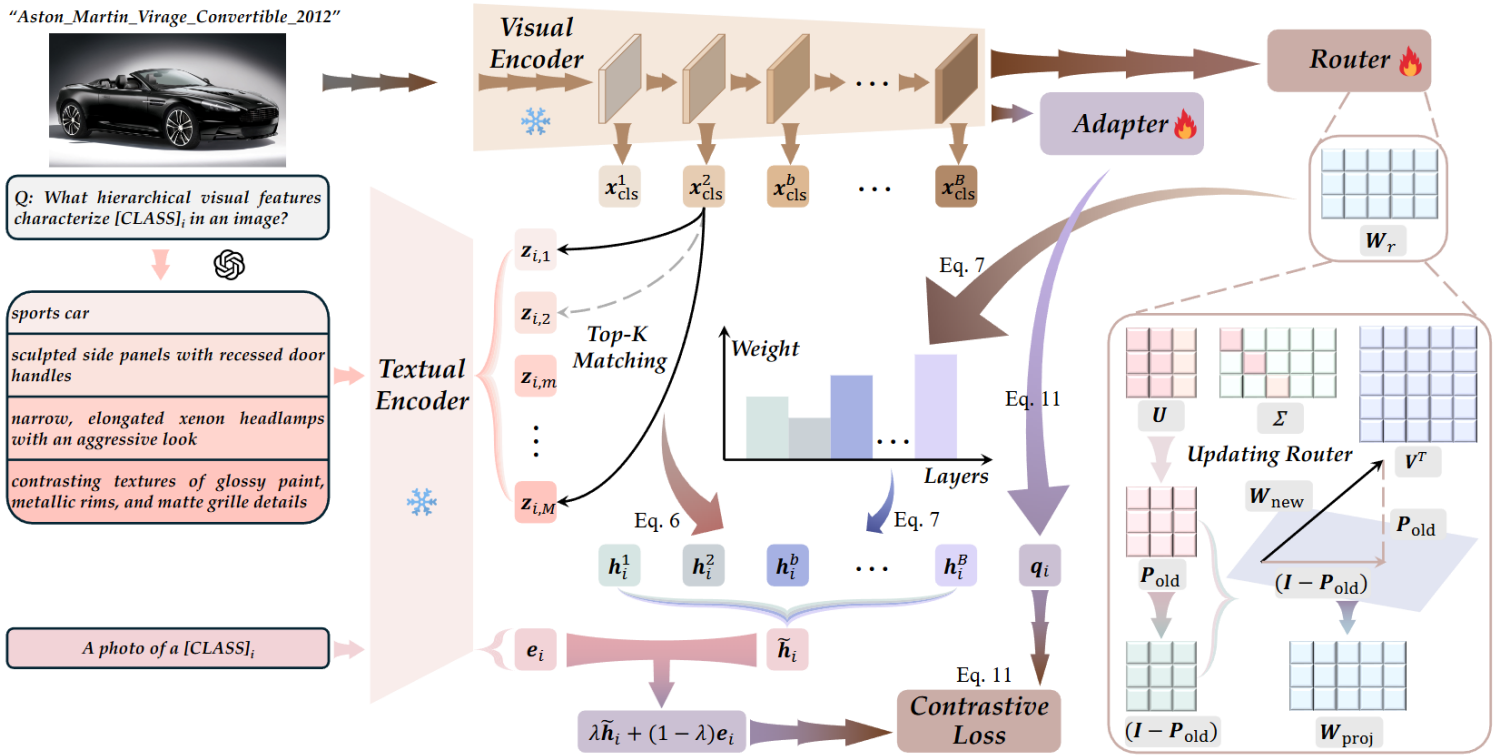
Build predictions with pair-wise discriminative features

Utilizing pair-wise discriminative features to refine the predictions

PTM for Expert Knowledge

- Using an LLM to recursively generate multi-level discriminative textual descriptors (coarse → fine) and matching them with multi-layer visual representations in CLIP. [Wen et al., CoRR'25]

Generating descriptions in a top-down manner via PTM

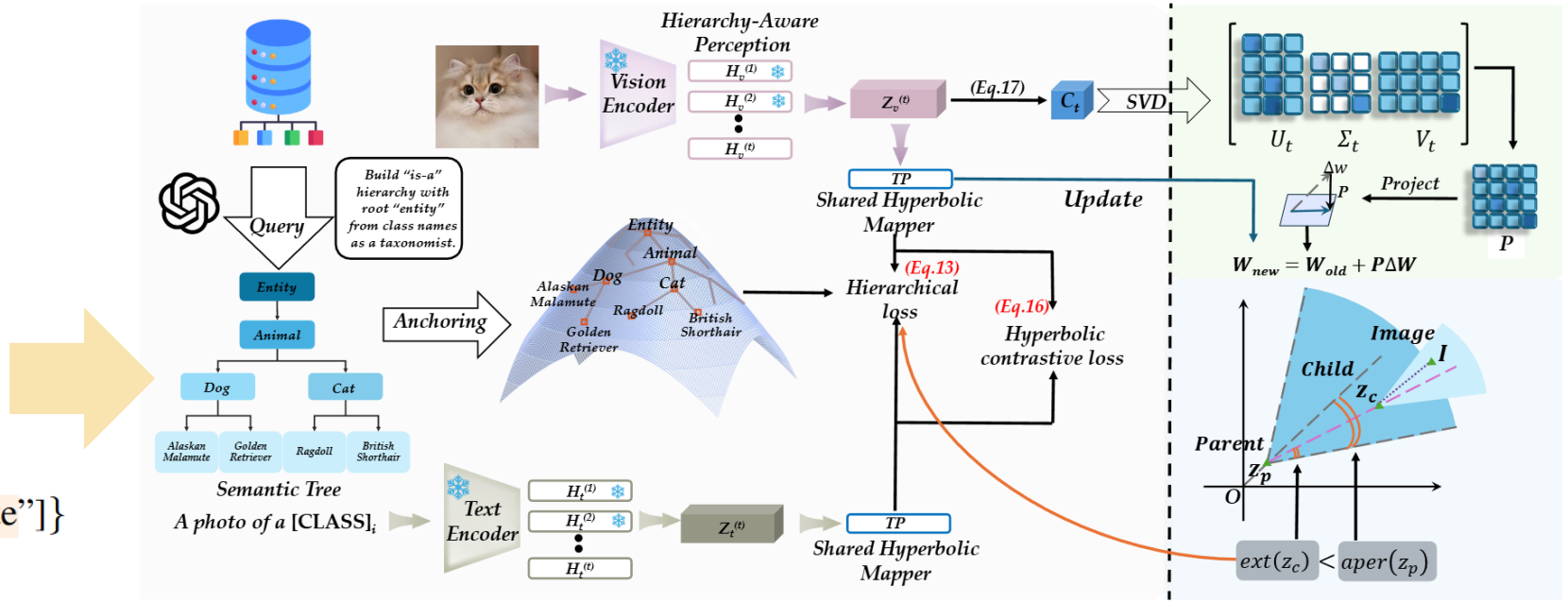


PTM for Expert Knowledge

- Leverages an external hierarchical semantic tree to embed features in hyperbolic space. [Hu et al., CoRR'25]

Q: You are a precise taxonomist. Given a list of class names, build a complete *is-a* hierarchy with a single root "entity". You may introduce abstract parent classes that are not included in the input list as needed to create a coherent hierarchy. Input list: ["alaskan malamute", "ragdoll", "golden retriever", "british shorthair"]

A: {"entity": ["animal"],
"animal": ["cat", "dog"],
"cat": ["ragdoll", "british shorthair"],
"dog": ["golden retriever", "alaskan malamute"]}

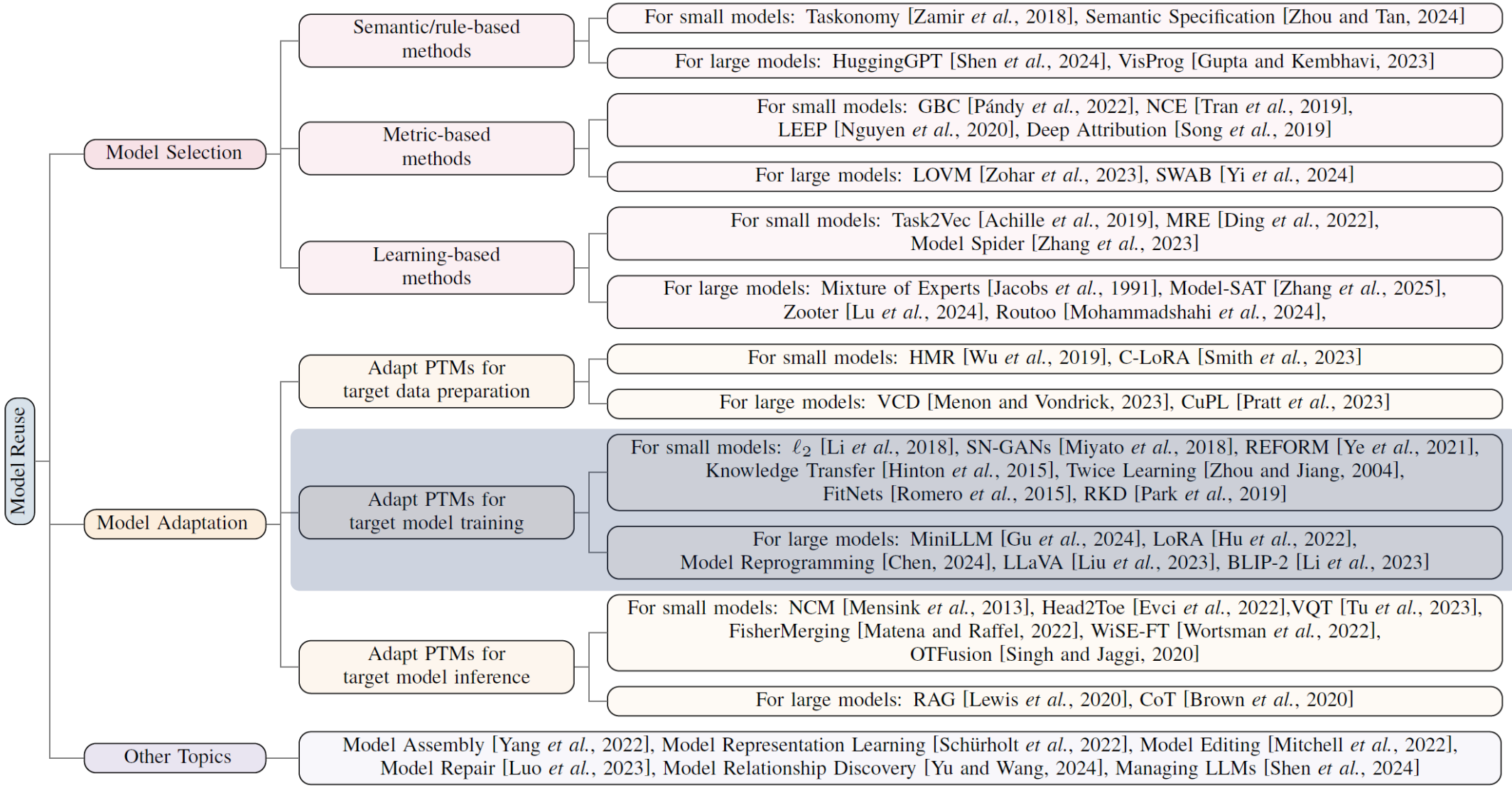


Generating semantic tree via PTM

Outline

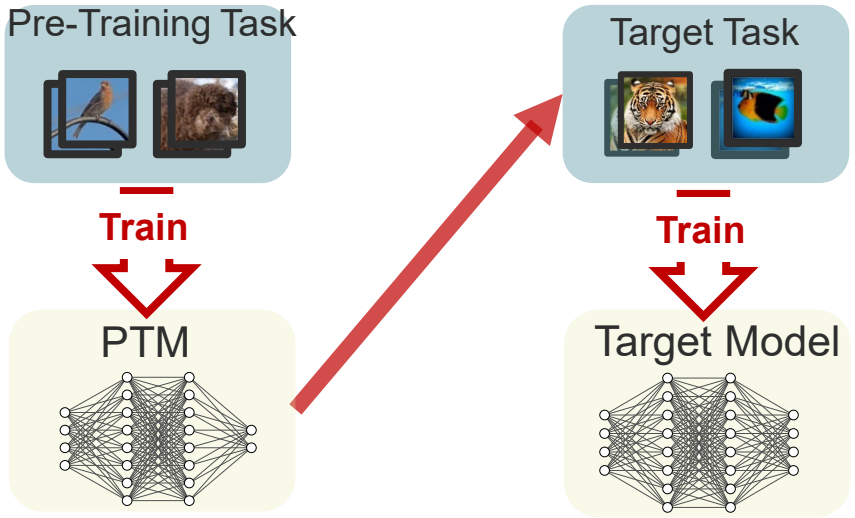
- *Taxonomy of Model Adaptation*
  - *Adapt PTMs for target data preparation*
  - ***Adapt PTMs for target model training***
  - *Adapt PTMs for target model inference*
- *Other topics in model reuse*
- *Conclusion*

Taxonomy of Model Reuse



Adapt PTMs for target model training

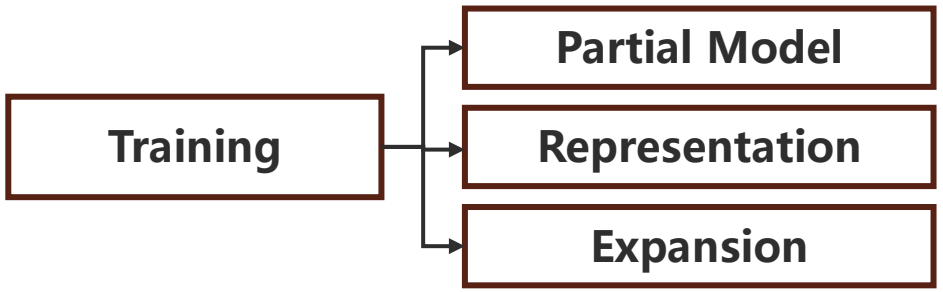
- Consider the target model f has the same architecture with the PTM g. To determine the parameter θ, could we fine-tune the whole model on target task $\mathcal{T}$?



*Requires **enough** target **data**.*

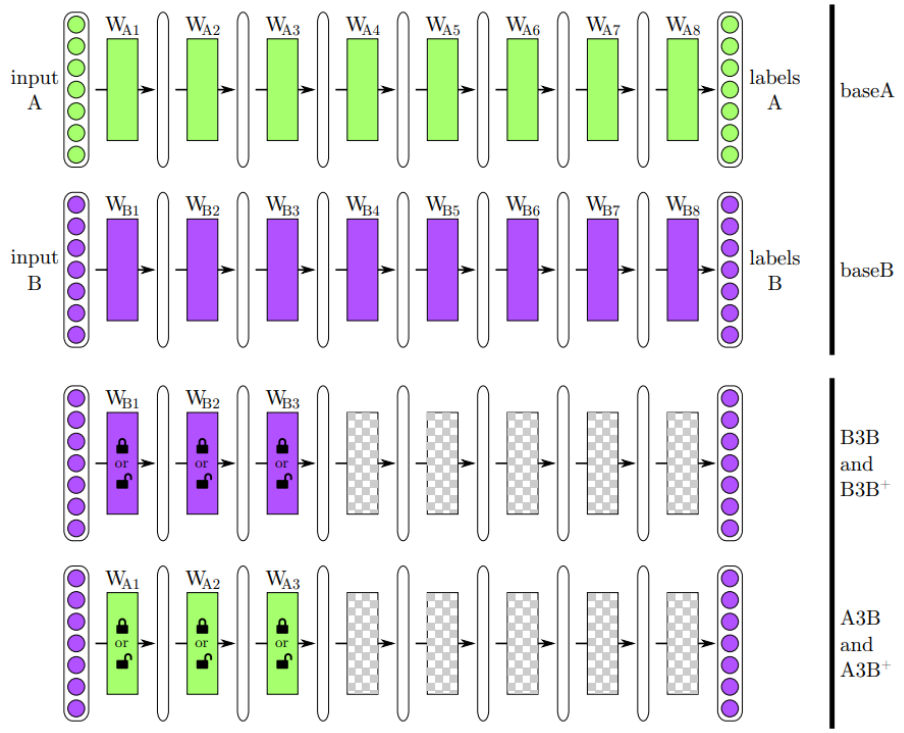
*The model may meet **catastrophic forgetting** when we want to facilitate the learned knowledge in the PM.*

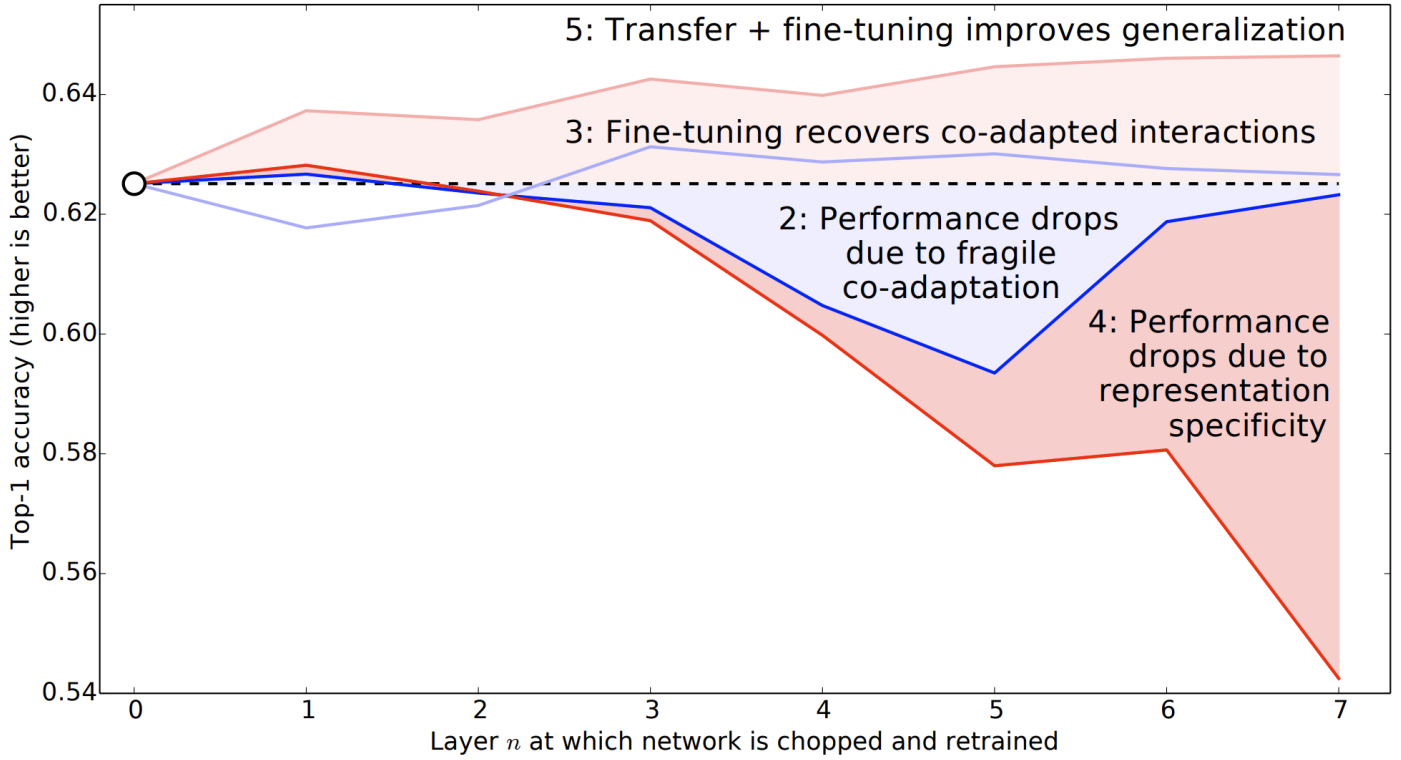
One solution is to reuse only ***a part of*** the PTM.



Fine-Tune the Whole Model

- Initialize a network with transferred features from almost any number of layers can produce a boost to generalization performance after fine-tuning to a new dataset [Yosinski et al., NeurIPS'14].





Tuning Partial Weights

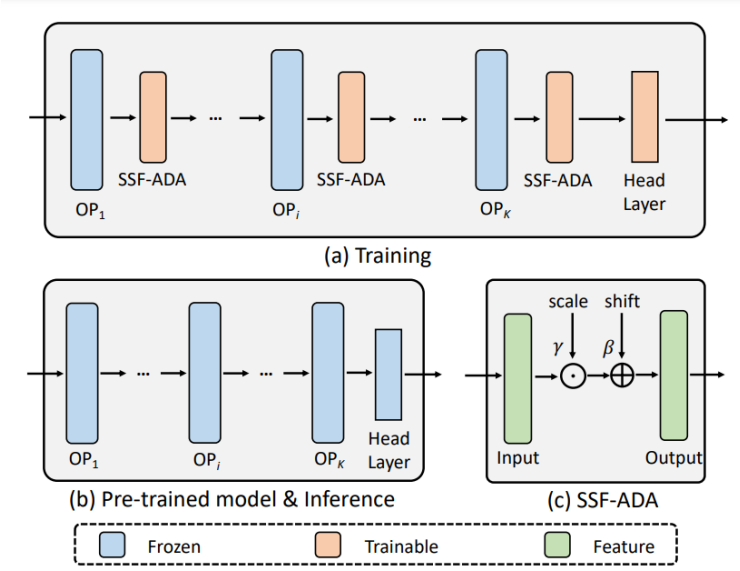
- Tune the whole model: model may overfit the target task and forget the pre-trained knowledge.
- Tune partial parameters: the ability of the model is limited.

*Tune **key** components to balance the efficiency and ability.*

- Partial-K [Jia et al., ECCV'22]: Tune the top-k layers of a PTM.
- Scale and Shift [Lian et al., NeurIPS'22]: tune the scale and shift parameters of the PTM.
- BitFit [Ben-Zaken et al., ACL'22] : tune (a subset of) bias-terms of the PTM

Tuning Partial Weights

- SSF only *Scale and Shift* the deep Features extracted by a pre-trained model to catch up with the performance of full finetuning [Lian et al., NeurIPS'22].



- BitFit is a sparse-finetuning method where only the ***bias**-terms* of the model (or a subset of them) are being modified [Ben-Zaken et al., ACL'22].

$$\mathbf{Q}^{m,\ell}(\mathbf{x}) = \mathbf{W}_q^{m,\ell}\mathbf{x} + \mathbf{b}_q^{m,\ell}$$
$$\mathbf{K}^{m,\ell}(\mathbf{x}) = \mathbf{W}_k^{m,\ell}\mathbf{x} + \mathbf{b}_k^{m,\ell}$$
$$\mathbf{V}^{m,\ell}(\mathbf{x}) = \mathbf{W}_v^{m,\ell}\mathbf{x} + \mathbf{b}_v^{m,\ell}$$

$$\mathbf{h}_1^\ell = att(\mathbf{Q}^{1,\ell}, \mathbf{K}^{1,\ell}, \mathbf{V}^{1,\ell}, .., \mathbf{Q}^{m,\ell}, \mathbf{K}^{m,\ell}, \mathbf{V}^{m,l})$$
$$\mathbf{h}_2^\ell = \text{Dropout}\left(\mathbf{W}_{m_1}^\ell \cdot \mathbf{h}_1^\ell + \mathbf{b}_{m_1}^\ell\right)$$
$$\mathbf{h}_3^\ell = \mathbf{g}_{LN_1}^\ell \odot \frac{(\mathbf{h}_2^\ell + \mathbf{x}) - \mu}{\sigma} + \mathbf{b}_{LN_1}^\ell$$

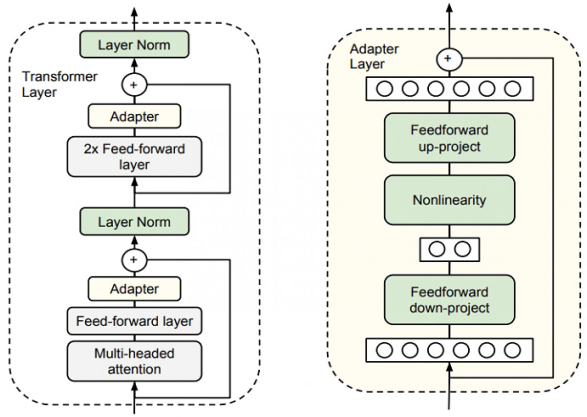
$$\mathbf{h}_4^\ell = \text{GELU}\left(\mathbf{W}_{m_2}^\ell \cdot \mathbf{h}_3^\ell + \mathbf{b}_{m_2}^\ell\right)$$
$$\mathbf{h}_5^\ell = \text{Dropout}\left(\mathbf{W}_{m_3}^\ell \cdot \mathbf{h}_4^\ell + \mathbf{b}_{m_3}^\ell\right)$$
$$\text{out}^\ell = \mathbf{g}_{LN_2}^\ell \odot \frac{(\mathbf{h}_5^\ell + \mathbf{h}_3^\ell) - \mu}{\sigma} + \mathbf{b}_{LN_2}^\ell$$

Tuning Add-ins

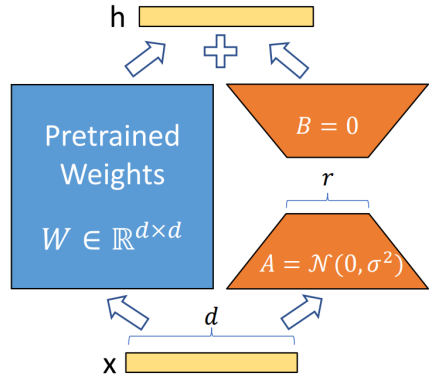
- Tune *add-in-like* parameters: adding extra structure and reusing the existed pre-trained knowledge with the original parameters remain fixed.

*Tune **key** components are efficiently learning incremental capabilities while reducing forgetting of existing knowledge.*

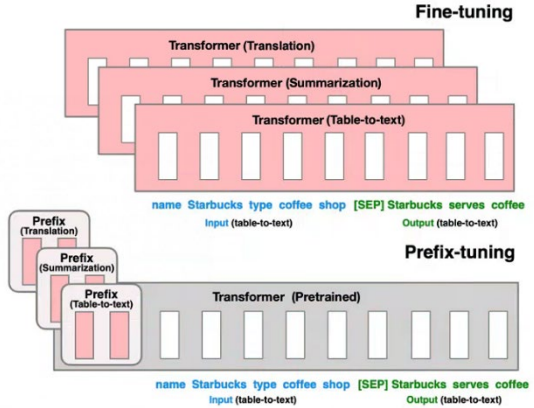
- **Adapter** [Houlsby et al., ICML'19]: Add and tune the adapter module after multiheaded attention and two feed-forward layers.
- **LoRA** [Hu et al., ICLR'22]: Inject and tune trainable rank decomposition matrices into query and value projection heads.
- **Prefix** [Li et al., ACL'21]: Combine a small, continuous task-specific vector (prefix) as the token-level trainable inputs.



Adapter: Adapting the outputs within a Transformer block and adjust the output before concat.



LoRA: Constructing low-rank tuning matrices to make it both storage- and compute-efficient.

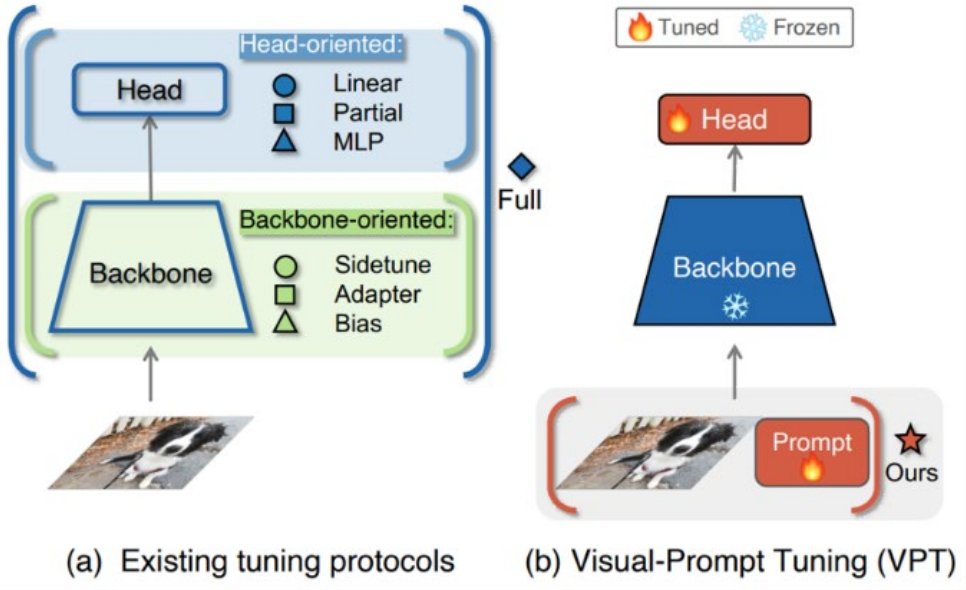


Prefix: Add the token-based adaptation items to harmonize inputs across variations.

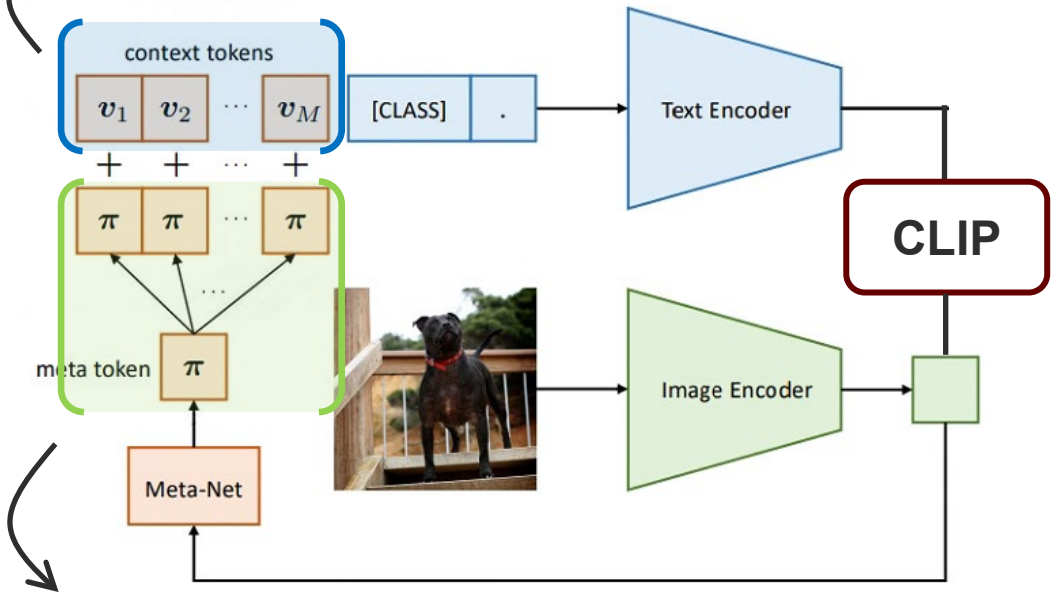
Tuning Add-ins

- Tune the PTM (ViT) with additional (visual or textual) prompts.

Visual prompt tuning (VPT) adds extra parameters in the input space while keeping the model backbone frozen [Jia et al., ECCV'22].



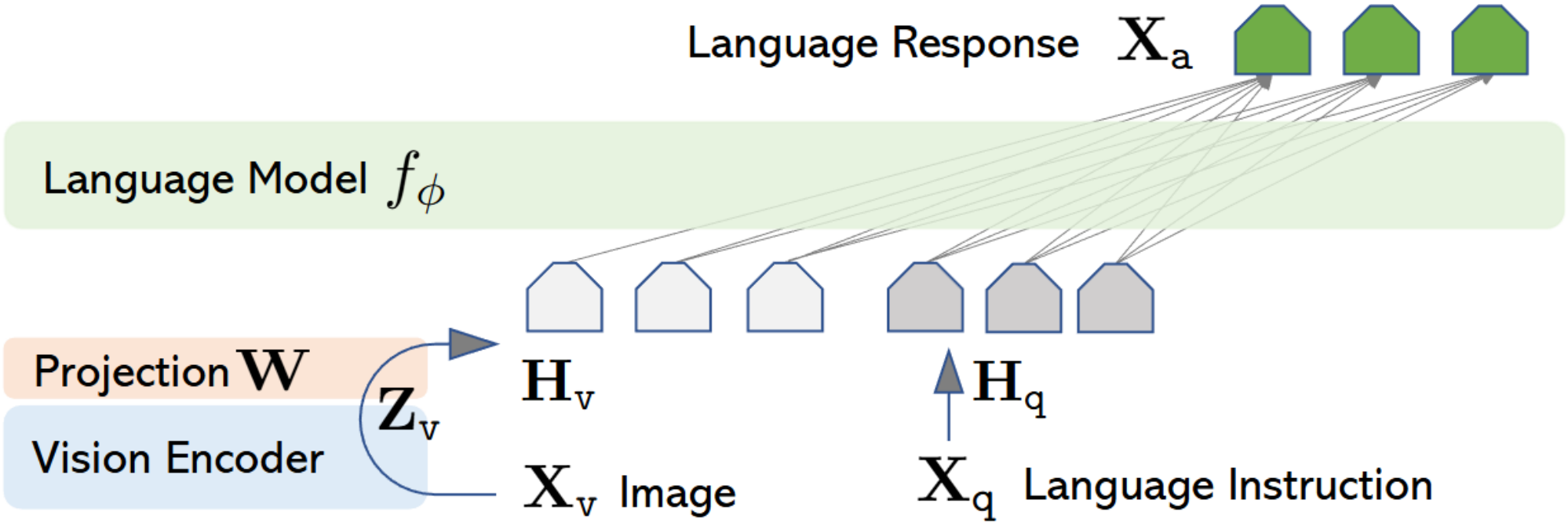
CoOp [Zhou et al., IJCV'22]:
Context optimization to prompt class words.



CoCoOp [Zhou et al., CVPR'22]:
Conditional Context optimization.

Merging Multimodal Modules

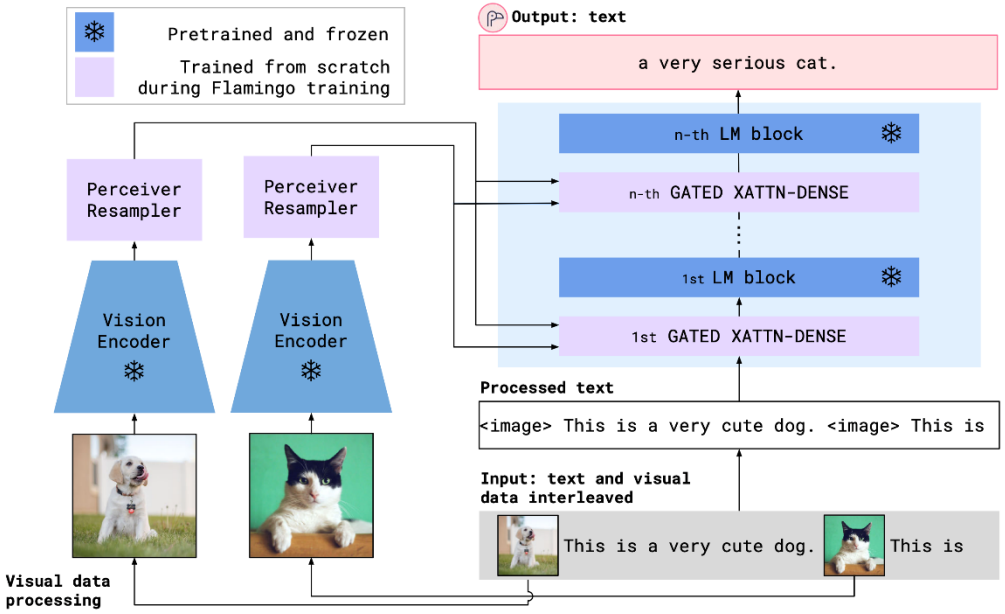
- Utilizing pre-trained image encoder and language model to build a multimodal LLM [Liu et al., NeurIPS'23]



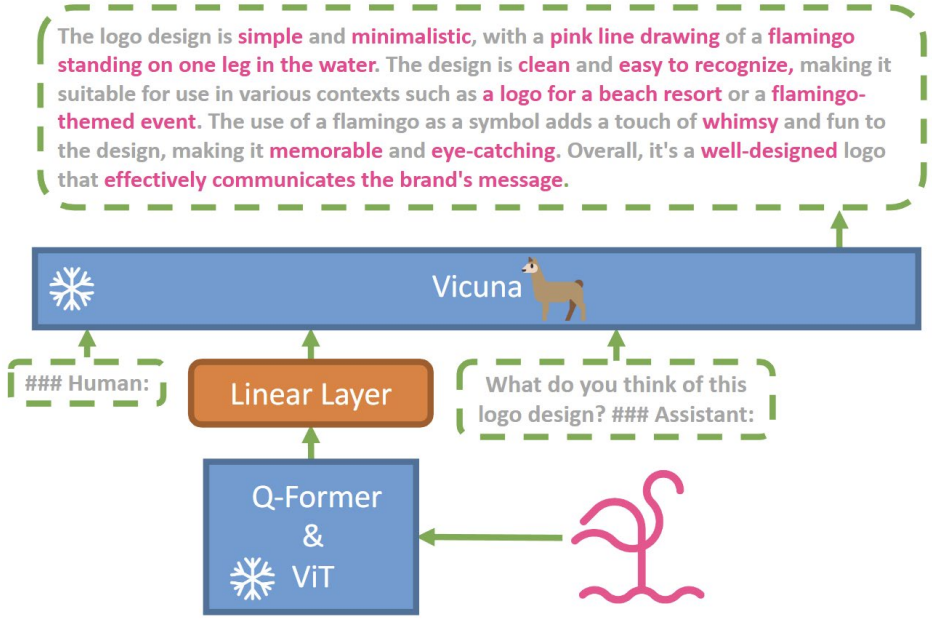
Merging Multimodal Modules

- Other multimodal LLMs can be built with other projectors

OpenFlamingo combines a pretrained vision encoder and a language model using cross attention layers. [Awadalla et al., CoRR'23].



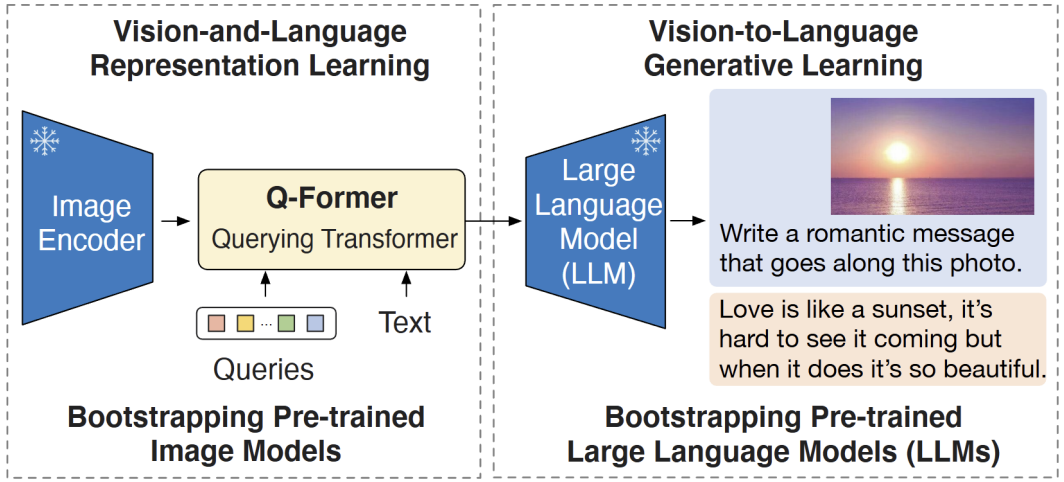
MiniGPT-4 only requires training the linear projection layer to align the visual features with the Vicuna. [Zhu et al., CoRR'23].



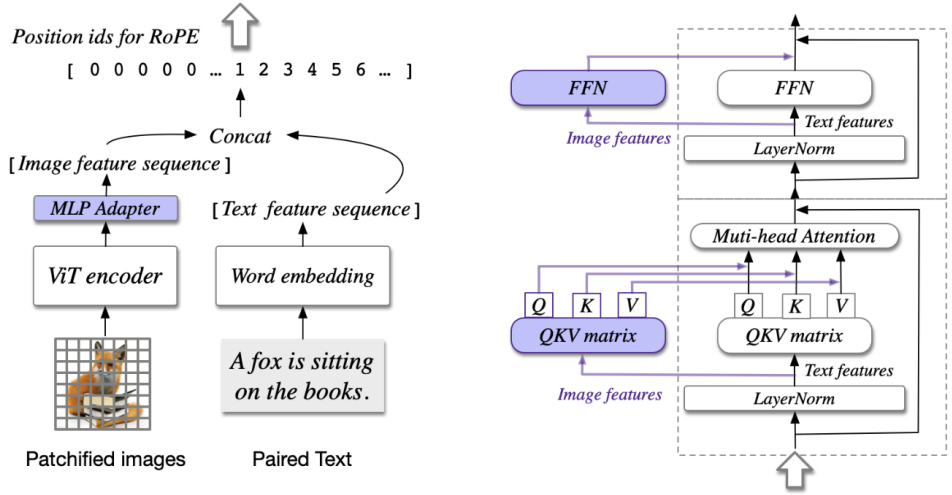
Merging Multimodal Modules

- Other multimodal LLMs can be built with other projectors

BLIP-2 bridges the modality gap between PTMs with a lightweight Querying Transformer, which is pretrained in two stages. [Li et al., ICML'23].

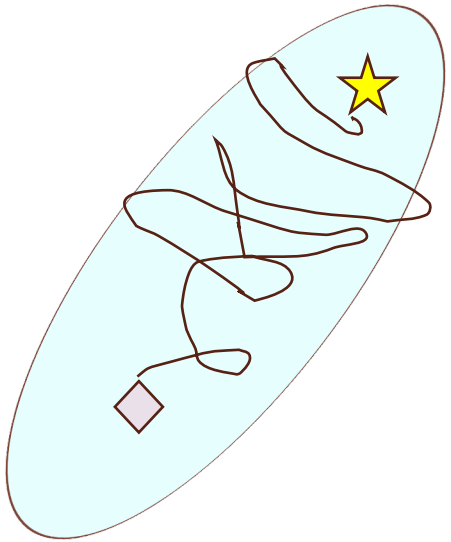


CogVLM bridges the gap between the frozen pretrained language model and image encoder by a trainable visual expert module in the attention and FFN layers.. [Wang et al., NeurIPS'24].

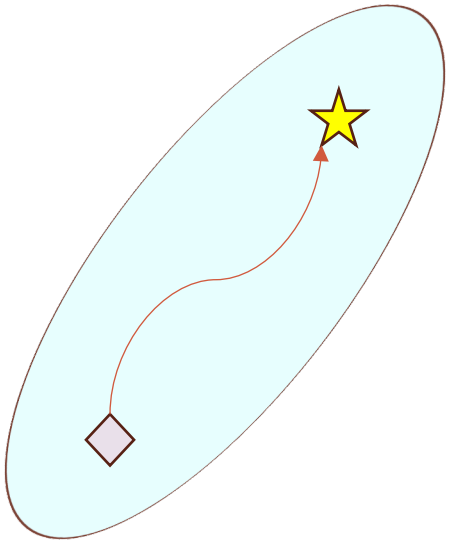


Reuse PTM for optimization

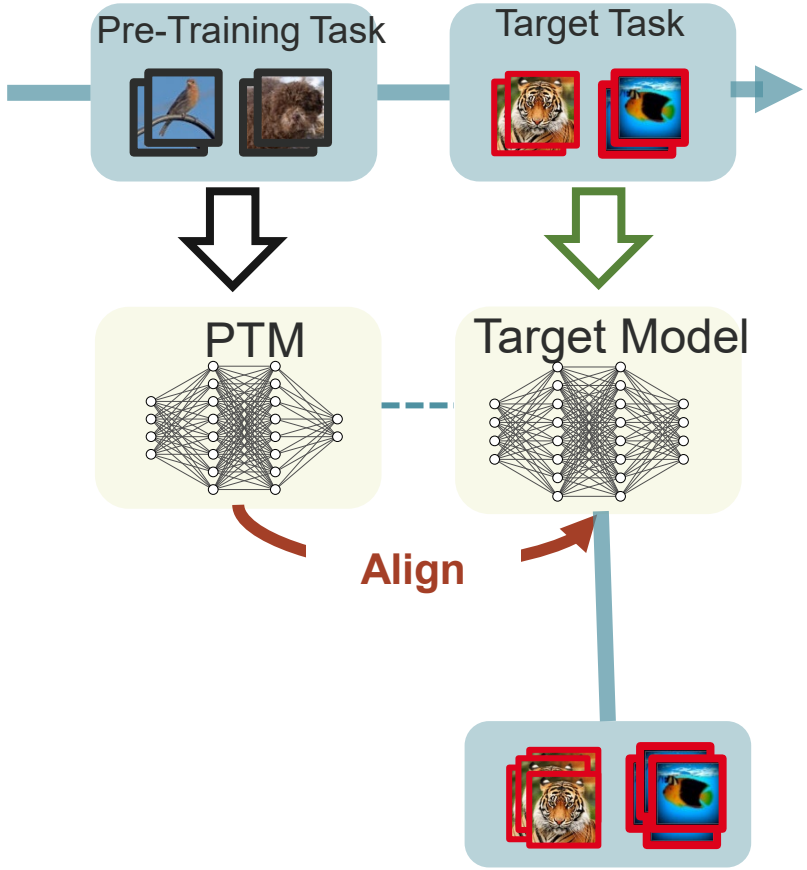
- Reuse PTM to find a better path from the initialization to the optimal point.



Without PTM



With PTM



Model Reuse via Regularization

- Hypothesis Transfer Learning (HTL): matches the weights between the target and PTM [Kuzborskij and Orabona, ICML'13] [Kuzborskij and Orabona, MLJ'16]. Given g_Θ, we learn f_θ via

$$\min_{f_\theta} \sum_{(\boldsymbol{x}_i, y_i)\sim\mathcal{T}} \ell(f_\theta(\boldsymbol{x}_i|g_\Theta), y_i) + \Omega(\theta, \Theta)$$

$\Omega(\cdot)$ measures the discrepancy between two sets of parameters, which could be implemented via different norms.

Regularize parameters with ℓ_2 norm

$$\Omega(\theta, \Theta) = \lambda\|\theta - \Theta\|_2^2$$

Layer-wise regularization for deep neural network.

Make prediction for the residual part. If $\Delta = \theta - \Theta$, then for a linear model

$$f_\theta(\boldsymbol{x}_i|g_\Theta) = g_\Theta(\boldsymbol{x}_i) + \langle\Delta, \boldsymbol{x}_i\rangle$$

$$\Omega(\theta, \Theta) = \lambda\|\Delta\|_2^2$$

Model Reuse via Regularization

- L^2-SP: consider the mixed norm, also the norm on the target parameter [Li et al., ICML'18]

$$\Omega(\theta, \Theta) = \lambda \|\theta - \Theta\|_2^2 + \lambda_2 \|\theta\|_2^2$$

- L^2-SP-Fisher: use the estimated Fisher information $\boldsymbol{F}$ to define the distance between θ, Θ

$$\Omega(\theta, \Theta) = \lambda (\theta - \Theta)^\top diag(\boldsymbol{F})(\theta - \Theta) + \lambda_2 \|\theta\|_2^2$$

- L^1-SP: $\ell 1$-norm is used

$$\Omega(\theta, \Theta) = \lambda \|\theta - \Theta\|_1^2 + \lambda_2 \|\theta\|_2^2$$

Table 2. Average classification accuracies (in %) of L^2, L^2-*SP* and L^2-*SP-Fisher* on 5 different runs. The source database is Places 365 for MIT Indoors 67 and ImageNet for Stanford Dogs 120 and Caltech 256.

	MIT Indoors 67	Stanford Dogs 120	Caltech 256 – 30	Caltech 256 – 60
L^2	79.6±0.5	81.4±0.2	81.5±0.2	85.3±0.2
L^2-*SP*	**84.2±0.3**	**85.1±0.2**	**83.5±0.1**	**86.4±0.2**
L^2-*SP-Fisher*	84.0±0.4	**85.1±0.2**	83.3±0.1	86.0±0.1

Why Hypothesis Transfer Helps?

- For the binary classification case with linear classifier [Kuzborskij and Orabona, MLJ'16],

Theorem 2 *Let $h_{\hat{\mathbf{w}},\boldsymbol{\beta}}$ be generated by Regularized ERM, given the m-sized training set S sampled i.i.d. from the target domain, source hypotheses $\{h_i^{src} : \|h_i^{src}\|_\infty \leq 1\}_{i=1}^n$, any source weights $\boldsymbol{\beta}$ obeying $\Omega(\boldsymbol{\beta}) \leq \rho$, and $\lambda \in \mathbb{R}_+$. Assume that $\ell(h_{\hat{\mathbf{w}},\boldsymbol{\beta}}(\mathbf{x}), y) \leq M$ for any $(\mathbf{x}, y)$ and any training set. Then, denoting $\kappa = \frac{H}{\sigma}$ and assuming that $\lambda \leq \kappa$, we have with probability at least $1 - e^{-\eta}$, $\forall \eta \geq 0$*

$$R(h_{\hat{\mathbf{w}},\boldsymbol{\beta}}) \leq \hat{R}_S(h_{\hat{\mathbf{w}},\boldsymbol{\beta}}) + \mathcal{O}\left(\frac{R^{src}\kappa}{\sqrt{m}\lambda} + \sqrt{\frac{R^{src}\rho\kappa^2}{m\lambda}} + \frac{M\eta}{m\log\left(1+\sqrt{\frac{M\eta}{u^{src}}}\right)}\right) \tag{4}$$

$$\leq \hat{R}_S(h_{\hat{\mathbf{w}},\boldsymbol{\beta}}) + \mathcal{O}\left(\frac{\kappa}{\sqrt{m}}\left(\frac{R^{src}}{\lambda} + \sqrt{\frac{R^{src}\rho}{\lambda}}\right) + \frac{\kappa}{m}\left(\frac{\sqrt{R^{src}M\eta}}{\lambda} + \sqrt{\frac{\rho}{\lambda}}\right)\right), \tag{5}$$

- For the multi-class case with linear classifier [Ye, Zhan, Jiang, Zhou, ICML'18][Zhao, Cai, Zhou, MLJ'20].
- Hypothesis transfer on deep neural network [Gouk, Hospedales, Pontil, ICLR'21].
- Other analysis [Aghbalou et al., ICML'23].

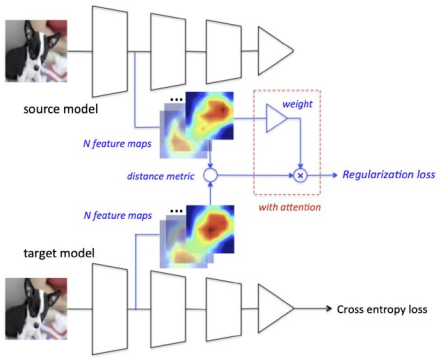
Model Reuse via Feature Matching

- Minimize the discrepancy between the features generated by PTM and the target model:

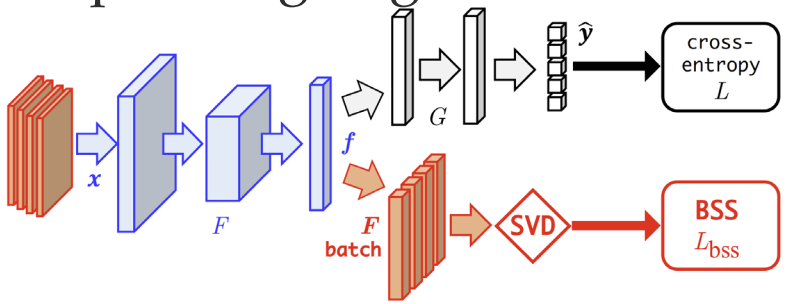
$$\min_{f_\theta} \sum_{(x_i,y_i)\sim\mathcal{T}} \ell(f_\theta(\boldsymbol{x}_i|g_\Theta), y_i) + \Omega\big(f_\theta(x_i), g_\Theta(x_i)\big)$$

- f_θ and g_Θ output the predictions (logit) or the middle layer features.

- DELTA [Li et al., ICLR'19]:
  - Feature alignment with attention.



- BSS [Chen et al., NeurIPS'19]:
  - Feature alignment based on corresponding angle.



Knowledge Matching as New Objective

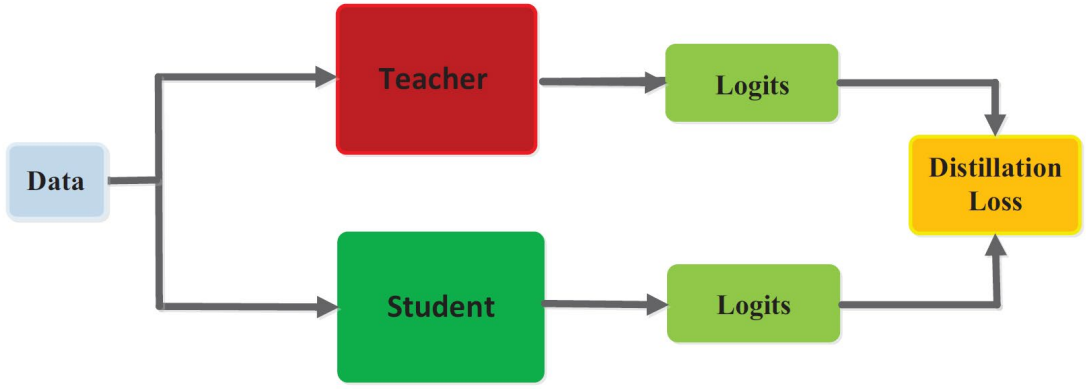
- $\Omega\left(f_\theta(x_i), g_\Theta(x_i)\right)$ measures the discrepancy between predictions.
- Knowledge distillation [Hinton et al., CORR'15] match the PTM and the target model in the prediction space, i.e., the output predictions of both models should be similar.

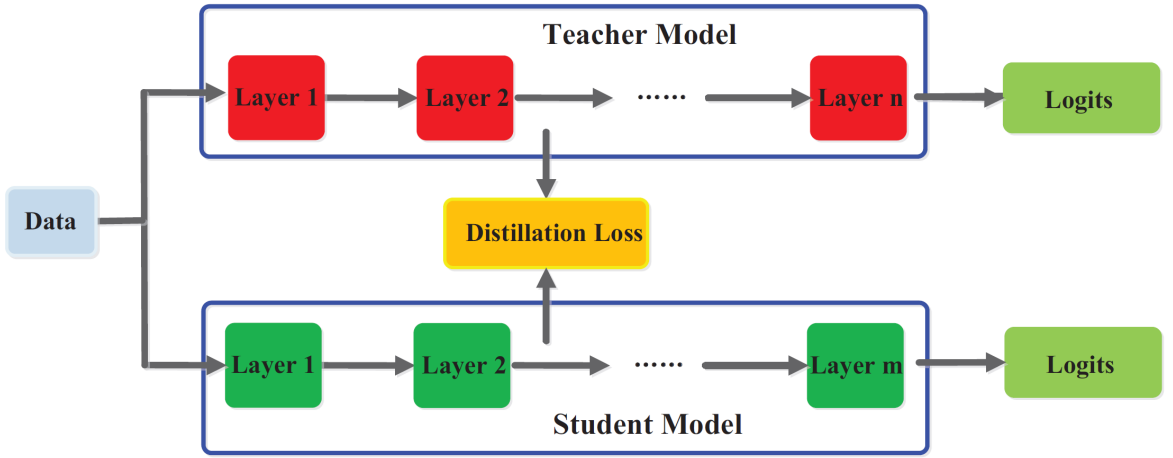
$$\Omega\left(f_\theta(x_i), g_\Theta(x_i)\right) = KL(f_\theta(x_i)/\tau, g_\Theta(x_i)/\tau)$$

The logit should be transformed with SoftMax operator.

The temperature τ calibrates the predictions.

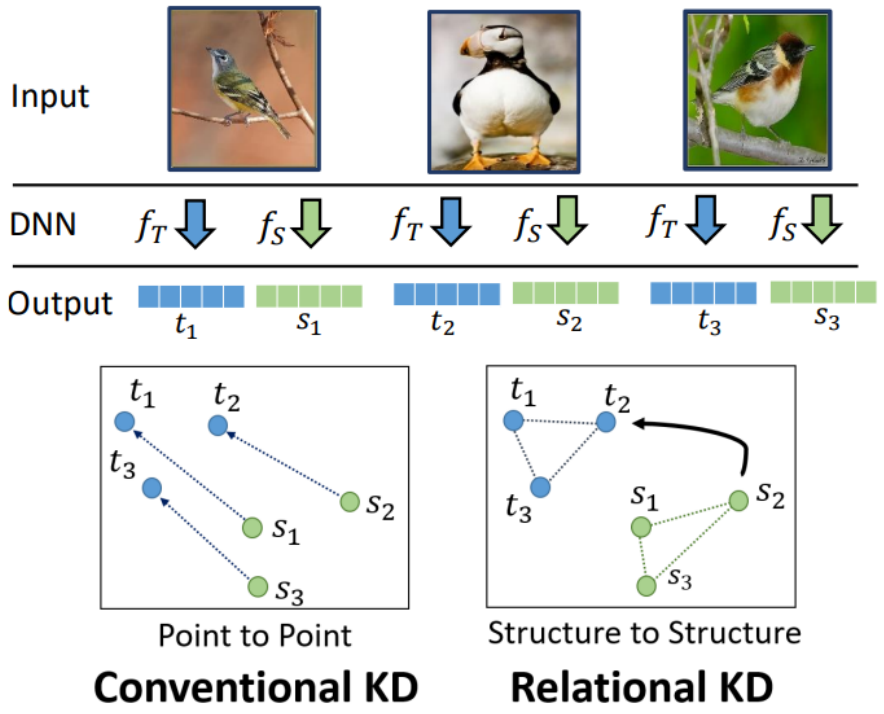
- Other kinds of divergence measure, or knowledge types are also considered [Gou et al., IJCV'21].



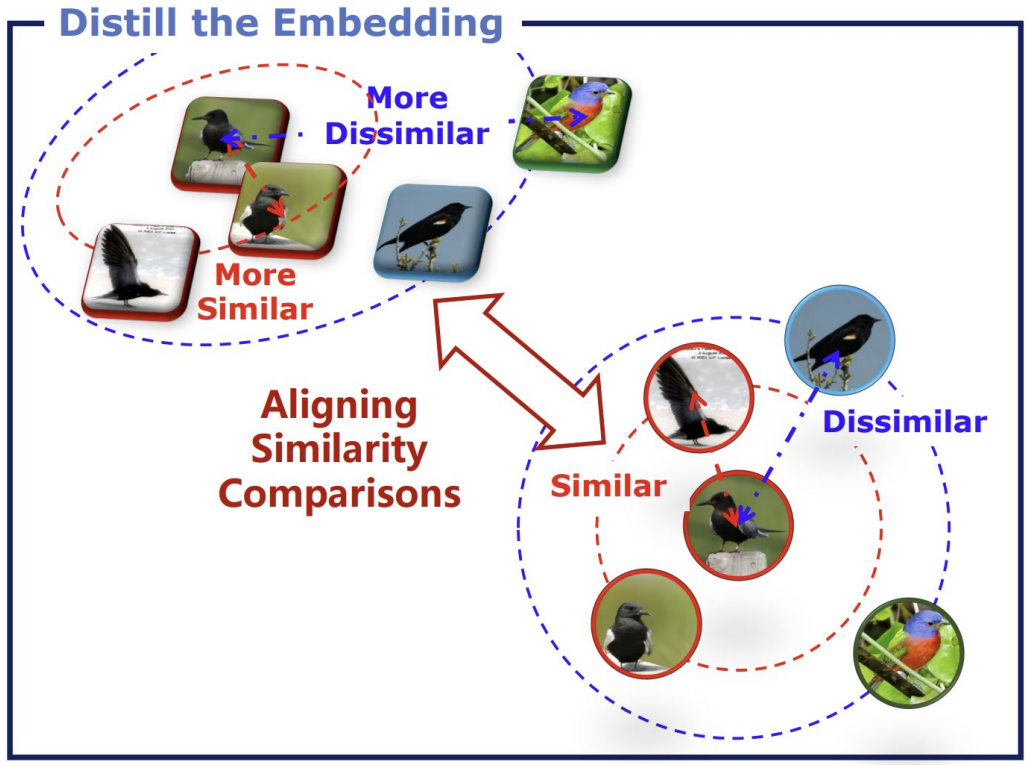


Knowledge Matching as New Objective

- Match the *relationships* between instances [Park et al., CVPR'19] [Ye, Lu, Zhan,CVPR'20], prove to be powerful tools for achieving alignment.



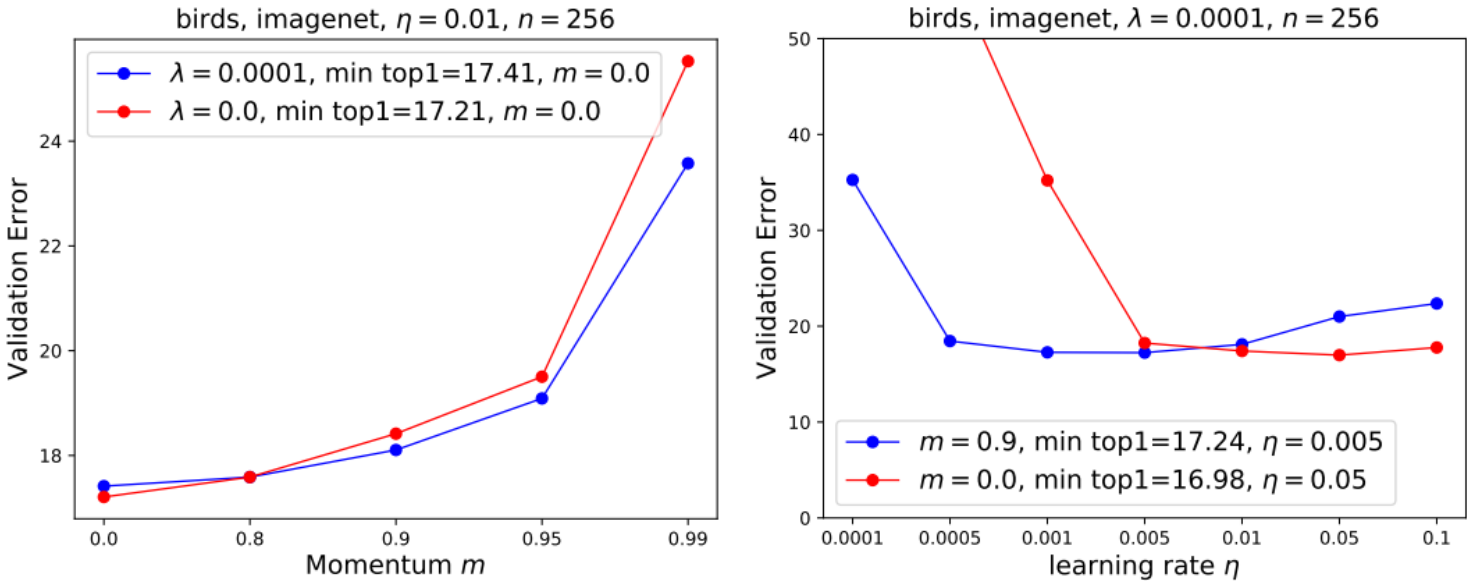
The angle between pairwise instances are matched.



The triplet or tuple-level comparisons are matched.

PTM Helps Optimization

- PTMs can also influence the optimization process of the target model. The choices of different hyper-parameters such as learning rate, weight decay [Mahajan et al., CoRR'18] [Kornblith, Shlens, Le, CVPR'19], *momentum* [Li et al., ICLR'20] during the vanilla fine-tuning of the target model are explored.



Small momentum works better for fine-tuning on domains that are close to the source domain

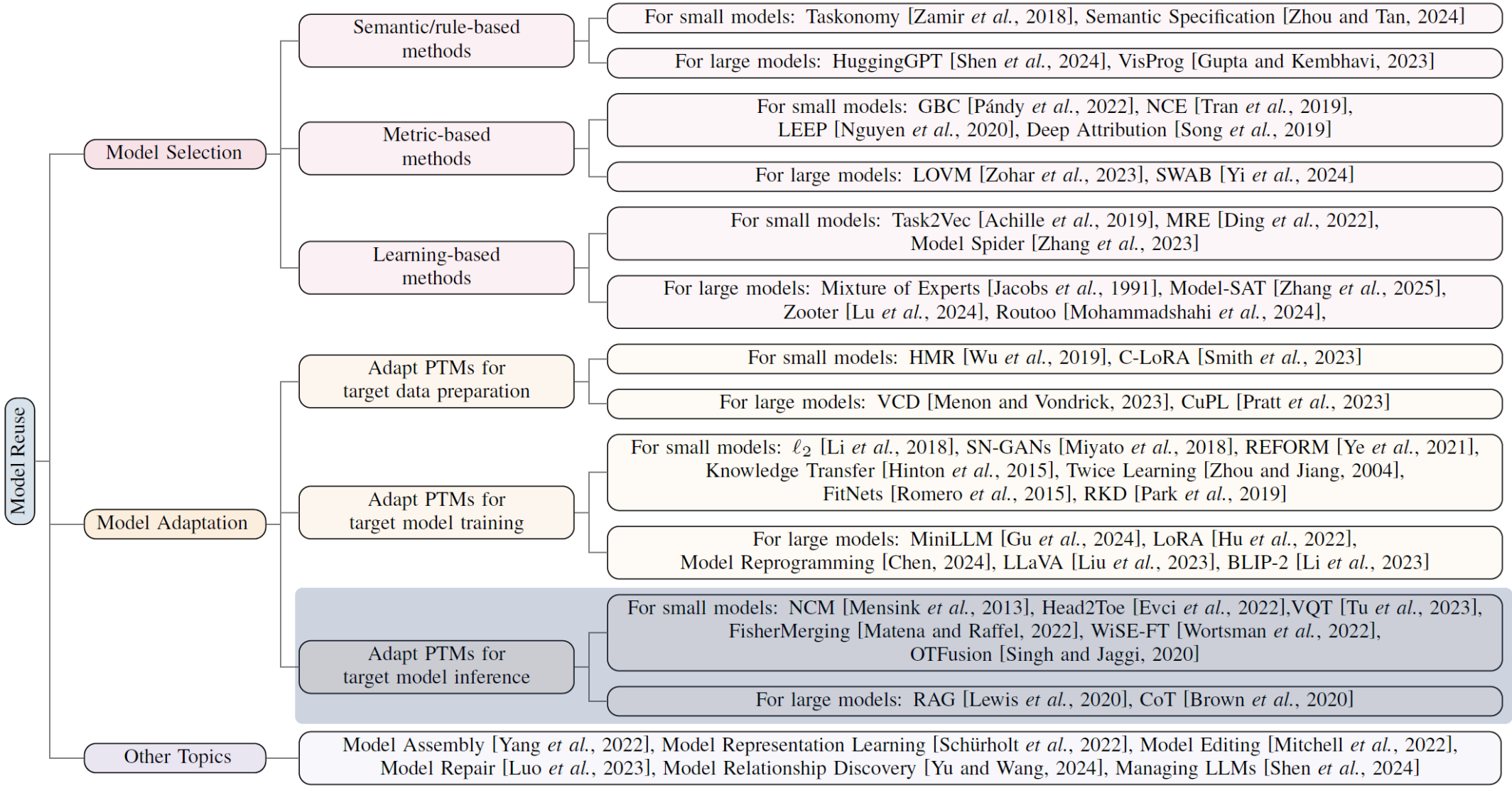
Optimal hyperparameters for fine-tuning depend on both the target dataset **and the similarity between the source domain and target domain**.

Regularization that keeps models close to the initial model does not necessarily apply for "dissimilar" datasets.

Outline

- *Taxonomy of Model Adaptation*
  - *Adapt PTMs for target data preparation*
  - *Adapt PTMs for target model training*
  - ***Adapt PTMs for target model inference***
- *Other topics in model reuse*
- *Conclusion*

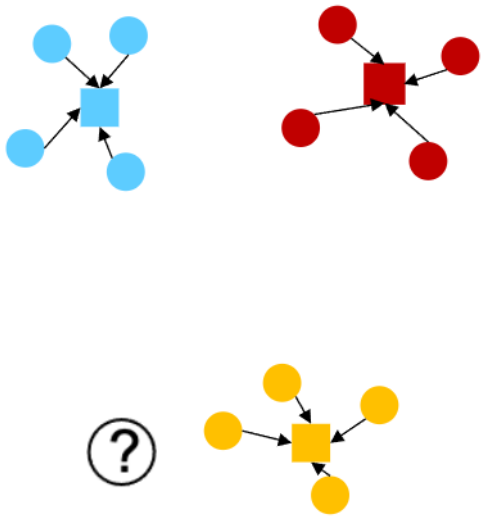
Taxonomy of Model Reuse



Adapt PTMs for target model inference

Main idea: construct an embedding space using the PTM, resulting in more discriminative features.

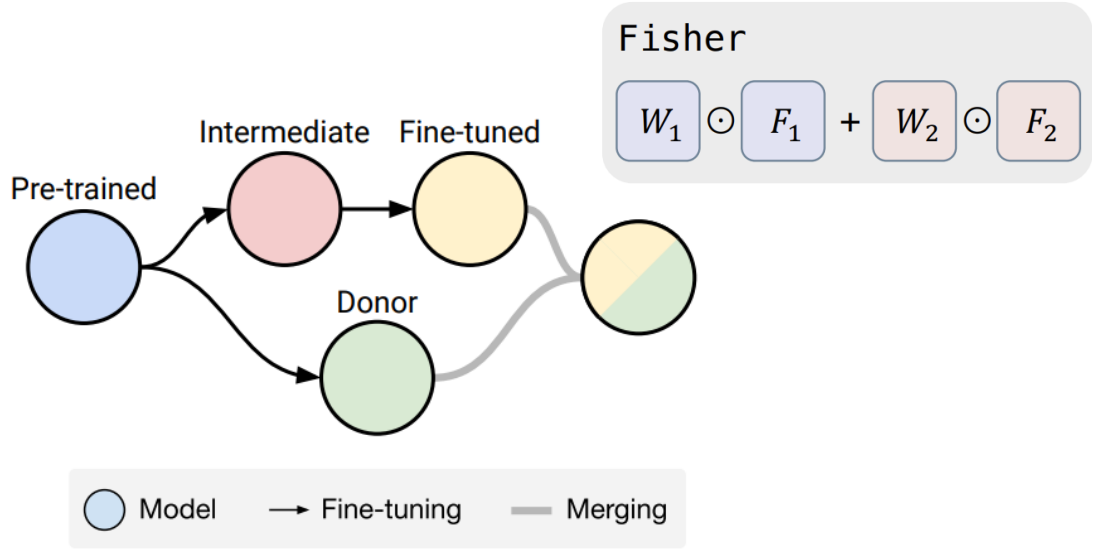
Directly using the pre-trained features as the encoding for target task inference



$$c^* = \underset{c \in \{1,\dots,C\}}{\operatorname{argmin}} \; d(\boldsymbol{x}, \boldsymbol{\mu}_c),$$

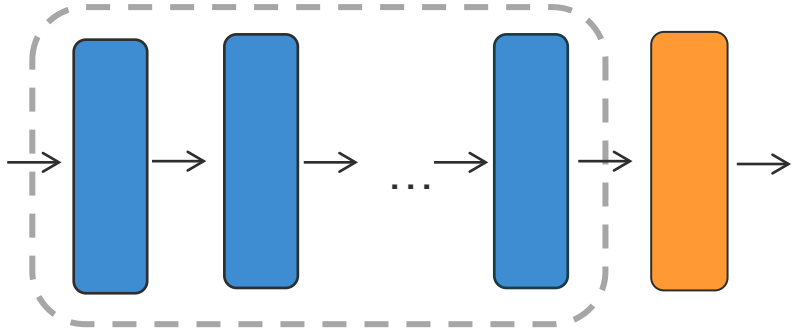
$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{i:y_i=c} \boldsymbol{x}_i,$$

Merging models with Fisher-weighted averaging [Matena et al., NIPS'22].



PTM as Better Representations

- Linear probing : freezing the backbone (PTM) and adjusting only the fully connected (FC) layers based on downstream task data.



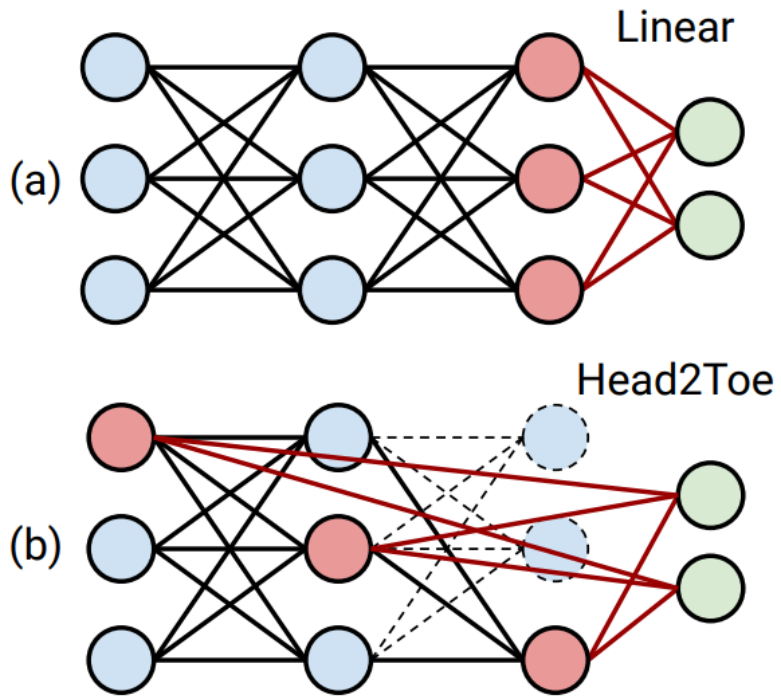
- The Nearest Class Mean (NCM) classifier assigns images to the class whose mean is closest. PTM could enhance this process by improving feature representation and metric learning [Mensink et al., TPAMI'13].
- A powerful baseline method in few-shot learning [Wang et al., CoRR'19], class-imbalance learning [Kang et al., ICLR'20], and class-incremental learning [Rebuffi et al., CVPR'17][Zhou et al., CoRR'23].

$$c^* = \underset{c\in\{1,\dots,C\}}{\operatorname{argmin}} \; d(\boldsymbol{x}, \boldsymbol{\mu}_c),$$

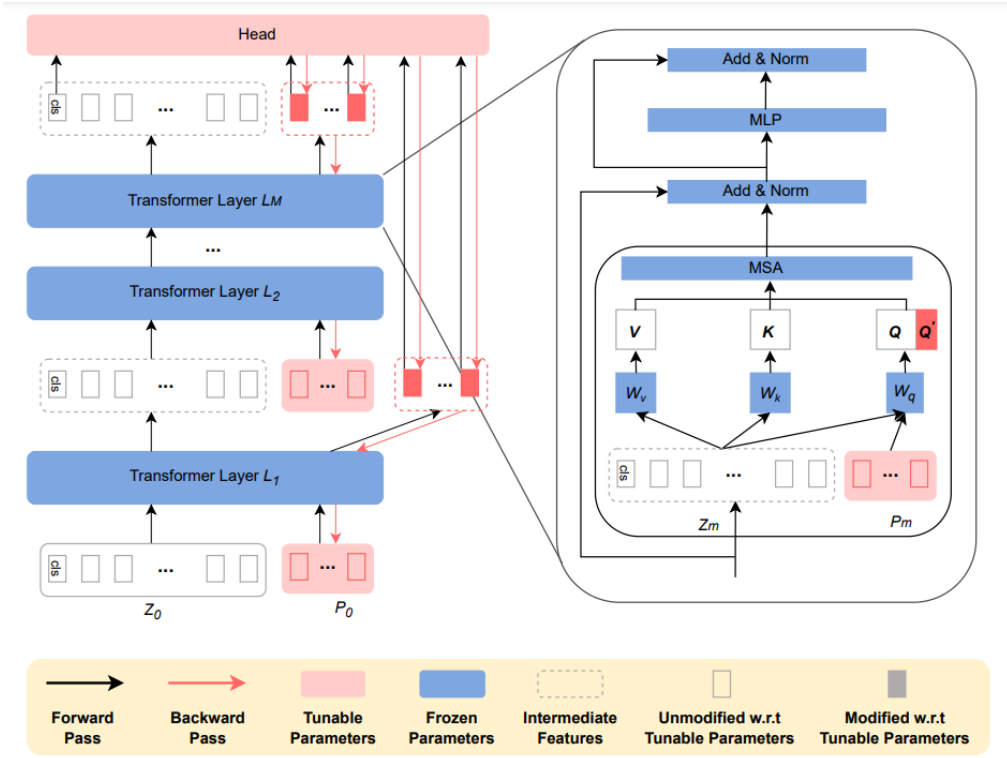
$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{i:y_i=c} \boldsymbol{x}_i,$$

PTM as Better Representations

- Head2Toe selects the most useful features from the *entire* network and trains a linear head on top [Evci et al., ICML'22].



- Visual Query Tuning (VQT) learns to select rather than adapt intermediate features [Tu et al., CVPR'23].



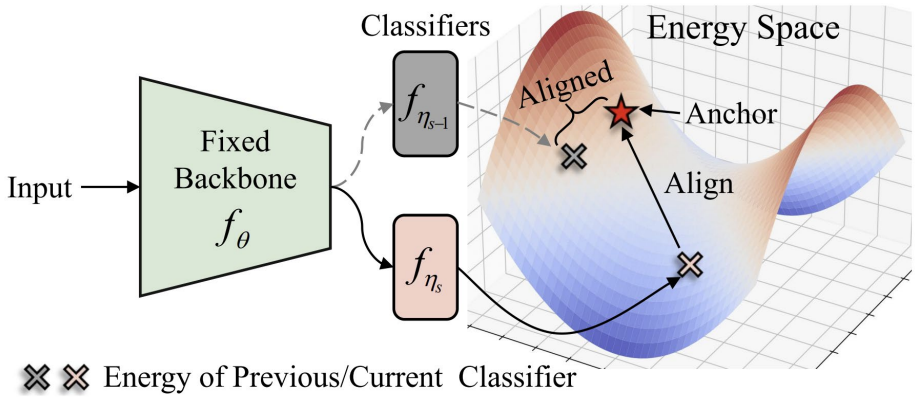
PTM for Prediction Aggregation

- Logit ensemble and probability ensemble (based on activations).

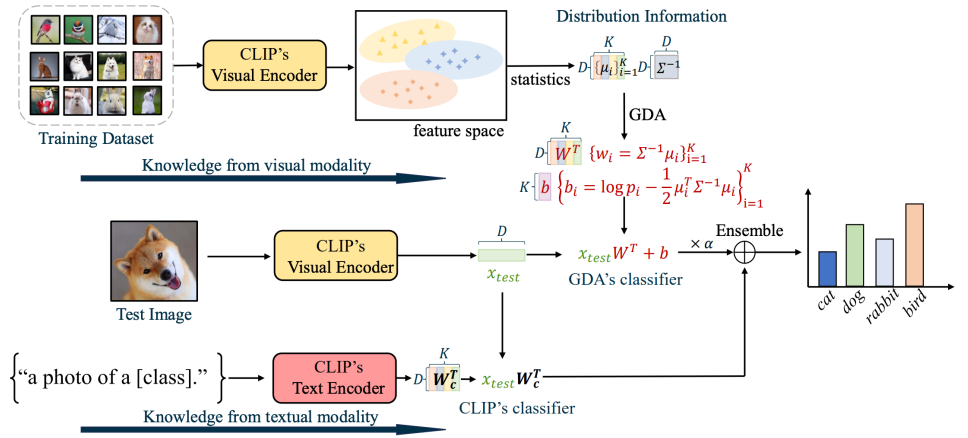
$$\mathrm{Agg}\left(f_\theta(\boldsymbol{x}_i), g_\Theta(\boldsymbol{x}_i)\right)$$

- Note: need to consider the calibration issue.

Create a set of classifiers individually based on the same PTM. In inference, vote these classifier heads by adopting a set of temperatures [Wang et al., AAAI'23].



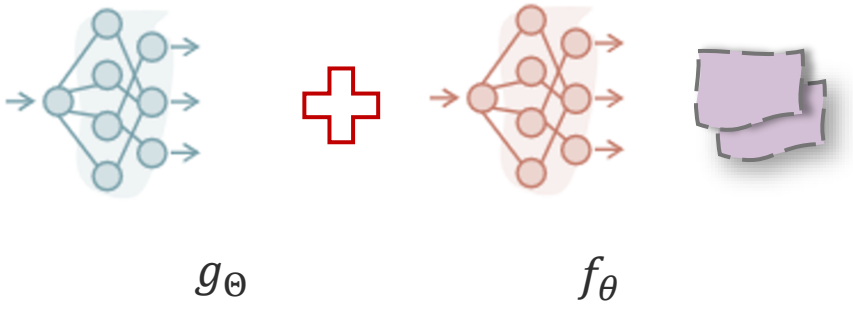
Apply Gaussian Discriminant Analysis together with the zero-shot classifier to the downstream classification of CLIP [Wang et al., ICLR'24].



Model Merge with Weight Average

- Average the weight of two homogeneous models.
- The size of the model does not increase in the inference stage.

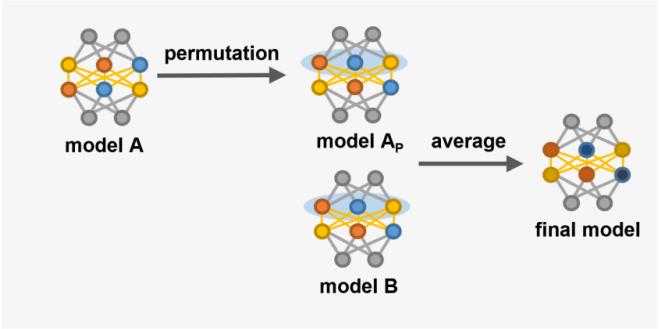
Weighted average the weights $\{\Theta_k\}$ directly, without training process.

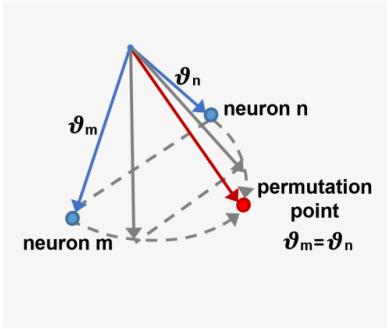


g_Θ f_θ

Could be extended to model reuse from multiple PTMs.

Due to the permutation symmetry of learned weights, an additional alignment should be made [Singh and Jaggi et al., NIPS'20] [Li et al., CORR'23].



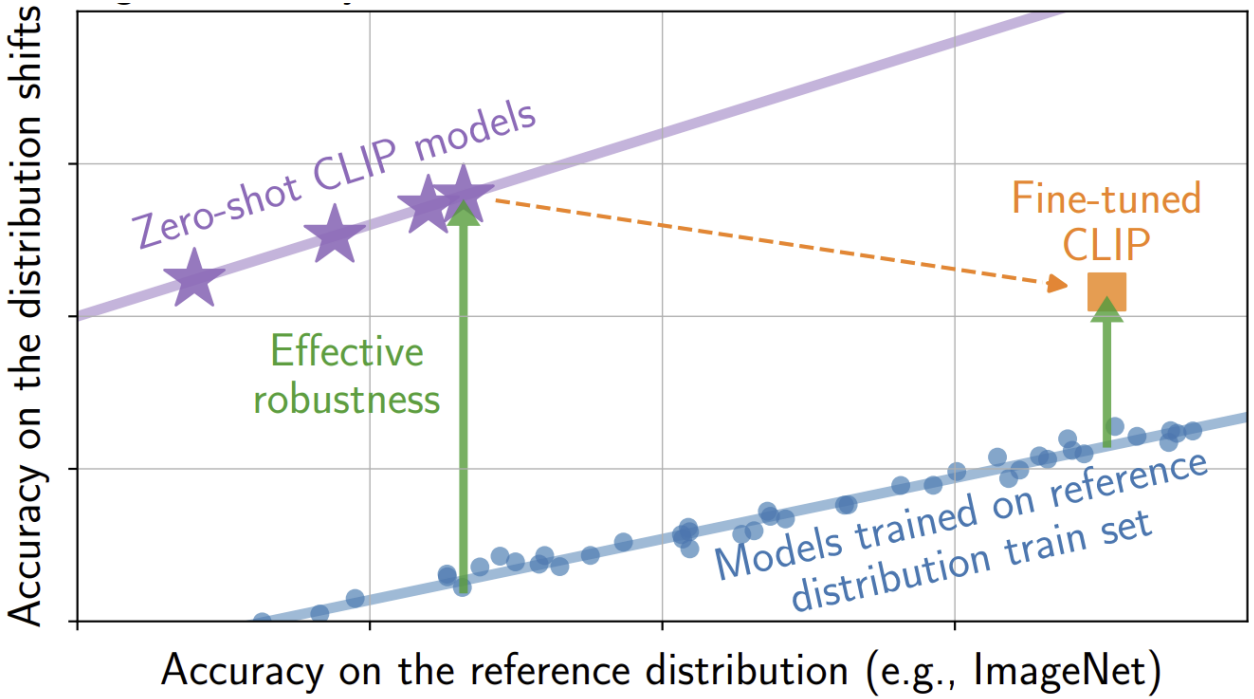


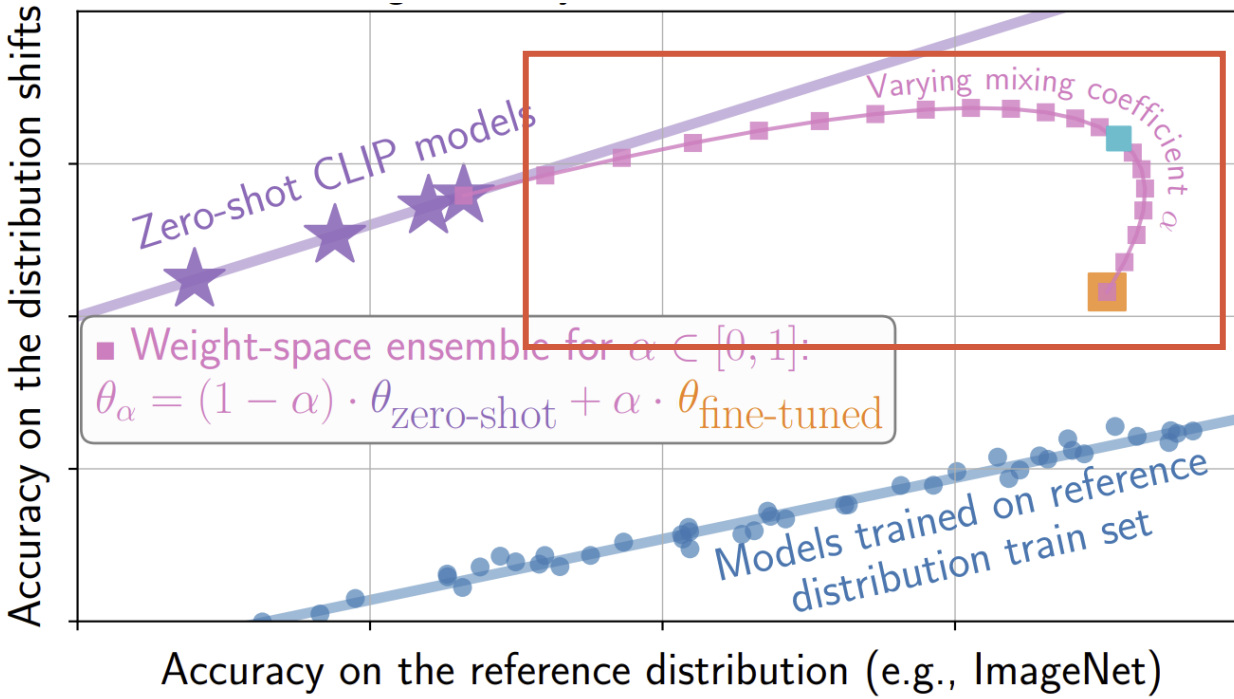
Model Merge Helps Optimization

- WiSE-FT [Wortsman et al., CVPR'22]: weight interpolation after standard fine-tune.
  - Θ is the initialization, θ is the fine-tuned weights on $\mathcal{T}$, then we set

$$\tilde{\theta} = (1-\alpha)\Theta + \alpha\theta$$

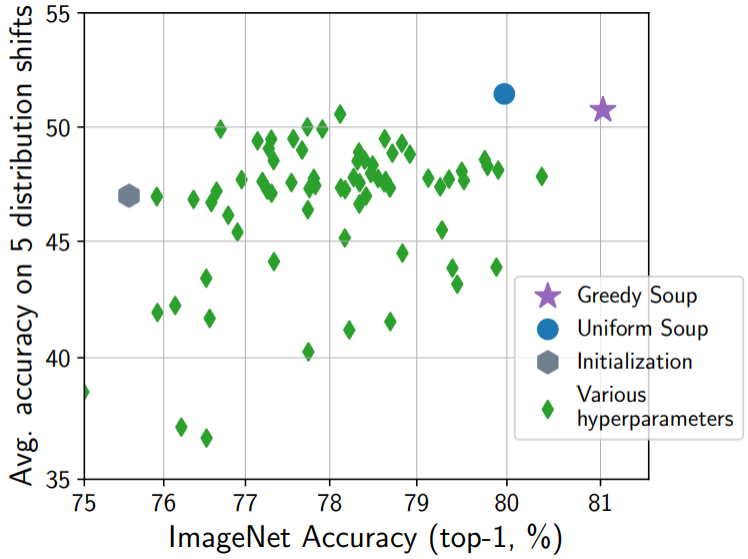
*Keep the **zero-shot** ability of a PTM after fine-tuned under distribution shift.*

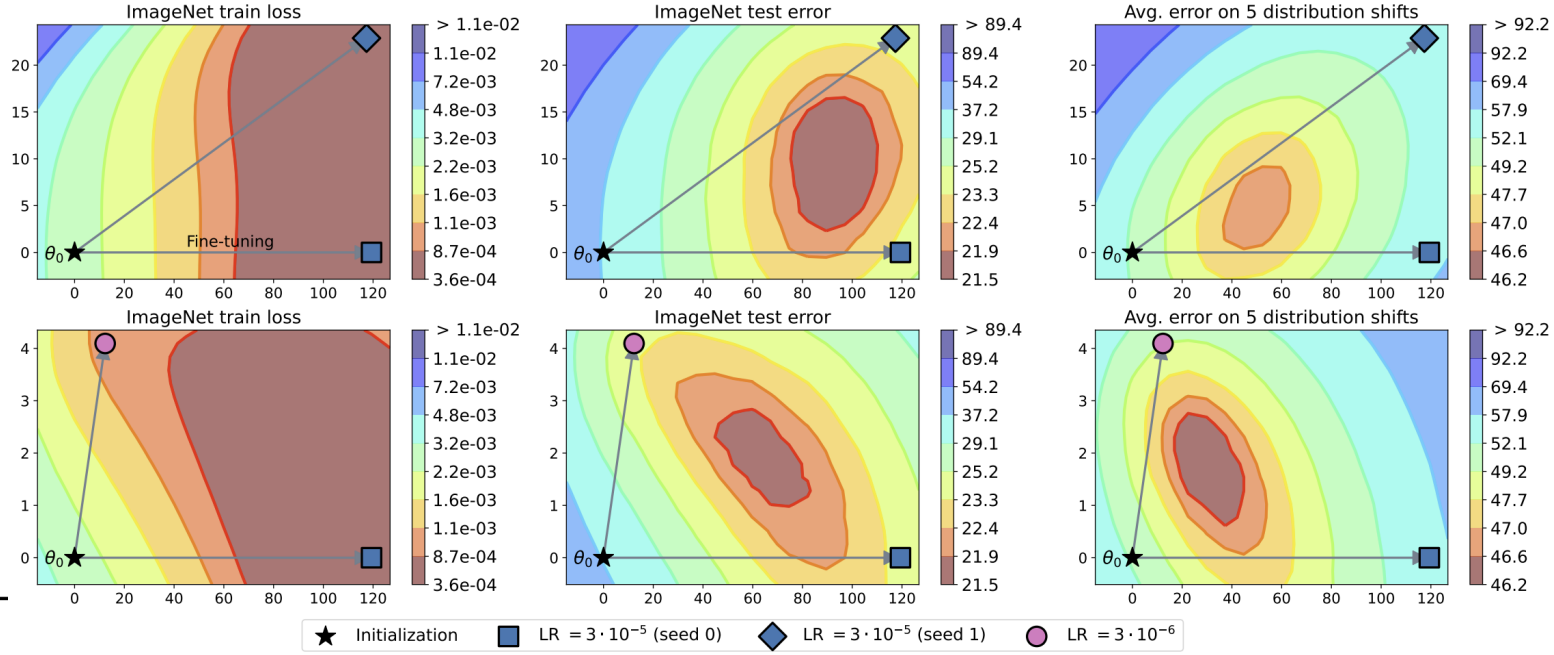




Model Merge Helps Optimization

- Model Soup [Wortsman et al., ICML'22] introduced a technique that involves averaging the weights of ***multiple*** models fine-tuned from the PTM.





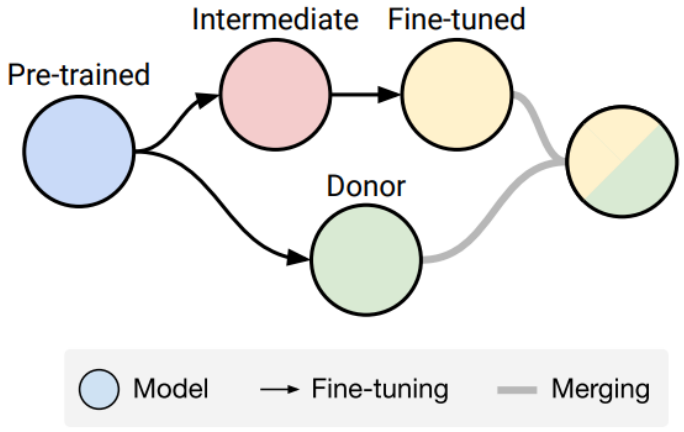
	Method	Cost
Best on val. set	$f(x, \arg\max_i \text{ValAcc}(\theta_i))$	$\mathcal{O}(1)$
Ensemble	$\frac{1}{k}\sum_{i=1}^{k} f(x, \theta_i)$	$\mathcal{O}(k)$
Uniform soup	$f\left(x, \frac{1}{k}\sum_{i=1}^{k} \theta_i\right)$	$\mathcal{O}(1)$
Greedy soup	Recipe 1	$\mathcal{O}(1)$

The solution with the highest accuracy is often ***lies between*** *fine-tuned models*

Model Merge

- Other strategies to merge PTMs.

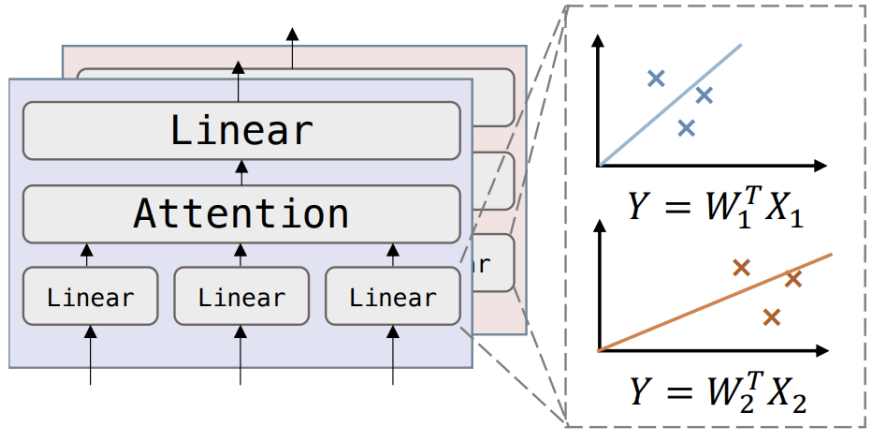
Merging Models with Fisher-Weighted Averaging [Matena et al., NIPS'22].



Fisher

$$W_1 \odot F_1 + W_2 \odot F_2$$

RegMean: minimize the prediction differences between the merged model and the individual models, with closed solution [Jin et al., ICLR'23].

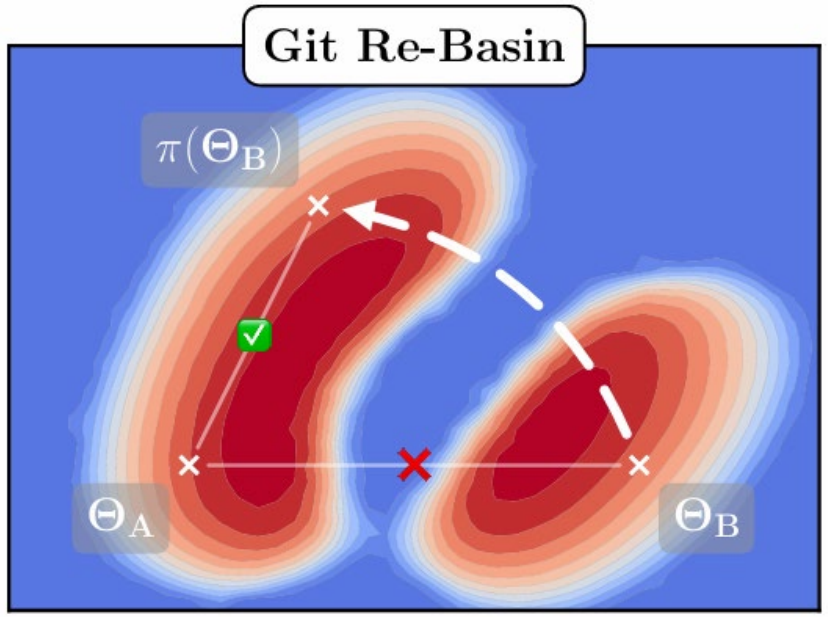


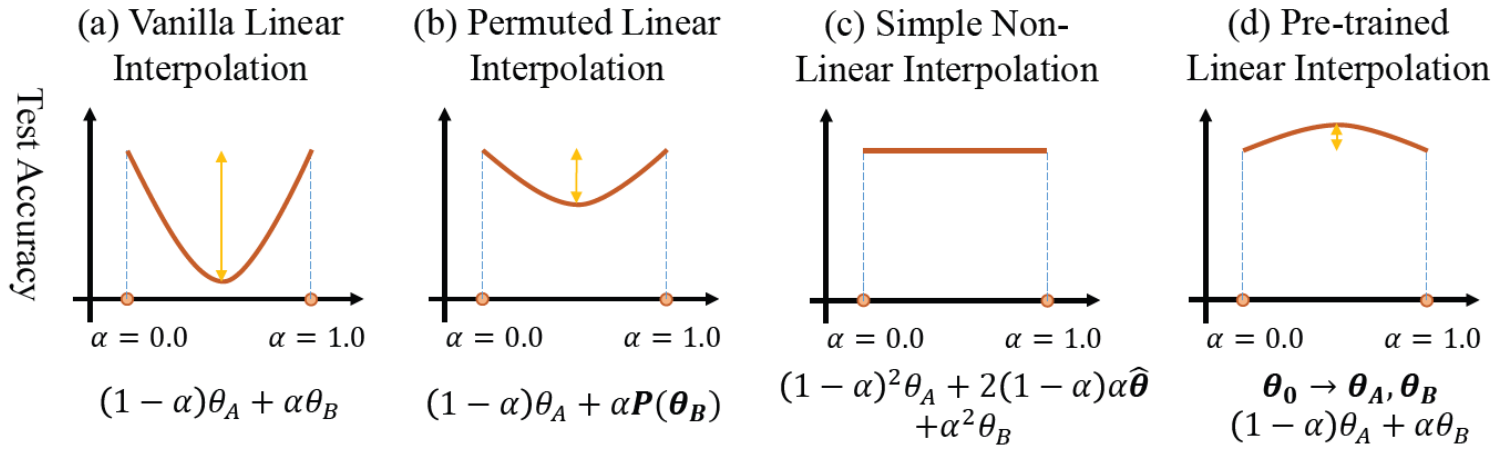
RegMean

$$\left[X_1^T X_1 + X_2^T X_2\right]^{-1}\left[X_1^T X_1 W_1 + X_2^T X_2 W_2\right]$$

Why Model Merging Helps?

- Linear Mode Connectivity (LMC): the minima obtained by gradient-based optimizer are not walled off in isolated valleys, and a direct linear path connecting two such independently trained networks usually always ***leaves a low-loss manifold*** [Garipov et al., NeurIPS'18].

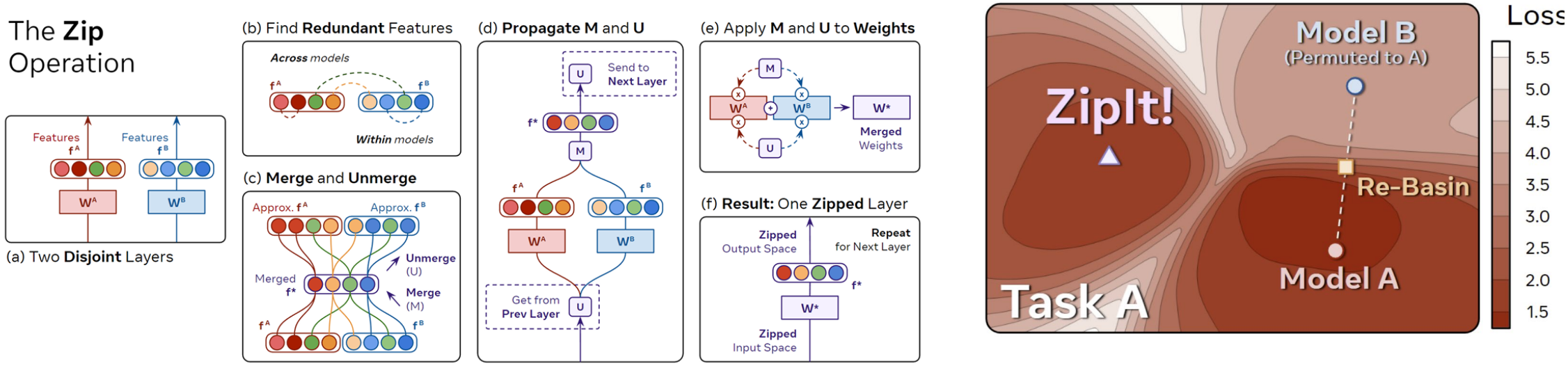




Model Fusion: Training-Free Ensemble

- Learn a mapping matrix to rectify the weights of different layers to address the model heterogeneity, e.g., OT-Fusion [Singh and Jaggi et al., NIPS'20] [Ye et al., TPAMI'21].

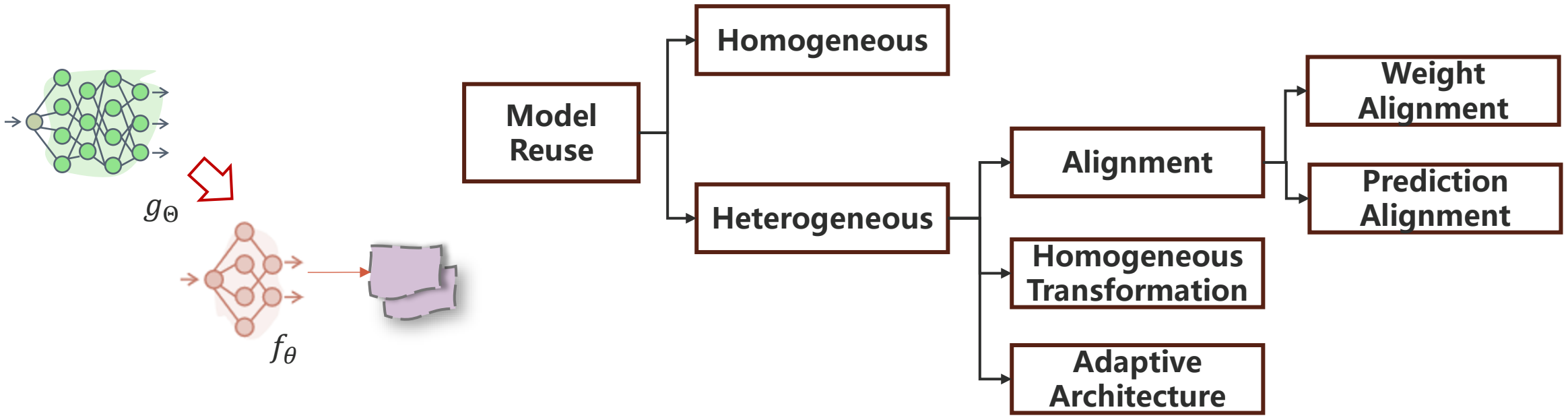
- "zip": merge features within each model. Then partially zip the models up until a specified layer, naturally creating a multi-head model [Stoica et al., ICLR'24].



Interpolation between Model A and a permuted Model B lies outside the minima for both tasks. ZipIt! Finds a model that lies in a low loss basin for both.

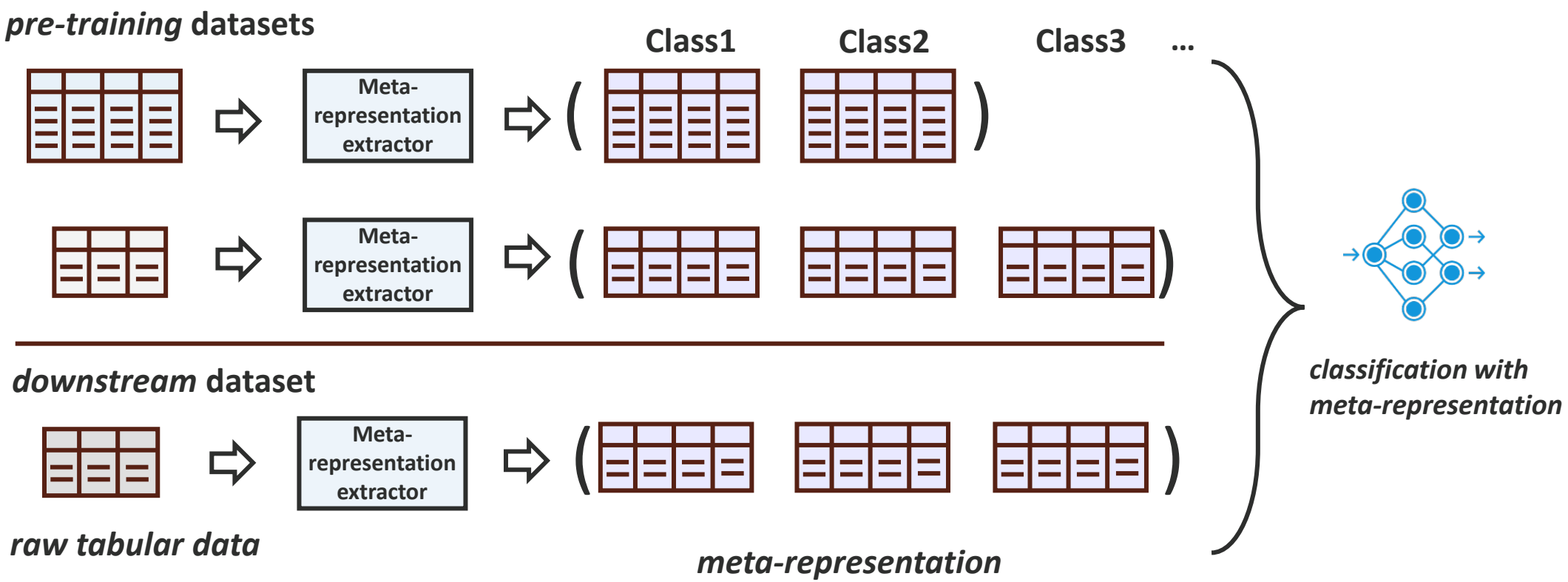
Heterogeneous Model Reuse

- When there exist differences between the pre-trained task and the target task, as well as their architectures, some model reuse approaches should be adapted accordingly.



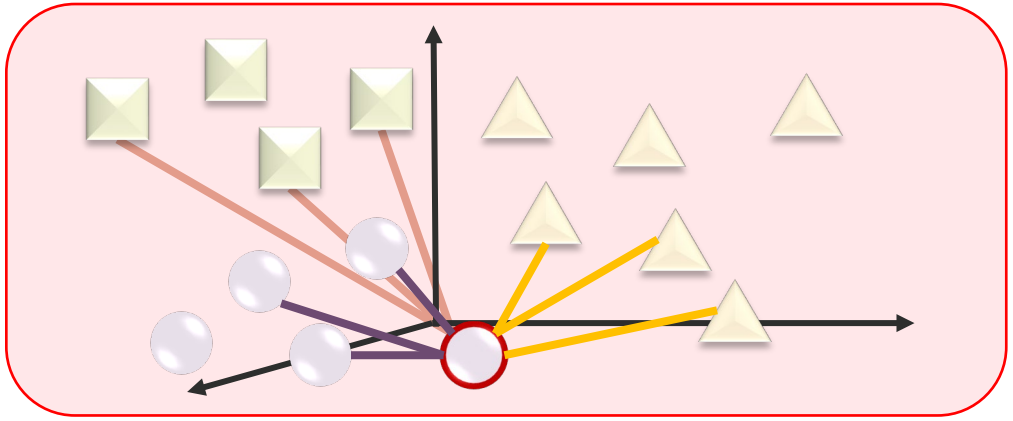
Homogeneous Data Transformation

- Transform all the datasets into the homogeneous form, so that the reduce the discrepancy between their models. For example, random projection on tabular data [Bonet et al., NeurIPSW'23].

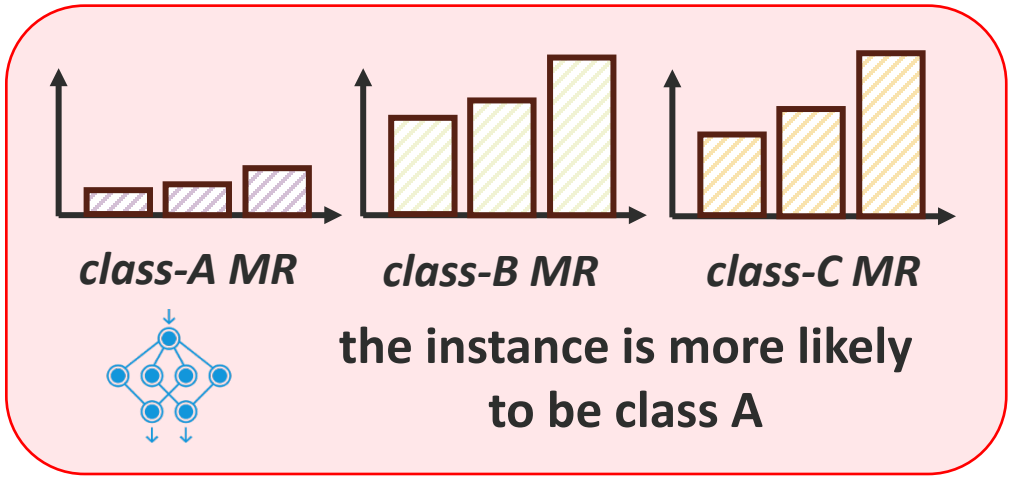


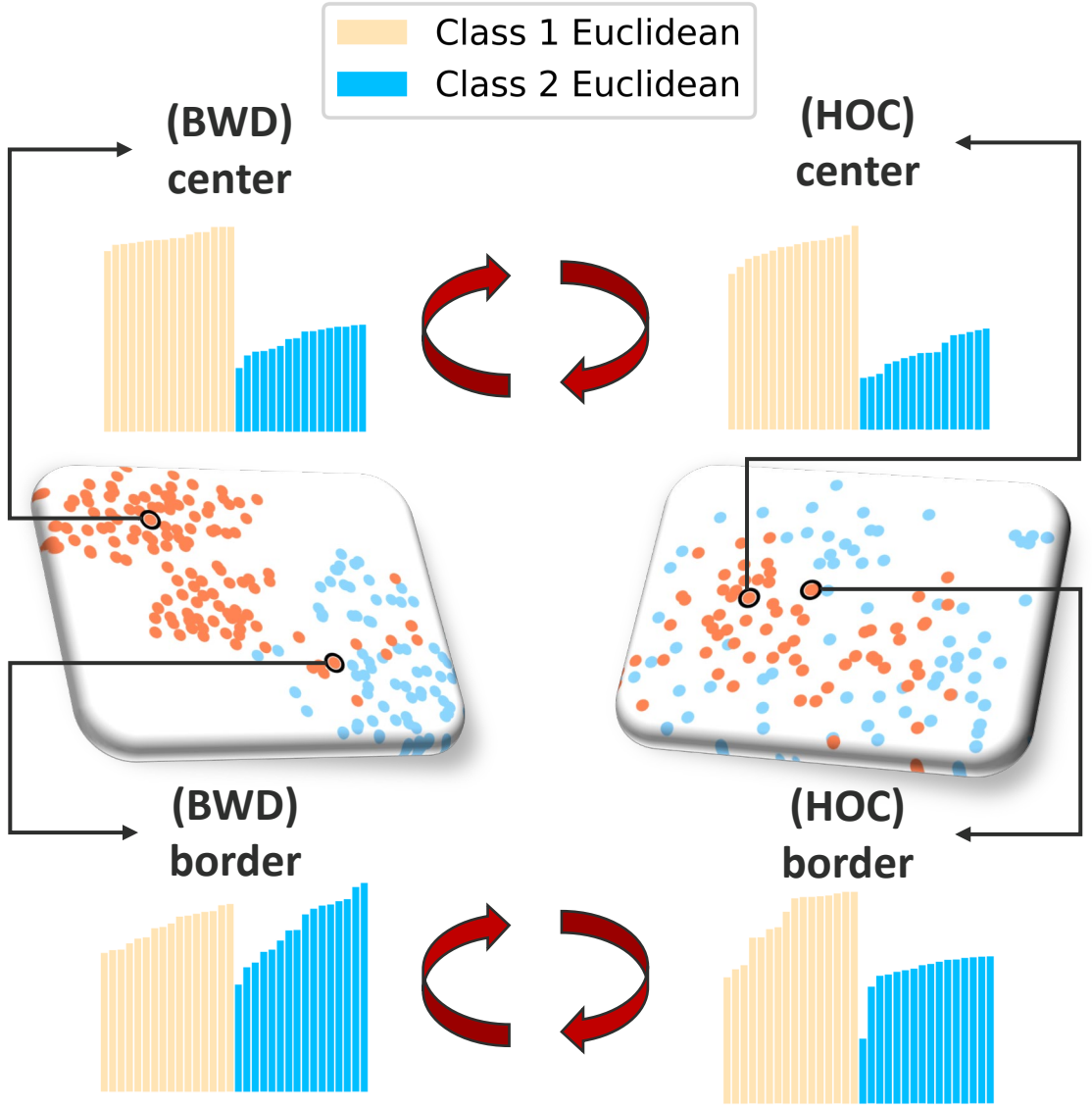
Homogeneous Data Transformation

Extract class-specific prototypes.



Calculate the distance to those prototypes,
Sort and select the K smallest.

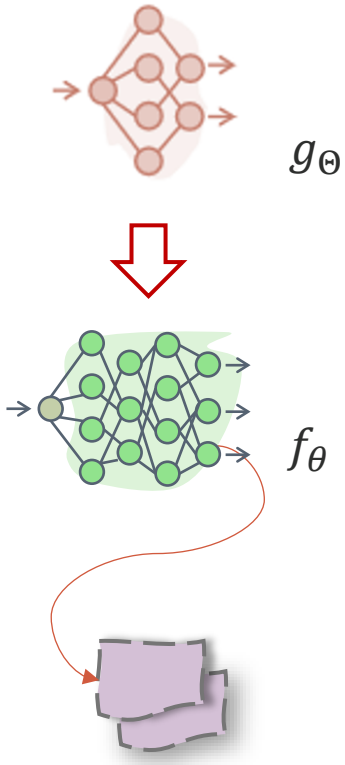




[Ye, Zhou, Zhan, NeurIPSW'23].

Model Growth

- Reuse a smaller PTM for a larger target model.
- Net2Net [Chen et al., ICLR'16] expand the width of neural networks by duplicating neurons. Given the pre-trained weights $\Theta_l \in \mathbb{R}^{d\times d}$, the target weight is $\theta_l \in \mathbb{R}^{D\times D}$, then



$$\theta_l = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{S}_{l-1}^\top \end{bmatrix} (\mathrm{diag}(\boldsymbol{S}_{l-1}\mathbf{1}) +)^{-1} \Theta_l \begin{bmatrix} \boldsymbol{I} & \boldsymbol{S}_l \end{bmatrix},$$

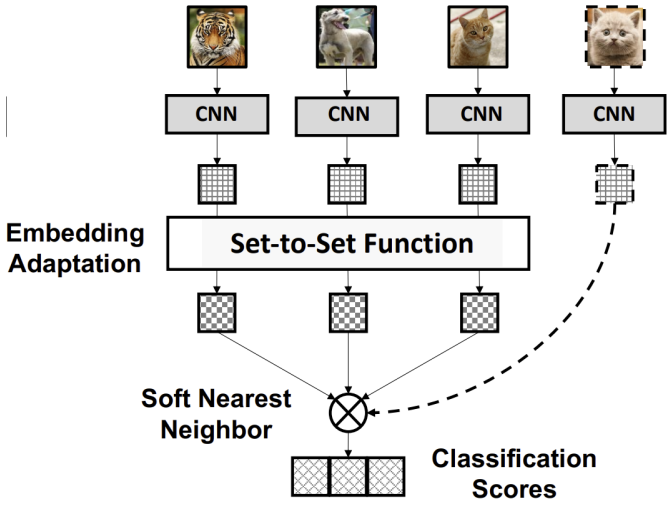
$\boldsymbol{S}_l \in \{0,1\}^{d\times(D-d)}$ is a random selection matrix that indicates the column indices to be duplicated at the l-th layer.

- bert2BERT [Chen et al., ACL'22].
- Learning to Grow (LiGO) [Wang et al., ICLR'23] learns a linear mapping between two parameter spaces before expanding the model size.
- Mango [Pan et al., NeurIPS'23]: consider the inter- and intra-interactions among the weights of both the pretrained and the target models.

Reuse Adaptive Model

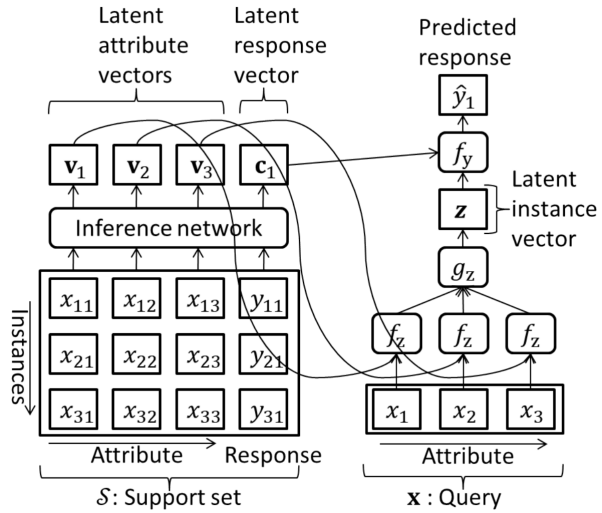
- Given input from heterogeneous tasks, we can either consider a dimension-invariant transformation or learn an adaptive model to fit heterogeneous tasks.
- Meta-learn a transformer and make prediction with its adaptation/in-context-learning ability.

Heterogeneous *classes*:

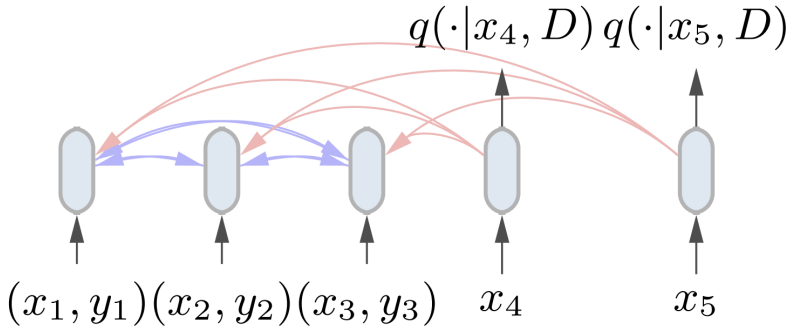


FEAT [Ye et al., CVPR'20]

Heterogeneous *features*:



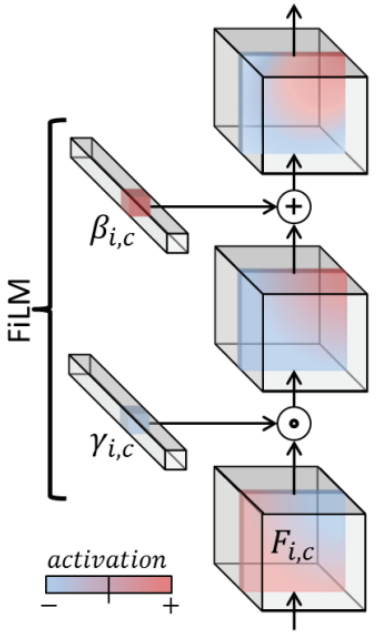
[Iwata, Kumagai, NeurIPS'20]

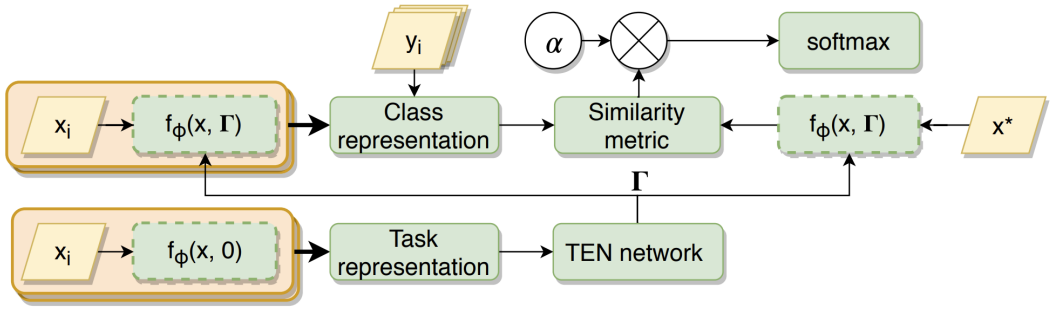


[Hollmann et al., ICLR'23]

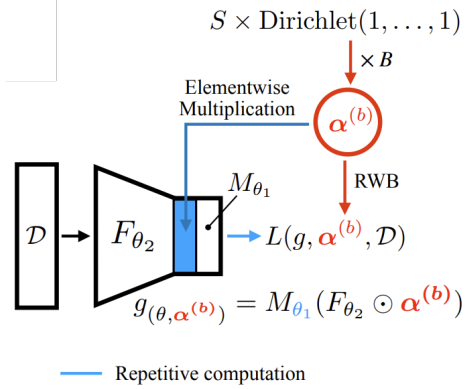
Tuning Adaptive Partial Weights

- Making the tunable part ***adaptive*** given the input.
- FiLM layers influence neural network computation via feature-wise affine transformation based on conditioning information [Perez et al., AAAI'18].
  - $scale\ \gamma = h_1(\boldsymbol{x}_i)$
  - $bias\ \beta = h_2(\boldsymbol{x}_i)$
  - $\mathrm{FILM}(\boldsymbol{F}_{i,c} \mid \gamma_{i,c}, \beta_{i,c}) = \gamma_{i,c}\,\boldsymbol{F}_{i,c} + \beta_{i,c}$





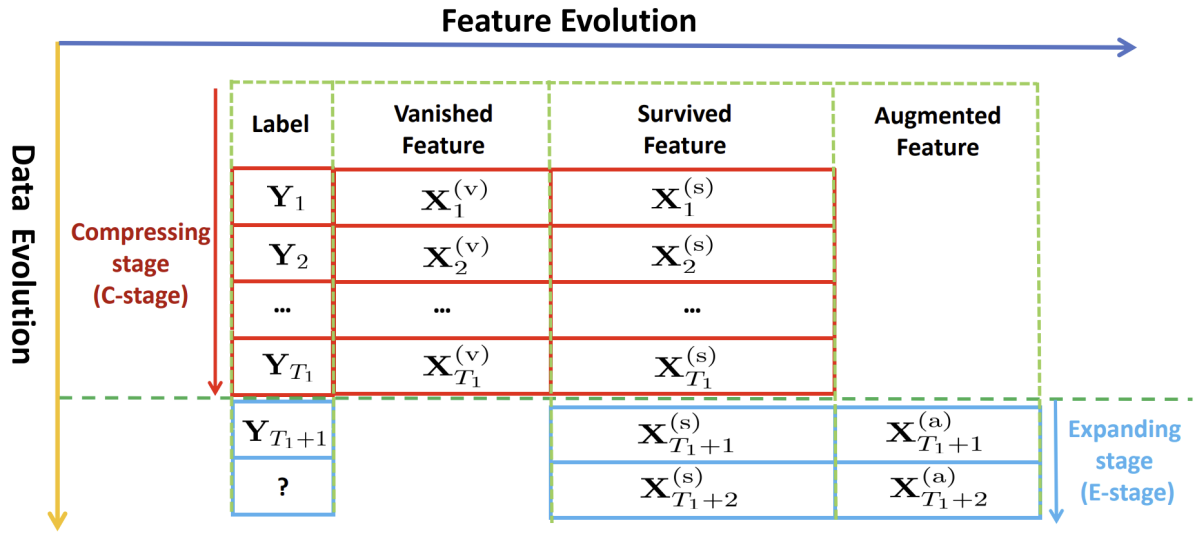
In few-shot learning: make the model adaptive given the support set [Oreshkin, Rodriguez, Lacoste, NeurIPS'18].

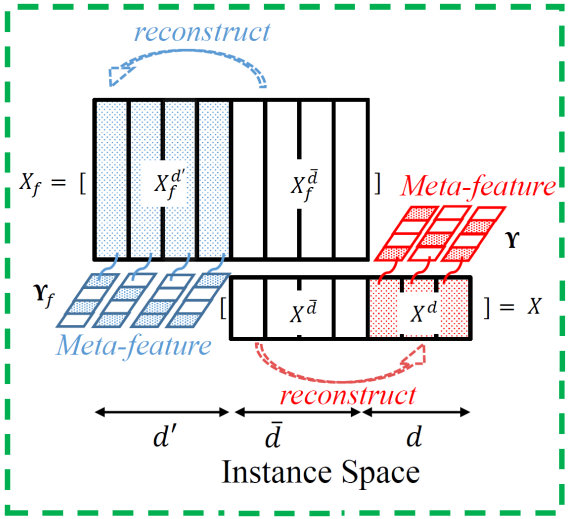


In neural ensemble, tune multiple models efficiently [Shin et al., NeurIPS'21].

Regularization + Semantic Mapping

- When dealing with heterogeneous weight spaces due to the change of ***features***, additional mapping and attention layers are applied to facilitate alignment.





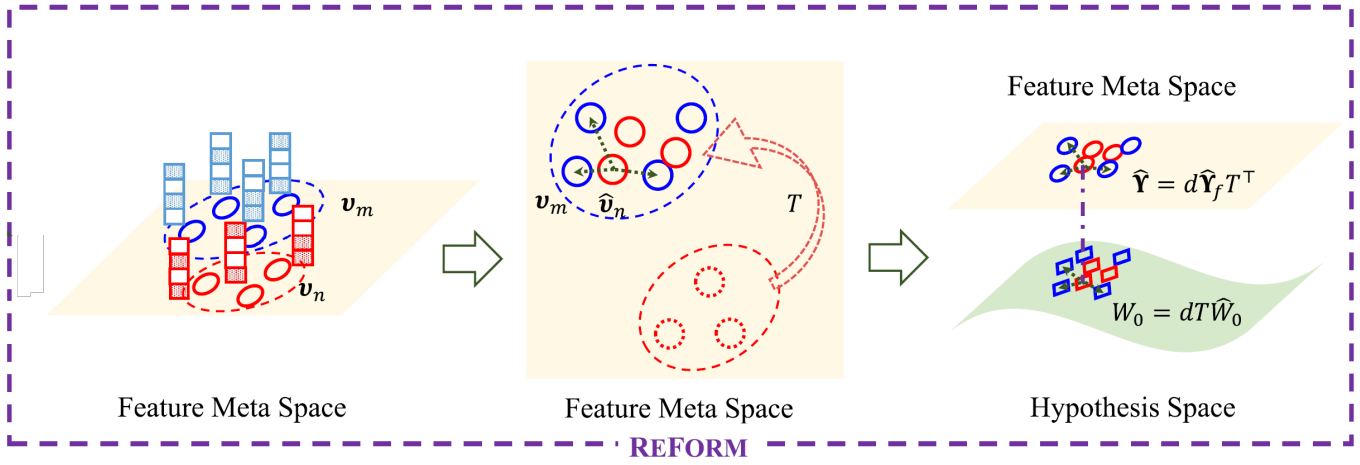
$$\Omega(\theta, \Theta) = \lambda \|\mathcal{T}(\theta) - \Theta\|_2^2$$

OPID [Hou, Zhou, TPAMI'18]: ***compress*** important information of vanished features into functions of survived features, and then ***expand*** to include the augmented features.

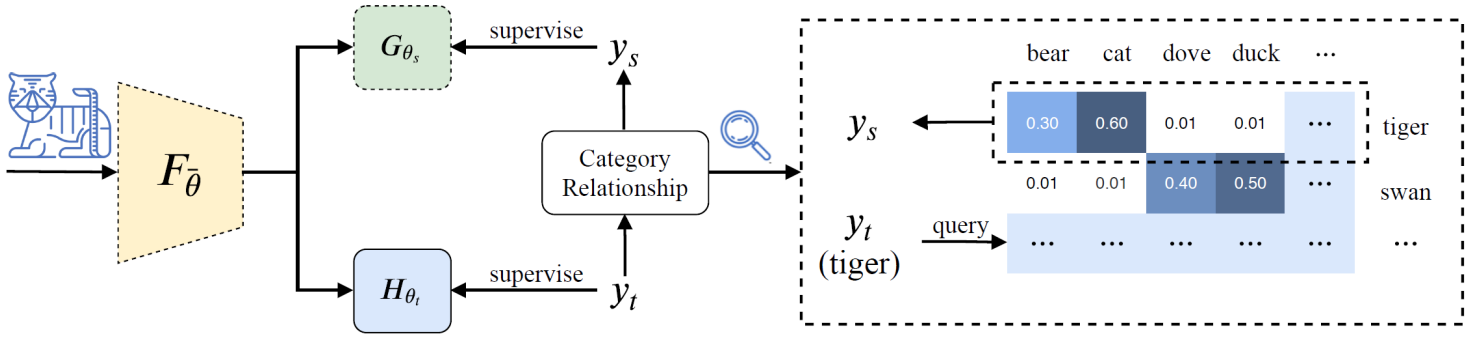
ReForm [Ye et al., TPAMI'21]: constructs ***meta-representation for features***, which depicts the relationship between features in different stages. Then, an OT map learned in the meta-space could be applied to the model space.

Semantic Mapping for Class Set Discrepancy

- When dealing with heterogeneous weight spaces due to the change of ***classes***, the semantic mapping could be applied to their predictions.



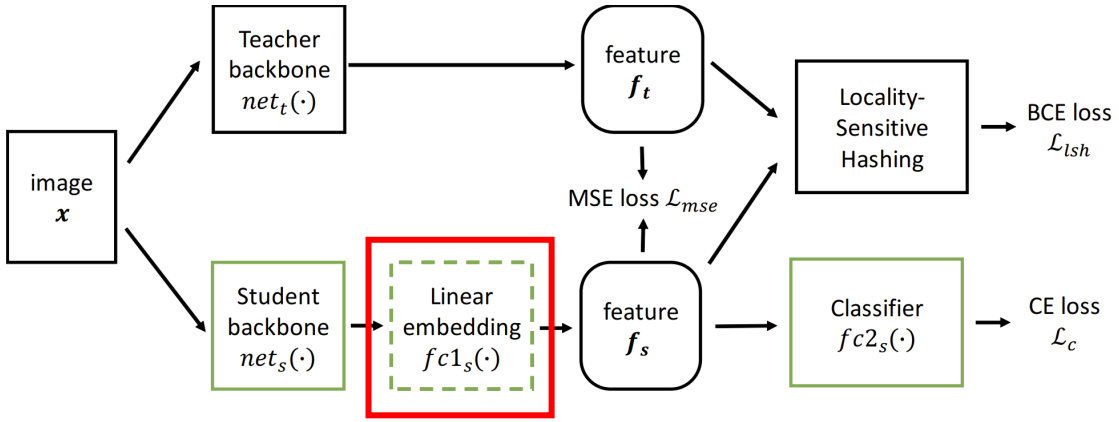
ReForm [Ye et al., TPAMI'21]: the meta-representation is constructed via the class center based on the feature extracted by PTM. Then apply the biased regularization.



Co-Tuning [You et al., NeurIPS'20]: discover the semantic mapping via the predictions of PTMs on the instances from the target task (among the source class set) and the target class labels of those instances.

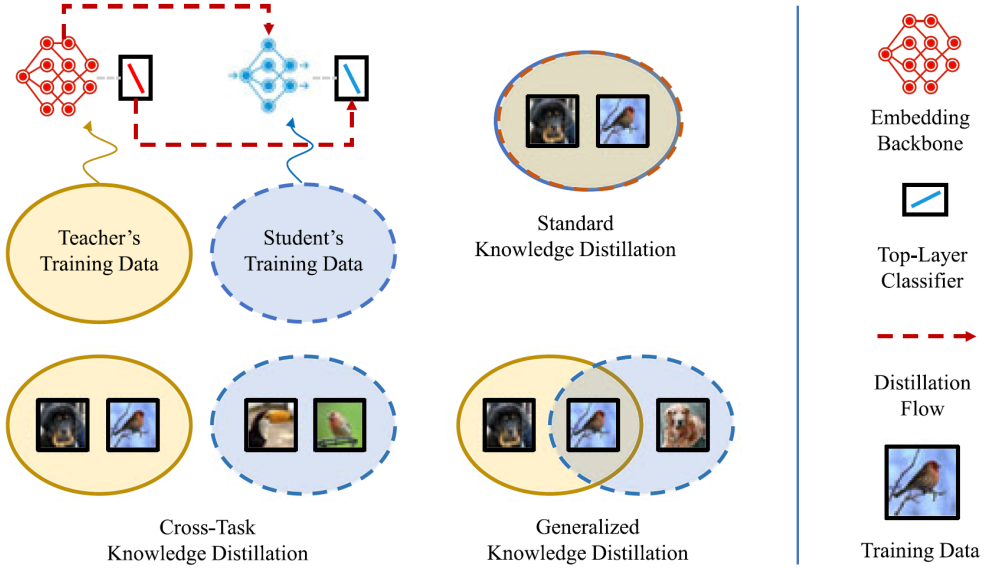
Generalized Knowledge Matching

- Apply a feature matching component to bridge the feature gap between target model and PTM [Wang, Ge, Wu, TPAMI'22].



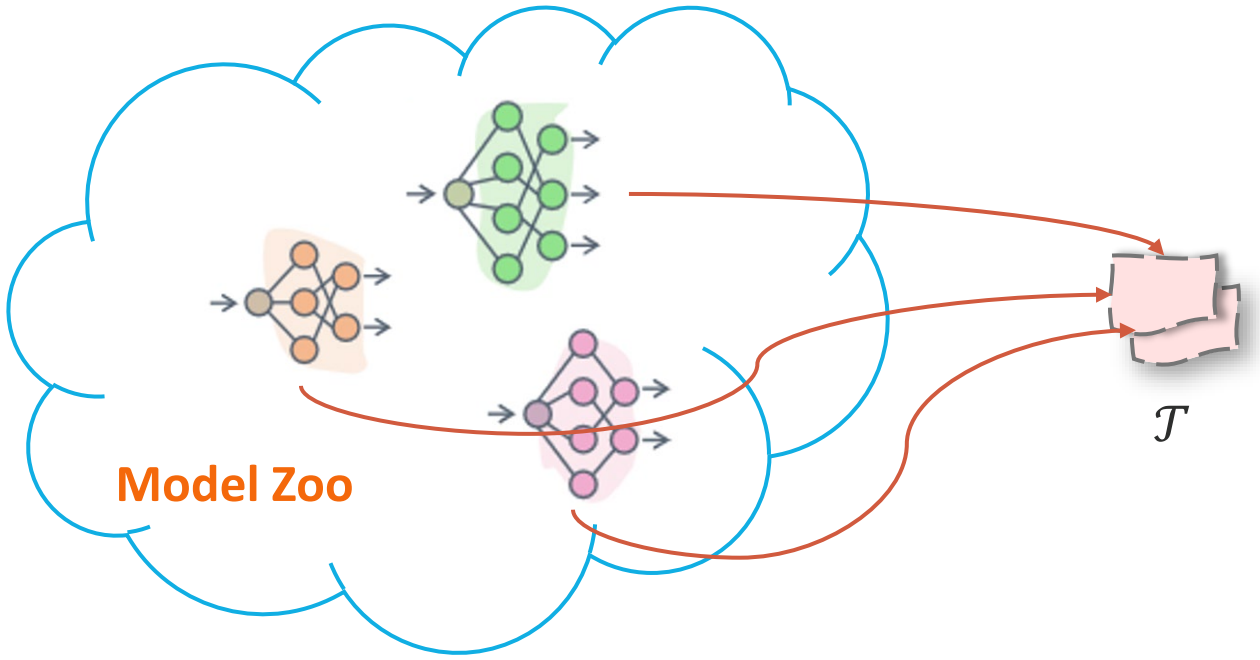
The linear embedding layer also rotate the feature space and align the features of two models.

- *Generalized* KD, where the student could have the same, different, or partially overlapped classes w.r.t. the teacher [Ye, Lu, Zhan, TPAMI'23].



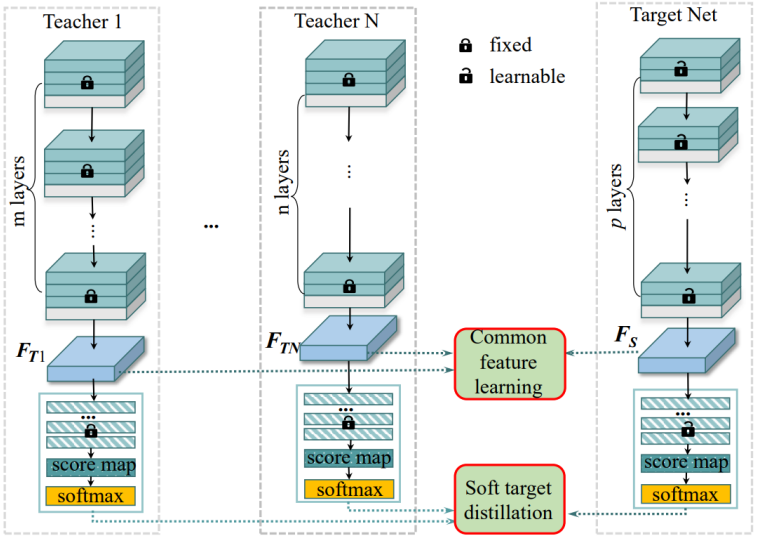
We can take advantage of relationship-based methods which do not depend on the feature dimension and magnitude.

Reuse Multiple PTMs



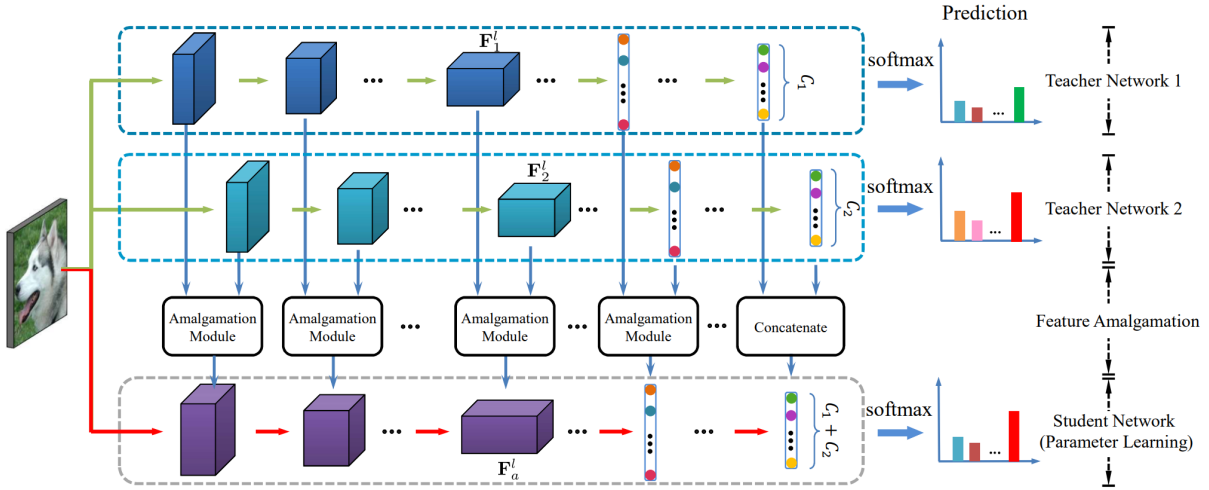
- The importance of different PTMs are diverse.
- Transferability metric to weight different PTMs.
- Select one of them to keep the efficiency and ability.

Knowledge Amalgamation



[Luo et al., IJCAI'19]

Features of all teachers are transformed into a **common space** and the student is enforced to imitate them all so as to amalgamate the intact knowledge.



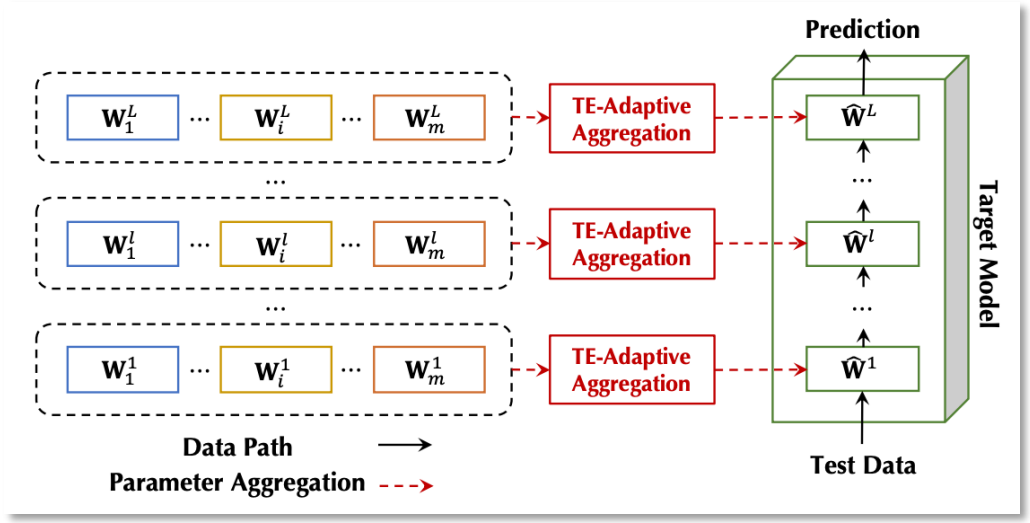
[Shen et al., AAAI'19]

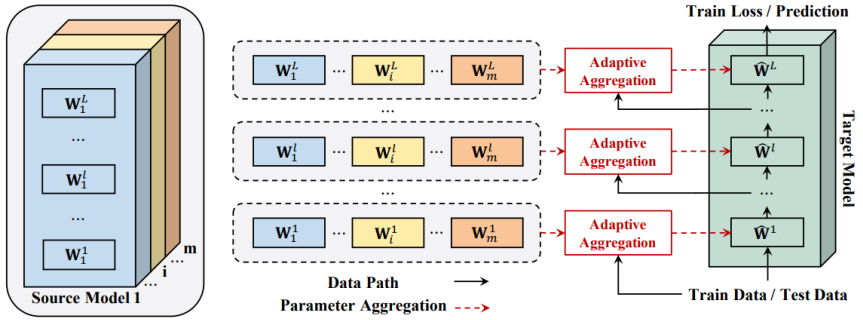
A two-step strategy: learn the compact *feature representations* from teachers and then the network parameters in a layer-wise manner so as to build the student model.

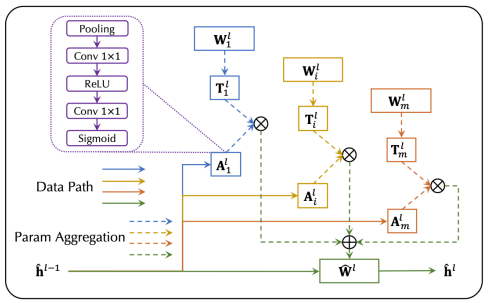
Zoo-Tuning

- Aggregate parameters from heterogeneous PTMs [Shu et al., ICML'21].

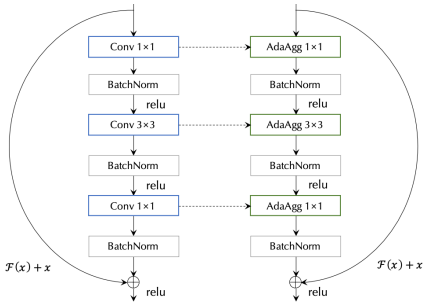
- With the learnable channel alignment layer and adaptive aggregation layer, Zoo-Tuning **adaptively aggregates channel aligned pretrained parameters to derive the target model**.







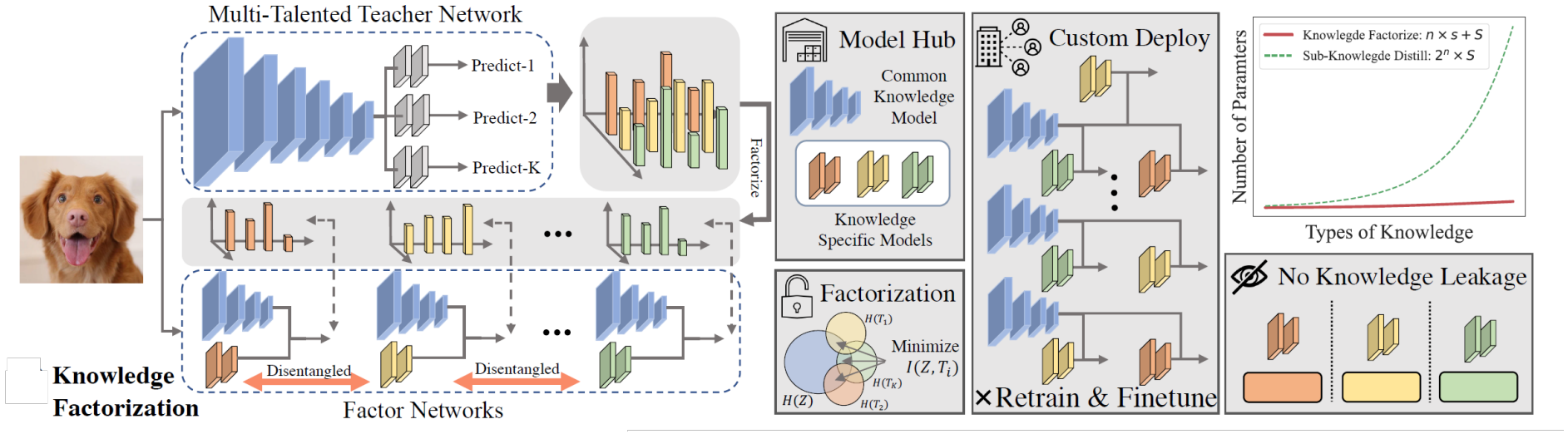
(a) Adaptive Aggregation (AdaAgg) Layer



(b) Integrating AdaAgg Layers in a Residual Block

Knowledge Factorization

- Factorize task-agnostic (shared across multiple tasks) and task-specific (for a certain task) knowledge from the teachers [Yang, Ye, Wang, ECCV'22].

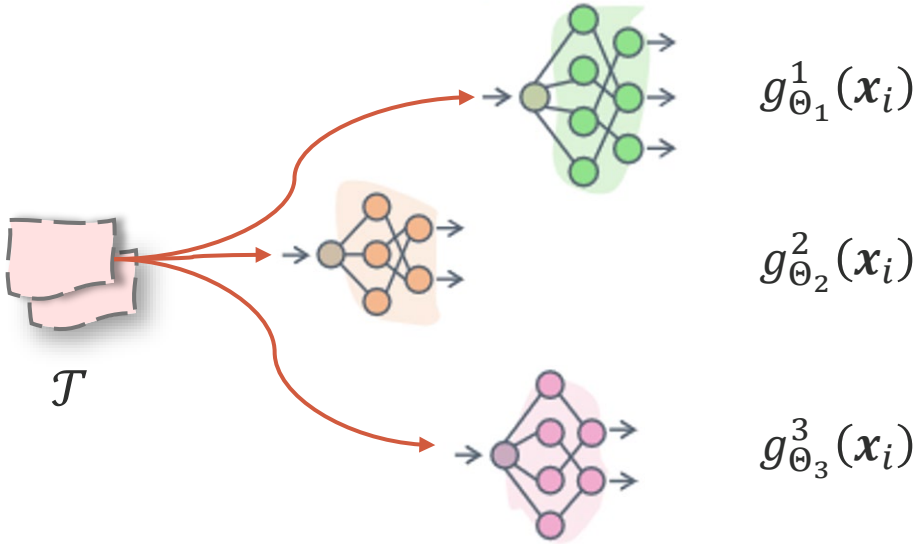


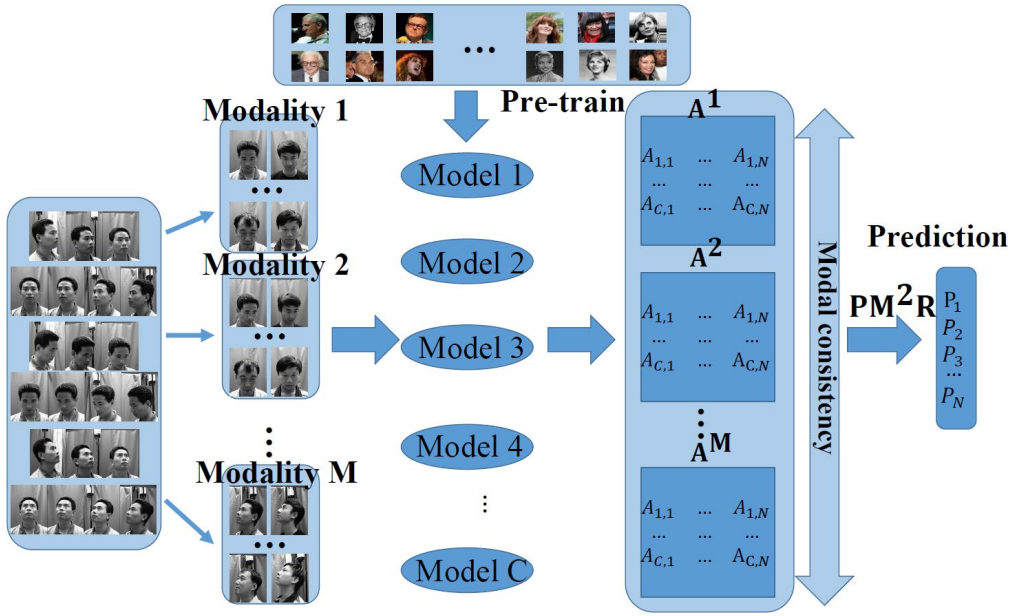
Given a pretrained teacher, KF **decomposes it into several factor networks**, each of which masters one specific knowledge factorized from the teacher, while remaining disentangled with respect to others.

- ***Structural* Factorization**: decompose the teacher into a set of factor networks. Each factor network comprises a shared common-knowledge network and a task-specific network.
- ***Representation* Factorization** disentangles the shared knowledge and task-level representations into statistically independent components.

Reuse Multiple Models

- Prediction aggregation [Ye et al., CIKM'15], which post-fuses the multiple predictions from various PTMs.

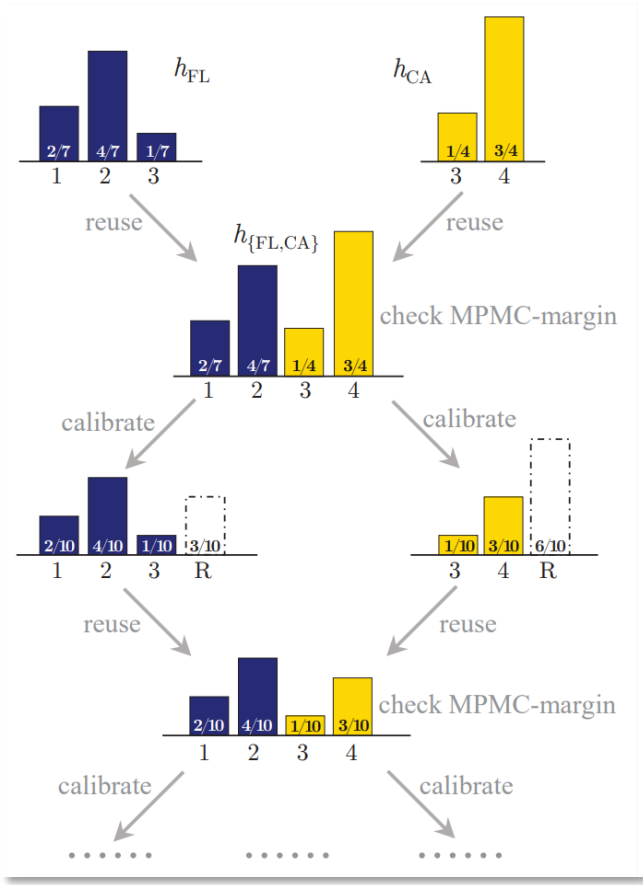




Reuse multiple multi-modal PTMs. Gather predictions into matrices $\{A_1, \ldots, A_M\}$, the final predictions are obtained with the consist weight propagation among different modalities [Yang et al., IJCAI'17].

Learning and Calibration

- Calibrate the prediction of multiple local models in Multi-Party Learning [Wu et al., ICML'19].



Algorithm 1 HMR

input:

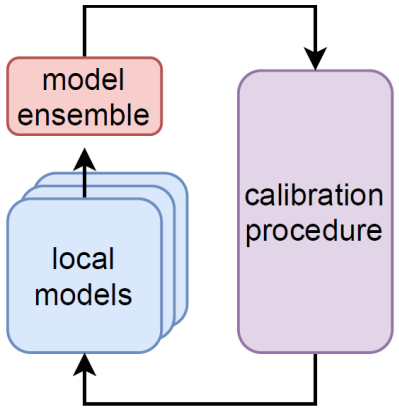
Parties $1, 2, \cdots, n$, each owns a local dataset S_i and a local model h_i. Example communication budget N.

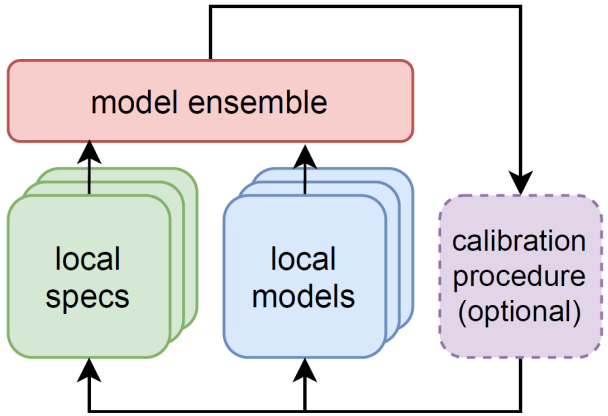
output:

Calibrated local models $h_1, \cdots, h_n$.

procedure:

1: Each party broadcasts its local model to others.
2: Inner iteration counter $T = 0$
3: **while** $T < N$ **do**
4: Sample a party i according to $|S_i| / \sum_{i=1}^{n} |S_i|$.
5: Party i randomly selects an example $(x, y) \in S_i$.
6: Party i computes MPMC-margin $\rho_H(x, y)$ according to (7). Records the party i^+, i^- and maximum incorrect class y^- as in (8).
7: **if** $\rho_H(x, y) \leq 0$ **then**
8: Party i sends (x, y, y^-) to i^+ and i^-.
9: Party i^+ *calibrates* h_{i^+} with (x, y, y^-).
10: Party i^- *calibrates* h_{i^-} with (x, y, y^-).
11: Party i^+ and i^- broadcast their updated model.
12: **if** $i^+ \neq i$ or $i^- \neq i$ **then**
13: $T = T + 1$.
14: **end if**
15: **end if**
16: **end while**





[Tang et al., IJCAI'23] extends the HMR paradigm which utilizes different types of specifications of local datasets in a different way, and aggregates the results.

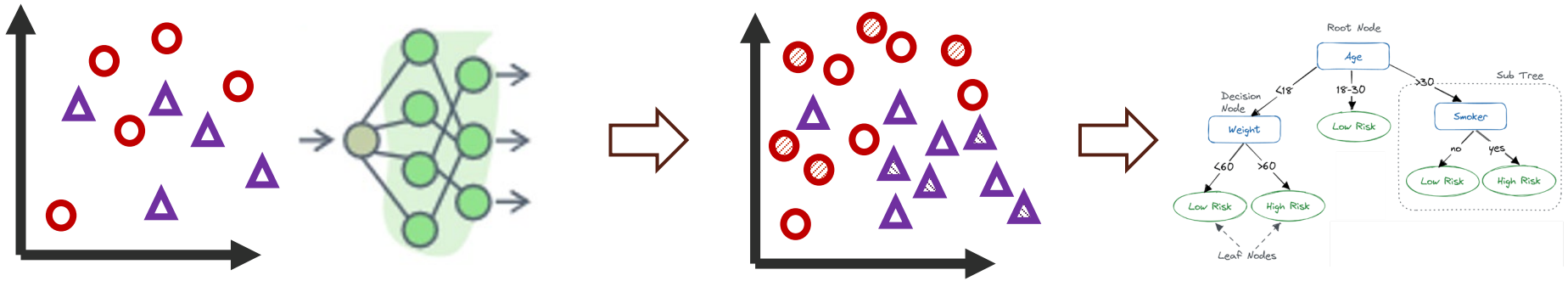
Outline

- *Taxonomy of Model Adaptation*
  - *Adapt PTMs for target data preparation*
  - *Adapt PTMs for target model training*
  - *Adapt PTMs for target model inference*
- ***Other topics in model reuse***
- *Conclusion*

Model Reuse for Interpretability

- Reuse a strong model, which helps improve the generalization ability of a interpretable model such as decision tree [Zhou, Jiang., TKDE'04].

Generate extra training examples.



Learn a strong model such as NN.

Learn an interpretable model over the augmented dataset (twice learning).

Reuse Cross-Metric Models

- Instead of optimizing domain-specific performance measures (e.g., accuracy, AUC, F1-Score, NDCG, MAP) independently, the metric-specific models could be reused from a PTM learned based on the main metric.

  CAPO [Li et al., TPAMI'13]: utilize the relatedness among multiple performance metrics, and implement the classifier in an additive form:

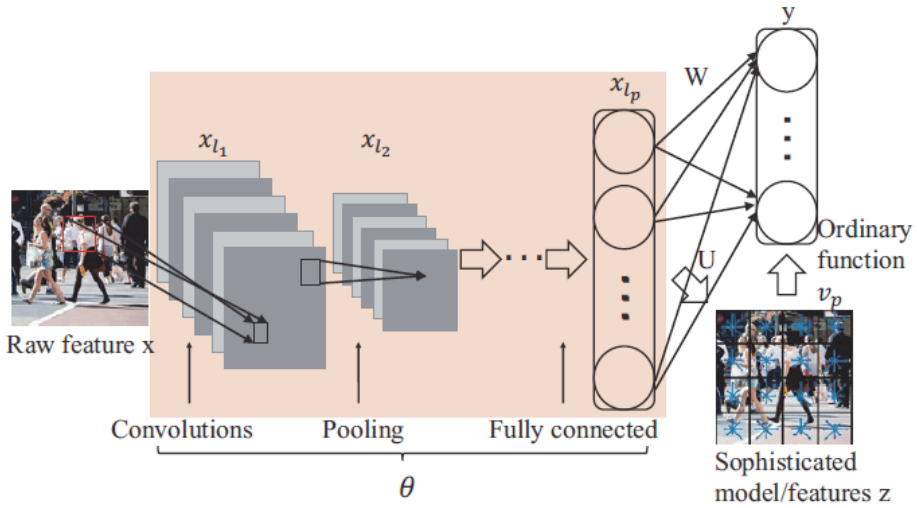
$$\min_{f_\theta} \sum_{(\boldsymbol{x}_i, y_i)\sim\mathcal{T}} \ell(f_\theta(\boldsymbol{x}_i|g_\Theta), y_i) + \Omega(\theta, \Theta)$$

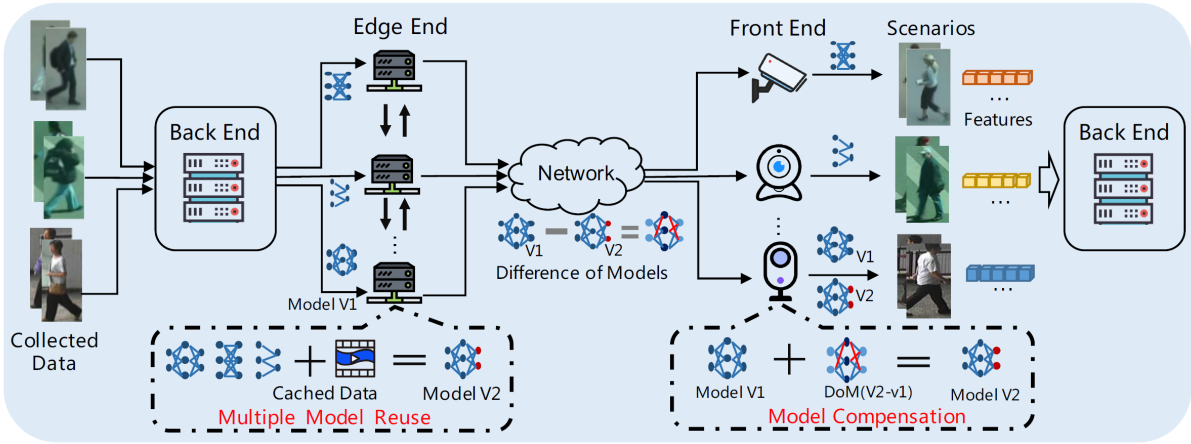
$$f_\theta(\boldsymbol{x}_i|g_\Theta) = \sum_{m=1}^{M} \alpha_m g^M_{\Theta_m}(\boldsymbol{x}_i) + \boldsymbol{w}^\top \Phi(\boldsymbol{x}_i)$$

  - Adapt-Boost [Ding et al., NeurIPS'18] reuse models for different objectives in a boosting manner.

Cross-Modal Reuse

- Reuse multi-modal models, so that the generalization ability of a certain branch (for one modal) could be improved in the inference stage.



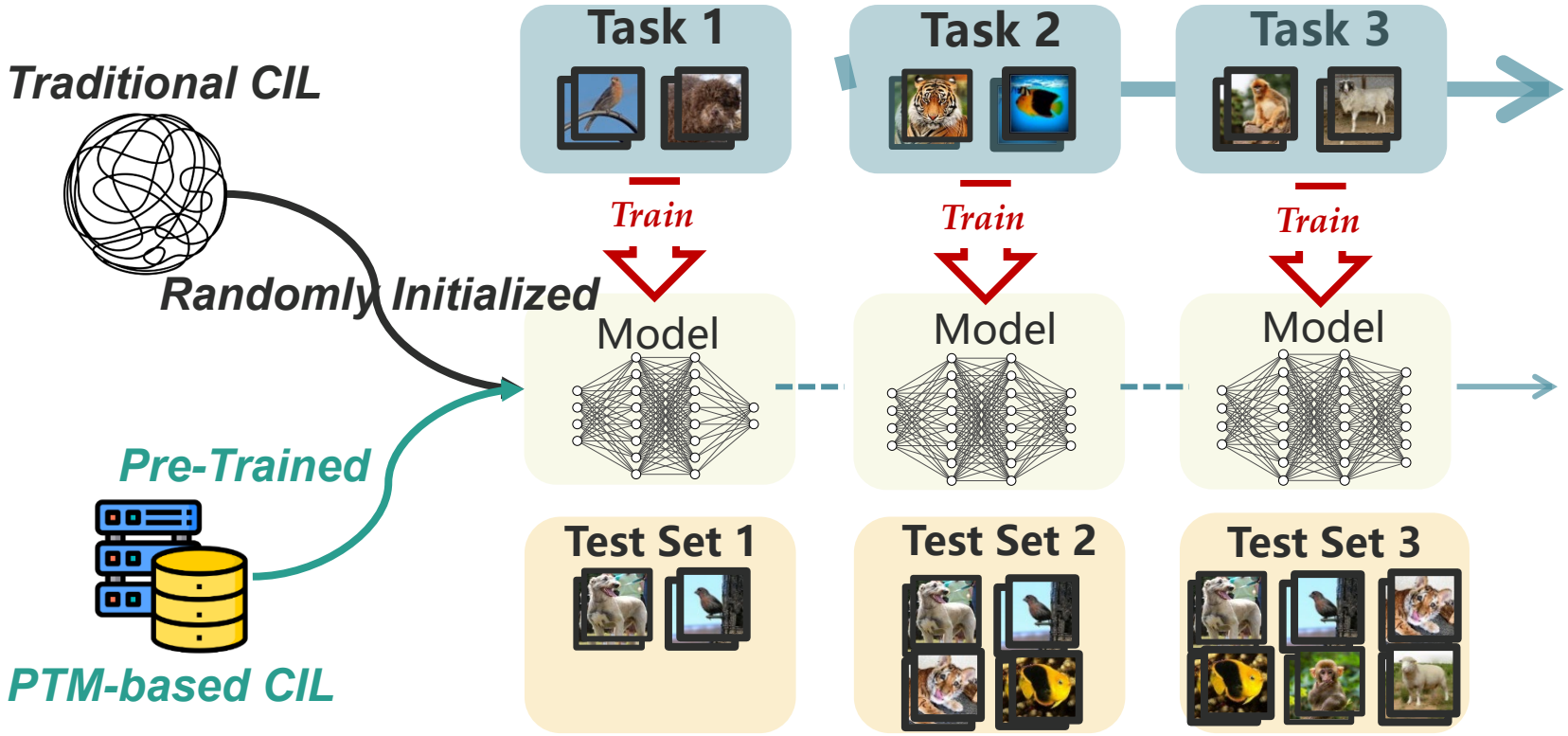


FMR [Yang et al., AAAI'17]: transfer the ability from tabular to image modality. FMR probabilistically "knocks down" specific blocks, which enables the target model to effectively memorize and retain valuable knowledge from the PTM.

Use the hidden layer representation of the source model to train the target depth model, which is superior to the approach using the limited data in target domain. In detail, the target hidden layer representation is improved by using it to reconstruct the source hidden-layer representations [Luo et al., ICME'19].

Example: Class-Incremental Learning

- Given a sequence of training tasks containing different classes, class-incremental learning aims to build a unified classifier for all seen classes



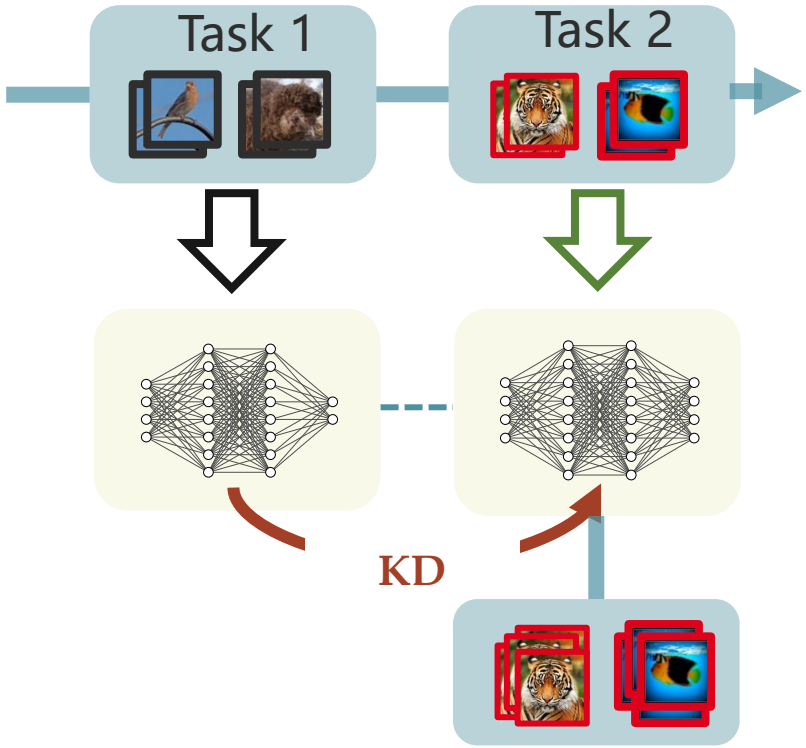
- In each training stage, the algorithms needs to ***reuse the current model and new dataset*** to incorporate new knowledge

Example: Class-Incremental Learning

- When training from scratch, the previous model can be utilized to provide supervision signals to prevent forgetting

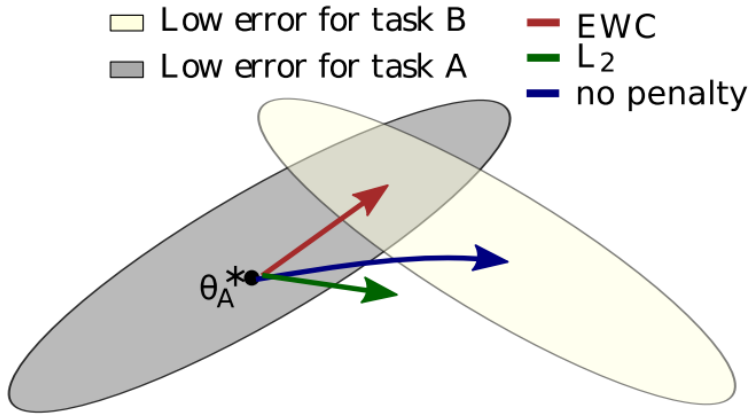
Training level: Distillation

Distilling logits to align the knowledge between old and new model [Li et al., ECCV'16].



Training level: Regularization

The importance of different parameters shall vary for different tasks. EWC builds regularization term by forcing important parameters to stay unchanged [Kirkpatrick et al., PNAS'17].



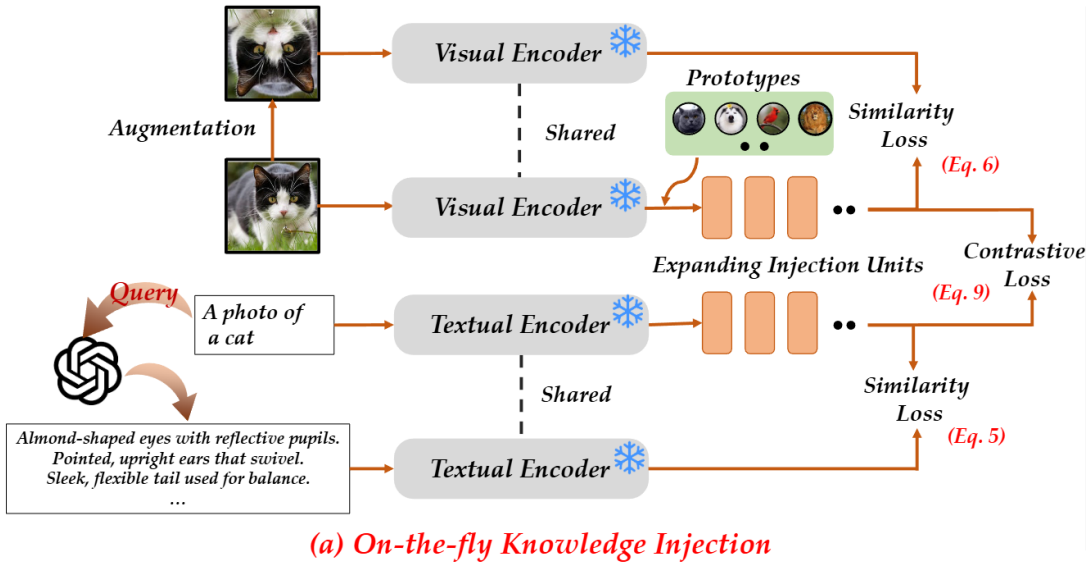
$$\min \ell(f(\mathbf{x}), y) + \frac{1}{2}\lambda \sum_k \Omega_k \left(\theta_k^{b-1} - \theta_k\right)^2$$

Example: Class-Incremental Learning

- When training with PTMs, extra models can serve as the generator for external knowledge and data.

Data level: Expert Knowledge Injection

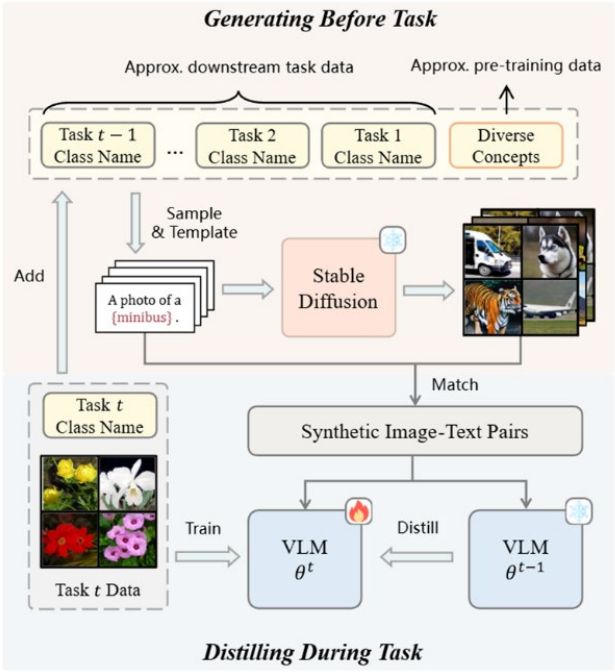
Generating descriptions for related classes to foster cross-modal alignment [Zhou et al., ICCV'25].



(a) On-the-fly Knowledge Injection

Data level: Data Generation

Generating images of previous classes with stable diffusion for replay to prevent forgetting [Wu et al., CVPR'25].

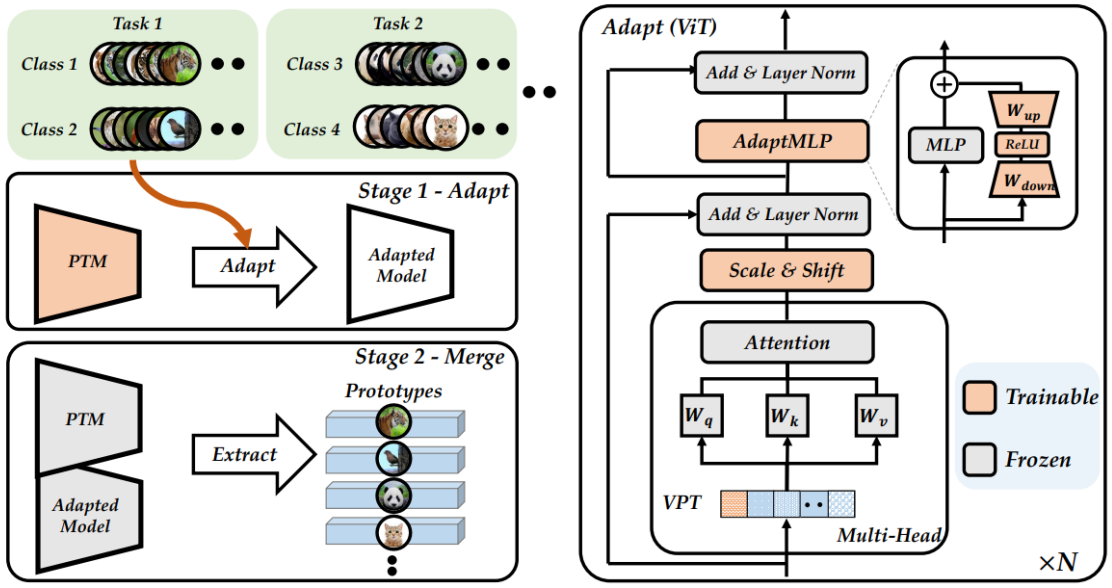


Example: Class-Incremental Learning

- When reusing pre-trained models, generalizable features can be adopted by model expansion

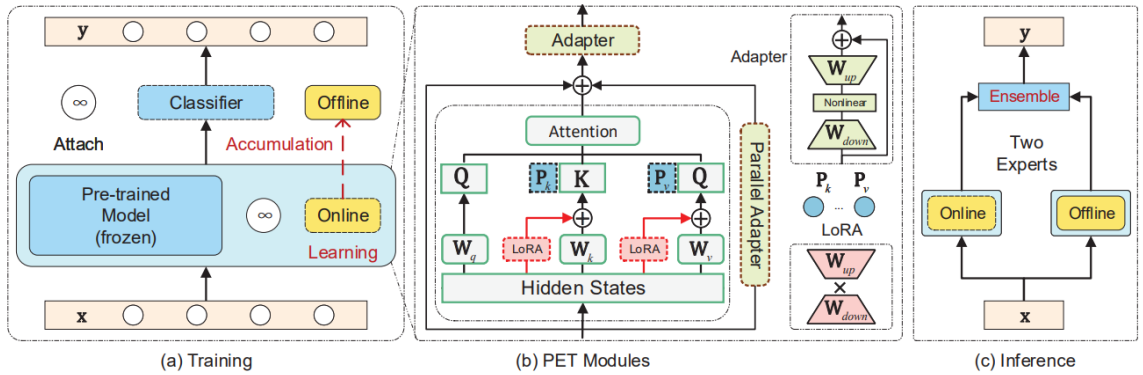
Inference level: Reuse Representation

Aggregate the features of PTM and adapted model for stronger representations [Zhou et al., IJCV'24].



Inference level: Model Merge

Building a dual-branch network. The online learner is updated via CE loss, while the offline learner is updated via EMA. During inference, the prediction is achieved via logit ensemble [Gao et al., ICCV'23].



Discussion: Making PTMs Reusable

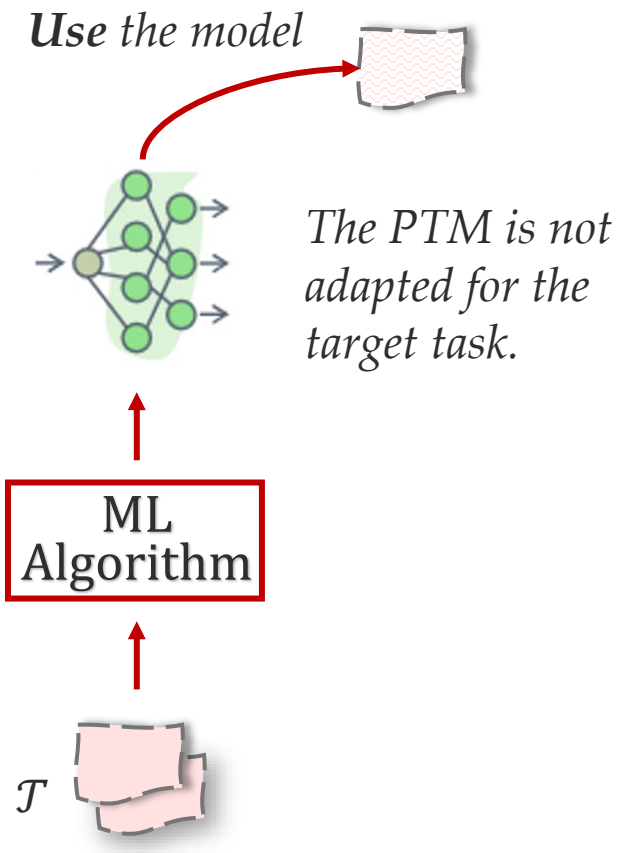
$$\min_{\boldsymbol{\phi}} \sum_{(\mathcal{S},\ \boldsymbol{\mathcal{Q}})} \sum_{(\boldsymbol{x},\boldsymbol{y})\sim\boldsymbol{\mathcal{Q}}} \ell(g(\boldsymbol{x};\boldsymbol{\phi},\boldsymbol{\mathcal{S}}),y)$$

[Vinyals et al. NIPS'16] [Finn et al. ICML'17]

- Mimic what the model will meet in the deployment stage in the pre-training.
- Make the model reusable (be reused in a more efficient and effective manner) for unseen tasks.

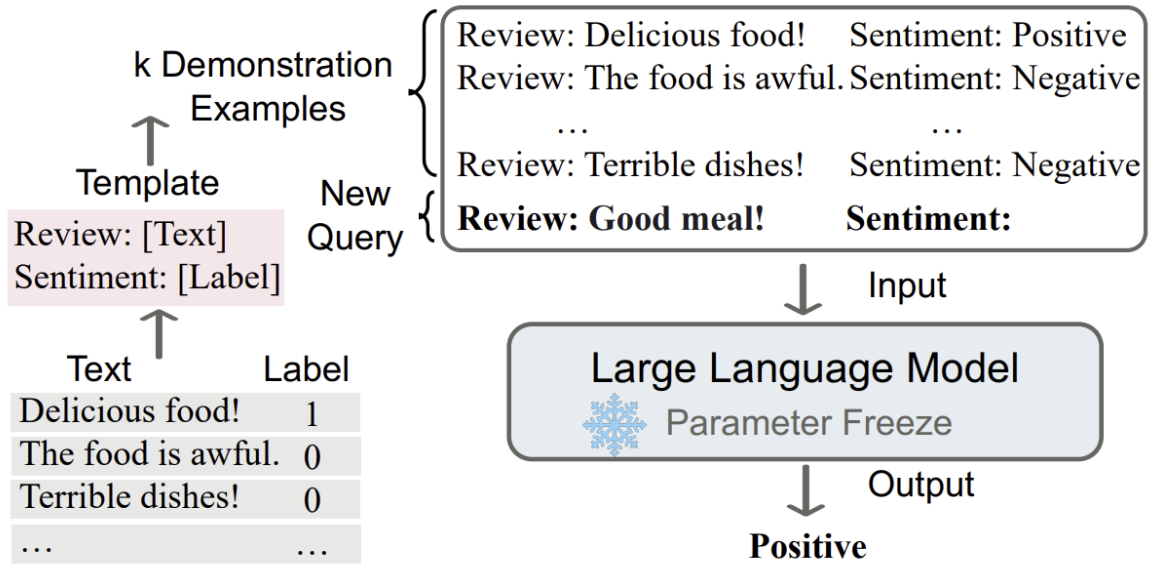
Discussion: Use Model and Model Reuse

- Use a model: direct make inference with the model.



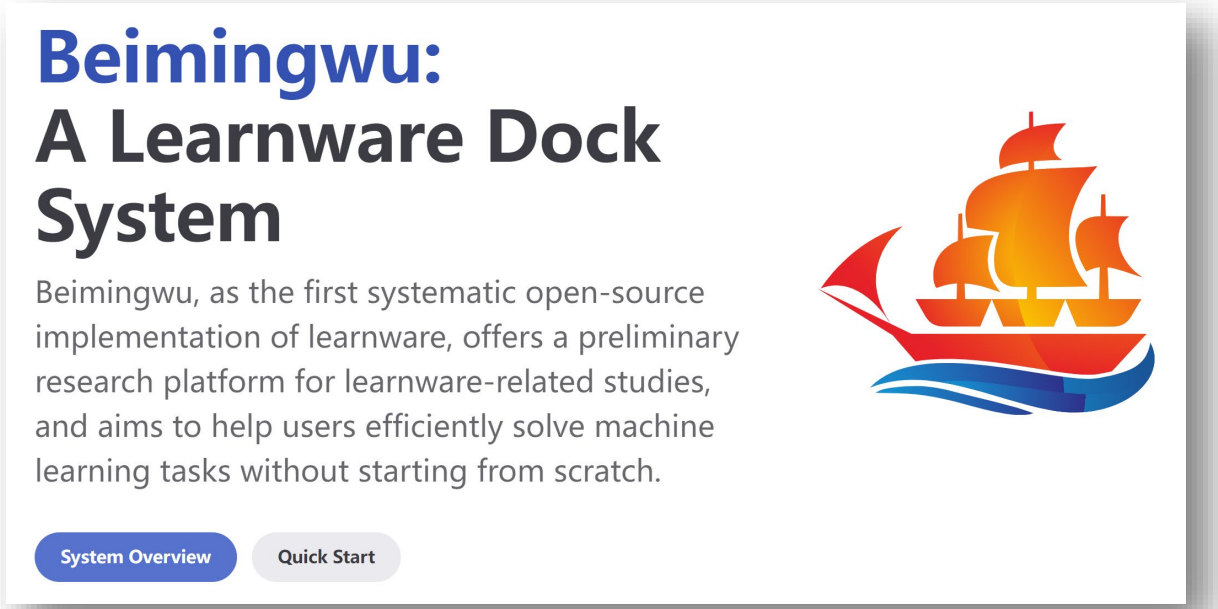
- Reuse by ``using'' the PTM: utilize the *in-context-learning* ability of LLMs [Dong et al., CoRR'23].

Taking the demonstration and a query as the input, large language models are responsible for making predictions



- *Transformer learn unseen models implicitly with demonstrations* [Garg et al., NeurIPS'22].
- *Transformer works in a similar way as gradient-descent optimizer implicitly* [Akyürek et al., ICLR'23].

Application Platform： Beimingwu



Support the entire process including the submitting, usability testing, organization, identification, deployment and reuse of learnwares.

Elements:

Learnware: unified unit
- Specify a unified learnware structure

Dock system architecture
- Design an integrated architecture

Key algorithms
- Implement baseline algorithms for each step

Infrastructure
- Engineering optimizations for computation and storage

An example based on Beimingwu

How to solve a new machine learning task with learnware dock system?

Beimingwu: a recently developed learnware dock system for research platform

```
# search learnwares
learnware_ids = client.search_learnware(user_info)

# load learnwares
learnware_list = client.load_learnware(learnware_ids)

# reuse and predict on user own task
y_predict = Reuser(learnware_list).predict(X)
```

Beimingwu provides **unified interfaces** to identify and deploy learnwares with **just a few key lines of code.**

With the constant submission of learnwares and advancements in algorithms, the interfaces will be increasingly powerful.

Use learnware dock system to solve new tasks

① Generate user info

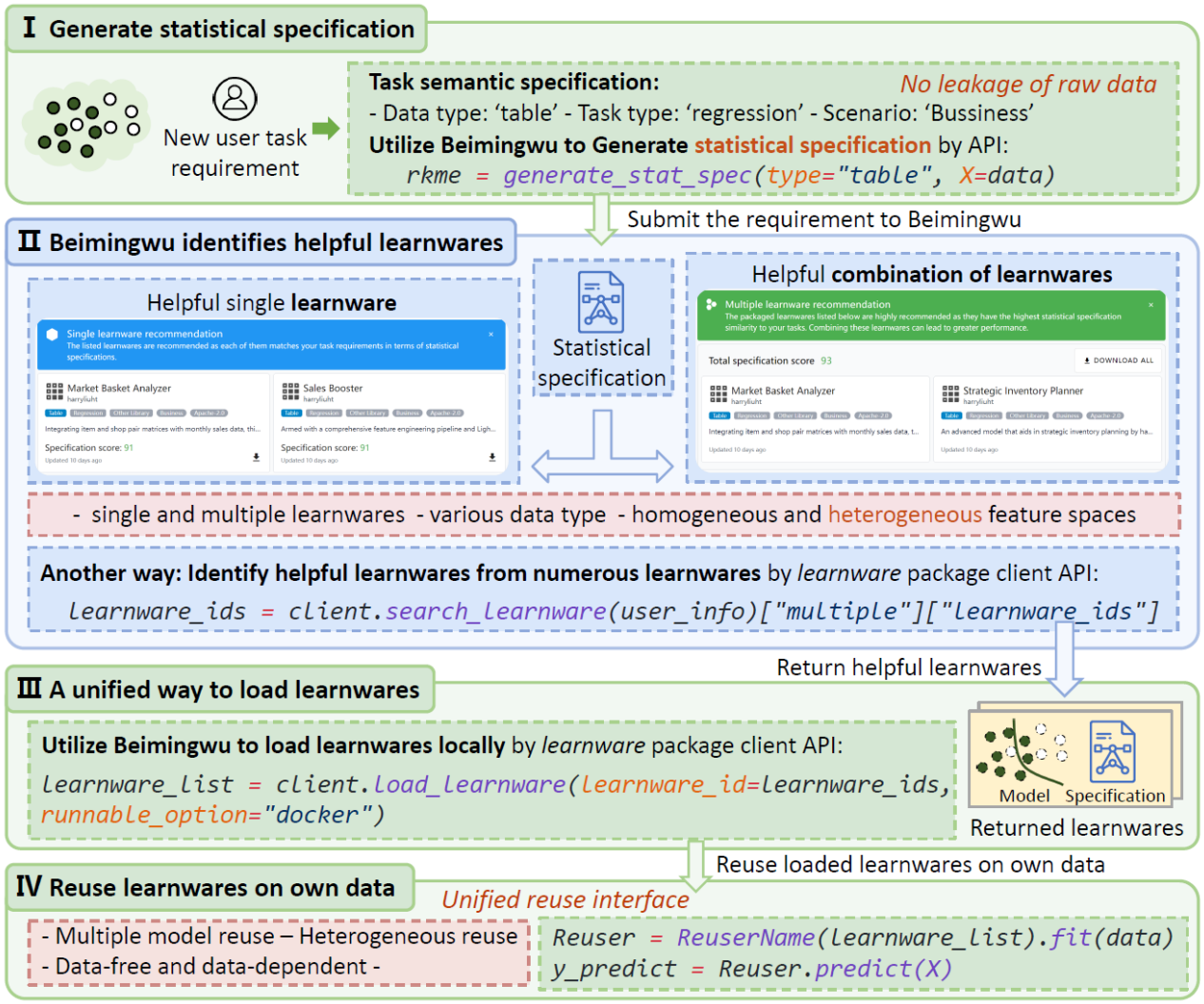
- statistical and semantic specification for user task

② Beimingwu identifies helpful learnwares

③ Load learnware locally

④ Reuse learnware on own data

With unified interfaces and architecture, the capabilities of the system will improve continuously through the constant submission of learnwares and advancements in algorithms.



Beimingwu: A learnware dock system

How to solve an arbitrary new task with Beimingwu

Beimingwu streamlines the model development through learnware paradigm

```
# generate statistical specification for task data
stat_spec = generate_stat_spec(data_type, X)
…
# search learnwares
learnware_ids = client.search_learnware(user_info)

# load learnwares
learnware_list = client.load_learnware(learnware_ids)

# reuse and predict on your task
y_predict = Reuser(learnware_list).predict(X)
```

1. **No need for extensive data**
2. **Minimal machine learning expertise**
3. **Local deployment of diverse models**
4. **No leakage of original data**

Conclusion

- Model reuse becomes an effective solution in various domains, such as for tabular data, visual, and language tasks.

- Model reuse methods can be categorized from various aspects
  - Homogeneous or heterogeneous
  - Reuse one-model or multiple models
  - Reuse from data-level, training-level, or inference-level.

  The ideas to reuse PTMs from multiple fields could be shared.

- To further boost the ability of model reuse, one important step is to identify which one or more PTMs to reuse from the model zoo.

南京大学 NANJING UNIVERSITY

LAMDA Learning And Mining from DatA

Thank You

Da-Wei Zhou 周大蔚
Nanjing University