Model Ratatouille: Recycling Diverse Models for Out-of-Distribution Generalization

Alexandre Ramé [1 2] **Kartik Ahuja** [1] **Jianyu Zhang** [1 3] **Matthieu Cord** [2 4] **Léon Bottou** [1 3] **David Lopez-Paz** [1]

Abstract

Foundation models are redefining how AI systems are built. Practitioners now follow a standard procedure to build their machine learning solutions: from a pre-trained foundation model, they fine-tune the weights on the target task of interest. So, the Internet is swarmed by a handful of foundation models fine-tuned on many diverse tasks: these individual fine-tunings exist in isolation without benefiting from each other. In our opinion, this is a missed opportunity, as these specialized models contain *rich and diverse* features. In this paper, we thus propose *model ratatouille*, a new strategy to recycle the multiple fine-tunings of the same foundation model on diverse auxiliary tasks. Specifically, we repurpose these auxiliary weights as initializations for multiple parallel fine-tunings on the target task; then, we average all fine-tuned weights to obtain the final model. This recycling strategy aims at maximizing the diversity in weights by leveraging the diversity in auxiliary tasks. Empirically, it improves the state of the art on the reference DomainBed benchmark for out-of-distribution generalization. Looking forward, this work contributes to the emerging paradigm of *updatable machine learning* where, akin to open-source software development, the community collaborates to reliably update machine learning models. Our code is released here.

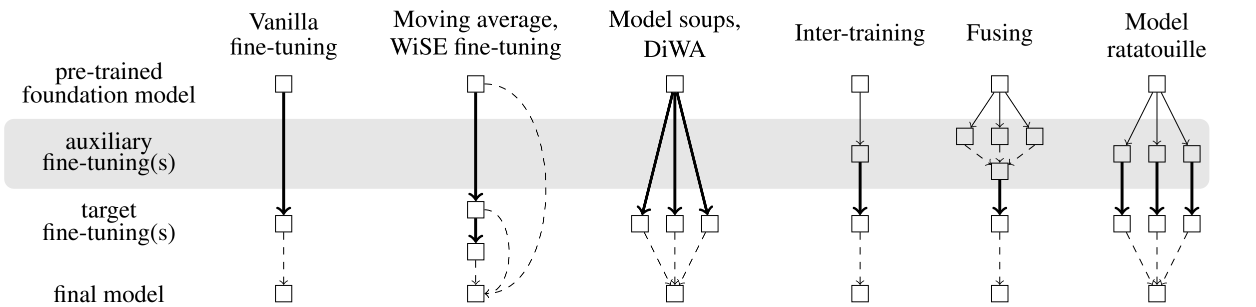
1. Introduction

The framework of *foundation models* (Bommasani et al., 2021) is fueling a spectacular adoption of machine learning solutions for real-world applications: also known as pre-trained models, these machine learning systems are trained on large-and-diverse data (Fang et al., 2022; Nguyen et al., 2022; Abnar et al., 2022) and easy to adapt to downstream tasks. Having ditched the "training from scratch" mentality, practitioners now follow a standardized two-step transfer learning strategy (Oquab et al., 2014). From some foundation model, they fine-tune on their target task with usually a limited amount of in-house data. Unfortunately, each of these fine-tunings risks latching onto specific patterns from the practitioners' training data (Arjovsky et al., 2019; Miller et al., 2020; Shah et al., 2020). Thus, these shortsighted models struggle to generalize on out-of-distribution (OOD) samples (Hendrycks & Dietterich, 2019; Taori et al., 2020; Gulrajani & Lopez-Paz, 2021; Hendrycks et al., 2021), negatively impacting human lives (Taylor et al., 2016; Zech et al., 2018). Increased OOD generalization would enable the responsible use of machine learning in real-world applications where robustness and safety are critical, such as medical imaging (DeGrave et al., 2021) and autonomous driving (Kuutti et al., 2020).

How to best fine-tune foundation models for OOD generalization is thus becoming a central topic of research. In particular, the recently discovered ability to average neural networks' weights (Izmailov et al., 2018; Neyshabur et al., 2020) has inspired a plethora of modern fine-tuning approaches. We illustrate some of them in Figure 1, such as moving averages (Izmailov et al., 2018), WiSE fine-tuning (Wortsman et al., 2022b), model soups (Wortsman et al., 2022a) and DiWA (Ramé et al., 2022). However, these strategies cannot accommodate the swarms of specialized fine-tunings of the same foundation model increasingly available in the Internet. Recent inter-training (Phang et al., 2018; Pruksachatkun et al., 2020) and fusing (Choshen et al., 2022b; Don-Yehiya et al., 2022) strategies recycle intermediate fine-tunings on auxiliary tasks to enrich the features before fine-tuning on the target task. However, the success of these recycling strategies usually depend on the similarity between the auxiliary and target tasks. We also argue in Section 2 that these strategies fail to fully leverage the diversity in auxiliary tasks, even though feature diversity improves OOD generalization (Laakom et al., 2021; Nayman et al., 2022; Jain et al., 2022; Zhang et al., 2022).

[1]Meta AI, Paris, France [2]Sorbonne Université, CNRS, ISIR, Paris, France [3]NYU, New-York, USA [4]Valeo.ai, Paris, France. Correspondence to: Alexandre Ramé <alexandre.rame@isir.upmc.fr>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.



	Vanilla fine-tuning	Moving average, WiSE fine-tuning	Model soups, DiWA	Inter-training	Fusing	Model ratatouille
OOD accuracy on DomainBed	63.3	66.5	67.6	65.6	65.8	**68.1** ✓

Figure 1. The different fine-tuning strategies discussed in this paper: vanilla fine-tuning (Oquab et al., 2014), moving average (Izmailov et al., 2018) and variants (Wortsman et al., 2022b), model soups (Wortsman et al., 2022a) and DiWA (Ramé et al., 2022), inter-training (Phang et al., 2018), fusing (Choshen et al., 2022b) and our proposed *model ratatouille*. They all start with a pre-trained foundation model. Some strategies fine-tune the pre-trained model on auxiliary tasks (thin solid arrows ⟶): these auxiliary fine-tunings can be performed by different contributors of the community on their own data. Then, all strategies perform fine-tuning on the target task of interest (thick solid arrows ⟶). Finally, the weights fine-tuned on the target task are used as is, or are averaged (dashed arrows - ->) into a final model. Ratatouille (i) enables compute parallelism throughout training, (ii) maximizes the amount of diversity in models' predictions, (iii) achieves state-of-the-art performance in DomainBed (Gulrajani & Lopez-Paz, 2021), the standard computer vision benchmark for OOD generalization and (iv) does not incur any inference or training overhead compared to a traditional hyperparameter search.

Thus, the central question of this paper is:

How can we best recycle diverse fine-tunings of a given foundation model towards strong out-of-distribution performance on our target task?

Our answer is a simple fine-tuning strategy we named *model ratatouille*,[1] illustrated in Figure 1 and described in Section 3. In a similar fashion to converting waste into reusable material for new uses, we take fine-tunings of the same foundation model on diverse auxiliary tasks and repurpose them as initializations to start multiple fine-tunings on the target downstream task. Specifically, we (i) fine-tune a copy of the foundation model on each auxiliary task, (ii) fine-tune each auxiliary model on the target task, and (iii) return as the final model the average of all target fine-tuned weights. In brief, while model soups (Wortsman et al., 2022a) averages multiple weights fine-tuned from a shared initialization, model ratatouille averages multiple weights fine-tuned from different initializations each inter-trained (Phang et al., 2018) on different auxiliary tasks. As we will see, ratatouille works because the fine-tunings remain linearly connected (Frankle et al., 2020; Mirzadeh et al., 2021) in the loss landscape (despite having different initializations) and thus can be averaged for improved performance.

We show the efficacy of model ratatouille in Section 4, where we set a new state of the art on DomainBed (Gulrajani & Lopez-Paz, 2021), the reference benchmark evaluating OOD generalization. We will show how we leverage the diversity across the auxiliary tasks to construct a final model with decreased over-fitting to task-specific patterns. As we discuss in our closing Section 5, this work contributes to the emerging paradigm of *updatable machine learning* (Raffel, 2023), where practitioners work in collaboration towards incrementally and reliably updating the capabilities of a machine learning model. As also highlighted in recent works (Matena & Raffel, 2022; Li et al., 2022a), we envision a future where deep neural networks are trained by following similar pipelines to the ones in open-source development with version control systems.

[1]We named our method after this traditional French dish for two main reasons. Firstly, the ratatouille is often used as a way to recycle leftover vegetables. Secondly, the ratatouille is better prepared by cooking each ingredient separately before mixing them: this technique ensures that each ingredient "will taste truly of itself", as noted by chef Joël Robuchon (Monaco, 2020).

2. Fine-Tuning for OOD Generalization

We start by describing our setup. We train a deep model $f_\theta = f_w \circ f_\phi$, where the featurizer f_ϕ is parametrized by the weights ϕ, the classifier f_w is parametrized by the weights w, and the joint model f_θ is parametrized by the concatenation weights $\theta = (w, \phi)$. We are dealing with out-of-distribution (OOD) generalization, and our aim is to find θ maximizing the test accuracy $\mathrm{acc}_{\mathrm{te}}(\theta)$. Specifically, while both train and test data correspond to the same target task—classifying images into a fixed set of classes—we allow a diversity (Ye et al., 2022) (a.k.a. covariate) distribution shift between the two, i.e., that the input distributions may change at test time. We highlight that this OOD generalization is critical in real-life applications, where the model needs to predict on samples from a new domain.

Vanilla fine-tuning. For OOD generalization, transfer learning (Oquab et al., 2014; Kirsch et al., 2022; Wenzel et al., 2022) with empirical risk minimization (Vapnik, 1992, ERM) is frustratingly difficult to beat (Gulrajani & Lopez-Paz, 2021), as measured on real-world datasets (Fang et al., 2023) such as PACS (Li et al., 2017), VLCS (Fang et al., 2013), OfficeHome (Venkateswara et al., 2017), TerraIncognita (Beery et al., 2018) or DomainNet (Peng et al., 2019). The recipe is (i) download a pre-trained featurizer with parameters ϕ^{pt}, (ii) plug a classifier w^{lp} compatible with the target task, and (iii) fine-tune the network with ERM on the target task. While the classifier w^{lp} could be initialized at random, linear probing (i.e., first learning only the classifier with frozen featurizer) improves results by preventing feature distortion (Kumar et al., 2022). For most users, particularly those with modest computation resources, the standard strategy is thus to transfer the knowledge from models pre-trained on large dataset such as ImageNet (Russakovsky et al., 2015), downloaded from public repositories such as `torchvision` (Marcel & Rodriguez, 2010), `huggingface` (Wolf et al., 2020) or `timm` (Wightman, 2019). The users usually launch multiple fine-tunings with different hyperparameters, and select the best based on some validation metric (Gulrajani & Lopez-Paz, 2021).

Weight averaging over epochs. Recently, *weight averaging* strategies came to the foreground (Szegedy et al., 2016; Izmailov et al., 2018; Draxler et al., 2018). While fine-tuning a pre-trained model, they saved and averaged checkpoints every few epochs to build the final model. Due to the non-linear nature of deep neural networks, the efficacy of weight averaging was a surprising observation, that Frankle et al. (2020) latter called the linear mode connectivity.

Observation 1 (LMC with different epochs (Izmailov et al., 2018))**.** *Two weights θ_a and θ_b, obtained at two different epochs of the same fine-tuning, satisfy the linear mode connectivity (LMC): for all $\lambda \in [0, 1]$,*

$$\text{acc}_{\text{te}}((1-\lambda)\cdot\theta_a + \lambda\cdot\theta_b) \gtrsim (1-\lambda)\cdot\text{acc}_{\text{te}}(\theta_a) + \lambda\cdot\text{acc}_{\text{te}}(\theta_b). \quad (1)$$

The LMC holds if the accuracy of the interpolated weights is above the interpolated accuracy. This definition is more restrictive than in the literature; for example, Frankle et al. (2020) only required less than 2% in error increase with regard to the worst endpoints. Consistently with Observation 1, recent works (Arpit et al., 2021; Cha et al., 2021; Wortsman et al., 2022b; Kaddour, 2022) weight average checkpoints along training to improve accuracies.

Weight averaging over runs. Perhaps motivated by these results, Neyshabur et al. (2020) (along with similar works (Nagarajan & Kolter, 2019; Frankle et al., 2020)) pushed the envelope of weight averaging techniques, and stated:

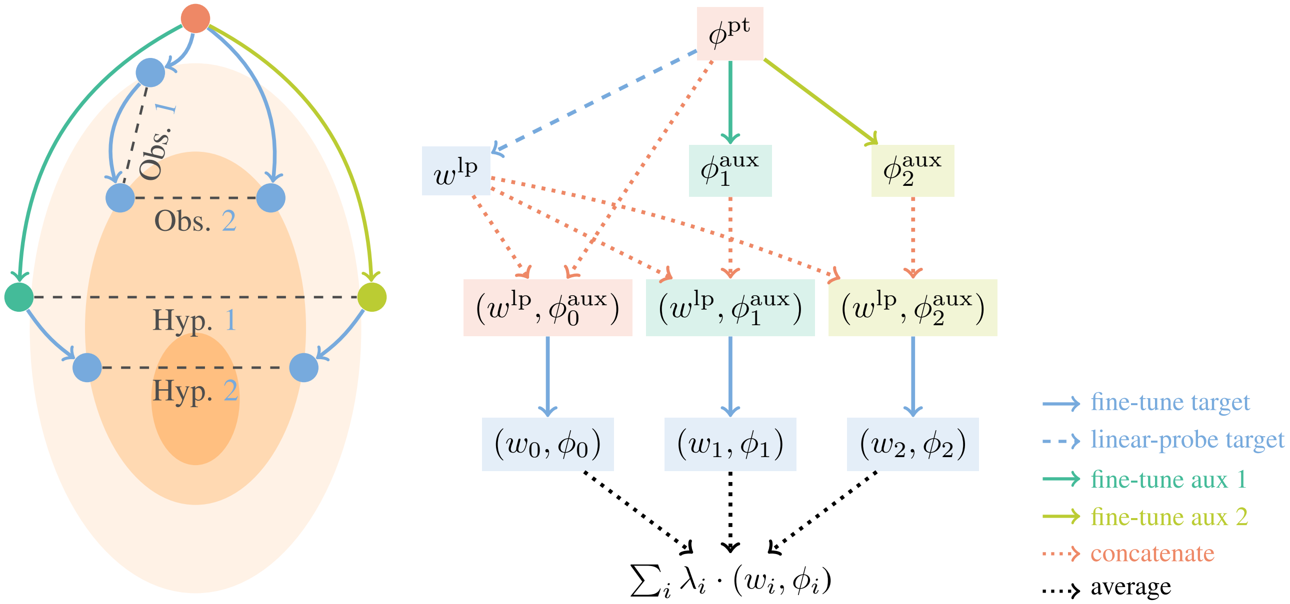
> there is no performance barrier between two instances of models trained from pre-trained weights, which suggests that the pre-trained weights guide the optimization to a flat basin of the loss landscape [. . .] Moreover, interpolating two random solutions from the same basin could generally produce solutions closer to the center of the basin, which potentially have better generalization performance than the endpoints.

Two *independent* fine-tunings—pre-trained similarly but differing in hyperparameter choices, data orders or other stochastic factors—also satisfy the LMC! More formally,

Observation 2 (LMC with different runs (Neyshabur et al., 2020))**.** *The LMC holds between θ_a and θ_b fine-tuned on the target task initialized from a shared pre-trained model.*

See Figure 2a for an illustration of Observations 1 and 2. Observation 2 inspired model soups (Wortsman et al., 2022a) and DiWA (Ramé et al., 2022)—the current state-of-the-art approaches for OOD generalization—to average all the weights obtained from a standard ERM hyperparameter search. However, the shared initialization constraint limits models diversity (Kuncheva & Whitaker, 2003; Aksela, 2003), especially when compared to methods that can combine arbitrary networks, for example via prediction averaging in deep ensembles (Lakshminarayanan et al., 2017).

Weight averaging over tasks. All the methods described so far fine-tune only on the target task: could auxiliary datasets, increasingly available online, be incorporated into the learning process to learn richer features? Such tasks could be an opportunity to recruit specialized features (Li et al., 2021a) that match our target task, ease optimization (Zhang et al., 2022; Zhang & Bottou, 2022), or "offer some high-level guidance to bridge the gaps between the pre-training and fine-tuning phases" (Chang & Lu, 2021). Following these ideas, *inter-training* (Phang et al., 2018; Pruksachatkun et al., 2020; Choshen et al., 2022a) performs an intermediate fine-tuning of the pre-trained model on some auxiliary task, before tackling the target task. However, the sequential nature of inter-training leads to catastrophic forgetting (Rebuffi et al., 2017) of useful knowledge contained in the original pre-trained model. Moreover, the choice of the auxiliary task plays a determinant role, since "when the wrong task is chosen, inter-training hurts results" (Choshen et al., 2022b). To address the shortcomings of inter-training, recent works (Choshen et al., 2022b; Don-Yehiya et al., 2022; Li et al., 2022a; Matena & Raffel, 2022; Ilharco et al., 2023; 2022) proposed to recycle weights fine-tuned on various auxiliary tasks. In particular, concurrent Choshen et al. (2022b) operates *fusing* at initialization; they (i) fine-tune one copy of the pre-trained model on each auxiliary task, (ii) average the auxiliary fine-tuning weights, and (iii) use such averaged model as the initialization for the target fine-tuning. By interpolation in weights, fusing combines into



(a) LMC conditions (b) Diagram of model ratatouille. (c) Arrow legend

Figure 2. Illustrations of (a) different linear mode connectivity (LMC) conditions, and (b) model ratatouille. In subplot (a), we illustrate Observation 1, about LMC between two checkpoints along the same target fine-tuning; Observation 2, about LMC between two target fine-tunings; Hypothesis 1, about LMC between two auxiliary fine-tunings; and Hypothesis 2, about LMC between two target fine-tunings initialized from auxiliary weights satisfying Hypothesis 1. In subplot (b), we offer a diagram of our proposed recycling strategy, where we (i) fine-tune a pre-trained model on auxiliary tasks, (ii) plug a linear probe on the pre-trained model and the auxiliary fine-tunings, (iii) fine-tune on the target task from each auxiliary weights, and (iv) return their weight average as the final model.

one single initialization the knowledge from multiple auxiliary tasks; yet fusing empirically provides only marginal gains in Section 4.1 for OOD generalization on DomainBed.

We posit that model fusing is performing weight averaging prematurely, destroying most diversity from auxiliary tasks even before the target task can benefit from it. To address these issues, next we propose *ratatouille*, a new recycling strategy that performs one target fine-tuning per auxiliary weights, and averages weights only as the very last step.

3. Model Ratatouille

3.1. Recycling Diverse Initializations

Our model ratatouille is a proposal to recycle diverse auxiliary fine-tunings of the same pre-trained model; it is compared against other fine-tuning strategies in Figure 1 and outlined in detail in Figure 2b. Ratatouille recycles these fine-tunings as diverse initializations to parallel fine-tunings on the target task. Compared to fusing, we delay the weight averaging, and in turn the destruction of diversity. Ratatouille follows this five-step recipe.

1. Download a featurizer ϕ^{pt} pre-trained on task T_0.
2. Fine-tune ϕ^{pt} on each auxiliary task T_i, obtaining $(w_i^{\text{aux}}, \phi_i^{\text{aux}})$ for $i = 0, \ldots, M-1$.
3. Replace each w_i^{aux} by w^{lp}, obtained by linear probing the original pre-trained model ϕ^{pt} on the target task T.
4. Fine-tune each $(w^{\text{lp}}, \phi_i^{\text{aux}})$ on the target task T, obtaining $\theta_i = (w_i, \phi_i)$ for $i = 0, \ldots, M-1$.
5. Return as final model $\sum_{i=0}^{M-1} \lambda_i \cdot \theta_i$. To select the interpolating coefficients, we use two strategies. The first "uniform" averages all weights with $\lambda_i = \frac{1}{M}$. The second "greedy" sorts the θ_i by descending accuracy on the in-distribution (ID) validation set, before greedily constructing an uniform average containing θ_i if and only if its addition lowers the ID validation accuracy.

If the weights from step 2 are made available online, ratatouille is without any training overhead compared to a traditional hyperparameter search. When compared to inter-training (Phang et al., 2018) and fusing (Choshen et al., 2022b), model ratatouille avoids the difficult choice of choosing one single initialization (Choshen et al., 2022a). The shared linear probe classifier facilitates LMC by preventing feature distortions (Kumar et al., 2022). Note that we consider the pre-training task as the auxiliary task "number zero" T_0; this resembles WiSE fine-tuning (Wortsman et al., 2022b) and aims at preserving the general-purpose knowledge contained in the original pre-trained model. The two selection strategies are those from model soups (Wortsman et al., 2022a; Ramé et al., 2022).

Successful weight averaging requires three conditions (Ramé et al., 2022). First, the weights must be individually accurate; by inter-training, ratatouille enriches the features and thus increases individual accuracies when the auxiliary tasks are well-chosen (Choshen et al., 2022a). Second, the weights should be sufficiently diverse to reduce variance. By removing the shared initialization constraint from model soups, ratatouille benefits from the additional diversity brought by specialization on various auxiliary tasks. In other words, ratatouille does not only rely on good similar auxiliary tasks; we also benefit from increased diversity, which was shown to be positively correlated with strong generalization (Laakom et al., 2021; Nayman et al., 2022; Jain et al., 2022; Zhang et al., 2022; Ramé et al., 2022). Third, the weights should be averageable; thus, for ratatouille to work, it requires a relaxation of the conditions under which the LMC holds, that we detail below.

3.2. Novel Linear Mode Connectivity Hypotheses

First, we introduce Hypothesis 1 that posits LMC between two models whose featurizers were fine-tuned on different auxiliary tasks.

Hypothesis 1 (LMC with different tasks)**.** *The LMC holds between (w, ϕ_a^{aux}) and (w, ϕ_b^{aux}) if ϕ_a^{aux} and ϕ_b^{aux} are featurizers fine-tuned on two auxiliary tasks initialized from the same pre-trained featurizer ϕ^{pt}. Here, w is the linear probe of ϕ^{pt} on the target task.*

Though this Hypothesis 1 was never formulated explicitly, it underlies fusing (Choshen et al., 2022b) and other strategies averaging auxiliary weights. Ratatouille relies on the following Hypothesis 2, which adds on top of Hypothesis 1 independent fine-tuning steps on the target task.

Hypothesis 2 (LMC with different auxiliary initializations)**.** *The LMC holds between θ_a and θ_b fine-tuned on the target task starting from initializations (w, ϕ_a^{aux}) and (w, ϕ_b^{aux}) satisfying Hypothesis 1.*

Hypothesis 2 is the first to posit the LMC between weights trained from different initializations. It hints towards a more general inheritance property: if two initializations satisfy LMC, then the two final weights would too.

We expect Hypotheses 1 and 2 to hold as long as the pre-training, auxiliary and target tasks are sufficiently similar, and if hyperparameters remain in a mild range. If they hold, we expect ratatouille to improve generalization abilities. But this, we can only answer empirically through proper experimentation.

4. Experiments

Our numerical experiments support three main claims, sorted in decreased granularity. First, Section 4.1 showcases the state-of-the-art (SoTA) results of ratatouille in DomainBed (Gulrajani & Lopez-Paz, 2021). Second, Section 4.2 illustrates how such gains arise from increased diversity across averaged models. Third, Section 4.3 empirically supports Hypotheses 1 and 2, the technical conditions enabling weight averaging's success. Finally, Section 4.4 discusses the impact of ratatouille for in-domain tasks. We invite the curious reader to consult our supplementary material. Among other experiments, we ablate in Appendix B the different components of ratatouille's procedure such as the number of auxiliary tasks, and propose in Appendix E a robust ratatouille to further improve performance. Our code is released at `https://github.com/facebookresearch/ModelRatatouille`.

4.1. SoTA Performance on DomainBed

Setup. Table 1 shows our main experiment comparing the various fine-tuning strategies on DomainBed (Gulrajani & Lopez-Paz, 2021), the reference benchmark evaluating OOD generalization. DomainBed contains five real-world datasets: PACS (Li et al., 2017), VLCS (Fang et al., 2013), OfficeHome (Venkateswara et al., 2017), TerraIncognita (Beery et al., 2018) and DomainNet (Peng et al., 2019). Each contains multiple domains about the same classification task: for example, the domains in OfficeHome are "Art", "ClipArt", "Product" and "Photo". Each domain is successively considered as the test while others are for training; we report the 22 per-domain results in Appendix F.2 but here analyze the averaged accuracy over the test domains. Standard deviations are obtained on 3 different random data splits. The network is a ResNet-50 (He et al., 2016) pre-trained on ImageNet (Russakovsky et al., 2015). Following DomainBed standards, each strategy leverages 20 runs with hyperparameters sampled from Table 3.

Approaches. Model soups (Wortsman et al., 2022a; Ramé et al., 2022) only differs from vanilla fine-tuning by the selection strategy: rather than selecting the model with highest ID validation accuracy out of the 20 runs, model soups either uniformly averages all weights or greedily selects some—as described in Section 3. For strategies leveraging auxiliary trainings, given a target dataset, we consider the other DomainBed's datasets as the auxiliary tasks. For example when tackling OfficeHome, out of the 20 runs, 4 are inter-trained on PACS, 4 on VLCS, 4 on TerraIncognita, 4 on DomainNet and 4 are directly transferred from ImageNet. Then, *model ratatouille is to inter-training as model soups is to vanilla fine-tuning*. In other words, while inter-training selects the best run based on ID accuracy, ratatouille applies the uniform or the greedy selection. Thus ratatouille

Table 1. Accuracies ($\%, \uparrow$) on the DomainBed (Gulrajani & Lopez-Paz, 2021) benchmark evaluating OOD generalization. Ratatouille sets a new SoTA by leveraging auxiliary tasks' diversity. The selection column indicates the weight selection strategy. The symbol "$*$" indicates inference overhead in functional ensembling. The symbol "†" indicates the averaging of all weights across 3 data splits.

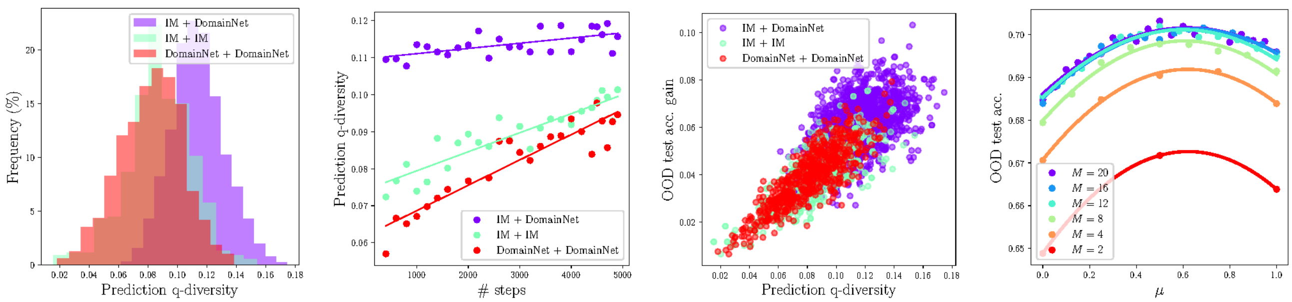
	Algorithm	**Selection**	**PACS**	**VLCS**	**OfficeHome**	**TerraInc**	**DomainNet**	**Avg**
	Vanilla fine-tuning	ID val	85.5 ± 0.2	77.5 ± 0.4	66.5 ± 0.3	46.1 ± 1.8	40.9 ± 0.1	63.3
	CORAL (Sun et al., 2016)	ID val	86.2 ± 0.3	78.8 ± 0.6	68.7 ± 0.3	47.6 ± 1.0	41.5 ± 0.1	64.6
	SWAD (Cha et al., 2021)	Loss-aware trajectory	88.1 ± 0.1	$\mathbf{79.1} \pm 0.1$	70.6 ± 0.2	50.0 ± 0.3	46.5 ± 0.1	66.9
	MA (Arpit et al., 2021)	Uniform trajectory	87.5 ± 0.2	78.2 ± 0.2	70.6 ± 0.1	50.3 ± 0.5	46.0 ± 0.1	66.5
	Deep ensembles* (Arpit et al., 2021)	Uniform	87.6	78.5	70.8	49.2	**47.7**	66.8
DiWA runs	Vanilla fine-tuning	ID val	85.9 ± 0.6	78.1 ± 0.5	69.4 ± 0.2	50.4 ± 1.8	44.3 ± 0.2	65.6
	Ensemble*	Uniform	88.1 ± 0.3	78.5 ± 0.1	71.7 ± 0.1	50.8 ± 0.5	47.0 ± 0.2	67.2
	Model soups	Uniform	88.7 ± 0.2	78.4 ± 0.2	72.1 ± 0.2	51.4 ± 0.6	47.4 ± 0.2	67.6
	Model soups	Greedy	88.0 ± 0.3	78.5 ± 0.1	71.5 ± 0.2	51.6 ± 0.9	$\mathbf{47.7} \pm 0.1$	67.5
	Model soups†	Uniform†	89.0	78.6	72.8	51.9	**47.7**	68.0
Our runs	Inter-training (Phang et al., 2018)	ID val	89.0 ± 0.0	77.7 ± 0.0	69.9 ± 0.6	46.7 ± 0.1	44.5 ± 0.1	65.6
	Ensemble* of inter-training	Uniform	89.2 ± 0.1	79.0 ± 0.2	72.7 ± 0.1	51.1 ± 0.3	47.2 ± 0.1	67.8
	Fusing (Choshen et al., 2022b)	ID val	88.0 ± 1.0	78.5 ± 0.8	71.5 ± 0.5	46.7 ± 1.8	44.4 ± 0.2	65.8
	Model ratatouille	Uniform	89.5 ± 0.1	78.5 ± 0.1	73.1 ± 0.1	51.8 ± 0.4	47.5 ± 0.1	68.1
	Model ratatouille	Greedy	$\mathbf{90.5} \pm 0.2$	78.7 ± 0.2	73.4 ± 0.3	49.2 ± 0.9	$\mathbf{47.7} \pm 0.0$	67.9
	Model ratatouille†	Uniform†	89.8	78.3	**73.5**	**52.0**	**47.7**	**68.3**

provides a single weight averaged network without any inference overhead. For real-world applications, auxiliary weights may be shared by the community; in that case, ratatouille is without training overhead, except when marked by the "†". Indeed, "†" symbol marks methods averaging $60 = 20 \times 3$ weights from 3 data splits, and thus benefiting from larger training budget. We further discuss ratatouille's training cost in Appendix B, and show in Appendix B.3 that ratatouile already performs well with only 5 runs. Ensembling strategies (marked by the symbol "$*$") average predictions with large inference overhead. For example, "ensemble* of inter-training" averages the predictions of the $M = 20$ models ratatouille averages in weights; we also report the scores from Arpit et al. (2021) for the deep ensembles* (Lakshminarayanan et al., 2017) of $M = 6$ models with different classifier initializations. For fusing, each run is initialized from $\sum_{i=0}^{4} \lambda_i \phi_i^{\text{aux}}$ where the λ_i hyperparameters sum to 1 and ϕ_i^{aux} are inter-trained on one the 4 other DomainBed's datasets or directly transferred from ImageNet. Finally, CORAL (Sun et al., 2016) is the best invariance approach; SWAD (Cha et al., 2021) and MA (Arpit et al., 2021) average weights along one training trajectory but differ in their selection strategy. The experimental setup and the approaches are further described in Appendix F.

Results. Table 1 shows that ratatouille achieves a new SoTA on DomainBed: with uniform selection, it achieves 68.1 and improves model soups by 0.5 points after averaging over all datasets. Precisely, model ratatouille beats model soups by 0.8 and 1.0 points on PACS and OfficeHome with uniform selection, and by 2.5 and 1.9 with greedy selection. On these two datasets, inter-training and fusing also succeed, yet they fail on TerraIncognita (both reach 46.7%) as all auxiliary tasks are distant from photos of animals in the wild; in contrast on TerraIncognita, ratatouille (51.8%) with uniform selection matches model soups (51.4%). This highlights the key strength of our ratatouille w.r.t. other recycling strategies such as fusing: namely, the robustness to the choice of auxiliary tasks. On VLCS, ratatouille is also generally beneficial (as visible in the per-domain results from Appendix F.2), except on one domain where the LMC breaks (as shown in Figure 13b from Appendix D). For DomainNet, ratatouille is SoTA though the gains are small w.r.t. model soups: we suspect this is because the initialization strategy becomes less critical for larger datasets (Chang & Lu, 2021) with more training epochs (see Figure 3b). In conclusion, ratatouille consistently improves generalization on DomainBed, and works best with appropriate auxiliary tasks: we remove the need to select only the *best* initialization. This is similar to model soups, that works best with appropriate hyperparameter ranges; they remove the need to select only the best set of hyperparameters.

4.2. Increased Diversity by Recycling

In Figure 3, we investigate how the diversity across models fine-tuned on the target task influences the OOD performance of their weight average. Here, we measure diversity with the prediction q-diversity (Kuncheva & Whitaker, 2003), which increases when models fail on different examples; this diversity measure is precisely defined in Appendix C.1, where we also arrive at similar conclusions using another diversity measure (Aksela, 2003). Following DiWA (Ramé et al., 2022), let the target task be OfficeHome,



(a) Diversity frequency. (b) Diversity vs. ft steps. (c) Acc. gain vs. diversity. (d) Acc. vs. mixing ratio.

Figure 3. Explorations on q-diversity (Kuncheva & Whitaker, 2003) and its positive impact on accuracy for the OOD test domain "Art" from OfficeHome. In (a), we compute the diversity between pairs of models either directly fine-tuned from ImageNet, either inter-trained on DomainNet: having one model from each initialization increases diversity. In (b), we plot this diversity along the 5k training steps. In (c), we observe that the more diverse the models, the higher the accuracy gain of their weight average compared to the average of their individual accuracies. In (d), we average M models: a proportion $(1-\mu)$ start directly from ImageNet, the others μ are inter-trained on DomainNet. The accuracy of the weight average is maximized when $\mu \approx 0.5$.

with "Art" as the test OOD domain; we thus train on the "ClipArt", "Product" and "Photo" domains. We consider models either only pre-trained on ImageNet or also inter-trained on DomainNet. These diversity experiments are applied on other DomainBed's datasets in Appendix C.2.

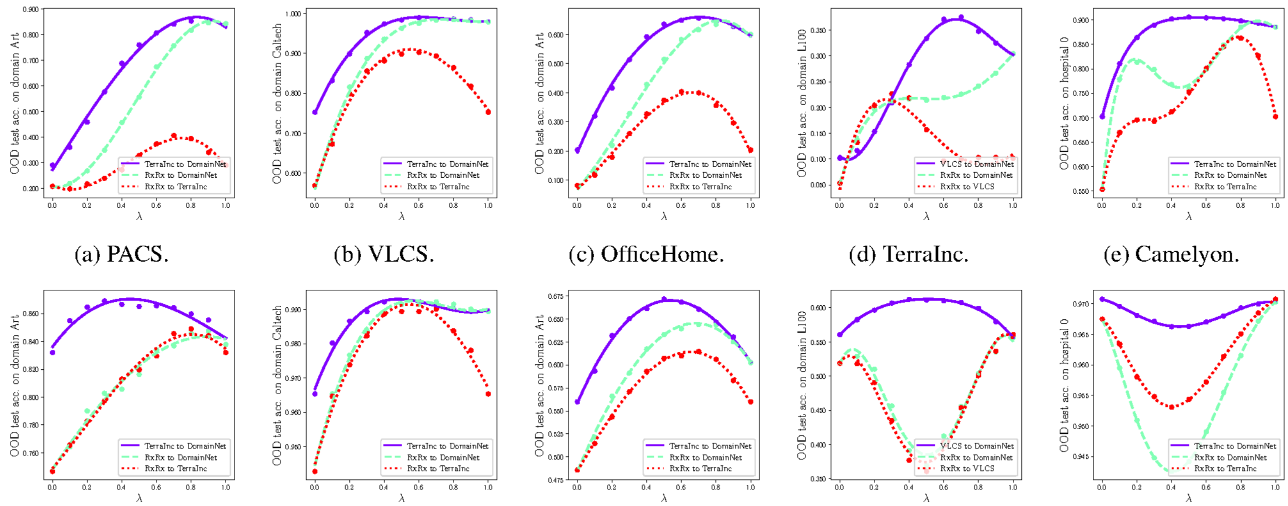
First, we verify that inter-training influences the diversity across fine-tuned models. Specifically, Figure 3a confirms that networks with different initializations are more diverse than networks initialized similarly. Then, Figure 3b verifies that this diversity gain comes from their initialization and remains along fine-tuning on the target task. Moreover, Figure 3c shows that diversity is positively linearly correlated with OOD generalization: specifically, we observe that having different initializations improves diversity and thus the accuracy of their weight average. Finally, in Figure 3d, we consider averaging M weights: a proportion $(1-\mu)$ start directly from ImageNet, the others μ were inter-trained on DomainNet. In the simplest case $M = 2$, using one model from each initialization leads to maximum accuracy; best performances are obtained around $\mu \approx 0.5$, where the final weight average has access to diverse initializations. In conclusion, each auxiliary task fosters the learning of diverse features (Li et al., 2021a; Gontijo-Lopes et al., 2022). Model ratatouille increases diversity and improves performance by removing a key limitation of model soups approaches (Wortsman et al., 2022a; Ramé et al., 2022); the need for all fine-tunings to start from a shared initialization.

4.3. Why Ratatouille Works

In Figure 4, we conclude our experiments by validating Hypotheses 1 and 2 when considering the five datasets from DomainBed. For the sake of completeness, we also analyze some successes and failure cases in "extreme" conditions when considering two distant unrelated medical datasets; RxRx (Taylor et al., 2019) and Camelyon (Koh et al., 2021) from the WILDS (Koh et al., 2021) benchmark. For each target task, we consider the first domain as the test OOD; the other domains are used for training.

We validate Hypothesis 1 in Figures 4a to 4e. For each dataset, we plot the test OOD accuracy for the weights $\left(w^{\text{lp}}, (1-\lambda)\cdot\phi_a^{\text{aux}} + \lambda\cdot\phi_b^{\text{aux}}\right)$, where the classifier w^{lp} is a linear probe of the ImageNet pre-trained featurizer $\phi_{\text{IM}}^{\text{pt}}$, and $\lambda\in[0,1]$ interpolates between ϕ_a^{aux} and ϕ_b^{aux}, obtained by fine-tuning on two auxiliary tasks initialized from $\phi_{\text{IM}}^{\text{pt}}$. First, we observe that task similarity influences OOD generalization since the test accuracies in Figure 4c agree with the fact that OfficeHome is most similar to DomainNet, not as similar to TerraIncognita, and most dissimilar to the medical dataset RxRx. Second, *the accuracy of the interpolated weights is above the interpolated accuracy*: this validates Hypothesis 1. The accuracy is even usually concave in λ.

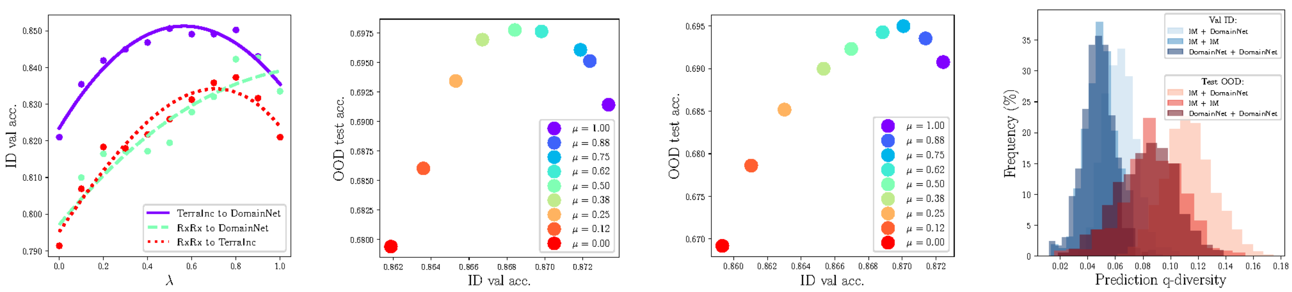
Similarly, we empirically support Hypothesis 2 in Figures 4f to 4j. For each dataset, we plot the test OOD accuracy obtained with weights $(1-\lambda)\cdot\theta_a + \lambda\cdot\theta_b$, where the coefficient $\lambda\in[0,1]$ interpolates between θ_a and θ_b, fine-tuned on the target task respectively starting from $(w^{\text{lp}}, \phi_a^{\text{aux}})$ and $(w^{\text{lp}}, \phi_b^{\text{aux}})$. We observe that Hypothesis 2 usually holds: for example, even recycling RxRx can help for OfficeHome on Figure 4h. Yet, Hypothesis 2 breaks on TerraIncognita and Camelyon in Figures 4i and 4j when RxRx is one of the two auxiliary tasks. In light of these results, we argue *that Hypothesis 2 holds as long as either the auxiliary or the target task is sufficiently similar to the pre-training task.* We speculate this prevents feature distortion (Kumar et al., 2022) and escaping a shared loss valley. Better understanding when LMC breaks is a promising research direction



(a) PACS. (b) VLCS. (c) OfficeHome. (d) TerraInc. (e) Camelyon.

(f) PACS. (g) VLCS. (h) OfficeHome. (i) TerraInc. (j) Camelyon.

Figure 4. Figures 4a to 4e validate Hypothesis 1 by plotting $\lambda \rightarrow \mathrm{acc}_{\mathrm{te}}\left(\left(w^{\mathrm{lp}}, (1-\lambda)\cdot\phi_a^{\mathrm{aux}} + \lambda\cdot\phi_b^{\mathrm{aux}}\right)\right)$, where w^{lp} is the linear probe of $\phi_{\mathrm{IM}}^{\mathrm{pt}}$, and ϕ_a^{aux} and ϕ_b^{aux} are fine-tuned on the two auxiliary datasets in the legend "Dataset$_a$ to Dataset$_b$". Figures 4f to 4j support Hypothesis 2 by plotting $\lambda \rightarrow \mathrm{acc}_{\mathrm{te}}((1-\lambda)\cdot\theta_a + \lambda\cdot\theta_b)$ where θ_a and θ_b are fine-tuned on the target task starting respectively from $(w^{\mathrm{lp}}, \phi_a^{\mathrm{aux}})$ and $(w^{\mathrm{lp}}, \phi_b^{\mathrm{aux}})$. We encounter two exceptions to Hypothesis 2 (Figures 4i and 4j), due to the fact that *neither* the auxiliary *nor* the target task bear enough similarity with the pre-training task.



(a) LMC in ID. (b) OOD vs. ID acc. (c) OOD vs. ID acc. (d) Diversity frequency.

Figure 5. The models were trained on ID domains "Clipart", "Product", and "Photo" from OfficeHome, thus "Art" is the OOD domain. First, in subplot (a), we validate Hypothesis 2 on the ID validation split. Then, we analyze the relations between diversity, ID and OOD accuracies. In subplot (b), we report the mean results when averaging $M = 8$ weights: $(1-\mu)$ are fine-tuned on OfficeHome directly from ImageNet, the others μ are inter-trained on DomainNet. We observe a lack of correlation between ID and OOD accuracies. We observe a similar trend in subplot (c), which mirrors the experiment from subplot (b) with the only difference that the proportion $(1-\mu)$ are inter-trained on PACS (rather than just transferred from ImageNet). In subplot (d), we compute the diversity (Kuncheva & Whitaker, 2003) between models either directly fine-tuned from ImageNet, either inter-trained on DomainNet. Though having different initializations increases diversity both in ID and in OOD, the diversity in ID remains smaller.

(Juneja et al., 2023; Lubana et al., 2022a); among other factors, we speculate that larger pre-training corpus (as in Qin et al. (2022)) or larger architectures (as in Li et al. (2022a)) may favor weight averaging strategies. In Appendix D, we further analyze Hypotheses 1 and 2, notably in a more complex setup where the intermediate tasks are successive fine-tunings on several auxiliary datasets.

4.4. Ratatouille for ID Tasks

Like previous weight averaging strategies (Izmailov et al., 2018; Wortsman et al., 2022a), model ratatouille also works for ID tasks; in particular, we verify in Figure 5a and in Appendix D.5 that the LMC holds in distribution. Yet, the gains are smaller in ID than in OOD, as confirmed by the lack of correlation between ID and OOD accuracies (Teney et al., 2022) in Figures 5b and 5c. This is explained by the fact that variance reduction (caused by weight averaging) is less beneficial in ID than in OOD. Theoretically, this is because, variance is smaller without distribution shift, as explained in Ramé et al. (2022). Empirically, this is consistent with models' diversity being smaller in ID, as shown in Figure 5d. Overall, diversity procedures are less useful in ID than in OOD. Ratatouille performs well OOD thanks to the diversity brought by diverse inter-trainings; for ID, we may sacrifice diversity and select one single optimal initialization. This finding contrasts with Miller et al. (2021) and goes against the prescription in Wenzel et al. (2022) that, "to make the model more robust on OOD data, the main focus should be to improve the ID classification error".

In conclusion, when aiming at OOD with ensembling strategies, our experiments suggest that there exists a trade-off between diversity and ID accuracy. This is critical for end-users as OOD is arguably more relevant than ID to ensure applicability in real-world applications, where train and test hardly ever follow the same distributions. This also explains occasional failures of the greedy selection (notably for TerraIncognita in Table 1): based on the ID validation accuracy, only a few runs are selected and averaged, causing smaller OOD accuracy than with the uniform selection.

5. Discussion: Towards Updatable Machine Learning

In the grand scheme of things, we see model recycling within the emerging *updatable machine learning* (Raffel, 2023) paradigm. The goal is to develop machine learning systems that can be incrementally improved and recombined, allowing for the collaborative creation of increasingly sophisticated AI systems. The core idea is to consider networks as pieces of software (Karpathy, 2017) and mirror the open-source development of software engineering via version control. Could it be possible that, someday, we could build decentralized open-source repositories, where we can clone, commit and merge neural networks towards an ever-improving AI system?

Recent works (Matena & Raffel, 2022; Li et al., 2022a; Don-Yehiya et al., 2022; Choshen et al., 2022a) and the proposed ratatouille give some primitives to learn neural networks in collaboration. Here, (i) cloning is simply weights downloading, (ii) commits are fine-tunings performed by individual contributors on their specific tasks, and (iii) branch merging is replaced by weight averaging. Advanced merging operations (Matena & Raffel, 2022; Li et al., 2022b; Jin et al., 2023) could help to better select the interpolating coefficients λ_i; neuron permutations strategies (Entezari et al., 2022; Ainsworth et al., 2023; Jordan et al., 2023) could remove the need for a shared pre-training, though (so far) these permutations have not improved models' accuracy.

In terms of privacy, such a federated learning setup (Li et al., 2019) where datasets can be kept private does indeed seem desirable. In terms of computation and sustainability, minimal communication across servers enable embarrassingly simple parallelization (Li et al., 2022a; Wortsman et al., 2023) and could reduce costs and CO2 emissions when training on multiple servers. This paradigm could also leverage the utilization of volunteer computing with single-GPU desktop machines, and complement approaches like Learning@home (Ryabinin & Gusev, 2020) or Petals (Borzunov et al., 2022). Finally, the contributors may potentially be incentivized financially through a system similar to blockchain technology (Sackfield, 2021).

If collaboration is the way forward, how can we ensure the *recyclability* of the shared models? In software engineering, practices such as unit tests greatly reduce the failure modes of programs; how can we borrow these ideas to *specify and test* neural networks? To measure models' shortcomings, we may leverage datasets as *test certificates* (Lopez-Paz et al., 2022). The community would monitor statistics on these datasets, e.g., accuracy, forgetting, and robustness against spurious correlations. Then, the reported scores could guide the choice of what models to clone, fine-tune, and merge. However, bad actors could directly include these datasets in their training data; then, should these external datasets be watermarked (Li et al., 2021b), or otherwise kept secret by some certifying authority?

These questions are all the more important as traditional foundation models (Bommasani et al., 2021) come with centralization and monetization, raise data privacy concerns, and lack transparency and reproducibility (Bommasani & Liang, 2021), which may hinder the democratization of AI. The ability to collaboratively improve weights represents a shift from *proprietary network training* to *open-source collaborative network building*, and could lead to the development of more responsible and reliable AI systems. We see this as an exciting possibility for the future of AI.

References

Abnar, S., Dehghani, M., Neyshabur, B., and Sedghi, H. Exploring the limits of large scale pre-training. In *ICLR*, 2022. (p. 1)

Ainsworth, S. K., Hayase, J., and Srinivasa, S. Git re-basin: Merging models modulo permutation symmetries. In *ICLR*, 2023. (p. 9)

Aksela, M. Comparison of classifier selection methods for improving committee performance. In *MCS*, 2003. (pp. 3, 6, and 16)

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint*, 2019. (p. 1)

Arpit, D., Wang, H., Zhou, Y., and Xiong, C. Ensemble of averages: Improving model selection and boosting performance in domain generalization. In *NeurIPS*, 2021. (pp. 3, 6, 21, 22, and 23)

Beery, S., Van Horn, G., and Perona, P. Recognition in Terra Incognita. In *ECCV*, 2018. (pp. 3, 5, and 21)

Beery, S., Agarwal, A., Cole, E., and Birodkar, V. The iwildcam 2021 competition dataset. *arXiv preprint*, 2021. (p. 24)

Bommasani, R. and Liang, P. Reflections on foundation models. `https://hai.stanford.edu/news/reflections-foundation-models`, 2021. (p. 9)

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint*, 2021. (pp. 1 and 9)

Borzunov, A., Baranchuk, D., Dettmers, T., Ryabinin, M., Belkada, Y., Chumachenko, A., Samygin, P., and Raffel, C. Petals: Collaborative inference and fine-tuning of large models. *arXiv preprint*, 2022. (p. 9)

Cha, J., Chun, S., Lee, K., Cho, H.-C., Park, S., Lee, Y., and Park, S. SWAD: Domain generalization by seeking flat minima. In *NeurIPS*, 2021. (pp. 3, 6, 21, 22, and 23)

Chang, T.-Y. and Lu, C.-J. Rethinking why intermediate-task fine-tuning works. *arXiv preprint*, 2021. (pp. 3 and 6)

Choshen, L., Venezian, E., Don-Yehia, S., Slonim, N., and Katz, Y. Where to start? analyzing the potential value of intermediate models. *arXiv preprint*, 2022a. (pp. 3, 4, 5, 9, and 21)

Choshen, L., Venezian, E., Slonim, N., and Katz, Y. Fusing finetuned models for better pretraining. *arXiv preprint*, 2022b. (pp. 1, 2, 3, 4, 5, 6, 14, 22, and 23)

DeGrave, A. J., Janizek, J. D., and Lee, S.-I. AI for radiographic COVID-19 detection selects shortcuts over signal. *Nature Machine Intelligence*, 2021. (p. 1)

Don-Yehiya, S., Venezian, E., Raffel, C., Slonim, N., Katz, Y., and Choshen, L. ColD fusion: Collaborative descent for distributed multitask finetuning. *arXiv preprint*, 2022. (pp. 1, 3, and 9)

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. (p. 19)

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. Essentially no barriers in neural network energy landscape. In *ICML*, 2018. (p. 3)

Eeckt, S. V. et al. Weight averaging: A simple yet effective method to overcome catastrophic forgetting in automatic speech recognition. *arXiv preprint*, 2022. (p. 20)

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. The role of permutation invariance in linear mode connectivity of neural networks. In *ICLR*, 2022. (p. 9)

Fang, A., Ilharco, G., Wortsman, M., Wan, Y., Shankar, V., Dave, A., and Schmidt, L. Data determines distributional robustness in contrastive language image pre-training (CLIP). In *ICML*, 2022. (p. 1)

Fang, A., Kornblith, S., and Schmidt, L. Does progress on ImageNet transfer to real-world datasets? *arXiv preprint*, 2023. (p. 3)

Fang, C., Xu, Y., and Rockmore, D. N. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *ICCV*, 2013. (pp. 3, 5, and 21)

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *ICML*, 2020. (pp. 2, 3, and 19)

Gontijo-Lopes, R., Dauphin, Y., and Cubuk, E. D. No one representation to rule them all: Overlapping features of training methods. In *ICLR*, 2022. (p. 7)

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization. In *ICLR*, 2021. (pp. 1, 2, 3, 5, 6, 14, 15, 21, and 22)

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016. (pp. 5 and 21)

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019. (p. 1)

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 2021. (p. 1)

Ilharco, G., Wortsman, M., Gadre, S. Y., Song, S., Hajishirzi, H., Kornblith, S., Farhadi, A., and Schmidt, L. Patching open-vocabulary models by interpolating weights. In *NeurIPS*, 2022. (p. 3)

Ilharco, G., Tulio Ribeiro, M., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. In *ICLR*, 2023. (p. 3)

Iwasawa, Y. and Matsuo, Y. Test-time classifier adjustment module for model-agnostic domain generalization. In *NeurIPS*, 2021. (p. 19)

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018. (pp. 1, 2, 3, 8, 14, and 20)

Jain, S., Tsipras, D., and Madry, A. Combining diverse feature priors. In *ICML*, 2022. (pp. 1 and 5)

Jin, X., Ren, X., Preotiuc-Pietro, D., and Cheng, P. Dataless knowledge fusion by merging weights of language models. 2023. (p. 9)

Jordan, K., Sedghi, H., Saukh, O., Entezari, R., and Neyshabur, B. Repair: Renormalizing permuted activations for interpolation repair. In *ICLR*, 2023. (p. 9)

Juneja, J., Bansal, R., Cho, K., Sedoc, J., and Saphra, N. Linear connectivity reveals generalization strategies. In *ICLR*, 2023. (p. 8)

Kaddour, J. Stop wasting my time! saving days of imagenet and BERT training with latest weight averaging. In *NeurIPS Workshop*, 2022. (p. 3)

Karpathy, A. Software 2.0. `https://karpathy.medium.com/software-2-0-a64152b37c35`, 2017. (p. 9)

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015. (p. 21)

Kirsch, A. C., Lakshminarayanan, B., Hu, C. H., Sculley, D., Phan, D., Tran, D., Snoek, J. R., Liu, J., Ren, J. J., van Amersfoort, J., Han, K., Buchanan, K., Murphy, K. P., Collier, M. P., Dusenberry, M. W., Band, N., Thain, N., Jenatton, R., Rudner, T. G. J., Gal, Y., Nado, Z., Mariet, Z., Wang, Z., and Ghahramani, Z. Plex: Towards reliability using pretrained large model extensions. In *ICML Workshop*, 2022. (p. 3)

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B., Haque, I., Beery, S. M., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. WILDS: A benchmark of in-the-wild distribution shifts. In *ICML*, 2021. (pp. 7 and 24)

Kumar, A., Raghunathan, A., Jones, R. M., Ma, T., and Liang, P. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *ICLR*, 2022. (pp. 3, 4, 7, and 21)

Kuncheva, L. I. and Whitaker, C. J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 2003. (pp. 3, 6, 7, 8, and 16)

Kuutti, S., Bowden, R., Jin, Y., Barber, P., and Fallah, S. A survey of deep learning applications to autonomous vehicle control. *T-ITS*, 2020. (p. 1)

Laakom, F., Raitoharju, J., Iosifidis, A., and Gabbouj, M. Within-layer diversity reduces generalization gap. In *ICML Workshop*, 2021. (pp. 1 and 5)

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017. (pp. 3 and 6)

Langnickel, L., Schulz, A., Hammer, B., and Fluck, J. BERT WEAVER: Using WEight AVERaging to enable lifelong learning for transformer-based models. *arXiv preprint*, 2022. (p. 21)

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Deeper, broader and artier domain generalization. In *ICCV*, 2017. (pp. 3, 5, and 21)

Li, M., Gururangan, S., Dettmers, T., Lewis, M., Althoff, T., Smith, N. A., and Zettlemoyer, L. Branch-Train-Merge: Embarrassingly parallel training of expert language models. *arXiv preprint*, 2022a. (pp. 2, 3, 8, and 9)

Li, Q., Wen, Z., and He, B. Federated learning systems: Vision, hype and reality for data privacy and protection. *arXiv preprint*, 2019. (p. 9)

Li, T., Huang, Z., Tao, Q., Wu, Y., and Huang, X. Trainable weight averaging for fast convergence and better generalization. *arXiv preprint*, 2022b. (p. 9)

Li, W.-H., Liu, X., and Bilen, H. Universal representation learning from multiple domains for few-shot classification. In *ICCV*, 2021a. (pp. 3 and 7)

Li, Y., Wang, H., and Barni, M. A survey of deep neural network watermarking techniques. *Neurocomputing*, 2021b. (p. 9)

Lopez-Paz, D., Bouchacourt, D., Sagun, L., and Usunier, N. Measuring and signing fairness as performance under multiple stakeholder distributions. *arXiv preprint*, 2022. (p. 9)

Lubana, E. S., Bigelow, E. J., Dick, R., Krueger, D., and Tanaka, H. Mechanistic lens on mode connectivity. In *NeurIPS Workshop*, 2022a. (p. 8)

Lubana, E. S., Trivedi, P., Koutra, D., and Dick, R. P. How do quadratic regularizers prevent catastrophic forgetting: The role of interpolation. In *CoLLAs*, 2022b. (p. 20)

Marcel, S. and Rodriguez, Y. Torchvision the machine-vision package of Torch. In *ACM*, 2010. (p. 3)

Maron, R. C., Hekler, A., Haggenmüller, S., von Kalle, C., Utikal, J. S., Müller, V., Gaiser, M., Meier, F., Hobelsberger, S., Gellrich, F. F., et al. Model soups improve performance of dermoscopic skin cancer classifiers. *European Journal of Cancer*, 2022. (p. 24)

Matena, M. and Raffel, C. Merging models with Fisher-weighted averaging. In *NeurIPS*, 2022. (pp. 2, 3, and 9)

Miller, J., Krauth, K., Recht, B., and Schmidt, L. The effect of natural distribution shift on question answering models. In *ICML*, 2020. (p. 1)

Miller, J. P., Taori, R., Raghunathan, A., Sagawa, S., Koh, P. W., Shankar, V., Liang, P., Carmon, Y., and Schmidt, L. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In *ICML*, 2021. (p. 9)

Mirzadeh, S. I., Farajtabar, M., Gorur, D., Pascanu, R., and Ghasemzadeh, H. Linear mode connectivity in multitask and continual learning. In *ICLR*, 2021. (p. 2)

Monaco, E. The right way to make ratatouille. `https://www.bbc.com/travel/article/20200812-the-right-way-to-make-ratatouille`, 2020. (p. 2)

Nagarajan, V. and Kolter, J. Z. Uniform convergence may be unable to explain generalization in deep learning. *NeurIPS*, 2019. (p. 3)

Nayman, N., Golbert, A., Noy, A., Ping, T., and Zelnik-Manor, L. Diverse ImageNet models transfer better. *arXiv preprint*, 2022. (pp. 1 and 5)

Neyshabur, B., Sedghi, H., and Zhang, C. What is being transferred in transfer learning? In *NeurIPS*, 2020. (pp. 1, 3, and 19)

Nguyen, T., Ilharco, G., Wortsman, M., Oh, S., and Schmidt, L. Quality not quantity: On the interaction between dataset design and robustness of CLIP. In *NeurIPS*, 2022. (p. 1)

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014. (pp. 1, 2, 3, and 14)

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. Moment matching for multi-source domain adaptation. In *ICCV*, 2019. (pp. 3, 5, and 21)

Phang, J., Févry, T., and Bowman, S. R. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint*, 2018. (pp. 1, 2, 3, 4, 6, 14, 21, 22, and 23)

Pruksachatkun, Y., Phang, J., Liu, H., Htut, P. M., Zhang, X., Pang, R. Y., Vania, C., Kann, K., and Bowman, S. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *ACL*, 2020. (pp. 1 and 3)

Qin, Y., Qian, C., Yi, J., Chen, W., Lin, Y., Han, X., Liu, Z., Sun, M., and Zhou, J. Exploring mode connectivity for pre-trained language models. In *EMNLP*, 2022. (p. 8)

Raffel, C. Building machine learning models like open source software. *ACM*, 2023. (pp. 2 and 9)

Ramé, A. and Cord, M. DICE: Diversity in deep ensembles via conditional redundancy adversarial estimation. In *ICLR*, 2021. (p. 16)

Ramé, A., Kirchmeyer, M., Rahier, T., Rakotomamonjy, A., Gallinari, P., and Cord, M. Diverse weight averaging for out-of-distribution generalization. In *NeurIPS*, 2022. (pp. 1, 2, 3, 5, 6, 7, 9, 14, 21, and 22)

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. iCaRL: Incremental classifier and representation learning. In *CVPR*, 2017. (p. 3)

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. ImageNet large scale visual recognition challenge. In *IJCV*, 2015. (pp. 3 and 5)

Ryabinin, M. and Gusev, A. Towards crowdsourced training of large neural networks using decentralized mixture-of-experts. *NeurIPS*, 2020. (p. 9)

Sackfield, W. SOTAMoon. `https://github.com/8W9aG/SOTAMoon`, 2021. (p. 9)

Shah, H., Tamuly, K., Raghunathan, A., Jain, P., and Netrapalli, P. The pitfalls of simplicity bias in neural networks. In *NeurIPS*, 2020. (p. 1)

Stojanovski, Z., Roth, K., and Akata, Z. Momentum-based weight interpolation of strong zero-shot models for continual learning. In *NeurIPS Interpolate Workshop*, 2022. (p. 20)

Sun, B., Feng, J., and Saenko, K. Return of frustratingly easy domain adaptation. In *AAAI*, 2016. (pp. 6, 22, and 23)

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. (pp. 3 and 20)

Taori, R., Dave, A., Shankar, V., Carlini, N., Recht, B., and Schmidt, L. Measuring robustness to natural distribution shifts in image classification. In *NeurIPS*, 2020. (p. 1)

Taylor, J., Yudkowsky, E., LaVictoire, P., and Critch, A. Alignment for advanced machine learning systems. *Ethics of Artificial Intelligence*, 2016. (p. 1)

Taylor, J., Earnshaw, B., Mabey, B., Victors, M., and Yosinski, J. RxRx1: An image set for cellular morphological variation across many experimental batches. In *ICLR Workshop*, 2019. (p. 7)

Teney, D., Lin, Y., Oh, S. J., and Abbasnejad, E. ID and OOD performance are sometimes inversely correlated on real-world datasets. *arXiv preprint*, 2022. (p. 8)

Vapnik, V. Principles of risk minimization for learning theory. In *NeurIPS*, 1992. (p. 3)

Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017. (pp. 3, 5, and 21)

Wenzel, F., Dittadi, A., Gehler, P. V., Simon-Gabriel, C.-J., Horn, M., Zietlow, D., Kernert, D., Russell, C., Brox, T., Schiele, B., Schölkopf, B., and Locatello, F. Assaying out-of-distribution generalization in transfer learning. In *NeurIPS*, 2022. (pp. 3 and 9)

Wightman, R. PyTorch Image Models. `https://github.com/rwightman/pytorch-image-models`, 2019. (pp. 3 and 19)

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. Transformers: State-of-the-art natural language processing. In *EMNLP*, 2020. (p. 3)

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., and Schmidt, L. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, 2022a. (pp. 1, 2, 3, 5, 7, 8, 14, and 22)

Wortsman, M., Ilharco, G., Kim, J. W., Li, M., Hajishirzi, H., Farhadi, A., Namkoong, H., and Schmidt, L. Robust fine-tuning of zero-shot models. In *CVPR*, 2022b. (pp. 1, 2, 3, 4, 14, and 20)

Wortsman, M., Gururangan, S., Li, S., Farhadi, A., Schmidt, L., Rabbat, M., and Morcos, A. S. lo-fi: distributed fine-tuning without communication. *TMLR*, 2023. (p. 9)

Ye, N., Li, K., Hong, L., Bai, H., Chen, Y., Zhou, F., and Li, Z. Ood-bench: Benchmarking and understanding out-of-distribution generalization datasets and algorithms. *CVPR*, 2022. (p. 2)

Yule, G. U. On the association of attributes in statistics. *Philosophical Transactions of the Royal Society of London.*, 1900. (p. 16)

Zech, J. R., Badgeley, M. A., Liu, M., Costa, A. B., Titano, J. J., and Oermann, E. K. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study. *PLOS Medicine*, 2018. (p. 1)

Zhang, J. and Bottou, L. Learning useful representations for shifting tasks and distributions. *arXiv preprint*, 2022. (p. 3)

Zhang, J., Lopez-Paz, D., and Bottou, L. Rich feature construction for the optimization-generalization dilemma. In *ICML*, 2022. (pp. 1, 3, and 5)

Model Ratatouille: Recycling Diverse Models for Out-of-Distribution Generalization

Supplementary material

This supplementary material is organized as follows:

- Appendix A describes the different fine-tuning strategies as equations.
- Appendix B analyzes ratatouille's components: Appendix B.1 ablates the number of auxiliary tasks, Appendix B.2 ablates the number of target fine-tuning steps and Appendix B.3 ablates the number of target fine-tuning runs.
- Appendix C enriches our diversity experiments.
- Appendix D further empirically analyzes the validity of Hypotheses 1 and 2 on additional setups.
- Appendix E introduces a new robust ratatouille strategy to (slightly) further improve performance.
- Appendix F describes and enriches our experiments on DomainBed (Gulrajani & Lopez-Paz, 2021).

A. Fine-Tuning Strategies as Equations

In Figure 1, we illustrated the different fine-tuning strategies. In Equation (2), we now provide an analytical formulation of these strategies with equations, where θ represents the weights, T_i the auxiliary tasks and T the target task.

$$
\begin{aligned}
&\theta = \mathrm{Train}\big(\theta^{\mathrm{pt}}, T\big), && \text{[Vanilla fine-tuning (Oquab et al., 2014)]}\\
&\theta = \mathrm{Train}\big(\theta^{\mathrm{pt}}, T, \mathrm{collect\_ckpts} = \mathrm{True}\big), && \text{[Moving average (Izmailov et al., 2018)]}\\
&\theta = (1-\lambda)\cdot \mathrm{Train}\big(\theta^{\mathrm{pt}}, T\big) + \lambda\cdot\theta^{\mathrm{pt}}, && \text{[WiSE fine-tuning (Wortsman et al., 2022b)]}\\
&\theta = \frac{1}{M}\sum_{i=0}^{M-1}\mathrm{Train}\big(\theta^{\mathrm{pt}}, T\big), && \text{[Model soups (Wortsman et al., 2022a)/DiWA (Ramé et al., 2022)]}\\
&\theta = \mathrm{Train}\big(\mathrm{Train}\big(\theta^{\mathrm{pt}}, T_i\big), T\big), && \text{[Inter-training (Phang et al., 2018)]}\\
&\theta = \mathrm{Train}\left(\sum_i \lambda_i\cdot \mathrm{Train}\big(\theta^{\mathrm{pt}}, T_i\big), T\right), && \text{[Fusing (Choshen et al., 2022b)]}\\
&\theta = \frac{1}{M}\sum_{i=0}^{M-1}\mathrm{Train}\big(\mathrm{Train}\big(\theta^{\mathrm{pt}}, T_i\big), T\big). && \text{[Model ratatouille (ours)]}
\end{aligned}
\tag{2}
$$

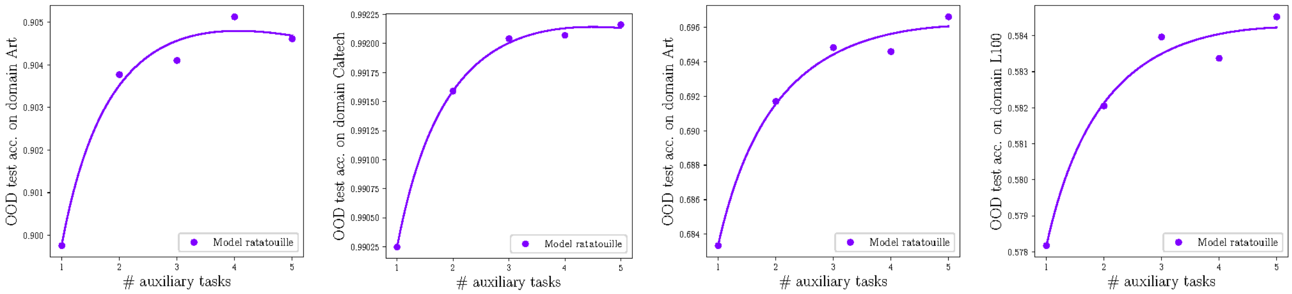
B. Ratatouille's Components Analysis

In this section we try to refine our understanding of the importance of various components in ratatouille.

Remark 1. *If auxiliary weights are shared by the community, recycling strategies cost no more than other fine-tuning strategies: recycling simply benefits from weights that would otherwise ignore each other and be discarded.*

B.1. Analysis of the Number of Auxiliary Tasks

In our Table 1, ratatouille leverages 5 auxiliary tasks for simplicity: ImageNet (which we consider as the auxiliary task "number zero"), and the 4 other datasets from DomainBed (out of the 5, as we leave out the target task to prevent any information leakage). In following Figure 6, we report the scores obtained using 1 to 5 auxiliary tasks: we always average $M = 20$ weights, the only difference is how they were initialized. When we have 1 auxiliary task, they were all inter-trained on this auxiliary task: when we have 2 auxiliary tasks, 10 are inter-trained on the first auxiliary task, 10 on the second: and etc. This validates that a greater number of auxiliary tasks leads to an increase in expected OOD accuracy. In the paper, we argue that this improvement is a result of the diversity gained through different specializations on different auxiliary tasks. We expect that further increasing the number of auxiliary datasets—beyond those from DomainBed—would further improve results.



(a) PACS. (b) VLCS. (c) OfficeHome. (d) TerraIncognita.

Figure 6. OOD accuracy (↑) for model ratatouille when increasing the number of auxiliary tasks and uniformly averaging all fine-tuned weights. For each target task, we consider the first domain as the test OOD; the other domains are used for training.

B.2. Analysis of the Number of Target Fine-Tuning Steps

One could argue that recycling auxiliary weights only benefit from longer training, part of which is delegated to the community. To invalidate this hypothesis, we ablate the number of training steps for model soups and ratatouille in Figure 7, on OfficeHome with "Art" as the OOD domain. We observe that even with unlimited number of training steps, model soups can not beat ratatouille. Therefore ratatouille's gains are made possible by fine-tuning on auxiliary datasets. We also observe that after a large number of epochs, the initialization becomes less important (as previously suggested in Figures 3b and 9b) and thus model ratatouille's gain over model soups decreases. In short, using the standard number of training steps (5000) provided by Domainbed is close to optimal.

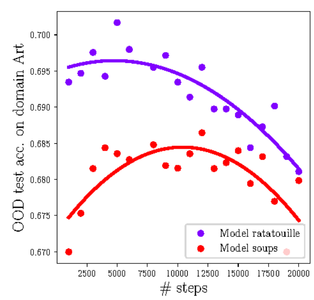
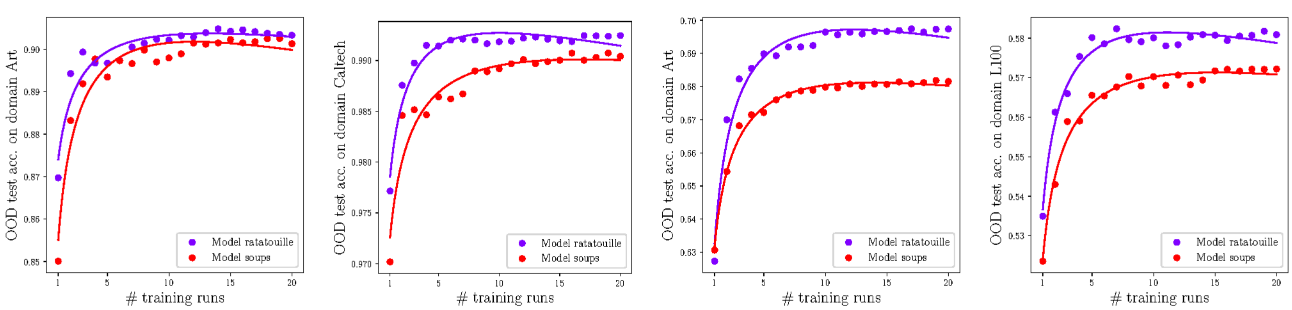


Figure 7. OfficeHome OOD accuracy with uniform averaging at different training steps.

B.3. Analysis of the Number of Target Fine-Tuning Runs

In our main experiment from Table 1, we train and average $M = 20$ independent weights, as 20 is the standard number of hyperparameter trials in DomainBed (Gulrajani & Lopez-Paz, 2021). In Figure 8 we ablate this value. We observe that a larger number of runs improves performance. If reducing the training budget is critical, one could already benefit from significant gains over model soups (and vanilla fine-tuning) with only 5 runs on the target task.



(a) PACS. (b) VLCS. (c) OfficeHome. (d) TerraIncognita.

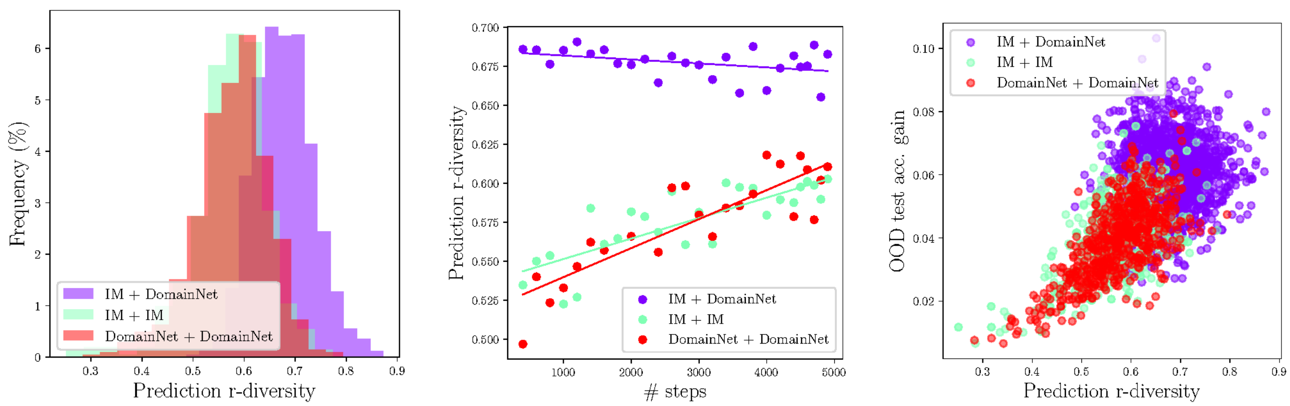
Figure 8. OOD accuracy (↑) for model ratatouille and model soups, when increasing the number of training runs and uniformly averaging all fine-tuned weights. For each target task, we consider the first domain as the test OOD; the other domains are used for training.

C. Diversity Experiments

C.1. Diversity Measures

As stated in "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy" (Kuncheva & Whitaker, 2003), "measuring diversity is not straightforward because there is no generally accepted formal definition". In Figure 3, we leverage the q-statistics Q, introduced in Yule (1900), brought up to date in Kuncheva & Whitaker (2003) and also used in Ramé & Cord (2021). Specifically, it is defined by $Q = \frac{N^{11}N^{00}-N^{01}N^{10}}{N^{11}N^{00}+N^{01}N^{10}}$, where N^{ij} is the number of times that the first classifier is (correct if $i=1$ or wrong if $i=0$) and the second classifier is (correct if $j=1$ or wrong if $j=0$). For example, N^{10} is the number of times that the first classifier is correct but not the second. Overall, classifiers which commit errors on different objects render Q small. To transform this similarity into a diversity measure that increases for more diverse classifiers, we report 1 minus the q-statistics, i.e., the r-diversity is $1-Q$.

In Figure 9, we leverage another diversity measure, the ratio-error (↑), introduced in Aksela (2003), brought up to date in Kuncheva & Whitaker (2003) and also used in Ramé & Cord (2021). This ratio-error is $\frac{N^{01}+N^{10}}{N^{00}}$ between the number of asynchronous errors and of simultaneous errors. This r-diversity leads to similar conclusions as with the q-diversity.

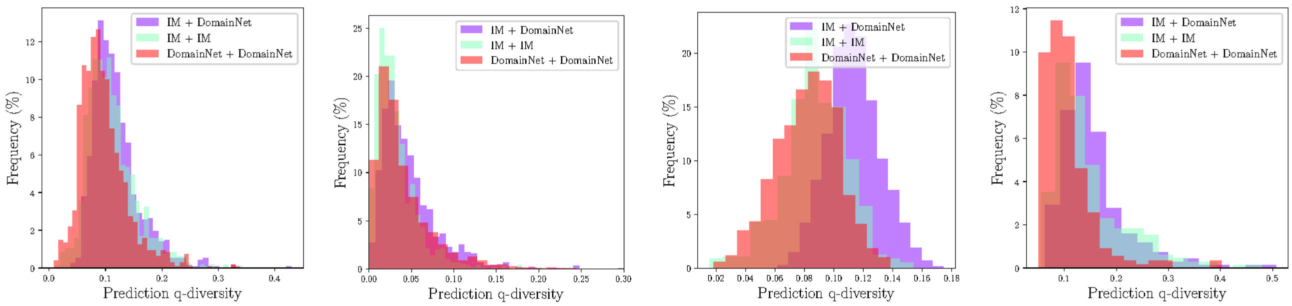


(a) R-diversity frequency. (b) R-diversity vs. fine-tuning steps. (c) Acc. gain vs. r-diversity.

Figure 9. We reproduce Figure 3 leveraging the ratio-error (Aksela, 2003) r-diversity measure.

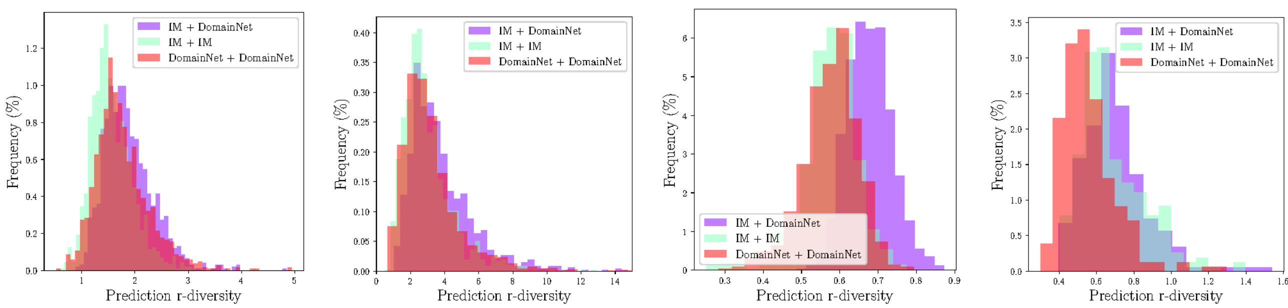
C.2. Additional Diversity Results

We now apply our diversity analysis on other DomainBed's datasets, where we consider the first domain as the test OOD; the other domains are used for training. Then, we compare the diversity—either measured with the q-statistics (in Figure 10) or in ratio-error (in Figure 11)—between two networks, either both directly transferred from ImageNet, either both inter-trained on DomainNet, either one directly transferred from ImageNet and the other inter-trained on DomainNet. Across all plots, we consistently observe that having different initializatons increases diversity, with the most pronounced shift seen on the OfficeHome and TerraIncognita datasets.



(a) PACS. (b) VLCS. (c) OfficeHome. (d) TerraIncognita.

Figure 10. Q-diversity in OOD.



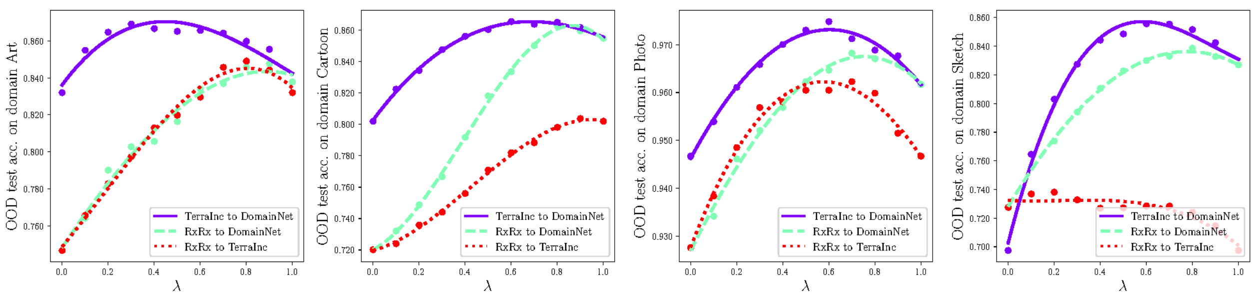
(a) PACS. (b) VLCS. (c) OfficeHome. (d) TerraIncognita.

Figure 11. R-diversity in OOD.

D. Linear Mode Connectivity Experiments

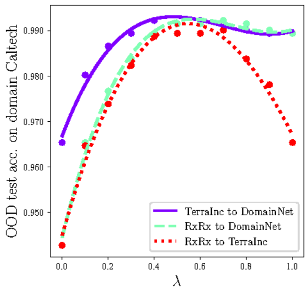
D.1. Linear Mode Connectivity per Target Dataset and Domain

We further empirically analyze our Hypothesis 2. In particular, we observe that the LMC usually holds except in two cases: (i) when the OOD test domain is the "LabelMe" domain from VLCS (in Figure 13b), and (ii) when both the target and the auxiliary tasks are distant from the pre-trained task, for example when tackling TerraIncognita or Camelyon with RxRx as an auxiliary task. We want to emphasize that we selected the "extreme" RxRx dataset precisely to test the empirical limits of the Hypothesis 2, but that, in practice, milder auxiliary tasks selection already helps for OOD generalization (and notably reaches SoTA performance in Section 4.1).

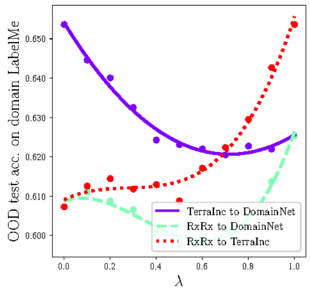


(a) "Art" as test. (b) "Cartoon" as test. (c) "Photo" as test. (d) "Sketch" as test.

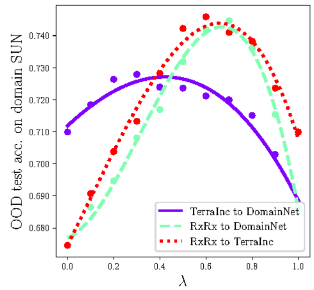
Figure 12. Empirical analysis of Hypothesis 2 on PACS.



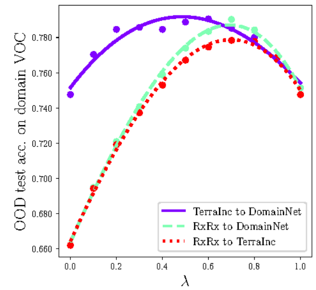
(a) "Caltech101" as test.



(b) "LabelMe" as test.

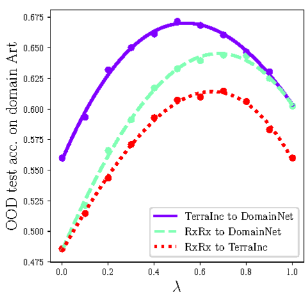


(c) "SUN09" as test.

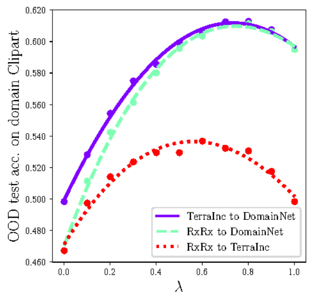


(d) "VOC2007" as test.

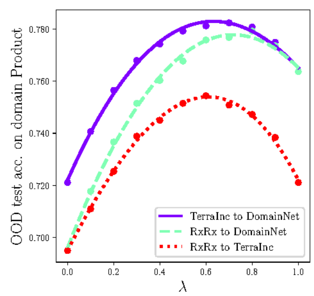
Figure 13. Empirical analysis of Hypothesis 2 on VLCS.



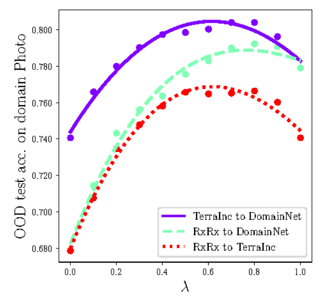
(a) "Art" as test.



(b) "Clipart" as test.

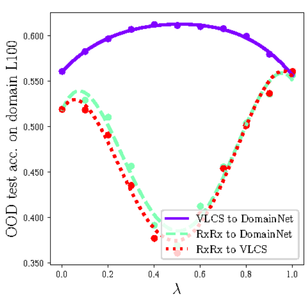


(c) "Product" as test.

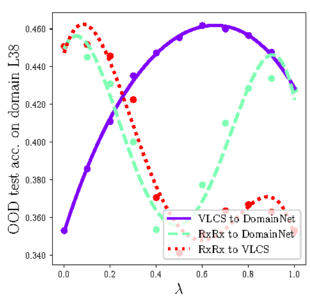


(d) "Photo" as test.

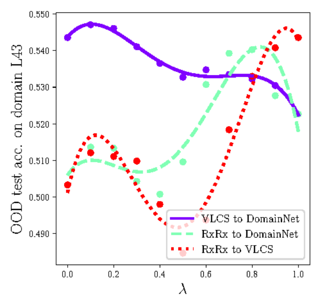
Figure 14. Empirical analysis of Hypothesis 2 on OfficeHome.



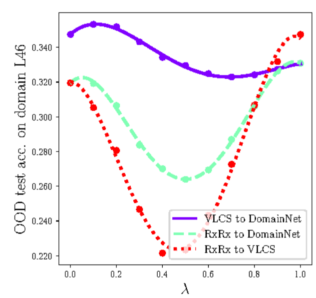
(a) "L100" as test.



(b) "L38" as test.

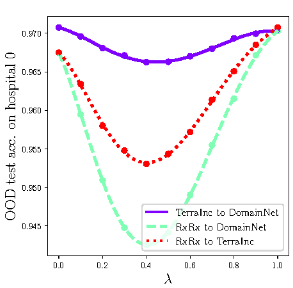


(c) "L43" as test.

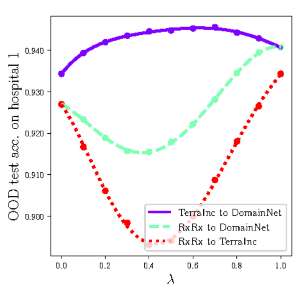


(d) "L46" as test.

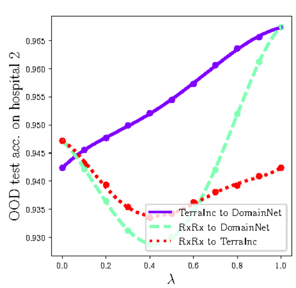
Figure 15. Empirical analysis of Hypothesis 2 on TerraIncognita.



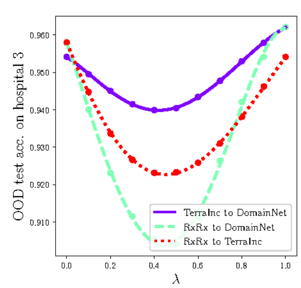
(a) "Hospital 0" as test.



(b) "Hospital 1" as test.



(c) "Hospital 2" as test.

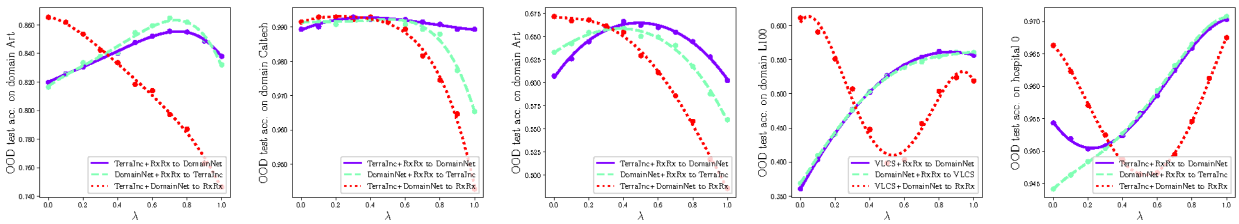


(d) "Hospital 3" as test.

Figure 16. Empirical analysis of Hypothesis 2 on Camelyon.

D.2. Linear Mode Connectivity across Three Weights

In the practical settings from Section 4.1, model ratatouille averages more than two weight inter-trained on different auxiliary tasks. For consistency, in Figure 17 we thus analyze LMC when interpolating across three fine-tuned weights. We observe the same successes, but also the same occasional failures when the target and task datasets are both simultaneously distant from the pre-training task, i.e., with RxRx as the auxiliary target and either TerraIncognita or Camelyon as the target task.

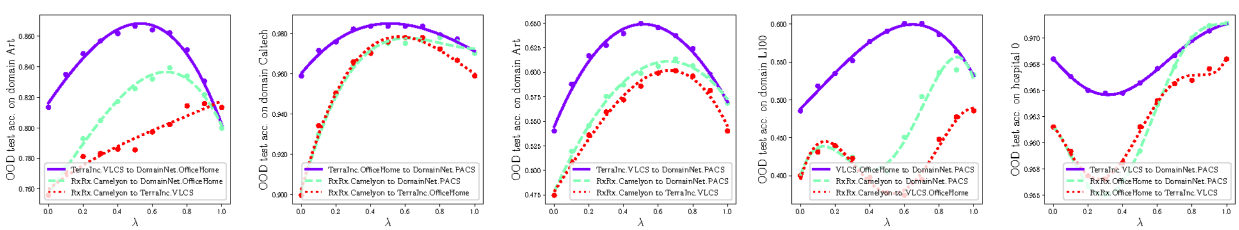


(a) PACS. (b) VLCS. (c) OfficeHome. (d) TerraIncognita. (e) Camelyon.

Figure 17. Empirical analysis of LMC when combining three weights. "Dataset$_a$+Dataset$_b$ to Dataset$_c$" means that the model for $\lambda = 0$ is the uniform weight average of θ_a and θ_b (fine-tuned on Dataset$_a$ and respectively on Dataset$_b$ before the target task) while the model θ_c for $\lambda = 1$ was fine-tuned on Dataset$_c$ before the target task; $0 < \lambda < 1$ interpolates between those three fine-tuned weights as $(1-\lambda)/2 \cdot \theta_a + (1-\lambda)/2 \cdot \theta_b + \lambda \cdot \theta_c$. On each target task, we consider the first domain as the test OOD domain.

D.3. Recycling of Weights Fine-tuned Sequentially on Multiple Datasets

In Figure 18, we empirically analyze Hypothesis 2 when the intermediate tasks are themselves several successive trainings on different auxiliary datasets. Thus the initialization for "TerraInc.VLCS" in Figure 18c was sequentially fine-tuned on two auxiliary tasks (TerraIncognita and then on VLCS) before tackling the target task (OfficeHome). The concavity of the curves validates the LMC in most setups. It hints towards a more general inheritance property of LMC: if two initializations satisfy the LMC, then the two fine-tuned weights too. Yet, analysis of this inheritance property is best left for future work.

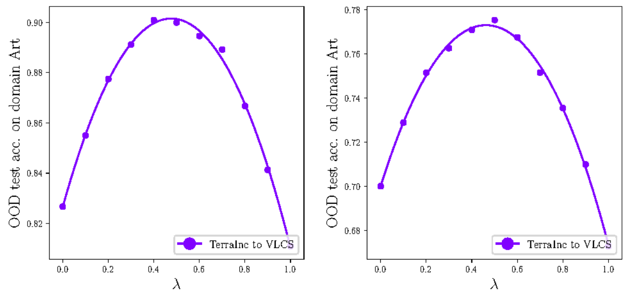


(a) PACS. (b) VLCS. (c) OfficeHome. (d) TerraIncognita. (e) Camelyon.

Figure 18. Empirical analysis of Hypothesis 2 when the intermediate tasks are themselves several successive fine-tunings on different auxiliary datasets. "Dataset$_a$.Dataset$_b$ to Dataset$_c$.Dataset$_d$" means that the model for $\lambda = 0$ was sequentially fine-tuned on Dataset$_a$ then Dataset$_b$ before fine-tuning on the target task, while the model for $\lambda = 1$ was sequentially fine-tuned on Dataset$_c$ then Dataset$_d$ before fine-tuning on the target task; $0 < \lambda < 1$ interpolates between those two fine-tuned weights.

D.4. Ratatouille with VITs architecture

We previously have experimented with the ResNet-50 architecture, the standard for DomainBed on which the OOD generalization community relies, enabling reproducibility and fair comparisons with concurrent papers. This exact same ResNet-50 architecture was the one used in the seminal works on LMC (Neyshabur et al., 2020; Frankle et al., 2020). Yet, the LMC is architecture agnostic. For the sake of completeness, we show in Figure 19 that the LMC holds with vision transformers (Dosovitskiy et al., 2021), namely the ViT-B16 "vit_base_patch16_224_in21k" from `timm` (Wightman, 2019), following the setup from Iwasawa & Matsuo (2021).

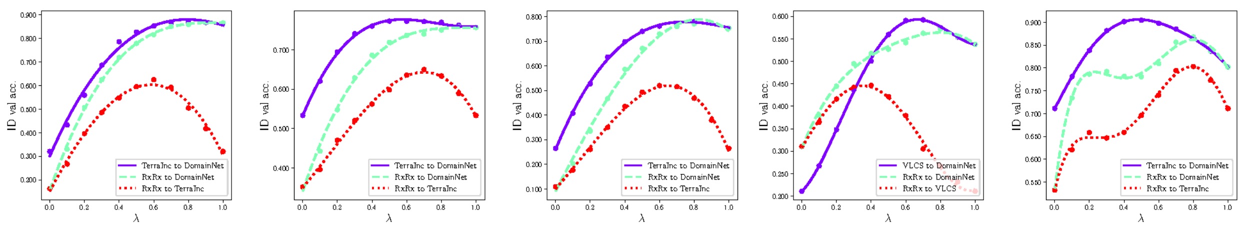


(a) PACS. (b) OfficeHome.

Figure 19. Empirical validation of Hypothesis 2 with VITs.

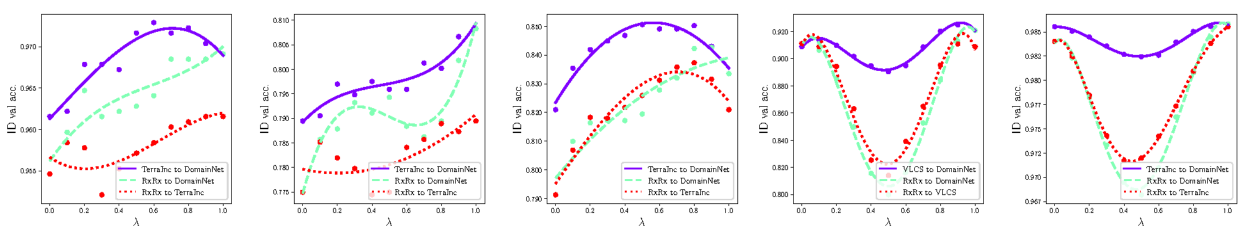
D.5. LMC in ID

In this section, we validate the LMC on ID samples, without distribution shift between train and test. The LMC holds in ID, except sometimes when RxRx is the auxiliary task, and with curves less concave than in OOD. These smaller gains when interpolating are because variance reduction via weight averaging is less beneficial in ID than in OOD.



(a) PACS. (b) VLCS. (c) OfficeHome. (d) TerraIncognita. (e) Camelyon.

Figure 20. Empirical analysis of Hypothesis 1 on the ID validation split. This mirrors the setup from Figures 4a to 4e.



(a) PACS. (b) VLCS. (c) OfficeHome. (d) TerraIncognita. (e) Camelyon.

Figure 21. Empirical analysis of Hypothesis 2 on the ID validation split. This mirrors the setup from Figures 4f to 4j.

E. Robust Inter-Training

Recycled soups leverage weights fine-tuned on various auxiliary tasks; these starting points may sometimes may too specialized, and less general than the initial pre-trained weights. To preserve the pre-trained general knowledge, in this section we consider a *robust inter-training* strategy where the initializations are robustified via moving average (Szegedy et al., 2016; Izmailov et al., 2018; Wortsman et al., 2022b) along the auxiliary fine-tuning. This follows recent evidence that moving average can reduce catastrophic forgetting (Lubana et al., 2022b; Eeckt et al., 2022; Stojanovski et al., 2022;

Langnickel et al., 2022). As in Equation (2), these new robust strategies can be written as:

$$\begin{aligned} \theta &= \mathrm{Train}\big(\mathrm{Train}\big(\theta^{\mathrm{pt}}, T_i, \mathrm{collect\_ckpts} = \mathrm{True}\big), T\big), && \text{[Robust inter-training]} \\ \theta &= \frac{1}{M} \sum_{i=0}^{M-1} \mathrm{Train}\big(\mathrm{Train}\big(\theta^{\mathrm{pt}}, T_i, \mathrm{collect\_ckpts} = \mathrm{True}\big), T\big). && \text{[Robust model ratatouille]} \end{aligned} \tag{3}$$

As we show in Table 2, moving average improves the initializations and thus the transfer abilities of inter-training, and as a consequence also improves model ratatouille. Better understanding how to further improve auxiliary initializations is an interesting research direction, already discussed in Choshen et al. (2022a).

Table 2. Accuracies (%, ↑) on the DomainBed (Gulrajani & Lopez-Paz, 2021) benchmark.

Algorithm	Selection	PACS	VLCS	OfficeHome	TerraInc	DomainNet	Avg
Vanilla fine-tuning	ID val	85.9 ± 0.6	78.1 ± 0.5	69.4 ± 0.2	50.4 ± 1.8	44.3 ± 0.2	65.6
Model soups	Uniform	88.7 ± 0.2	78.4 ± 0.2	72.1 ± 0.2	51.4 ± 0.6	47.4 ± 0.2	67.6
Model soups	Greedy	88.0 ± 0.3	78.5 ± 0.1	71.5 ± 0.2	51.6 ± 0.9	$\underline{47.7} \pm 0.1$	67.5
Model soups†	Uniform†	89.0	78.6	72.8	51.9	<u>47.7</u>	68.0
Inter-training (Phang et al., 2018)	ID val	89.0 ± 0.0	77.7 ± 0.0	69.9 ± 0.6	46.7 ± 0.1	44.5 ± 0.1	65.6
Model ratatouille	Uniform	89.5 ± 0.1	78.5 ± 0.1	73.1 ± 0.1	51.8 ± 0.4	47.5 ± 0.1	68.1
Model ratatouille	Greedy	$\underline{90.5} \pm 0.2$	78.7 ± 0.2	$\underline{73.4} \pm 0.3$	49.2 ± 0.9	$\underline{47.7} \pm 0.0$	67.9
Model ratatouille†	Uniform†	89.8	78.3	**73.5**	<u>52.0</u>	<u>47.7</u>	**68.3**
Robust inter-training	ID val	88.9 ± 0.5	77.8 ± 0.1	71.8 ± 0.6	47.3 ± 0.5	44.5 ± 0.2	66.1
Robust model ratatouille	Uniform	89.7 ± 0.1	$\underline{78.6} \pm 0.2$	73.0 ± 0.1	51.9 ± 0.2	47.4 ± 0.1	68.2
Robust model ratatouille	Restricted	$\mathbf{90.7} \pm 0.1$	$\mathbf{78.8} \pm 0.2$	$\underline{73.4} \pm 0.2$	50.6 ± 0.3	47.6 ± 0.1	68.2
Robust model ratatouille†	Uniform†	89.8	<u>78.6</u>	<u>73.4</u>	**52.1**	**47.8**	**68.3**

F. DomainBed

F.1. Experimental Details

Datasets. We consider PACS (Li et al., 2017), VLCS (Fang et al., 2013), OfficeHome (Venkateswara et al., 2017), TerraIncognita (Beery et al., 2018) and DomainNet (Peng et al., 2019). Domains are split into 80% (used as training and evaluation) and 20% (used as validation). When considered as the target task, each domain is successively considered as the test domain while others are for training. When considered as an auxiliary task, we train on all domains. Critically, the procedure to obtain the pool of initializations is agnostic to the target task or the test domain, and thus is done only once.

Training protocol. In all cases, we follow the training protocol from DomainBed. For each dataset, we perform a random search of 20 trials on the mild hyperparameter distributions described in Table 3. We use a ResNet-50 (He et al., 2016) pre-trained on ImageNet, with a dropout layer before the newly added dense layer and fine-tuned with frozen batch normalization layers. The optimizer is Adam (Kingma & Ba, 2015). The linear probe classifier are obtained with default hyperparameters from Table 3 and features extracted from the ImageNet pre-trained featurizer. All runs are trained for 5k steps, except on DomainNet for 15k steps as done in concurrent works (Arpit et al., 2021; Cha et al., 2021; Ramé et al., 2022). When the featurizer was inter-trained on auxiliary datasets, it remains frozen during the first 200 steps to prevent feature distortion (Kumar et al., 2022). As in Ramé et al. (2022); Cha et al. (2021), validation accuracy is calculated every 50 steps for VLCS, 500 steps for DomainNet and 100 steps for others. Our code is released at `https://github.com/facebookresearch/ModelRatatouille`.

Table 3. Hyperparameters, their default values and distributions for random search.

Hyperparameter	Default value	Random distribution (DomainBed)	Random distribution (Ours, DiWA and SWAD)
Learning rate	$5 \cdot 10^{-5}$	$10^{\mathcal{U}(-5,-3.5)}$	$[1, 3, 5] \cdot 10^{-5}$
Batch size	32	$2^{\mathcal{U}(3,5.5)}$	32
ResNet dropout	0	$[0, 0.1, 0.5]$	$[0, 0.1, 0.5]$
Weight decay	0	$10^{\mathcal{U}(-6,-2)}$	$[10^{-6}, 10^{-4}]$

Baselines. Vanilla fine-tuning was named Empirical Risk Minimization in previous papers; CORAL (Sun et al., 2016) is the best invariance-based approach; their scores are taken from DomainBed (Gulrajani & Lopez-Paz, 2021). MA (Arpit et al., 2021) and SWAD (Cha et al., 2021) average weights along the trajectory of a vanilla fine-tuning; their scores are taken from their respective papers. Deep ensembles* averages the predictions of $M = 6$ models, each trained with different classifier initializations on different data splits; the scores are taken from Arpit et al. (2021). Model soups (Wortsman et al., 2022a) averages the weights obtained from different vanilla fine-tunings; for fair comparison, we report the scores achieved in DiWA (Ramé et al., 2022) with linear probing. Fusing averages at initialization 5 auxiliary weights ϕ_i^{aux}; for each of the 20 runs and $0 \leq i < 5$, we sample $\kappa_i \sim \text{Unif}(0, 4)$ and choose $\lambda_i = \frac{e^{\kappa_i}}{\sum_{j=0}^{4} e^{\kappa_j}}$, i.e., the featurizer is initialized from $\sum_{i=0}^{4} \frac{e^{\kappa_i}}{\sum_{j=0}^{4} e^{\kappa_j}} \phi_i^{\text{aux}}$.

Model and weight selection. We consider the training-domain validation set protocol. From each run, we thus take the weights at the epoch with maximum accuracy on the ID validation dataset. The greedy weight selection is also based on this ID validation set. This greedy strategy is not possible for † approaches, that average uniformly the $M = 20 \times 3 = 60$ weights from the 3 data splits: indeed, there is no shared ID validation dataset.

F.2. Results per Target Dataset and Domain

Tables below detail results per domain for the 5 datasets from DomainBed. The average scores were reported in Table 1.

Table 4. Accuracy ($\%, \uparrow$) on PACS (best in **bold** and second underlined).

	Algorithm	Selection	Art	Cartoon	Photo	Sketch	Avg
	Vanilla fine-tuning	ID val	84.7 ± 0.4	80.8 ± 0.6	97.2 ± 0.3	79.3 ± 1.0	85.5 ± 0.2
	CORAL (Sun et al., 2016)	ID val	88.3 ± 0.2	80.0 ± 0.5	97.5 ± 0.3	78.8 ± 1.3	86.2 ± 0.3
	SWAD (Cha et al., 2021)	Loss-aware trajectory	89.3 ± 0.5	83.4 ± 0.6	97.3 ± 0.3	82.5 ± 0.8	88.1 ± 0.1
	MA (Arpit et al., 2021)	Uniform trajectory	89.1 ± 0.1	82.6 ± 0.2	97.6 ± 0.0	80.5 ± 0.9	87.5 ± 0.2
	Deep ensembles* (Arpit et al., 2021)	Uniform	88.3	83.6	96.5	81.9	87.6
DiWA runs	Vanilla fine-tuning	ID val	86.8 ± 0.8	80.6 ± 1.0	97.4 ± 0.4	78.7 ± 2.0	85.9 ± 0.6
DiWA runs	Ensemble*	Uniform	89.6 ± 0.2	81.6 ± 0.3	97.8 ± 0.2	83.5 ± 0.5	88.1 ± 0.3
DiWA runs	Model soups	Uniform	90.1 ± 0.2	82.8 ± 0.6	98.3 ± 0.1	83.3 ± 0.4	88.7 ± 0.2
DiWA runs	Model soups	Greedy	89.3 ± 0.2	82.8 ± 0.2	98.0 ± 0.1	82.0 ± 0.9	88.0 ± 0.3
DiWA runs	Model soups†	Uniform†	90.6	83.4	98.2	83.8	89.0
Our runs	Inter-training (Phang et al., 2018)	ID val	89.2 ± 1.0	85.3 ± 0.7	97.5 ± 0.0	84.2 ± 0.2	89.0 ± 0.0
Our runs	Ensemble* of inter-training	Uniform	90.4 ± 0.2	83.7 ± 0.3	97.9 ± 0.2	84.9 ± 0.3	89.2 ± 0.1
Our runs	Fusing (Choshen et al., 2022b)	ID val	90.8 ± 0.1	79.1 ± 1.4	98.0 ± 0.4	84.1 ± 2.1	88.0 ± 1.0
Our runs	Model ratatouille	Uniform	90.3 ± 0.2	84.4 ± 0.1	98.7 ± 0.1	84.8 ± 0.1	89.5 ± 0.1
Our runs	Model ratatouille	Greedy	$\mathbf{90.9} \pm 0.1$	$\mathbf{86.5} \pm 1.1$	98.6 ± 0.0	$\mathbf{85.9} \pm 0.4$	$\mathbf{90.5} \pm 0.2$
Our runs	Model ratatouille†	Uniform†	90.6	84.7	**98.8**	85.0	89.8

Table 5. Accuracy ($\%, \uparrow$) on VLCS (best in **bold** and second underlined).

	Algorithm	Selection	Caltech	LabelMe	SUN	VOC	Avg
	Vanilla fine-tuning	ID val	97.7 ± 0.4	64.3 ± 0.9	73.4 ± 0.5	74.6 ± 1.3	77.5 ± 0.4
	CORAL (Sun et al., 2016)	ID val	98.3 ± 0.1	$\mathbf{66.1} \pm 1.2$	73.4 ± 0.3	77.5 ± 1.2	78.8 ± 0.6
	SWAD (Cha et al., 2021)	Loss-aware trajectory	98.8 ± 0.1	63.3 ± 0.3	$\mathbf{75.3} \pm 0.5$	79.2 ± 0.6	$\mathbf{79.1} \pm 0.1$
	MA (Arpit et al., 2021)	Uniform trajectory	99.0 ± 0.2	63.0 ± 0.2	74.5 ± 0.3	76.4 ± 1.1	78.2 ± 0.2
	Deep ensembles* (Arpit et al., 2021)	Uniform	98.7	64.5	72.1	78.9	78.5
DiWA runs	Vanilla fine-tuning	ID val	98.1 ± 0.3	64.4 ± 0.3	72.5 ± 0.5	77.7 ± 1.3	78.1 ± 0.5
DiWA runs	Ensemble*	Uniform	98.5 ± 0.1	64.9 ± 0.1	73.4 ± 0.4	77.2 ± 0.4	78.5 ± 0.1
DiWA runs	Model soups	Uniform	98.8 ± 0.1	62.8 ± 0.2	73.9 ± 0.3	78.3 ± 0.1	78.4 ± 0.2
DiWA runs	Model soups	Greedy	98.4 ± 0.0	64.1 ± 0.2	73.3 ± 0.4	78.1 ± 0.8	78.5 ± 0.1
DiWA runs	Model soups†	Uniform†	98.9	62.4	73.9	78.9	78.6
Our runs	Inter-training (Phang et al., 2018)	ID val	98.2 ± 0.0	63.8 ± 0.5	72.3 ± 0.5	76.6 ± 0.2	77.7 ± 0.0
Our runs	Ensemble* of inter-training	Uniform	98.9 ± 0.1	64.7 ± 0.4	73.8 ± 0.5	78.6 ± 0.2	79.0 ± 0.2
Our runs	Fusing (Choshen et al., 2022b)	ID val	98.4 ± 0.4	64.8 ± 1.2	72.2 ± 0.9	78.5 ± 0.6	78.5 ± 0.8
Our runs	Model ratatouille	Uniform	$\mathbf{99.3} \pm 0.0$	60.8 ± 0.3	74.3 ± 0.3	$\mathbf{79.5} \pm 0.3$	78.5 ± 0.1
Our runs	Model ratatouille	Greedy	99.0 ± 0.0	62.4 ± 0.5	73.8 ± 0.3	$\mathbf{79.5} \pm 0.1$	78.7 ± 0.2
Our runs	Model ratatouille†	Uniform†	**99.3**	60.4	73.9	**79.5**	78.3

Table 6. Accuracy ($\%, \uparrow$) on OfficeHome (best in **bold** and second underlined).

	Algorithm	Selection	Art	Clipart	Product	Photo	Avg
	Vanilla fine-tuning	ID val	61.3 ± 0.7	52.4 ± 0.3	75.8 ± 0.1	76.6 ± 0.3	66.5 ± 0.3
	CORAL (Sun et al., 2016)	ID val	65.3 ± 0.4	54.4 ± 0.5	76.5 ± 0.1	78.4 ± 0.5	68.7 ± 0.3
	SWAD (Cha et al., 2021)	Loss-aware trajectory	66.1 ± 0.4	57.7 ± 0.4	78.4 ± 0.1	80.2 ± 0.2	70.6 ± 0.2
	MA (Arpit et al., 2021)	Uniform trajectory	66.7 ± 0.5	57.1 ± 0.1	78.6 ± 0.1	80.0 ± 0.0	70.6 ± 0.1
	Deep ensembles* (Arpit et al., 2021)	Uniform	65.6	58.5	78.7	80.5	70.8
DiWA runs	Vanilla fine-tuning	ID val	63.9 ± 1.2	54.8 ± 0.6	78.7 ± 0.1	80.4 ± 0.2	69.4 ± 0.2
	Ensemble*	Uniform	67.0 ± 0.1	57.9 ± 0.4	80.0 ± 0.2	81.7 ± 0.3	71.7 ± 0.1
	Model soups	Uniform	68.4 ± 0.2	58.2 ± 0.5	80.0 ± 0.1	81.7 ± 0.3	72.1 ± 0.2
	Model soups	Greedy	67.8 ± 0.5	57.2 ± 0.5	79.6 ± 0.1	81.4 ± 0.4	71.5 ± 0.2
	Model soups†	Uniform†	69.2	59.0	**80.6**	$\underline{82.2}$	72.8
Our runs	Inter-training (Phang et al., 2018)	ID val	65.3 ± 0.3	55.8 ± 2.2	78.6 ± 0.1	80.1 ± 0.2	69.9 ± 0.6
	Ensemble* of inter-training	Uniform	67.8 ± 0.1	60.5 ± 0.1	80.5 ± 0.2	82.0 ± 0.2	72.7 ± 0.1
	Fusing (Choshen et al., 2022b)	ID val	66.4 ± 0.5	59.8 ± 1.2	78.8 ± 0.2	81.0 ± 0.3	71.5 ± 0.5
	Model ratatouille	Uniform	69.8 ± 0.1	60.3 ± 0.2	80.4 ± 0.1	81.8 ± 0.2	73.1 ± 0.1
	Model ratatouille	Greedy	$\underline{70.0} \pm 0.2$	$\mathbf{60.8} \pm 1.0$	$\mathbf{80.6} \pm 0.1$	82.0 ± 0.2	$\underline{73.4} \pm 0.3$
	Model ratatouille†	Uniform†	**70.4**	$\underline{60.7}$	**80.6**	**82.3**	**73.5**

Table 7. Accuracy ($\%, \uparrow$) on TerraIncognita (best in **bold** and second underlined).

	Algorithm	Selection	L100	L38	L43	L46	Avg
	Vanilla fine-tuning	ID val	49.8 ± 4.4	42.1 ± 1.4	56.9 ± 1.8	35.7 ± 3.9	46.1 ± 1.8
	CORAL (Sun et al., 2016)	ID val	51.6 ± 2.4	42.2 ± 1.0	57.0 ± 1.0	39.8 ± 2.9	47.6 ± 1.0
	SWAD (Cha et al., 2021)	Loss-aware trajectory	55.4 ± 0.0	44.9 ± 1.1	59.7 ± 0.4	39.9 ± 0.2	50.0 ± 0.3
	MA (Arpit et al., 2021)	Uniform trajectory	54.9 ± 0.4	45.5 ± 0.6	60.1 ± 1.5	40.5 ± 0.4	50.3 ± 0.5
	Deep ensembles* (Arpit et al., 2021)	Uniform	53.0	42.6	60.5	40.8	49.2
DiWA runs	Vanilla fine-tuning	ID val	$\mathbf{59.9} \pm 4.2$	46.9 ± 0.9	54.6 ± 0.3	40.1 ± 2.2	50.4 ± 1.8
	Ensemble*	Uniform	55.6 ± 1.4	45.4 ± 0.4	$\mathbf{61.0} \pm 0.4$	$\mathbf{41.3} \pm 0.3$	50.8 ± 0.5
	Model soups	Uniform	56.3 ± 1.9	49.4 ± 0.7	59.9 ± 0.4	39.8 ± 0.5	51.4 ± 0.6
	Model soups	Greedy	$\underline{58.5} \pm 2.2$	48.2 ± 0.3	58.5 ± 0.3	41.1 ± 1.2	51.6 ± 0.9
	Model soups†	Uniform†	57.2	$\underline{50.1}$	60.3	39.8	$\underline{51.9}$
Our runs	Inter-training (Phang et al., 2018)	ID val	49.9 ± 1.7	44.3 ± 1.6	54.7 ± 0.4	37.9 ± 1.1	46.7 ± 0.1
	Ensemble* of inter-training	Uniform	58.1 ± 0.2	43.8 ± 0.4	$\mathbf{61.0} \pm 0.2$	$\mathbf{41.3} \pm 0.4$	51.1 ± 0.3
	Fusing (Choshen et al., 2022b)	ID val	52.8 ± 3.2	43.2 ± 2.3	55.2 ± 1.3	35.5 ± 0.3	46.7 ± 1.8
	Model ratatouille	Uniform	57.9 ± 0.2	$\underline{50.1} \pm 0.7$	59.8 ± 0.1	38.9 ± 0.5	51.8 ± 0.4
	Model ratatouille	Greedy	54.0 ± 2.0	47.7 ± 0.8	57.3 ± 0.8	37.9 ± 1.2	49.2 ± 0.9
	Model ratatouille†	Uniform†	57.9	**50.6**	60.2	39.2	**52.0**

Table 8. Accuracy ($\%, \uparrow$) on DomainNet (best in **bold** and second underlined).

	Algorithm	Selection	Clipart	Info	Painting	QuickDraw	Photo	Sketch	Avg
	Vanilla fine-tuning	ID val	58.1 ± 0.3	18.8 ± 0.3	46.7 ± 0.3	12.2 ± 0.4	59.6 ± 0.1	49.8 ± 0.4	40.9 ± 0.1
	CORAL (Sun et al., 2016)	ID val	59.2 ± 0.1	19.7 ± 0.2	46.6 ± 0.3	13.4 ± 0.4	59.8 ± 0.2	50.1 ± 0.6	41.5 ± 0.1
	SWAD (Cha et al., 2021)	Loss-aware trajectory	66.0 ± 0.1	22.4 ± 0.3	53.5 ± 0.1	16.1 ± 0.2	65.8 ± 0.4	55.5 ± 0.3	46.5 ± 0.1
	MA (Arpit et al., 2021)	Uniform trajectory	64.4 ± 0.3	22.4 ± 0.2	53.4 ± 0.3	15.4 ± 0.1	64.7 ± 0.2	55.5 ± 0.1	46.0 ± 0.1
	Deep ensembles* (Arpit et al., 2021)	Uniform	68.3	23.1	54.5	16.3	66.9	**57.0**	**47.7**
DiWA runs	Vanilla fine-tuning	ID val	63.4 ± 0.2	21.1 ± 0.4	50.7 ± 0.3	13.5 ± 0.4	64.8 ± 0.4	52.4 ± 0.1	44.3 ± 0.2
	Ensemble*	Uniform	$\underline{66.7} \pm 0.4$	22.2 ± 0.1	54.1 ± 0.2	15.1 ± 0.2	68.4 ± 0.1	55.7 ± 0.2	47.0 ± 0.2
	Model soups	Uniform	65.9 ± 0.4	23.0 ± 0.2	55.0 ± 0.3	16.1 ± 0.2	68.4 ± 0.1	55.7 ± 0.4	47.4 ± 0.2
	Model soups	Greedy	$\underline{66.7} \pm 0.2$	$\mathbf{23.3} \pm 0.2$	55.3 ± 0.1	16.3 ± 0.2	68.2 ± 0.0	$\underline{56.2} \pm 0.1$	$\mathbf{47.7} \pm 0.1$
	Model soups†	Uniform†	66.2	**23.3**	$\underline{55.4}$	16.5	**68.7**	56.0	**47.7**
Our runs	Inter-training (Phang et al., 2018)	ID val	63.5 ± 0.1	21.1 ± 0.1	51.2 ± 0.2	14.2 ± 0.2	64.7 ± 0.3	52.1 ± 0.1	44.5 ± 0.1
	Ensemble* of inter-training	Uniform	$\mathbf{66.8} \pm 0.2$	22.3 ± 0.0	54.2 ± 0.2	15.4 ± 0.2	68.3 ± 0.0	55.8 ± 0.2	47.2 ± 0.1
	Fusing (Choshen et al., 2022b)	ID val	63.6 ± 0.1	21.3 ± 0.1	51.4 ± 0.2	14.0 ± 0.2	64.1 ± 0.4	52.1 ± 0.3	44.4 ± 0.2
	Model ratatouille	Uniform	65.9 ± 0.2	23.0 ± 0.1	55.1 ± 0.0	16.5 ± 0.1	68.3 ± 0.0	55.8 ± 0.0	47.5 ± 0.1
	Model ratatouille	Greedy	66.5 ± 0.1	23.2 ± 0.1	55.3 ± 0.0	$\mathbf{16.7} \pm 0.1$	68.0 ± 0.0	56.0 ± 0.0	$\mathbf{47.7} \pm 0.0$
	Model ratatouille†	Uniform†	66.1	23.1	**55.5**	**16.7**	$\underline{68.5}$	56.0	**47.7**

F.3. Additional experiments

F.3.1. IMPROVED TERRAINCOGNITA

Ratatouille has significant gains over model soups on some datasets. Yet, the gains are indeed moderate on DomainNet (47.4% to 47.5% with uniform selection) and TerraIncognita (51.4% to 51.8%). In particular for TerraIncognita, this small gain is because other tasks from DomainBed are distant from photos of animals in the wild and even detrimental, explaining the very low performances of inter-trainings (46.7% versus 50.4% for ERM); ratatouille manages to fill the gap by increased diversity across fine-tunings. This is an evidence of ratatouille's robustness to the choice of auxiliary tasks. To validate that more similar auxiliary tasks can help on TerraIncognita, we run an additional experiment with iWildCam (Beery et al., 2021) as a (similar) auxiliary task: as detailed in Table 9, we reach 52.9% averaged accuracy.

Table 9. Accuracies (%, ↑) on TerraIncongita with uniform selection.

Algorithm	Auxiliary datasets	L100	L38	L43	L46	Avg
Soups	✗	56.3	49.4	59.9	39.8	51.4
Ratatouille	DomainBed's	57.9	50.1	59.8	38.9	51.8
Ratatouille	iWildCam	**59.8**	**50.3**	**60.0**	**41.4**	**52.9**

F.3.2. CAMELYON

We conduct some experiments on the Camelyon (Koh et al., 2021) dataset from the WILDS (Koh et al., 2021) benchmark, where the task is to classify "breast cancer metastases in whole-slide images of histological lymph node sections", with each hospital successively considered as the test while others are for training. The results in Table 10 show that model ratatouille consistently beats model soups on Camelyon for histopathology. These results may facilitate the adoption of ratatouille in the medical community (Maron et al., 2022).

Table 10. Accuracies (%, ↑) on Camelyon.

Selection	Algorithm	Hospital 1	Hospital 2	Hospital 3	Hospital 4	Hospital 5	Avg
Uniform	Soups	96.4	94.3	96.1	94.2	90.4	94.3
	Ratatouille	97.1	94.4	96.1	94.8	90.5	94.6
Greedy	Soups	97.4	95.1	96.5	96.1	90.6	95.1
	Ratatouille	97.5	95.3	96.7	96.6	90.8	95.4