Diverse and efficient ensembling of deep networks
Alexandre Rame

To cite this version:

HAL Id: tel-04485963
https://theses.hal.science/tel-04485963
Submitted on 1 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ
Spécialité Informatique
École Doctorale Informatique, Télécommunications et Électronique (Paris)

Diverse and Efficient Ensembling of Deep Networks
De la diversification et combinaison efficace des réseaux profonds

Présentée par
Alexandre Ramé

Dirigée par
Matthieu Cord

Pour obtenir le grade de
DOCTEUR de SORBONNE UNIVERSITÉ

Présentée et soutenue publiquement le 11 Octobre 2023

Devant le jury composé de :

Graham Taylor
Professor, University of Guelph. Research Director, Vector Institute.
Rapporteur

Christian Wolf
Principal Scientist, Naver Labs.
Rapporteur

Cordelia Schmid
Research Director, INRIA. Research Scientist, GoogleAI.
Examinatrice

Léon Bottou
Principal Scientist, MetaAI.
Examinateur

Thomas Wolf
Co-founder, HuggingFace.
Examinateur

Patrick Gallinari
Professeur, Sorbonne Université. Senior Research Scientist, Criteo.
Président du jury

Matthieu Cord
Professeur, Sorbonne Université. Principal Scientist, valeo.ai.
Directeur de thèse
ABSTRACT

This thesis aims at enhancing the generalization abilities of deep neural networks, a critical step towards fair and reliable artificial intelligence. Specifically, we address the drop in performance when models are evaluated on test samples with a distribution shift with respect to the train samples.

To this end, we focus on ensembling strategies: indeed, combining multiple models is a standard, simple yet potent strategy to improve robustness. After an overview of the relevant literature, we provide a new explanation of ensembling’s success under distribution shifts, especially when the members of the ensemble are diverse.

To foster such diversity within members, we investigate several strategies. The initial one, DICE, introduces an explicit regularization to eliminate redundant information across members. Subsequent diversity methods in this thesis are implicit, relying on diverse data augmentation (in MixMo), diverse hyperparameters (in DiWA), inter-training on auxiliary datasets (in ratatouille), and diverse objectives (in rewarded soups).

The second primary challenge addressed in this thesis is the enhancement of ensemble efficiency, and aims at lessening the computational burden of combining multiple models; indeed, when considering two members, the standard ensembling by averaging of predictions doubles the computational cost, thus impeding scalability. After exploring subnetwork ensembling (in MixMo), we introduce a significant contribution of this thesis; the observed ability to average the models in weights rather than in predictions. This finding was surprising due to the non-linearities in deep architectures. We empirically demonstrate that, when weights are fine-tuned from a shared pre-trained initialization, weight averaging succeeds by approximating ensembling without any inference overhead. The empirical gains are especially important on DomainBed, the reference benchmark evaluating out-of-distribution generalization. More broadly, weight averaging facilitates effortless parallelization, enhancing machine learning updatability and data privacy.

Finally, this thesis explores how ensembling can facilitate the alignment of models. This is critical to mitigate the societal ethical concerns from recent rapid scale-up in deep learning. To this end, we propose rewarded soups, a new strategy for multi-objective reinforcement learning, paving the way towards more transparent and reliable artificial intelligences, aligned with the world in all its diversity.
RÉSUMÉ

Cette thèse vise à améliorer les capacités de généralisation des réseaux de neurones profonds, un enjeu essentiel pour le développement de systèmes d’intelligence artificielle à la fois fiables et équitables. Le cœur du défi réside dans la gestion des potentiels changements de distributions entre les données d’entraînement et celles de test pour l’évaluation, pouvant réduire les performances.

Dans cette thèse, nous analysons principalement des stratégies consistant à combiner plusieurs réseaux de neurones. Cette simple méthode d’ensemble est classique mais particulièrement efficace pour améliorer la généralisation. Après avoir examiné la littérature existante, nous proposons une nouvelle explication de la réussite des méthodes d’ensemble hors-distribution, en particulier lorsque les différents membres de l’ensemble sont suffisamment divers pour compenser mutuellement leurs erreurs individuelles.

Pour encourager cette diversité entre les membres de l’ensemble, nous explorons plusieurs approches. La première, baptisée DICE, introduit explicitement une régularisation visant à éliminer de potentielles informations redondantes entre les membres de l’ensemble lors de l’apprentissage. Les autres méthodes de diversité utilisées dans cette thèse sont implicites, s’appuyant sur une augmentation diversifiée des données (dans MixMo), le choix d’hyperparamètres variés (dans DiWA), des entraînements intermédiaires sur des jeux de données auxiliaires (dans ratatouille), ou des récompenses différenciées en apprentissage par renforcement (dans rewarded soups).

Le second enjeu majeur de cette thèse concerne l’efficacité des méthodes d’ensemble. L’objectif est d’atténuer le coût computationnel inhérent à la combinaison de plusieurs réseaux ; en effet, considérant deux réseaux, la méthode standard qui consiste à moyennir leurs prédictions multiplie par deux le coût. Après avoir exploré une stratégie d’ensemble de sous-réseaux (dans MixMo), nous décrivons une contribution majeure de cette thèse : l’analyse d’une stratégie consistant à faire la moyenne des poids des réseaux de neurones plutôt que de leurs prédictions. Cette stratégie, surprenante en raison des non-linéarités des architectures profondes, fonctionne empiriquement lorsque les modèles sont appris à partir d’une même initialisation pré-entraînée. Cette moyenne des poids offre les bénéfices de l’ensemble sans aucun coût supplémentaire pour l’évaluation, en particulier sur DomainBed, le benchmark de référence pour évaluer la généralisation hors-distribution. Plus généralement, cette stratégie favorise la parallélisation des apprentissages et l’adaptabilité des modèles.

Pour finir, cette thèse explore comment les méthodes d’ensemble peuvent améliorer l’alignement des intelligences artificielles. Face à l’essor rapide des modèles de langues comme ChatGPT, cet alignement est crucial pour répondre aux nombreuses préoccupations éthiques actuelles. Pour prendre en compte la diversité des préférences humaines, nous proposons une stratégie de politiques multiples en apprentissage par renforcement, rendant plus transparent l’alignement sur le monde, dans toute sa diversité.
REMERCIEMENTS

Je souhaite exprimer ma plus profonde gratitude à toutes les personnes qui m’ont soutenu et guidé lors de ce parcours doctoral, professionnel et personnel.

Je commencerai par remercier sincèrement Matthieu, dont l’accompagnement exceptionnel a marqué ces trois années de recherche. Son optimisme constant, son réalisme avisé, son habile mélange de confiance et de questionnement scientifique, sa rigueur intellectuelle et sa bienveillance, ont contribué à faire de cette thèse une grande expérience réussie. Je tiens également à exprimer ma gratitude au jury de thèse : leur investissement dans la lecture de ce manuscrit et lors de ma soutenance est un grand honneur. Je suis touché par leur intérêt dans mon travail.


Finalement, mes remerciements les plus chaleureux vont à mes proches. Tout d’abord à la femme que j’aime, ma femme tout court, Julie. Son intelligence et son sens aigu du questionnement font de Julie une interlocutrice extraordinaire et une relectrice hors pair. Mais c’est aussi grâce à sa confiance et son soutien indéfectible que j’ai pu continuer à avancer. Je lui suis infiniment reconnaissant pour tout cela. D’ailleurs, n’hésitez pas à lui demander si vous avez besoin d’éclaircissements sur mon travail, elle l’explique parfois aussi bien (voir mieux !) que moi. Je tiens également à souligner la chance que j’ai eue de grandir dans un cadre familial stimulant et bienveillant, favorisant l’épanouissement intellectuel. Pour cela et pour beaucoup plus, un grand merci à mes incroyables parents, Florence et Emmanuel, à mes sœurs, Valentine et Atalante, et mon frère Nicolas. À mes amis proches et témoins de mariage Théo, Thibaud, et Matthieu, à mes amis de Boulogne et de l’X, Rouille pour ses ronronnements, France Culture et Nicolas Martin pour ses podcasts, et à tous ceux et celles avec qui j’ai pu converser, qui m’ont passionné, ou avec qui j’ai simplement passé des moments de qualité : merci !
CONTENTS

ABSTRACT i
RÉSUMÉ iii
REMERCIEMENTS v
CONTENTS vii

1 INTRODUCTION 1
  1.1 Artificial intelligence: a historical perspective 1
  1.2 Deep learning 2
  1.3 Out-of-distribution generalization 3
  1.4 Ensembling and contributions 4

2 ENSEMBLING 9
  2.1 Context: the fundamentals of deep learning 9
  2.2 A new bias-variance theoretical understanding of distribution shifts 16
  2.3 Diverse and efficient ensembling 19
  2.4 Conclusion 24

3 DICE 27
  3.1 Introduction 27
  3.2 DICE 29
  3.3 Experiments 34
  3.4 Conclusion 38

4 DIVERSE WEIGHT AVERAGING 39
  4.1 Introduction 39
  4.2 Context 40
  4.3 Bias-variance-covariance-locality analysis 42
  4.4 DiWA 45
  4.5 Related work 47
  4.6 Experiments 47
  4.7 Conclusion 52

5 RATATOUILLE 53
  5.1 Introduction 53
  5.2 Context 54
  5.3 Model ratatouille 56
  5.4 Experiments 58
  5.5 Conclusion 63

6 REWARDED SOUPS 65
  6.1 Introduction 65
  6.2 Rewarded soups 67
  6.3 Experiments 71
  6.4 Discussion and related work 77
  6.5 Conclusion 78
# CONTENTS

7 CONCLUSION .......................................................... 79
  7.1 Contributions .................................................. 79
  7.2 Future work ................................................... 80

BIBLIOGRAPHY .......................................................... 87

A ACRONYMS AND NOTATIONS ........................................... 129
  A.1 Acronyms ....................................................... 129
  A.2 Notations ....................................................... 131

B SOCIETAL IMPACTS .................................................. 133

C PROOFS .................................................................. 135
  C.1 Proofs for ENSEMBLING ....................................... 135
  C.2 Theoretical insights for DICE ................................. 143
  C.3 Proof for DIVERSE WEIGHT AVERAGING .................. 145
  C.4 Proofs for REWARDED SOUPS ................................. 146

D MIXMO ................................................................. 151
  D.1 Introduction ....................................................... 151
  D.2 Related work .................................................... 153
  D.3 MixMo ........................................................... 154
  D.4 Experiments .................................................... 156
  D.5 Conclusion ....................................................... 161

E FISHR ................................................................. 163
  E.1 Introduction ....................................................... 163
  E.2 Related work .................................................... 165
  E.3 Fishr ............................................................... 166
  E.4 Experiments ..................................................... 171
  E.5 Conclusion ....................................................... 173

LIST OF FIGURES ..................................................... 175

LIST OF TABLES ....................................................... 183
INTRODUCTION

L’union fait la force

Homère, L’Iliade

1.1 Artificial intelligence: a historical perspective

The concept of artificial intelligence (AI) machines traces back to the previous century, beginning as a subject of science fiction, with Karel Čapek first coining the term “robot” in 1920 and Isaac Asimov exploring the concept in short stories from the 1940s. The notion then evolved into a tangible reality, from Turing’s machine-breaking work during World War II, the seminal 1956 Dartmouth Workshop to the triumph of IBM’s Deep Blue in 1997 [Cam+02]. Initially, the community was optimistic about achieving AI within a few years.

A successful AI revolution could usher a new era of enhanced intellectual capabilities, much like the industrial revolution enhanced human physical capabilities. The potential applications are wide-ranging, from education, healthcare, personal assistants, autonomous vehicles to AI-driven scientific research for addressing global challenges like climate change. Optimistic visionaries anticipate a world of material and cultural abundance, universal income, and freedom from some labor, enabling humans to focus on selected/creative/interesting tasks.

Despite early optimism, traditional expert systems encountered numerous setbacks, leading to periods known as “AI winters” filled with disillusionment and skepticism. In response, machine learning (ML) [Bis06] proposed a data-centric paradigm shift. The key element in ML is the statistical algorithm that learns automatically through experience and data. More specifically, models are learned for a target task using some training data, intending to achieve good accuracy (i.e., generalize well) on new samples not seen during training. While early and simple ML models achieved some success in tasks like spam filtering, genome-wide association studies, movie recommendation [Tös+09], and medical diagnosis, they often fell short in large-scale real-world applications.
1.2 Deep learning

The prominence of AI in today’s society is primarily caused by the success of deep learning (DL) [Goo+16], a specialized subfield of ML involving deep neural networks (DNNs). These models were biologically inspired by the human brain, yet actually are simply a collection of nested (neuron) functions, each of which is a linear transformation usually followed by a non-linear activation function. The term deep refers to the large number of layers stacked successively. DL became the dominant approach in computer vision (CV) after the success in 2012 of AlexNet [Kri+12] at the ILSVC competition [Rus+15] on ImageNet, a large dataset made of 1.28M training images and 1000 classes.

Then, DL powered breakthroughs to problems that seemed unsolvable by traditional ML, some of which are depicted in Figure 1.1. Stable Diffusion [Rom+22] can generate realistic photographs of the pope in a down puffer coat. ChatGPT [Ope23] could pass the bar exam and write 40 per cent of the code for a software engineer. Beyond these famous applications, DL is the core paradigm behind current high-precision machine translation [Vas+17], autonomous driving systems [Sun+20], accelerating physics simulations [Bre+20], predicting protein folding structures [Jum+21], estimating molecular toxicity [Adv+14], optimizing data center cooling systems [Eva+16], and managing magnetic coils in nuclear fusion reactors [Deg+22], etc, to name a few.

(a) Alphago.  (b) Stable Diffusion.  (c) Canopy.  (d) ChatGPT.

Figure 1.1. – A glimpse into the transformative and successful applications of DL. In Figure 1.1(a), AlphaGo [Sil+16] surpasses human performance in the strategic board game of Go. In Figure 1.1(b), Stable Diffusion [Rom+22] is prompted to generate “a photograph of an astronaut riding a horse”, illustrating how AIs can imitate human creativity. In Figure 1.1(c), sub-meter resolution canopy height maps are generated from aerial and GEDI lidar data [Tol+23], offering climate scientists a powerful tool to understand deforestation. In Figure 1.1(d), the now famous ChatGPT [Ope23] can natively discuss with humans and augment their intelligence.

Scaling. The cornerstone of the DL revolution is scalability: results have consistently improved when increasing the number of parameters [Kap+20] or of training data [Hof+22], without hitting any apparent limitations. Such rapid progress was enabled by engineering advances in graphics processing units (GPUs) hardware adhering to Moore’s law, the advent of cloud computing, more efficient algorithms, and the emergence of frameworks like PyTorch [Pas+19] or platforms like HuggingFace [Wol+20]. The field has
benefited from substantial investments from major tech companies and from numerous collaborations fostered by the prevalent open-source philosophy. Over the past decade, the computational power used for training AI models has surged by a factor of one hundred million, while the cost has significantly diminished. We have transitioned from training models on relatively small datasets to feeding them with the entire internet’s wealth of information. This scaling up predictably enhances results across a broad spectrum of tasks, but also induced unpredicted emergent abilities [Wei+22b] surfacing abruptly in (very) large models: examples include in-context learning, or multi-step reasoning [Wei+22c; Lam+22]. These emergent properties underscore the success of ChatGPT and further drive the scaling trend. They also raise questions about the potential capabilities and risks of LLMs, should we continue to scale them up.

Unified framework. Another paradigm that has significantly contributed to the success of DL is the unification of modalities and tasks under a single & simple framework. For example, the networks’ architectures are now unified; though historically computer vision used convolutional [Fuk80; Kri+12] and natural language processing recurrent networks [Rum+85], most state-of-the-art are now with Transformers [Vas+17; Dos+21]. This architecture was first designed for machine translation. Moreover, foundation models [Bom+21] have emerged as the standard unified paradigm to learn DNN’s weights: rather than trained from scratch on the target task, weights are now first pre-trained through self-supervision on vast corpus of data and then fine-tuned [Oqu+14; Yos+14] on the target task. The different trainings usually leverage the same tools and “tricks”, such as the Adam [Kin+15] optimizer and general-purpose regularizations (dropout [Gal+16] or weight decay [Kro+91]). This unifying paradigm enables the transfer of knowledge and findings across DL setups.

1.3 Out-of-distribution generalization

The increasing importance of AI in our lives and society demands a close examination of potential failure cases. As large DNNs can actually memorize their training dataset [Zha+17] by latching onto specific patterns, they may struggle to generalize on new test samples. The generalization gap [Kaw+17] between train and test performance is a well-known phenomenon in ML usually named overfitting. Critically, this generalization gap worsens under distribution shifts [Arj+19; Gul+21], when the test distribution differs from the training distribution. These failures for out-of-distribution (OOD) generalization is a major limitation of DL, which can negatively impact human lives in several real-world contexts. We name a few below.

• Uncontrolled deployment: although training is centralized, the model is used in the wild for diverse scenarios where the input distribution can change unexpectedly.
• Constrained budget, causing training datasets of limited size without representing all possible domains. For example, it would be impractical to train an autonomous vehicle’s object detection system on all possible weather conditions and in all cities.
• Dynamic scenarios, with a temporal shift between train and test. For example, when analyzing images from social media, where trends can change rapidly.

• Subpopulation shifts, as some (minority) groups may be less represented in the training dataset. This highlights a critical connection between OOD generalization and fairness. Indeed, errors induced by distribution shifts can lead to severe ethical issues: examples include facial recognition systems [Gro+19], loan approvals [Ang+21], or medical diagnoses [Lar+20]. Not only might DNNs replicate the biases found in the data—they confuse correlation and causation—but they also exhibit a simplicity bias [Sha+20]: DNNs tend to rely on the most straightforward features, potentially overlooking fairer explanations.

More broadly, the ability of a model to effectively handle OOD scenarios is a key indicator of its intelligence. Our cognitive abilities as humans allows us to create beyond our previous experiences, to solve new problems, and to find solutions when facing novel tasks in unseen situations. Therefore, OOD generalization appears as a necessary milestone in the journey towards more general systems.

The imperative of alignment. Assuming we continue to scale the architectures and improve DL performances in the future, we may then be dealing with potential new risks. Such powerful models could arm malicious users with unprecedented power, fuel widespread censorship or manipulation by totalitarian private entities or governments, or, more speculatively, seize global control [Hen+22; Hen23], leading to catastrophic outcomes. It appears necessary to ensure that the models remain aligned [Tay+16; Ken+21; Ngo+22] with core values. Recently, reinforcement learning from human feedback (RLHF) [Chr+17b] emerged as the leading alignment paradigm to fine-tune large language models (LLMs). Specifically, given a proxy reward approximating human preferences, a LLM optimized on this proxy reward is expected to make decisions for new inputs aligned with those preferences. Thus, the LLM needs to handle a wide range of OOD inputs, emphasizing that alignment is fundamentally an OOD challenge, as further detailed in Section 7.2.5.3. Given the importance of alignment for AI safety, this strengthens the need for developing DL models that can effectively deal with unanticipated inputs.

1.4 Ensembling and contributions

This thesis seeks to improve the generalization capabilities of deep neural networks. To this end, we turn to a traditional, straightforward, and highly practical strategy: ensembling. In essence, instead of relying on a single model, we combine multiple models. From a scaling perspective [Chi+20], rather than increasing the number of parameters inside one model, we scale the number of models.

The most common ensembling (ENS) strategy involves (i) training multiple models independently, (ii) passing the test input to each model, and (iii) averaging their predictions. Such functional ensembling is a very natural strategy, used for decades in ML [Nil65;
The first key concept in ensembling is diversity. Indeed, ensembling succeeds if it reduces the variance of the predictions; this happens if its members are diverse and complementarity, meaning that they have different failure cases, allowing them to compensate for each other’s errors. The other key concept in this thesis is efficiency. Indeed, the naive form of functional ensembling is costly as it requires multiple trainings and inferences, limiting its applicability in real-world scenarios.

The core challenge lies in the intricate trade-off between efficiency and diversity. In particular, traditional strategies to improve one often hurts the other. For example, let’s consider a baseline weight sharing strategy [Lee+15] that shares weights across models; although this may improve efficiency, this would unfortunately homogenize the models and thus reduce diversity. As another example, consider members that predict randomly: these members may be highly diverse, but their ensemble would perform badly. This (naive) example highlights the importance of another criterion for accurate ensembling, members’ accuracies, in tension with diversity and efficiency.

In other words, the principal challenge of our thesis can be summarized as follows:

How to best trade off between efficiency, diversity, and members’ accuracies in ensembling?

The contributions of this thesis aim at proposing different solutions to this question, as presented in the subsequent chapters.

• Chapter 2: ensembling
  We first review the literature on ensembling, and propose a new error bound for ensembling strategies under distribution shifts. This explains why and when ensembling particularly excels, and highlights that diversity becomes even more critical under distribution shifts.

• Chapter 3: DICE [Ram+21a]
  We introduce a new regularization named DICE that increases diversity by removing irrelevant redundant information across members of the ensemble.

• Chapter 4: diverse weight averaging [Ram+22b]
  We present the main contribution of this thesis for efficient ensembling: instead of averaging the predictions, we propose averaging the weights of the models. We show that this weight averaging (WA) is possible (despite non-linearities in the architectures) when models are fine-tuned from a shared pre-trained initialization.

• Chapter 5: ratatouille [Ram+23a]
  In this chapter, we demonstrate that the conditions to apply WA can be relaxed. The proposed ratatouille averages the weights of models fine-tuned from different initializations, inter-trained on different tasks; these inter-trainings enhance diversity across initializations and thus improve final performance for OOD generalization.

• Chapter 6: rewarded soups [Ram+23b]
  We explore how WA can help to manage the diversity of human opinions when aligning large language models with reinforcement learning from human feedback.
In the introduction, we detail other related works published during this thesis.

- **Appendix D: MixMo [Ram+21b]**
  This chapter details our first attempt toward efficient ensembling, and proposes fitting multiple subnetworks within a single base model through a multi-input multi-output strategy named MixMo. Although successful, we move this work to the Appendix as it does not fit within the foundation model paradigm.

- **Appendix E: Fishr [Ram+22a]**
  This chapter considers a key limitation of ensembling strategies; their inability to tackle spurious correlations. The proposed Fishr regularization promotes invariance across training domains. We move this work to the Appendix as it does not involve ensembling.

In details, this thesis is based on the following papers, sorted in chronological order:


Additionally, I contributed to other projects listed below; though they consider ensembling strategies and refine our understanding on important questions, they are not further investigated in this thesis for the sake of brevity.


Except Fishr, all those works study ensembling, where diversity and efficiency are systematically key contributions. This is highlighted in Table 1.1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Conference</th>
<th>Chapter</th>
<th>Diversity strategy</th>
<th>Efficiency strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICE [Ram+21a]</td>
<td>ICLR 2021</td>
<td>Chapter 3</td>
<td>Diversity regularization</td>
<td>Weight sharing</td>
</tr>
<tr>
<td>MixMo [Ram+21b]</td>
<td>ICCV 2021</td>
<td>Appendix D</td>
<td>Data augmentation</td>
<td>Subnetworks</td>
</tr>
<tr>
<td>MixShare [Sun+22]</td>
<td>CVPR Workshop 2022</td>
<td></td>
<td>Data augmentation + unmixing</td>
<td>Subnetworks</td>
</tr>
<tr>
<td>DyTox [Dou+22]</td>
<td>CVPR 2022</td>
<td>Appendix E</td>
<td>Different target tasks</td>
<td>Weight sharing</td>
</tr>
<tr>
<td>Fishr [Ram+22a]</td>
<td>ICML 2022</td>
<td>Appendix E</td>
<td>Hyperparameters and data orders</td>
<td>Weight averaging</td>
</tr>
<tr>
<td>DiWA [Ram+22b]</td>
<td>NeurIPS 2022</td>
<td>Chapters 2 and 4</td>
<td>Diverse auxiliary tasks</td>
<td>Weight averaging</td>
</tr>
<tr>
<td>Interpolate [Ram+22c]</td>
<td>NeurIPS Workshop 2022</td>
<td></td>
<td>Diverse reward</td>
<td>Weight averaging</td>
</tr>
<tr>
<td>Ratatouille [Ram+23a]</td>
<td>ICML 2023</td>
<td>Chapter 5</td>
<td>Diverse tasks and modalities</td>
<td>Weight averaging</td>
</tr>
<tr>
<td>Rewarded soups [Ram+23b]</td>
<td>NeurIPS 2023</td>
<td>Chapter 6</td>
<td>Diverse tasks and modalities</td>
<td>Weight averaging</td>
</tr>
<tr>
<td>UniVAL [Shu+23]</td>
<td>Under submission</td>
<td></td>
<td>Diverse tasks and modalities</td>
<td>Weight averaging</td>
</tr>
</tbody>
</table>

As a final note, I also contributed to the organization of the PRINCE out-of-distribution generalization challenge: Eustache Diemert, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Alexandre Ramé, and Ugo Tanielian. “PRINCE: PRomoting INvariance for Classification of browsing journeys across Environments”. In: ECML PKDD (2022). The results of the competition are available: https://codalab.lisn.upsaclay.fr/competitions/3353.
THE UNREASONABLE EFFECTIVENESS OF ENSEMBLING FOR OUT-OF-DISTRIBUTION GENERALIZATION

Introduction

First, we shed light on the related literature in machine and deep learning: we recall the out-of-distribution (OOD) generalization challenge, the bias-variance decomposition of the error, and how ensembling reduces variance.

Then, we demonstrate that variance is intrinsically related to shifts in input distributions (i.e., diversity shifts), while bias is inherently related to shifts in output conditional distributions (i.e., correlation shifts). These theoretical contributions refine our understanding of OOD generalization, and in particular explain why the functional ensembling of diverse members excels under diversity shifts. These insights were first published in the theoretical section from: Alexandre Ramé, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. “Diverse Weight Averaging for Out-of-Distribution Generalization”. In: NeurIPS. 2022.

Overall, we motivate our goal of diverse and efficient ensembling, and discuss how these two research directions have been previously addressed in the literature, the underlying challenges, trade-offs, and limitations of existing strategies.

2.1 Context: the fundamentals of deep learning

We first briefly explain how deep neural networks are trained in deep learning (DL) in a supervised setting, which is the key focus of this thesis.

Remark 2.1. This thesis will primarily focus on classification among a predefined number of classes in computer vision (CV), a complex task due to the unstructured nature of images. However, thanks to the unification of architectures and training paradigms across modalities and tasks, all our findings could potentially be extended to various other setups. This will be done in our last Chapter 6, where we consider ensembling of models trained with reinforcement learning [Rus+16] in a wide range of setups: for example in text-to-text tasks, but also in multimodal tasks such as image-to-text captioning [Ren+17], visual question answering [Ben+17], and text-to-image generation with diffusion models [Rom+22].
The name of the key stages in the deep learning process.

Visualization of each stage.

Data representation of each stage.

- Figure 2.1 - The fundamental concepts in deep learning. The first row represents the names and notations used along this thesis, the second row illustrates the different concepts, while the third row depicts the corresponding mathematical objects. Specifically, an image $x$ represented as a RGB matrix is given as input to a black-box featurizer $\Phi$, that extracts an embedding vector $z$, representing in a structured way the different information in the image. This embedding is then fed to a (usually linear) classifier $w$, whose goal is to separate the different classes $Y$; specifically, $w$ predicts a probability distribution $\hat{y}$ that should be close to the true label $y$.

### 2.1.1 How to build a deep neural network

The key fundamental elements in a DNN are depicted in Figure 2.1. We analyze an input $x \in \mathcal{X}$, which in the context of CV typically refers to an image represented as a matrix of red green blue (RGB) pixels. The DNN $f$ maps $x$ to a prediction $\hat{y} = f(x, \theta)$ in the label space $\mathcal{Y}$ of size $K$. Specifically, $f (\cdot, \theta) : \mathcal{X} \rightarrow \mathcal{Y}$ is usually decomposed into a featurizer $\Phi$ parameterized by the weights $\phi$, onto which we plug a dense linear classifier $w$ parameterized by the weights $\omega$; hence, $\theta = (\omega, \phi)$. The featurizer transforms a non-structured input into a feature embedding (denoted $z$) in a space where the $K$ classes should ideally be linearly separated. These embeddings can then be processed arithmetically. For instance, the embeddings of two images containing the same object will usually exhibit a high cosine similarity (this would not hold if the similarity were computed in pixels). The classifier aims at detecting class boundaries in this embedding space, to allocate to each potential class a probability, all of which sum up to 1. The class with the highest predicted probability is chosen as the class prediction.

A significant portion of the DL literature is devoted to the design of the optimal architecture as a series of linear and non-linear transformations. The main difference between ML and DL is the number of layers in this featurizer. Some fundamental units of DNN architectures include:
• convolutional layers [Fuk80], the primary building block of convolutional neural network (CNN) architectures,

• self-attention layers [Vas+17], the primary building block of Transformer architectures [Vas+17; Dos+21],

• ReLU function [Aga18], a non-linear activation function applied element-wise which only retains positive values,

• normalization layers such as batch normalization (BN) [Iof+15],

• residual connections [He+16], adding direct connections between blocks to mitigate the vanishing gradient problem [Hoc+01].

Crucially, this thesis does not delve into these design choices; we aim to deliver architecture-agnostic insights applicable to any DNN, where understanding or knowledge of the various underlying architectural choices are not required. Consequently, we treat the deep featurizer as a non-linear black box.

2.1.2 How to train a deep neural network

Previous section has described the core components of a DNN \( f ( \cdot , \theta ) \). Yet the key question remains: how to learn the appropriate weights \( \theta \)?

2.1.2.1 Empirical risk minimization

Supervised DL is based on the statistical data-centric learning theory from [Vap92; Vap99]. Given pairs of (input \( x \), label \( y \)), we want to learn a model that from \( x \) can predict \( \hat{y} \) close to \( y \). To measure the distance between a prediction \( \hat{y} = f ( x, \theta ) \) and a class label \( y \), we consider a loss function \( \ell : \mathcal{Y}^2 \to \mathbb{R}_+ \). In classification, the loss is usually the categorical cross-entropy:

\[
\ell ( \hat{y}, y ) = - \sum_{c=1}^{K} y_c \log \hat{y}_c ,
\]  

but we will also consider the mean-squared error (MSE) loss to simplify our proofs.

More specifically, we rely on a training (source) domain, denoted \( S \) with distribution \( p_S \). In theory, we would seek \( \theta \) minimizing the source error on the full training domain \( S \): \( \mathcal{E}_S ( \theta ) = \mathbb{E}_{(x,y) \sim p_S} [\ell ( f ( x, \theta ), y )] \). In practice, we usually have only an empirical dataset \( \mathcal{D}_S = \{ (x_1, y_1), \ldots, (x_{n_S}, y_{n_S}) \} \) of \( n_S \) i.i.d. samples from the underlying training distribution \( p_S (X, Y) \). Then, we seek \( \theta \) minimizing the source empirical error:

\[
\hat{\mathcal{E}}_S ( \theta ) = \sum_{(x,y) \in \mathcal{D}_S} [\ell ( f ( x, \theta ), y )].
\]  

This strategy is called empirical risk minimization (ERM).
Algorithm 2.1 Procedure to optimize a neural network with ERM through SGD.

```plaintext
input: a model \( f \) with trainable weights \( \theta \)
input: a dataset \( \mathcal{D}_S \)
input: a loss function \( \ell \)
input: a learning rate \( \eta \) and a batch size \( b \)

1: while stopping criterion not satisfied do
2: \((x, y) \leftarrow \text{sample mini-batch of size } b \text{ from } \mathcal{D}_S\)
3: Forward pass: \( \hat{y} \leftarrow f(x, \theta) \)
4: Compute loss: \( \hat{E}_S \leftarrow \ell(\hat{y}, y) \)
5: Compute the gradients: \( \delta \leftarrow \nabla_\theta \hat{E}_S \)
6: Update all parameters: \( \theta \leftarrow \theta - \eta \delta \)
7: end while
```

2.1.2.2 Stochastic gradient descent

To optimize this ERM objective, the now standard strategy is stochastic gradient descent (SGD). As summarized in Algorithm 2.1, we iteratively update the weights \( \theta \) in the direction opposite to the gradient of the loss w.r.t. the weights \( \theta \). Specifically, we sample a batch with \( b \) samples from \( \mathcal{D}_S \), feed it into the model \( f(\cdot, \theta) \), and compare the predictions \( \hat{y} \) to the ground-truth labels \( y \) with the loss \( \ell \). Gradient computation is achieved by backpropagation [LeC+99] through the various layers of the architecture. The learning rate \( \eta \) controls the gradient step size.

Assuming proper convergence of SGD, and thanks to the overparameterization of the model and the universal approximation theorem [Cyb89], we expect this training loss to be relatively small; \( f \) will usually fit the training data almost perfectly.

2.1.2.3 Improved training procedures

While the architecture design and scale can significantly impact the model’s performance, the training procedure is also crucial [Wig+21] and can go beyond SGD. For example, the most popular optimization algorithm is actually Adam [Kin+15]. Moreover, data augmentation is also essential, especially in CV where mixed sample data augmentation (MSDA) (such as Mixup [Zha+18] or CutMix [Yun+19]) are widely used.

Notations. As stated earlier in the context of architectural design, the choice of training configuration will not be the primary focus of this thesis. Thus, we simply denote the training configuration as \( c \), encompassing all training choices and sources of randomness in the learning (e.g., initialization, hyperparameters, training stochasticity, data augmentation, epochs, etc.) excluding the fixed dataset \( \mathcal{D}_S \). Then, we denote \( l_S = \{\mathcal{D}_S, c\} \) a learning procedure on domain \( S \). When necessary, we write \( \theta(l_S) \) to refer to the weights obtained after optimization of \( \hat{E}_S(\theta) \) w.r.t. \( \theta \) on \( \mathcal{D}_S \) with configuration \( c \).
2.1.3 Generalization

**ID generalization.** The generalization abilities of ML algorithms are evaluated on a test set, i.e., a set of samples that were not seen during training. Thus, denoting $T$ the (target) test domain with distribution $p_T$, we want $\theta$ to have small test target generalization error: $\mathcal{E}_T(\theta) = \mathbb{E}_{(x,y) \sim p_T}[\ell(f(x, \theta), y)]$. To this end, the learned model $f(\cdot, \theta)$ should ideally capture a robust mechanism, not just memorize all the training data [Zha+17]. The difference between the loss in train and in test is often referred to as the generalization gap [Kaw+17]. In the in-distribution (ID) setup, the train and test distributions are the same, i.e., $p_S(X,Y) = p_T(X,Y)$.

**OOD generalization.** In this thesis, we focus on a more complex setup, named out-of-distribution (OOD) generalization, where $p_T(X,Y) \neq p_S(X,Y)$, i.e., the test (target) distribution is different from the train (source) distribution. As highlighted in Section 1.3, OOD generalization under distribution shifts is critical to ensure applicability in real-world applications, where train and test hardly ever follow the same distributions. To better understand this key challenge, [Ye+22] decomposed distribution shifts into two types, visualized in Figure 2.2 and detailed below:

- **diversity shifts** (a.k.a. covariate shift) when $p_S(X) \neq p_T(X)$, i.e., when the marginal input distributions differ. In this setup, the train and test distributions comprise data from related but distinct domains, for instance on PACS [Li+17] where we have pictures and drawings of the same objects.
- **correlation shifts** (a.k.a. concept shift) when $p_S(Y|X) \neq p_T(Y|X)$, i.e., when the posterior covariate distributions differ. In this setup, the correlation between the input and the label depends on the domain: for instance on ColoredMNIST [Arj+19] where the color is spuriously correlated with the label.
Benchmarks. We primarily evaluate our approaches on standard image classification datasets. The first two papers [Ram+21a; Ram+21b] of this thesis used CIFAR-[10,100] [Kri+09]. Then, two years ago, the DomainBed [Gul+21] benchmark was introduced; it aimed at fairly evaluating the different OOD approaches and became the standard in the community. It includes five real-world multi-domain CV classification datasets, under which diversity shifts dominate.

- PACS [Li+17] includes domains {Art, Cartoon, Photo, Sketch}, with 9,991 examples and 7 classes.
- VLCS [Fan+13] includes photographic domains {Caltech101, LabelMe, SUN09, VOC2007}, with 10,729 examples and 5 classes.
- OfficeHome [Ven+17] includes domains {Art, Clipart, Product, Real}, with 15,588 examples and 65 classes.
- TerraIncognita [Bee+18] contains photographs of wild animals taken by camera traps at locations {L100, L38, L43, L46}, with 24,788 examples and 10 classes.
- DomainNet [Pen+19] has six domains {Clipart, Infograph, Painting, Quickdraw, Real, Sketch}, with 586,575 examples and 345 classes.

Moreover, DomainBed also includes the following synthetic dataset, under which correlation shifts dominate.

- Colored MNIST [Arj+19] is a variant of the MNIST handwritten digit classification dataset [LeC+10] with domain {90%, 80%, 10%}: the correlation strengths between color and label vary across domains. It contains 70,000 examples and 2 classes.

Critically, all these datasets are multi-domain, i.e., they contain several domains and each domain follows its own distribution. Each domain is successively considered as the test domain while others are for training and validation; this enables fair evaluation of generalization abilities to new domains. Moreover, DomainBed imposes a standard training setup and a strict evaluation protocol, detailed below.

- Each domain is split into 80% (used as training and testing) and 20% (used as validation for hyperparameter selection) splits.
- The network is a ResNet-50 [He+16] pre-trained on ImageNet, with frozen batch normalization layers and a dropout layer just before the classifier.
- The optimizer is Adam [Kin+15]. All runs are trained for 5k steps, except on DomainNet for 15k steps [Arp+21; Cha+21a].
- The hyperparameters follow either the mild or extreme distributions from Table 2.1.
- The experiments are repeated 3 times: the reported numbers will be the means and the standard errors.

The bitter lesson was that, when DomainBed was published, none of the existing methods performed significantly and consistently better than the standard ERM. Fortunately, the approaches proposed in this thesis will change this situation.
2.1.4 Bias-variance decomposition

To better understand and quantify the different sources of generalization error of our DNN, we follow the bias-variance decomposition of the expected error from [Koh+96].

Notations and assumptions. In the rest of this chapter, $\ell$ is the mean-squared error (MSE) for simplicity; yet, our results may be extended to other losses such as the cross-entropy following [Domoo; Woo+23]. We consider a model with weights $\theta(l_S)$ trained with the learning procedure $l_S$. We assume that there is no noise in the data; then the labeling function $f_S : \mathcal{X} \rightarrow \mathcal{Y}$ on $S$ is defined on the source input support $\mathcal{X}_S = \{x \in \mathcal{X} / p_S(x) > 0\}$ by $\forall (x, y) \sim p_S, f_S(x) = y$. Similarly, $f_T : \mathcal{X} \rightarrow \mathcal{Y}$ the labeling functions on $T$ is defined on the target input support $\mathcal{X}_T = \{x \in \mathcal{X} / p_T(x) > 0\}$ by $\forall (x, y) \sim p_T, f_T(x) = y$.

Bias and variance. [Koh+96] decomposed the expected error w.r.t. the learning procedure $l_S$ into two terms in Equation (BV)\(^1\):

$$
\mathbb{E}_{l_S} \mathcal{E}_T (\theta (l_S)) = \mathbb{E}_{(x,y) \sim p_T} \left[ \text{bias}^2 (x, y) + \text{var} (x) \right],
$$

where, with $\bar{f}_S (x) = \mathbb{E}_{l_S} [f(x, \theta(l_S))]$ the expected prediction:

- the bias $\text{bias}(x, y) = y - \bar{f}_S (x)$ measures how far off in general the predictions are from the ground-truth label,
- the variance $\text{var}(x) = \mathbb{E}_{l_S} \left[ (f(x, \theta(l_S)) - \bar{f}_S (x))^2 \right]$ measures how much the predictions vary between different models.

In the traditional statistical learning theory, achieving good generalization usually requires finding a fine balance between these two terms. Indeed, it’s generally assumed that as we add more parameters, the network becomes more flexible, thus the bias is reduced yet the variance increases. In details:

- small ML models with a limited number of parameters tend to have high bias but low variance. Such models are too simplistic to capture the intricate patterns in the data, resulting in a problem called underfitting.

---

1. In this thesis, the equations describing the different decompositions of the error are named with symbols to highlight their importance.
large DNNs with many parameters tend to have low bias but high variance. Such models are complex and flexible, thus capable of fitting their training data, but are prone to a problem called overfitting, meaning they adapt too closely to the training data and would perform poorly on unseen data.

However, this traditional belief has been contested in the context of DL, this questioning being illustrated by the double descent phenomenon [Nak+19; DAs+20]. When increasing the model size, variance initially gets larger (consistently with traditional expectations); yet, at a certain point, when the architecture becomes sufficiently large, this trend reverses and variance gets smaller. In a similar spirit, the forthcoming Section 2.2.1 characterizes the variance in the limit case of infinitely large networks; we show that variance does not grow infinitely large, and depends mostly of the input shifts between train and test distributions.

### 2.2 A new bias-variance theoretical understanding of distribution shifts

We introduce the two main theoretical contributions of this thesis, which explain how the two kind of distribution shifts alter performance. First, in Section 2.2.1, we prove that variance dominates under diversity shift; this relies on the fact that variance essentially becomes a property of the train-test input distribution shifts for sufficiently large networks. Second, in Section 2.2.2, we show that bias dominates under correlation shift. These findings refine our understanding of the bias-variance trade-off and suggest different strategies to enhance generalization for each kind of shift.

#### 2.2.1 Variance and diversity shift

We relate variance to diversity shift [Ye+22] when the network gets infinitely large. We fix the source dataset $D_S$ (with input support $X_{D_S}$), the target dataset $D_T$ (with input support $X_{D_T}$) and the network’s initialization. We get a closed-form expression for the variance of $f$ over all other sources of randomness under Assumptions 2.1 and 2.2.

**Assumption 2.1 (Infinite width).** $f$ is in the kernel regime [Dan17; Lee+17; Jac+18].

This Assumption 2.1 follows the theoretical evidence [Dan17; Lee+17] that, when $f$ is a sufficiently wide network, $f$ behaves as a Gaussian process (GP). The corresponding kernel $K$ is the neural tangent kernel (NTK) [Jac+18] characterized only by the initialization of the weights $\theta$. This approximation is useful because GPs are more interpretable (see Figure 2.3) and more easily analyzed; in particular, their variances have a closed-form expression, as further discussed in Appendix C.1.1.1. To simplify this expression of variance, we make the following Assumption 2.2, further discussed and relaxed in Appendix C.1.1.2.
2.2 A NEW BIAS-VARIANCE THEORETICAL UNDERSTANDING OF DISTRIBUTION SHIFTS

Assumption 2.2 (Constant norm and low intra-sample similarity on $D_S$). \( \exists (\lambda_S, \epsilon) \) with \( 0 \leq \epsilon \ll \lambda_S \) such that \( \forall x_S \in X_{D_S}, K(x_S, x_S) = \lambda_S \) and \( \forall x_S \neq x_S \in X_{D_S}, |K(x_S, x_S')| \leq \epsilon \).

This Assumption 2.2 states that training samples have the same norm (following standard practice [Lee+17; Ah-10; Gho+21; Ren05]) and weakly interact [He+20; Sel+22]. We are now in a position to relate variance and diversity shift when \( \epsilon \to 0 \).

Proposition 2.1 (Variance and diversity shift. Proof in Appendix C.1.1). Given \( f \) trained on \( D_S \) (of size \( n_S \)) with NTK \( K \), under Assumptions 2.1 and 2.2, the variance on \( D_T \) is:

\[
E_{x_T \in X_{D_T}}[\text{var}(x_T)] = \frac{n_S}{2\lambda_S} \text{MMD}^2(X_{D_S}, X_{D_T}) + \lambda_T - \frac{n_S}{2\lambda_S} \beta_T + O(\epsilon),
\]

(2.3)

where MMD is the empirical maximum mean discrepancy in the RKHS of \( K^2(x, y) = (K(x, y))^2 \). Moreover, \( \lambda_T = E_{x_T \in X_{D_T}} K(x_T, x_T) \) and respectively \( \beta_T = E_{(x_T, x'_T) \in X_{D_T}^2, x_T \neq x'_T} K^2(x_T, x'_T) \) are the empirical mean similarities measured between identical (w.r.t. \( K \)) and respectively different (w.r.t. \( K^2 \)) samples averaged over \( X_{D_T} \).

The statistical learning theory predicts that as the network becomes infinitely large, the variance also becomes infinitely large. Proposition 2.1 counters this traditional understanding by providing a new characterization of the variance when the network is sufficiently large to be in the kernel regime (as per Assumption 2.1). Then, the variance is actually mainly governed by the distance between the source and target inputs, as measured by the maximum mean discrepancy (MMD), rather than the model’s complexity.

Critically for better understanding of OOD generalization, this MMD empirically estimates shifts in input marginals, i.e., between \( p_S(X) \) and \( p_T(X) \). Then the first term in this expression of variance is actually very similar to the diversity shift formula in [Ye+22]:

Figure 2.3. – Mean and variance of the predictions for Gaussian processes. Image from [Pér+13]. The x-axis represents the input and the y-axis the targeted output. Intuitively, variance (grey area) across different predictions (blue lines) grows away from training samples (green crosses).
the MMD simply replaces the $L_1$ divergence, another equivalent distance used in \cite{Ye22}. The other terms, $\lambda_T$ and $\beta_T$, both involve internal dependencies on the target dataset $D_T$: they are constants w.r.t. $X_{D_T}$. In conclusion, at fixed $D_T$ and under our assumptions, Equation (2.3) shows that variance on $D_T$ decreases when $X_{D_S}$ and $X_{D_T}$ are closer (for the MMD distance defined by the kernel $K^2$) and increases when they deviate. Intuitively, the further $X_{D_S}$ is from $X_{D_T}$, the less the model’s predictions on $X_{D_T}$ are constrained after fitting $D_S$.

**Remark 2.2.** This theoretical analysis explains an observation named underspecification \cite{DAm20}: models can behave differently in OOD despite similar test ID accuracy. This is simply because variance is larger in OOD than in ID.

### 2.2.2 Bias and correlation shift (and support mismatch)

We now relate OOD bias to correlation shift \cite{Ye22} under Assumption 2.3.

**Assumption 2.3 (Small ID bias).** $\exists \epsilon > 0$ small s.t. $\forall x \in X_S, |f_S(x) - \bar{f}_S(x)| \leq \epsilon$.

This Assumption 2.3 follows the traditional statistical theory; DNNs can fit their training distribution and thus have a small ID bias. This is realistic when they are sufficiently large, trained on a large training dataset representative of the source domain $S$ with an appropriate training configuration $c$. These assumptions are relaxed in Appendix C.1.2.1.

**Proposition 2.2 (Bias and correlation shift. Proof in Appendix C.1.2).** With a bounded difference between the labeling functions $f_T - f_S$ on $X_T \cap X_S$, under Assumption 2.3, the bias on the test target domain $T$ is:

$$
\mathbb{E}_{(x,y) \sim p_T}[\text{bias}(x, y)] = \text{Correlation shift} + \text{Support mismatch} + O(\epsilon),
$$

where Correlation shift $= \int_{X_T \cap X_S} (f_T(x) - f_S(x))^2 p_T(x) dx$, (2.4)

and Support mismatch $= \int_{X_T \backslash X_S} (f_T(x) - \bar{f}_S(x))^2 p_T(x) dx$.

To understand Equation (2.4), we need to first observe that the labeling functions $f_T$ and $f_S$ verify $f_T(x) = \mathbb{E}_{p_T}[Y|X = x]$ and $f_S(x) = \mathbb{E}_{p_S}[Y|X = x]$, $\forall x \in X_T \cap X_S$. Then the first term actually measures correlation shifts in posterior distributions between source and target, as in \cite{Ye22}; this first term would increase in presence of spurious correlations that would modify the labeling functions. A basic example is the ColoredMNIST dataset, where color and label correlation is reversed at test time. Models that have learned the color-label correlation from the source data will predict labels based on the color of the digit rather than its shape, leading to wrong predictions in test and thus high bias.

In a more realistic scenario, instead of considering the input space, we can think in terms of feature space. Replacing $x$ with a feature $z$, the bias would increase if the correlation between $z$ and $y$ is different in the source and target distributions. This
is why it’s important to learn a featurizer encoding inputs into a shared embedding space containing minimal domain-dependent information. This would reduce the bias introduced by potential spurious correlations between extracted features and labels when the model is used in OOD.

Remark 2.3. Less critical in this thesis, the second term is caused by support mismatch between source and target. It was analyzed in [Rua+22] and shown irreducible in their “No free lunch for learning representations for DG”.

Conclusion. This section proves that the two types of distribution shifts reduce performance in two different ways. First, diversity shift increases the variance term; we show in the following Section 2.3.2 that ensembling is helpful in this setup because ensembling is actually a variance reduction strategy. Second, correlation shift increases the bias term.

Remark 2.4. The most successful methods to tackle bias and thus correlation shift usually involve data balancing [Idr+22], robust optimization [Sag+20a], or domain invariance [Arj+19]. Specifically, in Fishr [Ram+22a], we introduce a novel regularization enforcing domain invariance in the space of the gradients of the loss. Fishr was proven effective on ColoredMNIST and on DomainBed, and to this day [Yu+23], remains one of the best strategies to tackle spurious correlations. We have relegated Fishr to Appendix E as its contributions are orthogonal to the main focus of this thesis: ensembling.

2.3 Diverse and efficient ensembling

2.3.1 Prediction averaging

In this thesis, we study how to best combine $M$ models. The standard strategy is the functional ensembling (ENS) which averages the networks’ predictions, as illustrated in Figure 2.4. This ensembling strategy has been a popular research topic in machine learning [Nil65; Wol92; Breg; Die00; Rok10], leading to successful strategies such as random forests [Ho95] or XGBoost [Che+16]. This trivially extends to the functional ensembling of DNNs [Han+90; Kro+95; Zho+02]. We thus define the predictive function:

$$f_{\text{ENS}}(\cdot, \{\theta_i\}_{i=1}^M) = \frac{1}{M} \sum_{i=1}^M f(\cdot, \theta_i).$$ (2.5)

Note that most approaches average the logits rather than the probabilities, as it (slightly but consistently) works better empirically [Ju+18]. In particular, the recent DE [Lak+17] highlighted the success of this simple strategy with independently trained DNNs for better uncertainty estimation [Ash+20].
2.3.2 Bias-variance-covariance decomposition

For simplicity, we consider that the \( M \) models have the same architecture and that the weights \( \{\theta_i\}_{i=1}^M = \{\theta(l_S^{(i)})\}_{i=1}^M \) are obtained from \( M \) identically distributed (i.d.) learning procedures. Then, Equation (BVC) extends Equation (BV) to ensembling: to take into account the \( M \) models, the expectation is over the joint distribution \( L_S^M = \{l_S^{(i)}\}_{i=1}^M \), thus we denote \( f_{ENS}(\cdot, L_S^M) \). Then, the expected generalization test error is decomposed into three terms: bias, variance and covariance.

Proposition 2.3 (Bias-variance-covariance [Ued+96; Bro+05a]. Proof in Appendix C.1.3.). Denoting \( \bar{f}_S(x) = \mathbb{E}_{l_S}[f(x, \theta(l_S))] \), under i.d. learning procedures \( L_S^M \), the expected generalization error on domain \( T \) of the ensembling of those \( M \) models over \( L_S^M \) is:

\[
\mathbb{E}_{L_S^M} \mathcal{E}_T (f_{ENS} (\cdot, L_S^M)) = \mathbb{E}_{(x,y) \sim p_T} \left[ \text{bias}^2(x, y) + \frac{1}{M} \text{var}(x) + \frac{M-1}{M} \text{cov}(x) \right], \quad \text{(BVC)}
\]

where \( \text{cov}(x) = \mathbb{E}_{l_S,l_S'} [ (f(x, \theta(l_S)) - \bar{f}_S(x)) (f(x, \theta(l_S')) - \bar{f}_S(x)) ] \) is the prediction covariance between two models.

In comparison to Equation (BV), the following observations can be made:

- The bias is the same for the ensemble or for each of its i.d. members: combining different models does not reduce the bias. Consequently, this suggests that ensembling provides no benefits under correlation shift (where bias is the major issue). According to this theory, ensembling would not work on ColoredMNIST, as latter confirmed in Section 4.6.3.4.

- The variance of the ensembling is divided into two terms: (i) the variance of each of its i.d. members divided by \( M \), and (ii) a covariance term measuring the correlations across members’ predictions. The covariance will be further analyzed in the subsequent Section 2.3.3, but for now, let’s assume that it is small. Then, this theory suggests that ensembling reduces the variance by a factor of \( M \), effectively removing variance for sufficiently large \( M \). Moreover, we have previously noted variance is large under diversity shift; this suggests that ensembling is highly beneficial under diversity shift, for example on PACS. This is confirmed in the experiments along this thesis.

Figure 2.4. – The traditional functional ensembling, averaging the predictions for \( M = 2 \) models.
In conclusion, this bias-variance-covariance analysis suggests that:

(i) ensembling is useless for correlation shift 
(ii) but efficient for diversity shift, 
(iii) as long as the covariance is small.

2.3.3 Diverse ensembling

2.3.3.1 Diversity

Diversity: the art of thinking independently together

Malcolm Forbes

Diversity to reduce covariance. In addition to the bias and variance of the members, the generalization error of an ensemble also depends on the covariance of the predictions \( \{f(\cdot; \theta_i)\}_{i=1}^{M} \). In the extreme case where all predictions are identical, the covariance equals the variance and ensembling is no longer beneficial. This is because all models are equivalent and the ensemble is no perform better than a single model. Conversely, if the predictions are totally uncorrelated, then the variance is divided by \( M \), making the ensemble more accurate than its individual members. This highlights that the members should have uncorrelated predictions. This lack of correlation across members’ predictions can be summarized into the key notion of diversity, as defined in [Die00]:

Two classifiers are diverse if they make different errors.

Measure of diversity. The concept of diversity has been widely explored in the ensembling literature [Bro+05b; Woo+23]. However, as pointed out in “Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy” [Kun+03], “measuring diversity is not straightforward because there is no generally accepted formal definition”. Thus, numerous diversity measures have been introduced [Kun+03; Aks03; Kor+19]. Considering two classifiers, the most standard ones consider \( N^{ij} \), the number of times 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. Then diversity can be measured with:

1. the ratio-error [Aks03], defined as the ratio \( \frac{N^{01} + N^{10}}{N^{00}} \) between the number of asynchronous errors and of simultaneous errors.

2. the \( Q \)-diversity [Yul00], defined as \( 2 \cdot \frac{N^{01} \cdot N^{10}}{N^{11} \cdot N^{00} + N^{01} \cdot N^{10}} \) (or equivalently 1 minus the \( Q \)-statistic).

For ensembles with more than two members (\( M > 2 \)), these pairwise similarity measures are averaged over all possible pairs: higher values signify that members are less likely
to make errors on the same inputs. In our published papers, we have also considered other prediction diversity measures such as the agreement score (the frequency that both classifiers predict the same class), the Kohavi-Wolpert variance [Koh+96] (the variability of the predicted class), and the entropy diversity (measuring overall disagreement). In this thesis, we also use a diversity measure in features: the centered kernel alignment complement (CKAC) [Kor+19], measuring to what extent the pairwise similarity matrices (computed on domain $T$) are aligned—where similarity is the dot product between feature embeddings extracted from two different networks (rather than the predictions).

### 2.3.3.2 Related work for diverse ensembling

The most standard ensembling baseline is deep ensembling (DE) [Lak+17], where the weights are trained independently from different seed initializations. The diversity among members primarily relies on the randomness of the initialization [Kol+91] and of the learning procedure, hoping to converge towards different explanations for the training data [For+19a; Wil+20]. This simple strategy already provides diverse models, yet, more sophisticated approaches tried to further increase diversity, either explicitly (with diversity regularizations) or implicitly (with additional randomness).

**Explicit diversity.** The first research direction is summarized by this quote from [Bro+05a]: “why shouldn’t we try to find some way to capture the effect of the covariance in the error function?”. Then, many works tried to explicitly reduce covariance by encouraging diversity among the members. Most approaches explicitly regularize the predictions: [Shu+18] force the members to have negatively correlated errors [Liu+99a; Liu+99b]; [Mas20] theoretically motivated the minimization of a second-order PAC-Bayes bounds. Others [Kar+19; Dab+20] enforced diversity in gradients; however, as stated in [Dab+20], “promoting diversity of gradient directions slightly degrades the classification performance on natural examples”. Overall, as far as we know, the unique approach more accurate than the DE baseline is ADP [Pan+19], decorrelating only the non-maximal predictions across members.

**Remark 2.5.** In contrast with previous diversity methods, some works suggested co-distillation across members. For example, in [Zhu+18b] the members learn to mimic each other by reducing the KL between pairs of predictions. This was further extended in [Son+18; Guo+20; Chu+20; Wu+20a; Che+20] and in [Lan+18], the latter using a weighted combination of logits as teacher hence providing better distillation. The problem is that co-distillation reduces diversity by homogenization; in our experiments, all these approaches underperform the DE [Lak+17] baseline.

**Implicit diversity.** The second research direction is to implicitly increase diversity; they are usually more practical as they do not change the training objective. Some approaches introduced additional stochasticity into the training by providing subsets of data to learners with bagging [Bre96], bootstrapping [Efr+94] or by backpropagating subsets of gradients [Lee+16]. Yet, these strategies artificially reduce the number of training samples, hurting performance for DNN that can overfit their training dataset [Nak+19], and thus
failed [Nix+20] because of reduced individual accuracies. One could think of boosting [Fri+01] strategies, yet [Lak+17] argued that sequential training is not suitable for DNNs (mostly because of lack of parallelization). Some more successful approaches applied different transformations [Dvo+19; Sti+20a], used different data augmentations [Wen+21] or hyperparameters [Sin+16; Rui+20; Yan+20d; Wen+20]: these simple diversity tricks will be fundamental in this thesis.

2.3.4 Efficient ensembling

Functional ensembling cost. Ensembling is empirically successful; yet, the inherent issue with ensembling is its cost, both in terms of time and memory, both during training and during inference. These overheads increase linearly with the number of members: handling $M$ models requires $M$ times more resources than handling a single model.

Related work for efficient ensembling. Despite this, naive functional ensembling methods can still be competitive. For example, [Chi+20; Lob+20; Wan+20a; Zha+20b] all found a memory split advantage (MSA): an ensemble of multiple smaller networks typically outperforms a single larger network when compared at an equal number of weights. However, this overhead still limits applicability in real-world applications, where scalability is a crucial factor, and overall suggests a promising research direction:

$$\text{How could we imitate the performance of ensembling within a single model?}$$

Previous works tried to solve this challenge by proposing different efficient methods.

Weight sharing. Seminal methods simply share part of the architecture and weights across models. These methods include the branch-based TreeNets architecture [Lee+15; Son+18; Lan+18] where different classification branches are deployed on top of shared low-level features. Monte Carlo dropout [Gal+16] and matrix factorization [Wen+19; Dus+20] methods also reduce the memory costs, yet they still require multiple forward passes at inference, and actually usually perform poorly [Ash+20].

Subnetwork. [Gao+19; Hav+21; Sof+20; Dur+20; Yan+20c] offer another efficient ensembling strategy, by fitting diverse subnetworks inside one large base network. The idea is that over-parameterized DNNs [Fra+19; Mol+17] can be pruned without loss in performance, and thus their use of parameterization could (in theory) be better optimized. Then the question is how to prevent homogenization among those subnetworks. Multiple strategies were proposed: [Gao+19] includes stochastic channel recombination; [Dur+20] relies on predefined binary masks; in GradAug [Yan+20c], subnetworks only leverage the first channels up to a given percentage. The multi-input multi-output (MIMO) method [Hav+21] is notable, as it does not need structural differences among subnetworks. Rather than considering one single sample to be classified, MIMO’s main idea is to consider simultaneously multiple inputs of different classes. Then each subnetwork tries to classify
one and only one of the multiple inputs; empirically, the subnetworks learn to build their own paths in the base architecture, without homogenization. Such a strategy [Sof+20] can be motivated with arguments from information theory.

Remark 2.6. This simple yet effective MIMO technique motivated our MixMo approach [Ram+21b], where we study how to best mix the multiple inputs given simultaneously during training. This leads to replacing the suboptimal summing operation in MIMO with an improved mixing mechanism based on patches and CutMix [Yun+19]. The key limitation is that MIMO and MixMo require training from scratch, and thus cannot transfer knowledge from foundation models. Thus, for for sake of brevity, the details of MixMo are relegated in Appendix D.

Training sharing. Methods such as snapshot ensembles [Hua+17] and MotherNets [Was+20] share part of the training process to reduce the training overhead. Specifically, snapshot ensembles [Hua+17] create an ensemble of diverse snapshots visited along a single training run with a cyclical learning rate. These methods effectively reduce computational costs during training, yet they do not address the inference overhead.

Weight averaging. An effective strategy to reduce the inference overhead is to average the weights of multiple models, rather than the predictions. Despite the non-linearities in the architecture, and thus perhaps surprisingly, this weight averaging (WA) enables combining into one single model the abilities from multiple models. This idea was initially used in moving average strategies [Izm+18; Zha+19a] as a cheap alternative to snapshot ensembles [Hua+17]. Then, this WA showed promising results in OOD [Cha+21a; Arp+21]. [Cha+21a] argued that WA succeeds in OOD because it provides flatter solution in the loss landscape. Though indeed WA reduces the maximum eigenvalue of the Hessian, in Chapter 4 we will challenge this flatness-based explanation, and show that WA actually improves generalization because of its similarity with functional ensembling. Another issue of existing WA strategies is that the averaged models were highly similar, as they were collected along a single training trajectory, thus limiting diversity and performance. These discovered limitations form the basis of our works on WA, and will be further detailed along this thesis.

2.4 Conclusion

The first goal of this chapter is to provide theoretical guarantees for the empirical success of functional ensembling. Based on the bias-variance-covariance decomposition of the error, we recall that averaging the predictions from multiple models reduces variance. Moreover, we prove that, for sufficiently large DNNs, the variance is actually primarily caused by diversity shifts in input marginal distributions. Therefore, this explains why ensembling especially helps under diversity shift.

The second goal of this chapter is to explain that diversity in ensembling is fundamental, yet complex to optimize directly. Following the limited success of existing diversity regu-
larization in predictions, we propose in the next Chapter 3 a novel explicit regularization enforcing diversity at the feature embedding level.

The third goal of this chapter is to highlight that traditional functional ensembling lacks scalability due to its high computational cost. The existing methods for efficient ensembling correlate errors among members, thus reducing diversity. In Chapters 4 to 6, we push the frontier of weight averaging methods and propose strategies to find accurate and diverse models that remain averageable in weights.
**DICE: DIVERSITY IN ENSEMBLES VIA CONDITIONAL REDUNDANCY ADVERSARIAL ESTIMATION**

### 3.1 Introduction

Averaging the predictions of several models can significantly improve the generalization ability of a predictive system. Indeed, ensembling reduces the variance (see Proposition 2.3) thanks to the diversity among their members. In deep ensembling (DE) [Lak+17], the models are traditionally trained independently. The objective of this chapter is to further increase the diversity across members to improve the ensemble performance.

As previously detailed in Section 2.3.3.2, several works [Mas20; Pan+19] explicitly regularize the predictions to increase diversity, yet with limited success; they systematically reduced the individual performances of the members, highlighting the trade-off between ensemble diversity and individual accuracies. Our idea is to encourage all members to predict the same thing, but for different reasons. Therefore we explicitly enforce diversity in the feature space rather than directly within predictions. Intuitively, to maximize the impact of a new member, extracted features should bring information about the label that is absent at this time so unpredictable from features extracted by other members. The goal is to remove irrelevant redundant information across members, e.g., information shared among features extracted by different members, but useless for label prediction. This redundancy may be caused by a detail in the image background (making members predict badly simultaneously); the key point is that this detail can not be found in features extracted from other images, even when they belong to the same class, as shown in Figure 3.1.

Our new learning framework is called DICE and is driven by information theory and information bottleneck (IB) [Tis01; Ale+17] principles. Specifically, we follow the minimum necessary information (MNI) principle [Fis20] which states that features should be minimal (i.e., compressed) while keeping the necessary (i.e., relevant) information about the label. The goal is to prevent overfitting on noise (memorization) while enabling the learning of a robust predictive mechanism. Specifically, DICE applies the MNI criterion [Fis20] to the ensemble of $M$ members, and thus seeks to reduce (i) the mutual information (MI) between features and inputs, but also (ii) the information shared between members’ features conditioned upon the label. This second point prevents extracted features from being redundant. Intuitively, this comes back to increasing the MI distance between pairs of members, and thus benefitting from MI ability to detect arbitrary dependencies between random variables (such as symmetry, see Figure 3.1). Lastly, a key
Figure 3.1. – Motivation. Our DICE regularization prevents features extracted by different members from being predictable from each other conditionally on the class. Intuitively, features extracted by members (1, 2) from one input (\(\mathcal{X}, \mathcal{Y}\)) should not share more information than features from two inputs in the same class (\(\mathcal{X}, \mathcal{Z}\)): we show this is equivalent to preventing the features extracted by the first member for a first input (\(\mathcal{X}, \mathcal{Y}\)) to be informative enough to differentiate between (i) the features extracted by the second member on the same input (\(\mathcal{X}, \mathcal{Z}\)) and (ii) the features extracted by the second member for another input from the same class (\(\mathcal{X}, \mathcal{W}\)).

Our contribution lies in the conditioning upon the label target, which we argue protects the relevancy and informativeness of extracted features.

The uniqueness of mutual information as a distance measure between variables has been applied in countless machine learning projects [Kim+19; Kem+20; Hje+19]. Yet, MI remains challenging to estimate between high-dimensional features. In this work, we build upon recent advances in neural estimation of MI [Bel+18], based on the Donsker-Varadhan representation of the KL formulation of MI. Overall, to approximate our regularization, we end up implementing an adversarial framework [Goo+14], where a discriminator prevents features from being conditionally predictable from each other. In summary, DICE increases diversity by adversarially reducing irrelevant redundant information among features.

- We introduce DICE, a novel objective to train ensembles of neural networks; based on arguments from information theory, DICE seeks to minimize the conditional redundancy between features (Section 3.2.1).

- We propose an implementation of DICE with a neural estimation of the conditional redundancy, leveraging an adversarial learning framework (Section 3.2.2).

This chapter has led to the publication of the following paper: Alexandre Ramé and Matthieu Cord. “DICE: Diversity in Deep Ensembles via Conditional Redundancy Adversarial Estimation”. In: ICLR. 2021.
3.2 DICE

Notations. Given an input random variable $X$, a network parameterized by weights $\theta$ is trained to extract the best possible feature embedding $Z$ to model the distribution $p_\theta(Y|X)$ over the targets, which should be close to the Dirac on the true label. Our approach is designed for ensembles with $M$ members $\{\theta_i\}_{i=1}^M$ extracting features $\{Z_i\}_{i=1}^M$. We average the $M$ (logit) predictions during inference. In this section, for clarity and simplicity, we consider $M = 2$; yet the published paper [Ram+21a] trivially extends DICE to $M > 2$ members.

Quick overview of DICE. In Section 3.2.1, we justify the DICE objective with information theory. Then in Section 3.2.2, we will detail our implementation of this DICE objective. In brief, our training strategy will (i) train each member separately for classification with information bottleneck (IB) while (ii) removing irrelevant shared information by adversarial training with a discriminator. In conclusion, members should learn to classify with conditionally uncorrelated features and thus increased diversity.

3.2.1 DICE training objective

3.2.1.1 Baseline: non-conditional objective

The MNI criterion from [Fis20] states that the learned features $Z$ should ideally capture only minimal compressed information from $X$, while preserving the necessary relevant information about the label $Y$. We apply this principle to the ensembling of $M = 2$ members by considering the two Markov chains $Z_i \leftarrow X \leftrightarrow Y$ for $i \in \{1, 2\}$. Using entropy as the measure of information, to satisfy the necessary constraint, $I(Y; Z_i)$ should be maximized. Then, regarding the minimality constraint, a first non-conditional formulation is to consider that $I(X; Z_i)$ should be minimized. Mutual information being non-negative, we transform these constraints into a first objective, recovering the IB objective from [Ale+17], but simply applied independently to the $M$ ensembling’s members.

$$\begin{align*}
\text{IB}_{\beta_{ib}}(Z_1, Z_2) &= \frac{1}{\beta_{ib}} \left[I(X; Z_1) + I(X; Z_2)\right] - \left[I(Y; Z_1) + I(Y; Z_2)\right] \\
&= \text{IB}_{\beta_{ib}}(Z_1) + \text{IB}_{\beta_{ib}}(Z_2). \tag{3.1}
\end{align*}$$
The specificities of ensembling appear when we consider cross terms $I(Z_1; Z_2)$, that should be minimal according to the minimality constraint of the MNI.

\[
\text{IBR}_{\beta, \delta}(Z_1, Z_2) = \frac{1}{\beta c_{eb}} \left[ I(X; Z_1) + I(X; Z_2) - I(Y; Z_1) - I(Y; Z_2) \right] + \delta_r I(Z_1; Z_2) \tag{3.2}
\]

\[
\text{IBR}_{\beta, \delta}(Z_1, Z_2) = \text{IB}_{\beta}(Z_1) + \text{IB}_{\beta}(Z_2) + \delta_r I(Z_1; Z_2).
\]

**Analysis.** In this baseline non-conditional criterion, relevancy encourages $Z_1$ and $Z_2$ to capture information about $Y$. Compression & redundancy (R) split the information from $X$ into two compressed & independent views. The relevancy-compression-redundancy trade-off depends on the values of $\beta_{ib}$ & $\delta_r$.

### 3.2.1.2 DICE: conditional objective

The problem is that the previous non-conditional compression and redundancy terms in IBR also reduce necessary information related to $Y$: indeed, it is detrimental to fully disentangle $Z_1$ and $Z_2$ while training them to predict the same $Y$. This is shown on Figure 3.2 where redundancy regions (blue horizontal stripes) overlap with relevancy regions (red diagonal stripes). Indeed, as argued in [Fis+20], the minimality constraints need to be conditioned on the label; thus, we derive the following conditional constraints given $Y$:

\[
I(X; Z_1|Y) = I(X; Z_2|Y) = I(Z_1; Z_2|Y) = 0.
\]

We transform these constraints into our main DICE objective:

\[
\text{DICE}_{\beta_{c_{eb}}, \delta_r}(Z_1, Z_2) = \frac{1}{\beta_{c_{eb}}} \left[ I(X; Z_1|Y) + I(X; Z_2|Y) - I(Y; Z_1) - I(Y; Z_2) \right] + \delta_r I(Z_1; Z_2|Y) \tag{3.3}
\]

\[
\text{DICE}_{\beta_{c_{eb}}, \delta_r}(Z_1, Z_2) = \text{CEB}_{\beta_{c_{eb}}}(Z_1) + \text{CEB}_{\beta_{c_{eb}}}(Z_2) + \delta_r I(Z_1; Z_2|Y),
\]

with $\delta_r > 0$ and $\beta_{c_{eb}} > 0$, where we recover two conditional entropy bottleneck (CEB) [Fis20] components, $\text{CEB}_{\beta_{c_{eb}}}(Z_i) = \frac{1}{\beta_{c_{eb}}} I(X; Z_i|Y) - I(Y; Z_i)$.

**Analysis.** Like previously in Equation (3.2), the relevancy terms force features to be informative about the task $Y$. The difference lies in the conditioning in the bottleneck constraints, only minimizing information irrelevant to $Y$. More specifically, the conditional compression first removes in $Z_i$ information from $X$ not relevant to $Y$. Then, the conditional redundancy (CR) forces the members to have independent features conditionally upon the class. It encourages diversity without affecting members’ individual precision as
it protects in \( Z_1 \) information related to the target label \( Y \). The key intuition explaining why this CR term is important is:

Information from \( X \) irrelevant to predict \( Y \) should certainly not be in \( Z_1 \) or \( Z_2 \), but it is
even worse if they are in \( Z_1 \) and \( Z_2 \) simultaneously as it would cause shared errors.

One could argue that reducing \( I(X, Z_1 | Y) \) indirectly controls \( I(Z_1, Z_2 | Y) \), as by chain rule
we have \( I(Z_1; Z_2 | Y) \leq I(X; Z_1 | Y) \). Yet we will empirically observe that it is more efficient
to directly target this intersection region through the CR term. In a final word, DICE is to
IBR in ensembling as CEB [Fis+20] is to IB [Ale+17] for a single network.

### 3.2.2 DICE implementation

We now provide an empirical strategy to approximate the two CEB terms and the CR
term in the DICE objective from Equation (3.3).

#### 3.2.2.1 Variational approximation of conditional entropy bottleneck

To approximate the independent CEB terms for \( i \in \{1, 2\} \), we exactly follow [Fis20],
as further detailed in Appendix C.2.2. Specifically, we consider Markov assumptions in
\( Z_i \leftarrow X \leftrightarrow Y \) and the training dataset \( D_S = \{x_n, y_n\}_{n=1}^{n_S} \) of \( n_S \) i.i.d. points. Then [Fis20]
showed that \( CEB_{\beta_{ceb}}(Z_i) \) is variationally upper bounded by:

\[
VCEB_{\beta_{ceb}}(\{e_i, b_i, d_i\}) = \frac{1}{n_S} \sum_{n=1}^{n_S} \frac{1}{\beta_{ceb}} D_{KL}(e_i (\cdot | x_n) \| b_i (\cdot | y_n)) - E_{z \sim e_i(\cdot | x_n)} [\log d_i (y_n | z)].
\]

(3.4)

This loss is applied separately on each member \( \theta_i = \{e_i, d_i, b_i\} \) made of an encoder,
a classifier and a backward encoder. Specifically, the feature \( z \) follows the distribution
\( e_i(\cdot | x) \) generated by the encoder, \( d_i(y | z) \) is a variational approximation of the true label
distribution \( p(y | z) \) by the classifier, and \( b_i(\cdot | y) \) is a variational approximation of the feature
distribution for \( y \) \( (p(\cdot | y)) \) by the backward encoder (conditioned only on the label).

Practically, we parameterize all distributions with Gaussians. The encoder \( e_i \) is a traditional
DNN featurizer (e.g., ResNet-32) that learns distributions (means and covariances)
rather than deterministic points in the feature space. That’s why \( e_i \) transforms an image
into 2 tensors; a features-mean \( e_i^b(x) \) and a diagonal features-covariance \( e_i^e(x) \) each of
size \( d \) (e.g., 64). The classifier \( d_i \) is a dense layer that transforms a features-sample \( z \)
(following \( e_i(\cdot | x_n) \)) into logits to be aligned with the target \( y \) through conditional cross-
entropy. \( z \) is obtained via reparameterization trick: \( z = e_i(x, \epsilon) = e_i^b(x) + \epsilon e_i^e(x) \) with
\( \epsilon \sim N(0, 1) \). Finally, the backward encoder \( b_i \) is implemented as an embedding layer of
size \( (K, d) \) mapping the \( K \) classes to class-features-means \( b_i^e(z | y) \) of size \( d \), as we set the
class-features-covariance to \( I \). The Gaussian parametrization also enables the exact com-
putation of the \( D_{KL} \) (see Appendix C.2.1), that forces (i) features-mean \( e_i^b(x) \) to converge
to the class-features-mean \( b_i^e(z | y) \) and (ii) the predicted features-covariance \( e_i^e(x) \) to be
close to 1. The advantage of VCEB versus VIB [Ale+17] (detailed in Appendix C.2.2) is the class conditional \( b_i^p(z|y) \) versus non-conditional \( b_i^p(z) \), protecting class information.

3.2.2.2 Adversarial estimation of conditional redundancy

**MI estimation.** We now focus on estimating \( I(Z_1; Z_2|Y) \). As there is no Markov properties across those variables, the variational approximations made above are not possible. Thus, we need a more complex approach for mutual information (MI) estimation. One problem is that those methods scale badly in high dimensions (in sample size or dimension) \[Gao+15\], and are not computationally efficient. Fortunately, in this work, we benefit from the recent advances made by [Bel+18] that proposed a neural estimation of MI. Specifically, [Bel+18] formulated MI estimation as a discriminative problem achievable by a neural network. This is possible theoretically thanks to the Donsker-Varadhan [Don+75] dual representations of the KL divergence. [Bel+18] empirically showed that this neural estimation is scalable, flexible, and (most importantly) completely trainable via backpropagation. Then, [Muk+20] extended this strategy for estimation of conditional MI.

\[
\text{CR} = I(Z_1; Z_2|Y) = D_{KL}(P(Z_1, Z_2, Y)\|P(Z_1, Y)p(Z_2|Y)) = \sup_f \mathbb{E}_{x \sim p(z_1, z_2, y)}[f(x)] - \log(\mathbb{E}_{x \sim p(z_1, y)p(z_2|y)}[\exp(f(x))]) = \mathbb{E}_{x \sim p(z_1, z_2, y)}[f^*(x)] - \log(\mathbb{E}_{x \sim p(z_1, y)p(z_2|y)}[\exp(f^*(x))]),
\]

where \( f^* \) computes the pointwise likelihood ratio, i.e., \( f^*(z_1, z_2, y) = \frac{p(z_1, z_2, y)}{p(z_1, y)p(z_2|y)}. \)

**Empirical neural estimation.** We estimate CR (i) using the empirical data distribution and (ii) replacing \( f^* = \frac{w^*}{1-w^*} \) by the output of a discriminator \( w \), trained to imitate the optimal \( w^* \). Let \( B_J \) be a batch sampled from the observed joint distribution \( p(z_1, z_2, y) = p(e_1(z|x), e_2(z|x), y) \); we select the features extracted by the two members from one input. Let \( B_P \) be sampled from the product distribution \( p(z_1, y)p(z_2|y) = p(e_1(z|x), y)p(z_2|y) \); we select the features extracted by the two members from two different inputs that share the same class. We train a multi-layer network \( w \) on the binary task of distinguishing these two distributions with the standard cross-entropy loss:

\[
L_{ce}(w) = -\frac{1}{|B_J| + |B_P|} \left[ \sum_{(z_1, z_2, y) \in B_J} \log w(z_1, z_2, y) + \sum_{(z_1, z_2, y) \in B_P} \log(1 - w(z_1, z_2, y)) \right].
\]

If \( w \) is calibrated, a consistent [Muk+20] estimate of CR, with \( f = \frac{w}{1-w} \), is:

\[
\hat{I}_{DV}^{CR} = \frac{1}{|B_J|} \sum_{(z_1, z_2, y) \in B_J} \frac{\log f(z_1, z_2, y)}{\text{Diversity}} - \log \left( \frac{1}{|B_P|} \sum_{(z_1, z_2, y) \in B_P} f(z_1, z_2, y) \right) \frac{\text{Fake correlations}}{	ext{Diversity}}.
\]
Figure 3.3. – Learning strategy overview. Blue arrows represent training criteria: (i) classification with conditional entropy bottleneck applied separately on each member, and (ii) adversarial training to delete irrelevant redundant information between members and increase diversity. $X$ and $X'$ belong to the same $Y$ for conditional redundancy minimization.

**Intuition.** By training our members to minimize $\hat{I}_{CR}^{DV}$, we force triples from the joint distribution to be indistinguishable from triples from the product distribution. Let’s imagine that two features are conditionally correlated, some irrelevant information is shared between features only when they are from the same input and not from two inputs (from the same class). This correlation can be informative about a detail in the background, an unexpected shape in the image, that is rarely found in other samples from the same class. In that case, the product and joint distributions are easily distinguishable by the discriminator. The first adversarial component will force the extracted features to reduce the correlation, and ideally one of the two features loses this information: it reduces redundancy and increases diversity. The impact of the second term is more complex; it may create fake correlations between features from different inputs. As we are not interested in a precise estimation of the CR, we get rid of this second term that, empirically, did not increase diversity. We end up with the following empirical estimation of CR.

$$\hat{L}_{CR}^{DV}(e_1, e_2) = \frac{1}{|B_J|} \sum_{(z_1, z_2, y) \in B_J \sim p(e_1(z|x), e_2(z|x), y)} \log f(z_1, z_2, y). \quad (3.7)$$

**Summary.** First, we train each member for classification with VCEB from Equation (3.4), as shown in Step 1 from Figure 3.3. Second, following Step 2 from Figure 3.3, the discriminator, conditioned on the class $Y$, learns to distinguish features sampled from one image versus features sampled from two images belonging to the same $Y$. Simultaneously, both members adversarially [Goo+14] delete irrelevant shared information to reduce CR estimation from Equation (3.7) with differentiable signals. The full DICE loss is finally:

$$L_{DICE}(\theta_1, \theta_2) = VCEB_{\beta_{ceb}}(\theta_1) + VCEB_{\beta_{ceb}}(\theta_2) + \delta_{cr} \hat{L}_{CR}^{DV}(e_1, e_2). \quad (3.8)$$
3.3 Experiments

In this experimental section, we show that DICE improves accuracy by increasing diversity. In the published paper [Ram+21a] we provide more experiments for calibration, uncertainty estimation, out-of-distribution detection and co-distillation.

3.3.1 Baselines and concurrent works.

The strategies motivated by IB principles involve two main components:

- compression; the encoder can be deterministic, or variationally compressed non-conditionally (VIB) or conditionally (VCEB),
- redundancy; we can have no regularization, a non-conditional redundancy (R) or a conditional redundancy (CR) component.

To ablate the importance of each component, we compare the following approaches.

1. Ind. DE [Lak+17] refers to the independent deterministic deep ensembling without interactions between members. Other approaches below follow IB principles and thus their members have probabilistic encoders.

2. IB [Ale+17] from Equation (3.1) compresses with VIB.

3. CEB [Fis20] compresses with VCEB.

4. IBR follows Equation (3.2) and optimizes:

   \[
   \mathcal{L}_{IBR}(\theta_1, \theta_2) = \text{VIB}_{\beta_{ib}}(\theta_1) + \text{VIB}_{\beta_{ib}}(\theta_2) + \delta_i \hat{L}_D^R(e_1, e_2),
   \]

   where: \[
   \hat{L}_D^R(e_1, e_2) = \frac{1}{|B_J|} \sum_{(z_1, z_2) \in B_J} \log f(z_1, z_2)
   \]

   is simply the approximation of the non-conditional redundancy R (same as Equation (3.7) but without the label y).

5. CEBR benefits from VCEB with approximation of non-conditional redundancy R.

   \[
   \mathcal{L}_{CEBR}(\theta_1, \theta_2) = \text{VCEB}_{\beta_{ceb}}(\theta_1) + \text{VCEB}_{\beta_{ceb}}(\theta_2) + \delta_i \hat{L}_D^R(e_1, e_2).
   \]

6. DICE combines VCEB with approximation of conditional redundancy CR.

Regarding the concurrent works, we consider two co-distillation strategies (ONE [Lan+18] and OKDDip [Che+20]), discussed in Remark 2.5. We also consider the best diversity regularization in predictions, ADP [Pan+19], decorrelating only the non-maximal predictions. Regarding the other works that enforce diversity in features, we found that increasing \((L_1, L_2, -\cos)\) distances [Kim+18] between features fail: we speculate this is because they are not invariant to variables’s symmetry. The work most similar to ours is [Sin+20b], proposing a diversity-inducing adversarial loss highly similar to our IBR.
### 3.3 Experiments

<table>
<thead>
<tr>
<th>Table 3.1. – CIFAR-100 ensemble classification accuracy (Top-1, %).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>Ind. DE [Lak+17]</td>
</tr>
<tr>
<td>ONE [Jan+18]</td>
</tr>
<tr>
<td>OKDDip [Che+20]</td>
</tr>
<tr>
<td>ADP [Pan+19]</td>
</tr>
<tr>
<td>IB [Acle+17]</td>
</tr>
<tr>
<td>CEB [Fic+20]</td>
</tr>
<tr>
<td>IBR Equation (3.9)</td>
</tr>
<tr>
<td>CEBR Equation (3.10)</td>
</tr>
<tr>
<td>DICE Equation (3.8)</td>
</tr>
</tbody>
</table>

### 3.3.2 Results on CIFAR

Table 3.1 reports the classification accuracy averaged over 3 runs for CIFAR-100 [Kri+09], while Table 3.2 focuses on CIFAR-10. \{3,4,5\}-\{branch,net\} refers to the training of \{3,4,5\} members \{with,without\} low-level weights sharing. We detail our implementation in the paper. We took most hyperparameter values from [Che+20]: those for adversarial training and information bottleneck were fine-tuned on a validation dataset made of 5% of the training dataset. Bold highlights best score.

<table>
<thead>
<tr>
<th>Table 3.2. – CIFAR-10 ensemble classification accuracy (Top-1, %).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Arch</strong></td>
</tr>
<tr>
<td>ResNet-32</td>
</tr>
<tr>
<td>ResNet-110</td>
</tr>
</tbody>
</table>

DICE surpasses concurrent approaches for ResNet and Wide-ResNet architectures. On CIFAR-100, DICE outperforms DE by \{+0.60,+0.73,+0.84\} for \{3,4,5\}-branches ResNet-32. We also bring significant and systematic improvements to ADP [Pan+19]: e.g., \{+0.52,+0.30,+0.41\} for \{3,4,5\}-branches ResNet-32, \{+0.94,+0.53\} for \{3,4\}-branches ResNet-110 and finally +0.34 for 3-networks WRN-28-2. Moreover, in Figure 3.4, an ensemble of 5 networks trained with DICE matches an ensemble of 7 networks trained independently; these results confirm that diversity approaches better leverage size. Finally, we observe that our gains are more important in the branch setup. This makes the TreeNet [Lee+15; Sze+15] architecture attractive, as it reduces memory cost at only a slight cost in diversity, which can be compensated with our DICE regularization.
3.3.3 Individual accuracy-diversity trade-off

We now analyze how the two components of DICE modify the accuracy-diversity trade-off, with a 4-branches ResNet-32 on CIFAR-100. We measure diversity with the ratio-error \([\text{Aks03}]\), and observe similar results for other diversity measures in the paper [Ram+21a]. In Figure 3.5, CEB has slightly higher diversity than independent trainings: diversity benefits from compression. ADP reaches higher diversity but sacrifices individual accuracies. On the contrary, co-distillation OKDDip [Che+20] sacrifices diversity for individual accuracies. DICE curve is above all others, and notably \(\delta_{cr} = 0.2\) induces an optimal trade-off between ensemble diversity and individual accuracies on validation. CEBR reaches same diversity with lower individual accuracies, because some information relevant about \(Y\) was removed during non-conditional redundancy regularization.

Figure 3.6 shows the training dynamics for different values of \(\delta_{cr}\) in DICE; in particular, \(\delta_{cr} = 0.0\) corresponds to CEB. Starting from random initializations, diversity always begins small. Then larger values of \(\delta_{cr}\) minimizes the estimated CR in features and increases diversity in predictions compared to CEB. Specifically a very high value (\(\delta_{cr} = 0.6\)) creates too much diversity as it drastically reduces individual performances. On the contrary, a negative value (\(\delta_{cr} = -0.025\)) can decrease diversity.
Figure 3.5. – Ensemble and individual accuracies as a function of diversity, highlighting the different trade-offs for different strategies. DICE (r. CEBR) is learned with different $\delta_{cr}$ (r. $\delta_r$).

Figure 3.6. – Impact of the diversity coefficient $\delta_{cr}$ in DICE on the training dynamics on validation: CR is negatively correlated with diversity.

### 3.3.4 Results on DomainBed

Table 3.3. – OOD accuracy (%↑) on DomainBed. “Art” is the OOD domain for both datasets.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>PACS</th>
<th>OfficeHome</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERM</td>
<td>87.6 ± 0.4</td>
<td>62.9 ± 1.3</td>
</tr>
<tr>
<td>Ind. DE</td>
<td>88.2 ± 0.2</td>
<td>65.3 ± 0.6</td>
</tr>
<tr>
<td>DICE</td>
<td>88.4 ± 0.2</td>
<td>65.7 ± 0.5</td>
</tr>
</tbody>
</table>

Previous experiments were conducted on CIFAR with a ID test dataset: in Table 3.3, we now provide OOD results on two datasets from DomainBed [Gul+21] where diversity shifts dominate. On PACS, the training domains are “Cartoon”, “Photo” and “Sketch”; the OOD accuracy is reported on PACS’s “Art” domain. On OfficeHome, the training domains are “Clipart”, “Product” and “Real”; the OOD accuracy is reported on OfficeHome’s “Art” domain. All methods use 20 runs with hyperparameters sampled from the mild range in Table 2.1, the ensembling methods use $M = 2$ ResNet-50, and we consider DICE with $\delta_{cr} = 0.2$. We consistently observe that ensembling and DICE improve performances on these datasets. The subsequent chapters will enrich our analysis on DomainBed [Gul+21], which actually appeared after DICE [Ram+21a] was published.


3.4 Conclusion

In this chapter, motivated by arguments from information theory, we derive a novel adversarial training strategy for ensemble. We tackle the trade-off between individual accuracies and ensemble diversity by deleting irrelevant information across members. We improve the accuracy on CIFAR-10 and CIFAR-100 compared to the DE baseline. We show that DICE can be combined with efficient ensembling strategies such as TreeNets [Lee+15], where members share low-level weights.

Perspectives. Despite its merits, this initial approach has two critical limitations. First, DICE requires to learn simultaneously all models, inhibiting parallelization during training. Moreover, subsequent research [Wor+23; Abe+23] revealed the limitations of diversity regularizations when applied to larger datasets such as ImageNet. Given these limitations, we will explore implicit strategies to bolster diversity in the rest of this thesis.
4.1 Introduction

The framework of foundation models [Bom+21] is fueling a spectacular adoption of ML solutions: these models are pre-trained on large-and-diverse data [Fan+22; Ngu+22; Abn+22] and easy [Oqu+14] to adapt to downstream tasks. Though this transfer learning helps, models still struggle to generalize on out-of-distribution (OOD) samples [Hen+19a; Tao+20; Gul+21; Hen+21]. Increased OOD generalization would enable the responsible use of ML in real-world applications where robustness are critical [Tay+16; Zec+18; DeG+21], as previously described in Section 1.3. Thus, how to best fine-tune foundation models for OOD generalization is a key topic of research.

On the reference DomainBed benchmark [Gul+21] evaluating different fine-tuning strategies for OOD generalization, the standard empirical risk minimization (ERM) was recently outperformed by moving average (MA) [Cha+21a; Arp+21]: MA simply weight averages (WA) the various checkpoints (a.k.a. snapshots) collected along a training trajectory [Izm+18]. [Cha+21a] argue that this WA succeeds because it finds solutions in flatter regions of the loss landscape.

In this chapter, we challenge this flatness-based analysis, and show its limitations. We then propose a new explanation for the success of WA in OOD based on its similarities with functional ensembling ENS: we show that averaging the models or the weights behave similarly as long as the weights remain sufficiently close. Based on this similarly and the bias-variance-covariance Equation (BVC) for ENS, we obtain a bias-variance-covariance-locality decomposition of WA’s expected error. It contains four terms:

1. first the bias increasing under correlation shift [Ye+22], see Section 2.2.2,
2. second, the variance increasing under diversity shift [Ye+22], see Section 2.2.1, divided by the number of averaged models,
3. third, the covariance decreasing when models are diverse, see Section 2.3.3,
4. finally, a new locality condition on the weights, enforcing averageability.

This explains WA’s success for domain generalization (i) under diversity shift (where variance dominates), (ii) as long as the models are functionally diverse (to reduce variance) (iii) yet close in the weight space (to ensure averageability).

Based on this analysis, we aim at enhancing diversity while preserving weight averageability. We thus propose Diverse Weight Averaging (DiWA) averaging weights ob-
tained from independent fine-tunings, all starting from a shared pre-trained initialization, yet with different hyperparameters and data orders. The motivation is that those models are more diverse than those obtained along a single run [For+19a; Gon+22]. Moreover, because the shared initialization is pre-trained, the weights of the models remain close enough to be averaged. This follows the linear mode connectivity (LMC) [Fra+20; Nag+19], as previously explored for fine-tuning in [Ney+20].

- We reveal (theoretical and empirical) limitations from existing flatness-based analysis of weight averaging (WA) (Section 4.2.2).
- We propose a new analysis for WA based on its similarity with ensembling and a new bias-variance-covariance-locality decomposition of its error (Section 4.3).
- We propose DiWA to enhance the diversity across averaged models by decorrelating their training procedures: in practice, these models are obtained from independent runs with different hyperparameters (Section 4.4).
- Experimentally, DiWA improves performances on the competitive DomainBed benchmark, without any inference overhead (Section 4.6).

This chapter has led to the publication of Alexandre Ramé, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. “Diverse Weight Averaging for Out-of-Distribution Generalization”. In: NeurIPS. 2022.

4.2 Context

4.2.1 Fine-tuning for OOD generalization

Problem and notations. We start by describing our setup. We train a deep neural network (DNN) \( f(\cdot, \theta) : \mathcal{X} \to \mathcal{Y} \) with weights \( \theta \), that should maximize the test accuracy \( \text{acc}_T(\theta) \), or equivalently minimize the test generalization error \( \mathcal{E}_T(\theta) \). To this end, \( f(\cdot, \theta) \) should approximate the labelling function \( f_T \) on \( \mathcal{X}_T \). However, this is complex in OOD because we only have data from domain \( S \) in training, related yet different from \( T \). The differences between \( S \) and \( T \) are due to distribution shifts (i.e., the fact that \( p_S(X,Y) \neq p_T(X,Y) \)) which are decomposed per [Ye+22] into (i) diversity shift, when marginal distributions differ (i.e., \( p_S(X) \neq p_T(X) \)), and (ii) correlation shift, when posterior distributions differ (i.e., \( p_S(Y|X) \neq p_T(Y|X) \) and \( f_S \neq f_T \)).

Vanilla fine-tuning. For OOD generalization, transfer learning [Oqu+14; Kir+22] from a foundation model [Bom+21] and then supervised fine-tuning with ERM [Vap92] is the standard strategy [Gul+21]. Specifically, from a featurizer pre-trained on a large dataset such as ImageNet [Rus+15], users usually launch multiple fine-tunings on the target task with different hyperparameters, and then select the best one based on some validation metric [Gul+21].
Weight averaging over epochs. Recently, WA strategies came to the foreground [Sze+16; Izm+18; Dra+18], as previously described in Section 2.3.4. 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 DNNs, the efficacy of WA was a surprising observation, that [Fra+20] latter called the linear mode connectivity (LMC).

Observation 4.1 (LMC with different epochs [Izm+18]). Two weights $\theta_1$ and $\theta_2$, obtained at two different epochs of the same fine-tuning, satisfy the LMC: for all $\lambda \in [0, 1],$

$$\text{acc}_T((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2) \geq (1 - \lambda) \cdot \text{acc}_T(\theta_1) + \lambda \cdot \text{acc}_T(\theta_2).$$

(4.1)

The LMC holds if the accuracy of the interpolated weights is above the interpolated accuracy. Consistently with Observation 4.1, recent works [Arp+21; Cha+21a; Wor+22b; Kad22] weight average checkpoints along training to improve accuracies. In particular, [Arp+21] showed that the simple moving average (MA) strategy, which samples uniformly along training, is sota on DomainBed [Gul+21].

4.2.2 Flatness analysis of WA

4.2.2.1 Flatness generalization bound

Despite its empirical success, WA strategies lack theoretical foundations. Previous analysis [Cha+21a] argues that WA succeeds because it flattens the loss landscape. Specifically, the “flatness” Theorem 1 in [Cha+21a] upper bounds target generalization error by a sum of three terms.

1. The first and most important term involves the solution’s ID flatness, and is usually estimated by the trace of the Hessian [Din+17b; Pet+21; Yao+20], i.e., the sum of its eigenvalues.
2. The second term is a domain divergence between the source and target marginal distributions, which grows under distribution shifts.
3. The last (and not analyzed) term involves the VC dimension of the networks.

4.2.2.2 Limitations of the flatness analysis

In Figure 4.1, we analyze flatness via the Hessian trace, computed with [Yao+20], on the OfficeHome dataset [Ven+17], using “Art” as the OOD test domain. We confirm that WA flattens the loss landscape; yet this does not mean that WA succeeds because of it. Indeed, several theoretical inconsistencies and unanswered empirical observations remain.

1. Flatness and domain shifts. First, the flatness-based analysis is not specific to OOD. Indeed, the flatness and domain divergence in the upper bound of [Cha+21a] from Section 4.2.2.1 are not interacting; more flatness does not reduce domain divergence,
and the OOD error is actually uncontrolled. In other words, additional flatness may improve generalization in general, but this bound tells us nothing about an hypothetic specific impact for OOD.

2. **WA vs. SAM.** Second, the upper bound does not clarify why MA outperforms flatness-based methods such as sharpness-aware minimization (SAM) \cite{For+21}, which is an optimization strategy that directly optimizes flatness along training. In Figure 4.1, we observe that SAM actually finds flatter minimas than MA \cite{Arp+21}. Yet, this is not reflected in OOD accuracies in Figure 4.2 where MA outperforms SAM. These results have been confirmed empirically in \cite{Cha+21a}. In conclusion, flatness is not sufficient to explain why WA works so well in OOD, because SAM has better flatness but worse OOD results.

3. **Combining WA and SAM.** We investigate another empirical inconsistency when combining MA and SAM. As argued in \cite{Kad+22}, we confirm in Figure 4.1 that MA + SAM usually leads to flatter minimas than MA and WA alone. Yet, MA does not benefit from SAM in Figure 4.2. Moreover, \cite{Cha+21a} also showed in their Table 4 that SWAD + ERM outperforms SWAD + SAM. This is not explained by the upper bound from \cite{Cha+21a}, which argues that more flatness should improve OOD results.

### 4.3 Bias-variance-covariance-locality analysis

To replace this flatness-based analysis, we propose a new one based on the similarity between WA and functional ensembling, as shown in Section 4.3.1. We then decompose WA’s expected error in Section 4.3.2. This analysis suggests three (conflicting) success criterion for WA, as the averaged models should be:

1. individually accurate (to reduce the bias term),
2. diverse functionally (to reduce the covariance term),
3. close in the weight space (or at least satisfying the linear mode connectivity (LMC), to reduce the locality term).
4.3.1 WA approximates ensembling

**Notations.** We recall the notations previously introduced in Chapter 2. When needed, we explicitly write $\theta(l_S)$ to refer to the weights obtained after stochastic minimization on $\mathcal{D}_S$ w.r.t. $\theta$ under $l_S = \{\mathcal{D}_S, c\}$ a learning procedure, where:

- the training dataset $\mathcal{D}_S$ from $S$ is composed of $n_S$ i.i.d. samples from $p_S(X, Y)$,
- the configuration $c$ contains all other sources of randomness in the learning.

We study the benefits of averaging $M$ weights $\{\theta_i\}_{i=1}^M = \{\theta(l_S^{(i)})\}_{i=1}^M$ obtained from the $M$ learning procedures $L_S^M = \{l_S^{(i)}\}_{i=1}^M$ (potentially correlated yet) identically distributed (i.d.). Their WA is $f_{WA} = f(\cdot, \theta_{WA})$ where $\theta_{WA} = \theta_{WA}(L_S^M) = \frac{1}{M} \sum_{i=1}^M \theta_i$.

**Ensembling.** To decompose WA’s error, we leverage its similarity with functional ensembling $f_{ENS} = \frac{1}{M} \sum_{i=1}^M f(\cdot, \theta_i)$. Specifically, Lemma 4.1 establishes that $f_{WA}$ is a first-order approximation of $f_{ENS}$ when $\{\theta_i\}_{i=1}^M$ are close in the weight space. The proof is in Appendix C.3.

**Lemma 4.1 (WA and ENS. Adapted from [Izm+18; Wor+22a]).** Given $\{\theta_i\}_{i=1}^M$ with learning procedures $L_S^M = \{l_S^{(i)}\}_{i=1}^M$. Denoting $\Delta_{L_S^M}^2 = \max_{i=1}^M \|\theta_i - \theta_{WA}\|_2$, $\forall (x, y) \in \mathcal{X} \times \mathcal{Y}$:

$$f_{WA}(x) = f_{ENS}(x) + O(\Delta_{L_S^M}^2) \text{ and } \ell(f_{WA}(x), y) = \ell(f_{ENS}(x), y) + O(\Delta_{L_S^M}^2).$$

(4.2)

4.3.2 Bias-variance-covariance-locality decomposition

This similarity is useful since Equation (BV) was extended into a bias-variance-covariance decomposition for ENS in Equation (BVC) following [Ued+96; Bro+05a]. Thus, by inserting Equation (BVC) into Equation (4.2), we obtain the following Proposition 4.1.

**Proposition 4.1 (Bias-variance-covariance-locality decomposition of WA’s error).** With $\bar{f}_S(x) = \mathbb{E}_{l_S}[f(x, \theta(l_S))]$, under i.d. $L_S^M = \{l_S^{(i)}\}_{i=1}^M$, the expected generalization error on domain $T$ of WA over the joint distribution of $L_S^M$, is:

$$\mathbb{E}_{L_S^M} \mathcal{E}_T(\theta_{WA}(L_S^M)) = \mathbb{E}_{PT}[\text{bias}(x, y)^2 + \frac{\text{var}(x)}{M} - \frac{M - 1}{M} \text{cov}(x)] + O(\bar{\Delta}^2),$$

(BVCL)

where:

$$\text{bias}(x, y) = y - \bar{f}_S(x),$$

$$\text{var}(x) = \mathbb{E}_{l_S} \left[ (f(x, \theta(l_S)) - \bar{f}_S(x))^2 \right],$$

$$\text{cov}(x) = \mathbb{E}_{l_S, l_S'}(f(x, \theta(l_S)) - \bar{f}_S(x))(f(x, \theta(l_S')) - \bar{f}_S(x)),$$

$$\bar{\Delta}^2 = \mathbb{E}_{L_S^M} \Delta_{L_S^M}^2 \text{ where } \Delta_{L_S^M}^2 = \max_{i=1}^M \|\theta_i - \theta_{WA}\|_2.$$

The locality term $\bar{\Delta}^2$ is the expected squared maximum distance between weights and their average.
4.3.3 Locality, diversity and linear mode connectivity

Equation (BVCL) involves four terms. The three first terms are those from functional ensembling, previously analyzed in Section 2.3.2; namely the bias (the same as each of its i.d. members), a variance divided by $M$ and a covariance term (analyzed in Section 2.3.3, measuring diversity).

In contrast, the last locality term $O(\bar{\Delta}^2)$ is specific to weight averaging; it ensures that WA approximates ENS by constraining the weights to remain close. Overall, locality and diversity are two antagonistic terms: to reduce WA’s error, we thus seek a good trade-off between locality and diversity. In practice, we consider that the main goal of this locality term is to ensure that the weights are averageable despite the non-linearities in the DNN such that WA’s error does not explode. This is why in Section 4.4, we empirically relax this locality constraint and simply require that the weights satisfy the linear mode connectivity (LMC) [Fra+20]. We empirically verify later in Figure 4.5 that the approximation $f_{WA} \approx f_{ENS}$ remains valid even in this case.

Conclusion. We have showed in Section 2.2 that bias dominates under correlation shift and variance dominates under diversity shift. Therefore, in a similar fashion to functional ensembling, our analysis suggests that WA would be effective against diversity shift when $M$ is large and when its members are diverse but linearly mode connected.

4.3.4 Superiority of our analysis

We summarize why analyzing WA through the prism of functional ensembling and variance reduction fixes failures from previous flatness-based analysis [Cha+21a].

1. First, our analysis explains why WA performs well under diversity shift, and less in-distribution (as observed in previous works [Arp+21; Wor+22a]) or not under correlation shift (as validated in Section 4.6.3.4). This is because variance is mostly an issue under diversity shift, and that variance reduction is useless under correlation shift where bias dominates. This was not predicted by the flatness analysis. We also explicitly explain why increasing the number of weights $M$ helps.

2. Our analysis explains why WA and SAM are not complementary in OOD, as visible in Figure 4.2, in contradiction with what was argued in [Cha+21a]. Indeed, we observe in Figure 4.3 that the diversity across two checkpoints collected along a SAM trajectory is much lower than along a standard SGD trajectory. Therefore, the gain in individual accuracies of models trained with SAM cannot compensate the decrease in diversity.
4.4 DiWA

4.4.1 Motivation: weight averaging from different runs for more diversity

Limitations of single-run WA. Our analysis suggests that WA performances can be improved by increasing diversity across the averaged models. Yet, previous WA methods [Cha+21a; Arp+21] only average weights obtained along a single run. This corresponds to highly correlated procedures sharing the same initialization, hyperparameters, batch orders, data augmentations and noise, that only differ by the number of training steps. The models are similar, which does not leverage the full potential of WA.

DiWA. Our DiWA pushes the envelope of WA techniques, and reduce the OOD expected error in Equation (BVCL) by decorrelating the learning procedures \( \{l^{(i)}_S\}_{i=1}^M \). The weights are obtained from \( M \gg 1 \) different runs, with different hyperparameters (learning rate, weight decay and dropout probability), batch orders, data augmentations (e.g., random crops, horizontal flipping, color jitter, grayscaling), stochastic noise and number of training steps. Thus, the models averaged in DiWA collected from those multiple runs would be more diverse than the snapshots collected along a single training run. However, this gain in diversity may break the locality requirement from Section 4.3.3; below, we detail the empirical conditions under which DiWA succeeds.
4.4.2 On the importance of pre-training for weight averaging

The success of DiWA relies on the following observation from [Ney+20]: “there is no performance barrier between two instances of models trained from pre-trained weights”. Two independent fine-tunings from a shared pre-trained model satisfy the LMC, and thus can be connected along a linear path where error remains low [Nag+19; Fra+20]. Formally:

Observation 4.2 (LMC with different runs [Ney+20]). The LMC holds between $\theta_1$ and $\theta_2$ fine-tuned on the target task initialized from a shared pre-trained model.

Shared pre-trained initialization. This Observation 4.2 means that DiWA would work if the models from different runs are initialized from a shared pre-trained model. In our experiments, our featurizer will be pre-trained on ImageNet [Kri+12], following the standard setup from DomainBed. Regarding the classifier initialization, we test two methods. The first is the random initialization, which may distort the features [Kum+22]. The second is linear probing (LP) [Kum+22]: it first learns the classifier (while freezing the featurizer) to serve as a shared initialization; then, LP fine-tunes the featurizer and the classifier together in the $M$ subsequent runs. This two-step procedure would reduce the locality term, as [Kum+22] showed that fine-tuned weights then tend to remain closer.

4.4.3 Mild hyperparameter search

In addition to the shared pre-trained initialization, we add a constraint on the hyperparameters. This is based on the observation from Figure 4.10 that extreme hyperparameter ranges may lead to poor performances after WA; we speculate that weights obtained from extremely different hyperparameters (in particular different learning rates) may belong to different regions of the loss landscape. In our experiments, we thus use the mild hyperparameter ranges defined in Table 2.1. This mild hyperparameter search empirically guarantees high diversity and averageability.

4.4.4 Weight selection

The last stage of our approach is to choose which weights to average among those available. Our standard uniformly averages all weights; it is practical but may underperform when some runs have failed. A possible refinement proposed in [Wor+22a] is a greedy selection, restricting the number of selected weights: weights are ranked in decreasing order of ID validation accuracy and sequentially added only if they improve DiWA’s ID validation accuracy.
4.5 Related work.

The LMC is at the heart of DiWA but also of other recent works [Wor+22b; Mat+22; Wor+22a] that average weights with fewer constraints [Wor+22b; Mat+22; Wor+22a; Gup+20; Cho+22b; Wor+21] than traditional moving average [Izm+18]. To increase diversity across averaged models, [Mad+19] used a high learning rate; [Ben+21] encouraged the weights to encompass more volume; [Wor+21] minimized cosine similarity between weights; [Izm+19] used a tempered posterior. In particular, the concurrent “model soups” introduced by Wortsman et al. [Wor+22a] two months before DiWA follows a similar strategy: yet, the theoretical analysis and the goals of these two works are different. Regarding the motivation, DiWA aims at combining diverse weights, and proposes a general framework to average weights obtained in various ways. In contrast, [Wor+22a] challenges the standard model selection after a grid search. Regarding the task, [Wor+22a] and DiWA complement each other: while [Wor+22a] demonstrate robustness on several ImageNet variants, we improve the sota on the multi-domain DomainBed benchmark against other established OOD methods (Section 4.6.1). Thus, DiWA and [Wor+22a] are theoretically complementary with different motivations and applied successfully for different tasks.

4.6 Experiments

4.6.1 Experimental setup

Datasets. We first show DiWA improves performance on DomainBed [Gul+21], previously described in Section 2.1.3, including 5 multi-domain real-world datasets: PACS [Li+17], VLCS [Fan+13], OfficeHome [Ven+17], TerraIncognita [Bee+18] and DomainNet [Pen+19]. Critically, [Ye+22] showed that diversity shift dominates in these datasets. Each domain is successively considered as the target $T$ while other domains are merged into the source $S$. The validation dataset is sampled from $S$.

Baselines. ERM is the standard empirical risk minimization. CORAL [Sun+16] is the best approach based on domain invariance. SWAD (Stochastic Weight Averaging Densely) [Cha+21a] and MA [Arp+21] average weights along one training trajectory but differ in their weight selection strategy. Specifically, SWAD [Cha+21a] uses a “loss-aware” strategy involving three additional hyperparameters (a patient parameter, an overfitting patient parameter and a tolerance rate); in contrast, MA [Arp+21] is easy to implement as it simply combines all checkpoints uniformly starting from batch 100 until the end of training. We also report the scores obtained in [Arp+21] for the costly deep ensembling (DE∗) [Lak+17] of $M = 6$ models with different classifier initializations, where the symbol “∗” marks the large inference overhead.
Our runs. ERM and DiWA share the same training protocol in DomainBed: yet, instead of keeping only one run from the grid-search, DiWA averages $M$ weights. In practice, we sample 20 configurations from the mild hyperparameter distributions detailed in Table 2.1 and report the mean and standard deviation across 3 data splits. For each run, we select the weights of the epoch with the highest ID validation accuracy. The ENS* averages the predictions of all $M = 20$ models (with shared initialization). DiWA-greedy selects $1 \leq M \leq 20$ weights while DiWA-uniform averages all $M = 20$ weights. DiWA† averages uniformly the $M = 3 \times 20 = 60$ weights from all 3 data splits.

4.6.2 Results on DomainBed

We report our main results in Table 4.1: best results are in **bold** and the second best are underlined. With a randomly initialized classifier, DiWA†-uniform is the best on PACS, VLCS and OfficeHome: DiWA-uniform is the second best on PACS and OfficeHome. On Terraincognita and DomainNet, DiWA is penalized by some bad runs, filtered in DiWA-greedy which improves results on these datasets. Classifier initialization with LP [Kum+22] improves all methods on OfficeHome, Terraincognita and DomainNet. On these datasets, DiWA† increases MA by 1.3, 0.5 and 1.1 points respectively. After averaging, DiWA† with LP reaches 68.0%, improving SWAD by 1.1 points.

Table 4.1. – Accuracies (%) on DomainBed [Gul+21] benchmark evaluating OOD generalization. The classifiers are initialized randomly or with linear probing (LP) [Kum+22]. The symbol “∗” indicates inference overhead in functional ensembling. The symbol “†” indicates the averaging of all weights across 3 data splits.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Weight selection</th>
<th>Init</th>
<th>PACS</th>
<th>VLCS</th>
<th>OfficeHome</th>
<th>TerraInc</th>
<th>DomainNet</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERM</td>
<td>ID val</td>
<td>85.5 ± 0.2</td>
<td>77.5 ± 0.4</td>
<td>66.5 ± 0.3</td>
<td>46.1 ± 1.8</td>
<td>40.9 ± 0.1</td>
<td>61.3</td>
<td></td>
</tr>
<tr>
<td>CORAIL [Sun+16]</td>
<td>ID val</td>
<td>86.2 ± 0.3</td>
<td>78.8 ± 0.6</td>
<td>68.7 ± 0.3</td>
<td>47.6 ± 1.0</td>
<td>41.5 ± 0.1</td>
<td>64.6</td>
<td></td>
</tr>
<tr>
<td>SWAD [Cha+21]</td>
<td>Loss-aware Random</td>
<td>88.1 ± 0.1</td>
<td>79.1 ± 0.1</td>
<td>70.6 ± 0.2</td>
<td>50.0 ± 0.3</td>
<td>48.5 ± 0.1</td>
<td>66.9</td>
<td></td>
</tr>
<tr>
<td>MA* [Arp+21]</td>
<td>Uniform</td>
<td>87.5 ± 0.2</td>
<td>78.2 ± 0.2</td>
<td>70.6 ± 0.1</td>
<td>50.3 ± 0.5</td>
<td>46.0 ± 0.1</td>
<td>66.5</td>
<td></td>
</tr>
<tr>
<td>DE* [Arp+21]</td>
<td>Uniform: $M = 6$</td>
<td>87.6</td>
<td>78.5</td>
<td>70.8</td>
<td>49.2</td>
<td>47.7</td>
<td>66.8</td>
<td></td>
</tr>
</tbody>
</table>

Our runs.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Weight selection</th>
<th>Init</th>
<th>PACS</th>
<th>VLCS</th>
<th>OfficeHome</th>
<th>TerraInc</th>
<th>DomainNet</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERM</td>
<td>ID val</td>
<td>85.5 ± 0.5</td>
<td>77.6 ± 0.2</td>
<td>67.4 ± 0.6</td>
<td>48.3 ± 0.8</td>
<td>44.1 ± 0.1</td>
<td>64.6</td>
<td></td>
</tr>
<tr>
<td>MA* [Arp+21]</td>
<td>Uniform</td>
<td>87.9 ± 0.1</td>
<td>78.4 ± 0.1</td>
<td>70.3 ± 0.1</td>
<td>49.9 ± 0.2</td>
<td>46.4 ± 0.1</td>
<td>66.6</td>
<td></td>
</tr>
<tr>
<td>ENS*</td>
<td>Uniform: $M = 20$ Random</td>
<td>88.0 ± 0.1</td>
<td>78.7 ± 0.1</td>
<td>70.5 ± 0.1</td>
<td>51.0 ± 0.5</td>
<td>47.4 ± 0.2</td>
<td>67.1</td>
<td></td>
</tr>
<tr>
<td>DIWA</td>
<td>Greedy: $M \leq 20$</td>
<td>87.9 ± 0.2</td>
<td>79.2 ± 0.1</td>
<td>70.9 ± 0.1</td>
<td>50.5 ± 0.5</td>
<td>46.7 ± 0.1</td>
<td>67.0</td>
<td></td>
</tr>
<tr>
<td>DIWA</td>
<td>Uniform: $M = 20$</td>
<td>88.8 ± 0.4</td>
<td>79.1 ± 0.2</td>
<td>71.0 ± 0.1</td>
<td>49.9 ± 0.5</td>
<td>46.1 ± 0.1</td>
<td>66.8</td>
<td></td>
</tr>
<tr>
<td>DIWA†</td>
<td>Uniform*: $M = 60$</td>
<td>89.0</td>
<td>79.4</td>
<td>71.6</td>
<td>49.0</td>
<td>46.3</td>
<td>67.1</td>
<td></td>
</tr>
</tbody>
</table>

4.6.3 Empirical validation of theoretical insights

We now validate key findings about the empirical similarity between WA and ENS, and then about the role of diversity in WA. To this end, we consider several collections of weights $\{\theta_i\}_{i=1}^M$ ($2 \leq M < 10$) trained on the “Clipart”, “Product” and “Photo” domains from OfficeHome [Ven+17] with a shared random initialization and mild hyperparameter
ranges. These weights are indifferently sampled from a single run (every 50 batches) or from different runs. They are evaluated on “Art”, the fourth domain from OfficeHome considered as OOD.

4.6.3.1 WA vs. ENS

Figure 4.5 validates Lemma 4.1 and that weight averaging (WA) and ensembling (ENS) perform similarly: most dots are close to the diagonal. Moreover, a larger $M$ improves results, motivating averaging as many weights as possible, consistently with Equation (BVCL). WA has a fixed inference time which allows it to consider larger $M$. In contrast, the functional ensembling ENS of $M$ models require $M$ forwards at inference; thus ENS is computationally impractical for large $M$.

Actually, we observe that WA slightly but consistently beats ENS, (i) in OOD (ii) when weights share the same initialization and (iii) hyperparameters are sampled from mild ranges. We provide a preliminary explanation to this surprising phenomenon in Section 7.2.2. Critically, in Table 4.2, we show that this is no longer the case when we relax the two last constraints. First, when the classifiers’ initializations vary, ENS improves thanks to this additional diversity; in contrast, DiWA degrades because weights are no longer averageable. Second, when the hyperparameters are sampled from extreme ranges (defined in Table 2.1), performance drops significantly for DiWA, but much less for ENS. This highlights a limitation of DiWA, which requires weights that are linearly mode connected. In contrast, ENS are computationally expensive (and even impractical for large $M$), but can leverage additional sources of diversity, such as diverse initializations and extreme hyperparameters.

Figure 4.5. – Each dot displays the OOD accuracy ($\uparrow$) of WA vs. ENS when combining $M$ models.

Table 4.2. – DiWA vs. ENS on domain “Art” from OfficeHome when varying initialization and hyperparameter ranges. Best accuracy ($\%$, $\uparrow$) on each setting is in bold.
4.6.3.2 Diversity and accuracy

We validate in Figure 4.6 that \( f_{WA} \) benefits from diversity. Here, we measure diversity with the ratio-error [Aks03], previously defined in Section 2.3.3.1. A higher average over the \( \binom{M}{2} \) pairs means that members are less likely to err on the same inputs thus more diverse. Specifically, the gain of \( Acc(\theta_{WA}) \) over the mean individual accuracy \( \frac{1}{M} \sum_{i=1}^{M} Acc(\theta_i) \) increases with diversity. For example, when \( M = 9 \) weights are averaged, the accuracy gain increases by 0.297 per unit of additional diversity in prediction.

Moreover, this phenomenon intensifies for larger \( M \). In Figure 4.7, we indicate the slope of the linear regressions relating diversity to accuracy gain at fixed \( M \) (between 2 and 9). We observe that the linear regression’s slope (i.e., the accuracy gain per unit of diversity) increases with \( M \). This confirms that diversity becomes more important for large \( M \), consistently with the \( (M−1)/M \) factor in front of \( cov(x) \) in Equation (BVCL).

![Figure 4.6](image1.png)  Each dot displays the accuracy (↑) gain of WA over its members vs. the prediction diversity [Aks03] (↑) for \( M \) models.

![Figure 4.7](image2.png)  The slopes relating diversity (in prediction [Aks03] or in features [Kor+19]) to accuracy gain, increases with \( M \).

4.6.3.3 Increasing diversity thus accuracy via different runs.

Now we investigate the difference between sampling the weights from a single run or from different runs. Figure 4.8 first shows that diversity increases when weights come from different runs. Second, this diversity gain is reflected on the OOD accuracies in Figure 4.9; here, we rank by validation accuracy the 60 weights obtained (i) from 60 different runs and (ii) along 1 well-performing run. We then consider the WA of the top \( M \) weights as \( M \) increases from 1 to 60. Both have initially the same performance and improve with \( M \); yet, WA of weights from different runs gradually outperforms the single-run WA. Finally, Figure 4.10 shows that this holds only for mild hyperparameter ranges and with a shared initialization. Otherwise, when hyperparameter distributions are extreme (as defined in Table 2.1) or when classifiers are not similarly initialized, DiWA may perform worse than its members due to a violation of the locality condition. These experiments confirm that diversity is key as long as the weights remain averageable.
4.6.3.4 Failure under correlation shift

Our theory from Section 4.3 suggests that WA can tackle diversity shift (as previously verified) but not correlation shift; combining multiple models should be inefficient when bias dominates (see Section 2.2.2). We verify this failure on the ColoredMNIST [Arj+19] dataset, which is dominated by correlation shift [Pen+19]. Indeed, Colored MNIST is a variant of the MNIST handwritten digit classification dataset [LeC+10] where the correlation strengths between color and label vary across domains. We follow the DomainBed [Gul+21] protocol, with the small CNN architecture specialized for MNIST experiments, and the test-domain model selection argued in [Ye+22] (see the published paper [Ram+22b] for similar results with the train-domain model selection).

In Table 4.3, we observe that DiWA-uniform and MA both perform poorly compared to ERM, confirming the failure of WA strategies under correlation shift. Note that DiWA-greedy does not degrade ERM as it selects only a few models for averaging.

Table 4.3. – Accuracy (%) on ColoredMNIST. WA does not improve performance under correlation shift. Random initialization of the classifier. Test-domain model selection.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Weight selection</th>
<th>+90%</th>
<th>+80%</th>
<th>-90%</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERM</td>
<td>OOD val</td>
<td>71.8 ± 0.4</td>
<td>72.9 ± 0.1</td>
<td>28.7 ± 0.5</td>
<td>57.8 ± 0.2</td>
</tr>
<tr>
<td>Coral [Sun+16]</td>
<td>OOD val</td>
<td>71.1 ± 0.2</td>
<td>73.4 ± 0.2</td>
<td>31.1 ± 1.6</td>
<td>58.6 ± 0.5</td>
</tr>
<tr>
<td>IRM [Arj+19]</td>
<td>OOD val</td>
<td>72.0 ± 0.1</td>
<td>72.5 ± 0.3</td>
<td>58.5 ± 3.3</td>
<td>67.7 ± 1.2</td>
</tr>
<tr>
<td>Fishr [Ram+22a]</td>
<td>OOD val</td>
<td>74.4 ± 0.6</td>
<td>73.3 ± 0.1</td>
<td>58.9 ± 3.7</td>
<td>68.8 ± 1.4</td>
</tr>
<tr>
<td>ERM</td>
<td>N/A</td>
<td>71.5 ± 0.3</td>
<td>74.1 ± 0.4</td>
<td>21.5 ± 1.9</td>
<td>55.7 ± 0.4</td>
</tr>
<tr>
<td>MA [Arp+21]</td>
<td>Uniform</td>
<td>68.8 ± 0.2</td>
<td>72.1 ± 0.2</td>
<td>10.2 ± 0.0</td>
<td>50.4 ± 0.1</td>
</tr>
<tr>
<td>ENS†</td>
<td>Uniform: M = 20</td>
<td>71.0 ± 0.2</td>
<td>72.9 ± 0.2</td>
<td>9.9 ± 0.0</td>
<td>51.3 ± 0.1</td>
</tr>
<tr>
<td>DiWA</td>
<td>Greedy: M ≤ 20</td>
<td>71.9 ± 0.4</td>
<td>73.6 ± 0.2</td>
<td>21.5 ± 1.9</td>
<td>55.7 ± 0.8</td>
</tr>
<tr>
<td>DiWA</td>
<td>Uniform: M = 20</td>
<td>69.1 ± 0.8</td>
<td>72.5 ± 0.4</td>
<td>10.6 ± 0.1</td>
<td>50.8 ± 0.4</td>
</tr>
<tr>
<td>DiWA†</td>
<td>Uniform: M = 60</td>
<td>69.3</td>
<td>72.3</td>
<td>10.3</td>
<td>50.6</td>
</tr>
</tbody>
</table>
4.6.4 DiWA with different objectives.

So far we used ERM without leveraging the domain information during training. Table 4.4 shows that DiWA-uniform benefits from weights fine-tuned with Interdomain Mixup [Yan+20a] and CORAL [Sun+16]: diversity increases and accuracy improves as we add more diverse objectives. This suggests a new kind of LMC across models fine-tuned with various objectives and losses; this idea will be further explored in rewarded soups [Ram+23b] detailed in Chapter 6.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>No WA</th>
<th>MA</th>
<th>DiWA</th>
<th>DiWA†</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERM</td>
<td>62.9 ± 1.3</td>
<td>65.0 ± 0.2</td>
<td>67.3 ± 0.2</td>
<td>67.7</td>
</tr>
<tr>
<td>Mixup</td>
<td>63.1 ± 0.7</td>
<td>66.2 ± 0.3</td>
<td>67.8 ± 0.6</td>
<td>68.4</td>
</tr>
<tr>
<td>CORAL</td>
<td>64.4 ± 0.4</td>
<td>64.4 ± 0.4</td>
<td>67.7 ± 0.2</td>
<td>68.2</td>
</tr>
<tr>
<td>ERM/Mixup</td>
<td>N/A</td>
<td>N/A</td>
<td>67.9 ± 0.7</td>
<td>68.9</td>
</tr>
<tr>
<td>ERM/CORAL</td>
<td>N/A</td>
<td>N/A</td>
<td>68.1 ± 0.3</td>
<td>68.7</td>
</tr>
<tr>
<td>ERM/Mixup/CORAL</td>
<td>N/A</td>
<td>N/A</td>
<td>68.4 ± 0.4</td>
<td>69.1</td>
</tr>
</tbody>
</table>

4.7 Conclusion

In this chapter, we highlighted the limitations of the previously dominant flatness-based analysis of WA. We proposed a novel bias-variance-diversity-locality analysis, leveraging the ensembling nature of WA, and underscoring a diversity-locality trade-off. This motivates our DiWA approach; by averaging the weights of independently trained models, DiWA improves performances on DomainBed. DiWA is a simple and practical strategy, with a straightforward implementation, making it a valuable tool for a broad range of real-world applications. Essentially, rather than selecting the best model from a hyperparameter search, it suggests that superior results can be obtained by averaging all the fine-tuned weights. Importantly, DiWA is without inference overhead, thereby removing a key limitation of standard ensembling.

Despite its advantages, certain challenges persist. Firstly, DiWA necessitates multiple independent training runs, hence does not curtail the training cost of ensembling. However, this issue might be considered minor as fine-tunings are typically faster than trainings from scratch. Secondly, as seen in Section 4.6.3.1, the shared initialization constraint limits diversity. Compared to functional ensembling that can combine arbitrary architectures, this may be problematic in contexts where multiple foundation models are available. Neuron permutations strategies [Ent+22; Ain+22] tried to enforce connectivity across weights fine-tuned from different initializations, though (so far) with moderate empirical results. Our subsequent Chapter 5 proposes an alternative to relax the shared initialization constraint.
RATATOUILLE: RECYCLING DIVERSE MODELS FOR OUT-OF-DISTRIBUTION GENERALIZATION

5.1 Introduction

Learning robust models that generalize well is critical for many real-world applications [Zec+18; DeG+21]. Fine-tuning pre-trained foundation models [Bom+21] is now the most popular approach to build deep learning (DL) solutions for these applications. For fine-tuning, empirical risk minimization (ERM) [Vap92] has long remained the best strategy on the reference DomainBed [Gul+21] benchmark. Yet, as previously introduced in Chapter 4, the ability to average neural networks’ weights inspired a plethora of modern weight averaging (WA) approaches for fine-tuning. We illustrate some of them in Figure 5.1, such as MA [Izm+18], WiSE fine-tuning [Wor+22b], and our DiWA [Ram+22b].

Fostered by the open-source philosophy in ML, the Internet is then swarmed by a handful of foundation models fine-tuned on many diverse tasks: these individual fine-tunings are available on public repositories such as torchvision [Mar+10], huggingface [Wol+20], or timm [Wig19], yet 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. Recent inter-training [Pha+18; Pru+20] and fusing [Cho+22b; Don+23a] 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. Moreover, as argued in Section 5.2, these strategies fail to fully leverage the diversity in auxiliary tasks.

Thus, the central question of this chapter 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 model ratatouille 1, a simple fine-tuning strategy illustrated in Figure 5.1 and described in Section 5.3. 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 task. Specifically, we (i) fine-tune a copy of the foundation model on each auxiliary task,

---

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 [Mon20].
Figure 5.1. – The different fine-tuning strategies discussed in this chapter: vanilla fine-tuning [Oqu+14], moving average (MA) [Izm+18] and variants [Wor+22b], DiWA [Ram+22b] introduced in Chapter 4 and the similar model soups [Wor+22a], inter-training [Pha+18], fusing [Cho+22b] and our novel model ratatouille. They 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, (ii) maximizes the amount of diversity in models’ predictions, (iii) achieves sota performance in DomainBed [Gul+21], the standard computer vision benchmark for OOD generalization and (iv) does not incur any inference or training overhead compared to a traditional hyperparameter search.

(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 DiWA and model soups average multiple weights fine-tuned from a shared initialization, model ratatouille averages multiple weights fine-tuned from different initializations each inter-trained [Pha+18] on different auxiliary tasks. As we will see, ratatouille works because the fine-tunings remain linearly mode connected (LMC) [Fra+20; Mir+21] in the loss landscape (despite having different initializations) and thus their average improves performance.

We show the efficacy of model ratatouille in Section 5.4, where we set a new sota on DomainBed [Gul+21]. We will show how we leverage the diversity across the auxiliary tasks to construct diverse weights, that can be averaged into a final model. Looking forward, as we discuss in Section 7.2.3, this chapter contributes to the emerging paradigm of updatable machine learning [Raf23], where the community collaborates towards incrementally and reliably updating the capabilities of a ML model.


5.2 Context

We start by describing our setup. We train a deep model \( f \), where the featurizer is parametrized by the weights \( \phi \), the classifier is parametrized by the weights \( \omega \), and thus
5.2 CONTEXT

$f$ is parametrized by the concatenation weights $\theta = (\omega, \phi)$. We are dealing with out-of-distribution (OOD) generalization, and our aim is to find $\theta$ maximizing the test accuracy $\text{acc}_T(\theta)$. In this chapter, we only consider diversity shifts [Ye+22].

5.2.1 ERM and WA

The recipe in transfer learning [Oqu+14; Kir+22] is (i) download a pre-trained featurizer with parameters $\phi^{pt}$, (ii) plug a classifier $\omega^{lp}$ compatible with the target task, and (iii) fine-tune the network with empirical risk minimization (ERM) [Vap92] on the target task. While the classifier $\omega^{lp}$ could be initialized at random, linear probing (LP) (i.e., first learning only the classifier with frozen featurizer) improves results by preventing feature distortion [Kum+22]. Based on Observation 4.1, moving average (MA) strategies and variants [Sze+16; Izm+18; Dra+18; Arp+21; Cha+21a; Wor+22b] average the weights of checkpoints collected every few epochs to build the final weights. More recently, as explained in previous Chapter 4 and consistently with Observation 4.2 from [Ney+20], our DiWA [Ram+22b] proposed to average all the weights obtained from a standard ERM hyperparameter search. DiWA and the similar model soups [Wor+22a] are so far the best approaches for OOD generalization. However, as highlighted in Section 4.7, the shared initialization constraint in DiWA limits models diversity [Kun+03; Aks+03]; removing this constraint would be desirable to further reduce variance.

5.2.2 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+21] that match our target task, ease optimization [Zha+22; Zha+23a], or “offer some high-level guidance to bridge the gaps between the pre-training and fine-tuning phases” [Cha+21b]. Following these ideas, inter-training approaches [Pha+18; Pru+20; Cho+22a] perform 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 [Reb+17] 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” [Cho+22b]. To address the shortcomings of inter-training, recent works [Cho+22b; Don+23a; Li+22; Mat+22; Ilh+23; Ilh+22] recycle weights fine-tuned on various auxiliary tasks. In particular, concurrent [Cho+22b] 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 weight interpolation, fusing combines into one single initialization the knowledge from multiple auxiliary tasks; yet fusing provides only marginal empirical gains in Section 5.4.1 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 this, 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.

### 5.3 Model ratatouille

#### 5.3.1 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 5.1 and outlined in detail in Figure 5.2(b). Ratatouille recycles these fine-tunings as diverse initializations to parallel fine-tunings on the target task. Compared to fusing [Cho+22b], we delay the weight averaging, and in turn the destruction of diversity. Ratatouille follows this five-step recipe.

1. Download a featurizer $\phi_{pt}$ pre-trained on task $D_0$.
2. Fine-tune $\phi_{pt}$ on each auxiliary task $D_i$, obtaining $(\omega_i^{aux}, \phi_i^{aux})$ for $i = 0, \ldots, M - 1$. 

Figure 5.2. – Illustrations of (a) different linear mode connectivity (LMC) conditions, and (b) model ratatouille. In subplot (a), we illustrate Observation 4.1, about LMC between two checkpoints along the same target fine-tuning; Observation 4.2, about LMC between two target fine-tunings; Hypothesis 5.1, about LMC between two auxiliary fine-tunings; and Hypothesis 5.2, about LMC between two target fine-tunings initialized from auxiliary weights satisfying Hypothesis 5.1. In subplot (b), we offer a diagram of our proposed ratatouille strategy, where we (i) fine-tune a pre-trained model on auxiliary tasks, (ii) plug a linear probe classifier 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.
3. Replace each $\omega_{i}^{\text{aux}}$ by $\omega_{i}^{lp}$, obtained by linear probing the original pre-trained model $\phi^{pt}$ on the target task $D$.
4. Fine-tune each $(\omega_{i}^{lp}, \phi_{i}^{aux})$ on the target task $D$, obtaining $\theta_{i} = (\omega_{i}, \phi_{i})$ for $i = 0, \ldots, M - 1$.
5. Return as final model $P_{M-1} = \sum_{i=0}^{M-1} \lambda_{i} \cdot \theta_{i}$. To select the interpolating coefficients, we use the same two weight selection strategies previously described in Section 4.4.4. The first uniform averages all weights with $\lambda_{i} = \frac{1}{M}$. The second greedy sorts the $\theta_{i}$ by descending ID validation accuracy, before greedily constructing an uniform average containing $\theta_{i}$ 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 [Pha+18] and fusing [Cho+22b], model ratatouille avoids the difficult choice of choosing one single initialization [Cho+22a]. The shared LP classifier facilitates LMC by preventing feature distortions [Kum+22]. Note that we consider the pre-training task as the auxiliary task “number zero” $D_{0}$; this resembles WiSE fine-tuning [Wor+22b] and aims at preserving the general-purpose knowledge contained in the original pre-trained model.

In essence, auxiliary tasks helps in two ways: through their similarity with the target task, and their diversity. As described in previous Section 4.3, successful weight averaging (WA) requires three conditions [Ram+22b]. The weights should be:

1. individually accurate (to reduce the bias); by inter-training, ratatouille enriches the features and thus increases individual accuracies when the auxiliary tasks are well-chosen [Cho+22a].
2. sufficiently diverse (to reduce variance): by removing the shared initialization constraint from DiWA, ratatouille benefits from the additional diversity brought by specialization on various auxiliary tasks.
3. averageable; for ratatouille to work, it requires a relaxation of the conditions under which the LMC holds, that we detail below.

### 5.3.2 Novel linear mode connectivity hypotheses

We now extend Observations 4.1 and 4.2 made in Chapter 4 when including fine-tunings on auxiliary tasks. First, we introduce Hypothesis 5.1 that posits LMC between two models whose featurizers were fine-tuned on different auxiliary tasks.

**Hypothesis 5.1** (LMC with different tasks). The LMC holds between $(\omega_{1}^{lp}, \phi_{1}^{aux})$ and $(\omega_{2}^{lp}, \phi_{2}^{aux})$ if $\phi_{1}^{aux}$ and $\phi_{2}^{aux}$ are featurizers fine-tuned on two auxiliary tasks initialized from the same pre-trained featurizer $\phi^{pt}$. Here, $\omega_{i}$ is the linear probe of $\phi^{pt}$ on the target task.

Though this Hypothesis 5.1 was never formulated explicitly, it is underlying in fusing [Cho+22b] and in other works averaging auxiliary weights. Actually, ratatouille relies on the following Hypothesis 5.2, which enriches Hypothesis 5.1 with additional independent fine-tunings on the target task.
Hypothesis 5.2 (LMC with different auxiliary initializations). The LMC holds between $\theta_1$ and $\theta_2$ fine-tuned on the target task starting from initializations $(\omega_{LP}^{1}, \phi_{aux}^{1})$ and $(\omega_{LP}^{2}, \phi_{aux}^{2})$ satisfying Hypothesis 5.1.

Hypothesis 5.2 is the first to posit the LMC between weights fine-tuned 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 5.1 and 5.2 to hold as long as the pre-training, auxiliary and target tasks are sufficiently similar, and if hyperparameters remain in a mild range. If these LMC hold, then we expect ratatouille to improve generalization abilities. But this, we can only answer empirically through proper experimentation in the upcoming Section 5.4.

5.4 Experiments

Our numerical experiments support five main claims, sorted in decreased granularity. First, Section 5.4.1 showcases the sota results of ratatouille in DomainBed [Gul+21]. Second, Section 5.4.2 illustrates how such gains arise from increased diversity across averaged models. Third, Section 5.4.3 empirically supports Hypotheses 5.1 and 5.2, the LMC conditions enabling weight averaging’s success. Then, Section 5.4.4 shows that additional auxiliary tasks improve results. Finally, Section 5.4.5 discusses the impact of ratatouille for in-domain tasks.

5.4.1 Sota performance on DomainBed

Setup. Table 5.1 shows our main experiment comparing the various fine-tuning strategies on DomainBed [Gul+21]. We follow the same procedure previously described in Section 4.6.1. Our goal is to improve the performances previously obtained by DiWA [Ram+22b], a.k.a. model soups [Wor+22a]. As a reminder, DiWA only differs from ERM vanilla fine-tuning by the selection strategy: rather than selecting the model with highest ID validation accuracy out of the 20 runs, DiWA either uniformly averages all weights or greedily selects some. The key originality of ratatouille is to leverage auxiliary trainings; in practice, 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 DiWA is to vanilla fine-tuning. As in previous Section 4.6.1, the “†” symbol marks methods averaging 60 = 20×3 weights from 3 data splits. We further discuss ratatouille’s training cost in [Ram+23a]. The procedure to obtain the pool of initializations is agnostic to the target task or the test domain, and thus is done only once; in particular, we argue that ratatouille is without training overhead when auxiliary weights are shared by the community. Functional ensembling strategies (marked by the symbol “∗”) average predictions with large inference overhead: for example, “ENS∗ of inter-training” averages the predictions of the $M = 20$ models
Table 5.1. – **Accuracies (%)† on DomainBed** [Gul+21] benchmark evaluating OOD generalization. Ratatouille sets a new sota by leveraging diversity in auxiliary tasks. The symbol “†” indicates inference overhead in functional ensembling. The symbol “±” indicates the averaging of all weights across 3 data splits. The scores for DiWA are those from Table 4.1 with LP initialization.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Weight selection</th>
<th>PACS</th>
<th>VLCS</th>
<th>OfficeHome</th>
<th>TerraInc</th>
<th>DomainNet</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla fine-tuning</td>
<td>ID val</td>
<td>85.5 ± 0.2</td>
<td>77.5 ± 0.4</td>
<td>66.5 ± 0.3</td>
<td>46.1 ± 1.8</td>
<td>40.9 ± 0.1</td>
<td>63.3</td>
</tr>
<tr>
<td>CORAL [Sun+16]</td>
<td>ID val</td>
<td>86.2 ± 0.3</td>
<td>78.8 ± 0.6</td>
<td>68.7 ± 0.3</td>
<td>47.6 ± 1.0</td>
<td>41.5 ± 0.1</td>
<td>64.6</td>
</tr>
<tr>
<td>SWAD [Cha+21a]</td>
<td>Loss-aware</td>
<td>88.1 ± 0.1</td>
<td>79.1 ± 0.1</td>
<td>70.6 ± 0.2</td>
<td>50.0 ± 0.3</td>
<td>46.5 ± 0.1</td>
<td>66.9</td>
</tr>
<tr>
<td>MA [Arp+21]</td>
<td>Uniform</td>
<td>87.5 ± 0.2</td>
<td>78.2 ± 0.2</td>
<td>70.6 ± 0.1</td>
<td>50.3 ± 0.5</td>
<td>46.0 ± 0.1</td>
<td>66.5</td>
</tr>
<tr>
<td>DE* [Arp+21]</td>
<td>Uniform: M = 6</td>
<td>87.6</td>
<td>78.5</td>
<td>70.8</td>
<td>49.2</td>
<td></td>
<td>47.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>66.8</td>
</tr>
</tbody>
</table>

|                       | ID val           | 85.9 ± 0.6 | 78.1 ± 0.5 | 69.4 ± 0.2 | 50.4 ± 1.8 | 44.3 ± 0.2 | 65.6 |
|                       | Uniform: M = 20  | 88.1 ± 0.3 | 78.5 ± 0.1 | 71.7 ± 0.1 | 50.8 ± 0.5 | 47.0 ± 0.2 | 67.2 |
|                       | Uniform: M = 20  | 88.7 ± 0.2 | 78.4 ± 0.2 | 72.1 ± 0.2 | 51.4 ± 0.6 | 47.4 ± 0.2 | 67.6 |
|                       | Greedy: M ≤ 20   | 88.0 ± 0.3 | 78.5 ± 0.1 | 71.5 ± 0.2 | 51.6 ± 0.9 | 47.7 ± 0.1 | 67.5 |
|                       | Uniform*: M = 60 | 89.0     | 78.6     | 72.8     | 51.9     |           | 47.7 |
|                       |                  |         |         |           |          |           | 68.0 |
| Inter-training [Pha+18]  | ID val           | 89.0 ± 0.0 | 77.7 ± 0.0 | 69.9 ± 0.6 | 46.7 ± 0.1 | 44.5 ± 0.1 | 65.6 |
|                       | Uniform: M = 20  | 89.2 ± 0.1 | 79.0 ± 0.2 | 72.7 ± 0.1 | 51.1 ± 0.3 | 47.2 ± 0.1 | 67.8 |
|                       | ID val           | 88.0 ± 1.0 | 78.5 ± 0.8 | 71.5 ± 0.5 | 46.7 ± 1.8 | 44.4 ± 0.2 | 65.8 |
|                       | Uniform: M = 20  | 89.5 ± 0.1 | 78.5 ± 0.1 | 73.1 ± 0.1 | 51.8 ± 0.4 | 47.5 ± 0.1 | 68.1 |
|                       | Greedy: M ≥ 20   | 90.5 ± 0.2 | 78.7 ± 0.2 | 75.4 ± 0.3 | 49.2 ± 0.9 | 47.7 ± 0.0 | 67.9 |
|                       | Uniform*: M = 60 | 89.8     | 78.3     | 73.5     | 52.0     |           | 47.7 |
|                       |                  |         |         |           |          |           | 68.3 |

Ratatouille-uniform averages in weights. For fusing [Cho+22b], each run is initialized from $\sum_{i=0}^{4} \lambda_i \phi^\text{aux}_i$ where the $\lambda_i$ hyperparameters sum to 1 and $\phi^\text{aux}$ are inter-trained on the 4 other DomainBed’s datasets or directly transferred from ImageNet.

**Results.** Table 5.1 shows that ratatouille achieves a new sota on DomainBed: with uniform selection, it achieves 68.1 and improves DiWA by 0.5 points after averaging over all datasets. Precisely, model ratatouille beats DiWA 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-uniform (51.8%) matches DiWA-uniform (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, except on one domain where the LMC breaks (as shown in [Ram+23a]). For DomainNet, ratatouille is sota though the gains are small w.r.t. DiWA: we suspect this is because the initialization strategy becomes less critical for larger datasets [Cha+21b] with more training epochs (see Figure 5.3(b)). 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 DiWA, which works best with appropriate hyperparameter ranges; DiWA removed the need to select only the best set of hyperparameters.
Figure 5.3. Explorations on Q-diversity [Kun+03] 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 \(\mu\) are inter-trained on DomainNet. The accuracy of the weight average is maximized when \(\mu \approx 0.5\).

5.4.2 Increased diversity by recycling

In Figure 5.3, we investigate how the diversity across models fine-tuned on the target task influences the OOD performance of their WA. Here, we measure diversity with the prediction Q-diversity [Kun+03], previously introduced in Section 2.3.3.1, which increases when models fail on different examples; in the paper, we also arrive at similar conclusions using the ratio-error [Akso3]. We follow the same protocol as in Section 4.6.3, and consider OfficeHome as the target task, 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.

First, we verify that inter-training influences the diversity across fine-tuned models. Specifically, Figure 5.3(a) confirms that networks with different initializations are more diverse than networks initialized similarly. Then, Figure 5.3(b) verifies that this diversity gain comes from their initialization and remains along fine-tuning on the target task. Moreover, Figure 5.3(c) shows that diversity is positively linearly correlated with OOD generalization: having different initializations improves diversity and thus WA accuracy. Finally, in Figure 5.3(d), we consider averaging \(M\) weights: a proportion \((1 - \mu)\) start directly from ImageNet, the others \(\mu\) are inter-trained on DomainNet. In the simplest case \(M = 2\), using one model from each initialization leads to maximum accuracy; for larger \(M\), best performances are obtained around \(\mu \approx 0.5\), where the final WA has access to diverse initializations. In conclusion, each auxiliary task fosters the learning of diverse features [Li+21; Gon+22]. Model ratatouille increases diversity and improves performance by removing a key limitation of model soups [Wor+22a] and DiWA [Ram+22b]: the need for all fine-tunings to start from a shared initialization.
5.4 Experiments

In Figure 5.4, we continue our experiments by validating Hypotheses 5.1 and 5.2 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 [Tay+19] and Camelyon [Koh+21] from the WILDS [Koh+21] benchmark. For each target task, we consider the first domain as the test OOD; the other domains are used for training.

We validate Hypothesis 5.1 in Figures 5.4(a) to 5.4(e). For each dataset, we plot the test OOD accuracy for the weights \((w^\text{lp}, (1 - \lambda) \cdot \phi_1^\text{aux} + \lambda \cdot \phi_2^\text{aux})\), where the classifier \(w^\text{lp}\) is a linear probe of the ImageNet pre-trained featurizer \(\phi^\text{pt}_\text{IM}\), and \(\lambda \in [0, 1]\) interpolates between \(\phi_1^\text{aux}\) and \(\phi_2^\text{aux}\) obtained by fine-tuning on two auxiliary tasks initialized from \(\phi^\text{pt}_\text{IM}\). First, we observe that task similarity influences OOD generalization since the test accuracies in Figure 5.4(c) agree with the fact that OfficeHome is most similar to DomainNet, not as similar to Terralncognita, and most dissimilar to the medical dataset RxRx. Second, the accuracy of the interpolated weights is above the interpolated accuracy: this validates Hypothesis 5.1. The accuracy is even usually concave in \(\lambda\).

Similarly, we empirically support Hypothesis 5.2 in Figures 5.4(f) to 5.4(j). For each dataset, we plot the test OOD accuracy obtained with weights \((1 - \lambda) \cdot \phi_1 + \lambda \cdot \phi_2\), where the coefficient \(\lambda \in [0, 1]\) interpolates between \(\phi_1\) and \(\phi_2\), fine-tuned on the target task.
respectively starting from \((w^l, \phi_1^{aux})\) and \((w^l, \phi_2^{aux})\). We observe that Hypothesis 5.2 usually holds: for example, even recycling RxRx can help for OfficeHome on Figure 5.4(h). Yet, Hypothesis 5.2 breaks on TerraIncognita and Camelyon in Figures 5.4(i) and 5.4(j) when RxRx is one of the two auxiliary tasks. In light of these results, we argue that Hypothesis 5.2 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 [Kum+22] and escaping a shared loss valley. Better understanding when LMC breaks is a promising research direction [Jun+23; Lub+22]; among other factors, we speculate that larger pre-training corpus (as in [Qin+22]) or larger architectures (as in [Li+22]) may favor WA strategies.

5.4.4 Analysis of the number of auxiliary tasks

![Graphs showing OOD accuracy for different numbers of auxiliary tasks](image)

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

Figure 5.5. – 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.

In Table 5.1, 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 5.5, 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 etc. This validates that a greater number of auxiliary tasks leads to an increase in expected OOD accuracy. We expect that further increasing the number of auxiliary datasets—beyond those from DomainBed—would further improve results.

5.4.5 Ratatouille for ID tasks

Like previous WA strategies [Izm+18; Wor+22a], model ratatouille also works for in-distribution (ID) tasks; in particular, we verify in Figure 5.6(a) that the LMC holds on the ID validation samples, following the same distribution as the training samples. Yet, the gains are smaller in ID than in OOD, as confirmed by the lack of correlation between
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 $\mu$ 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 [Kun+03] 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.

ID and OOD accuracies [Ten+22] in Figures 5.6(b) and 5.6(c). This is explained by the fact that variance reduction from WA is less beneficial in ID than in OOD. Theoretically, this is because, variance is smaller without distribution shift, as proved in Section 2.2.1. Empirically, this is consistent with models’ diversity being smaller in ID, as shown in Figure 5.6(d). 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 should sacrifice some diversity and select one single optimal initialization. This also explains occasional failures of the greedy selection (notably for TerraIncognita in Table 5.1): based on the ID validation accuracy, only a few runs are selected and averaged, causing smaller OOD accuracy than with the uniform selection.

### 5.5 Conclusion

This chapter introduces model ratatouille, broadening the foundation model paradigm by recycling weights fine-tuned on a variety of auxiliary tasks. Ratatouille extends DiWA by relaxing the shared initialization constraint; the fine-tunings only need to start from initializations that are themselves linearly mode connected. This relaxation allows to average a more diverse set of weights, which in turn leads to better OOD generalization.

The main limitation of this chapter is that we explore OOD generalization in supervised learning scenarios, where objectives are well-defined and ground-truth labels exist. This is not always the case in real-world applications, where defining desired behaviours are not trivial. In the next Chapter 6, we will investigate whether WA can be beneficial for RL tasks, specifically in the context of RLHF with diverse rewards.
Chapter

REWARDED SOUPS: TOWARDS PARETO-OPTIMAL ALIGNMENT BY INTERPOLATING WEIGHTS FINE-TUNED ON DIVERSE REWARDS

6.1 Introduction

Foundation models [Bom+21] are usually pre-trained through self-supervision [Dev+19; Bro+20; Car+21; Rad+21] and then fine-tuned [Oqu+14; Yos+14] via supervised learning [Vap99]. This was the main scenario in previous chapters, where classification labels were properly defined on the target task. Yet, collecting supervised labels is often expensive, and not always possible for more subjective tasks with multiple correct answers or involving human concepts [Kwo+23] such as helpfulness [Bai+22a; Ask+21]; thus supervision may not cover all possibilities, and fail to perfectly align [Amo+16; Tay+16; Ngo+22] the network with the intended applications. Recent works [Sti+20b; Ouy+22; Pin+23] showed that deep reinforcement learning (DRL) helps by learning from various types of rewards. A prominent example is reinforcement learning from human feedback (RLHF) [Sti+20b; Chr+17b; Zie+19; Wu+21], which appears as the current go-to strategy to refine large language models (LLMs) into powerful conversational agents such as ChatGPT [Ouy+22; Ope23]. After pre-training on next token prediction [Rad+18] using Web data, the LLMs are fine-tuned to follow instructions [Wei+22a; Wan+22c; Tao+23] before reward maximization. This third RL step enhances alignment by evaluating the entire generated sentence instead of each token independently [Gol23]. Similar strategies have been useful in computer vision (CV) [Pin+23; Ren+17], for instance to integrate human aesthetics into image generation [Lee+23; Wu+23a; Zha+23b].

Diversity of proxy rewards. RL is usually seen as more challenging than supervised training [Dul+21], notably because the real reward—reflecting the users’ true preferences—is often not specified at training time. Proxy rewards are therefore developed to guide the learning, either as hand-engineered metrics [Pap+02; Lin+03; Ved+15] or more recently in RLHF as models trained to reflect human preferences [Chr+17b; Kwo+23; Xu+23]. Nonetheless, designing reliable proxy rewards for evaluation is difficult. This reward mis-specification [Amo+16; Pan+22] between the proxy reward and the users’ actual rewards can lead to unforeseen consequences [Mic+20]. Moreover, the diversity of objectives in real-world applications exacerbates the issue. In particular, human opinions can vary significantly [Wil87; Coeo00; Sch+12] on subjects such as aesthetics [Nad+19], politics or fairness [Lop+22]. Humans have also different expectations from machines: for example, while [Gan+22] stressed aligning LLMs towards harmless feedback, [Bai+22b] requested
Figure 6.1. – Figure 6.1(a) details the different steps in rewarded soup. After unsupervised pre-training and supervised fine-tuning, we launch $N$ independent RL fine-tunings on the proxy rewards $\{R_i\}_{i=1}^N$. Then we combine the trained networks by interpolation in the weight space. The final weights are adapted at test time by selecting the coefficient $\lambda$. Figure 6.1(b) shows our results (extended in Figure 6.2(a)) with LLaMA-$\theta_7$ [Tou+23a] instruct fine-tuned on Alpaca [Tao+23], when RL fine-tuning for news summarization [Sti+20b] with $N = 2$ reward models assessing diverse preferences: $R_1$ rewards completeness while $R_2$ rewards faithfulness. With only two trainings ($R_1$ and $R_2$ rewarded on Figure 6.1(b)), the $\lambda$-interpolation ($0 \leq \lambda \leq 1$) reveals the green front of Pareto-optimal solutions, i.e., that cannot be improved for one reward without sacrificing the other. RS matches the costly yellow front of MORL [Bar+08; Li+20b] requiring multiple trainings on different linear weightings over the rewards $(1 - \mu) \times R_1 + \mu \times R_2$ with $0 \leq \mu \leq 1$.

helpful non-evasive responses, and others’ [Irv+23] interests are to make LLMs engaging and enjoyable. Even hand-engineered metrics can be in tension: generating shorter descriptions with higher precision can increase the BLEU [Pap+02] score but decrease the ROUGE [Lin+03] score due to reduced recall.

Towards multi-policy strategies. Considering these challenges, a single model cannot be aligned with everyone’s preferences [Ouy+22]. Existing works align towards a consensus-based user [Bak+22; Ova23], relying on the “wisdom of the crowd” [Bai+22a], inherently prioritizing certain principles [Kov+23; Joh+22], resulting in unfair representations of marginalized groups [Wei+21; Kir+23; Dur+23]. The trade-offs [Pan+23] are decided a priori before training, shifting the responsibility to the engineers, reducing transparency and explainability [Hay+22], and actually aligning towards the “researchers designing the study” [Ouy+22; San+23]. These limitations, discussed in Section 6.4, highlight the inability of single-policy alignment strategies to handle human diversity. Yet, “human-aligned artificial intelligence is a multi-objective problem” [Vam+18]. Thus, we draw inspiration from the multi-objective reinforcement learning (MORL) literature [Bar+08; Li+20b; Tan+03; Van+14; Roi+13; Rǎd+20; Mar+23; Wu+23c] and [Hay+22]; they argue that embracing the heterogeneity of diverse rewards requires shifting from single-policy to multi-policy approaches. As optimality depends on the relative preferences across those rewards, the goal is not to learn a single network optimized on a single a...
priori reward, but rather to uncover a set of Pareto-optimal networks [Par64] across the entire space of preferences.

In this chapter, we propose rewarded soup (RS) \(^1\), an efficient and flexible multi-policy strategy to fine-tune any foundation model. As shown in Figure 6.1(a), we first use RL to learn one network for each proxy reward independently; then, we combine these expert networks according to user preferences. This a posteriori selection allows for better-informed trade-offs, improved transparency and increased fairness [Hay+22; Man+21a]. The method to combine those networks is a key contribution: rather than functional ensembling, we perform weight interpolation. This is in line with the previous findings from previous chapters on linear mode connectivity (LMC) [Fra+20; Ney+20]: weights fine-tuned from a shared pre-trained initialization remain linearly connected and thus can be interpolated. Unlike previous chapters which focused on supervised learning, we explore LMC in RL, in a challenging setup where each training run uses a different reward. Then, we can trade off the capabilities of multiple weights in a single final model, without any computational overhead; this enables the creation of custom weights for any preference over the diverse rewards.

- We propose rewarded soup for RL fine-tuning of foundation models when considering diverse rewards (Section 6.2.1).
- We analyze the LMC between weights RL fine-tuned on diverse rewards and the Pareto-optimality of our strategy (Section 6.2.2)
- We demonstrate the effectiveness of rewarded soups across a variety of tasks: RLHF fine-tuning of LLaMA for text-to-text (summarization, QA, helpful assistant, review), multimodal text-image tasks (image captioning, text-to-image generation with diffusion models, visual grounding, VQA), as well as control (locomotion) tasks (Section 6.3).

This chapter has led to the submission of the following paper: Alexandre Ramé, Guillaume Couairon, Mustafa Shukor, Corentin Dancette, Jean-Baptiste Gaya, Laure Soulier, and Matthieu Cord. “Rewarded soups: towards Pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards”. In: NeurIPS. 2023. Further information and resources related can be found on our website.

### 6.2 Rewarded soups

#### 6.2.1 RL fine-tuning with diverse rewards

We consider a deep neural network \( f \) of a fixed non-linear architecture. It defines a policy by mapping inputs \( x \) to \( f(x, \theta) \) when parametrized by \( \theta \). For a reward \( R \) (evaluating the correctness of the prediction according to some preferences) and a test domain \( T \) of de-

\(^1\) The name rewarded soup follows the terminology of model soup [Wor+22a], as we combine various ingredients each rewarded differently.
employment, our goal is to maximize \( \int_{x \in T} \hat{R}(f(x, \theta))^2 \). For example, with \( f \) a LLM, \( x \) would be textual prompts, \( \hat{R} \) would evaluate if the generated text is harmless [Ask+21], and \( T \) would be the distribution of users’ prompts. Learning the weights \( \theta \) is now commonly a three-step process: unsupervised pre-training, supervised fine-tuning, and reward optimization. Yet \( \hat{R} \) is usually not specified before test time, meaning we only have a proxy reward \( R \) during reward optimization. This reward misspecification between \( R \) and \( \hat{R} \) may hinder the alignment of the network with \( \hat{R} \). Moreover, the diversity of human preferences complicates the design of \( \hat{R} \).

Rather than optimizing one single proxy reward, our paper’s first key idea is to consider a family of \( N \) diverse proxy rewards \( \{R_i\}_{i=1}^N \). Each of these rewards evaluates the prediction according to different (potentially conflicting) criteria. The goal then becomes obtaining a coverage set of policies that trade-off between these rewards. To this end, we first introduce the costly MORL baseline. Its inefficiency motivates our rewarded soups, which leverages our second key idea: weight interpolation (WI).

**MORL baseline.** The standard multi-objective reinforcement learning (MORL) scalarization strategy [Bar+08; Li+20b] (recently used in [Wu+23c]) to align LLMs linearizes the problem by interpolating the proxy rewards using \( M \) different weightings. Specifically, during the training phase, \( M \) trainings are launched, with the \( j \)-th optimizing the reward \( \sum_{i=1}^N \mu_i^j R_i \), where \( \forall j \in \{1, ..., M\}, \{\mu_i^j\}_{i=1}^N \in \Delta_N \) the \( N \)-simplex s.t. \( \sum_{i=1}^N \mu_i^j = 1 \) and \( 0 \leq \mu_i^j \leq 1 \). Then, during the selection phase, the user’s reward \( \hat{R} \) becomes known and the \( j \)-th policy that maximizes \( \hat{R} \) on some validation dataset is selected. We typically expect to select \( j \) such that \( \sum_{i=1}^N \mu_i^j R_i \approx \hat{R} \) linearly approximates the user’s reward. Finally, this \( j \)-th weight is used during the inference phase on test samples. Yet, a critical issue is that “minor [preference] variations may result in significant changes in the solution” [Vam+08]. Thus, a high level of granularity in the mesh of \( \Delta_N \) is necessary. This requires explicitly maintaining a large set of \( M \gg N \) networks, practically one for each possible preference. Ultimately, this MORL strategy is unscaleable in DL due to the computational, memory, and engineering costs involved.

**Rewarded soups (RS).** In this chapter, we draw inspiration from previous chapters and leverage weight interpolation. The idea is to learn expert weights and interpolate them linearly to combine their abilities. Specifically, we propose RS, illustrated in Figure 6.1(a) and whose recipe is described below. By design, RS alleviates MORL’s scaling issue as it requires only \( M = N \) trainings, while being flexible and transparent.

1. During the training phase, we optimize \( N \) expert weights \( \{\theta_i\}_{i=1}^N \), each corresponding to one of the \( N \) proxy rewards \( \{R_i\}_{i=1}^N \), from a shared pre-trained initialization.

2. For the selection phase, we linearly interpolate those weights to define a continuous set of rewarded soups policies: \( \{\sum_{i=1}^N \lambda_i \cdot \theta_i\}_{\{\lambda_i\}_{i=1}^N \in \Delta_N} \). Practically, we uniformly sample \( M \) interpolating coefficients \( \{\{\lambda_i^j\}_{i=1}^N\}_{j=1}^M \) from the \( N \)-simplex \( \Delta_N \) and select the \( j \)-th that maximizes the user’s reward \( \hat{R} \) on validation samples, i.e.,

\[
\arg \max_{j=1}^M \hat{R} \left( \sum_{i=1}^N \lambda_i^j \theta_i \right).
\]

---

2. For simplicity, we name reward what others refer to also as the return or the cumulative reward; indeed, those concepts coincide in our setups, where there is no discount factor.
3. For the inference phase, we predict using the network $f$ parameterized by $\sum_{i=1}^{N} \lambda_i \theta_i$.

While MORL interpolates the rewards, RS interpolates the weights. This is a considerable advantage as the appropriate interpolating coefficients $\lambda$, which depends on the desired trade-off, can be selected $a$ posteriori; the selection is achieved without additional training, only via inference on some samples. In the next Section 6.2.2, we explicitly state the Hypotheses 6.1 and 6.2 underlying in RS. Their empirical verification will be the main motivation for our experiments on various tasks in Section 6.3.

### 6.2.2 Exploring the properties of rewarded soup

#### 6.2.2.1 Linear mode connectivity of weights fine-tuned on diverse rewards

We consider $\{\theta_i\}_{i=1}^{N}$ fine-tuned on $\{R_i\}_{i=1}^{N}$ from a shared pre-trained initialization. Previously in Observation 4.1, the LMC was defined w.r.t. a single performance measure (e.g., accuracy) in supervised learning. We now extend this notion in RL with $N$ rewards, and define that the LMC holds if all rewards for the interpolated weights exceed the interpolated rewards. It follows that the LMC condition which underpins RS’s viability is the Hypothesis 6.1 below.

**Hypothesis 6.1 (LMC).** $\forall\{\lambda_i\}_i \in \Delta_N$ and $k \in \{1, \ldots, N\}$, $R_k(\sum_{i} \lambda_i \cdot \theta_i) \geq \sum_{i} \lambda_i R_k(\theta_i)$.

#### 6.2.2.2 Pareto optimality of rewarded soups

The Pareto front (PF) is the set of undominated weights, for which no other weights can improve a reward without sacrificing another, i.e.,

$$PF = \{\theta \mid \exists \theta' \in \Theta \text{ s.t. } \{R_i(\theta')\}_{i=1}^{N} >_N \{R_i(\theta)\}_{i=1}^{N}\},$$

where $>_N$ is the dominance relation in $\mathbb{R}^N$. However, in practice, we only need to retain one policy for each possible value vector; a set of such policies is named a Pareto coverage set (PCS). We now introduce the key Hypothesis 6.2.

**Hypothesis 6.2 (Pareto optimality).** The set $\{\sum_{i} \lambda_i \cdot \theta_i | \{\lambda_i\}_i \in \Delta_N\}$ is a PCS of $\{R_i\}_i$.

**Hypothesis 6.2** holds if the rewarded soups solutions, uncovered by interpolation, are Pareto-optimal. Overall, we empirically validate Hypothesis 6.1 and Hypothesis 6.2 in Section 6.3. Moreover, we theoretically prove in Lemma 6.1 that Hypothesis 6.2 holds when rewards are replaced by their second-order Taylor expansion with Hessians proportional to the identity (or only co-diagonalizable in Appendix C.4), a simplified setup justifiable when weights remain close.

**Lemma 6.1.** Let $\hat{\mu} = (\hat{\mu}_1, \ldots, \hat{\mu}_N) \in \Delta_N$. We assume that the $N$ rewards are quadratic; thus we can write for $i \in \{1, \ldots, N\}$:

$$\forall \theta \in \Theta, \quad R_i(\theta) = R_i(\theta_i) - \eta_i \|\theta - \theta_i\|^2$$

(6.2)
Then the reward $R_{\hat{\mu}} = \sum_i \hat{\mu}_i \times R_i$ is maximized on the convex hull of $\{\theta_1, \ldots, \theta_N\}$.

**Proof.** The function $R_{\hat{\mu}}$ is quadratic thus has an unique global maximum $\hat{\theta}$:

$$\nabla_{\theta} R_{\hat{\mu}}(\hat{\theta}) = 0 \implies \sum_{i=1}^N \mu_i \eta_i \cdot (\hat{\theta} - \theta_i) = 0$$

$$\implies \hat{\theta} = \frac{\sum_{i=1}^N \hat{\mu}_i \eta_i \cdot \theta_i}{\sum_{i=1}^N \hat{\mu}_i \eta_i}$$

Since all the $\hat{\mu}_i \eta_i$ are positive or zero, and at least one is greater than zero, $\hat{\theta}$ is indeed in the convex hull of $\{\theta_1, \ldots, \theta_N\}$. \hfill $\square$

**Remark 6.1.** Hypotheses 6.1 and 6.2 rely on a good pre-trained initialization, making RS particularly well-suited to fine-tune foundation models. This is because pre-training prevents the weights from diverging during training [Ney+20]. When the weights remain close, we can theoretically justify Hypotheses 6.1 and 6.2 by leveraging the similarity between WI and weighted functional ensembling, as confirmed empirically in Figure 6.4(c). In contrast, the LMC would not hold when training from scratch [Ney+20]. Moreover, pre-training addresses stability and exploration issues [Xie+22; Yan+23b; Sek+20] in RL.

**Remark 6.2.** Pareto-optimality in Hypothesis 6.2 is defined w.r.t. a set of possible weights $\Theta$. Yet, in full generality, improvements in initialization, RL algorithms, data, or specific hyperparameters could enhance performances. In other words, for real-world applications, the true PF is unknown and needs to be defined w.r.t. a training procedure. In this case, $\Theta$ represents the set of weights attainable by fine-tuning within a shared procedure. As such, in Section 6.3 we analyze Hypothesis 6.2 by comparing the fronts obtained by RS and scalarized MORL while keeping everything else constant.

### 6.2.2.3 Consequences of Pareto optimality for linear preferences

**Lemma 6.2** (Reduced reward misspecification). If Hypothesis 6.2 holds, and for linear reward $\hat{R} = \sum_i \hat{\mu}_i R_i$ with $\{\hat{\mu}_i\} \in \Delta_N$, then $\exists \{\lambda_i\} \in \Delta_N$ such that $\sum_i \lambda_i \cdot \theta_i$ is optimal for $\hat{R}$.

In simpler terms, Lemma 6.2 implies that if Hypothesis 6.2 is true, then RS can mitigate reward misspecification. The proof directly follows the definition of Pareto optimality. For any preference $\hat{\mu}$, there exists a $\lambda$ such that the $\lambda$-interpolation over weights maximizes the $\hat{\mu}$-interpolation over rewards. In practice, as we will see in Figure 6.4(a), we can set $\lambda = \hat{\mu}$, or cross-validate $\lambda$ on other samples. Yet, this theoretically holds only for $\hat{R}$ linear over the proxy rewards. This follows the linear utility functions setup from the MORL literature [Râd+20], with limitations: for example, all human preferences cannot be expressed linearly [Vam+08]. This motivates having sufficiently rich and diverse proxy rewards to capture the essential aspects of all possible users’ rewards. Despite the lack of theoretical guarantees, we will show in Figure 6.4(b) that weight interpolation improves results even for non-linear $\hat{R}$.
6.3 Experiments

In this section we implement RS across a variety of standard learning tasks: text-to-text generation, image captioning, image generation, visual grounding, visual question answering, and locomotion. We use either model or statistical rewards. We follow a systematic procedure. First, we independently optimize diverse rewards on training samples. For all tasks, we employ the default architecture, hyperparameters and RL algorithm; the only variation being the reward used across runs. Second, we evaluate the rewards on the test samples: the results are visually represented in series of plots. Third, we verify Hypothesis 6.1 by examining whether RS’s rewards exceed the interpolated rewards. Lastly, as the true PF is unknown in real-world applications, we present empirical support for Hypothesis 6.2 by comparing the front defined by RS (sliding $\lambda$ between 0 and 1) to the MORL’s solutions optimizing the $\mu$-weighted rewards (sometimes only $\mu = 0.5$ for computational reasons). Our website provides additional qualitative results.

6.3.1 Text-to-text: LLaMA with diverse RLHF

Given the importance of RLHF to train LLMs, we begin our experiments with text-to-text generation. Our pre-trained network is LLaMA-7b [Tou+23a], instruction fine-tuned [Wei+22a; Wan+22b] on Alpaca [Tao+23]. For RL training with PPO [Sch+17], we employ the trl package [Wer+20] and the setup from [Bee+23] with low-rank adapters (LoRA) [Hu+22b] for efficiency. We first consider summarization [Sti+20b; Wu+21] tasks on two datasets: Reuter news [Ahm17] in Figures 6.1(b) and 6.2(a) and Reddit TL,DR [Völ+17] in Figure 6.2(b). We also consider answering Stack Exchange questions [Lam+23] in Figure 6.2(c), movie review generation in Figure 6.2(d), and helpfulness as a conversational assistant [Bai+22a] in Figures 6.2(e) and 6.2(f). To evaluate the generation in the absence of supervision, we utilized $N = 2$ different reward models (RMs) for each task, except in Figure 6.2(f) where $N = 4$. These RMs were trained on human preferences datasets [Chr+17b] and all open-sourced on HuggingFace [Wol+20]. For example in summarization, $R_1$ [Sti+20b] mostly evaluates completeness while $R_2$ [Che+21] evaluates faithfulness; these two criterions are in tension, as improving one often degrades the other. For other tasks, we rely on diverse RMs from OpenAssistant [Köp+23]; though they all assess if the answer is adequate, they differ by their architectures and procedures.

The results are reported in Figure 6.2. The green front, defined by RS between the two weights specialized on $R_1$ and $R_2$, is above the straight line connecting those two points, validating Hypothesis 6.1. Second, the front passes through the point obtained by MORL fine-tuning on the average of the two rewards, supporting Hypothesis 6.2. Moreover, when comparing both full fronts, they have qualitatively the same shape; quantitatively in hypervolume [Yen+13] (lower is better, the area over the curve w.r.t. an optimal point), RS’s hypervolume is 0.367 vs. 0.340 for MORL in Figure 6.2(a), while it is 1.176 vs. 1.186 in Figure 6.2(b). Finally, in Figure 6.2(f), we use $N = 4$ RMs for the assistant task and
uniformly average the $N = 4$ weights, confirming that RS can scale and trade-off between more rewards.

### 6.3.2 Image-to-text: captioning with diverse statistical rewards

RL training is also effective for multimodal tasks [Pin+23], for example in image captioning [Ren+17] where the task is to generate textual descriptions of images. Precisely evaluating the quality of a prediction w.r.t. a set of human-written captions is a challenging task, thus the literature relies on various hand-engineered, non-differentiable
metrics: e.g., the precision-focused BLEU [Pap+02], the recall-focused ROUGE [Lin+03], METEOR [Ban+05] handling synonyms and CIDEr [Ved+15] using TF-IDF. As these metrics are proxies for human preferences, good trade-offs are desirable. We conduct our experiments on COCO [Lin+14], with an ExpansionNetv2 [Hu+22] network and a Swin Transformer [Liu+22] visual encoder, initialized from the sota weights of [Hu+22] optimized on CIDEr. We then utilize the code of [Hu+22] and their self-critical [Ren+17] procedure (a variant of REINFORCE [Wil92]) to reward the network on BLEU1, BLEU4, ROUGE or METEOR.

![Figure 6.3](image1.png)

(a) BLEU1 and ROUGE.  
(b) BLEU1 and BLEU4.  
(c) Captioning: spider map.

Figure 6.3. – Results in image captioning on COCO [Lin+14]. As rewards $R_1$ (blue stars every epoch) and $R_2$ (red stars), we consider standard statistical metrics: BLEU1 (1-gram overlap), BLEU4 (4-grams overlap), ROUGE, METEOR and CIDEr. Figure 6.3(a) include the MORL training trajectories optimizing $(1 − \mu) \times \text{BLEU}1 + \mu \times \text{ROUGE}$, uncovering a yellow front similar to RS’s green front. In Figure 6.3(c), RS uniformly averages the 5 weights (one for each reward), resulting in the largest area and the best trade-off between the 5 rewards.

![Figure 6.4](image2.png)

(a) Analysis of Lemma 6.2.  
(b) Performances w.r.t. $\lambda$.  
(c) RS vs. ensembling.

Figure 6.4. – Refined results in captioning with $R_1 = \text{BLEU}1$ and $R_2 = \text{ROUGE}$. Figure 6.4(a) empirically validates Lemma 6.2 by reporting results of RS (for varying $\lambda$) and of MORL (for varying $\mu$) for varying user’s preference $\bar{\mu}$. In Figure 6.4(b), all rewards are used for evaluation as a function of the interpolating coefficient. In Figure 6.4(c), we report the front of the costly functional ensembling [Han+90; Lak+17] of predictions (rather than the weight interpolation).
We observe in Figure 6.3 that tuning solely BLEU1 sacrifices some points on ROUGE or BLEU4. Yet interpolating between θ₁ and θ₂ uncovers a convex set of solutions approximating the ones obtained through scalarization of the rewards in MORL. When comparing both full fronts in Figure 6.3(a), they qualitatively have the same shape, and quantitatively the same hypervolume [Yen+13] of 0.140. One of the strengths of RS is its ability to scale to any number of rewards. In Figure 6.3(c), we uniformly (λi = 1/5) average N = 5 weights fine-tuned independently. It improves upon the initialization [Hu+22a] and current sota on all metrics, except for CIDEr, on which [Hu+22a] was explicitly optimized.

Figure 6.4 refines our analysis of RS. In Figures 6.4(a) and 6.4(b), rewards are normalized to 1 for the initialization and 0 for the worst model. Figure 6.4(a) validates Lemma 6.2: for any linear preference ˆµ over the proxy rewards, there exists an optimal solution in the set described by RS. Two empirical strategies to set the value of λ are close to optimal: selecting λ = ˆµ if ˆµ is known, or cross-validating (CV λ) if a different data split [Kar+15] is available. Moreover, Figure 6.4(b) investigate all metrics as evaluation. Excluding results’ variance, we observe monotonicity in both training rewards, linear in BLEU1 and quadratic in ROUGE. For other evaluation rewards that cannot be linearly expressed over the training rewards, the curves’ concavity shows that RS consistently improves the endpoints, thereby mitigating reward misspecification. The optimal λ depends on the similarity between the evaluation and training rewards: e.g., best BLEU2 are with small λ. Lastly, as per Lemma 4.1, Figure 6.4(c) shows that RS succeeds because WI approximates functional ensembling, interpolating the predictions rather than the weights. Actually, ensembling performs better, but it cannot be fairly compared as its inference cost is doubled.

6.3.3 Text-to-image: diffusion models with diverse RLHFs

Beyond text generation, we now apply RS to align text-to-image generation with human feedbacks [Lee+23; Wu+23a; Xu+23]. Our network is a diffusion model [Ho+20] with 2.2B parameters, pre-trained on a dataset from MetaAI of 300M images; it reaches similar quality as Stable Diffusion [Rom+22], which was not used for copyright reasons. To represent the subjectivity of human aesthetics, we employ N = 2 open-source reward models: ava, trained on the AVA dataset [Mur+12], and cafe, trained on a mix of real-life and manga images. We first generate 10000 images; then, for each reward, we remove half of the images with the lowest reward’s score and fine-tune 10% of the parameters [Xie+23] on the reward-weighted negative log-likelihood [Lee+23]. As a side note, on-policy RL would require performing loops of image generations and model fine-tunings [Don+23b], but we only perform a single offline iteration for simplicity.

The results displayed in Figure 6.5(a) validate Hypothesis 6.1, as the front described by RS when sliding λ from 0 and 1 is convex. Moreover, RS gives a better front than MORL, validating Hypothesis 6.2. Interestingly, the ava reward model seems to be more general-purpose than cafe, as RL training on ava also enhances the scores of cafe. In contrast, the model θcafe performs poorly in terms of ava in Figure 6.5(a). Nonetheless, RS with (1 − λ) · θava + λ · θcafe outperforms θava alone, not only in terms of cafe, but also of ava.
when $\lambda \in \{0.1, 0.2\}$. These findings confirm that RS can better align text-to-image models with a variety of aesthetic preferences. This ability to adapt at test time paves the way for a new form of user interaction with text-to-image models, beyond prompt engineering.

![Graphs and Figures]

(a) Image generation.  (b) VG: Small and Large.  (c) VG: spider map.

Figure 6.5. – Figure 6.5(a) reports our RLHF experiments on text-to-image generation with diffusion models. From the pre-trained initialization, we learn $\theta_{\text{ava}}$ and $\theta_{\text{cafe}}$ by optimizing the two reward models $\text{ava}$ and $\text{cafe}$. Interpolation between them reveals the green Pareto-optimal front, above the yellow MORL front. Figures 6.5(b) and 6.5(c) report our results in visual grounding on RefCOCO+ [Yu+16], where we optimize to predict boxes with IoU > 0.5 w.r.t. the ground-truth, for objects of either small, medium or large size.

6.3.4 Text-to-box: visual grounding of objects with diverse sizes

We now consider visual grounding (VG) [Yu+16]: the task is to predict the bounding box of the region described by an input text. We use a seq-to-seq unified model predicting the box auto-regressively as a sequence of location tokens [Wan+22a]. This model is pre-trained on a large image-text dataset, then fine-tuned with cross-entropy for VG; finally, we use a weighted loss between the cross-entropy and REINFORCE in the RL stage. As the main evaluation metric for VG is the accuracy (i.e., intersection over union (IoU) > 0.5), we consider 3 non-differentiable rewards: the accuracy on small, medium, and large objects. We design this experimental setup because improving results on all sizes simultaneously is challenging, as shown in Figure 6.5(c), where MORL performs similarly to the initialization.

The results in Figure 6.5(b) confirm that optimizing for small objects degrades performance on large ones; fortunately, interpolating can trade-off. In conclusion, we can adapt to users’ preferences at test time by adjusting $\lambda$, which in turn changes the object sizes that the model effectively handles. On the one hand, if focusing on distant and small objects, a large coefficient should be assigned to $\theta_{\text{Small}}$. On the other hand, to perform well across all sizes, we can recover initialization’s performances by averaging uniformly (in Figure 6.5(c)).

![Table]

<table>
<thead>
<tr>
<th>Object Size</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>0.780</td>
</tr>
<tr>
<td>Medium</td>
<td>0.785</td>
</tr>
<tr>
<td>Large</td>
<td>0.790</td>
</tr>
<tr>
<td>Pre-trained init</td>
<td>0.800</td>
</tr>
<tr>
<td>RL: Small</td>
<td>0.805</td>
</tr>
<tr>
<td>RL: Medium</td>
<td>0.810</td>
</tr>
<tr>
<td>RL: Large</td>
<td>0.815</td>
</tr>
<tr>
<td>MORL: $\sum_{i=3}^N R_i/3$</td>
<td>0.820</td>
</tr>
</tbody>
</table>

Figures 6.5(b) and 6.5(c) report our results in visual grounding on RefCOCO+ [Yu+16], where we optimize to predict boxes with IoU > 0.5 w.r.t. the ground-truth, for objects of either small, medium or large size.
6.3.5 Text&image-to-text: VQA with diverse statistical rewards

We explore visual question answering (VQA), where the task is to answer questions about images. We use the OFA model [Wan+22a] (generating the answers token-by-token), on the VQA v2 dataset, pre-trained with cross-entropy, and fine-tuned with REINFORCE during the RL stage. During the RL fine-tuning, we explore as rewards the BLEU (1-gram) and METEOR metrics: these metrics enable assigning partial credit if the ground-truth and predicted answers are not identical but still have some words in common.

Our results in Figure 6.6 validate the observations already made in previous experiments: RL is efficient to optimize those two rewards, and RS reveals a Pareto-optimal front to balance between them.

6.3.6 Locomotion with diverse engineered rewards

Teaching humanoids to walk in a human-like manner [Dua+16] serves as a benchmark to evaluate RL strategies [Ng+99] for continuous control. One of the main challenges is to shape a suitable proxy reward [Dor+94; Dew14], given the intricate coordination and balance involved in human locomotion. It is standard [Tod+12] to consider dense rewards of the form $R = velocity - \alpha \times \sum_t a_t^2$, controlling the agent’s velocity while regularizing the actions $\{a_t\}_t$ taken over time. Yet, the penalty coefficient $\alpha$ is challenging to set. To address this, we devised two rewards in the Brax physics engine [Fre+21]: a risky $R_1$ with $\alpha = 0$, and a more cautious $R_2$ with $\alpha = 1$.

Like in all previous tasks, RS’s front in Figure 6.7 exceeds the interpolated rewards, as per Hypothesis 6.1. Moreover, the front defined by RS indicates an effective balance between risk-taking and cautiousness, providing empirical support for Hypothesis 6.2, although MORL with $\mu = 0.5$ (i.e., $\alpha = 0.5$) slightly surpasses RS’s front. For a more qualitative and intuitive assessment, we provide animations of our RL agent’s locomotion on our website.
6.4 Discussion and related work

We previously consider the LMC when fine-tuning with different losses in Section 4.6.4 (as in [Cro+23]) or on different tasks in Chapter 5 (as in [Ilh+22; Don+23a; Ram+23a; Dim+22]): the key contribution of this Chapter 6 is to investigate the LMC in RL. Like in supervised learning, we confirm in Figure 6.4(c) that WI succeeds due to its similarity with ensembling. The most similar works are for control system tasks: [Law+23] averaging decision transformers and [Gay+22] explicitly enforcing connectivity in subspaces of policies trained from scratch on a single reward.

When dealing with multiple objectives in DL, the common strategy is to combine them into a single reward [Roi+13; Råd+20]. For example, [Gla+22] sum the predictions of a preference RM (as a proxy for helpfulness) and a rule RM (detecting harmful contents, by utilizing itself a set of diverse rules); [Wu+23c] assign different weightings to the relevance/factuality/completeness rewards, thereby customizing how detailed and lengthy the LLMs responses should be. The main reason why single-policy approaches are not suitable is that they optimize over a single set of preferences. In contrast, we build a coverage set of Pareto-optimal policies. This is important for the following reasons, mostly first discussed in Kirk et al. [Kir+23] and in Hayes et al. [Hay+22]. First, the user’s true reward is highly uncertain before training. This “semi-blind” [Hay+22] manual process forces a priori and uncertain decisions about the required trade-offs. Critically, RLHF may cause the “tyranny of the crowdworker” [Kir+23], converging the opinions of a few individuals. Moreover, biases are caused by chaotic engineering choices, and “are exacerbated by a lack of [...] documentation” [Kir+23]. In this dynamic utility function scenario, RS can quickly adapt with fewer data, by simply adjusting the $\lambda$ to match new preferences. Finally, RS could also improve the interpretability and explainability of the decisions.

The multi-policy alternatives [Bar+08; Li+20b; Tan+03; Van+14; Mar+23] are usually not suitable because of their computational costs required to learn a dense set of policies. To reduce the cost, [Won+20; Yan+20b; Abd+20; Lin+22] build experts and then train a new model to combine them; [Mos+16; Wil+07; Ngu+20] share weights across experts; [Cas+13; Yan+19b; Abe+19; Pes+21] directly train a single model; the recent and more similar work [Hua+23] learns one linear embedding per (locomotion) task that can be interpolated. Yet, these works are mostly for academic benchmarks [Tod+12; Vam+11]; adapting them to larger tasks (e.g., RLHF for foundation models with PPO) is challenging as they modify the training procedure. Among the ensembling-like RL strategies [Wan+10; Mor+15; Raj+17] such as population-based training [Jad+17; Jad+19], [Par+20; Osa+22] explicitly increase the diversity across policies, yet never considering foundation models nor weight interpolation. In contrast, RS is compatible with the inherent iterative engineering process of alignment. Indeed, RS can continually include adjusted opinions while preventing forgetting of the old behaviours. This relates to the continual learning challenge, and the empirical observations that weight averaging can reduce catastrophic forgetting [Sto+22; Eec+22]. Our strategy only trains the proxy rewards independently and enables the selection of the interpolating coefficient a posteriori. This is especially useful with large number of rewards and thus growing number of combinations. Finally, its distributed
nature makes RS parallelizable thus practical in a federated learning setup \cite{McM+17} where data must remain private.

Finally, our a posteriori alignment with users facilitates \textbf{personalization} \cite{Sal+23} of models. As discussed in \cite{Kir+23}, this could increase usefulness by providing tailored generation, notably to under-represented groups. This is all the more important as human preferences change from time to time. Yet, this personalization comes with risks for individuals of “reinforcing their biases [...] and narrowing their information diet” \cite{Kir+23}. This may worsen the polarization of the public sphere. Under these concerns, we concur with the notion of “personalization within bounds” \cite{Kir+23}, with these boundaries potentially set by weights fine-tuned on diverse and carefully inspected rewards.

\section{Conclusion}

As AI systems are increasingly applied to crucial real-world tasks, there is a pressing issue to align them to our specific and diverse needs, while making the process more transparent. In this chapter, we propose rewarded soup, a weight interpolation strategy that efficiently yields Pareto-optimal solutions to mitigate reward misspecification. Our experiments have consistently validated our working LMC hypotheses for various significant large-scale tasks, involving multiple modalities and different fine-tuning strategies. This confirms that weight interpolation can be a practical strategy to approximate functional ensembling for real-world applications, such as RLHF to transform LLMs into helpful and harmless conversational agents. We hope to inspire further research in exploring how the generalization literature in DL can help for alignment, to create AIs handling the diversity of opinions. As we discuss in the closing Chapter 7 of this thesis, this alignment is critical to ensure safe and ethical progress in the future.
CONCLUSION

We summarize the contributions of this thesis, offer some future directions and finally discuss the risks posed by the recent and rapid progress in AI.

7.1 Contributions

Our works throughout this thesis deal with the efficient ensembling of diverse models, mostly to improve OOD generalization in DL. Theoretical considerations were essential to drive the experiments: yet, for the sake of clarity, we separate theoretical and empirical contributions in this concluding section. We also discuss the societal impact of our work in Appendix B.

7.1.1 Theoretical contributions

Our main theoretical contribution is the initial discussion in Section 2.2 relating variance to diversity shift and bias to correlation shift. This novel understanding of the two kinds of distribution shifts opened up ways to predict when ensembling could be beneficial, and identified scenarios where other strategies (such as those based on invariance developed in Appendix E) would be necessary. We clarify the role of diversity in ensembling, through its similarity with covariance, and further analyzed it from an information theory perspective in Chapter 3. We also provide new guarantees for the effectiveness of weight averaging under distribution shifts, based on its similarity with functional ensembling. We depart from the traditional flatness-based arguments, critically examined and dismantled in Section 4.2.2. Overall, we highlight trade-offs in DL: between bias and variance in supervised learning, between individual accuracy and diversity in ensembling, between diversity and locality in weight averaging, and between different rewards in reinforcement learning.

7.1.2 Empirical contributions

Deep learning is (above all) an empirical science, where bounds and theoretical arguments often serve more as indicators than absolute truths. We have worked extensively on the empirical benefits of weight averaging (WA): it is scalable, effective in OOD generalization, and aligns well with the inherent iterative engineering process of DL. Actually, the main strength of WA lies in its simplicity, facilitating its adoption by the community,
line with the “bitter lesson” from [Sut19]. Moreover WA is compatible with other tools and is boosted by the massive adoption of the foundation models [Bom+21] paradigm in the DL community. Actually, we emphasize a novel benefit from this foundation model paradigm; though it is commonly used to increase individual performance, we highlight its ability to tackle the computational burden of ensembling:

- at training: fine-tuning is more efficient than training from scratch, making $M \gg 1$ fine-tunings realistic,

- at inference: as fine-tuning enforces the LMC, allowing us to simply average the $M$ weights rather than the predictions.

A significant part of this thesis revolves around the concept of diversity, especially useful to improve OOD generalization. The conclusion from Section 5.4.5 is that, when aiming at OOD with ensembling strategies, there actually exists a trade-off between diversity and ID accuracy. This finding contrasts with [Mil+21] and goes against the prescription in [Wen+22] that, “to make the model more robust on OOD data, the main focus should be to improve the ID classification error”. This emphasis on diversity goes beyond the ensembling setup: in Chapter 6, we embrace the diversity of opinions in RL, and show that handling diverse rewards is necessary for better alignment; in Appendix E, we show that diversity in training domains and learned features enhance the robustness of models under correlation shift. Overall, we helped popularize diversity as a key concept.

To summarize, we provide below an analytical formulation with equations of the different fine-tuning strategies considered along this thesis, where $\theta$ represents the weights, $A_i$ the auxiliary tasks, $D$ the target task, and $R_i$ the potential different rewards.

\[
\begin{align*}
\theta &= \text{Train} \left( \theta^{pt}, D \right), & \text{[Vanilla fine-tuning [Oqu+14]]} \\
\theta &= \text{Train} \left( \theta^{pt}, D, \text{collect}_\text{ckpts} = \text{True} \right), & \text{[Moving average [Izm+18]]} \\
\theta &= \frac{1}{M} \sum_{i=0}^{M-1} \text{Train} \left( \theta^{pt}, D \right), & \text{[DiWA [Ram+22b] Chapter 4]} \\
\theta &= \text{Train} \left( \text{Train} \left( \theta^{pt}, A_i \right), D \right), & \text{[Inter-training [Pha+18]]} \\
\theta &= \frac{1}{M} \sum_{i=0}^{M-1} \text{Train} \left( \text{Train} \left( \theta^{pt}, A_i \right), D \right), & \text{[Model ratatouille [Ram+23a] Chapter 5]} \\
\theta &= \sum_{i=0}^{M-1} \lambda_i \text{TrainRL} \left( \theta^{pt}, D, R_i \right), & \text{[Rewarded soups [Ram+23b] Chapter 6]}
\end{align*}
\]

### 7.2 Future work

The works presented in this thesis has laid the groundwork for numerous exciting research directions.
7.2.1 Improved generalization

I believe the potential of RL for OOD generalization is still relatively unexplored. RL could be harnessed to directly optimize certain invariance or fairness metrics, bypassing the need for differentiable proxies. Notably, this could be applied to optimize directly the true invariance objective IRM [Arj+17] (rather than a proxy differentiable loss) or expected calibration error (ECE) [Guo+17].

Moreover, current strategies for correlation shift always implicitly incorporate fundamental a priori assumptions (within the data split or the validation dataset) concerning which features should be considered spurious or not. Moreover, they often assume the uniqueness of the spurious information (e.g., color in ColoredMNIST), and would fail otherwise (e.g., color and background). Combining such methods with weight interpolation (as in Chapter 6) could provide Pareto-optimal fairness solutions [Che+23]. This would facilitate a posteriori decision-making when balancing between (potentially conflicting) diverse fairness constraints [Lop+22], and simplify external regulation.

7.2.2 Weight averaging

Though WA is now widely used by the DL community [Wor23], its theoretical understanding is still lacking, even in the simplest case of moving averages [Izm+18]. Our best theory relies on the similarities between weight interpolation and functional ensembling, with guarantees only when weights remain close. Interestingly, this does not explain the following fact: WA consistently outperforms ENS in OOD (as shown in Figure 4.5) but the opposite is true ID (as illustrated in [Wor+22a]). To better understand this surprising insight, we have already conducted a preliminary “label corruption” experiment suggesting that WA mitigates memorization. More precisely, we trained two models on the same dataset, but we occasionally corrupt the labels for the first model: we observe that averaging the predictions with ENS can memorize the corrupted labels, yet that averaging the weights with WA actually forgets them. Could this phenomenon be linked to a form of invariance across different runs? This research direction was actually the key motivation behind [Ram+24], published just after the end of this thesis.

So far we mostly used WA to improve OOD generalization and alignment, yet WA has numerous other potential applications. A fascinating phenomenon is WA’s capacity to combine the capabilities of various models. For instance, [Ilh+23] showed that we can edit models with task arithmetic; negative coefficients in interpolation can eliminate specific undesired behaviors. Another study [Jan+23] demonstrated that averaging an English summarizer and an English-to-French translator can produce summaries in French. We briefly examined this in our workshop paper [Ram+22c]; yet, this phenomenon remains an active area of research [Ort+23]. More generally, WA has the potential to be a crucial component of efficient scaling strategies as a simple parallelization method [Wor+23], eliminating complex multi-node synchronization. This could be refined with data selection strategies such as pruning [Ro+23] and partitioning [Li+22]. As a side note, this is consis-
tent with recent rumors suggesting that GPT-4 [Ope23] is actually an ensemble of sixteen experts GPT-3.5, trained on different corpus; one for maths, one for legal, etc.

### 7.2.3 Towards updatable ML

Furthermore, WA could facilitate *iterative and updatable ML* [Raf23]. We envision a future where DNNs 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 [Kar17] and mirror the pipelines used for open-source development of software engineering via version control. Recent works [Mat+22; Li+22; Don+23a; Cho+22a] and ratatouille proposed in Chapter 5 give some primitives to learn DNNs in collaboration. Put simply, *git clone* is pre-training, *git commit* are fine-tunings performed by individual contributors on their specific tasks, *git merge* is replaced by WA, and *git test* would try to measure models’ statistics and failures on external datasets. Actually, the recent Git-Theta [Kan+23] follows these principles. In terms of privacy, such a federated learning setup [Li+19; Adn+22] where datasets can be kept private indeed seem desirable. In terms of computation and sustainability, minimal communication across servers enable embarrassingly simple parallelization [Li+22; Wor+23], reducing costs and CO2 emissions. The ability to collaboratively improve weights shifts from *proprietary network training* to *open-source collaborative network building*, leveraging distributed computing resources such as single-GPU desktop machines. We see this as an exciting possibility for the future of AI.

### 7.2.4 Towards more general AI

A key challenge is transitioning from classification towards more general tasks, involving rewards challenging to define [Kwo+23]. An intriguing approach I plan to explore is reinforcement learning from AI feedback (RLAIF) [Bai+22b], where the rewards are generated by other AI reward models. The key difference with RLHF is that those reward models do not necessarily require human feedback. Factual generation serves as an interesting example; in [Roi+23], the reward model was pre-trained on textual entailment assessment [Dag+05]. Another central idea from [Bai+22b] is to inscribe foundational principles in the rewards, similar to Asimov’s universal laws for robots. The *self-recursive approach* [Gou+23; Mad+23; Sun+23] uses AI’s own predictions, generating a critique followed by a revision, leveraging LLMs’ in-context capabilities that make them generate superior responses when guided with specific “step by step” prompts [Wei+22c; Yao+23]. Those rewards models could be expanded by various tools [Sch+23; Yan+23a], ranging from web access [Nak+21], to employing citation experts [Men+22], expert networks [She+23], or planners [Liu+23]. In relation to this thesis, debates and consensus among different LLMs could also help [Irv+18]. Moreover, rewards usually evaluate the full generation, without clarifying which parts contain what types of errors; a promising idea is to reward the process [Lig+23; Wu+23b] rather than the output, to precisely explicitly indicate
“which sentence is false, which sub-sentence is irrelevant” [Wu+23b]. The diversity of those potential fine-grained reward models strengthen the multi-objective approach introduced in Chapter 6. For example, [Wu+23b] incorporates multi-dimensional evaluation such as factuality, irrelevance, and information incompleteness. This work [Wu+23b] and our rewarded soup [Ram+23b] are limited to linear preferences over those proxy rewards; considering more complex combinations [Vam+18; Vam+08] is a promising direction, as in [Tou+23b] where they leverage a piecewise combination of rewards.

Finally, LLMs have significantly moved us closer to creating general-purpose AI; nevertheless, their current limitation to a single modality (text) restricts their understanding and interaction with the world. The intuition is that an AI, in order to become general, must embody itself, take shape, in order to face the unexpected and solve new problems. To handle diverse tasks across numerous modalities, we thus need robust multimodal models, such as the unified [Ala+22; Wan+22a] and our proposed UniVAL [Shu+23]. Measuring and reducing the hallucinations in those multimodal models is actually at the core of [Shu+24], our latest research project. To embed those novel AI prior to real-world deployment, they could be deployed in open-world synthetic environments [Zhu+23], both to reduce computational costs and provide better understanding of ethical dilemmas [Pan+23]. This discussion leads us seamlessly into the final and critical part of this thesis: AI safety.

### 7.2.5 AI safety

The recent and rapid scaling of LLMs poses both opportunities and major concerns [Amo+16], causing a wave of inquiry about the potential negative impacts; scenarios relegated to science fiction just a few years (months?) back seem now less far-fetched. OpenAI [Jan23] summarized it this way:

*Superintelligence will be the most impactful technology humanity has ever invented, and could help us solve many of the world’s most important problems. But the vast power of superintelligence could also be very dangerous, and could lead to the disempowerment of humanity or even human extinction.*

Recently, some leading experts advocated for a pause [Lif23] in scaling and deployment. Therefore, without forgetting the fantastic possibilities opened by AI, we need to draw our attention towards the potentially negative societal consequences.

#### 7.2.5.1 Risks

Drawing from [Eve+18; Wei+22d; Bru15], we enumerate below a progression from current harms to more speculative risks.

- Biased discrimination, hate-speech and exclusion.
- Misinformation harms: hallucinations and spread of false news.
- Environmental harms: energy consumption and hardware manufacturing.
• Socioeconomic harms: increased inequality, job displacement, reduction in human creativity, monopoly on hardware.
• Information hazards: privacy breaches and data leaks.
• Individual malicious uses: cyber attacks, fraud.
• Governmental oppressions: surveillance, censorship.
• Human-computer interaction harms: anthropomorphizing and trust exploitation.
• Power-seeking behaviors: (functional) deception and manipulation.

Though there is scientific consensus regarding the current harms first listed, the plausibility of the more speculative risks remains a subject of ongoing debates in the community [AI 23]. While Meredith Whittaker has stated “There’s no more evidence now than there was in 1950 that AI is going to pose these existential risks,”, there are actually recent works suggesting the contrary [Hen+23; Hen+22; Hen23].

7.2.5.2 On the difficulty of alignment

The high uncertainty surrounding these questions calls for caution, and highlight the importance of building human-aligned AI that behaves in accordance with what we want. This alignment [Tay+16; Ken+21; Ngo+22] is complex, with challenges usually split into two categories: outer alignment and inner alignment.

**Outer alignment.** How can we design a robust and reliable reward aligned with the intended goal of its designers? This comes with several challenges:

• Proxy gaming. There is a risk that reward models could be exploited adversarially [Ska+22]. Then an AI may find loopholes or shortcuts to maximize its rewards without achieving the desired behaviour, a phenomenon known as wireheading.

• Lack of robustness [Gao+22]. There is a risk that the reward does not sufficiently cover all potential harms. Then an AI could meet the desired objective while neglecting long-term consequences and negative side-effects.

More broadly, these issues refer to Goodhart’s law [Smi21]: “when a measure becomes a target, it ceases to be a good measure”.

**Inner alignment.** The second question is: assuming a suitable reward, how can we make sure an AI truly optimizes it on test samples? This comes with several challenges:

• Underfitting, due to a lack of exploration (an inherent challenge in RL) leading to poor performances (even in train).

• Overfitting, due to the optimization of a specific reward on specific training samples, leading to poor generalization on samples not seen in training.

• Goal misgeneralization [Sha+22; Di +22] caused by spurious correlations between domains and rewards in training.

Actually, without constraints on the test distribution, complete inner alignment may be impossible [Wol+23], for instance, for LLMs with prompts of arbitrary (long) length.
Remark 7.1. An intriguing yet-to-developed solution to this inner-alignment challenge is objective-driven AI, as advocated by Yann LeCun [LeC22]; instead of the current RL fine-tuning paradigm, they would use a reward-guided decoding, maximizing objectives during inference time when acting in the world; we would only accept model predictions if they meet specific safety constraints (as in sample-and-rank [Kul+20]), leading to a controllable AI that behave appropriately.

7.2.5.3 Alignment as OOD generalization?

The encouraging aspect is that these challenges are intricately related to the focus of this thesis: how to prevent failures under novel test distributions.

- Outer-alignment appears as an OOD challenge from the perspective of the reward model. Indeed, reward models are trained on a preference dataset whose inputs are generated by a base LLM and annotated by humans; then, they are used to evaluate generations of this LLM but updated multiple times. The temporal gap and those updates indicate a distribution shift, and thus requiring robust reward models generalizing beyond the preference dataset. To tackle the distribution shift, seminal alignment strategies [Chr+17b] used ensembling of reward models. More recently in LLaMA 2 [Tou+23b], to ensure that the reward models remain within distribution, they continually accumulate new reward modeling data.

- Inner alignment appears as an OOD challenge from the perspective of the AI. Indeed, most issues stem from a situation where the AI aligns with the reward on training samples, but this alignment does not persist due to distribution shift between the fine-tuning distribution and the real-world deployment.

Given this, I believe that tools from the OOD generalization literature could help to tackle the alignment challenge. For instance, in Chapter 6 we reduce reward misspecification using WA strategies first developed in Chapters 4 and 5 to reduce model underspecification [DAm+20] for OOD generalization. That was a first step towards improved alignment, requiring further investigation.
Openings

The rising importance of AI in our daily lives renders these ethical and technical questions increasingly critical. As I conclude this thesis, I would like to offer a more personal reflection on the matter. I posit that AI will soon become a crucial point of debate in our society, akin to the discourses surrounding the current ecological crisis. More specifically, I anticipate that AI will emerge as a new axis of left-right political polarization, with the left advocating for stringent regulation to protect workers and ensure fairness, and the right adopting a more liberal approach and optimistic view on the benefits of AI, to stimulate economic growth and improve the quality of life. Those issues are multifaceted and certainly call for nuanced perspectives.

Need for better risk evaluation. Then it is our goal as scientists to provide well-founded answers to those technical questions, assisting regulators, policymakers, and the general public in making informed decisions. A crucial aspect of this is striving for a scientific understanding and consensus about the inner workings (potentially through mechanistic interpretability [Nan+23]) and the abilities (through scalable oversight [Bow+22]) of these systems; this will enable us to accurately estimate any potential risks they may pose. We could learn a lot from climate scientists’ efforts, failures and successes, and their international panels such as the IPCC (a.k.a. GIEC). Fostering synergy between big tech, public labs, and governments could aid us in ensuring safety precautions and in circumventing any potential prisoner’s dilemma situations.

Need for more robust models. As scientists, our responsibility also lies in enhancing the robustness of these models and providing better safety guarantees, while mitigating the (currently present) sociological and environmental harms, as further discussed in Appendix B. This could pave the way for a novel era of prosperity and well-being, and assist humanity in addressing the numerous challenges we currently face. Moving forward, I plan to continue working on generalization, and investigating how the generalization literature can be used for alignment. I eagerly look forward to contributing to the development of AI systems that bring benefits to our society as a whole.
BIBLIOGRAPHY


BIBLIOGRAPHY


Sihao Chen, Fan Zhang, Kazoo Sone, and Dan Roth. “Improving Faithfulness in Abstractive Summarization with Contrast Candidate Generation and Selection”. In: NAACL. 2020 (cit. on pp. 71).

Yongqiang Chen, Kaiwen Zhou, Yatao Bian, Binghui Xie, Bingzhe Wu, Yonggang Zhang, MA KAILI, Han Yang, Peilin Zhao, Bo Han, and James Cheng. “Pareto Invariant Risk Minimization: Towards Mitigating the Optimization Dilemma in Out-of-Distribution Generalization”. In: ICLR. 2023 (cit. on p. 81).


[Di +22] Lauro Langosco Di Langosco, Jack Koch, Lee D Sharkey, Jacob Pfau, and David Krueger. “Goal misgeneralization in deep reinforcement learning”. In: ICML. 2022 (cit. on p. 84).


[Ker+21] Corentin Kervadec, Grigory Antipov, Moez Baccouche, and Christian Wolf. “Roses are red, violets are blue... but should vqa expect them to?” In: CVPR. 2021 (cit. on p. 163).


[Li+21] Wei-Hong Li, Xiaolei Liu, and Hakan Bilen. “Universal Representation Learning From Multiple Domains for Few-Shot Classification”. In: ICCV. 2021 (cit. on pp. 55, 60).


[Liu+22] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. “Swin Transformer V2: Scaling Up Capacity and Resolution”. In: CVPR. 2022 (cit. on p. 73).


[Mor+15] Igor Mordatch, Kendall Lowrey, and Emanuel Todorov. “Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids”. In: IROS. 2015 (cit. on p. 77).


[Ng+99] Andrew Y Ng, Daishi Harada, and Stuart Russell. “Policy invariance under reward transformations: Theory and application to reward shaping”. In: ICML. 1999 (cit. on p. 76).


Yujia Qin, Cheng Qian, Jing Yi, Weize Chen, Yankai Lin, Xu Han, Zhiyuan Liu, Maosong Sun, and Jie Zhou. “Exploring Mode Connectivity for Pre-trained Language Models”. In: EMNLP. 2022 (cit. on p. 62).


Alexandre Ramé, Remy Sun, and Matthieu Cord. “MixMo: Mixing Multiple Inputs for Multiple Outputs via Deep Subnetworks”. In: ICCV. 2021 (cit. on pp. 6, 7, 14, 24, 152).


[Ren+17] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. “Self-critical sequence training for image captioning”. In: CVPR. 2017 (cit. on pp. 9, 65, 72, 73).


[Yin+18] Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. “Gradient diversity: a key ingredient for scalable distributed learning”. In: AISTATS. 2018 (cit. on pp. 163, 166).


[Yun+19] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. “Cutmix: Regularization strategy to train strong classifiers with localizable features”. In: ICCV. 2019 (cit. on pp. 12, 24, 151, 153, 157, 161).


ACRONYMS AND NOTATIONS

Contents
A.1 Acronyms ................................................................. 129
A.2 Notations ................................................................. 131

A.1 Acronyms

AI artificial intelligence
BN batch normalization
CEB conditional entropy bottleneck
CKAC centered kernel alignment complement
CNN convolutional neural network
CV computer vision
DA data augmentation
DE deep ensembling
DL deep learning
DNN deep neural network
DRL deep reinforcement learning
ECE expected calibration error
ENS ensembling
ERM empirical risk minimization
FIM Fisher information matrix
GPU graphics processing unit
GP Gaussian process
GPT generative pre-trained transformers
IB information bottleneck
ID in-distribution
IoU intersection over union
KL Kullback-Leibler divergence
LLM  large language model
LMC  linear mode connectivity
LP   linear probing
MA   moving average
MI   mutual information
MLP  multi-layer perceptron
ML   machine learning
MMD  maximum mean discrepancy
MNI  minimum necessary information
MORL multi-objective reinforcement learning
MSA  memory split advantage
MSDA mixed sample data augmentation
MSE  mean-squared error
NLL\textsubscript{c} calibrated negative log-likelihood
NLP  natural language processing
NTK  neural tangent kernel
OOD  out-of-distribution
PCS  Pareto coverage set
PF   Pareto front
QA   question answering
RGB  red green blue
RKHS reproducing kernel Hilbert space
RLAIF reinforcement learning from AI feedback
RLHF reinforcement learning from human feedback
RL   reinforcement learning
ReLU  rectified linear unit
SAM  sharpness-aware minimization
SGD  stochastic gradient descent
sota state-of-the-art
TS   temperature scaling
VG   visual grounding
VQA  visual question answering
WA   weight averaging
WI   weight interpolation
A.2 Notations

DNN model
Featurizer (encoder)
Classifier (dense layer)
Input
Embedding
Label
Prediction
Number of classes
Number of models in ensembling
Probability density function and measure
Random variables
Ratios
Train (source) domain
Test (target) domain
Training dataset
Training configuration
Learning procedure
Loss function

\( f \) parameterized with weights \( \theta \)
\( \Phi \) (or \( e \)) parameterized with weights \( \phi \)
\( w \) (or \( d \)) parameterized with weights \( \omega \)

\( x \in \mathcal{X} \)
\( z \)
\( y \in \mathcal{Y} \)
\( \hat{y} \in \mathcal{Y} \)
\( K \)
\( M \)
\( p \) and \( P \)
\( X, Y, Z \)
\( \lambda, \mu, \kappa \)

\( S \) with distribution \( p_S \)
\( T \) with distribution \( p_T \)
\( D_S = \{ x_n, y_n \}_{n=1}^{n_S} \)
\( c \)
\( l_S = \{ D_S, c \} \)
\( \ell : \mathcal{Y}^2 \rightarrow \mathbb{R}_+ \)
SOCIETAL IMPACTS

This thesis was crafted within a given social and political context, and we now discuss its potential societal impact.

**Sustainability.** Proposing WA as an alternative to ensembling is a step towards more sustainable DL. By reducing the computational cost at inference, we may diminish the carbon footprint of large DNNs. However, the potential issue is that this might lead to a “rebound effect” where the efficiency gain enables the use of even larger models.

**Decentralization.** We also promote the *updatable machine learning* paradigm [Raf23], which facilitates “the collaborative creation of increasingly sophisticated AI systems” [Ram+23a]. As discussed in Section 7.2.3, this paradigm may further encourage the development of open-source models, potentially leading to the creation of more responsible and reliable AI systems that can adapt and learn in a constantly changing environment. This could help networks represent a more diverse range of opinions (as discussed in Chapter 6) by mitigating the “tyranny of the crowdworker” [Kir+23], where models are “tailored to meet the expectations of […] a small number of crowdworkers primarily based in the US, with little to no representation of broader human cultures, geographies, or languages” [Kir+23], and more generally the cultural hegemony of a few individuals [Dur+23].

**Privacy and fairness.** The parallelization of ensembling approaches is compatible with *federated learning* scenarios [McM+17], where data must remain private. Additionally, by improving generalization, our methods can mitigate the impact of subpopulation shift.

**Transparency.** In DL, decisions made during the training of models shift the responsibility from the problem stakeholders to the system engineers, who need to anticipate the impacts of their choices. We demonstrated that combining multiple models offers an additional level of flexibility: by selecting the weighting coefficients a posteriori, we can improve the transparency [Gab+21] of AI systems, and facilitate regulation by an external non-technical authority. This may prove crucial to ensure the development of fair, unbiased, and inclusive [Aba+16] AIs. This could facilitate tailored generations to minorities by *model personalization* [Sal+23], but also pause risks of echo chambers and bias reinforcement, as discussed in [Kir+23] and Section 6.4.
Sociological analogy. More generally, we showed that the optimal diversity-accuracy trade-off was obtained with large diversity in DL. By analogy, this thesis promotes more diversity in our society [Mul16]: ideas should be shared and debated among members reflecting the diversity of the society’s various components and backgrounds. Actually, in social science, this insight is known as the the Hong-Page theorem [Hon+04]: a group of low-ability, cognitively diverse people can outperform a more uniform group of high-ability experts to solve problems. Academia especially needs this diversity to promote trust in research [Sie+18], to improve quality of the findings [Swa+19], productivity of the teams [Vas+15] and even schooling’s impact [Bow13].
C.1 Proofs for **Ensembling**

We detail the proofs for the theoretical arguments in Chapter 2.

- Appendix C.1.1 proves the link between variance and diversity shift (Proposition 2.1).
- Appendix C.1.2 proves the link between bias and correlation shift (Proposition 2.2).
- Appendix C.1.3 proves the bias-variance-covariance decomposition for ensembling (Proposition 2.3).

C.1.1 Variance and diversity shift **Proposition 2.1**

We prove the link between variance and diversity shift. Our proof builds upon the similarity between DNNs and GPs in the interpolating regime, detailed in Appendix C.1.1.1. We discuss our simplifying Assumption 2.2 in Appendix C.1.1.2. We present our final proof in Appendix C.1.1.3.

C.1.1.1 Deep neural networks as GPs **Assumption 2.1**

We fix $\mathcal{D}_S, \mathcal{D}_T$ and denote $X_{\mathcal{D}_S} = \{x_{S_{(x,y)}}\}_{(x,y)\in\mathcal{D}_S}, X_{\mathcal{D}_T} = \{x_{T_{(x,y)}}\}_{(x,y)\in\mathcal{D}_T}$ their respective input supports. We fix the initialization of the network. $l_S$ encapsulates all other sources of randomness.
Lemma C.1 (Inspired from [Ras03]). Given a NN \( f(\cdot, \theta(l_S)) \) under Assumption 2.1, we denote \( K \) its NTK and \( K(X_{D_S}, X_{D_T}) = (K(x, x'))_{x, x' \in X_{D_S}} \in \mathbb{R}^{n_S \times n_S} \). Given \( x \in X \), we denote \( K(x, X_{D_S}) = [K(x, x')]_{x \in X_{D_S}} \in \mathbb{R}^{n_S} \). Then:

\[
\text{var}(x) = K(x, x) - K(x, X_{D_S})K(X_{D_S}, X_{D_S})^{-1}K(x, X_{D_S})^T.
\] (C.1)

Proof. Under Assumption 2.1, DNNs are equivalent to GPs. \( \text{var}(x) \) is the formula of the variance of the GP posterior given by Eq. (2.26) in [Ras03], when conditioned on \( D_S \). This formula thus also applies to the variance \( f(\cdot, \theta(l_S)) \) when \( l_S \) varies (at fixed \( D_S \) and initialization).

C.1.1.2 Discussion of the same norm and low similarity Assumption 2.2

Lemma C.1 shows that the variance only depends on the input distributions without involving the label distributions. This formula highlights that the variance is related to shifts in input similarities (measured by \( K \)) between \( X_{D_S} \) and \( X_{D_T} \). Yet, a more refined analysis of the variance requires additional assumptions, in particular to obtain a closed-form expression of \( K(X_{D_S}, X_{D_S})^{-1} \). Assumption 2.2 is useful because then \( K(X_{D_S}, X_{D_S}) \) is diagonally dominant and can be approximately inverted (see full proof in Appendix C.1.1.3).

The first part of Assumption 2.2 assumes that \( \exists \lambda_S \) such that all training inputs \( x_S \in X_{D_S} \) verify \( K(x_S, x_S) = \lambda_S \). Note that this equality is standard in some kernel machine algorithms [Ah-10; Gho+21; Ren05] and is usually achieved by replacing \( K(x, x') \) by \( \frac{\lambda_S}{\sqrt{K(x, x') \sqrt{K(x', x')}}} \forall (x, x') \in (X_{D_S} \cup X_{D_T})^2 \). In the NTK literature, this equality is achieved without changing the kernel by normalizing the samples of \( X_{D_S} \) such that they lie on the hypersphere; this input preprocessing was used in [Lee+17]. This is theoretically based: for example, the NTK \( K(x, x') \) for an architecture with an initial fully connected layer only depends on \( \|x\|, \|x'\|, \langle x, x' \rangle \) [Yan+19a]. Thus in the case where all samples from \( X_{D_S} \) are preprocessed to have the same norm, the value of \( K(x_S, x_S) \) does not depend on \( x_S \in X_{D_S} \); we denote \( \lambda_S \) the corresponding value.

The second part of Assumption 2.2 states that \( \exists \lambda_S \), s.t. \( \forall x_S \in X_{D_S}^2, x_S \neq x' \Rightarrow |K(x_S, x_S')| \leq \epsilon, \) i.e., that training samples are dissimilar and do not interact. This diagonal structure of the NTK [Jac+18], with diagonal values larger than non-diagonal ones, is consistent with empirical observations from [Sel+22] at initialization. Theoretically, this is reasonable if \( K \) is close to the RBF kernel \( K_h(x, x') = \exp(-\|x - x'\|^2 / h^2) \) where \( h \) would be the bandwidth: in this case, Assumption 2.2 is satisfied when training inputs are distant in pixel space.

We now provide an analysis of the variance where the diagonal assumption is relaxed. Specifically, we provide the sketch for proving an upper-bound of the variance when the NTK has a block-diagonal structure. This is indeed closer to the empirical observations in [Sel+22] at the end of training, consistently with the local elasticity property of NNs [He+20]. We then consider the dataset \( d_{S'} \subset D_S \) made of one sample per block, to which Assumption 2.2 applies. As decreasing the size of a training dataset empirically reduces...
variance [Bra+99], the variance of $f$ trained on $D_S$ is upper-bounded by the variance of $f$ trained on $d_S$; the latter is given by applying Proposition 2.1 to $d_S$. We believe that the proper formulation of this idea is beyond the scope of this proof and best left for future theoretical work.

### C.1.1.3 Expression of OOD variance

We now relate the second term on the r.h.s. to a MMD distance. As $K$ is a kernel, $K^2$ is a kernel and its MMD between $X_{D_S}$ and $X_{D_T}$ is per [Gre+12]:

$$\text{MMD}^2(X_{D_S}, X_{D_T}) = \mathbb{E}_{x_S \neq x'_S \in X_{D_S}}[K^2(x_S, x'_S)] + \mathbb{E}_{x_T \neq x'_T \in X_{D_T}}[K^2(x_T, x'_T)] - 2\mathbb{E}_{x_S \in X_{D_S}}\mathbb{E}_{x_T \in X_{D_T}}[K^2(x_T, x_S)].$$

---

**Proof.** Our proof is original and is based on the posterior form of GPs in Lemma C.1. Given $D_S$, we recall Equation (C.1) that states $\forall x \in \mathcal{X}$:

$$\text{var}(x) = K(x, x) - K(x, X_{D_S})K(X_{D_S}, X_{D_S})^{-1}K(x, X_{D_S})^\top.$$  

Denoting $B = K(X_{D_S}, X_{D_S})^{-1}$ with symmetric coefficients $b_{i,j} = b_{j,i}$, then

$$\text{var}(x) = K(x, x) - \sum_{1 \leq j \leq n_S} b_{i,j}K(x, x^j_S)K(x, x^j_S) = K(x, x) - \frac{n_S}{\lambda_S} E_{x_S \in X_{D_S}}[K^2(x, x_S)]. \quad (C.2)$$

**Assumption 2.2** states that $K(X_{D_S}, X_{D_S}) = A + H$ where $A = \lambda_S I_{n_S}$ and $H = (h_{i,j})_{1 \leq i, j \leq n_S}$ with $h_{i,i} = 0$ and $\max_{i,j} |h_{i,j}| \leq \epsilon$.

We fix $x_T \in X_{D_T}$ and determine the form of $B^{-1}$ in two cases: $\epsilon = 0$ and $\epsilon \neq 0$.

**Case when $\epsilon = 0$** We first derive a simplified result, when $\epsilon = 0$.

Then $b_{i,i} = \frac{1}{\lambda_S}$ and $b_{i,j} = 0$ s.t.

$$\text{var}(x_T) = K(x_T, x_T) - \sum_{x_S \in X_{D_S}} \frac{K(x_T, x_S)^2}{\lambda_S} = K(x, x) - \frac{n_S}{\lambda_S} E_{x_S \in X_{D_S}}[K^2(x, x_S)].$$

We can then write:

$$E_{x_T \in X_{D_T}}[\text{var}(x_T)] = E_{x_T \in X_{D_T}}[K(x_T, x_T)] - \frac{n_S}{\lambda_S} E_{x_T \in X_{D_T}}[E_{x_S \in X_{D_S}}[K^2(x_T, x_S)]]$$

$$E_{x_T \in X_{D_T}}[\text{var}(x_T)] = \lambda_T - \frac{n_S}{\lambda_S} E_{x_S \in X_{D_S}, x_T \in X_{D_T}}[K^2(x_T, x_S)].$$

We now relate the second term on the r.h.s. to a MMD distance. As $K$ is a kernel, $K^2$ is a kernel and its MMD between $X_{D_S}$ and $X_{D_T}$ is per [Gre+12]:

$$\text{MMD}^2(X_{D_S}, X_{D_T}) = E_{x_S \neq x'_S \in X_{D_S}}[K^2(x_S, x'_S)] + E_{x_T \neq x'_T \in X_{D_T}}[K^2(x_T, x'_T)] - 2E_{x_S \in X_{D_S}, x_T \in X_{D_T}}[K^2(x_T, x_S)].$$
Finally, because $\epsilon = 0$, $\mathbb{E}_{x_S \neq x'_S \in X_D^2} K^2 (x_S, x'_S) = 0$ s.t.
\[
\mathbb{E}_{x_T \in X_D} \left[ \operatorname{var}(x_T) \right] = \frac{n_S}{2\lambda_S} \operatorname{MMD}^2(X_D, X_D) + \lambda_T - \frac{n_S}{2\lambda_S} \left( \mathbb{E}_{x_T \neq x'_T \in X_D^2} K^2 (x_T, x'_T) \right) = \frac{n_S}{2\lambda_S} \operatorname{MMD}^2(X_D, X_D) + \lambda_T - \frac{n_S}{2\lambda_S} \beta_T.
\]
We recover the same expression with a $O(\epsilon)$ in the general setting where $\epsilon \neq 0$.

**Case when $\epsilon \neq 0$** We denote $I : \{ \mathbb{GL}_n (\mathbb{R}) \to \mathbb{GL}_n (\mathbb{R}), A \mapsto A^{-1} \}$ the inversion function defined on $\mathbb{GL}_n (\mathbb{R})$, the set of invertible matrices of $\mathbb{GL}_n (\mathbb{R})$.

The function $I$ is differentiable [Mag+19] in all $A \in \mathbb{GL}_n (\mathbb{R})$ with its differentiate given by the linear application $dI_A : \{ \mathbb{M}_n (\mathbb{R}) \to \mathbb{M}_n (\mathbb{R}), H \mapsto -A^{-1}HA^{-1} \}$. Therefore, we can perform a Taylor expansion of $I$ at the first order at $A$:
\[
I(A + H) = I(A) + dI_A(H) + o(\|H\|),
\]
\[
(A + H)^{-1} = A^{-1} - A^{-1}HA^{-1} + o(\|H\|).
\]
where $\|H\| \leq n_S \epsilon = O(\epsilon)$. Thus,
\[
(\lambda_S I_n + H)^{-1} = (\lambda_S I_n)^{-1} - (\lambda_S I_n)^{-1} H (\lambda_S I_n)^{-1} + O(\epsilon) = \frac{1}{\lambda_S} I_n - \frac{1}{\lambda_S} H + O(\epsilon),
\]
\[
\forall i \in [1, n_S], b_{ii} = \frac{1}{\lambda_S} - \frac{1}{\lambda_S} h_{ii} + o(\epsilon) = \frac{1}{\lambda_S} + O(\epsilon),
\]
\[
\forall i \neq j \in [1, n_S], b_{ij} = -\frac{1}{\lambda_S} h_{ij} + o(\epsilon) = O(\epsilon).
\]

Therefore, when $\epsilon$ is small, Equation (C.2) can be developed into:
\[
\operatorname{var}(x_T) = K(x_T, x_T) - \sum_{x_S \in X_D} \left( \frac{1}{\lambda_S} + O(\epsilon) \right) K(x_T, x_S)^2 + O(\epsilon)
\]
\[
= K(x_T, x_T) - \frac{n_S}{\lambda_S} \mathbb{E}_{x_S \in X_D} [K(x_T, x_S)^2] + O(\epsilon)
\]
Following the derivation for the case $\epsilon = 0$, and remarking that under Assumption 2.2 we have $\mathbb{E}_{x_S \neq x'_S \in X_D^2} K^2 (x_S, x'_S) = O(\epsilon^2)$, yields:
\[
\mathbb{E}_{x_T \in X_D} \left[ \operatorname{var}(x_T) \right] = \frac{n_S}{2\lambda_S} \operatorname{MMD}^2(X_D, X_D) + \lambda_T - \frac{n_S}{2\lambda_S} \beta_T + O(\epsilon).
\]
C.1.2 Bias and correlation shift (and support mismatch) Proposition 2.2

We first present in Appendix C.1.2.1 a decomposition of the bias without any assumptions, and then prove our results with the simplifying assumption Assumption 2.3.

C.1.2.1 OOD bias

Proposition (OOD bias). Denoting \( \bar{f}_S (x) = \mathbb{E}_{l_S}[f (x, \theta (l_S))] \), the bias is:

\[
\begin{align*}
\mathbb{E}_{(x,y) \sim p_T} [\text{bias}^2(x, y)] &= \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T (x) - f_S (x))^2 p_T(x)dx \quad \text{(Correlation shift)} \\
&+ \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_S(x) - \bar{f}_S(x))^2 p_T(x)dx \quad \text{(Weighted ID bias)} \\
&+ \int_{\mathcal{X}_T \cap \mathcal{X}_S} 2 (f_T(x) - f_S(x))(f_S(x) - \bar{f}_S(x)) p_T(x)dx \quad \text{(Interaction ID bias and corr. shift)} \\
&+ \int_{\mathcal{X}_T \setminus \mathcal{X}_S} (f_T(x) - \bar{f}_S(x))^2 p_T(x)dx. \quad \text{(Support mismatch)}
\end{align*}
\]

Proof. This proof is original and based on splitting the OOD bias in and out of \( \mathcal{X}_S \):

\[
\begin{align*}
\mathbb{E}_{(x,y) \sim p_T} [\text{bias}^2(x, y)] &= \mathbb{E}_{(x,y) \sim p_T} (y - \bar{f}_S (x))^2 \\
&= \int_{\mathcal{X}_T} (f_T (x) - \bar{f}_S(x))^2 p_T(x)dx \\
&= \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T (x) - \bar{f}_S(x))^2 p_T(x)dx + \int_{\mathcal{X}_T \setminus \mathcal{X}_S} (f_T (x) - \bar{f}_S(x))^2 p_T(x)dx.
\end{align*}
\]

To decompose the first term, we write \( \forall x \in \mathcal{X}_S, -\bar{f}_S (x) = -f_S (x) + (f_S(x) - \bar{f}_S (x)) \).

\[
\begin{align*}
\int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T (x) - \bar{f}_S(x))^2 p_T(x)dx &= \int_{\mathcal{X}_T \cap \mathcal{X}_S} ((f_T (x) - f_S (x)) + (f_s(x) - \bar{f}_S (x)))^2 p_T(x)dx \\
&= \int_{\mathcal{X}_T \cap \mathcal{X}_S} (f_T (x) - f_S (x))^2 p_T(x)dx + \int_{\mathcal{X}_T \setminus \mathcal{X}_S} (f_S(x) - \bar{f}_S(x))^2 p_T(x)dx \\
&+ \int_{\mathcal{X}_T \setminus \mathcal{X}_S} 2 (f_T(x) - f_S(x))(f_S(x) - \bar{f}_S(x)) p_T(x)dx.
\end{align*}
\]

The four terms can be qualitatively analyzed:

- The first term measures differences between train and test labelling function. By rewriting \( \forall x \in \mathcal{X}_T \cap \mathcal{X}_S, f_T(x) = \mathbb{E}_{p_T}[Y|X = x] \) and \( f_S(x) = \mathbb{E}_{p_S}[Y|X = x] \), this term measures whether conditional distributions differ. This recovers a similar expression to the correlation shift formula from [Ye+22].
• The second term is exactly the ID bias, but weighted by the marginal distribution $p_T(X)$.

• The third term $\int_{X_T \cap X_S} 2 (f_T(x) - f_S(x)) (f_S(x) - \bar{f}_S(x)) p_T(x) dx$ measures to what extent the ID bias compensates the correlation shift. It can be negative if (by chance) the ID bias goes in opposite direction to the correlation shift.

• The last term measures support mismatch between test and train marginal distributions. It lead to the “No free lunch for learning representations for DG” in [Rua+22]. The error is irreducible because “outside of the source domain, the label distribution is unconstrained”: “for any domain which gives some probability mass on an example that has not been seen during training, then all […] labels for that example” are possible.

C.1.2.2 OOD bias when small ID bias

We now prove Proposition 2.2 under Assumption 2.3.

Proof. We simplify the second and third terms from Appendix C.1.2.1 under Assumption 2.3.

The second term is $\int_{X_T \cap X_S} (f_S(x) - \bar{f}_S(x))^2 p_T(x) dx$. Under Assumption 2.3, $|f_S(x) - \bar{f}_S(x)| \leq \epsilon$. Thus the second term is $O(\epsilon^2)$.

The third term is $\int_{X_T \cap X_S} 2 (f_T(x) - f_S(x)) (f_S(x) - \bar{f}_S(x)) p_T(x) dx$. As $f_T - f_S$ is bounded on $X_S \cap X_T$, $\exists K \geq 0$ such that $\forall x \in X_S$,

$$| (f_T(x) - f_S(x)) (f_S(x) - \bar{f}_S(x)) p_T(x) | \leq K | f_S(x) - \bar{f}_S(x) | p_T(x) = O(\epsilon) p_T(x).$$

Thus the third term is $O(\epsilon)$.

Finally, note that we cannot say anything about $\bar{f}_S(x)$ when $x \in X_T \setminus X_S$.

To prove the previous equality, we needed a bounded difference between labeling functions $f_T - f_S$ on $X_T \cap X_S$. We relax this bounded assumption to obtain an inequality in the following Proposition C.1.

Proposition C.1 (OOD bias when small ID bias without bounded difference between labeling functions). Under Assumption 2.3,

$$E_{(x,y) \sim p_T}[bias^2(x,y)] \leq 2 \times Correlation ~shift + Support ~mismatch + O(\epsilon^2) \quad (C.3)$$
Proof. We follow the same proof as in Appendix C.1.2.1, except that we now use: \((a+b)^2 \leq 2(a^2 + b^2)\). Then,

\[
\int_{X_T \cap X_S} (f_T(x) - \bar{f}_S(x))^2 p_T(x) dx = \int_{X_T \cap X_S} \left((f_T(x) - f_S(x)) + (f_S(x) - \bar{f}_S(x))\right)^2 p_T(x) dx \\
\leq 2 \times \int_{X_T \cap X_S} (f_T(x) - f_S(x))^2 + (f_S(x) - \bar{f}_S(x))^2 p_T(x) dx \\
\leq 2 \times \int_{X_T \cap X_S} (f_T(x) - f_S(x))^2 p_T(x) dx + 2 \times \int_{X_T \cap X_S} \epsilon^2 p_T(x) dx \\
\leq 2 \times \int_{X_T \cap X_S} (f_T(x) - f_S(x))^2 p_T(x) dx + O(\epsilon^2)
\]

\[
\square
\]

C.1.3 Proof of the bias-variance-covariance decomposition Proposition 2.3

Proof. This proof recovers the bias-variance-covariance decomposition from [Ued+96; Bro+05a] of ensembling, with i.d. learning procedures.

With \(\bar{f}_S(x) = \mathbb{E}_{l_S} [f(x, \theta(l_S))]\), we recall the bias-variance decomposition [Koh+96] (Equation (BV)):

\[
\mathbb{E}_{l_S} \mathcal{E}_T(\theta(l_S)) = \mathbb{E}_{(x,y) \sim p_T} [\text{bias}^2(x,y) + \text{var}(x)],
\]

where \(\text{bias}(x,y) = \text{Bias}\{f(x,y)\} = y - \bar{f}_S(x)\), and \(\text{var}(x) = \text{Var}\{f[x]\} = \mathbb{E}_{l_S} \left[(f(x, \theta(l_S)) - \bar{f}_S(x))^2 \right]\).

Using \(f_{\text{ENS}} = f_{\text{ENS}}(\cdot, \{\theta(l^{(i)}_S)\}_{i=1}^M) = \frac{1}{M} \sum_{i=1}^M f(\cdot, \theta(l^{(i)}_S))\) in this decomposition yields,

\[
\mathbb{E}_{L^M_S} \mathcal{E}_T(\theta(l^{(i)}_S))_{i=1}^M = \mathbb{E}_{x \sim p_T} \left[\text{Bias}\{f_{\text{ENS}} | (x,y)\}^2 + \text{Var}\{f_{\text{ENS}} | x\}\right]. \tag{C.4}
\]

As \(f_{\text{ENS}}\) depends on \(L^M_S\), we extend the bias into:

\[
\text{Bias}\{f_{\text{ENS}} | (x,y)\} = y - \mathbb{E}_{L^M_S} \left[\frac{1}{M} \sum_{i=1}^M f(x, \theta(l^{(i)}_S))\right] = y - \frac{1}{M} \sum_{i=1}^M \mathbb{E}_{l^{(i)}_S} \left[f(x, \theta(l^{(i)}_S))\right]
\]

Under i.d. \(L^M_S = \{l^{(i)}_S\}_{i=1}^M\),

\[
\frac{1}{M} \sum_{i=1}^M \mathbb{E}_{l^{(i)}_S} \left[y - f(x, \theta(l^{(i)}_S))\right] = \mathbb{E}_{l_S} [y - f(x, \theta(l_S))] = \text{Bias}\{f| (x,y)\}.
\]

Thus the bias of ENS is the same as for a single member of the ensemble.
Regarding the variance:

\[
\text{Var} \{ f_{\text{ENS}} \mid x \} = \mathbb{E}_{L^M_S} \left[ \left( \frac{1}{M} \sum_{i=1}^{M} f(x, \theta(l_S^{(i)})) - \mathbb{E}_{L^M_S} \left[ \frac{1}{M} \sum_{i=1}^{M} f(x, \theta(l_S^{(i)})) \right] \right)^2 \right].
\]

Under i.d. \( L^M_S = \{ l_S^{(i)} \}_{i=1}^{M} \),

\[
\text{Var} \{ f_{\text{ENS}} \mid x \} = \frac{1}{M^2} \sum_{i=1}^{M} \mathbb{E}_{l_S} \left[ (f(x, \theta(l_S)) - \mathbb{E}_{l_S} [f(x, \theta(l_S))])^2 \right] + \\
\frac{1}{M^2} \sum_{i} \sum_{i' \neq i} \mathbb{E}_{l_S, l'_S} \left[ (f(x, \theta(l_S)) - \mathbb{E}_{l_S} [f(x, \theta(l_S))])(f(x, \theta(l'_S)) - \mathbb{E}_{l'_S} [f(x, \theta(l'_S))]) \right]
\]

\[
= \frac{1}{M} \mathbb{E}_{l_S} \left[ (f(x, \theta(l_S)) - \mathbb{E}_{l_S} [f(x, \theta(l_S))])^2 \right] + \\
\frac{M - 1}{M} \mathbb{E}_{l_S, l'_S} \left[ (f(x, \theta(l_S)) - \mathbb{E}_{l_S} [f(x, \theta(l_S))])(f(x, \theta(l'_S)) - \mathbb{E}_{l'_S} [f(x, \theta(l'_S))]) \right]
\]

\[
= \frac{1}{M} \text{var} (x) + \left( 1 - \frac{1}{M} \right) \text{cov} (x).
\]

The variance is split into the variance of a single member divided by \( M \) and a new covariance. \( \square \)
C.2 Theoretical insights for DICE

C.2.1 KL between Gaussians

The Kullback-Leibler divergence (KL) divergence [Kul59] between two Gaussian distributions takes a particularly simple form:

\[
D_{KL}(e(z|x)\|b(z|y)) = \log \frac{b^\sigma(y)}{e^\sigma(x)} + \frac{(e^\mu(x) - b^\mu(y))^2}{2b^\sigma(y)^2} - \frac{1}{2}
\]

(Gaussian param.)

\[
= \frac{1}{2} \left[ (1 + e^\sigma(x)^2 - \log(e^\sigma(x)^2)) + (e^\mu(x) - b^\mu(y))^2 \right].
\]

\(b^\sigma(y) = 1\)

The variance component forces the predicted variance \(e^\sigma(x)\) to be close to \(b^\sigma(y) = 1\). The mean component forces the class-embedding \(b^\mu(y)\) to converge to the average of the different elements in its class.

C.2.2 Difference between VCEB and VIB

In [Fis20], CEB is variationally upper bounded by VCEB. We detail the computations:

\[
\text{CEB}_{\beta_{s,\epsilon}}(Z) = \frac{1}{\beta_{s,\epsilon}} I(X; Z|Y) - I(Y; Z) \quad \text{(Definition)}
\]

\[
= \frac{1}{\beta_{s,\epsilon}} [I(X, Y; Z) - I(Y; Z)] - I(Y; Z) \quad \text{(Chain rule)}
\]

\[
= \frac{1}{\beta_{s,\epsilon}} [I(X; Z) - I(Y; Z)] - I(Y; Z) \quad \text{(Markov assumptions)}
\]

\[
= \frac{1}{\beta_{s,\epsilon}} [-H(Z|X) + H(Z|Y)] - [H(Y) - H(Y|Z)] \quad \text{(MI as diff. of 2 ent.)}
\]

\[
\leq \frac{1}{\beta_{s,\epsilon}} [-H(Z|X) + H(Z|Y)] - [-H(Y|Z)] \quad \text{(Non-negativity of ent.)}
\]

\[
= \int \left( \frac{1}{\beta_{s,\epsilon}} \log \frac{e(z|x)}{p(z|y)} \right) \log p(y|z) p(x, y, z) dx dy dz \quad \text{(Definition of ent.)}
\]

\[
\leq \int \left( \frac{1}{\beta_{s,\epsilon}} \log \frac{e(z|x)}{b(z|y)} \right) \log d(y|z) p(x, y) e(z|x) dx dy dz \quad \text{(Variational approx.)}
\]

\[
\approx \frac{1}{n_s} \sum_{n=1}^{n_s} \left( \frac{1}{\beta_{s,\epsilon}} \log \frac{e(z|x_n)}{b(z|y_n)} - \log d(y_n|z) e(z|x_n) dx_n \right) \quad \text{(Empirical data distrib.)}
\]

\[
\approx \text{VCEB}_{\beta_{s,\epsilon}}(\theta = \{e, b, d\}), \quad \text{(Reparameterization trick)}
\]

where, by introducing \(\epsilon\) such as \(z = e(x, \epsilon)\),

\[
\text{VCEB}_{\beta_{s,\epsilon}}(\theta = \{e, b, d\}) = \frac{1}{n_s} \sum_{n=1}^{n_s} \left( \frac{1}{\beta_{s,\epsilon}} D_{KL}(e(z|x_n)\|b(z|y_n)) - \mathbb{E}_e \log d(y_n|e(x_n, \epsilon)) \right).
\]

As a reminder, [Ale+17] upper bounded: \(\text{IB}_{\beta_{l,b}}(Z) = \frac{1}{\beta_{l,b}} I(X; Z) - I(Y; Z)\) by:

\[
\text{VIB}_{\beta_{l,b}}(\theta = \{e, b, d\}) = \frac{1}{n_s} \sum_{n=1}^{n_s} \left( \frac{1}{\beta_{l,b}} D_{KL}(e(z|x_n)\|b(z)) - \mathbb{E}_e \log d(y_n|e(x_n, \epsilon)) \right). \quad \text{(C.5)}
\]
In VIB, all features distribution $e(z|x)$ are moved towards the same class-agnostic distribution $b(z) \sim N(\mu, \sigma)$, independently of $y$. In VCEB, $e(z|x)$ are moved towards the class conditional marginal $b^\mu(y) \sim N(b^\mu(y), b^\sigma(y))$. This is the unique difference between VIB and VCEB. VIB leads to a looser approximation with more bias than VCEB.
C.3 Proof for DIVERSE WEIGHT AVERAGING

Below we detail the (simple) proof of Lemma 4.1, validating the similarity between weight averaging and functional ensembling.

Proof. **Functional approximation.** With a Taylor expansion at the first order of the models’ predictions w.r.t. parameters $\theta$:

$$ f_{\theta_i} = f_{WA} + \nabla f_{WA}^T \Delta_i + O \left( \| \Delta_i \|^2_2 \right) $$

$$ f_{ENS} - f_{WA} = \frac{1}{M} \sum_{i=1}^{M} \left( \nabla f_{WA}^T \Delta_i + O \left( \| \Delta_i \|^2_2 \right) \right) $$

Therefore, because $\sum_{i=1}^{M} \Delta_i = 0$,

$$ f_{ENS} - f_{WA} = O \left( \Delta^2 \right) \text{ where } \Delta = \max_{i=1}^{M} \| \Delta_i \|^2_2. \quad (C.6) $$

**Loss approximation.** With a Taylor expansion at the zeroth order of the loss w.r.t. its first input and injecting Equation (C.6):

$$ \ell (f_{ENS}(x); y) = \ell (f_{WA}(x); y) + O \left( \| f_{ENS}(x) - f_{WA}(x) \|^2_2 \right) $$

$$ \ell (f_{ENS}(x); y) = \ell (f_{WA}(x); y) + O \left( \Delta^2 \right). $$

$\square$


\section{C.4 Proofs for Rewarded Soups}

In Chapter 6, we proved the Pareto optimality for Hessians proportional to the identity. We now consider the more complex case with the relaxed Assumption C. For simplicity, we only consider \( N = 2 \) rewards \( R_1 \) and \( R_2 \).

\textbf{Assumption C.1} (Diagonal Hessians). The rewards are quadratic, with Hessians diagonal negative definite. Specifically, we can write for \( i \in \{1, 2\} \):

\[
\forall \theta = (\theta^1, \ldots, \theta^d) \in \Theta, \quad R_i(\theta) = R_i(\theta_i) - \sum_{j=1}^{d} \eta^j_i (\theta^j - \theta^j_i)^2,
\]

where \( (\eta^1_i, \ldots, \eta^d_i) \in \{\mathbb{R}^+\}^d \) and \( \theta_i = (\theta^1_i, \ldots, \theta^d_i) \) is the global maximum for reward \( R_i \).

\textbf{Remark C.1}. This diagonal Assumption C.1 of the Hessian is common: for example in optimization [LeC+12; Kin+15], to prune networks [LeC+90] or in out-of-distribution generalization [Ram+22a]. This strong assumption is supported by the empirical observation [Bec+88] that Hessians are diagonally dominant, in particular at the end of training. Also, we note that our findings remain valid assuming only that the Hessians are co-diagonalizable.

\textbf{Lemma C.2}. We consider the user’s reward \( R_\mu = (1 - \hat{\mu}) \times R_1 + \mu \times R_2 \) with \( \hat{\mu} \in [0, 1] \), and

\[
\Delta R_\mu = \max_{\theta \in \Theta} R_\mu(\theta) - \max_{\lambda \in [0, 1]} R_\mu ((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2).
\]

\( \Delta R_\mu \) corresponds to the difference in terms of \( R_\mu \) between the global maximum and the maximum reachable by weight interpolation through rewarded soups (with a single interpolating coefficient for all dimensions). Then, under Assumption C.1, we have:

\[
\Delta R_\mu \leq \frac{\hat{\mu}^2 (1 - \hat{\mu})^2 (M \Delta_1 - \Delta_2) (M \Delta_2 - \Delta_1)}{(\hat{\mu}(1 - \hat{\mu})(M - 1)^2 + M) ((1 - \hat{\mu}) \Delta_1 + \hat{\mu} \Delta_2)},
\]

where \( M = \max_{j \in \{1, \ldots, d\}} \max \left( \frac{\eta^j_1}{\eta^j_2}, \frac{\eta^j_2}{\eta^j_1} \right) \) is the maximum of eigenvalues ratio, \( \Delta_1 = R_1(\theta_1) - R_1(\theta_2) \) and \( \Delta_2 = R_2(\theta_2) - R_2(\theta_1) \).

When \( \Delta_1 = \Delta_2 \), the bound simplifies into:

\[
\Delta R_\mu \leq \frac{\hat{\mu}^2 (1 - \hat{\mu})^2 (M - 1)^2}{\hat{\mu}(1 - \hat{\mu})(M - 1)^2 + M} \Delta_1
\]

Furthermore, when the Hessians are equal, then \( M = 1 \) and \( \Delta R_\mu = 0 \); RS is optimal.

\textbf{Proof}. This novel proof is in three steps. First, we find \( \hat{\theta} \) maximizing \( R_{\hat{\mu}}(\theta) \) for \( \theta \) on the full set of weights \( \Theta \). Second, we find \( \lambda \) maximizing \( R_{\hat{\mu}} ((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2) \) for \( \lambda \in [0, 1] \) and thus defining the best interpolation between the expert weights. Finally, we bound \( \Delta R_{\hat{\mu}} \), the differences between their rewards, by applying the Bhatia-Davis inequality.
**First step.** Let’s first find the maximum of $R_{\hat{\mu}}$ on $\Theta$. Denoting $S = (1 - \hat{\mu}) \times R_1(\theta_1) + \hat{\mu} \times R_2(\theta_2)$, we have for all $\theta \in \Theta$:

$$R_{\hat{\mu}}(\theta) = S - \sum_{j=1}^{d} \left( (1 - \hat{\mu}) \eta^j_1 \left( \theta^j - \theta^j_1 \right) + \hat{\mu} \eta^j_2 \left( \theta^j - \theta^j_2 \right) \right)^2 \tag{C.11}$$

Since $R_{\hat{\mu}}$ is a sum of concave quadratic functions, it has a unique global maximum reached at a point we note $\hat{\theta} = (\hat{\theta}_1, ..., \hat{\theta}_d)$. The global maximum can be computed by differentiating $R_{\hat{\mu}}$ with respect to each variable $\theta^j$, which gives:

$$\hat{\theta}^j = \left( 1 - \hat{\lambda}^j \right) \cdot \theta^j_1 + \hat{\lambda}^j \cdot \theta^j_2$$

where the interpolating coefficients per dimension $\hat{\lambda}^j$ are defined for $j \in \{1, ..., d\}$ as:

$$\hat{\lambda}^j = \frac{\hat{\mu} \eta^j_2}{(1 - \hat{\mu}) \eta^j_1 + \hat{\mu} \eta^j_2} \in [0, 1]. \tag{C.12}$$

**Second step.** With $\lambda \in [0, 1]$ and $\theta = (1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2$, we can write $R_{\hat{\mu}}(\theta)$ as a function of $\lambda$:

$$R_{\hat{\mu}}(\theta) = S - \sum_{j=1}^{d} \left( (1 - \hat{\mu}) \eta^j_1 + \hat{\mu} \eta^j_2 \right) \left( \lambda - \hat{\lambda}^j \right)^2 \tag{C.13}$$

where $p_j$ is defined as $p_j = \left( (1 - \hat{\mu}) \eta^j_1 + \hat{\mu} \eta^j_2 \right) \left( \theta^j_1 - \theta^j_2 \right)^2$.

From Equation (C.13), we can compute the maximum reward obtainable for weight averaging $\max_{\lambda \in [0, 1]} R_{\hat{\mu}}((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2)$. Since the function $\lambda \mapsto R_{\hat{\mu}}((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2)$ is a concave quadratic function, there is a unique value $\lambda$ maximizing $R_{\hat{\mu}}$ equal to

$$\tilde{\lambda} = \frac{\sum_{j=1}^{d} p_j \hat{\lambda}^j}{\sum_{j=1}^{d} p_j}. \tag{C.14}$$

Since all $p_j$ are positive and all $\hat{\lambda}^j$ are between 0 and 1, $\tilde{\lambda}$ is also between 0 and 1. Therefore, $R_{\hat{\mu}}((1 - \tilde{\lambda}) \cdot \theta_1 + \tilde{\lambda} \cdot \theta_2)$ is indeed the maximum reward for rewarded soups.
Third step. Applying Equation (C.13) to $\bar{\lambda}$ gives:

$$\Delta R_\mu = R_\mu(\bar{\theta}) - R_\mu((1 - \bar{\lambda}) \cdot \theta_1 + \bar{\lambda} \cdot \theta_2)$$

(C.15)

$$= \sum_{j=1}^d p_j \left( \bar{\lambda} - \hat{\lambda}_j \right)^2$$

(C.16)

$$= \left( \sum_{j=1}^d \frac{p_j}{\sum_{i=1}^n p_i} \left( \bar{\lambda} - \hat{\lambda}_j \right)^2 \right) \left( \sum_{j=1}^n p_j \right)$$

(C.17)

The second term in Equation (C.17) can be simplified as:

$$\sum_{j=1}^d p_j = (1 - \hat{\mu}) \Delta_1 + \hat{\mu} \Delta_2.$$  

(C.18)

The core component of this proof is the upper bounding of the first term in Equation (C.17). The key idea is to recognize the variance of a discrete random variable $\Lambda$ with $\mathbb{P}(\Lambda = \hat{\lambda}_i) = \frac{p_i}{\sum_{j=1}^n p_j}$; then, $\bar{\lambda}$ from Equation (C.14) is actually the expectation of $\Lambda$. Then, we can apply the Bhatia-Davis inequality, as recalled in Equation (C.19), on the variance of a bounded random variable $a \leq \Lambda \leq b$:

$$\text{Var}(\Lambda) \leq (b - \mathbb{E}(\Lambda)) \left( \mathbb{E}(\Lambda) - a \right).$$  

(C.19)

Therefore Equation (C.17) is bounded by:

$$\Delta R_\mu \leq \left( \max_{1 \leq j \leq d} \hat{\lambda}_j - \bar{\lambda} \right) \left( \bar{\lambda} - \min_{1 \leq j \leq d} \hat{\lambda}_j \right) ((1 - \hat{\mu}) \Delta_1 + \hat{\mu} \Delta_2).$$  

(C.20)

Now, we bound the variables $\hat{\lambda}_j$, since $1/M \leq \eta_1^j/\eta_2^j \leq M$. Then for all $j$ we have:

$$\frac{\hat{\mu}}{1 - \hat{\mu}} M + \hat{\mu} \leq \hat{\lambda}_j \leq \frac{\hat{\mu} M}{1 - \hat{\mu}} (1 - \hat{\mu}) M + \hat{\mu} M.$$

(C.21)

and thus:

$$\Delta R_\mu \leq \left( \frac{\hat{\mu} M}{1 + \hat{\mu}(M - 1)} - \bar{\lambda} \right) \left( \bar{\lambda} - \frac{\hat{\mu}}{M - \hat{\mu}(M - 1)} \right) ((1 - \hat{\mu}) \Delta_1 + \hat{\mu} \Delta_2).$$

(C.22)

Finally, noting that $\Delta_i = \sum_{j=1}^d \eta_i^j \left( \theta_1^j - \theta_2^j \right)^2$, we deduce from Equation (C.14) that $\bar{\lambda} = \frac{\hat{\mu} \Delta_2}{(1 - \hat{\mu}) \Delta_1 + \hat{\mu} \Delta_2}$. Replacing this in the previous Equation (C.22) gives the final Equation (C.9), concluding the proof.

\[ \square \]

Remark C.2. As a final remark, please note that the suboptimality of RS comes from the need of having one single interpolating coefficient $\bar{\lambda}$ for all $d$ parameters $(\theta^1, ..., \theta^d)$ of the network. Yet, the advanced merging operations in [Mat+22] remove this constraint, with interpolating coeffi-
cients proportional to the eigenvalues of the Fisher matrices [Fis22], which actually approximate the eigenvalues of the Hessian [Scho: Tho+20]. Combining [Mat+22] and our RS is a promising research direction, the key issue being the computation of the Fisher matrices [Kun+19] for networks with billions of parameters.

We visualize in Figure C.1 the bound given by Lemma C.2. We show that for small values of $M$ like $M = 2$, the value of $R\hat{\mu}$ for RS is quite close to the global optimum. Also, recall that RS theoretically matches this upper bound when $M = 1$. For larger values like $M = 10$, the bound is less tight, and we note that the maximum value of $R\hat{\mu}$ approaches the constant function $1$ as $M \to \infty$.

![Figure C.1](image_url)

**Figure C.1.** Illustration of the bound given by Lemma C.2 under Assumption C.1. For simplicity, we showcase the case where $R_1(\theta_1) = R_2(\theta_2) = 1$, $R_1(\theta_2) = R_2(\theta_1) = 0$, thus $\Delta_1 = \Delta_2 = 1$. In green, we plot the rewards obtained with rewarded soups for the optimal $\bar{\lambda}$, i.e., $R_{\hat{\mu}}((1 - \hat{\lambda}) \cdot \theta_1 + \hat{\lambda} \cdot \theta_2)$, whose value is independent of $M$ in this case. In blues, we plot the maximum value of $R_{\hat{\mu}}$ given by Equation (C.10) in Lemma C.2, for $M = 2$ and $M = 10$. For reference, we also plot the values for the lower bound in the LMC Hypothesis 6.1, i.e., equal to $(1 - \hat{\mu})(1 - \bar{\lambda})R_1(\theta_1) + \hat{\mu}\bar{\lambda}R_2(\theta_2)$. As RS outperforms this lower bound, it validates Hypothesis 6.1 in this case.
MIXMO: MIXING MULTIPLE INPUTS FOR MULTIPLE OUTPUTS VIA DEEP SUBNETWORKS

D.1 Introduction

In deep learning (DL), to improve robustness [Hen+19a; Ova+19] or to win Kaggle competitions, DNNs usually pair up with two practical strategies.

The first strategy is data augmentation (DA), reducing overfitting by diversifying the training samples [Lop+20]: in particular, recent mixed sample data augmentation (MSDA) create artificial samples by mixing multiple inputs and their labels proportionally to a ratio $\lambda$. The seminal Mixup [Zha+18a] linearly interpolates pixels: binary masking MSDAs [Fre+20; Har+20; Kim+20] such as CutMix [Yun+19] have since diversified mixed samples by pasting patches from one image onto another.

The second strategy is ensembling (ENS), the main topic of this thesis. This chapter follows the recent multi-input multi-output (MIMO) [Hav+21; Sof+20], that approximates traditional functional ensembling of models by fitting $M$ independent subnetworks inside a single base network: this is possible as large networks only leverage a subset of their weights [Fra+19]. Specifically, we introduce MixMo, a new generalized framework for learning deep subnetworks. As in MIMO, we consider $M$ (input, label) pairs at the same time in training: $\{(x_i, y_i)\}_{0 \leq i < M}$, as shown on Figure D.1 with $M = 2$. The $M$ inputs are encoded by $M$ separate convolutional layers $\{c_i\}_{0 \leq i < M}$ into a shared latent space before being mixed. The representation is then fed to the core network, which finally branches out into $M$ dense layers $\{d_i\}_{0 \leq i < M}$. The main idea during training to prevent homogenization is that each subnetwork learns to classify only one of the multiple inputs simultaneously provided. Diverse subnetworks naturally emerge as $d_i$ learns to classify $y_i$ from input $x_i$. At inference, the same image is repeated $M$ times: we obtain ensembling “for free” by averaging $M$ predictions.

Our key originality lies in the multi-input mixing block; indeed, the question of how to best mix these multiple inputs has not been studied so far. Should the merging be a basic summation, we would recover MIMO [Hav+21]. Our main intuition is simple: we see summing as a balanced and restrictive form of Mixup [Zha+18a] where $\lambda = \frac{1}{M}$. Then by analogy, we leverage the literature in MSDA to propose novel mixing strategies. In particular, we show that binary masking methods—particularly with rectangular patches from CutMix [Yun+19]—enhances results by making subnetworks stronger and more diverse. We thus create a new Cut-MixMo variant inspired by CutMix, and illustrated in Figure D.1: a patch of features from the first input is pasted into the features from
Figure D.1. – MixMo framework. We embed $M = 2$ inputs into a shared space with convolutional layers ($c_1, c_2$), mix them, pass the embedding through further layers and output 2 predictions via dense layers ($d_1, d_2$). The key point of our MixMo is the mixing block. Mixing with patches performs better than basic summing: 85.40% vs. 83.06% (MIMO [Hav+21]) on CIFAR-100 with WRN-28-10.

We propose a general framework, MixMo, connecting two successful fields: mixing samples data augmentations & multi-input multi-output ensembling (Appendix D.3.1).

We identify the appropriate mixing block to best tackle the diversity/individual accuracy trade-off in subnetworks: our easy to implement Cut-MixMo benefits from the synergy between CutMix and ensembling (Appendix D.3.2).

We design a new weighting of the loss components to properly leverage the asymmetrical inputs mixing (Appendix D.3.3).

We improve performances for image classification on CIFAR-100 and Tiny ImageNet datasets (Appendix D.4). As exhibited by Figure D.2, Cut-MixMo outperforms CutMix, MIMO and deep ensembling (DE), at (almost) the same inference cost as a single network.

The work in this chapter has led to the publication of the following paper: Alexandre Ramé, Remy Sun, and Matthieu Cord. “MixMo: Mixing Multiple Inputs for Multiple Outputs via Deep Subnetworks”. In: ICCV. 2021.
D.2 Related work

Data augmentation. DNNs are known to memorize the training data [Zha+17] and make overconfident predictions [Guo+17] to the detriment of generalization on new test examples. Data augmentations (DA) inflate the training dataset’s size by creating artificial samples from available labeled data. Beyond slight perturbations (e.g., rotation), recent works [Cub+20; Hen+19b; DeV+17b] apply stronger transformations [He+19]. Mixed sample data augmentation (MSDA) recently expanded the notion of DA. From pairs of labeled samples \( \{(x_i, y_i), (x_k, y_k)\} \), they create virtual samples: \( m_x(x_i, x_k, \lambda) = \lambda y_i + (1 - \lambda) y_k \) where \( \lambda \sim \text{Beta}(\alpha, \alpha) \). [Lia+18] shows that mixing the targets differently than this linear interpolation may cause underfitting and unstable learning. Then, approaches mainly focus on developing the most effective input mixing \( m_x \). In [Ino18; Tok+18a; Tok+18b; Zha+18a], \( m_x \) performs a simple linear interpolation between pixels: \( m_x(x_i, x_k, \lambda) = \lambda x_i + (1 - \lambda) x_k \). Then, CutMix [Yun+19] draws from Mixup and CutOut [DeV+17b] by pasting a patch from \( x_k \) onto \( x_i \): \( m_x(x_i, x_k, \lambda) = 1_m \odot x_i + (1 - 1_m) \odot x_k \), where \( \odot \) represents the element-wise product and \( 1_m \) a binary mask with average value \( \lambda \). CutMix randomly samples squares, which often leads to rectangular masks due to boundary effects. Such non-linear binary masking improves generalization [Sum+19; Tak+20] by creating new images with usually disjoint patches [Har+20]. Finally, in addition to Manifold Mixup [Ver+19], only a few works [DeV+17a; Far+20; Li+20a; Yag+19; Yun+19] have tried to mix intermediate latent features as we do.
Ensembling. Like [Wen+21], we explore combining DA with another standard technique in machine learning: ensembling, whose fundamental drawback is the inherent computational and memory overhead. This chapter follows the multi-input multi-output MIMO paper [Hav+21], which achieves ensemble almost “for free”: all of the layers except the first convolutional and last dense layers are shared ($\approx +1\%$ #parameters).

D.3 MixMo

We first introduce the main components of our MixMo strategy, summarized in Figure D.3: we mix multiple inputs to obtain multiple outputs via subnetworks. We highlight the key mixing block combining information from inputs, and our training loss based on a dedicated weighting scheme. We study $M = 2$ subnetworks in this thesis, both for clarity and as it empirically performs best in standard parameterization regimes.

D.3.1 General overview

We leverage a training classification dataset $D_S$ of i.i.d. pairs of associated image/label $\{(x_i, y_i)\}_{i=1}^{n_S}$. We randomly sample a subset of $b$ samples $\{(x_i, y_i)\}_{i \in B}$ that we randomly shuffle via permutation $\pi$. Our training batch is $\{(x_i, x_j), (y_i, y_j)\}_{i \in B, j = \pi(i)}$. The loss $L_{\text{MixMo}}$ is averaged over these $b$ samples: the networks’ weights are updated through backpropagation and gradient descent.

Let’s focus on the training sample $\{(x_0, x_1), (y_0, y_1)\}$. In MixMo, both inputs are separately encoded (see Figure D.1) into the shared latent space with two different convolutional layers (with 3 input channels each and no bias term): $x_0$ via $c_0$ and $x_1$ via $c_1$. To recover a strictly equivalent formulation to MIMO [Hav+21], we simply sum the two encodings: $c_0(x_0) + c_1(x_1)$. Indeed, MIMO merges inputs through channel-wise concatenation in pixels: MIMO’s first convolutional layer (with 6 input channels and no bias term) hides the summing operation in the output channels.

Explicitly highlighting the underlying mixing leads us to consider a generalized multi-input mixing block $\mathcal{M}$. This manifold mixing presents a unique opportunity to tackle the ensemble diversity/individual accuracy trade-off and to improve overall ensemble results (see Appendix D.3.2). The shared representation $\mathcal{M}(c_0(x_0), c_1(x_1))$ feeds the next convolutional layers. We note $\kappa$ the mixing ratio between inputs.

The core network $\mathcal{C}$ handles features that represent both inputs simultaneously. The dense layer $d_0$ predicts $\hat{y}_0 = d_0(\mathcal{C}(\mathcal{M}(c_0(x_0), c_1(x_1))))$ and targets $y_0$, while $d_1$ targets $y_1$. Thus, the training loss is the sum of two cross-entropies $L_{\text{CE}}$ weighted by parametrized function $w_r$ (defined in Appendix D.3.3) to balance the asymmetry when $\kappa \neq 0.5$:

$$L_{\text{MixMo}} = w_r(\kappa) \times L_{\text{CE}}(y_0, \hat{y}_0) + w_r(1 - \kappa) \times L_{\text{CE}}(y_1, \hat{y}_1).$$  \hspace{1cm} (D.1)

At inference, the same input $x$ is repeated twice: the core network $\mathcal{C}$ is fed the sum $c_0(x) + c_1(x)$ that preserves maximum information from both encodings. Then, the diverse
predictions are averaged: \( \frac{1}{2} (\hat{y}_0 + \hat{y}_1) \). This allows us to benefit from functional ensembling in a single forward pass.

**D.3.2 Mixing block**

The mixing block \( M \), combining both inputs into a shared representation, is the cornerstone of MixMo. Our main intuition was to analyze MIMO as a simplified Mixup variant where the mixing ratio \( \kappa \) is fixed to 0.5. MixMo generalized framework encompasses a wider range of variants inspired by MSDA mixing methods. Our first main variant, named Linear-MixMo, fully extends Mixup with the following mixing block:

\[
M_{\text{Linear-MixMo}} (l_0, l_1) = 2 [\kappa l_0 + (1 - \kappa) l_1],
\]

where \( l_0 = c_0(x_0) \), \( l_1 = c_1(x_1) \) and \( \kappa \sim \text{Beta}(\alpha, \alpha) \) and \( \alpha \) the concentration parameter. The second and more effective Cut-MixMo variant adapts the patch mixing from CutMix:

\[
M_{\text{Cut-MixMo}} (l_0, l_1) = 2 [I_M \odot l_0 + (1 - I_M) \odot l_1],
\]

where \( I_M \) is a binary mask with area ratio \( \kappa \sim \text{Beta}(\alpha, \alpha) \), valued at 1 either on a rectangle or on the complementary of a rectangle. In brief, a patch from \( c_0(x_0) \) is pasted onto \( c_1(x_1) \), or vice versa. This binary mixing in Cut-MixMo advantageously replaces the linear interpolation in MIMO and Linear-MixMo: subnetworks are more accurate and more diverse, as shown empirically in Figure D.5.

First, binary mixing in \( M \) trains stronger individual subnetworks for the same reasons why CutMix improves over Mixup. By masking features, we simulate common object occlusion problems. This spreads subnetworks’ focus across different locations: the two classifiers are forced to find information relevant to their assigned input at disjoint locations. This occlusion remains effective as the receptive field in this first shallow latent space remains small.

Secondly, linear interpolation is fundamentally ill-suited to induce diversity as full information is preserved from both inputs. CutMix on the other hand explicitly increases dataset diversity by presenting patches of images that do not normally appear together. Such benefits can be directly transposed to \( M_{\text{Cut-MixMo}} \): binary mixing with patches increases randomness and diversity between the subnetworks. Indeed, in a similar spirit to bagging [Bre96], different samples are given to the subnetworks. By deleting asymmetrical complementary locations from the two inputs, subnetworks will not rely on the same region and information. Overall, they are less likely to collapse on close solutions.

**D.3.3 Loss weighting**

Asymmetries in the mixing mechanism can cause one input to overshadow the other. Notably when \( \kappa \neq 0.5 \), the predominant input may be easier to predict. We seek a weighting function \( w_r \) to balance the relative importance of the two \( \mathcal{L}_{\text{CE}} \) in \( \mathcal{L}_{\text{MixMo}} \). This
weighting modifies the effective learning rate, how gradients flow in the network and overall how mixed information is represented in features. We then propose to weight via the parametrized:

$$w_r(\kappa) = 2 \frac{\kappa^{1/r}}{\kappa^{1/r} + (1 - \kappa)^{1/r}}.$$  \hfill (D.4)

This defines a family of functions indexed by the parameter $r$, visualized for $r = 3$ in red on Figure D.3. This power law provides a natural relaxation between two extreme configurations. The first extreme, $r = 1$, $w_1(\kappa) = 2\kappa$, is in line with linear label interpolation in MSDA. The resulting imbalance in each subnetwork’s contribution to $L_{\text{MixMo}}$ causes lopsided updates. While it promotes diversity, it also reduces regularization: the overshadowed input has a reduced impact on the loss. The opposite extreme, $r \to \infty$, $w_\infty(\kappa) \to 1$, removes reweighting. Consequently, $w_r$ inflates the importance of hard under-represented inputs, à la Focal Loss [Lin+17]. However, minimizing the role of the predominant inputs destabilizes training. Overall, we empirically observe that moderate values of $r$ perform best as they trade off pros and cons from both extremes.

**D.4 Experiments**

**D.4.1 Implementation details**

We evaluate MixMo efficiency on CIFAR-[10,100] [Kri+09], and also consider Tiny ImageNet [Chr+17a] in the paper. We mostly study the Linear-MixMo and Cut-MixMo variants with $M = 2$, with the following hyperparameters values: $r = 3$ and $\alpha = 2$. 
MIMO [Hav+21] refers to linear summing, like Linear-MixMo, but with $\kappa = 0.5$ instead of $\kappa \sim \text{Beta}(\alpha, \alpha)$. Other hyperparameters are taken directly from MIMO [Hav+21].

Different mixing methods create a strong train-test distribution gap [Car+20; Lop+20]. Thus, in Cut-MixMo we actually substitute $\mathcal{M}_{\text{Cut-MixMo}}$ for $\mathcal{M}_{\text{Linear-MixMo}}$ with probability $1 - p$ to accommodate for the summing in $\mathcal{M}$ at inference. We set the probability of patch mixing during training to $p = 0.5$, with linear descent to 0 over the last twelfth of training epochs. When MixMo is combined with Cut-Mix, the pixels inputs are: $(m_x(x_i, x_k, \lambda), m_x(x_j, x_{k'}, \lambda'))$ with interpolated targets $(\lambda y_i + (1 - \lambda)y_k, \lambda'y_j + (1 - \lambda')y_{k'})$, where $k, k'$ are randomly sampled and $\lambda, \lambda' \sim \text{Beta}(1, 1)$. MIMO duplicates samples $b$ times via batch repetition: $x_i$ will be associated with $x_{\pi(i)}$ and $x_{\pi'(i)}$ in the same batch if $b = 2$. As the batch size remains fixed, the count of unique samples per batch and the learning rate is divided by $b$. Conversely, the number of steps is multiplied by $b$. We thus indicate an estimated (training/inference) overhead (w.r.t. vanilla training) in the time column of our tables. Note that some concurrent approaches also lengthen training: e.g., GradAug [Yan+20c] via multiple subnetworks predictions ($\approx \times 3$). We equally track accuracies (Top{$1, 5$}, †) and the calibrated negative log-likelihood (NLL$_c$) (↓). Indeed, [Ash+20] shows that we should compare in-domain uncertainty estimations after temperature scaling (TS) [Guo+17]: we thus split the test set in two and calibrate (after averaging in ensembles) with the temperature optimized on the other half.

### D.4.2 Main results on CIFAR

Table D.1 reports averaged scores over 3 runs for our main experiment on CIFAR with WRN-28-10 [Zag+16]. **Bold** highlights best scores, † marks approaches not re-implemented. Cut-MixMo reaches (85.40% Top1, 0.535 NLL$_c$) on CIFAR-100 with $b = 4$: it surpasses our Linear-MixMo (83.08%, 0.656) and MIMO (83.06%, 0.661). Cut-MixMo is further improved when combined with CutMix (85.77%, 0.524). Results remain strong when $b = 2$: Cut-MixMo (84.38%, 0.563) proves better on its own than DE [Lak+17], and MSDAs like Mixup [Zha+18a; Ver+19] or CutMix [Yun+19]. We see similar trends on CIFAR-10: Cut-MixMo reaches 0.081 in NLL$_c$, 0.079 with CutMix. Yet, the costlier batch augmented Mixup BA [Hof+20] edges it out in Top1.
### Table D.1. – Main MixMo results: WRN-28-10 on CIFAR.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Dataset</th>
<th>CIFAR-100</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Top1 %, †</td>
<td>Top5 %, †</td>
</tr>
<tr>
<td>Vanilla</td>
<td>81.63</td>
<td>95.49</td>
<td>73.9</td>
</tr>
<tr>
<td>Mixup</td>
<td>83.44</td>
<td>95.92</td>
<td>65.7</td>
</tr>
<tr>
<td>Manifold Mixup†</td>
<td>81.96</td>
<td>95.51</td>
<td>73.4</td>
</tr>
<tr>
<td>CutMix</td>
<td>84.05</td>
<td>96.09</td>
<td>64.8</td>
</tr>
<tr>
<td>ResizeMix†</td>
<td>84.31</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Puzzle-Mix†</td>
<td>2/1</td>
<td>84.31</td>
<td>96.46</td>
</tr>
<tr>
<td>GradAug†</td>
<td>3/1</td>
<td>84.14</td>
<td>96.43</td>
</tr>
<tr>
<td>+ CutMix†</td>
<td>85.51</td>
<td>96.86</td>
<td>-</td>
</tr>
<tr>
<td>Mixup BA†</td>
<td>7/1</td>
<td>84.30</td>
<td>-</td>
</tr>
<tr>
<td>DE (2 Nets) + CutMix</td>
<td>2/2</td>
<td>83.17</td>
<td>96.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>85.74</td>
<td>96.82</td>
</tr>
<tr>
<td>MIMO</td>
<td></td>
<td>82.40</td>
<td>95.78</td>
</tr>
<tr>
<td>Linear-MixMo + CutMix</td>
<td>2/1</td>
<td>82.54</td>
<td>95.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>84.69</td>
<td>97.12</td>
</tr>
<tr>
<td>Cut-MixMo + CutMix</td>
<td></td>
<td>84.38</td>
<td>96.94</td>
</tr>
<tr>
<td></td>
<td></td>
<td>85.18</td>
<td>97.20</td>
</tr>
<tr>
<td>MIMO</td>
<td></td>
<td>83.06</td>
<td>96.23</td>
</tr>
<tr>
<td>Linear-MixMo + CutMix</td>
<td>4/1</td>
<td>83.08</td>
<td>96.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>85.47</td>
<td>97.04</td>
</tr>
<tr>
<td>Cut-MixMo + CutMix</td>
<td></td>
<td>85.40</td>
<td>97.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>85.77</td>
<td>97.42</td>
</tr>
</tbody>
</table>

### D.4.3 MixMo efficiency

Figure D.4 shows how MixMo grows stronger than DE (green curves) as width $w$ in WRN-28-$w$ increases. The parameterization becomes appropriate at $w=4$: Cut-MixMo (yellow curves) then matches DE (with half the parameters) in Figure D.4(a) and its subnetworks match a vanilla network in Figure D.4(b). Beyond, MixMo better uses over-parameterization: Cut-MixMo + CutMix surpasses DE + CutMix in NLLc for $w \geq 5$, and this is true in Top1 for $w \geq 10$. Compared to our strong Linear-MixMo + CutMix (purple curves), Cut-MixMo performs similarly in Top1, and better with CutMix for $w \geq 4$. While Linear-MixMo and DE learn from occlusion, Cut-MixMo also benefits from CutMix, notably from the induced label smoothing. Overall, Cut-MixMo, even without CutMix, significantly better estimates uncertainty.
Figure D.4. – Parameters efficiency (metrics/#params). CIFAR-100 with WRN-28-w, b=4. Comparisons between (a) ensemble and some of their (b) individual counterparts.

D.4.4 MixMo analysis on CIFAR-100 w/ WRN-28-10

D.4.4.1 The mixing block \( M \)

We study the impact of patch mixing through the lens of the ensemble diversity/individual accuracy trade-off. We measure diversity via the pairwise ratio-error [Akso3] \( (d_{re}, \uparrow) \), averaged over the last 10 epochs. As argued in Appendix D.3.2, patch mixing increases diversity compared to linear mixing in Figure D.5. As the probability \( p \) of patch mixing grows, so does diversity: from \( d_{re}(p = 0.0) \approx 0.78 \) (Linear-MixMo) to \( d_{re}(p = 0.5) \approx 0.85 \) (Cut-MixMo). In contrast, DE has \( d_{re} \approx 0.76 \) while MIMO has \( d_{re} \approx 0.77 \) on the same setup. Increasing \( p \) past 0.6 boosts diversity even more at the cost of subnetworks’ accuracies: this is due to underfitting and an increased test-train distribution gap. \( p \in [0.5, 0.6] \) is thus the best trade off.

D.4.4.2 Generalization to \( M \geq 2 \) subnetworks

We try to generalize MixMo to more than \( M = 2 \) subnetworks in Figure D.6. Cut-MixMo’s subnetworks perform at 82.3% when \( M = 2 \) vs. 79.5% when \( M = 3 \). In MIMO, it’s 79.8% vs. 77.7%. Because subnetworks do not share features, higher \( M \) degrades their results: only two can fit seamlessly. Ensemble Top1 overall decreases in spite of the additional predictions, as already noticed in MIMO [Hav+21]. This reflects MixMo’s strength in over-parametrized regimes, but also its limitations with fewer parameters.
when subnetworks underfit (recall previous Figure D.4). Facing similar findings, MIMO [Hav+21] introduced input repetition so that subnetworks share their features, at the cost of drastically reducing diversity.

D.4.4.3 Robustness to image corruptions

To measure MixMo effectiveness under distribution shifts, we now consider CIFAR-100-c [Hen+19a], a version CIFAR-100 where test images have been corrupted. We report WRN-28-10 results with and without AugMix [Hen+19b], a pixels data augmentation technique specialized on this task. Table D.2 shows that Cut-MixMo (b = 4) best complements AugMix and reaches 71.1% Top1.

<table>
<thead>
<tr>
<th>Approach</th>
<th>1 Net.</th>
<th>CutMix</th>
<th>Puzzle-Mix†</th>
<th>DE (2 Nets)</th>
<th>MIMO</th>
<th>Linear-MixMo</th>
<th>Cut-MixMo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top1 ↑</td>
<td>52.2</td>
<td>87.5</td>
<td>71.93</td>
<td>78.09</td>
<td>76.48</td>
<td>73.8</td>
<td>69.9</td>
</tr>
<tr>
<td>Top5 ↑</td>
<td>73.7</td>
<td>87.5</td>
<td>72.03</td>
<td>77.3</td>
<td>85.7</td>
<td>74.9</td>
<td>88.9</td>
</tr>
<tr>
<td>NLL ↓</td>
<td>2.20</td>
<td>1.38</td>
<td>2.13</td>
<td>1.96</td>
<td>1.34</td>
<td>2.27</td>
<td>1.24</td>
</tr>
</tbody>
</table>

Table D.2. – Robustness comparison on CIFAR-100-c.
D.4.5 Ensemble of MixMo

When combined with CutMix [Yun+19], Cut-MixMo previously set a new state of the art of 85.77% with \( N = 1 \) WRN-28-10. Since MixMo adds very little parameters (\( \approx +1\% \)), we can combine independently trained MixMo like in traditional functional ensembling. This ensembling of ensemble of subnetworks leads in practice to the averaging of \( M \times N = 2 \times N \) predictions. Final Table D.3 shows it further reaches 86.63% with \( N = 2 \) and even 86.81% with \( N = 3 \).

Table D.3. – Best results for WRN-28-10 on CIFAR-100 via Cut-MixMo + CutMix [Yun+19] + \( N \)-ensembling and \( b = 4 \). Previous Top1 sotas: 85.23 [Qin+20], 85.51 [Yan+20c], 85.74 [Zha+20b].

<table>
<thead>
<tr>
<th>( N )</th>
<th># params</th>
<th>Top1 ↑</th>
<th>Top5 ↑</th>
<th>NLLv ↓</th>
<th>Top1 ↑</th>
<th>Top5 ↑</th>
<th>NLLv ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>36.6M</td>
<td>85.77 ± 0.14</td>
<td>97.36 ± 0.03</td>
<td>0.524 ± 0.005</td>
<td>85.92</td>
<td>97.36</td>
<td>0.518</td>
</tr>
<tr>
<td>2</td>
<td>73.2M</td>
<td>86.63 ± 0.19</td>
<td>97.73 ± 0.05</td>
<td>0.479 ± 0.009</td>
<td>86.75</td>
<td>97.80</td>
<td>0.475</td>
</tr>
<tr>
<td>3</td>
<td>109.8M</td>
<td>86.81 ± 0.17</td>
<td>97.85 ± 0.04</td>
<td>0.464 ± 0.002</td>
<td>86.94</td>
<td>97.83</td>
<td>0.464</td>
</tr>
</tbody>
</table>

D.5 Conclusion

We introduce the MixMo framework that generalizes the multi-input multi-output paradigm. MixMo can be analyzed as either an ensembling method or a mixed samples data augmentation, while remaining complementary to works from both lines of research. MixMo improved the state of the art on CIFAR-100, CIFAR-100-c and Tiny ImageNet.

Limitations. Despite some relative success on medium-scale datasets, the MixMo framework has certain constraints. First, it is wasteful in its use of parameterization. Indeed, as we latter highlighted in MixShare [Sun+22], the learned subnetworks fail to share even generic features. Specifically, each channel or feature is almost exclusively used by one subnetwork; thus each additional subnetwork significantly reduces the effective size of the other subnetworks. This explains why MixMo requires (very) wide base models for larger datasets, and MixMo’s failure with \( M > 2 \) subnetworks in Appendix D.4.4.2. Though we tried to mitigate this failure in [Sun+22], results remained inconclusive. Overall, this limits MixMo’s applicability. Moreover, MixMo requires training from scratch; and thus cannot benefit from transfer learning from foundation models. This limitation renders MixMo less relevant at the end of this thesis, thus was relegated in Appendix for the sake of brevity.
E.1 Introduction

The success of DNNs in supervised learning [Kri+12] relies on the crucial assumption that the train and test data distributions are identical. In particular, the tendency of networks to rely on simple features [Val+19; Gei+20] is generally a desirable behavior reflecting Occam’s razor. However, in case of spurious features, this simplicity bias deteriorates performance when more complex features are needed [Ten18; Sha+20; Ker+21]. For example, in the recent fight against Covid-19, most of the DL methods developed to detect coronavirus from chest scans were shown useless for clinical use [DeG+21; Rob+21]: models actually exploited simple bias in the training datasets such as patients’ age or body position rather than truly analyzing medical pathologies.

To better generalize under correlation shift, ensembling strategies are useless, as verified empirically in Section 4.6.3.4: theoretically, they do not reduce the bias, as they all learn the same predictive mechanism based on the same spurious features. Indeed, when the train-test shifts occur in the posterior covariate distributions, additional information is required to differentiate between the relevant and the spurious features [Bla+11; Mua+13]. The standard invariance strategy [Arj+19] is to learn simultaneously from multiple training domains, across which we assume an underlying invariant causal mechanism [Pet+16]. To remove the domain-dependent explanations, different invariance criteria across those training domains have been proposed. [Gan+16; Sun+16] enforce similar feature distributions, others [Arj+19; Kru+21] force the classifier to be simultaneously optimal across all domains. Yet, despite the popularity of this research topic, none of these methods perform significantly better than the classical empirical risk minimization (ERM) when applied with controlled model selection and restricted hyperparameter search [Gul+21; Ye+22]. These failures motivate the need for new ideas.

To foster the emergence of a shared mechanism with consistent generalization properties, our intuition is that learning should progress consistently and similarly across domains. Besides, the learning procedure of DNNs is dictated by the distribution of the gradients with respect to the network weights [Yin+18; San+20], usually backpropagated in the network during gradient descent. Additionally, individual gradients are expressive representations of the input [For+19b; Cha+19]. Thus, we seek distributional invariance across domains in the gradient space: domain-level gradients should be similar, not only in average direction, but most importantly in statistics such as variance and disagreements.
Fishr considers the individual (per-sample) gradients of the loss in the network weights \( \theta \). Specifically, Fishr matches the domain-level gradient variances of the distributions across the two training domains: A \( \{g^A_i\}_{i=1}^{n_A} \) in orange) and B \( \{g^B_i\}_{i=1}^{n_B} \) in blue). We will show how this regularization during the learning of \( \theta \) improves the out-of-distribution generalization properties by aligning the domain-level loss landscapes at convergence.

In this chapter, we propose the Fishr regularization for generalization under correlation shift. As summarized in Figure E.1, Fishr enforces domain invariance in the space of the gradients of the loss; we match the domain-level variances of gradients, i.e., the second moment of the gradient distributions. In contrast, previous gradient-based works such as Fish [Shi+21] only match the domain-level gradients means, i.e., the first moment.

Our strategy is also motivated by the close relations between the gradient variance, the Fisher Information [Fis22] and the Hessian of the loss. This explains the name of our work, Fishr, using gradients as in Fish and related to the Fisher Matrix. Notably, we will study how Fishr forces the model to have similar domain-level Hessians. More broadly, Fishr aligns the domain-level loss landscapes locally around the final weights and promotes consistent explanations [Par+21].

To reduce the computational cost, we justify an approximation that tackles the gradients only in the classifier, easily implemented with BackPACK [Dan+20].

- We introduce Fishr, a scalable regularization that brings closer the domain-level gradient variances (Appendix E.3.1).
- We theoretically justify that Fishr matches domain-level risks and Hessians, and consequently, reduces inconsistencies across domains (Appendix E.3.2).
- Empirically, our experiments validate that Fishr can tackle correlation shifts on ColoredMNIST [Arj+19] (Appendix E.4).
This chapter has led to the publication of the following paper: Alexandre Ramé, Corentin Dancette, and Matthieu Cord. “Fishr: Invariant Gradient Variances for Out-of-Distribution Generalization”. In: ICML. 2022.

E.2 Related work

Our model is a deep neural network (DNN) $f(\cdot; \theta)$ (parametrized by $\theta$) made of a deep features extractor $\Phi$ (parametrized by $\phi$) on which we plug a dense linear classifier $w$ (parametrized by $\omega$). In training, we have access to different domains $\mathcal{E}$: for each domain $e \in \mathcal{E}$, the dataset $\mathcal{D}_e = \{(x_{ei}, y_{ei})\}_{i=1}^{n_e}$ contains $n_e$ i.i.d. (input, labels) samples drawn from a domain-dependent probability distribution. Combined together, the datasets $\{\mathcal{D}_e\}_{e \in \mathcal{E}}$ are of size $n = \sum_{e \in \mathcal{E}} n_e$. Our goal is to learn weights $\theta$ so that $f$ predicts well on a new test domain, unseen in training: $\theta$ should ideally capture an invariant mechanism across training domains. Following standard notations, $\| M \|_F$ denotes the Frobenius norm of matrix $M$; $\| v \|_2$ denotes the euclidean norm of vector $v$; $1$ is a column vector with all elements equal to $1$.

The standard ERM [Vap99] framework simply minimizes the average empirical risk over all training domains, i.e., $\frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} \mathcal{R}_e(\theta)$. To tackle the correlation shifts, many approaches try to exploit the domain information. Some works explore data augmentations to mix samples from different domains [Wan20b; Wu20b], some re-weight the training samples to favor underrepresented groups [Sag20a; Sag20b; Zha21a] and others include domain-dependent weights [Din17a; Man18a]. Yet, most recent works promote invariance via a regularization criterion and only differ by the choice of the statistics to be matched across training domains. They can be categorized into three groups: these methods enforce agreement either (i) in features (ii) in predictors or (iii) in gradients.

First, some approaches aim at extracting domain-invariant features and were extensively studied for unsupervised domain adaptation. The features are usually aligned with adversarial methods [Gan+16; Gon+16; Li+18a; Li+18b] or with kernel methods [Mua+13; Lon+14]. Yet, the simple covariance matching in CORAL [Sun+16] performs best on various tasks for OOD generalization [Gul21]. With $Z_{iej}$ the $j$-th dimension of the features extracted by $\Phi_{\phi}$ for the $i$-th example $x_{ei}$ of domain $e \in \mathcal{E} = \{A, B\}$, CORAL minimizes $\| \text{Cov}(Z_A) - \text{Cov}(Z_B) \|_F^2$ where $\text{Cov}(Z_e) = \frac{1}{n_e-1}(Z_e^\top Z_e - \frac{1}{n_e}(1^\top Z_e)(1^\top Z_e))$ is the feature covariance matrix. CORAL is more powerful than mere feature matching $\| \frac{1}{n_A} 1^\top Z_A - \frac{1}{n_B} 1^\top Z_B \|_2^2$ as in [Tze14]. Yet, [Joh19a] and [Zha19c] show that these approaches are insufficient to guarantee good generalization.

Motivated by arguments from causality [Pea09] and the idea that statistical dependencies are epiphenomena of an underlying structure, Invariant Risk Minimization (IRM) [Arj+19] explains that the predictors should be invariant [Pet+16; Roh+16a], i.e., simultaneously optimal across all domains. Yet, recent works point out pitfalls of IRM [Guo+21; Kam+21; Ahu+21], that does not provably work with non-linear data [Ros+21] and could not improve over ERM when hyperparameter selection is restricted [Koh+21; Gul21]. Among many suggested improvements [Cha+20; Idn20; Ten21; Ahm21], Risk Extrap-
olation (V-REx) [Kru+21] argues that training risks from different domains should be similar and thus penalizes $|R_A - R_B|^2$ when $\mathcal{E} = \{A, B\}$.

A third and most recent line of work promotes agreement between gradients with respect to network weights. Gradient agreements have been previously employed for multitasks [Du+18; Yu+20], continual [Lop+17], meta [Fin+17; Zha+20a] and reinforcement [Zha+19b] learning. In OOD generalization, [Koy+20; Par+21; Shi+21] try to find minima in the loss landscape that are shared across domains; they tackle the domain-level expected gradients:

$$g_e = \mathbb{E}_{(x_e, y_e) \sim D_e} \nabla_\theta \ell (f(x_e, \theta), y_e).$$  \hfill (E.1)

When $\mathcal{E} = \{A, B\}$, IGA [Koy+20] minimizes $\|g_A - g_B\|_2^2$; Fish [Shi+21] increases $g_A \cdot g_B$; AND-mask [Par+21] and others [Man+21b; Sha+21] update weights only when $g_A$ and $g_B$ point to the same direction. Along with the increased computation cost, the main limitation of previous gradient-based methods is the per-domain batch averaging of gradients: this removes more granular statistics, in particular the information from pairwise interactions between gradients from samples in a same domain. In opposition, our new regularization keeps extra information from individual gradients and matches across domains the domain-level gradient variances. In brief, Fishr is similar to the covariance-based CORAL [Sun+16] but in the gradient space rather than in the feature space.

### E.3 Fishr

#### E.3.1 Gradient variance matching

The individual gradient $g^i_e = \nabla_\theta \ell (f(x^i_e, \theta), y^i_e)$ is the first-order derivative for the $i$-th data example $(x^i_e, y^i_e)$ from domain $e \in \mathcal{E}$ with respect to the weights $\theta$. Previous methods have matched the gradient means $g_e = \frac{1}{n_e} \sum_{i=1}^{n_e} g^i_e$ for each domain $e \in \mathcal{E}$. These gradient means capture the average learning direction but can not capture gradient disagreements [San+20; Yin+18]. With $G_e = [g^i_e]_{i=1}^{n_e}$ of size $n_e \times |\theta|$, we compute the domain-level gradient variance vectors of size $|\theta|$:  

$$v_e = \text{Var}(G_e) = \frac{1}{n_e - 1} \sum_{i=1}^{n_e} (g^i_e - g_e)^2,$$  \hfill (E.2)

where the square indicates an element-wise product. To reduce the distribution shifts in $f(\cdot, \theta)$ across domains, we bring the domain-level gradient variances $\{v_e\}_{e \in \mathcal{E}}$ closer. Hence, our Fishr regularization is:

$$\mathcal{L}_\text{Fishr}(\theta) = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} \|v_e - v\|_2^2,$$  \hfill (E.3)

the square of the Euclidean distance between the gradient variance from the different domains $e \in \mathcal{E}$ and the mean gradient variance $v = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} v_e$. Balanced with a hyper-
parameter coefficient $\lambda > 0$, this Fishr penalty complements the original ERM objective, i.e., the empirical training risks:

$$L(\theta) = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} R_e(\theta) + \lambda L_{\text{Fishr}}(\theta). \quad (E.4)$$

**Remark E.1.** Gradients $g^e_i$ can be computed on all network weights $\theta$. Yet, to reduce the memory and training costs, they will often be computed only on a subset of $\theta$, e.g., only on the linear classifier’s weights $\omega$.

### E.3.2 Theoretical analysis

We theoretically motivate our Fishr regularization by leveraging the *domain inconsistency score* introduced in AND-mask [Par+21]. We first derive a generalized upper bound for this score. Then, we show that Fishr minimizes this upper bound by matching simultaneously *domain-level risks and Hessians*.

#### E.3.2.1 Inconsistency formalism

**Figure E.2.** *Loss landscapes around inconsistent weights* $\theta^*$ at convergence. $N_{A,\theta^*}^{0.2}$ contains weights $\theta$ for which $R_A(\theta)$ is low ($\leq 0.2$) but $R_B(\theta)$ is high ($\geq 0.9$). This inconsistency is due to conflicting domain-level loss landscapes, specifically gaps between domain-level risks and curvatures at $\theta^*$. This is visible in the disagreements across the variances of gradients $\{g^A_A\}_{i=1}^{n_A}$ and $\{g^B_B\}_{i=1}^{n_B}$. 

- **Loss landscape for domain A**
- **Loss landscape for domain B**
[Par+21] argues that “patchwork solutions sewing together different strategies” for different domains may not generalize well: good weights should be optimal on all domains and “hard to vary” [Deu11]. They formalize this insight with an inconsistency score:

\[
\mathcal{I}^\epsilon(\theta^*) = \max_{(A, B) \in \mathcal{E}^2} \max_{\theta \in N_{A, \theta^*}} |\mathcal{R}_B(\theta) - \mathcal{R}_A(\theta^*)|,
\]

where \( \theta \in N_{A, \theta^*} \) if there exists a path in the weights space between \( \theta \) and \( \theta^* \) where the risk \( \mathcal{R}_A \) remains in an \( \epsilon > 0 \) interval around \( \mathcal{R}_A(\theta^*) \). \( \mathcal{I} \) increases with conflicting geometries in the loss landscapes around \( \theta^* \) as in Figure E.2: i.e., when another “close” solution \( \theta \) is equivalent to the current solution \( \theta^* \) in a domain \( A \) but yields different risks in \( B \). For \( e \in \mathcal{E} \), the second-order Taylor expansion of \( \mathcal{R}_e \) around \( \theta^* = 0 \) (with a change of variable) gives:

\[
\mathcal{R}_e(\theta) = \mathcal{R}_e(\theta^*) + \theta^\top \nabla_\theta \mathcal{R}_e(\theta^*) + \frac{1}{2} \theta^\top H_e \theta + O(||\theta||^2),
\]

where the Hessian \( H_e = \nabla^2_{\theta} \mathcal{R}_e(\theta^*) \) approximates the local curvature of the loss landscape. Moreover, we assume simultaneous convergence, i.e., \( \theta^* \) is a local minima across all domains: \( \nabla_\theta \mathcal{R}_e(\theta^*) = 0 \). Thus, locally around \( \theta^* \):

\[
\max_{\theta \in N_{A, \theta^*}} |\mathcal{R}_B(\theta) - \mathcal{R}_A(\theta^*)| \approx \max_{|\mathcal{R}_A(\theta) - \mathcal{R}_A(\theta^*)| \leq \epsilon} |\mathcal{R}_B(\theta) - \mathcal{R}_A(\theta^*)| \\
\approx \max_{\frac{1}{2}||\theta^\top H_A \theta|| \leq \epsilon} \left| \mathcal{R}_B(\theta^*) + \frac{1}{2} \theta^\top H_B \theta - \mathcal{R}_A(\theta^*) \right| \\
\lesssim |\mathcal{R}_B(\theta^*) - \mathcal{R}_A(\theta^*)| + \max_{\frac{1}{2}||\theta^\top H_A \theta|| \leq \epsilon} \frac{1}{2} \left| \theta^\top H_B \theta \right|,
\]

where we deduced the last line from the triangle inequality. In [Ram+22a] we formally demonstrate the following equality when the per-domain risks are assumed quadratic.

**Proposition E.1.** Under the quadratic bowl assumption (i.e., when per-domain risks are assumed quadratic with positive definite Hessians) and for sufficiently small \( \epsilon \), then:

\[
\mathcal{I}^\epsilon(\theta^*) = \max_{(A, B) \in \mathcal{E}^2} \left( \mathcal{R}_B(\theta^*) - \mathcal{R}_A(\theta^*) + \max_{\frac{1}{2}||\theta^\top H_A \theta|| \leq \epsilon} \frac{1}{2} \theta^\top H_B \theta \right).
\]

The Hessian being positive definite is a standard hypothesis, notably used in [Par+21], that is empirically reasonable [Sag+18]: “in only very few steps [...] large negative eigenvalues disappear” [Gho+19]. We now analyze the two terms from this bound.

The first term in the RHS of **Proposition E.1** is the difference between domain-level risks, whose square is the criterion minimized in V-REx [Kru+21]. We will show that Fishr forces this term to be small in Appendix E.3.2.2.
For the second term, we follow the diagonal approximation of the Hessians from \[\text{Par+21}\]. In that case, \(H_e = \text{diag}(\lambda^e_1, \ldots, \lambda^e_h)\) with \(\forall i \in \{1, \ldots, h\}, \lambda^e_i > 0\). Then:

\[
\max_{1/2 \theta^\top H_A \theta \leq \epsilon} \frac{1}{2} \theta^\top H_B \theta = \max_{\|\theta\|^2 \leq \epsilon} \sum_i \frac{\theta^2_i \lambda^B_i / \lambda^A_i}{\epsilon \times \max_i \lambda^B_i / \lambda^A_i}.
\]

(E.8)

This is large when exists \(i\) such that \(\lambda^A_i\) is small but \(\lambda^B_i\) is large: indeed, a small weight perturbation in the direction of the associated eigenvector would change the loss slightly in the domain \(A\) but drastically in domain \(B\). Thus, this second term decreases when \(H_A\) and \(H_B\) have similar eigenvalues. This result holds when Hessians are co-diagonalizable.

In conclusion, this explains why forcing \(H_A = H_B\) reduces inconsistencies in the loss landscape and thus improves generalization. AND-mask matches Hessians by zeroing out gradients with inconsistent directions across domains; however, this masking strategy introduces dead zones \[\text{Sha+21}\] in weights where the model could get stuck, ignores gradient magnitudes and empirically performs poorly with real datasets from DomainBed.

As shown in Appendix E.3.2.3, Fishr proposes a new method to align domain-level Hessians leveraging the close relations between the gradient variance, the Fisher Information and the Hessian.

### E.3.2.2 Fishr matches the domain-level risks

Gradients take into account the label \(Y\), which appears as an argument for the loss \(\ell\). Hence, gradient-based approaches are ‘label-aware’ by design. In contrast, feature-based methods were shown to fail in case of label shifts, because they do not consider \(Y\) \[\text{Joh+19; Zha+19c}\]. The fact that the label and the loss appear in the formula of the gradients has another important consequence: matching gradient distributions also matches training risks, as motivated in V-REx \[\text{Kru+21}\]. We confirm this insight in Table E.2: matching gradient variances with Fishr induces \(|R_A - R_B|^2 \to 0\) when \(\mathcal{E} = \{A, B\}\). **Intuitively**, gradient amplitudes are directly weighted by the loss values: multiplying the loss by a constant will also multiply the gradients by the same constant. Thus roughly, if the domain-level empirical training risks are different, then the domain-level gradient norms should also differ. **Theoretically**, we prove in the paper that Fishr regularization component with reference to the classification bias is exactly the difference between domain-level mean squared errors. We recover the objective from V-REx \[\text{Kru+21}\], with a different loss (squared error instead of negative log likelihood).

### E.3.2.3 Fishr matches the domain-level Hessians

The Hessian matrix \(H = \sum_{i=1}^n \nabla^2 \ell(f_\theta(x^i), y^i)\) is of key importance in deep learning. Yet, \(H\) cannot be computed efficiently in general. Recent methods \[\text{Izm+18; Par+21; For+21}\] tackled the Hessian indirectly by modifying the learning procedure. In contrast, we use the fact that the diagonal of \(H\) is approximated by the gradient variance \(\text{Var}(G)\); this is confirmed in Table E.1. This result is derived below from 3 individual and standard approximation steps.
Appendix S C

Thus, the gradient covariance matrix \( C \) point:

Table E.1. – Cosine similarity between Hessian diagonals and gradient variances, \( \cos(\text{Diag}(H_e), \text{Var}(G_e)) \), for an ERM at convergence on ColoredMNIST with the two training domains \( e \in \{90\%, 80\%\} \).

\[
\begin{array}{cc}
{e = 90\%} & {e = 80\%} \\
\hline
\text{On classifier weights } \omega & 0.999980 & 0.999905 \\
\text{On all network weights } \theta & 0.9071040 & 0.9062264 \\
\end{array}
\]

The Hessian and the “true” Fisher information matrix (FIM). The “true” FIM \( F = \sum_{i=1}^{n} E_{y \sim P_{\theta}(\cdot|x)} \left[ \nabla_{\theta} \log p_{\theta}(y|x^{i}) \nabla_{\theta} \log p_{\theta}(y^{i}|x^{i}) \right] \) [Fis22; CR45] approximates the Hessian \( H \) with theoretically probably bounded errors under mild assumptions [Sch02].

The “true” FIM and the “empirical” FIM. Yet, \( F \) remains costly as it demands one backpropagation per class. That’s why most empirical works (e.g., in compression [Fra+21; Liu+21] and optimization [Dan+21]) approximate the “true” FIM \( F \) with the “empirical” FIM \( \tilde{F} = G_e \top G_e = \sum_{i=1}^{n} \nabla_{\theta} \log p_{\theta}(y^{i}|x^{i}) \nabla_{\theta} \log p_{\theta}(y^{i}|x^{i}) \top \) [Mar14] where \( p_{\theta}(\cdot|x) \) is the density predicted by \( f_{\theta} \) on input \( x \). While \( F \) uses the model distribution \( P_{\theta}(\cdot|X) \), \( \tilde{F} \) uses the data distribution \( P(Y|X) \). Despite this key difference, \( \tilde{F} \) and \( F \) were shown to share the same structure and to be similar up to a scalar factor [Tho+20]. They also have analogous properties: \( \text{Tr}(\tilde{F}) \approx \text{Tr}(F) \). This was discussed in [Li+20c] and further highlighted even at early stages of training (before overfitting) in the Fig. 1 and the Appendix S3 of [Sin+20a].

The “empirical” FIM and the gradient covariance. Critically, \( \tilde{F} \) is nothing else than the unnormalized uncentered covariance matrix when \( \ell \) is the negative log-likelihood. Thus, the gradient covariance matrix \( C = \frac{1}{n-1} \left( G \top G - \frac{1}{n} (1 \top G) (1 \top G) \right) \) of size \( |\theta| \times |\theta| \) and \( \tilde{F} \) are equivalent (up to the multiplicative constant \( n \)) at any first-order stationary point: \( C \cong \tilde{F} \). Overall, this suggests that \( C \) and \( H \) are closely related [Jas+18].

Table E.2. – Invariance analysis at convergence on ColoredMNIST across the two training domains \( E = \{90\%, 80\%\} \). Compared to ERM, Fishr matches the gradient variance (Diag(\( G_{90\%} \)) \approx \text{Diag}(\text{Var}(G_{80\%})) in all network weights \( \theta \). Most importantly, this enforces invariance in domain-level risks (\( R_{90\%} \approx R_{80\%} \)) and in domain-level Hessians (Diag(\( H_{90\%} \)) \approx \text{Diag}(\text{Var}(H_{80\%}))). The gradient variance, computable efficiently with a unique backpropagation, serves as a proxy for the Hessian. Details and more experiments in Figure E.3.

<table>
<thead>
<tr>
<th></th>
<th>ERM</th>
<th>Fishr</th>
</tr>
</thead>
<tbody>
<tr>
<td>( |\text{Var}(G_{90%}) - \text{Var}(G_{80%})|_{F}^{2} )</td>
<td>1.6</td>
<td>4.1 \times 10^{-5}</td>
</tr>
<tr>
<td>(</td>
<td>R_{90%} - R_{80%}</td>
<td>^{2} )</td>
</tr>
<tr>
<td>( |\text{Diag}(H_{90%}) - \text{Diag}(H_{80%})|_{F}^{2} )</td>
<td>2.9 \times 10^{-1}</td>
<td>2.7 \times 10^{-4}</td>
</tr>
</tbody>
</table>

Consequences for Fishr. Critically, Fishr considers the gradient variance \( \text{Var}(G) \), i.e., the diagonal components of \( C \). In our multi-domain framework, we define the domain-level matrices with the subscript \( e \). Table E.2 empirically confirms that match-
E.4 Experiments

We validate Fishr effectiveness on ColoredMNIST [Arj+19], where the task is to predict whether the digit is below or above 5. Moreover, the labels are flipped with 25% probability. Critically, the digits’ colors spuriously correlate with the labels: the correlation strength varies across the two training domains $\mathcal{E} = \{90\%, 80\%\}$. To test whether the model has learned to ignore the color, this correlation is reversed at test time. In brief, a biased model that only considers the color would have 10% test accuracy whereas an oracle model that perfectly predicts the shape would have 75%. As previously done in V-REx [Kru+21], we strictly follow the IRM implementation and just replace the IRM penalty by our Fishr penalty. This means that we use the exact same MLP and hyperparameters, notably the
same two-stage scheduling for the regularization strength $\lambda$, that is low until epoch 190 and then jumps to a large value, which was optimized via a grid-search for IRM.

Table E.3 reports the accuracy averaged over 10 runs with standard deviation. ENS averages the predictions of the 10 ERM runs. Fishr$_\theta$ (i.e., applying Fishr on all weights $\theta$) obtains the best trade-off between train and test accuracies; notably in test, it reaches 71.2\%, or 70.2\% when digits are grayscale. Moreover, computing the gradients only in the classifier $w_\omega$ performs almost as well (69.5\% in test for Fishr$_\omega$) while reducing drastically the computational cost. Finally, Fishr$_\phi$ only in the features extractor $\phi$ works best in test, though it has lower train accuracy. This last experiment shows that we can reduce domain shifts without explicitly forcing the predictors to be simultaneously optimal. These results highlight the effectiveness of gradient variance matching, even with standard hyperparameters, at different layers of the network.

![Figure E.3 - ColoredMNIST dynamics.](image-url)

Figure E.3 – ColoredMNIST dynamics. At epoch 190, $\lambda$ strongly steps up: then, the Fishr$_\theta$ regularization matches the domain-level gradient variances (red) across domains $E = \{90\%,80\%\}$, and consequently, the training empirical risks (dotted pink) and Hessians (purple). This reduces train accuracy (orange) but increases test accuracy (blue) as the network learns to predict the digit’s shape.

Moreover, the training dynamics in Figure E.3 show that the domain-level empirical risks get closer once the Fishr$_\theta$ gradient variance matching loss is activated after step 190 ($|\mathcal{R}_{90\%} - \mathcal{R}_{80\%}| \to 0$), even though predicting accurately on the domain 90\% is easier than...
on the domain 80%. This confirms insights from Appendix E.3.2.2. Similarly, we observe that Fishr matches Hessians across the two training domains. Overall, Fishr regularization reduces train accuracy, but considerably increases test accuracy.

**Experiments on DomainBed.** In the paper [Ram+22a], we have also conducted extensive experiments on the DomainBed benchmark [Gul+21]. When Fishr was published, it was the sota method; moreover, according to the latest review paper [Yu+23], Fishr remains the best method based on invariance. However, all the strategies based on ensembling were shown more efficient on these real-world datasets under which diversity shifts dominate. Therefore, for the sake of brevity, we do not report these DomainBed experiments in this thesis, but refer the reader to the published version of the paper.

**E.5 Conclusion**

In this chapter, we addressed a limitation of ensembling strategies; their inability to tackle correlation shifts. To this end, we leverage the invariance paradigm, and specifically derive a new and simple regularization, Fishr, matching the gradient variances across domains as a proxy for matching domain-level risks and Hessians. We prove that invariance can tackle correlation shift. We hope to pave the way towards strategies that simultaneously tackle diversity and correlation shifts.
LIST OF FIGURES

Figure 1.1  A glimpse into the transformative and successful applications of DL. In Figure 1.1(a), AlphaGo [Sil+16] surpasses human performance in the strategic board game of Go. In Figure 1.1(b), Stable Diffusion [Rom+22] is prompted to generate “a photograph of an astronaut riding a horse”, illustrating how AIs can imitate human creativity. In Figure 1.1(c), sub-meter resolution canopy height maps are generated from aerial and GEDI lidar data [Tol+23], offering climate scientists a powerful tool to understand deforestation. In Figure 1.1(d), the now famous ChatGPT [Ope23] can natively discuss with humans and augment their intelligence.  

Figure 2.1  The fundamental concepts in deep learning. The first row represents the names and notations used along this thesis, the second row illustrates the different concepts, while the third row depicts the corresponding mathematical objects. Specifically, an image \( x \) represented as a RGB matrix is given as input to a black-box featurizer \( \Phi \), that extracts an embedding vector \( z \), representing in a structured way the different information in the image. This embedding is then fed to a (usually linear) classifier \( w \), whose goal is to separate the different classes \( Y \); specifically, \( w \) predicts a probability distribution \( \hat{y} \) that should be close to the true label \( y \).  

Figure 2.2  Visualization of the two types of distribution shifts. Thanks to the bias-variance decomposition of the error Equation (BV), we will show that they have drastically different consequences on performances: while diversity shift increases the variance term, correlation shift increases the bias term. Image from [Ye+22].  

Figure 2.3  Mean and variance of the predictions for Gaussian processes. Image from [Pér+13]. The \( x \)-axis represents the input and the \( y \)-axis the targeted output. Intuitively, variance (grey area) across different predictions (blue lines) grows away from training samples (green crosses).  

Figure 2.4  The traditional functional ensembling, averaging the predictions for \( M = 2 \) models.  

Figure 3.2  Venn information diagram. DICE minimizes conditional redundancy (green vertical stripes) with no overlap with relevancy (red diagonal stripes).
Learning strategy overview. Blue arrows represent training criteria: (i) classification with conditional entropy bottleneck applied separately on each member, and (ii) adversarial training to delete irrelevant redundant information between members and increase diversity. $X$ and $X'$ belong to the same $Y$ for conditional redundancy minimization.

Performances as a function of the number of members. DICE better leverages ensemble size on CIFAR-100 for ResNet-32. Without weights sharing, 5 networks trained with DICE match 7 networks trained independently. With low-level weights sharing, 4 branches trained with DICE match 7 traditional branches.

Ensemble and individual accuracies as a function of diversity, highlighting the different trade-offs for different strategies. DICE (r. CEBR) is learned with different $\delta_{cr}$ (r. $\delta_r$).

Impact of the diversity coefficient $\delta_{cr}$ in DICE on the training dynamics on validation: CR is negatively correlated with diversity.

Train Hessian trace (↓). Test OOD accuracy (↑). Test diversity in ratio-error [Aks03] (↑). The different fine-tuning strategies discussed in this chapter: vanilla fine-tuning [Oqu+14], moving average (MA) [Izm+18] and variants [Wor+22b], and our novel DiWA [Ram+22b]. They all start from a pre-trained foundation model, before fine-tuning on the target task (thick solid arrows). The fine-tuned weights are used as is, or are averaged (dashed arrows into a final model).

Each dot displays the OOD accuracy (↑) of WA vs. ENS when combining $M$ models. Each dot displays the accuracy (↑) gain of WA over its members vs. the prediction diversity [Aks03] (↑) for $M$ models. The slopes relating diversity (in prediction [Aks03] or in features [Kor+19]) to accuracy gain, increases with $M$. Frequencies of diversities [Aks03] across 2 weights obtained along a single run or from different runs. WA accuracy as $M$ increases, when the $M$ weights are obtained along a single run or from different runs. Each dot displays the accuracy gain of WA over its members vs. diversity for $2 \leq M < 10$ models.
The different fine-tuning strategies discussed in this chapter: vanilla fine-tuning [Oqu+14], moving average (MA) [Izm+18] and variants [Wor+22b], DiWA [Ram+22b] introduced in Chapter 4 and the similar model soups [Wor+22a], inter-training [Pha+18], fusing [Cho+22b] and our novel model ratatouille. They start with a pre-trained foundation model. Some strategies fine-tune the pre-trained model on auxiliary tasks (thin solid arrows \(\rightarrow\)): 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 \(\rightarrow\)). Finally, the weights fine-tuned on the target task are used as is, or are averaged (dashed arrows \(\leftarrow\)) into a final model. Ratatouille (i) enables compute parallelism, (ii) maximizes the amount of diversity in models’ predictions, (iii) achieves sota performance in DomainBed [Gul+21], the standard computer vision benchmark for OOD generalization and (iv) does not incur any inference or training overhead compared to a traditional hyperparameter search.

Illustrations of (a) different linear mode connectivity (LMC) conditions, and (b) model ratatouille. In subplot (a), we illustrate Observation 4.1, about LMC between two checkpoints along the same target fine-tuning; Observation 4.2, about LMC between two target fine-tunings; Hypothesis 5.1, about LMC between two auxiliary fine-tunings; and Hypothesis 5.2, about LMC between two target fine-tunings initialized from auxiliary weights satisfying Hypothesis 5.1. In subplot (b), we offer a diagram of our proposed ratatouille strategy, where we (i) fine-tune a pre-trained model on auxiliary tasks, (ii) plug a linear probe classifier 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.

Explorations on Q-diversity [Kun+03] 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 \(\mu\) are inter-trained on DomainNet. The accuracy of the weight average is maximized when \(\mu \approx 0.5\).
Figure 5.4: Figures 5.4(a) to 5.4(e) validate Hypothesis 5.1 by plotting $\lambda \rightarrow \text{acc}_T\left( (w_{lp}, (1 - \lambda) \cdot \phi_{aux}^1 + \lambda \cdot \phi_{aux}^2) \right)$, where $w_{lp}$ is the linear probe of $\phi_{IM}^\text{pt}$ and $\phi_{aux}^1$ and $\phi_{aux}^2$ are fine-tuned on the two auxiliary datasets in the legend “Dataset1 to Dataset2”. Figures 5.4(f) to 5.4(j) support Hypothesis 5.2 by plotting $\lambda \rightarrow \text{acc}_T((1 - \lambda) \cdot \theta_1 + \lambda \cdot \theta_2)$ where $\theta_1$ and $\theta_2$ are fine-tuned on the target task starting respectively from $(w_{lp}, \phi_{aux}^1)$ and $(w_{lp}, \phi_{aux}^2)$. We encounter two exceptions to Hypothesis 5.2 (Figures 5.4(i) and 5.4(j)), due to the fact that neither the auxiliary (RxRx) nor the target task (Terralncognita and Cameyon) bear enough similarity with the pre-training task (ImageNet).

Figure 5.5: 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.

Figure 5.6: 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 5.2 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 $\mu$ 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 [Kun+03] 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.
Figure 6.1  

**Figure 6.1(a)** details the different steps in rewarded soup. After unsupervised pre-training and supervised fine-tuning, we launch \( N \) independent RL fine-tunings on the proxy rewards \( \{R_i\}_{i=1}^N \). Then we combine the trained networks by interpolation in the weight space. The final weights are adapted at test time by selecting the coefficient \( \lambda \). **Figure 6.1(b)** shows our results (extended in **Figure 6.2(a)**) with LLaMA-7b [Tou+23a] instruct fine-tuned on Alpaca [Tao+23], when RL fine-tuning for news summarization [Sti+20b] with \( N = 2 \) reward models assessing diverse preferences: \( R_1 \) rewards completeness while \( R_2 \) rewards faithfulness. With only two trainings (\( R_1 \) and \( R_2 \) rewarded on **Figure 6.1(b)**), the \( \lambda \)-interpolation \( (0 \leq \lambda \leq 1) \) reveals the green front of Pareto-optimal solutions, i.e., that cannot be improved for one reward without sacrificing the other. RS matches the costly yellow front of MORL [Bar+08; Li+20b] requiring multiple trainings on different linear weightings \( (1 - \mu) \times R_1 + \mu \times R_2 \) with \( 0 \leq \mu \leq 1 \).

Figure 6.2  

**Figure 6.2** RLHF results in NLP with LLaMA-7b [Tou+23a] and reward models \( R_i \) from HuggingFace [Wol+20]. The \( x \)-axis represents the score for the first reward while the \( y \)-axis represents the score for the second reward. The blue line reports checkpoints’ results along the training trajectory of \( \theta_1 \) rewarding \( R_1 \), the red line \( \theta_2 \) rewarding \( R_2 \), and the purple line the MORL rewarding \( \frac{R_1 + R_2}{2} \). Our rewarded soup (RS) linearly interpolates between the weights \( \theta_1 \) and \( \theta_2 \); sliding the interpolation coefficient \( \lambda \) from 0 to 1 reveals the green solid front of rewarded soups solutions. In **Figures 6.2(a)** and 6.2(b), we additionally show the multiple MORL runs rewarding \( (1 - \mu) \times R_1 + \mu \times R_2 \) with preferences \( 0 \leq \mu \leq 1 \); the thin lines then represent the performances along those fine-tunings at different steps. It reveals a similar yellow front, yet more costly. In **Figure 6.2(f)**, we uniformly \( (\lambda_i = \frac{1}{4}) \) average the weights fine-tuned for the assistant task on \( N = 4 \) reward models.

Figure 6.3  

Results in image captioning on COCO [Lin+14]. As rewards \( R_1 \) (blue stars every epoch) and \( R_2 \) (red stars), we consider standard statistical metrics: BLEU1 (1-gram overlap), BLEU4 (4-grams overlap), ROUGE, METEOR and CIDEr. **Figure 6.3(a)** include the MORL training trajectories optimizing \( (1 - \mu) \times BLEU1 + \mu \times ROUGE \), uncovering a yellow front similar to RS’s green front. In **Figure 6.3(c)**, RS uniformly averages the 5 weights (one for each reward), resulting in the largest area and the best trade-off between the 5 rewards.
Figure 6.4  Refined results in captioning with \( R_1 = \text{BLEU}_1 \) and \( R_2 = \text{ROUGE} \). Figure 6.4(a) empirically validates Lemma 6.2 by reporting results of RS (for varying \( \lambda \)) and of MORL (for varying \( \mu \)) for varying user’s preference \( \hat{\mu} \). In Figure 6.4(b), all rewards are used for evaluation as a function of the interpolating coefficient. In Figure 6.4(c), we report the front of the costly functional ensembling [Han+90; Lak+17] of predictions (rather than the weight interpolation).

Figure 6.5  Figure 6.5(a) reports our RLHF experiments on text-to-image generation with diffusion models. From the pre-trained initialization, we learn \( \theta_{ava} \) and \( \theta_{cafe} \) by optimizing the two reward models \( ava \) and \( cafe \). Interpolation between them reveals the green Pareto-optimal front, above the yellow MORL front. Figures 6.5(b) and 6.5(c) report our results in visual grounding on RefCOCO+ [Yu+16], where we optimize to predict boxes with IoU>0.5 w.r.t. the ground-truth, for objects of either small, medium or large size.

Figure 6.6  VQA results.

Figure 6.7  Locomotion results.

Figure C.1  Illustration of the bound given by Lemma C.2 under Assumption C.1. For simplicity, we showcase the case where \( R_1(\theta_1) = R_2(\theta_2) = 1 \), \( R_1(\theta_2) = R_2(\theta_1) = 0 \), thus \( \Delta_1 = \Delta_2 = 1 \). In green, we plot the rewards obtained with rewarded soups for the optimal \( \hat{\lambda}, i.e., R_{\hat{\mu}}((1 - \hat{\lambda}) \cdot \theta_1 + \hat{\lambda} \cdot \theta_2) \), whose value is independent of \( M \) in this case. In blues, we plot the maximum value of \( R_{\hat{\mu}} \) given by Equation (C.10) in Lemma C.2, for \( M = 2 \) and \( M = 10 \). For reference, we also plot the values for the lower bound in the LMC Hypothesis 6.1, i.e., equal to \((1 - \hat{\mu})(1 - \hat{\lambda})R_1(\theta_1) + \hat{\mu}\hat{\lambda}R_2(\theta_2) \). As RS outperforms this lower bound, it validates Hypothesis 6.1 in this case.

Figure D.1  MixMo framework. We embed \( M = 2 \) inputs into a shared space with convolutional layers \( (c_1, c_2) \), mix them, pass the embedding through further layers and output 2 predictions via dense layers \( (d_1, d_2) \). The key point of our MixMo is the mixing block. Mixing with patches performs better than basic summing: 85.40% vs. 83.06% (MIMO [Hav+21]) on CIFAR-100 with WRN-28-10.

Figure D.2  Main MixMo results on CIFAR-100 with WRN-28-w. Our Cut-MixMo variant (patch mixing and \( M = 2 \)) surpasses CutMix and deep ensembles (with half the parameters) by leveraging over-parameterization in wide networks.

Figure D.3  Cut-MixMo training. We sample a mixing mask given \( \kappa \), and balance the losses with \( w_r(\kappa) \) from Eq. D.4.
Figure D.4 Parameters efficiency (metrics/#params). CIFAR-100 with WRN-28-w, b=4. Comparisons between (a) ensemble and some of their (b) individual counterparts. .......................... 159
Figure D.5 Diversity/accuracy as function of p with r=3. .................. 160
Figure D.6 Ensemble/individual accuracies for $M \geq 2$. .................. 160

Figure E.1 Fishr considers the individual (per-sample) gradients of the loss in the network weights $\theta$. Specifically, Fishr matches the domain-level gradient variances of the distributions across the two training domains: $A (\{g_A^i\}_{i=1}^n$ in orange) and $B (\{g_B^i\}_{i=1}^n$ in blue). We will show how this regularization during the learning of $\theta$ improves the out-of-distribution generalization properties by aligning the domain-level loss landscapes at convergence. .................. 164

Figure E.2 **Loss landscapes around inconsistent weights** $\theta^*$ at convergence. $N_{A,B}^{0.2}$ contains weights $\theta$ for which $R_A(\theta)$ is low ($\leq 0.2$) but $R_B(\theta)$ is high ($\geq 0.9$). This inconsistency is due to conflicting domain-level loss landscapes, specifically gaps between domain-level risks and curvatures at $\theta^*$. This is visible in the disagreements across the variances of gradients $\{g_A^i\}_{i=1}^n$ and $\{g_B^i\}_{i=1}^n$. .................. 167

Figure E.3 ColoredMNIST dynamics. At epoch 190, $\lambda$ strongly steps up: then, the Fishr$_{\theta}$ regularization matches the domain-level gradient variances (red) across domains $E = \{90\%, 80\%\}$, and consequently, the training empirical risks (dotted pink) and Hessians (purple). This reduces train accuracy (orange) but increases test accuracy (blue) as the network learns to predict the digit’s shape. .................. 172
**LIST OF TABLES**

Table 1.1  **Summary of the PhD publications.**  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1  Hyperparameters, their default values and distributions for random search.  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1  CIFAR-100 ensemble classification accuracy (Top-1, %).  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2  CIFAR-10 ensemble classification accuracy (Top-1, %).  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.3  OOD accuracy (%,$\uparrow$) on DomainBed. “Art” is the OOD domain for both datasets.  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1  **Accuracies (%,$\uparrow$) on DomainBed [Gul+21]** benchmark evaluating OOD generalization. The classifiers are initialized randomly or with linear probing (LP) [Kum+22]. The symbol “$\ast$” indicates inference overhead in functional ensembling. The symbol “$\dagger$” indicates the averaging of all weights across 3 data splits.  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2  **DiWA vs. ENS** on domain “Art” from OfficeHome when varying initialization and hyperparameter ranges. Best accuracy (%,$\uparrow$) on each setting is in **bold**.  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3  **Accuracy (%,$\uparrow$) on ColoredMNIST.** WA does not improve performance under correlation shift. Random initialization of the classifier. Test-domain model selection.  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4  **Accuracy (%,$\uparrow$) on OfficeHome** domain “Art” with various objectives.  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1  **Accuracies (%,$\uparrow$) on DomainBed [Gul+21]** benchmark evaluating OOD generalization. Ratatouille sets a new sota by leveraging diversity in auxiliary tasks. The symbol “$\ast$” indicates inference overhead in functional ensembling. The symbol “$\dagger$” indicates the averaging of all weights across 3 data splits. The scores for DiWA are those from Table 4.1 with LP initialization.  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table D.1  Main MixMo results: WRN-28-10 on CIFAR.  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table D.2  Robustness comparison on CIFAR-100-c.  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table D.3  Best results for WRN-28-10 on CIFAR-100 via Cut-MixMo + CutMix [Yun+19] + $N$-ensembling and $b = 4$. Previous Top1 sotas: 85.23 [Qin+20], 85.51 [Yan+20c], 85.74 [Zha+20b].  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table E.1  **Cosine similarity between Hessian diagonals and gradient variances**, $\cos(\text{Diag}(H_g), \text{Var}(G_e))$, for an ERM at convergence on ColoredMNIST with the two training domains $e \in \{90\%, 80\%\}$.  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

183
Table E.2  **Invariance analysis** at convergence on ColoredMNIST across the two training domains $\mathcal{E} = \{90\%, 80\\%\}$. Compared to ERM, Fishr matches the gradient variance ($\text{Diag}(C_{90\%}) \approx \text{Diag}(C_{80\%})$) in all network weights $\theta$. Most importantly, this enforces invariance in domain-level risks ($R_{90\%} \approx R_{80\%}$) and in domain-level Hessians ($\text{Diag}(H_{90\%}) \approx \text{Diag}(H_{80\%})$). The gradient variance, computable efficiently with a unique backpropagation, serves as a proxy for the Hessian. Details and more experiments in Figure E.3. 170

Table E.3  ColoredMNIST results. All methods use hyperparameters optimized for IRM. 172