M-HOF-Opt: Multi-Objective Hierarchical Output Feedback Optimization via Multiplier Induced Loss Landscape Scheduling

Xudong Sun[1] **Nutan Chen**[2] **Alexej Gossmann**[5] **Matteo Wohlrapp**[4]
Yu Xing[3] **Emilio Dorigatti**[1] **Carla Feistner**[6] **Felix Drost**[1]
Daniele Scarcella[1] **Lisa Helen Beer**[4] **Carsten Marr**[1]
[1]Institute of AI for Health, Computational Health Center, Helmholtz Munich, Germany
[2]Machine Learning Research Lab, Volkswagen Group, Munich, Germany
[3]Decision and Control, EECS, KTH Royal Institute of Technology, Stockholm, Sweden
[4]Technical University Munich, Germany
[5]U.S. FDA/CDRH, Silver Spring, MD, USA (work conducted here, no longer affiliated)
[6] Applied Geology, GeoZentrum Nordbayern, Friedrich-Alexander-Universität, Erlangen-Nürnberg, Germany

Abstract

A probabilistic graphical model is proposed, modeling the joint model parameter and multiplier evolution, with a hypervolume based likelihood, promoting multi-objective descent in structural risk minimization. We address multi-objective model parameter optimization via a surrogate single objective penalty loss with time-varying multipliers, equivalent to online scheduling of loss landscape. The multi-objective descent goal is dispatched hierarchically into a series of constraint optimization sub-problems with shrinking bounds according to Pareto dominance. The bound serves as setpoint for the low-level multiplier controller to schedule loss landscapes via output feedback of each loss term. Our method forms closed loop of model parameter dynamic, circumvents excessive memory requirements and extra computational burden of existing multi-objective deep learning methods, and is robust against controller hyperparameter variation, demonstrated on domain generalization tasks.

1 INTRODUCTION

This work initializes efforts to utilize feedback mechanism, constraint optimization, hierarchical and optimal control into deep learning by treating neural network training as a dynamic process. We tackle two fold challenges in multi-objective deep learning and automatic multiple-multiplier adaptation in structural risk minimization: In many deep learning fields including domain generalization (Gulrajani and Lopez-Paz, 2020), the loss function for neural networks amounts to a summation of multiple terms, with a multiplier weighting each term. As the number of loss terms increases, the space of choice for those multipliers grows dramatically. In addition, when the validation set is drawn from a different distribution than the target domain, the validation performance does not faithfully reflect the target domain performance (Chen et al., 2022b) and can easily reach and maintain a saturated value during training. All these factors complicate and add difficulties to the process of model selection using conventional hyperparameter tuning (Probst et al., 2019).

Inspired by feedback control theory (Doyle et al., 2013) and feedback optimization in power systems (Picallo et al., 2020; He et al., 2023; Hauswirth et al., 2024), we treat the dynamic system of model parameters driven by the low level optimization algorithm as an uncontrolled plant, with the model parameters as state and the many loss terms as output. We use state-dependent multiplier induced loss landscapes (see Figure 4) as control input and designed a hierarchical control structure (Dietterich et al., 1998; Kulkarni et al., 2016; Sun et al., 2020) to drive the system towards a multi-objective descent goal, without modifying the internal dynamic of low level neural network optimization algorithms.

We demonstrate that our method removes the combination curse of dimensionality of multi-dimensional multipliers. Our method only has a small number of hyperparameters for our controllers while achieving

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

robust performance when changing them. We attribute the improved performance of our method to the automatic trade-off of different loss terms.

Our major contributions are:

- We propose a novel multi-objective optimization algorithm via constraint optimization with shrinking bound. See Proposition 3.1.
- We propose a probabilistic graphical model for depicting the joint model parameter and multiplier adaptation process with a hypervolume based likelihood, promoting multi-objective descent of each loss term, as an alternative model selection criteria in absence of trustworthy validation data.
- We bring hierarchical control to solve the joint model parameter and multiplier inference of the probabilistic graphical model as a sequential decision process through an optimal control formulation and optimize the multi-objective descent goal hierarchically by breaking the goal into a series of constraint optimization sub-goals.
- We bring feedback control theory into deep learning via treating the neural network model parameter dynamic system driven by low level optimization algorithm as an uncontrolled plant. We propose to use state-dependent multiplier-induced loss landscapes as control input and offer initial theoretical analysis of the multi-objective descent behavior of our closed loop system.

2 PRELIMINARIES

2.1 Structural risk minimization

Structural Risk Minimization (SRM) is a principle in learning algorithms that balances training performance and model complexity via optimizing the penalized loss in Equation (1).

$$
\begin{aligned}
L(\theta, \mu, \mathcal{D}_{tr}) = \ell(\theta, \mathcal{D}_{tr}) + \mu^T R(\theta, \mathcal{D}_{tr}) \\
\mu, R(\cdot) \in \mathbb{R}^d_+
\end{aligned}
\tag{1}
$$

In Equation (1), we consider loss L with training data $\mathcal{D}_{tr}$ and model parameters θ. $\ell(\theta, \mathcal{D}_{tr})$ represents the empirical risk and $R(\theta, \mathcal{D}_{tr})$ represents the regularization term of the loss function, also referred to as the penalty function, and thus L the penalized loss. $\mathbb{R}^d_+$ is the d dimensional positive octant. Here μ is the penalty (weight) multiplier, a special case of general hyperparameters (e.g. learning rate) in deep learning.

2.2 Notation

In the following sections, for brevity, we use $\ell(\cdot)$ and $R(\cdot)$ to indicate the implicit dependence of ℓ and R on $\mathcal{D}_{tr}$ and θ. We also write $R(\theta, \cdot)$ where we need an explicit dependence on θ and use $\cdot$ to implicitly represent other arguments like $\mathcal{D}_{tr}$. We use subscript to indicate the component of $R(\cdot)$, e.g., if $\mu = [\beta, \gamma]$, then $R_\beta(\cdot)$ corresponds to the component of $R(\cdot)$ weighted by β. We use superscript k in bracket to index the optimization iteration (See Remarks 3.1 and 3.2).

Definition 2.1 (Model parameter dynamic system)**.** When optimizing Equation (1) with multiplier μ iteratively, we use $\theta^{(k+1)} = f_\theta(\mu, \theta^{(k)}, \mathcal{D}_{tr}, \ell(\cdot), R(\cdot))$ to represent the map bringing $\theta^{(k)}$ to its next value $\theta^{(k+1)}$.

Following $R(\cdot), \ell(\cdot)$, we have $f_\theta(\theta, \cdot)$, $f_\theta(\mu, \cdot)$ and $f_\theta(\cdot)$ to implicitly represent omitted arguments of f_θ. We use θ^+ for new value of θ after an operation without explicitly stating how many iterations are needed, resulting in notation $\theta^+ = f_\theta(\mu, \theta, \cdot)$ and $\theta^+ = f_\theta(\theta, \cdot)$.

2.3 Multi-objective optimization

Definition 2.2 (R dominance and non-dominance)**.** We use $R(\theta_1, \cdot) \prec R(\theta_2, \cdot)$ (see Equation (1)) to indicate each component of the d dimensional vector $R(\theta_1, \cdot)$ is $\leq$ the corresponding component of $R(\theta_2, \cdot)$. We use $\not\prec$ as the negation of $\prec$. The clause $\ell(\theta_1, \cdot), R(\theta_1, \cdot) \not\prec \ell(\theta_2, \cdot), R(\theta_2, \cdot)$ AND $\ell(\theta_2, \cdot), R(\theta_2, \cdot) \not\prec \ell(\theta_1, \cdot), R(\theta_1, \cdot)$ defines an equivalence relation which we denote as $\theta_1 \sim\equiv \theta_2$.

Definition 2.3 (Reachability set and value)**.** Starting from $\theta^{(0)}$, under dynamic system $\theta^+ = f_\theta(\theta, \cdot)$, we use $s\mathcal{R}\left(\theta^{(0)}, f_\theta(\cdot)\right)$ to represent the set of model parameters (θ points) that can be reached. Accordingly, we use $v\mathcal{R}_{\ell(\cdot),R(\cdot)}\left(\theta^{(0)}, f_\theta(\cdot)\right)$ to represent the corresponding set of multi-objective function values.

Definition 2.4 (Non-dominant set map)**.** If $\theta_1 \sim\equiv \theta_2$, where $\theta_1 \in s\mathcal{R}(\theta^{(0)}, f_\theta(\cdot))$ and $\theta_2 \in s\mathcal{R}(\theta^{(0)}, f_\theta(\cdot))$ as in Definition 2.3, we define $\theta_2 \in \mathcal{C}_{\ell(\cdot),R(\cdot)}\left(\theta_1, f_\theta(\cdot), \theta^{(0)}\right)$ the map from a representing element θ_1 to its equivalence class, which contains θ_2. We use $\mathcal{E}_{\ell(\cdot),R(\cdot)}\left(\theta_1, f_\theta(\cdot), \theta^{(0)}\right)$ to represent the set of $\ell(\cdot), R(\cdot)$ values for each element of $\mathcal{C}_{\ell(\cdot),R(\cdot)}\left(\theta_1, f_\theta(\cdot), \theta^{(0)}\right)$.

2.4 Conventional multiplier tuning

The multiplier μ in Equation (1) is a hyperparameter that affects the evolution of the model parameters θ in the optimization process for L defined in Equation (1), which leads to optimized model parameter $\theta_{\mu,\mathcal{D}_{tr}}$ in Equation (2), where we omit the potential dependence of $\theta_{\mu,\mathcal{D}_{tr}}$ on its initial condition $\theta^{(0)}$ for notational simplicity.

$$\theta_{\mu,\mathcal{D}_{tr}} = \arg\min_{\theta} L(\theta, \mu, \mathcal{D}_{tr}). \tag{2}$$

Hyperparameter optimization aims to adjust the multiplier μ in alignment with specific performance metrics O (e.g. validation set accuracy for a classification task) in Equation (3) evaluated on the validation set $\mathcal{D}_{val}$, which leads to the selected hyperparameter $\mu_{\mathcal{D}_{tr},\mathcal{D}_{val}}$.

$$\mu_{\mathcal{D}_{tr},\mathcal{D}_{val}} = \arg\min_{\mu} O(\theta_{\mu,\mathcal{D}_{tr}}, \mathcal{D}_{val}). \tag{3}$$

The process of hyperparameter optimization in Equation (3) leads to selected model $\theta_{\mu_{\mathcal{D}_{tr},\mathcal{D}_{val}},\mathcal{D}_{tr}}$, which can be conceptualized as a problem of algorithm configuration (Hutter et al., 2007, 2014; López-Ibáñez et al., 2016; Sun et al., 2020) where the hyperparameter configures a machine learning algorithm.

To solve Equation (3) iteratively, each iteration relies on a complete training cycle to optimize Equation (2). The selection of μ depends on validation set $\mathcal{D}_{val}$, which can only be drawn from a different distribution compared to the target domain for the domain generalization problem (Appendix C.2). Additionally, the performance metric O on the in-domain validation set can reach and maintain a saturated value during training with neural networks of high expressive power.

3 METHODS

In this section, we elaborate on our multi-objective hierarchical output feedback optimization (M-HOF-Opt) as a control strategy in Figure 1 for the multidimensional multiplier adaptation in Equation (1). We propose a probabilistic graphical model for the joint model parameter and multiplier adaptation process in Section 3.1, which leads to an optimal control formulation of parameter estimation, extending the classical hyperparameter tuning-based approach to parameter-multiplier co-evolution in Section 3.2.

3.1 Probabilistic graphical model for joint decision of model parameter and multiplier

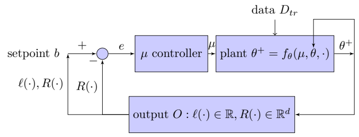
data D_{tr}
setpoint b
$+$
$-$
e
μ controller
μ
plant $\theta^+ = f_\theta(\mu, \theta, \cdot)$
θ^+
$\ell(\cdot), R(\cdot)$
$R(\cdot)$
output $O : \ell(\cdot) \in \mathbb{R}, R(\cdot) \in \mathbb{R}^d$

Figure 1: Control diagram illustrating the hierarchical output feedback optimization process with multi-objective setpoint adaptation. The uncontrolled plant corresponds to an open loop dynamic $\theta^+ = f_\theta(\mu, \theta, \cdot)$ defined in Definition 2.1, via optimizing Equation (1) with a low level optimization algorithm lacking feedback. The μ controller adjusts the multiplier μ, thus schedules the loss landscape as illustrated in Figure 4, based on the difference e between the setpoint b and the measured output components $R(\cdot)$, guiding the optimization of model parameters θ through a feedback loop. The setpoint b is adjusted via feedback from $\ell(\cdot)$ and $R(\cdot)$, which forms a higher hierarchy. See Figure 2 for the probabilistic description of the closed loop behavior of this control diagram.

Gradient based optimization algorithms for neural networks define a dynamic system for model parameters where the weight multiplier in Equation (1) configures such a dynamic system via defining a loss landscape (see Figure 4). Due to the stochastic nature of the gradient with different subsample and mini-batch distributions for each iteration, it is appropriate to use a probabilistic graphical model (Koller and Friedman, 2009) to describe such a stochastic process, which has been used to describe sequential decision processes like reinforcement learning (Levine, 2018; Sun and Bischl, 2019). Instead of fixed multipliers, we also model multiplier choice adaptively and propose the probabilistic graphical model in Figure 2 to describe the sequential decision process of the joint model parameter and multiplier adaptation.

In the upper part of Figure 2, we model the generative process of N observations of X (e.g. an image) and the corresponding supervision signal Y (e.g. a class label or self-supervision). The path $\beta \rightarrow D_{ob} \rightarrow X$ models domain-specific data generation (information not captured by Y), where D_{ob} modeling the datasite Sun et al. (2019a) with hyper-prior β.

In the lower part of Figure 2, with superscript k indexing the iteration of the decision process, we use $\mu^{(k)}$ to represent the multiplier of the loss in Equation (1) at the kth iteration. The multiplier $\mu^{(k)}$ serves as a configuration parameter for the optimization process of Equation (1) that affects the value of $\theta_\mu^{(k)}$ (evolved from its previous value $\theta^{(k-1)}$). $\theta_\mu^{(k)}$ then becomes the initial value for the next iteration. This can be described in Equation (7). The model parameter $\theta^{(k)}$ and observed data X, supervision signal Y co-parent the performance indicator $O^{(k)}$ as output of the dynamic system. See different realizations of O in Remark 3.1 and Remark 3.2.

The plate replicates B in the lower part represents the number of optimization iterations (each iteration correspond to N observations). The adaptive generation of the next multiplier $\mu^{(k+1)}$ depends on the previous value $\mu^{(k)}$ (the self-loop dashed arrow in Figure 2)

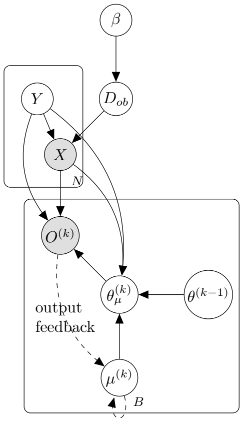

Figure 2: Probabilistic graphical model for the sequential decision process of joint model parameter and multiplier adaptation in multi-domain structral risk minimization in Equation (1). See Figure 1 for the control diagram counterpart of the same process.

and the feedback information of the output $O^{(k)}$ (long dashed arrow), as in Equation (4).

$$\mu^{(k+1)} = f_\mu(\mu^{(k)}, O^{(k)}, \cdot) \tag{4}$$

Remark 3.1. A realization of f_μ can be based on the acquisition function in Bayesian optimization (Garnett, 2023) for approximating Equation (3). See an example described in Sun et al. (2019a), where O in Equation (4) is chosen to be the validation set prediction performance and iteration index k corresponds to a whole training cycle of Equation (2) to iteratively achieve Equation (3).

Remark 3.2. Different from the Bayesian optimization realization of this probabilistic graphical model, which changes multipliers at a much slower timescale than that for the evolution of model parameters in neural network training, the hierarchical output feedback optimization we proposed in Figure 1 and elaborated in Section 3.2 and following sections operates at the timescale of epoch, thus forms another realization of this probabilistic graphical model with the iteration index k corresponding to an epoch. In addition, we have $O(\cdot) = \ell(\cdot), R(\cdot)$ in Figure 2 in contrast to validation set performance used in Bayesian optimization.

If the joint decision process of model parameter and multiplier is optimal, from a statistical inference point of view (See Remark 3.5), we are conducting a sequential inference (i.e., finding a sequence of μ in Equation (6)), such that at the last step B, the final value $\theta_{\mu^{(B)}}$ has the best possible hyper-volume (in a stochastic sense, see Figure 3) with respect to output O (see Remark 3.1 and Remark 3.2).

We introduce an objective function for estimating $\theta_{\mu^{(B)}}$ based on the profile likelihood in eq. (5) in accordance with the probabilistic graphical model in Figure 2. This likelihood promotes a high $e\mathcal{HV}(\cdot)$ in Definition 3.1 with respect to $\theta_{\mu^{(B)}}$. Ξ is the normalization factor.

$$\mathcal{P}(\theta_{\mu^{(B)}}) = \frac{1}{\Xi}\exp\left(e\mathcal{HV}_{O(\cdot)}(\theta_{\mu^{(B)}}, \theta^{(0)}, f_\theta, \cdot)\right) \tag{5}$$

Definition 3.1. $e\mathcal{HV}_{O(\cdot)=\ell(\cdot),R(\cdot)}(\theta, \theta^{(0)}, f_\theta, \cdot)$ maps θ to the dominated hypervolume (Zitzler and Künzli, 2004; Zitzler et al., 2007; Guerreiro et al., 2021) of $\mathcal{E}_{\ell(\cdot),R(\cdot)}\left(\theta, f_\theta(\cdot), \theta^{(0)}\right)$ in Definition 2.4 with respect to reference point $\ell(\theta^{(0)}, \cdot) \in \mathbb{R}, R(\theta^{(0)}, \cdot) \in \mathbb{R}^d$. As illustrated in Figure 3.

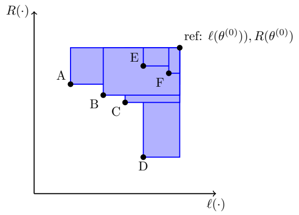

Figure 3: Illustration of $e\mathcal{HV}$ in Definition 3.1: We take $[\ell(\theta^{(0)}, \cdot), R(\theta^{(0)})]$ as reference point. Suppose $\{A, B, C, D, E, F\}$ in the illustration constitutes function values of the reachable set $v\mathcal{R}\left(\theta^{(0)}, f_\theta(\cdot)\right)$in Definition 2.3. For any θ corresponding to the point A in the illustration with coordinate $[\ell(\theta, \cdot), R(\theta, \cdot)]$, $e\mathcal{HV}$ maps θ to $\mathcal{C}_{\ell(\cdot),R(\cdot)}\left(\theta, f_\theta(\cdot), \theta^{(0)}\right)$, then to $\mathcal{E}_{\ell(\cdot),R(\cdot)}\left(\theta, f_\theta(\cdot), \theta^{(0)}\right)$ (see Definition 2.4), which is the point set {A,B,C,D}, then it calculates the dominated hypervolume with respect to the reference point (union of the shaded rectangles).

Remark 3.3. The probabilistic graphical model in Figure 2 can also be used to describe other multi-objective deep learning methods (Mahapatra and Rajan, 2021) via online combinatorial choice of multipliers.

Remark 3.4. The probabilistic graphical model also provides a potential possibility to impose a Bayesian interpretation to the joint model parameter training process and multiplier tuning for frequentist machine learning, the deeper investigation of which we leave for future work.

Remark 3.5 (The dual relationship between estimation and control)**.** Analogous to how optimal control Lewis et al. (2012) can be formulated as maximum likelihood estimation in a probabilistic graphical model (Levine, 2018; Sun and Bischl, 2019), the estimation of $\theta^{(B)}$ based on the profile likelihood like objective in Equation (5) for Figure 2 is also a sequential decision process and can be solved via optimal control.

3.2 Hierarchical control formulation of joint model parameter and multiplier optimization

3.2.1 Optimal control formulation

Continuing from Remark 3.5, to reformulate the estimation problem as an optimal control problem, we present the sequential decision on choosing multipliers $\mu^{(1)}, \ldots, \mu^{(B)}$ defined in Equation (1) with respect to the objective function in Equation (6) at iteration B which promotes optimization of Equation (5).

$$\min_{\mu^{(1)},\ldots,\mu^{(k)},\ldots,\mu^{(B)}} e\mathcal{HV}_{O(\cdot)}(\theta_{\mu^{(B)}}, \theta^{(0)}, f_\theta, \cdot) \tag{6}$$

$$\text{s.t. } \theta^{(k)} = f_\theta(\mu^{(k)}, \theta^{(k-1)}, \cdot) \tag{7}$$

$$O^{(k)} = \left[\ell(\theta^{(k)}|\ldots), R(\theta^{(k)}|\ldots)\right] \tag{8}$$

$$k = 0, 1, \ldots, B \tag{9}$$

Here, we use θ to denote the neural network weights, which can be regarded as the state of a dynamic system, where the state transition is governed by a low level model parameter optimization dynamic f_θ in Equation (7) (e.g. Kingma and Ba (2015)). From a control theory point of view, f_θ depicts the uncertain dynamic of a plant to be controlled, while the task here is to define a controller to generate a μ sequence to guide the uncertain dynamic of θ with respect to plant output O, as depicted in Figure 1. We treat the loss terms $R(\cdot)$ and $\ell(\cdot)$ as the output of the system in Equation (8). From a statistical point of view, f_θ corresponds to the parent structure of $\theta^{(k)}$ in Figure 2, which drives θ to the next value.

Remark 3.6 (closed loop dynamic of model parameter)**.** A concrete form of Equation (4) will detail how μ is updated by a controller f_μ. Combining Equations (4) and (7), we have

$$\theta^{(k+1)} = f_\theta(f_\mu(\mu^{(k)}, O^{(k)}), \theta^{(k)}, \cdot) \tag{10}$$

$$= f^k_{\theta,\mu}(\mu^{(0)}, \theta^{(0)}, \cdot) \tag{11}$$

where we use $f^k_{\theta,\mu}$ to represent the k-times compound recursive evaluations of f_θ and f_μ in Equation (10) until the dependence is only on $\theta^{(0)}, \mu^{(0)}$. We call Equation (11) the *closed-loop* dynamics for θ.

Although we have no complete knowledge of the behavior of the low level optimization algorithms described by f_θ in Equation (7), the penalized loss from Equation (1) often descends after some iterations (possibly with oscillations). This leads to Definition 3.2.

Definition 3.2 (Penalized loss descent operator)**.** A penalized loss descent operator $\mathcal{G}$ satisfies that if $\theta^+ \in \mathcal{G}_\mu(\theta; \ell(\cdot), R(\cdot))$, then $\ell(\theta^+) + \mu R(\theta^+) < \ell(\theta) + \mu R(\theta)$

Remark 3.7 (Penalized Loss Descent Assumption)**.** We take the frequent existence and occurrences of such operators defined in Definition 3.2 during neural network training as a mild assumption in our following discussion. For instance, we could run f_θ in Equation (7) in one step to have $\ell(\theta^{(k+1)}) + \mu^{(k+1)} R(\theta^{(k+1)}) < \ell(\theta^{(k)}) + \mu R(\theta^{(k)})$, or in more than one steps to achieve Definition 3.2.

Our final goal, however, is to ensure a joint descent of $\ell(\cdot)$ and $R(\cdot)$, which leads to Definition 3.3.

Definition 3.3. A Pareto-descent operator takes θ to $\theta^{(+)}$ (via one or several iterations) which descents $R(\cdot)$ and $\ell(\cdot)$ simultaneously, i.e.,

$$\ell(\theta^+) < \ell(\theta) \quad \text{and} \quad R(\theta^+) \preceq R(\theta) \tag{12}$$

3.2.2 Multi-objective optimization via shrinking the reference signal in constrained optimization

How can we design a closed-loop dynamic system in Remark 3.6 to ensure a Pareto descent of $O(\cdot)$, leading to multi-objective optimization in Definition 3.3 or Equation (6)? To resolve this, we reduce it to a simpler problem, i.e., when bounding the regularization term $R(\cdot)$ by a time-varying reference bound $b^{(k)}$ (a.k.a. setpoint) in Equation (18), what value can we achieve for $\ell(\cdot)$ at Equation (16). If we could ensure the reference bound $b^{(k)}$ changes monotonically, multi-objective optimization can be achieved.

Thus, we define the multi-dimensional reference bound $b^{(k)}$ (setpoint) in Equations (13) to (14):

$$b^{(0)} = \rho R^{(0)}, 0 < \rho \in \mathbb{R} < 1 \tag{13}$$

$$b^{(k)} = g_b(\ell^{(0:k)}(\cdot), R^{(0:k)}(\cdot)) \tag{14}$$

where $g_b(\cdot)$ represents the mapping from the previous value at the training step $k-1$ to the new value at step k, defined as Equation (15).

$$b^{(k)} = \begin{cases} R^{(k)}, \text{ if } R^{(k)} \prec b^{(k-1)} \textbf{ and} \\ \ell^{(k)} < \min\limits_{j=0,\cdots,k-1} \ell^{(j)}, & \text{(15a)} \\ b^{(k-1)}, \text{ otherwise} & \text{(15b)} \end{cases}$$

Remark 3.8. The shrinkage of $b^{(k)}$ in Equation (15) also depends on $\ell(\cdot)$ decrease, thus indicates multi-objective Pareto-descent in Definition 3.3.

Proposition 3.1 (Pareto-descent via constrained optimization with shrinking bound)**.** With $b^{(k)}$ defined in Equation (15), suppose the following constrained optimization Bertsekas (2014) problem in Equations (16) to (18) starting with $\theta^{(k)}$, under $s_k + m_k$ number of iterations ($s_k > 0$ and $m_k \geq 0$) has a solution

$$\min_{\mu^{(k+1)},\ldots,\mu^{(k+s_k+m_k)}} \ell^{(k+s_k+m_k)}(\cdot) \tag{16}$$

$$\text{s.t. } R^{(j_1)}(\cdot) \not\prec b^{(k)}, j_1 = k, \ldots, k+s_k-1 \tag{17}$$

$$\theta^{(j+1)} = f_\theta(\mu^{(j+1)}, \theta^{(j)}, \cdot)\ (Equation\ (7))$$

$$R^{(k+j_2)}(\cdot) \prec b^{(k)}, j_2 = s_k, \ldots, s_k + m_k \tag{18}$$

we achieve multi-objective descent in Definition 3.3 at step $k + s_k + m_k$ compared to step k.

Proof. See Appendix A. □

Corollary 3.1 (Approximation of constraint optimization)**.** Regardless of the attainability of the minimization in Equation (16), as long as we have

$$\ell^{(k+s_k+m_k)}(\cdot) < \ell^{(k)}(\cdot) \tag{19}$$

we achieve multi-objective descent at step $k + s_k + m_k$.

Proof. See Appendix A. □

We discuss how to approximate the constrained optimization in Section 3.2.3.

3.2.3 Loss landscape scheduling

The evolution of the model parameters can be regarded as a plant with uncertain dynamic f_θ in Equation (7), where the only control we have at hand is the sequence $\mu^{(k)}$, which provides different loss landscapes at different iterations as depicted in Figure 4. How can we design such loss landscape sequence to promote Equation (18) to happen? To simplify the discussion, first consider $\mu, R(\cdot) \in \mathbb{R}_+^{d=1}$. Before the constraint in Equation (18) is satisfied, design $\mu^{(k+1)}, \ldots, \mu^{(k+s_k)}$ to be an increasing sequence Bazaraa et al. (2013), which gives more weight to the corresponding regularization term in $R(\cdot)$, to approximate the constraint optimization in Equation (16) by optimizing Equation (1) for each μ until Equation (18) is satisfied. However, will the increased weight on the regularization term in $R(\cdot)$ result in deteriorated $\ell(\cdot)$? One possibility can be, although $f_\theta(\mu^{(k+1)}, \theta^{(k)}, \cdot)$ in Equation (7) brings $\theta^{(k)}$ to the next value $\theta^{(k+1)}$ which decreases the penalized loss in Equation (1) with $\mu^{(k+1)}$, $\theta^{(k+1)}$ evaluates to a deteriorated penalized loss with respect to the old $\mu^{(k)}$ as shown in Figure 4. To describe this behavior during the increasing process of μ, we have Equation (21) in Definition 3.4. This situation is of particular interest due to the possibility of overcoming local minima of a fixed loss landscape, as indicated in Figure 4.

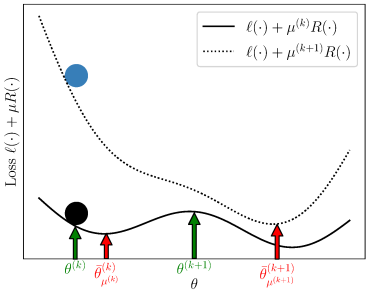

Figure 4: Illustration of multiplier induced loss landscape scheduling, used as control signal in Figure 1: The lifted landscape $\ell(\cdot) + \mu^{(k+1)}R(\cdot)$ (dotted curve) scheduled at iteration $k+1$, enables the model parameter dynamic to overcome the local minimum $\bar{\theta}^{(k)}_{\mu^{(k)}}$ of the old loss landscape $\ell(\cdot) + \mu^{(k)}R(\cdot)$ (solid curve) scheduled at iteration k, as can be imagined via the two balls with different colors rolling along the corresponding loss landscapes. This showcases a scenario corresponding to Definition 3.4: In comparison to $\theta^{(k)}$, $\theta^{(k+1)}$ corresponds to a decreased $\ell(\cdot) + \mu^{(k+1)}R(\cdot)$ value but increased $\ell(\cdot) + \mu^{(k)}R(\cdot)$ value.

Definition 3.4. A reg-Pareto slider $\mathcal{A}(\cdot)$ with respect to function $\ell(\cdot)$ and $R(\cdot)$ is defined to be a set map, s.t. if $\{\theta^{(k+1)}, \mu^{(k+1)}\} \in \mathcal{A}(\ell(\cdot), R(\cdot); \theta^{(k)}, \mu^{(k)})$ then

$$\ell(\theta^{(k+1)}) + \mu^{(k+1)}R(\theta^{(k+1)}) \leq \ell(\theta^{(k)}) + \mu^{(k+1)}R(\theta^{(k)}) \tag{20}$$

$$\ell(\theta^{(k+1)}) + \mu^{(k)}R(\theta^{(k+1)}) \geq \ell(\theta^{(k)}) + \mu^{(k)}R(\theta^{(k)}) \tag{21}$$

$$\mu^{(k+1)} \geq \mu^{(k)} \tag{22}$$

3.2.4 Analysis

Based on Definition 3.4, we have the following conclusion in Proposition 3.2 where we show the increase of $\ell(\cdot)$ as an expense to decrease of $R(\cdot)$ is bounded.

Proposition 3.2 (Bounded Pareto Trade-off)**.** Let

$$\{\theta^{(k+1)}, \mu^{(k+1)}\} \in \mathcal{A}(\ell(\cdot), R(\cdot); \theta^{(k)}, \mu^{(k)}) \tag{23}$$

from definition 3.4 assume $R(\cdot) \in \mathbb{R}^{d=1}$, then

$$R(\theta^{(k+1)}) \leq R(\theta^{(k)}) \tag{24}$$

$$\ell(\theta^{(k+1)}) \geq \ell(\theta^{(k)}) \tag{25}$$

$$\ell(\theta^{(k+1)}) - \ell(\theta^{(k)}) \leq \mu^{(k+1)}(R(\theta^{(k)}) - R(\theta^{(k+1)})) \tag{26}$$

Proof. See Appendix A. □

Remark 3.9 (Multi-dimensional R)**.** In the case of $R(\cdot) \in \mathbb{R}^{d>1}$, when all components of μ increase, Equation (20) and Equation (21) imply at least one component of $R(\cdot)$ will decrease. We defer to Remark 3.11 and Section 3.2.5 for the discussion on when some components of μ increase while other components decrease.

Remark 3.10 (Decrease of $\ell(\cdot)$)**.** Proposition 3.2 give a condition of decreasing $R(\cdot)$ at the expense of increasing $\ell(\cdot)$ (bounded though by Equation (26)) at iteration k. When the gradient component from $\ell(\cdot)$ and $R(\cdot)$ agrees with each other at another iteration, we could expect both objectives to decrease as in Definition 3.3. It can also be the case at a particular iteration, $\ell(\cdot)$ decreases at the expense of $R(\cdot)$ increases.

What is left unclear, however, is after the accumulative effect of several iterations, whether these exists an s_k, m_k ensuring Equation (19). To answer this, we derive Proposition 3.3, which gives a bound on the accumulative change of $\ell(\cdot)$ with $B = s_k + m_k$.

Proposition 3.3 (Single-step ℓ increase bound)**.** Suppose that for all $k = 0, \ldots, B-1$

$$\ell(\theta^{(k+1)}) + \mu^{(k+1)} R(\theta^{(k+1)}) \leq \ell(\theta^{(k)}) + \mu^{(k+1)} R(\theta^{(k)})$$

and $\mu^{(k+1)} > \mu^{(k)}$. Then

$$\ell(\theta^{(B)}) \leq \ell(\theta^{(0)}) + S_> + S_<, \tag{27}$$

where

$$S_> = \sum_{k \in \mathcal{K}_>} \mu^{(k+1)} (R(\theta^{(k)}) - R(\theta^{(k+1)})),$$

$$S_< = \sum_{k \in \mathcal{K}_<} \mu^{(k+1)} (R(\theta^{(k)}) - R(\theta^{(k+1)})),$$

$$\mathcal{K}_> = \{k \in \{0, \ldots, B-1\} : \textit{ Equation } (21) \text{ holds.}\},$$

$$\mathcal{K}_< = \{0, \ldots, B-1\} \setminus \mathcal{K}_>.$$

Proof. See Appendix A. □

Based on Proposition 3.3, we have the following conjecture to ensure Equation (19)

Conjecture 3.1 (Multi-step ℓ decrease)**.** Further decompose $S_<$ from Proposition 3.3 into $S_<^-$ and $S_<^+$ via decomposing $\mathcal{K}_<$ in Proposition 3.3 into $\mathcal{K}_<^-$ and $\mathcal{K}_<^+$. Here $\mathcal{K}_<^-$ corresponds to the situations of $\ell(\cdot)$ descent with $S_<^- < 0$. And $\mathcal{K}_<^+$ corresponds to the $\ell(\cdot)$ ascent cases but upperbounded by $S_<^+ = \sum_{k \in \mathcal{K}_<^+} \mu^{(k+1)} (R(\theta^{(k)}) - R(\theta^{(k+1)}))$. Then $\exists$ sequence $\{\mu^{(k)}\}$ s.t.

$$\ell(\theta^{(B)}) - \ell(\theta^{(0)}) \leq S_> + S_<^+ + S_<^- < 0 \tag{28}$$

Remark 3.11 (Trade-off Multi-dimensional μ)**.** In the case of $d > 1$, as long as one loss component of $R^{(k)}(\cdot)$, without loss of generality, say $R_1^{(k)}(\cdot)$ is not bounded by its corresponding setpoint component $b_1^{(k)}(\cdot)$, we can still increase the value of the μ component $\mu_1^{(k)}$ to give more weight to the corresponding loss component, until the loss component in question decreases below the setpoint. However, an increased μ component $\mu_1^{(k)}$ corresponds to more weight on the gradient component corresponding to $R_1^{(k)}(\cdot)$, which makes it harder for other components of $R(\cdot)$ and $\ell(\cdot)$ to decrease. Therefore, instead of always increasing each component of μ, if one component of R has overshoot over the setpoint (i.e. constraint switched from being not satisfied to satisfied with respect to a particular loss component), we decrease the μ value to give the other components of μ and R more feasible space to adjust themselves.

Remark 3.12 (Multi-step Pareto Descent)**.** We could extend Conjecture 3.1 to $\exists \mu$ sequence such that after B iterations, Pareto descent in Definition 3.3 could be achieved (equivalently, setpoint should shrink see Remark 3.8). We did observe such a Pareto descent behavior in our experiment for the $d > 1$ case of Equation (40), e.g. see Figure 5 and Figure 6.

Algorithm 1 Multi-objective hierarchical output feedback optimization

```
procedure M-HOF-OPT($\mu^{(0)}, \theta^{(0)}, \rho, \cdot$)
    Initialize $\mu^{(0)}$, calculate $b^{(0)} = \rho R^{(0)}$.
    Compute $K_I$ based on Remark 3.13.
    while budget $B$ not reached do
        Update $\mu$ from controller in Equation (31).
        Update $\theta$ via Equation (7) with updated $\mu$.
        Adapt setpoint according to Equation (14).
    end while
    return $\theta^{(B)}$                                   ▷
end procedure
```

3.2.5 Output feedback PI-like controller

Based on Remark 3.11, with the adaptive law from Equations (14) and (15) for setpoint b, we design the following controller (see Figure 1) for μ:

$$e^{(k)} = R(\cdot) - b^{(k)} \tag{29}$$

$$\delta_I^{(k+1)} = (1 - \xi_d)\delta_I^{(k)} + \xi_d e^{(k)} \tag{30}$$

$$\mu^{(k+1)} = \max(\mu^{(k)} \exp^{\max(K_I \delta_I^{(k+1)}, v_{sat})}, \mu_{clip}) \tag{31}$$

$$K_I > 0 \tag{32}$$

$$K_I \in \mathbb{R}^d \tag{33}$$

$$k = 0, 1, \ldots, B \tag{34}$$

In Equation (29), we calculate how far the current output is away from the setpoint b, which gets passed through a moving average in Equation (30) with ξ_d being the coefficient of moving average in Equation (30) (Rezende and Viola, 2018). μ is the output of the controller in Equation (31), K_I is the control gain for PI (proportional-integration) (Johnson and Moradi, 2005) like control, $K_I \delta_I^{(k+1)}$ is component wise multiplication, exp and max are computed component wise,

v_{sat} is the exponential shoulder saturation, which defines the maximum rate of change of multipliers. μ_{clip} is the upper bound for μ. Note that μ_{clip} determines the dynamic range of μ while v_{sat} determines an upper bound about how fast μ changes.

Remark 3.13. To avoid arbitrary choice of $K_I \in \mathbb{R}^d$, we use $\delta^{(0)} = R^{(0)} - b^{(0)}$ in Equation (30) where $b^{(0)} = \rho R^{(0)}$ (a percentage ρ). We divide by ηv_{sat} with $0 < \eta < 1$ to get the value for K_I so that the hyperparameter for our algorithm is $\rho, \eta \in \mathbb{R}$ instead of $K_I \in \mathbb{R}^d$.

The whole process is summarized in Algorithm 1, which details Figure 1.

3.2.6 Multi-objective setpoint based model selection

In general, the feasibility in Equation (18) becomes more difficult to be met each time the setpoint b shrinks. After several setpoint shrinkages, the difficulty of attaining the feasibility can lead to an oscillating behavior of $R(\cdot)$ around the setpoint with uncontrolled amplitude, as shown in Figure 5. To circumvent this behavior, the up-to-event best model is selected at the last setpoint shrinkage.

3.3 Related work and discussion

We discuss the advantages of our method compared to other multiplier scheduling and multi-objective optimization methods in Appendix B.

4 EXPERIMENTS

We conduct experiments and benchmarks to answer the following questions: Is M-HOF-Opt capable of adjusting multidimensional multipliers automatically to drive each component of $R(\cdot)$ at different scales to the setpoint? Does the setpoint shrink, which implies multi-objective descent? See Figure 5 and Figure 6 in Section 4.1.

Will wrong multiplier choices for warmup and fixed multiplier training schemes result in catastrophic effects? In contrast, does varying controller hyperparameters have detrimental effects on M-HOF-Opt? See Figure 7 in Section 4.2, as well as Figure 8 in Appendix C.4.

Does changing the lower-level optimization algorithm (i.e. the open-loop plant or the feedback-free model parameter dynamic system M-HOF-Opt has to control) have an impact on the performance of M-HOF-Opt? See Appendix C.5.

With respect to the above questions, we got favorable results for M-HOF-Opt in all experiments which we analyze below. The experimental and benchmark setting is detailed in Appendix C.1 with implementation in `https://github.com/marrlab/DomainLab/tree/mhof` from *DomainLab* (Sun et al., 2024).

4.1 Illustration of multi-objective descent

Since the domain generalization model DIVA (Ilse et al., 2020) with math formulation explained in Appendix C.2.1 has 6 loss terms[1], it is a good example to demonstrate the advantage of our algorithm in trading off many loss terms. Our hierarchical output feedback optimization training scheme produced the dynamics of $R(\cdot)$ shown in Figure 5 together with the corresponding setpoints b and multipliers. Note that the multiplier $\beta_y, \gamma_d, \mu_{recon}$ and their corresponding loss components operate at different numerical ranges, but our controller still manages to drive the different $R(\cdot)$ loss terms down at different scales and rates.

Remark 4.1 (setpoint shrinkage)**.** Regarding the dynamics of how the setpoint b defined in Equations (13) to (14) decreases in Figure 5: Initially, Equation (17) holds, in which case we do not update the setpoint, even if some but not all the blue curves (corresponding to $R(\cdot)$ components) are below the red curves (components of the setpoint). After all constraints for each component of $R(\cdot)$ are satisfied as described in Equation (18), the setpoint (red curves) gets adapted to a new value (which we term shrinkage), as a new goal to be reached by the $R(\cdot)$ loss in Equation (46). Note that the shrinkage of setpoint depends on Pareto dominance in Equation (15), while non-dominance defined an equivalent relation in Definition 2.2.

Figure 6 shows the output portrait of $\ell(\cdot)$ versus a component of $R(\cdot)$ of our training scheme This confirms the multi-step Pareto descent discussed in Remark 3.12.

With the multi-objective setpoint based model selection criteria in Section 3.2.6, our method admits a multi-objective descent of the selected model compared to the initial output, such that the uncontrolled behavior at the end of iterations can be safely ignored.

4.2 Benchmark results

We demonstrate the power of M-HOF-Opt in automatic adaptation of multipliers via benchmarking out-of-domain generalization performance in Figure 7 for training DIVA Ilse et al. (2020) compared to baseline training schemes. From Figure 7, we can observe that the generalization performance of baselines is highly sensitive to multiplier combination. In contrast, our

[1] At time of writing, we are not aware of any other deep learning model with more loss terms.

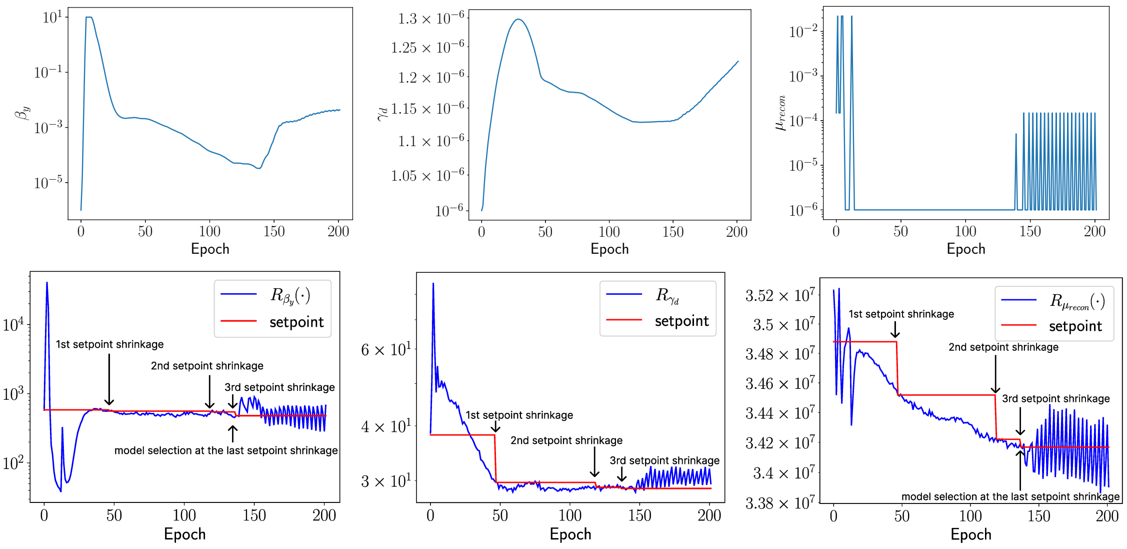

Figure 5: Our method drives different loss terms of $R(\cdot)$ at different scales and rates towards the setpoint, which further promotes the setpoint shrinkage. In the **Top row**, we show the multiplier dynamic as controller output signal in Equation (31) across training epochs. In the **Bottom row**, we present the tracking behavior of the corresponding regularization loss $R(\cdot)$ in Equation (40) with respect to setpoint b defined in Equation (14).

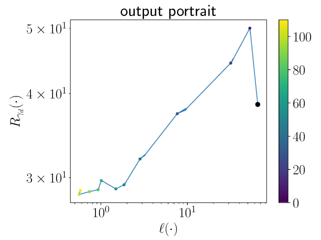

Figure 6: Our training scheme adapts the multi-dimensional multiplier μ, steering the optimization process towards a configuration that balances the trade-off between minimizing output $\ell(\cdot) = \mathbb{E}_{q_{\phi_y}(z_y|x)}[\log q_{\omega_y}(y|z_y)]$ in Equation (41) and output $R_{\gamma_d}(\cdot) = \mathbb{E}_{q_{\phi_d}(z_d|x)}[\log q_{\omega_d}(d|z_d)]$ in Equation (46). In this plot, the solid large point corresponds to the initial output. We use gradually changing colors to indicate the training iterations (epochs), indicated by the color bar. For improved visualization, we exclusively utilized the initial 120 epochs and plotted data points solely for every 10 epochs. This figure corresponds to the experimental setting in Figure 5.

method, as supported by the adaptive behaviors visualized in Figure 5, automatically adjusts the multipliers during the training process, leading to a robust performance with respect to changes of controller hyperparameters and multiplier initial condition $\mu^{(0)}$. Our method does not need hyperparameter searching of the multiplier, thus saves a fair amount of computational resources. For additional experiments, see Appendix C.4.

5 CONCLUSION

This work uses control theory to address the issue of combinatorial choice for multidimensional multipliers weighting many loss terms in structural risk minimization for deep neural networks by proposing a novel multi-objective optimization algorithm via constraint optimization with shrinking bound. We develop a probabilistic graphical model for joint model parameter and multiplier optimization with respect to a multi-objective descent of all loss terms. The estimation of model parameter leads to an automatic adjustment scheme for the multipliers, adopting a hierarchical control scheme, which breaks the multi-objective optimization problem into a series of constraint optimization sub-problems. Each sub-problem is configured with a self-adaptive multi-objective setpoint updated via Pareto dominance. A PI-like multiplier controller drives the loss term to satisfy the setpoint constraint for each sub-problem. Our method operates at the timescale of epoch level during training, thus circumvents the need for exhaustive multiplier search and saves tremendous computational resources compared to methods like Bayesian optimization. It also circumvents the excessive memory requirements and heavy computational burden of existing multi-objective deep learning methods. Our method demonstrates robust out-of-domain generalization performance against controller hyperparameter variation compared to other multiplier choice or scheduling schemes which produce very unstable behavior in the training due to the need for a combinatorial choice of multipliers.

Acknowledgements

We sincerely thank Dr. Changxin Liu and Dr. Prof. Jann Rolfes for spending hours on proofreading the manuscript and offering many helpful suggestions to the manuscript. We thank Tomas Chobola for *Inkscape* support. We thank the anonymous reviewers for offering suggestions to enhance the readability of our paper.

References

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Mokhtar S Bazaraa, Hanif D Sherali, and Chitharanjan M Shetty. *Nonlinear programming: theory and algorithms*. John wiley & sons, 2013.

Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019.

Nutan Chen, Patrick van der Smagt, and Botond Cseke. Local distance preserving auto-encoders using continuous knn graphs. In *Topological, Algebraic and Geometric Learning Workshops 2022*, pages 55–66, 2022a.

Yongqiang Chen, Kaiwen Zhou, Yatao Bian, Binghui Xie, Bingzhe Wu, Yonggang Zhang, Kaili Ma, Han Yang, Peilin Zhao, Bo Han, et al. Pareto invariant risk minimization: Towards mitigating the optimization dilemma in out-of-distribution generalization. *arXiv preprint arXiv:2206.07766*, 2022b.

Thomas G Dietterich et al. The maxq method for hierarchical reinforcement learning. In *ICML*, volume 98, pages 118–126, 1998.

John C Doyle, Bruce A Francis, and Allen R Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

Andreia P Guerreiro, Carlos M Fonseca, and Luís Paquete. The hypervolume indicator: Computational problems and algorithms. *ACM Computing Surveys (CSUR)*, 54(6):1–42, 2021.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

Adrian Hauswirth, Zhiyu He, Saverio Bolognani, Gabriela Hug, and Florian Dörfler. Optimization algorithms as robust feedback controllers. *Annual Reviews in Control*, 57:100941, 2024.

Zhiyu He, Saverio Bolognani, Jianping He, Florian Dörfler, and Xinping Guan. Model-free nonlinear feedback optimization. *IEEE Transactions on Automatic Control*, 2023.

Frank Hutter, Holger H Hoos, and Thomas Stützle. Automatic algorithm configuration based on local search. In *Aaai*, volume 7, pages 1152–1157, 2007.

Frank Hutter, Manuel López-Ibánez, Chris Fawcett, Marius Lindauer, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. Aclib: A benchmark library for algorithm configuration. In *Learning and Intelligent Optimization: 8th International Conference, Lion 8, Gainesville, FL, USA, February 16-21, 2014. Revised Selected Papers 8*, pages 36–40. Springer, 2014.

Maximilian Ilse, Jakub M Tomczak, Christos Louizos, and Max Welling. Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning*, pages 322–348. PMLR, 2020.

Michael A Johnson and Mohammad H Moradi. *PID control*. Springer, 2005.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference for Learning Representations*, 2015.

A. Klushyn, N. Chen, R. Kurle, B. Cseke, and P. van der Smagt. Learning hierarchical priors in VAEs. *Advances in Neural Information processing Systems*, 32, 2019.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Matan Levi, Idan Attias, and Aryeh Kontorovich. Domain invariant adversarial learning. *arXiv preprint arXiv:2104.00322*, 2021.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Frank L Lewis, Draguna Vrabie, and Vassilis L Syrmos. *Optimal control*. John Wiley & Sons, 2012.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *Advances in neural information processing systems*, 32, 2019.

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5:5, 2017.

Debabrata Mahapatra and Vaibhav Rajan. Exact pareto optimal search for multi-task learning: Touring the pareto front. *arXiv e-prints*, pages arXiv–2108, 2021.

Miguel Picallo, Saverio Bolognani, and Florian Dörfler. Closing the loop: Dynamic state estimation and feedback optimization of power grids. *Electric Power Systems Research*, 189:106753, 2020.

Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1):1934–1965, 2019.

Alexandre Rame, Corentin Dancette, and Matthieu Cord. Fishr: Invariant gradient variances for out-of-distribution generalization. In *International Conference on Machine Learning*, pages 18347–18377. PMLR, 2022.

D. J. Rezende and F. Viola. Taming VAEs. *CoRR*, 2018.

Michael Ruchte and Josif Grabocka. Scalable pareto front approximation for deep multi-objective learning. In *2021 IEEE international conference on data mining (ICDM)*, pages 1306–1311. IEEE, 2021.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Huajie Shao, Shuochao Yao, Dachun Sun, Aston Zhang, Shengzhong Liu, Dongxin Liu, Jun Wang, and Tarek Abdelzaher. Controlvae: Controllable variational autoencoder. In *International Conference on Machine Learning*, pages 8655–8664. PMLR, 2020.

Anthony Sicilia, Xingchen Zhao, and Seong Jae Hwang. Domain adversarial neural networks for domain generalization: When it works and how to improve. *Machine Learning*, pages 1–37, 2023.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *Advances in neural information processing systems*, 29, 2016.

Xudong Sun and Bernd Bischl. Tutorial and survey on probabilistic graphical model and variational inference in deep reinforcement learning. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, number 1908.09381, 2019.

Xudong Sun and Florian Buettner. Hierarchical variational auto-encoding for unsupervised domain generalization. *ICLR 2021 RobustML, arXiv preprint arXiv:2101.09436*, 2021.

Xudong Sun, Andrea Bommert, Florian Pfisterer, Jörg Rahnenführer, Michel Lang, and Bernd Bischl. High dimensional restrictive federated model selection with multi-objective bayesian optimization over shifted distributions. *arXiv preprint arXiv:1902.08999*, 2019a.

Xudong Sun, Alexej Gossmann, Yu Wang, and Bernd Bischl. Variational resampling based assessment of deep neural networks under distribution shift. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, number 1906.02972, 2019b.

Xudong Sun, Jiali Lin, and Bernd Bischl. Reinbo: Machine learning pipeline conditional hierarchy search and configuration with bayesian optimization embedded reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 68–84. Springer International Publishing, 2020.

Xudong Sun, Carla Feistner, Alexej Gossmann, George Schwarz, Rao Muhammad Umer, Lisa Beer, Patrick Rockenschaub, Rahul Babu Shrestha, Armin Gruber, Nutan Chen, et al. Domainlab: A modular python package for domain generalization in deep learning, 2024.

Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *International conference on parallel problem solving from nature*, pages 832–842. Springer, 2004.

Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *Evolutionary Multi-Criterion Optimization: 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007. Proceedings 4*, pages 862–876. Springer, 2007.

CHECKLIST

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes: See Section 2, Section 3, Appendix C.2]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes: See Appendix B]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable to camera-ready submission]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes: See Section 3]
   (b) Complete proofs of all theoretical results. [Yes: See Appendix A]
   (c) Clear explanations of any assumptions. [Yes: See Section 3]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes: See Section 4, Appendix C.1]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes: See Appendix C.1]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes: See caption of Figure 7]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes: See Appendix D]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes: See Appendix C.1, Appendix E]
   (b) The license information of the assets, if applicable. [Yes: See Appendix E]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes: See Section 4]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

APPENDIX

A MATHEMATICAL PROOFS

In this seciton, we restate theoretical results in the main text with the same statement index and provide proofs.

Proposition 3.1 (Pareto-descent via constrained optimization with shrinking bound)**.** With $b^{(k)}$ defined in Equation (15), suppose the following constrained optimization Bertsekas (2014) problem in Equations (16) to (18) starting with $\theta^{(k)}$, under $s_k + m_k$ number of iterations ($s_k > 0$ and $m_k \geq 0$) has a solution

$$\min_{\mu^{(k+1)},\ldots,\mu^{(k+s_k+m_k)}} \ell^{(k+s_k+m_k)}(\cdot) \quad (16)$$

$$\text{s.t. } R^{(j_1)}(\cdot) \not\prec b^{(k)}, j_1 = k, \ldots, k+s_k-1 \quad (17)$$

$$\theta^{(j+1)} = f_\theta(\mu^{(j+1)}, \theta^{(j)}, \cdot) \ (\textit{Equation } (7))$$

$$R^{(k+j_2)}(\cdot) \prec b^{(k)}, j_2 = s_k, \ldots, s_k + m_k \quad (18)$$

we achieve multi-objective descent in Definition 3.3 at step $k + s_k + m_k$ compared to step k.

Proof. Here $s_k > 0$ is the number of iterations maintaining constraint in Equation (17), and $s_k + m_k$ is the number of iterations to ensure the decrease of ℓ subject to decreasing R in Equation (18).

Note that $b^{(k+j)}$ remains fixed before s_k, i.e. $b^{(k)} = \cdots = b^{(k+s_k+m_k-1)}$ due to Equation (15). Between s_k till $s_k + m_k$, due to Equation (15), R has decreased.

Due to Equation (18), the new reference bound $b^{(k+s_k)}$ (setpoint) has been respected, so $R^{(k+s_k+j)}$ with $0 \leq j \leq m_k$ still remains better than $R^{(k)}$, even though it can be worse than intermediate values $R^{(k+s_k+j^{'})}$ with $0 < j^{'} < j$.

Now due to Equation (15), the minimization of $\ell^{(k+s_k+m_k)}$ cannot lead to values worse than $\ell^{(k)}$ because otherwise Equation (18) cannot be respected.

Thus both ℓ and R decrease at step $k + s_k + m_k$ compared to step k.

□

Corollary 3.1 (Approximation of constraint optimization)**.** Regardless of the attainability of the minimization in Equation (16), as long as we have

$$\ell^{(k+s_k+m_k)}(\cdot) < \ell^{(k)}(\cdot) \tag{19}$$

we achieve multi-objective descent at step $k + s_k + m_k$.

Proof. Feasibility in Equation (19) and the condition in Equation (18) ensures Pareto-descent in Definition 3.3, following the proof in Proposition 3.1. □

Proposition 3.2 (Bounded Pareto Trade-off)**.** Let

$$\{\theta^{(k+1)}, \mu^{(k+1)}\} \in \mathcal{A}(\ell(\cdot), R(\cdot); \theta^{(k)}, \mu^{(k)}) \tag{23}$$

from definition 3.4 assume $R(\cdot) \in \mathbb{R}^{d=1}$, then

$$R(\theta^{(k+1)}) \leq R(\theta^{(k)}) \tag{24}$$

$$\ell(\theta^{(k+1)}) \geq \ell(\theta^{(k)}) \tag{25}$$

$$\ell(\theta^{(k+1)}) - \ell(\theta^{(k)}) \leq \mu^{(k+1)}(R(\theta^{(k)}) - R(\theta^{(k+1)})) \tag{26}$$

Proof. Multiply -1 to both sides of Equation (21), add the results to Equation (20), we have

$$(\mu^{(k+1)} - \mu^{(k)})R(\theta^{(k+1)}) \leq (\mu^{(k+1)} - \mu^{(k)})R(\theta^{(k)}) \tag{35}$$

which implies

$$(\mu^{(k+1)} - \mu^{(k)}) \left(R(\theta^{(k+1)}) - R(\theta^{(k)})\right) \leq 0 \tag{36}$$

Since $\mu^{(k+1)} > \mu^{(k)}$

$$R(\theta^{(k+1)}) - R(\theta^{(k)}) \leq 0 \tag{37}$$

Equation 21 and 20 give

$$0 \leq \mu^{(k)}(R(\theta^{(k)}) - R(\theta^{(k+1)})) \leq \tag{38}$$

$$\ell(\theta^{(k+1)}) - \ell(\theta^{(k)}) \leq \mu^{(k+1)}(R(\theta^{(k)}) - R(\theta^{(k+1)})) \tag{39}$$

□

Proposition 3.3 (Single-step ℓ increase bound)**.** Suppose that for all $k = 0, \ldots, B-1$

$$\ell(\theta^{(k+1)}) + \mu^{(k+1)}R(\theta^{(k+1)}) \leq \ell(\theta^{(k)}) + \mu^{(k+1)}R(\theta^{(k)})$$

and $\mu^{(k+1)} > \mu^{(k)}$. Then

$$\ell(\theta^{(B)}) \leq \ell(\theta^{(0)}) + S_> + S_<, \tag{27}$$

where

$$\begin{aligned} S_> &= \sum_{k \in \mathcal{K}_>} \mu^{(k+1)}(R(\theta^{(k)}) - R(\theta^{(k+1)})), \\ S_< &= \sum_{k \in \mathcal{K}_<} \mu^{(k+1)}(R(\theta^{(k)}) - R(\theta^{(k+1)})), \\ \mathcal{K}_> &= \{k \in \{0, \ldots, B-1\} : \textit{ Equation } (21) \text{ holds.}\}, \\ \mathcal{K}_< &= \{0, \ldots, B-1\} \setminus \mathcal{K}_>. \end{aligned}$$

Proof. For $k \in \mathcal{K}_>$ (Equation (21) holds), from Proposition 3.2 we have that

$$\ell(\theta^{(k+1)}) - \ell(\theta^{(k)}) \leq \mu^{(k+1)}(R(\theta^{(k)}) - R(\theta^{(k+1)})).$$

For $k \in \mathcal{K}_<$, it holds that

$$\ell(\theta^{(k+1)}) + \mu^{(k)}R(\theta^{(k+1)}) < \ell(\theta^{(k)}) + \mu^{(k)}R(\theta^{(k)}),$$

implying

$$\ell(\theta^{(k+1)}) - \ell(\theta^{(k)}) < \mu^{(k)}(R(\theta^{(k)}) - R(\theta^{(k+1)})).$$

Decomposing

$$\begin{aligned} &\ell(\theta^{(B)}) - \ell(\theta^{(0)}) \\ &= \sum_{k=0}^{B-1} [\ell(\theta^{(k+1)}) - \ell(\theta^{(k)})] \\ &= \Big(\sum_{k \in \mathcal{K}_>} + \sum_{k \in \mathcal{K}_<} \Big) [\ell(\theta^{(k+1)}) - \ell(\theta^{(k)})] \end{aligned}$$

yields the conclusion.

□

B RELATED WORK AND DISCUSSION

In this section, we discuss the advantages of our method compared to other multiplier scheduling and multi-objective optimization methods, especially in time and space (memory) complexity. Compared to hyperparameter tuning including Bayesian optimization as discussed in Remark 3.2, since in general the next multiplier has to be generated after one training round, in the extreme worst case, the number of searches reduces to grid search of B_d^d number of combinations (where B_d is the number of grid points for each multiplier component, d is the dimension of the multiplier). Our method however requires only a single run.

B.1 Multiplier adaptation in deep learning

Multipliers warmup (Sønderby et al., 2016; Ilse et al., 2020) or phase-in (Sicilia et al., 2023) requires ultimate multiplier values for each loss term, thus necessitates hyperparameter (ultimate multiplier values) optimization, see also analysis in Remark C.1. Other works, such as (Rezende and Viola, 2018; Klushyn et al., 2019), has introduced constraint optimization by representing the weights of loss terms using Lagrange multipliers, designed to prevent over-regularisation of a single term within the loss function, which is extended to multiple loss terms in (Chen et al., 2022a). In addition, similar multiplier adaptation schemes were proposed by (Shao et al., 2020), who only dealt with two loss terms, and by (Chen et al., 2022a), where the multiplier changes at a rate proportional to the distance of the corresponding loss component from a setpoint, as discussed in Section 3.2.5. However, their methods relied on a fixed constraint bound for each loss term and fixed proportionate gain for each loss term (defined to be K_I in Equation (31) in our work), necessitating a hyperparameter search, which is challenging when the number of terms in the loss increases. Moreover, for some regularizers, such as the ones based on KL divergence, it is challenging to estimate the constraint bound range. Specifically in the field of domain generalization, it is not obvious how to choose the constraint bound value and other hyperparameters due to the lack of observations from the target domain, where the validation set from the training domain can easily reach a saturated value due to the high expressive power of modern neural networks.

In comparison, we deal with many loss terms with multidimensional multipliers and introduced Equation (14) to progressively adapt the constraint bound and Remark 3.13 to choose the multidimensional controller gain value, thus avoid exhaustive hyperparameter search. In addition to the control theory formulation for multiplier adjustment, we also provided a probabilistic graphical model interpretation in Section 3.1 and a landscape scheduling interpretation in Figure 4. Furthermore, we discussed our method under the multi-objective optimization scheme compared to single-objective consideration from other works. Different from the single layer controller in (Chen et al., 2022a), we proposed a hierarchical controller in Section 3.2.5 to break the multi-objective optimization problem into a series of constraint optimization subproblems, each configured with a self-adaptive multi-objective setpoint updated via Pareto dominance.

B.2 Multi-objective optimization for large-scale neural networks

Earlier multi-objective deep learning (Sener and Koltun, 2018; Lin et al., 2019) suffers from increased computational cost with the number of losses, explicit computation of the gradient with respect to each loss term and gradient norms, making it practically infeasible Mahapatra and Rajan (2021) for large neural networks and large datasets. For instance, Pareto Invariance Risk Minimization (PAIR) (Chen et al., 2022b) studies domain generalization with the use of the multi-objective optimization technique Exact Pareto Optimal Search (EPOS) (Mahapatra and Rajan, 2021), which requires explicit full gradient information for each loss term and computation of the C matrix with complexity m^2n (Mahapatra and Rajan, 2021). With m being the number of loss terms (equivalent to $d+1$ in our paper), and n the dimension of gradient, which is super high in modern neural networks (e.g. 200 million for ResNet50), thus results in heavy memory requirements and excessive computation burden, and hinders the deployment to large-scale networks (Chen et al., 2022b). The Dirichlet sampling of preference vectors was introduced, and the aggregated optimal was computed under the sampled preference vectors (Ruchte and Grabocka, 2021). However, their sampling of preference vectors introduced data replications. In addition, their method needs to augment the preference vector with the original input, which added complexity of applying their method to different data modes. For instance, for image input, they need first to transform the preference vector to image mode via transpose convolution. Further, their method needs an extra hyperparameter to weight the additional penalty loss to force the solution to obey the preference vectors.

In comparison, our multi-objective optimization method (M-HOF-Opt) need only d scalar multiplications, thus the extra computation is negligible and can be used in large-scale neural networks. EPOS Mahapatra and Rajan (2021) also requires the user to provide a preference vector which can be difficult to set, say with 6 objectives for our case. In addition, Chen et al. (2022b) reported hyperparameter tuning for the preference vector and step length is needed for the method to work properly. In comparison, our method, M-HOF-Opt is preference vector free. It ensures the Pareto descent due to the update rule of our setpoint. There is no requirement to search for step length.

Note that the EPOS method Mahapatra and Rajan (2021) is searching for Exact Pareto optimal (Mahapatra and Rajan, 2021), which might not exist. Chen et al. (2022b) divides the training into two phases where the first phase uses single-objective optimization favoring ERM loss while the second phase is for balancing different loss terms. However, how many computation resources should one set for the first phase is not clear. Furthermore, the first stage (phase) can affect the exact Pareto optimality.

C EXPERIMENTAL DETAILS

C.1 Experimental Setting

This section presents our experiments training the domain generalization model DIVA (Ilse et al., 2020) with 6 loss terms defined in Equation (40) on the widely used domain generalization benchmark dataset PACS (Li et al., 2017) with different training schemes. Due to the excessive memory requirements and heavy computation of earlier multi-objective deep learning methods discussed in Appendix B, which hinders their application, we compare our training scheme with other multiplier scheduling techniques. Given that the *sketch* domain within the PACS dataset is considered the most inherently challenging, it was chosen as the sole leave-one-out domain to test the performance of domain generalization.

To further show the power of M-HOF-Opt in automatically balancing different loss terms, we combined two domain generalization algorithms IRM (Arjovsky et al., 2019) and DIAL (Levi et al., 2021).

In our experiment, we used the Facebook Research version of ResNet from DomainBed[2] (Gulrajani and Lopez-Paz, 2020) with a learning rate of 5e-5 and batch size of 32.

We first introduce domain generalization and detail the loss formulation of the above mentioned methods and their loss combination, then we present the benchmark setting in Appendix C.3, Appendix C.4 and the corresponding experimental results.

C.2 Domain generalization structral risk

Domain generalization aims at enabling the trained neural network to achieve robust generalization to unseen domains exhibiting distribution shifts (Gulrajani and Lopez-Paz, 2020; Sun et al., 2019b). Many domain generalization methods (Ganin et al., 2016; Li et al., 2018; Levi et al., 2021; Carlucci et al., 2019; Ilse et al., 2020; Sun and Buettner, 2021; Rame et al., 2022) promotes this goal via domain invariant representation by adding domain invariant regularization losses $R(\cdot)$ upon task-specific losses $\ell(\cdot)$ (e.g. classification loss) on the training data in Equation (1).

C.2.1 Domain Invariant Variational Autoencoding (DIVA)

As an example of $\ell(\cdot) + \mu^T R(\cdot)$ loss for domain generalization, we label each component of the loss of Domain Invariant Variational Autoencoding (DIVA) (Ilse et al., 2020) in the following equations, where we use x to denote the input instance (e.g. input image), y to denote the corresponding supervision class label, and d the domain label. DIVA (Ilse et al., 2020) aims to disentangle domain specific information with learned features z_d and class specific information with z_y, and has two corresponding classification losses $\mathbb{E}_{q_{\phi_y}(z_y|x)}[\log q_{\omega_y}(y|z_y)]$ (classifying the correct class label based on z_y) and $\mathbb{E}_{q_{\phi_d}(z_d|x)}[\log q_{\omega_d}(d|z_d)]$ (classifying the correct domain label based on z_d).

$$L(\theta, \mu, x, y, d) = \ell(\cdot) + \mu^T R(\cdot) \tag{40}$$

$$= \gamma_y \mathbb{E}_{q_{\phi_y}(z_y|x)}[\log q_{\omega_y}(y|z_y)] + \tag{41}$$

$$\mu_{recon} \mathbb{E}_{q_{\phi_d}(z_d|x), q(z_x|x), q(z_y|x)} \log p_{\theta_r}(x|z_d, z_x, z_y) \tag{42}$$

$$+ [-\beta_x KL(q_{\phi_x}(z_x|x)||p_{\theta_x}(z_x))] \tag{43}$$

$$+ [-\beta_y KL(q_{\phi_y}(z_y|x)||p_{\theta_y}(z_y|y))] \tag{44}$$

$$+ [-\beta_d KL(q_{\phi_d}(z_d|x)||p_{\theta_d}(z_d|d))] \tag{45}$$

$$+ \gamma_d \mathbb{E}_{q_{\phi_d}(z_d|x)}[\log q_{\omega_d}(d|z_d)] \tag{46}$$

Here, KL denotes Kullback-Leibler divergence, $\mathbb{E}$ denotes expectation, $p(\cdot)$ stands for the prior distribution or the distribution of the generative model, and $q(\cdot)$ stands for the approximate posterior distribution. For more details, refer to (Ilse et al., 2020). In the above situation, we have $\mu = [\mu_{recon}, \beta_x, \beta_y, \beta_d, \gamma_d]$ and $\theta = [\phi_y, \phi_d, \theta_r, \theta_d, \theta_x, \theta_y, \phi_x, \phi_y, \phi_d, \omega_d, \omega_y]$ with each entry representing the weight (with bias) of a neural network.

In (Ilse et al., 2020), $\mu_{recon} = 1.0$ in Equation (42). Additionally, γ_y and γ_d are maintained as constants. γ_y is corresponding to $\ell(\cdot) = \gamma_y \mathbb{E}_{q_{\phi_y}(z_y|x)}[\log q_{\omega_y}(y|z_y)]$ in Equation (41), and γ_d in Equation (46) is associated with one component of $R(\cdot)$. However, the combined choice of γ_y and γ_d significantly influences the generalization performance, as shown by our experimental findings in Figure 7.

Remark C.1 (feedforward scheduler)**.** For the multiplier $\beta_x, \beta_y, \beta_d$ corresponding to other components of $R(\cdot)$, Ilse et al. (2020) used a warm-up strategy to increase them gradually from a small value to a predefined value. This, however, still requires a choice of the ultimate values for $\beta_x, \beta_y, \beta_d$. In (Ilse et al., 2020), these ultimate values are simply set to one. We coin this kind of multiplier scheduling strategy a *feedforward*

[2] https://github.com/facebookresearch/DomainBed/blob/main/domainbed/networks.py

scheme. Note that this strategy does not reduce the number of hyperparameters to be selected, since ultimate values for the multipliers still have to be specified.

C.2.2 Invariant Risk Minimization (IRM) and Domain Invariant Adversarial Learning (DIAL)

Let Φ be the feature extraction neural network, w be the task network (e.g. classifier), let d index available training domains. Invariant Risk Minimization (IRM) uses the following optimization:

$$\min_{\Phi,w} \sum_d \ell^{(d)}(w \circ \Phi(X)) + \mu \sum_d \|\nabla_{w|w=1.0} \ell^{(d)}(w \circ \Phi(X))\|^2 \tag{47}$$

while Domain Invariant Adversarial Learning proposed using adversarial perturbed image (X_{adv} in comparison to original data X) as new training samples to form adversarial loss. If we combine these two, we essentially get the following loss:

$$\begin{aligned} &\ell + \mu^T R \\ &= \min_{\Phi,w} \sum_d \ell^{(d)}(w \circ \Phi(X)) + \\ &\mu_1 \sum_d \|\nabla_{w|w=1.0} \ell^{(d)}(w \circ \Phi(X))\|^2 + \\ &\mu_2 \ell(w \circ \Phi(X_{adv})) \end{aligned} \tag{48}$$

C.3 Benchmark settings for DIVA

In the benchmark, we sample different combinations of hyperparameters, (including multiplier for baselines and controller hyperparameter for our method) for each method to be compared. For baselines, we sample $\gamma_d \in \{1, 1001, 100001\}$ and $\gamma_y \in \{1, 1001, 100001\}$ value combinations, and set the ultimate values of β_y, β_d to be 1.0, following (Ilse et al., 2020). Our method does not need to set ultimate values for μ, so we sample controller hyperparameters, as well as different initial conditions $\mu^{(0)}$ (Each component of the $\mu^{(0)}$ vector set to be the same value).

C.4 Benchmark settings and results for IRM and DIAL

For baseline (named "Fixed Multipliers" in Figure 8), we choose μ_1, μ_2 in Equation (48) from $0.01, 0.1, 1.0, 10$.

For our method M-HOF-Opt, we choose different controller hyperparameters, with $\mu^{(0)} = 1e^{-6}, 0.001$, $\eta = 0.325, 0.775$ (uniform between 0 and 1), $\mu^{(clip)} = 10, 100, 1000$.

In Figure 8, each box plot inside the rectangle correspond to a hyperparameter configuration. For baseline "Fixed Multipliers" in Figure 8, this corresponds to μ_1, μ_2 values in Equation (48). For M-HOF-Opt, each box plot corresponds to different controller hyperparameters.

From Figure 8, we can see that M-HOF-Opt performs robust on out-of-domain generalization against controller hyperparameter changes, while when searching for fixed μ_1, μ_2 combinations in Equation (48) lead to worse results.

C.5 Effects of different low level optimizers

In this section, we investigate if our algorithm M-HOF-Opt still works when using different low level (gradient descent) optimization algorithms. Using the same setting as in Appendix C.4, we obtain Table 1 for changing *Adam* with *AdamW* (Loshchilov et al., 2017) and Table 2 for stacking on top of the low level optimization algorithm with *CosineAnnealingLR*[3]. The results show that M-HOF-Opt is robust against changes of low level optimization algorithms.

Method	**Mean**	**Std**
Warmup	0.767	0.0467
Fixed Multipliers	0.748	0.052
M-HOF-Opt	0.806	0.0456

Table 1: M-HOF-Opt still achieves better out-of-domain generalization accuracy when replacing Adam with AdamW as low level optimizer. For benchmark and experimental setting, see Appendices C.1 and C.5.

Method	**Mean**	**Std**
Warmup	0.750	0.0441
Fixed Multipliers	0.7322	0.0570
M-HOF-Opt	0.7873	0.0326

Table 2: M-HOF-Opt still achieves favorable out-of-domain generalization accuracy when stacking *CosineAnnealingLR* as another hierarchy on top of the low level gradient based optimization algorithm. M-HOF-Opt treats the stacked *CosineAnnealingLR* and lower level gradient optimization algorithm as the uncontrolled plant. See Appendices C.1 and C.5 for benchmark and experimental setting.

[3] `https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.CosineAnnealingLR.html`

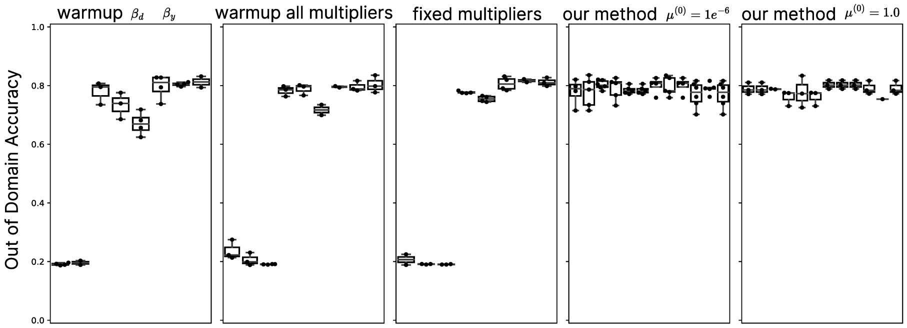

Figure 7: Our automatic multiplier adjustment scheme ensures robust out-of-domain test accuracy when training DIVA with the modified *ResNet50* (Gulrajani and Lopez-Paz, 2020) on PACS dataset (testing on domain *sketch* and training on domain *photo*, *art-painting* and *cartoon*). Each panel (i.e., subplot) corresponds to a training scheme. Each box plot inside a panel corresponds to a specific hyperparameter combination (i.e. multipliers value for baselines and controller hyperparameters for M-HOF-Opt). Inside each box plot, we repeat the experiment with different random seeds corresponding to the dots scatter. In the first panel, we warm up only β_y, β_d in Equations (44) and (45) to their ultimate value 1.0 while sampling fixed multiplier γ_d and γ_y while fixing $\mu_{recon} = 1.0$. In the second panel, additionally, we also warm up μ_{recon} in Equation (42) to 1.0 and warm up γ_d in Equation (46) to the sampled value, and keep γ_y fixed to be the sampled value. In the third panel, we use fixed constant multipliers without warm-up while keep $\beta = 1.0$ and $\mu_{recon} = 1.0$ and only sample γ_d, γ_y combinations. The last two panels correspond to results utilizing our multi-objective hierarchical feedback optimization training method, and we let μ include all multipliers except $\gamma_y = 1.0$ in Equation (41). For simplicity, when we write $\mu^{(0)} = 1.0$ where $\mu^{(0)} \in \mathbb{R}_+^d$, we mean each component of $\mu^{(0)}$ equals 1.0. For reproducibility, see `https://github.com/marrlab/DomainLab/tree/mhof`.

D COMPUTER RESOURCES

All experiments were conducted on our internal computation clusters with NVIDIA V100 GPUs.

E LICENSES FOR EXISTING ASSETS

Code Our implementation leverages several open-source libraries:

- **PyTorch**: BSD 3-Clause License. `https://pytorch.org/`
- **NumPy**: BSD 3-Clause License. `https://numpy.org/`
- **Matplotlib**: Matplotlib License. `https://matplotlib.org/`

In our experiment, we used the Facebook Research version of ResNet from DomainBed (`https://github.com/facebookresearch/DomainBed/blob/main/domainbed/networks.py`) which is MIT License.

Data We used the PACS dataset for non-commercial research purposes and we cited the author.

F CONTRIBUTIONS

XS proposed the idea, developed the theories and algorithms, implemented the code, designed and carried out the experiments, processed experimental data and generated the figures, wrote the manuscript. NC proposed the PI-like multiplier adaptation in Section 3.2.5, helped XS with PI-like controller implementation, held various discussions, proofread and improved the manuscript. AG initiated the visualization code for Figure 5 and Figure 6, refined by XS, proofread and improved the manuscript. YX developed Proposition 3.3 and corresponding remarks with XS, proofread and improved the constraint optimization part of the theory, discussed on control theory part. CF initiated the benchmark code with refinements from XS, proofread Proposition 3.2. MW improved the code for *DomainLab*, especially M-HOF-Opt. ED and FD tested and improved the code for benchmark, proofread the manuscript. FD further improved the *tensorboard* visualization of the algorithm. LB proofread Proposition 3.2, tested the benchmark code. DS tested the benchmark code, assisted XS in collecting experimental results. CM supervised the project, offered extensive proofreads of the manuscript.

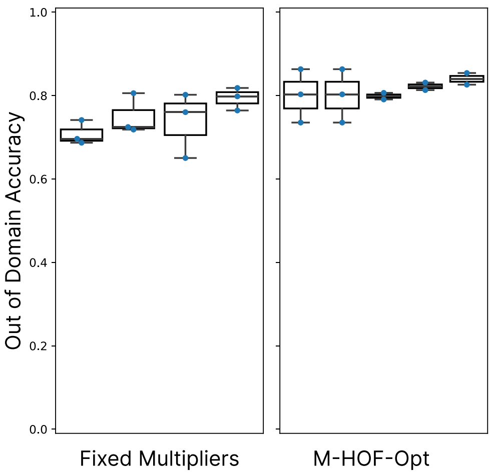

Figure 8: M-HOF-Opt achieves better out-of-domain generalization accuracy compared to using fixed multipliers, when combining regularization loss from Domain Invariant Adversarial Learning (DIAL) and Invariant Risk Minimization (IRM) in Appendix C.2.2. Each panel (i.e., subplot) corresponds to a training scheme. Each box plot inside a panel corresponds to a specific hyperparameter combination (i.e. multiplier values for baseline “Fixed Multipliers” and controller hyperparameters for M-HOF-Opt). Inside each box plot, we repeat the experiment with different random seeds corresponding to the dots scatter. From the figure, we can see that the baseline performance varies a lot as the multiplier combination changes while M-HOF-Opt remains robust against controller hyperparameter variations. See Appendices C.1 and C.4 for benchmark setting.