# YOU ONLY TRAIN ONCE: GRADIENT-BASED LOSS HYPERPARAMETER OPTIMIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The process of creating and optimizing a learned model inevitably involves multiple training runs which potentially feature different architectural designs, input and output encodings, and losses. Our method, You Only Train Once (YOTO), contributes to limiting training to one shot w.r.t. the latter aspect of selection and weighting of losses. It achieves this by automatically optimizing loss weight hyperparameters of learned models in one shot via standard gradient-based optimization, treating these hyperparameters as regular parameters of the networks and learning them. To this end, we leverage the differentiability of the composite loss formulation which is widely used for optimizing multiple empirical losses simultaneously. We model this formulation as a novel layer, parameterized with a softmax operation which satisfies the inherent positivity constraints on loss hyperparameters while avoiding degenerate empirical gradients. We complete our joint end-to-end optimization scheme by defining a novel regularization loss on the learned hyperparameters, which models a uniformity prior among the employed losses while ensuring boundedness of the identified optima. We evidence the efficacy of YOTO in jointly optimizing loss hyperparameters and regular model parameters in one shot by comparing it to the commonly used grid search and random search across transformers and CNNs for two representative vision tasks, i.e. the regression task of 3D estimation and the classification task of semantic segmentation, and showing that it consistently outperforms both grid and random search on unseen test data at a fraction of the compute.

## 1 INTRODUCTION

Artificial intelligence has been driven in recent years via very large hierarchical parametric models in the form of neural networks, whose success is largely due to the invention of effective and efficient algorithms for optimizing them (Rumelhart et al., 1986). The most widely used class of such algorithms is based on gradient descent (GD), because gradients of the minimization objective with respect to model parameters can be computed fast and thus afford numerous rapid iterations of the algorithm, which helps to quickly improve the objective even with stochastic updates that use very small fractions of the dataset (Bottou, 1998; LeCun et al., 1998) and to converge to fair local optima.

While optimizing a given instance of a neural network architecture on a given data-based objective, i.e. a *given empirical loss*, is thus by now well-understood and solvable, (i) selecting an optimal architecture itself as well as (ii) optimally *selecting and weighting empirical losses themselves still pose fundamental challenges* to the community. The first of these two points has given rise to the relatively new research area of neural architecture search (NAS) (Zoph & Le, 2017; Liu et al., 2019). Moreover, both (i) and (ii) involve the optimization of *hyperparameters* (HPs), i.e. parameters which are of a different, potentially non-differentiable or even non-continuous nature than regular parameters of neural network layers. Hyperparameter optimization (HPO) (Maclaurin et al., 2015; Pedregosa, 2016; Bengio, 2000; Rajeswaran et al., 2019) consists precisely in optimizing such higher-level parameters of either the architecture or the losses. However, the crux of both NAS and HPO in practice is that they typically require multiple complete training runs in order to first compute the validation performance of the network before making a single iteration in the architecture space or the HP space. Each outer iteration of this meta-optimization is thus extremely inefficient compared to inner, gradient-based iterations in the parameter space.

In this work, we focus on point (ii), i.e. on the selection and weighting of losses. We start with the observation that a large portion of HPO experiments in practice lies in the optimization of *HPs which weigh multiple empirical losses* and which typically appear as weights of a linear combination of such losses. A piece of evidence for the ubiquity of this setting comes from randomly examining the 20 papers which appear at the top of the online list of the CVPR 2024 proceedings (cvp, 2024): 10 out of the 20 optimize such a linear combination of distinct empirical losses. More broadly, notable vision models that fall in this setting include Gaussian Splatting (Kerbl et al., 2023), pix2pix (Isola et al., 2017), the R-CNN series (Girshick, 2015; Ren et al., 2015; He et al., 2017), and DETR (Carion et al., 2020), and respective language models include PaLM (Chowdhery et al., 2023), Switch Transformers (Fedus et al., 2022), and DistilBERT (Sanh et al., 2019). Because of the moderate dimensionality of this HP space and the aforementioned typical inefficiency and intensity of sophisticated HPO techniques, practitioners in learning areas such as vision or language commonly resort to simple yet computationally intensive brute-force approaches, such as grid search in this space (Fedus et al., 2022), to optimize these loss HPs. However, such approaches suffer from the curse of dimensionality, which makes them cumbersome even for optimizing two HPs when a suitable range of values is not known a priori. Can we instead directly optimize loss weight HPs simultaneously and jointly with regular model parameters *in one shot*, building on standard gradient-based methods?

We present a novel optimization algorithm, named You Only Train Once (YOTO), and experimental validation of it that jointly answer this question affirmatively. YOTO hinges on the linearity of the above composite empirical loss to the involved loss weight HPs, which allows us to express this loss as the ultimate, differentiable layer of the overall end-to-end model. In turn, gradients with respect to the loss HPs can be computed and backpropagated in order to update the latter together with the regular parameters of the network, i.e. to apply standard gradient-based learning on them. We bake the inherent positivity of loss weights in our novel loss layer by operating in a logarithmic space, and decouple the loss scale from the learning rate via a softmax parameterization which ensures normalization. Akin to weight decay (Loshchilov & Hutter, 2019) used for regular network weights which reside in a Euclidean space, we complete YOTO with a novel hyperparameter decay for regularization, which uses the gradient of a negated entropy term and a softplus term that promote uniformity and upper-boundedness, respectively. We evidence—through experiments on two central regression and classification tasks in vision—that *letting the loss HPs of neural networks be jointly optimized with the networks' regular parameters* via normal gradient-based optimization delivers models that *exceed the generalization capability of gradient-based optimization of only regular parameters*, in which loss HPs are segregated from regular parameters, kept fixed in each optimization run, and optimized by computationally far more intensive and slow brute-force or meta-learning-based approaches. Last but not least, YOTO exhibits robustness to stochasticity and initialization.

## 2 Background and Related Work

HPO is as old as machine learning itself (Akaike, 1974; Box & Wilson, 1951), as it arises from the ubiquitous empirical need for an optimal experimental design and thus plays a critical role in the field. HPs not only determine the generalization capabilities of trained models, but may actually decide which method constitutes the state of the art. The distinction between standard optimization of parametric models and HPO has been invariably based on the *nested* nature of the latter (Franceschi et al., 2024). Formally, let $f(\boldsymbol{\lambda})$ denote a learned mapping configured by the HP vector $\boldsymbol{\lambda}$ and consider a function $\mathcal{M}$ that evaluates the performance of $f$ on a learning task. The composition $h(\boldsymbol{\lambda}) = \mathcal{M} \circ f(\boldsymbol{\lambda})$ is referred to as the response function. In this setting, HPO is invariably viewed as optimizing $h$ with respect to $\boldsymbol{\lambda}$ in the literature (Bengio, 2000; Snoek et al., 2012; Maclaurin et al., 2015; Klein et al., 2017; Lorenzo et al., 2017). Under this regime, a diverse set of methods have been proposed for HPO, ranging from (i) gradient-based methods which compute approximate "hypergradients" of $h$ w.r.t. $\boldsymbol{\lambda}$ (Bengio, 2000; Pedregosa, 2016; Larsen et al., 1998; Franceschi et al., 2017; Maclaurin et al., 2015; Franceschi et al., 2018; Grazzi et al., 2020; Franceschi et al., 2024; Liao et al., 2018), to (ii) model-based methods (Bergstra et al., 2011; Hutter et al., 2011; Snoek et al., 2012; Klein et al., 2017; Falkner et al., 2018), which are intrinsically characterized by a sequential operation in exploring the HP space, and to (iii) population-based methods (Hansen & Ostermeier, 1996; Jaderberg et al., 2017; Lorenzo et al., 2017; Loshchilov & Hutter, 2016; Tao et al., 2020), which are by contrast parallel but require the computationally intensive maintenance of a multitude

of concurrent HP estimates. Across all these classes of HPO as well as HPO methods that combine them (Tao et al., 2020; Falkner et al., 2018), one needs to run an *entire optimization* of the mapping $f$ w.r.t. its regular parameters $\mathbf{w}$ on a standard empirical loss to fit $f$ to the available data, e.g. a full training run for a neural network, in order to evaluate the response function $h$ at a *single* point $\boldsymbol{\lambda}$ in the HP space and optimize the outer objective, in what is known as *bi-level optimization*. Multi-fidelity HPO (Li et al., 2018a;b; Karnin et al., 2013; Jamieson & Talwalkar, 2016; Klein et al., 2017; Falkner et al., 2018; Swersky et al., 2013; 2014; Bornschein et al., 2020) attempts to mitigate the fundamental inefficiency of bi-level optimization by tuning the regular parameters $\mathbf{w}$ in the main optimization faster via early stopping or training on partial data, but does not cancel the fact that a single update in the HP space is orders of magnitude slower than an update of regular parameters, e.g. via GD.

The standard argument for justifying the disjoint, bi-level optimization of regular parameters and HPs is that the analytical form of $h$ is typically either unknown or impractical to handle. Yet these problems also exist for the regular parameters $\mathbf{w}$ which the learned mapping $f$ needs to optimize for and they have been largely solved for complex differentiable models by backpropagation-based GD. While the training objectives of mappings $f$ cannot be differentiated with respect to all their HPs, we recognize that this possibility does exist for the weights $\boldsymbol{\lambda} \in \mathbb{R}^{K+1}$ of the widely used composite empirical loss in equation 1 below. Moreover, the regular parameters $\mathbf{w}$ are typically also not tuned directly on the performance function $\mathcal{M}$, but rather on the original optimization objective or loss. Thus, there is no fundamental reason preventing us from rethinking HPO, treating these loss weight HPs $\boldsymbol{\lambda}$ as regular parameters of $f$ and optimizing them regularly on the same empirical loss as regular parameters rather than on $\mathcal{M}$.

Particularly related to our optimization of a composite loss function with multiple independent components is multi-objective HPO (Emmerich et al., 2011; Keane, 2006; Knowles, 2006; Hernández-Lobato et al., 2016; Zhang & Golovin, 2020; Belakaria et al., 2019). A key difference is that these works focus rather on multiple *performance* or validation-level objectives than on multiple *training-level* objectives, i.e. empirical losses. Also related is constrained HPO (Gelbart et al., 2014; Gardner et al., 2014; Letham et al., 2019; Perrone et al., 2019), which typically expresses constraints in terms of the value of the validation-level objective and thereby indirectly constrains the feasible set of HPs, whereas our optimized loss HPs have direct intrinsic constraints on their values per se, as they need to be positive. Finally, our work is related to NAS (Zoph & Le, 2017; Yang et al., 2020; Zhou et al., 2020; Dong et al., 2021; Izquierdo et al., 2021; Zela et al., 2018; Dong & Yang, 2019; Dai et al., 2021; Ren et al., 2021; Elsken et al., 2019) in the wide sense and shares analogies with methods from that area proposing a differentiable formulation (Liu et al., 2019; Zela et al., 2020), but we solely focus on continuous HPs involved in the optimized loss rather than on discrete or categorical parameters involved in the models' modules.

# 3 Loss Hyperparameter Optimization via Standard Gradient-Based Learning

## 3.1 Preliminaries

In the design of parametric learned mappings $f$ which are optimized by minimizing a basic empirical loss $l_0$ with respect to the parameters $\mathbf{w}$ of $f$, practitioners often introduce additional loss terms $l_i$, $i \in \{1, \dots, K\}$ in the overall empirical loss, which typically play an auxiliary role and help obtain mappings that generalize better to unseen data during inference. The overall empirical loss $L_e$ becomes

$$L_e(f(\mathbf{w})|\boldsymbol{\lambda}) = \sum_{i=0}^{K} \lambda_i l_i(f(\mathbf{w})), \tag{1}$$

where $\lambda_i > 0$, $i \in \{0, \dots, K\}$, are positive HPs which serve as weights of the respective loss terms and which also need to be optimized besides the regular parameters $\mathbf{w}$ of the mapping. We term the empirical loss $L_e$ of equation 1 as *composite empirical loss*. However, the standard practice is to optimize $\boldsymbol{\lambda}$ on the *validation* set, based on the final performance metrics of $f(\mathbf{w})$ having been optimized on the *training* set, e.g. through GD. The former optimization of $\boldsymbol{\lambda}$ typically involves a non-end-to-end, inefficient search over several fixed values of $\boldsymbol{\lambda}$, which suffers from the curse of dimensionality w.r.t. the number $K + 1$ of HPs.

### 3.2 Composite Loss Layer

First, we recognize that the common formulation of the composite empirical loss in equation 1 is essentially a linear parametric mapping of the $K+1$ inputs $l_i(f(\mathbf{w}))$ to the output $L_e$, parameterized by $\boldsymbol{\lambda}$. Thus, we can *extend* the typical end-to-end formulation of parametric mappings from the commonly defined end of predictions $f(\mathbf{w})$ to the ultimate end of $L_e$ itself as

$$L_e = g(\boldsymbol{\lambda}) \circ \mathbf{l} \circ f(\mathbf{w}) = g(\mathbf{l}(f(\mathbf{w})), \boldsymbol{\lambda}) = \boldsymbol{\lambda}^T \mathbf{l}(f(\mathbf{w})), \tag{2}$$

where $\mathbf{l}(f(\mathbf{w})) = (l_0(f(\mathbf{w})), \ldots, l_K(f(\mathbf{w}))^T$. We term $g$ the composite loss layer. Since $g$ is linear, it is *differentiable*, hence amenable to standard gradient-descent-based methods and backpropagation for optimization. That is, we can optimize the HPs $\boldsymbol{\lambda}$ of $g$ as regular parameters at each iteration of gradient-based optimization, together with the regular parameters $\mathbf{w}$ of $f$. This paradigm shift leaves the computation of the gradients of the regular parameters $\mathbf{w}$ w.r.t. $L_e$ unchanged, as

$$\frac{\partial L_e}{\partial \mathbf{w}} = \frac{\partial \left(\boldsymbol{\lambda}^T \mathbf{l}(f(\mathbf{w}))\right)}{\partial \mathbf{w}} = \boldsymbol{\lambda}^T \frac{\partial \mathbf{l}(f(\mathbf{w}))}{\partial \mathbf{w}} = \sum_{i=0}^{K} \lambda_i \frac{\partial l_i(f(\mathbf{w}))}{\partial \mathbf{w}}. \tag{3}$$

However, because of the positivity constraints $\lambda_i > 0$, $i \in \{0, \ldots, K\}$, we cannot directly apply standard unconstrained gradient optimization on $\boldsymbol{\lambda} \in \mathbb{R}^{K+1}$ merely with the formulation of equation 2.

### 3.3 Softmax Parameterization

Instead of directly learning the weights $\lambda_i$ themselves, one can learn the exponents

$$\mu_i = \log(\lambda_i) \in \mathbb{R},\ i \in \{0, \ldots, K\}, \tag{4}$$

which are amenable to unconstrained gradient-based optimization. This ensures that the weights, which now become exponentials $\lambda_i = \exp(\mu_i)$, are positive by definition.

Nonetheless, this exponential trick alone makes the gradients w.r.t. the HPs degenerate, in particular positive. More formally, if we substitute equation 4 into equation 1 and differentiate w.r.t. $\mu_i$, we obtain $\frac{\partial L_e}{\partial \mu_i} = \exp(\mu_i) l_i(f(\mathbf{w})) > 0$, which implies that all gradient-based updates will reduce the exponents $\mu_i$ and hence the weights $\lambda_i$. Intuitively, there is the incentive to reduce the weight of each term of the composite loss, so that the overall loss $L_e$ is reduced too.

We circumvent this degeneracy by introducing a softmax parameterization of the composite loss layer. This parameterization adopts the exponential trick above, but also additionally introduces a normalization factor, which creates competition between different HPs and loss terms. More formally, we formulate the composite loss layer as

$$L_e = \sum_{i=0}^{K} \exp(\mu_i) l_i(f(\mathbf{w})) \Bigg/ \sum_{j=0}^{K} \exp(\mu_j) = (\text{Softmax}(\boldsymbol{\mu}))^T \mathbf{l}(f(\mathbf{w})). \tag{5}$$

For the weights $\boldsymbol{\lambda} = \text{Softmax}(\boldsymbol{\mu})$, besides the preservation of the positivity constraints, we also have $\sum_{i=0}^{K} \lambda_i = 1$, implying that the composite loss $L_e$ now becomes a *convex combination* of the individual loss terms $l_i$. With the softmax parameterization of equation 5, the HP gradients become

$$\frac{\partial L_e}{\partial \mu_i} = \exp(\mu_i) \left( \sum_{\substack{j=0 \\ j \neq i}}^{K} (l_i(f(\mathbf{w})) - l_j(f(\mathbf{w}))) \exp(\mu_j) \right) \Bigg/ \left( \sum_{j=0}^{K} \exp(\mu_j) \right)^2. \tag{6}$$

The proof of equation 6 is in Appendix A. Because of the differences $l_i(f(\mathbf{w})) - l_j(f(\mathbf{w}))$ which appear in the numerator on the RHS of equation 6, the HP gradients can be either positive or negative, thus solving the above degeneracy. As what is important to learn in the formulation of equation 5 for the composite loss layer are the *relative* values of the exponents $\mu_i$, the associated degrees of freedom are actually $K$. Thus, we freeze the exponent $\mu_0$ corresponding to the basic loss $l_0$ to 0 throughout the optimization and only learn $\mu_i$ for $i \in \{1, \ldots, K\}$, i.e. for the auxiliary losses, using the gradients from equation 6.

Another important attribute of the softmax parameterization is that it preserves the *scale* of the composite loss to the same level as that of the basic loss $l_0$. The basic loss $l_0$ is typically the starting point in the design of a mapping which may subsequently include auxiliary losses $l_i, i \geq 1$. The scale of $l_0$ affects the optimal scale of the learning rate $\eta$, for optimizers which are not scale-invariant, such as stochastic gradient descent or SGDW (Loshchilov & Hutter, 2019), because (i) the gradient is a linear operator and directly inherits a change of scale in the loss, and (ii) the learning rate acts as a multiplier of the loss gradient, so a change of scale in the loss effectively acts as a change of scale in the learning rate. By merely modifying $l_0$ to a composite loss $L_e$ via equation 1 without constraining $\lambda_i$, one would change the scale of the overall loss by a factor of $\sum_{i=0}^{K} \lambda_i \neq 1$, effectively changing the learning rate and altering the operating range of such scale-dependent optimizers. By contrast, our softmax parameterization preserves the original scale of the learning rate thanks to its convex combination character. This point is important in conducting comparisons including our approach on existing methods which use scale-dependent optimizers, such as SGD or SGDW, but originally do not apply this normalization and alter the effective learning rate when introducing auxiliary losses by setting $\sum_{i=0}^{K} \lambda_i \neq 1$.

### 3.4 Regularization

Our treatment of loss HPs as regular parameters optimized via standard GD updates necessitates the application of regularization to these HPs, similarly to what is standard practice for regular parameters. However, the difference in the mathematical functionality of our HP exponents $\boldsymbol{\mu}$ and of regular parameters $\mathbf{w}$ of the learned network mapping which typically reside in a standard Euclidean space dictates a differentiation in their respective regularization. In the latter case, weights in $\mathbf{w}$ typically multiply features that assume arbitrary real values, so a regularity prior consists in small magnitudes and is typically pursued via a squared $L_2$ penalty which is equivalent to a *weight decay* towards $\mathbf{0}$ (Loshchilov & Hutter, 2019).

By contrast, our exponents $\mu_i$ are passed through a softmax to multiply positive loss values. Thus, the resulting softmax vector can be viewed as a discrete probability distribution over the various loss terms $l_i$, the data-related bias of which towards any individual loss can be regularized via the *negated entropy* of the distribution that favors "simpler" distributions close to uniform. Moreover, to regularize the *absolute* scale of the exponents $\mu_i$, to which negated entropy is invariant, we include a *softplus* term for each learnable $\mu_i$ to our regularization loss $L_r$, so that the latter becomes

$$L_r(\boldsymbol{\mu}) = \rho \left( \sum_{i=0}^{K} \frac{\exp(\mu_i)}{\sum_{j=0}^{K} \exp(\mu_j)} \log \left( \frac{\exp(\mu_i)}{\sum_{j=0}^{K} \exp(\mu_j)} \right) + \sum_{i=1}^{K} \log\left(1 + \exp(\mu_i)\right) \right), \tag{7}$$

where $\rho > 0$ is the single non-learnable *hyperparameter decay* which our method introduces. In particular, $\rho$ is shared by all loss HPs, akin to the standard *weight decay* $\lambda$ (Loshchilov & Hutter, 2019) which serves as a single non-learnable HP shared by all weights $\mathbf{w}$ of the model. In this way, we substitute the selection of multiple loss-specific HPs $\mu_i$ with a single generic HP decay $\rho$ having a range of values that is common across multiple diverse learned models and needing minimal tuning, similar to the standard treatment of weight decay of regular parameters in neural networks. Moreover, we keep the same weight decay or other regularization for $\mathbf{w}$ as the original optimization algorithm on which YOTO is implemented in each case, as equation 7 only pertains to $\boldsymbol{\mu}$.

For the gradients of our regularizer w.r.t. the HPs, $\partial L_r / \partial \mu_i$, we prove in Appendix B that

$$\frac{1}{\rho} \frac{\partial L_r}{\partial \mu_i} = \frac{\exp(\mu_i) \left( \sum_{j=0}^{K} \exp(\mu_j)(\mu_i - \mu_j) \right)}{\left( \sum_{j=0}^{K} \exp(\mu_j) \right)^2} + \frac{\exp(\mu_i)}{1 + \exp(\mu_i)}, \; i \in \{1, \ldots, K\}. \tag{8}$$

### 3.5 Initialization and Overall YOTO Algorithm

While Sec. 3.3 and 3.4 detail the gradients for updating loss HPs within a given iteration of gradient-based optimization, how to initialize the exponents $\mu_i$, $i \in \{1, \ldots, K\}$, is also of central importance. The strategy that we follow in practice in our experiments is to initialize these $\mu_i$, which correspond to auxiliary empirical loss terms, uniformly with the same value $\mu_{i,0} \leftarrow \log(\epsilon)$, where $\epsilon$ is a small

**Algorithm 1** YOTO with SGDW with momentum

**given** initial learning rate $\alpha \in \mathbb{R}$, momentum factor $\beta_1 \in \mathbb{R}$, weight decay $\lambda > 0$, hyperparameter decay $\rho > 0$, initialization parameter $0 < \epsilon < 1$
**initialize** time step $t \leftarrow 0$, parameter vector $\mathbf{w}_{t=0} \in \mathbb{R}^n$, loss HP exponent vector $\boldsymbol{\mu}_{t=0} \leftarrow (0, \log(\epsilon), \ldots, \log(\epsilon)) \in \mathbb{R}^{K+1}$, first moment vector for parameters $\mathbf{m}_{t=0} \leftarrow \mathbf{0}$, first moment vector for HPs $\mathbf{n}_{t=0} \leftarrow \mathbf{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
**repeat**
  $t \leftarrow t + 1$
  $\nabla L_{e,t}(\mathbf{w}_{t-1}), \nabla L_{e,t}(\boldsymbol{\mu}_{t-1}) \leftarrow \text{SelectBatch}(\mathbf{w}_{t-1}, \boldsymbol{\mu}_{t-1})$ ▷ compute empirical gradient, using i.a. equation 6
  $\mathbf{g}_t \leftarrow \nabla L_{e,t}(\mathbf{w}_{t-1})$
  $\mathbf{h}_t \leftarrow \nabla L_{e,t}(\boldsymbol{\mu}_{t-1})$
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$
  $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + \eta_t \alpha \mathbf{g}_t$
  $\mathbf{n}_t \leftarrow \beta_1 \mathbf{n}_{t-1} + \eta_t \alpha \mathbf{h}_t$
  $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} - \mathbf{m}_t - \eta_t \alpha \lambda \mathbf{w}_{t-1}$
  $\boldsymbol{\mu}_t \leftarrow \boldsymbol{\mu}_{t-1} - \mathbf{n}_t - \eta_t \alpha \rho \nabla L_{r,t}(\boldsymbol{\mu}_{t-1})$ ▷ include gradient of regularizer in update using equation 8
**until** *stopping criterion is met*
**return** optimized parameters $\mathbf{w}_t$, optimized loss HPs $\boldsymbol{\mu}_t$

positive number. Combined with the fact that $\mu_{0,t} = 0$ for the basic empirical loss $l_0$, this implies that we start optimization with weak overall contributions $\lambda_{i,0} l_i$ for the auxiliary empirical losses, i.e. for $i \in \{1, \ldots, K\}$, and let the model potentially increase these contributions automatically in the case where such an increase is also beneficial for optimizing the more heavily weighted basic loss contribution, $\lambda_0 l_0$, or keep the former contributions at a low value throughout the optimization. In both cases, the model *selects automatically* which losses are useful for optimization and to what degree. As we evidence in Sec. 4, the optimal $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ values after convergence of a certain network are rather stable across a wide range of initializations, hinting on the insensitivity of YOTO to the particular initialization. A full instance of our YOTO algorithm when combined with SGDW (Loshchilov & Hutter, 2019) is presented in Algorithm 1; note that YOTO can be combined with any optimizer.

## 4 Experiments

We select the key network optimization area of computer vision for the validation of our method, i.e. the same area on which previous seminal optimization works such as AdamW (Loshchilov & Hutter, 2019) have been validated. AdamW considered for experimental validation only a single vision task and a single network architecture, i.e. image classification and ResNet (He et al., 2016) CNNs. We instead consider a more diverse set of vision tasks and network architectures to validate YOTO. In particular, we select (i) the central *regression* task of monocular depth and 3D estimation and (ii) the key *classification* task of semantic segmentation. Moreover, our experimental comparisons are performed both on *transformer* networks, as we use a Vision Transformer (ViT) (Dosovitskiy et al., 2021) architecture for task (i), and on *CNNs*, as we use a ResNet (He et al., 2016) architecture for task (ii).

### 4.1 Regression: Depth and 3D Estimation

We first apply YOTO to the state-of-the-art monocular metric depth and 3D estimation method of UniDepth (Piccinelli et al., 2024), which uses a ViT backbone (Dosovitskiy et al., 2021), for optimizing its loss HPs. In particular, UniDepth originally optimizes a composite empirical loss $L_e$ based on its metric dense 3D predictions $(\mathbf{Z}_{\text{log}}, \boldsymbol{\Theta}, \boldsymbol{\Phi})$ and the respective ground-truth maps $(\mathbf{Z}^*_{\text{log}}, \boldsymbol{\Theta}^*, \boldsymbol{\Phi}^*)$, computed as

$$L_e = \lambda_0 \text{SILog}(\mathbf{Z}_{\text{log}}, \mathbf{Z}^*_{\text{log}}) + \lambda_1 \text{MSE}\left((\boldsymbol{\Theta}, \boldsymbol{\Phi}), (\boldsymbol{\Theta}^*, \boldsymbol{\Phi}^*)\right) + \lambda_2 l_{\text{con}}, \tag{9}$$

where SILog is the "scale-invariant" loss in logarithmic space used widely in depth estimation (Eigen et al., 2014), MSE is the standard mean-squared-error loss introduced in UniDepth for estimation of dense camera intrinsics, and $l_{\text{con}}$ is the geometric invariance loss utilized in UniDepth for consistency of internal geometric network features to geometric augmentations. The official implementation of UniDepth sets $\lambda_0 = 1$, $\lambda_1 = 0.25$, and $\lambda_2 = 0.1$. Moreover, the initial learning rate is $\alpha = 10^{-4}$.

Table 1: Comparison of YOTO vs. grid search for optimizing UniDepth (Piccinelli et al., 2024) and its two independent loss HPs for monocular metric depth and 3D estimation. "Grid": grid search over two dimensions to identify optimal loss HP values. Performance metrics and HPs $(\exp(\mu_1), \exp(\mu_2))$ are reported for all models at completion of training. A.Rel: absolute relative depth error (%), RMS: root mean square depth error (in $m$), $\mathrm{RMS}_{\log}$: log variant of RMS, $\delta_1$: percentage of depth inlier pixels for threshold ratio 1.25, $\mathrm{SI}_{\log}$: scale-invariant depth error in log-scale, CD: metric 3D Chamfer distance, $\mathrm{F}_A$: area under the curve of 3D $\mathrm{F}_1$-score (Örnek et al., 2022) up to $1/20$ of the datasets' maximum depth, $\downarrow$: lower is better, $\uparrow$: higher is better. "Mean" metrics are averaged over the two benchmarks, which is only possible for scale-independent metrics: A.Rel, $\delta_1$, $\mathrm{SI}_{\log}$, and $\mathrm{F}_A$.

| Method | $(\exp(\mu_1), \exp(\mu_2))$ | nuScenes (Caesar et al., 2020) | | | | | | | SUN-RGBD (Song et al., 2015) | | | | | | | Mean | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A.Rel ↓ | RMS ↓ | $\mathrm{RMS}_{\log}$ ↓ | $\delta_1$ ↑ | $\mathrm{SI}_{\log}$ ↓ | CD ↓ | $\mathrm{F}_A$ ↑ | A.Rel ↓ | RMS ↓ | $\mathrm{RMS}_{\log}$ ↓ | $\delta_1$ ↑ | $\mathrm{SI}_{\log}$ ↓ | CD ↓ | $\mathrm{F}_A$ ↑ | A.Rel ↓ | $\delta_1$ ↑ | $\mathrm{SI}_{\log}$ ↓ | $\mathrm{F}_A$ ↑ |
| Grid | (0.001, 0.01) | 32.5 | 6.36 | 0.338 | 48.9 | 24.96 | 1.45 | 43.0 | 11.0 | 0.285 | 0.127 | 92.2 | 8.16 | 0.134 | 76.1 | 21.8 | 70.6 | 16.56 | 59.6 |
| Grid | (0.001, 0.1) | 31.7 | 6.38 | 0.334 | 50.3 | 25.03 | 1.41 | 44.7 | 11.1 | 0.287 | 0.128 | 92.1 | 8.16 | 0.134 | 76.2 | 21.4 | 71.2 | 16.60 | 60.5 |
| Grid | (0.001, 1) | 33.5 | 6.35 | 0.344 | 45.1 | 25.30 | 1.47 | 41.3 | 11.2 | 0.287 | 0.128 | 92.1 | 8.13 | 0.136 | 75.6 | 22.4 | 68.6 | 16.72 | 58.5 |
| Grid | (0.01, 0.01) | 34.5 | 6.43 | 0.350 | 43.8 | 25.32 | 1.49 | 41.1 | 11.3 | 0.289 | 0.130 | 91.5 | 8.24 | 0.136 | 75.7 | 22.9 | 67.7 | 16.78 | 58.4 |
| Grid | (0.01, 0.1) | 33.8 | 6.44 | 0.346 | 45.9 | 25.25 | 1.47 | 42.1 | 11.1 | 0.286 | 0.127 | 92.1 | 8.17 | 0.134 | 76.1 | 22.5 | 69.0 | 16.71 | 59.1 |
| Grid | (0.01, 1) | 33.2 | 6.36 | 0.344 | 44.8 | 25.86 | 1.46 | 41.5 | 10.7 | 0.284 | 0.125 | 92.8 | 8.20 | 0.134 | 76.3 | 22.0 | 68.8 | 17.03 | 58.9 |
| Grid | (0.25, 0.01) | 34.8 | 6.53 | 0.351 | 45.5 | 25.23 | 1.47 | 41.1 | 11.2 | 0.290 | 0.129 | 92.0 | 8.12 | 0.135 | 76.1 | 23.0 | 68.8 | 16.68 | 58.6 |
| Grid | (0.25, 0.1)–orig | 33.0 | 6.48 | 0.341 | 48.5 | 25.19 | 1.44 | 43.1 | 11.1 | 0.286 | 0.128 | 92.2 | 8.15 | 0.133 | 76.4 | 22.1 | 70.4 | 16.67 | 59.8 |
| Grid | (0.25, 1) | 34.2 | 6.50 | 0.350 | 45.9 | 25.82 | 1.45 | 42.2 | 11.0 | 0.286 | 0.127 | 92.2 | 8.17 | 0.134 | 76.2 | 22.6 | 69.1 | 17.00 | 59.2 |
| Grid | (1, 0.01) | 35.8 | 6.63 | 0.358 | 44.4 | 25.36 | 1.49 | 40.7 | 10.9 | 0.286 | 0.127 | 92.6 | **8.09** | 0.133 | 76.6 | 23.4 | 68.5 | 16.73 | 58.7 |
| Grid | (1, 0.1) | 35.2 | 6.69 | 0.354 | 47.1 | 25.09 | 1.49 | 41.8 | 11.0 | 0.288 | 0.128 | 92.6 | 8.13 | 0.133 | 76.5 | 23.1 | 69.9 | 16.61 | 59.2 |
| Grid | (1, 1) | 31.8 | **6.24** | 0.334 | 48.4 | 25.51 | 1.39 | 43.4 | 11.0 | 0.287 | 0.128 | 92.3 | 8.17 | 0.133 | 76.5 | 21.4 | 70.4 | 16.84 | 60.0 |
| Grid | (10, 0.01) | 37.6 | 6.75 | 0.366 | 42.6 | 25.18 | 1.51 | 38.6 | 10.8 | 0.284 | 0.126 | 92.6 | 8.10 | 0.132 | 76.7 | 24.2 | 67.6 | 16.64 | 57.7 |
| Grid | (10, 0.1) | 36.5 | 6.66 | 0.361 | 44.7 | 25.40 | 1.46 | 40.4 | 11.0 | 0.287 | 0.128 | 92.5 | 8.18 | 0.132 | 76.6 | 23.8 | 68.6 | 16.79 | 58.5 |
| Grid | (10, 1) | 32.8 | 6.56 | 0.338 | **52.6** | **24.61** | 1.40 | 44.1 | **10.6** | **0.282** | **0.124** | **93.0** | **8.09** | **0.131** | **77.1** | 21.7 | **72.8** | **16.35** | 60.6 |
| YOTO | (0.04678, 0.04671) | **30.7** | 6.28 | **0.327** | 51.8 | 24.76 | **1.36** | **45.4** | 11.0 | 0.283 | 0.127 | 92.2 | 8.14 | 0.132 | 76.7 | **20.9** | 72.0 | 16.45 | **61.0** |

The AdamW optimizer (Loshchilov & Hutter, 2019) is employed by UniDepth, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay $\lambda = 0.1$. We recognize that this loss formulation involves one basic empirical loss term, i.e. SILog, and two auxiliary terms, i.e. MSE and $l_{\text{con}}$. We compare in Table 1:

1. a $5 \times 3$ *grid search* in log space involving *15 distinct training runs* across normalized, non-learnable loss HPs $(\lambda_0, \lambda_1, \lambda_2) = (\exp(\mu_0), \exp(\mu_1), \exp(\mu_2))/\sum_{i=0}^{2} \exp(\mu_i)$, where $(\exp(\mu_0), \exp(\mu_1), \exp(\mu_2)) \in \{1\} \times \{0.001, 0.01, 0.25, 1, 10\} \times \{0.01, 0.1, 1\}$, i.e. the search includes i.a. the original–"orig" loss HP combination of UniDepth $(1, 0.25, 0.1)$, and
2. YOTO, involving *a single training run* that automatically optimizes the loss HPs jointly with UniDepth's regular weights. In this setting, YOTO is implemented on AdamW, analogously to Algorithm 1, using $\rho = 20$ and $\epsilon = 0.1$ and otherwise keeping exactly the original UniDepth optimization settings mentioned above, identically to the grid search runs.

Moreover, across the comparison of Table 1, the training set is composed of the ScanNet (Dai et al., 2017), Argoverse2 (Wilson et al., 2021), and Waymo (Sun et al., 2020) sets—which were all originally used by Piccinelli et al. (2024) too—for a total of ca. 700K images, and we train the same ViT-L (Dosovitskiy et al., 2021) backbone as Piccinelli et al. (2024) originally did on mini-batches of size 64 for 300K iterations. This results in each training run of both YOTO and grid search taking 4-5 days on 4 RTX4090s. We evaluate all models zero-shot on two datasets unseen during training: the outdoor nuScenes (Caesar et al., 2020) with 36,114 images and the indoor SUN-RGBD (Song et al., 2015) with 4,396 images.

YOTO Pareto-dominates the grid search in terms of performance and wallclock training time: the YOTO UniDepth model, *trained in one shot*, *outperforms all 15 grid-search models* across the two benchmarks on the majority of the seven diverse depth and 3D metrics, *at 6.7% of the grid search's total wallclock time*. The superiority of YOTO in these evaluation metrics, which are not explicitly optimized as losses at training but are rather only used at testing, crucially implies that optimizing HPs on training-set losses with YOTO improves *testing* performance as well. Note that we did not experiment at all with tuning $\rho$ and $\epsilon$ for YOTO in this main comparison for depth and 3D estimation. We rather *trained only once* using the specified $\rho = 20$ and $\epsilon = 0.1$, even though the entire YOTO optimization algorithm is new and no suitable parameter ranges were previously known.

We evidence the *robustness to initialization* of our algorithm by conducting a sensitivity analysis w.r.t. initialization of the HPs $\mu_i, i = 1, 2$, in Figure 1. In particular, we train 25 different YOTO models for UniDepth, by initializing $\mu_1$ and $\mu_2$ differently and varying their respective initializations, $\log(\epsilon_1)$ and $\log(\epsilon_2)$, across a regular grid in log space from $10^{-3}$ to 10, while keeping hyperparameter decay fixed at $\rho = 2$. Note that we use a different value for $\rho$ than in Table 1's comparison, which ensures that we indeed only train YOTO once with the default $\rho = 20$ in the main comparison, utilizing a different configuration for the present analysis that requires multiple training runs. We create a phase portrait by plotting the $(\mu_1(t), \mu_2(t))$ trajectories of all 25 models. Despite the broad initialization

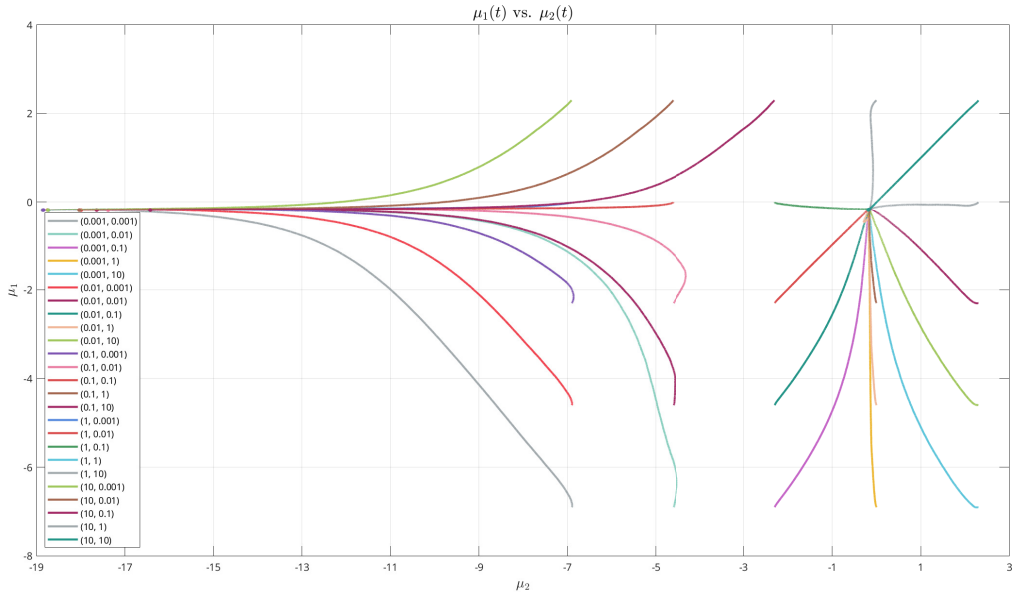


Figure 1: Phase portrait of YOTO training trajectories in the log-space of UniDepth's independent HPs, $\mu_1$ and $\mu_2$, demonstrating the insensitivity of YOTO to initialization. We vary the initialization parameter $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2)$ across a $5 \times 5$ grid of values while keeping hyperparameter decay fixed at $\rho = 2$, and plot the $(\mu_1(t), \mu_2(t))$ trajectories of HPs throughout model optimization with YOTO. Different trajectories are referred in the legend by their respective $(\epsilon_1, \epsilon_2)$ values which initialize $(\exp(\mu_1), \exp(\mu_2))$. Final points of trajectories are marked as dots. Best viewed on a screen and zoomed in.

spectrum in Figure 1, all 25 models converge to either of *only two points* or attractors in the HP space. The one point attractor is at $(\mu_1, \mu_2) \approx (-0.18, -0.18)$, with both auxiliary losses contributing non-negligibly at convergence. The other point attractor also has $\mu_1 \approx -0.18$, but for it $\exp(\mu_2) \approx 0$. In essence, models that converge to the latter point attractor have automatically selected only the camera MSE loss weighted via $\mu_1$ as a helpful auxiliary loss, and have "rejected" the invariance $l_{\text{con}}$ loss weighted via $\mu_2$, the contribution of which is negligible to them at convergence. This automatic loss rejection is broadly useful as it performs model selection. In addition, we have found that models which converge to the former point attractor have better performance on SUN-RGBD than models which converge to the latter point attractor, and vice versa for nuScenes. This implies that the two point attractors represent two diverse local optima in the HP space and the regular weight space. This finding supports the invariance of YOTO to initialization, as the weight space of a given large neural network is known (Choromanska et al., 2015) to exhibit different local optima of similar high quality, to which optimization may converge for different configurations, e.g. different initialization.

### 4.2 Classification: Semantic Segmentation

The second, classification-level vision task we consider is semantic segmentation, using the state-of-the-art domain-generalizing method of CISS (Sakaridis et al., 2025). In particular, we use the DeepLabv2 (Chen et al., 2018) version of CISS based on a ResNet backbone (He et al., 2016) and its basic formulation with losses computed only from the source domain, referred to as CISS-source. In particular, CISS-source originally optimizes a composite empirical loss $L_e$ for an encoder-decoder segmentation network $f = \omega \circ \phi$, where $\phi$ is the encoder and $\omega$ the decoder. Based on: (i) a pair of internal feature tensors that the network produces using a pair of images $I_s$ and $I_t$, i.e. $\phi(I_s)$ and $\phi(g(I_s, I_t))$ where $g$ is a non-learned mapping, (ii) the softmax output $f(I_s)$ for the first image $I_s$, and (iii) the respective ground-truth label map $Y_s$, we compute

$$L_e = \lambda_0 l_{\text{CE}}(f(I_s), Y_s) + \lambda_1 l_{\text{inv}}(\phi(I_s), \phi(g(I_s, I_t))), \tag{10}$$

where $l_{\text{CE}}$ is the standard cross-entropy loss which is used in semantic segmentation (Long et al., 2015) and $l_{\text{inv}}$ is a feature invariance loss that penalizes differences between corresponding elements of the two internal feature tensors. The original implementation of CISS sets $\lambda_0 = 1$, $\lambda_1 = 10$, the

constant learning rate to $\alpha = 2.5 \times 10^{-4}$, and uses the SGDW optimizer (Loshchilov & Hutter, 2019), with $\beta_1 = 0.9$ and weight decay $\lambda = 2$. The formulation of equation 10 includes one basic empirical loss term, i.e. $l_{\text{CE}}$, and one auxiliary term, i.e. $l_{\text{inv}}$. In the present comparisons, we keep the above optimization settings, but because SGDW is not scale-invariant, we always normalize the loss weights such that $\lambda_0 + \lambda_1 = 1$. We compare in Figure 2: (i) a grid search through 13 normalized, non-learnable loss HPs $(\lambda_0, \lambda_1)$, where the 1D grid is in the space of the log ratio of the two HPs, $\log_{10}(\lambda_1/\lambda_0)$, and includes the original CISS $\log_{10}$ ratio of 1, (ii) a random search with 5 samples of this log ratio drawn from a uniform distribution in the interval $[-2, 2]$, and (iii) the YOTO optimization of the loss HPs combined with SGDW based on Algorithm 1, setting $\epsilon = \exp(-4)$ and varying $\rho \in \{20, 50, 200, 500\}$.

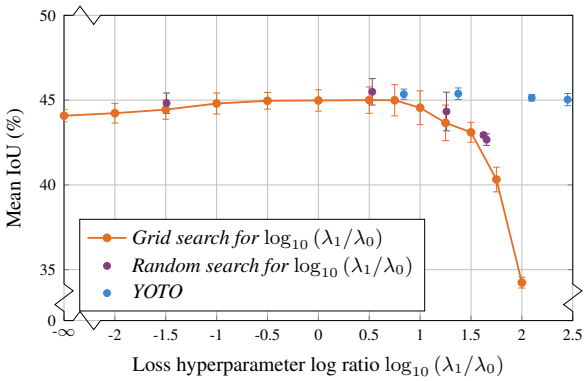


Figure 2: Comparison of YOTO vs. grid search and random search for optimizing CISS-source (Sakaridis et al., 2025) and its loss HPs for domain-generalizing semantic segmentation. Means and standard deviations (1-$\sigma$) of performance ($y$-axis) are plotted for each examined configuration. The means and standard deviations of the log ratios of optimized HPs with YOTO ($x$-axis) are also plotted, but the latter standard deviations are very small, hence imperceptible in the plot.

This range of values for the HP decay $\rho$ intentionally overlaps with that for 3D estimation, to verify that $\rho$ needs minimal tuning across diverse networks and tasks, similarly to the weight decay $\lambda$ of regular network parameters. We train 3 models with different random seeds for every point of the grid and random searches and for each $\rho$ in YOTO. The training set comprises the standard benchmark of Cityscapes→ACDC (Cordts et al., 2016; Sakaridis et al., 2021) for normal-to-adverse domain generalization, where only Cityscapes labels are used. We train with mini-batches of size 2, i.e. one Cityscapes image $I_s$ and one ACDC image $I_t$, for 148,750 iterations, which takes ca. 2 days on an RTX4090 for each training run. Validation is performed on the held-out validation set of ACDC, following Sakaridis et al. (2025) and using standard mean Intersection over Union (IoU) (Long et al., 2015). YOTO again dominates the grid search, consistently outperforming all 13 grid-search models for all values of $\rho$. As $\rho$ increases for YOTO, performance remains stable and above that of grid search, while the free HP $\mu_1$ decreases evenly, showcasing the *insensitivity of YOTO to the precise value of* $\rho$. YOTO is also on a par w.r.t. performance with the best HP identified by the random search and outperforms all four others. Moreover, YOTO exhibits far better robustness to stochasticity, as performance variation at each $\rho$ across training runs is much lower than that of grid- and random-search models and YOTO reliably converges to the same point in HP space with negligible variance. The optimization trajectories for $\rho = 200$ of the free HP $\mu_1(t)$ (cf. Appendix C) are also virtually identical across the 3 runs. These findings evidence the stability of YOTO.

**Discussion.** While YOTO bypasses the need to manually tune loss HPs, in doing so it introduces two new HPs: the hyperparameter decay $\rho$ which weighs the regularization loss for all $\mu_i$, and the initialization parameter $\epsilon$. Resp. $\epsilon$, we have firmly evidenced that our algorithm is insensitive to it in terms of performance and final loss HP values. Resp. $\rho$, we have shown that akin to weight decay $\lambda$, it has a standard range, i.e. $[20, 200]$, that works well across settings, limiting the effort to tune it.

## 5 Conclusion

We have presented YOTO, an automatic, gradient-based optimization algorithm that tunes loss weight HPs and regular parameters of learned models jointly in one shot via standard GD. The joint optimization in YOTO allows the model to explore a richer parametric space than with existing HPO approaches, leading to consistent gains in final testing performance besides its obvious gains in efficiency due to its one-shot regime. We view YOTO as a first yet firm step in the direction of directly optimizing HPs on empirical losses, which has wide beneficial implications on the practical experimental optimization of the majority of learned parametric models and opens new avenues for effective and efficient optimization of other types of HPs through standard GD.

## REFERENCES


CVPR 2024 open access repository. https://openaccess.thecvf.com/CVPR2024?day=all, 2024. Accessed: 2025-05-15. 2

Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. 2

Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2019. 3

Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural Computation*, 12(8): 1889–1900, 2000. 1, 2

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, 2011. 2

Jorg Bornschein, Francesco Visin, and Simon Osindero. Small data, big decisions: Model selection in the small-data regime. In *International Conference on Machine Learning*, 2020. 3

Léon Bottou. Online learning and stochastic approximations. In David Saad (ed.), *Online Learning in Neural Networks*, pp. 9–42. Cambridge University Press, 1998. 1

George EP Box and Kenneth B Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13(1):1–38, 1951. 2

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 7

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, 2020. 2

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4): 834–848, 2018. 8

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gerard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2015. 8

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. 2

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 9

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 7

Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, and Joseph E. Gonzalez. FBNetV3: Joint architecture-recipe search using predictor pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four GPU hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

Xuanyi Dong, Mingxing Tan, Adams Wei Yu, Daiyi Peng, Bogdan Gabrys, and Quoc V Le. AutoHAS: Efficient hyperparameter and architecture search. In *2nd Workshop on Neural Architecture Search at International Conference on Learning Representations (ICLR)*, 2021. 3

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 6, 7

David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, 2014. 6

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019. 3

Michael TM Emmerich, André H Deutz, and Jan Willem Klinkenberg. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *IEEE Congress of Evolutionary Computation (CEC)*, 2011. 3

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, 2018. 2, 3

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. 2

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, 2017. 2

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, 2018. 2

Luca Franceschi, Michele Donini, Valerio Perrone, Aaron Klein, Cédric Archambeau, Matthias Seeger, Massimiliano Pontil, and Paolo Frasconi. Hyperparameter optimization in machine learning. *CoRR*, abs/2410.22854, 2024. 2

Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. Bayesian optimization with inequality constraints. In *International Conference on Machine Learning*, 2014. 3

Michael A Gelbart, Jasper Snoek, and Ryan P Adams. Bayesian optimization with unknown constraints. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2014. 3

Ross Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, 2015. 2

Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, 2020. 2

Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996. 2

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6, 8

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

Daniel Hernández-Lobato, Jose Hernández-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective Bayesian optimization. In *International Conference on Machine Learning*, 2016. 3

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, 2011. 2

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

Sergio Izquierdo, Julia Guerrero-Viu, Sven Hauns, Guilherme Miotto, Simon Schrodi, André Biedenkapp, Thomas Elsken, Difan Deng, Marius Lindauer, and Frank Hutter. Bag of baselines for multi-objective joint neural architecture search and hyperparameter optimization. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021. 3

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *CoRR*, abs/1711.09846, 2017. 2

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, 2016. 3

Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, 2013. 3

Andy J Keane. Statistical improvement criteria for use in multiobjective design optimization. *AIAA*, 44(4):879–891, 2006. 3

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4): 1–14, 2023. 2

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial Intelligence and Statistics*, 2017. 2, 3

Joshua Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1): 50–66, 2006. 3

Jan Larsen, Claus Svarer, Lars Nonboe Andersen, and Lars Kai Hansen. Adaptive regularization in neural network modeling. In *Neural Networks: Tricks of the Trade*, pp. 113–132. Springer, 1998. 2

Yann LeCun, Léon Bottou, Geneviève B. Orr, and Klaus-Robert Müller. Efficient backprop. In Geneviève B. Orr and Klaus-Robert Müller (eds.), *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pp. 9–50. Springer, 1998. 1

Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. Constrained Bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019. 3

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018a. 3

Lisha Li, Kevin Jamieson, Afshin Rostamizadeh, Katya Gonina, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. *CoRR*, abs/1810.05934, 2018b. 3

Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, KiJung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard Zemel. Reviving and improving recurrent back-propagation. In *International Conference on Machine Learning*, 2018. 2

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. 1, 3

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 8, 9

Pablo Ribalta Lorenzo, Jakub Nalepa, Michal Kawulok, Luciano Sanchez Ramos, and José Ranilla Pastor. Particle swarm optimization for hyper-parameter selection in deep neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017. 2

Ilya Loshchilov and Frank Hutter. CMA-ES for hyperparameter optimization of deep neural networks. *CoRR*, abs/1604.07269, 2016. 2

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 2, 5, 6, 7, 9

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, 2015. 1, 2

Evin Pınar Örnek, Shristi Mudgal, Johanna Wald, Yida Wang, Nassir Navab, and Federico Tombari. From 2D to 3D: Re-thinking benchmarking of monocular depth prediction. *arXiv preprint arXiv:2203.08122*, 2022. 7

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning*, 2016. 1, 2

Valerio Perrone, Iaroslav Shcherbatyi, Rodolphe Jenatton, Cedric Archambeau, and Matthias Seeger. Constrained Bayesian optimization with max-value entropy search. *arXiv preprint arXiv:1910.07003*, 2019. 3

Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. UniDepth: Universal monocular metric depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 6, 7

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, 2019. 1

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Comput. Surv.*, 54(4), May 2021. 3

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. 2

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 1

Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: The Adverse Conditions Dataset with Correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 9

Christos Sakaridis, David Bruggemann, Fisher Yu, and Luc Van Gool. Condition-invariant semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(4):3111–3125, 2025. 8, 9, 16, 17

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. 2

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012. 2

Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 7

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo Open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 7

Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2013. 3

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw Bayesian optimization. *CoRR*, abs/1406.3896, 2014. 3

Zhiqiang Tao, Yaliang Li, Bolin Ding, Ce Zhang, Jingren Zhou, and Yun Fu. Learning to mutate with hypergradient guided population. In *Advances in Neural Information Processing Systems*, 2020. 2, 3

Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Advances in Neural Information Processing Systems*, 2021. 7

Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. NAS evaluation is frustratingly hard. In *International Conference on Learning Representations*, 2020. 3

Arber Zela, Aaron Klein, Stefan Falkner, and Frank Hutter. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. *CoRR*, abs/1807.06906, 2018. 3

Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020. 3

Richard Zhang and Daniel Golovin. Random hypervolume scalarizations for provable multi-objective black box optimization. In *International Conference on Machine Learning*, 2020. 3

Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. EcoNAS: Finding proxies for economical neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. 1, 3

## A Derivation of Composite Empirical Loss Gradients

We derive the expression which is provided in equation 6 for the gradients of our composite empirical loss $L_e$ with respect to the hyperparameters $\mu_i$.

*Proof.* Starting from equation 5, we apply the quotient rule to get

$$\frac{\partial L_e}{\partial \mu_i} = \frac{\frac{\partial}{\partial \mu_i}\left(\sum_{j=0}^{K}\exp(\mu_j)l_j(f(\mathbf{w}))\right)\sum_{j=0}^{K}\exp(\mu_j) - \left(\sum_{j=0}^{K}\exp(\mu_j)l_j(f(\mathbf{w}))\right)\frac{\partial}{\partial \mu_i}\left(\sum_{j=0}^{K}\exp(\mu_j)\right)}{\left(\sum_{j=0}^{K}\exp(\mu_j)\right)^2}. \tag{11}$$

By comparison of equation 11 with equation 6, we observe that the denominators of the RHSs are identical. Thus, we simply proceed to show the equality of the respective numerators. In particular, we expand the derivatives in the numerator on the RHS of equation 11 as

$$\exp(\mu_i)l_i(f(\mathbf{w}))\left(\sum_{j=0}^{K}\exp(\mu_j)\right) - \exp(\mu_i)\left(\sum_{j=0}^{K}\exp(\mu_j)l_j(f(\mathbf{w}))\right). \tag{12}$$

Next, we factorize equation 12 as

$$\exp(\mu_i)\left(\sum_{j=0}^{K}\left(\left(l_i(f(\mathbf{w})) - l_j(f(\mathbf{w}))\right)\exp(\mu_j)\right)\right). \tag{13}$$

Finally, the proof is completed by eliminating the zero term in the sum of equation 13 for $j = i$ as

$$\exp(\mu_i)\left(\sum_{\substack{j=0\\ j\neq i}}^{K}\left(\left(l_i(f(\mathbf{w})) - l_j(f(\mathbf{w}))\right)\exp(\mu_j)\right)\right). \tag{14}$$

□

## B Derivation of Regularization Loss Gradients

We derive the expression which is provided in equation 8 for the gradients of our regularization loss $L_r$ with respect to the hyperparameters $\mu_i$.

*Proof.* Starting from equation 7, we first note that the gradient of the second, softplus term of the RHS is trivial to obtain via the chain rule, so we need to prove that the gradient of the first, negated entropy term on the RHS of equation 7 is equal to the first term on the RHS of equation 8. We leverage the linearity of the gradient operator to exchange it with the outer sum of the negated entropy term of equation 7, as well as the product rule over each term of that sum to obtain the following gradient:

$$\sum_{j=0}^{K}\left(\frac{\partial}{\partial \mu_i}\left(\frac{\exp(\mu_j)}{\sum_{k=0}^{K}\exp(\mu_k)}\right)\log\left(\frac{\exp(\mu_j)}{\sum_{k=0}^{K}\exp(\mu_k)}\right) + \frac{\exp(\mu_j)}{\sum_{k=0}^{K}\exp(\mu_k)}\frac{\partial}{\partial \mu_i}\left(\log\left(\frac{\exp(\mu_j)}{\sum_{k=0}^{K}\exp(\mu_k)}\right)\right)\right) \tag{15}$$

We then break the sum into its $i$-th term and the rest $K$ terms for which $j \neq i$. By calculating away the gradients in equation 15 and performing eliminations, the $i$-th term of the sum becomes

$$\frac{\exp(\mu_i)\left(\sum_{k=0}^{K}\exp(\mu_k)-\exp(\mu_i)\right)}{\left(\sum_{k=0}^{K}\exp(\mu_k)\right)^2}\left(1+\log\left(\frac{\exp(\mu_i)}{\sum_{k=0}^{K}\exp(\mu_k)}\right)\right). \tag{16}$$

The rest of the terms of the aforementioned sum from equation 15, after respective calculations of the gradients and eliminations, become

$$\frac{\sum_{\substack{j=0\\j\neq i}}^{K}\left(-\exp(\mu_i)\exp(\mu_j)\left(1+\log\left(\frac{\exp(\mu_j)}{\sum_{k=0}^{K}\exp(\mu_k)}\right)\right)\right)}{\left(\sum_{k=0}^{K}\exp(\mu_k)\right)^2}. \tag{17}$$

We observe that the denominators of both addends in equation 16 and equation 17 are identical to that of the first term on the RHS of equation 8. Thus, we are left to prove that the sum of the numerators of the two aforementioned addends is equal to the numerator of the first term on the RHS of equation 8. With proper renaming of indices and grouping of sums, these two numerators sum to

$$\begin{aligned}
&\exp(\mu_i)\left(\sum_{j=0}^{K}\exp(\mu_j)\right)-\sum_{j=0}^{K}\exp(\mu_i)\exp(\mu_j)-\exp(\mu_i)\sum_{\substack{j=0\\j\neq i}}^{K}\exp(\mu_j)\log\left(\frac{\exp(\mu_j)}{\sum_{k=0}^{K}\exp(\mu_k)}\right)\\
&+\exp(\mu_i)\left(\sum_{j=0}^{K}\exp(\mu_j)-\exp(\mu_i)\right)\log\left(\frac{\exp(\mu_i)}{\sum_{k=0}^{K}\exp(\mu_k)}\right)\\
&=\exp(\mu_i)\left(\sum_{j=0}^{K}\exp(\mu_j)\left(\log\left(\frac{\exp(\mu_i)}{\sum_{k=0}^{K}\exp(\mu_k)}\right)-\log\left(\frac{\exp(\mu_j)}{\sum_{k=0}^{K}\exp(\mu_k)}\right)\right)\right)\\
&=\exp(\mu_i)\left(\sum_{j=0}^{K}\exp(\mu_j)(\mu_i-\mu_j)\right).
\end{aligned} \tag{18}$$

□

## C Stability of Optimization Trajectories of Hyperparameters Against Stochasticity in Training

We further evidence the robustness of YOTO to stochasticity, complementing the results of Figure 2 which pertain to training CISS (Sakaridis et al., 2025) with our method. In particular, beyond the above results showcasing the stable performance and hyperparameter convergence across 3 runs with different random seeds used for training, for $\rho = 200$ we show in Figure 3 the 3 optimization trajectories of the free hyperparameter, $\mu_1(t)$, for these 3 different runs. All 3 trajectories are virtually identical with each other, with a very minor variation between them in the "transition" phase between 20,000 and 40,000 steps of gradient descent, where $\mu_1$ grows faster. This further corroborates the insensitivity of YOTO to stochasticity in training, not only with regard to the converged models, but also with regard to the intrinsic training dynamics.

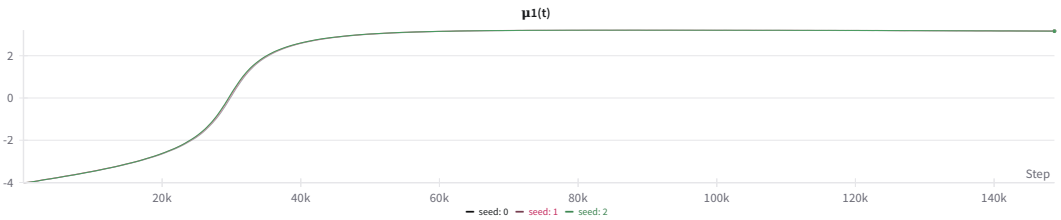


Figure 3: Analysis of sensitivity of YOTO to stochasticity in training, performed for training CISS (Sakaridis et al., 2025). The trajectory $\mu_1(t)$ of the hyperparameter $\mu_1$ over training steps $t$ is plotted for 3 different training runs. The 3 training runs are the same as those used to generate the result in Figure 2 for $\rho = 200$. Across these 3 runs, we vary the random seed in the set $\{0, 1, 2\}$. Different models are referred in the legend by their respective seed values. All three trajectories are virtually identical and can only be distinguished by zooming in at the interval between 20k and 40k training steps. Best viewed on a screen at full zoom.