International Journal of Computer Vision (2026) 134:8
https://doi.org/10.1007/s11263-025-02625-x

Investigating Uncertainty Weighting for Multi-Task Learning: Insights and Analytical Alternative

Lukas Kirchdorfer[1,2] · Tobias Sesterhenn[3] · Christian Bartelt[3] · Heiner Stuckenschmidt[1] · Lukas Schott[2] · Jan M. Köhler[2]

Received: 31 March 2025 / Accepted: 1 November 2025 / Published online: 23 December 2025
© The Author(s) 2025

Abstract
Multi-task learning (MTL) enables a single neural network to solve multiple tasks simultaneously, offering efficiency and improved generalization potential through shared representations. A central challenge in MTL is balancing task-specific losses during training to avoid performance degradation. While uncertainty-based loss weighting (UW) is a popular and competitive approach, we argue that it suffers from several limitations, including overfitting, rigid homoscedastic assumptions, and a lack of theoretical grounding for various loss functions. Therefore, we propose *Soft Optimal Uncertainty Weighting* (UW-SO), a novel loss weighting method that builds on UW by deriving analytically optimal weights and applying softmax normalization with adaptable temperature parameter, thereby alleviating several of the shortcomings of UW. Through extensive experiments across diverse datasets and architectures, we show that UW-SO achieves superior and robust performance compared to a variety of existing loss weighting methods. Additionally, we provide insights into the effects of temperature selection and propose measures to reduce computational demand.

Keywords Multi-Task Learning · Loss Weighting · Deep Learning · Computer Vision

1 Introduction

Multi-task learning (MTL) has emerged as a powerful paradigm in deep learning, enabling a single model to learn and perform multiple tasks simultaneously by leveraging shared representations across related objectives (Caruana, 1997; Ruder, 2017; Vandenhende, Georgoulis, Van Gansbeke, Proesmans, Dai, & Van Gool, 2021). This approach has gained increasing traction both in academic research and real-world applications, particularly in domains such as autonomous driving and robotics, where real-time inference and limited computational resources demand efficient, multi-functional neural networks (Ishihara, Kanervisto, Miura, and Hautamäki, 2021). By jointly training on several tasks, MTL offers the potential for improved generalization, resource efficiency, and knowledge transfer. However, despite its promise, MTL introduces new challenges—most notably, the difficulty of balancing task performance during training. Without careful coordination, some tasks may dominate or be neglected, resulting in suboptimal overall performance. While recent work has explored solutions ranging from architectural innovations (Hu & Singh, 2021; Heuer, Mantowsky, Bukhari, and Schneider, 2021) to gradient-based optimiza-

Communicated by Michael Möller.

Lukas Schott and Jan M. Köhler are Joint senior authors.

✉ Lukas Kirchdorfer
lkirchdo@mail.uni-mannheim.de

Tobias Sesterhenn
tobias.sesterhenn@tu-clausthal.de

Christian Bartelt
christian.bartelt@tu-clausthal.de

Heiner Stuckenschmidt
heiner.stuckenschmidt@uni-mannheim.de

Lukas Schott
lukas.schott@de.bosch.com

Jan M. Köhler
jan.koehler@de.bosch.com

1 University of Mannheim, Mannheim, Germany

2 Bosch Center for Artificial Intelligence, Renningen, Germany

3 Clausthal University of Technology, Clausthal, Germany

tion techniques (Liu, Liu, Jin, Stone, and Liu, 2021; Yu, Kumar, Gupta, Levine, Hausman, and Finn, 2020), one of the most straightforward yet impactful strategies is *loss weighting*, where task-specific contributions to the overall training objective are dynamically adjusted.

Despite the simplicity of assigning equal weights to all tasks—commonly referred to as *Equal Weighting* (EW)—this approach suffers from several critical shortcomings. In practice, different tasks often rely on distinct loss functions, such as L_1 loss for regression and cross-entropy loss for classification, each with inherently different scales. Moreover, task difficulty, data noise, and prediction uncertainty can vary substantially across tasks, even when the same loss function is used, resulting in imbalanced loss magnitudes. These discrepancies make it necessary to dynamically adjust task weights during training. As a result, numerous methods have been proposed to address this issue through adaptive loss weighting schemes (Liu, Li, Kuang, Xue, Chen, Yang, Liao, and Zhang, 2021; Chen, Ngiam, Huang, Luong, Kretzschmar, Chai, and Anguelov, 2020; Javaloy & Valera, 2022; Yu, Kumar, Gupta, Levine, Hausman, and Finn, 2020; Liu, Liu, Jin, Stone, and Liu, 2021; Kendall, Gal, and Cipolla, 2018; Baijiong, Feiyang, and Yu, 2021; Chennupati, Sistu, Yogamani, and A Rawashdeh, 2019; Liu, Johns, and Davison, 2019; Chen, Badrinarayanan, Lee, and Rabinovich, 2018). Among them, the method of *Uncertainty Weighting* (UW) by Kendall et al. (2018) has gained particular attention for its theoretical grounding and strong empirical results. UW learns task-specific weights by modeling homoscedastic uncertainty. However, we identify and empirically demonstrate several limitations of this approach—such as its susceptibility to overfitting, the restrictive assumption of purely homoscedastic uncertainty, and its lack of theoretical justification for many commonly used loss functions in deep learning—ultimately hindering its robustness and reliability across diverse datasets and settings.

In this work, we make two key contributions. First, we conduct a thorough investigation of Uncertainty Weighting, identifying and analyzing several of its core limitations. Second, we introduce a novel loss weighting method-*Soft Optimal Uncertainty Weighting* (UW-SO), which builds upon UW while addressing its shortcomings. Rather than learning task weights via gradient descent under the homoscedastic assumption, we derive the analytical solution of UW, revealing that optimal task weights correspond to the inverse of task losses. We then apply a softmax normalization with a temperature parameter to these weights, enabling controlled and balanced task weighting throughout training. Our extensive experiments across multiple datasets and network architectures demonstrate that UW-SO consistently outperforms UW and exhibits improved robustness, notably mitigating overfitting. Following the recommendations of Xin et al. Xin et al. (2022), we further ensure the validity and comparability of our results by applying method-specific tuning of learning rate and weight decay, which was often overlooked in previous works.

This paper presents an extended and revised version of our earlier work (Kirchdorfer, Elich, Kutsche, Stuckenschmidt, Schott, and Köhler, 2024), in which we introduced the first version of our UW-SO method for weighting losses in MTL. The current paper builds upon that foundation in three major directions. First, we substantially deepen the discussion on uncertainty-based loss weighting by identifying and analyzing a range of theoretical and empirical limitations inherent to the UW approach. Second, we provide a more comprehensive theoretical analysis of the UW-SO method, explicitly highlighting how it addresses the shortcomings of UW. Third, we extend our experimental evaluation and ablation studies by exploring different strategies for determining the optimal softmax temperature and by examining the effect of batch size on the performance of several loss weighting methods.

The remainder starts with discussing related work in Section 2, followed by an in-depth investigation of uncertainty-based loss weighting in Section 3. Then, Section 4 details our proposed loss weighting method UW-SO, before reporting on several evaluation experiments in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

Research on MTL can broadly be categorized into three main areas. The first line of work focuses on neural network architectures, particularly on how features should be shared across tasks to maximize performance (Duong, Cohn, Bird, and Cook, 2015; Yang & Hospedales, 2016; Misra, Shrivastava, Gupta, and Hebert, 2016; Liu, Johns, and Davison, 2019; Xu, Ouyang, Wang, and Sebe, 2018; Maninis, Radosavovic, and Kokkinos, 2019). In this work, we use a neural backbone with a separate prediction head for each task, sharing the lower-layer parameters across tasks. The second area investigates task affinities, also referred to as task groupings, which aim to capture interdependencies between tasks and leverage them to improve joint training (Li, Sharma, and Zhang, 2024; Fifty, Amid, Zhao, Yu, Anil, and Finn, 2021; Standley, Zamir, Chen, Guibas, Malik, and Savarese, 2020; Li, Ju, Sharma, and Zhang, 2023). The third area, multi-task optimization (MTO), focuses on how to appropriately balance tasks during training to mitigate issues such as negative transfer and imbalance in task progress. This can be further divided into two subcategories: loss-weighting and gradient-based methods.

Loss weighting methods address the challenge of weighting task-specific losses appropriately. Most relevant for our work, Uncertainty Weighting (**UW**) (Kendall, Gal, and Cipolla, 2018) weights different losses by learning the

respective task-specific homoscedastic uncertainty. We adapt this by computing task weights based on the analytically optimal solution of UW and normalizing the results through a softmax function (see Section 4.2). In contrast to the weighted sum of losses, the geometric loss strategy (**GLS**) computes the geometric mean of task losses (Chennupati, Sistu, Yogamani, and A Rawashdeh, 2019). This approach eliminates the need for additional hyperparameters, but its multiplicative nature makes it numerically unstable when applied to a large number of tasks. Dynamic Weight Averaging (**DWA**) assigns higher weights to tasks whose losses decrease more slowly relative to others, thereby encouraging the model to focus on harder or slower-learning tasks (Liu, Johns, and Davison, 2019). In contrast to UW, DWA is a heuristic method and, similarly to our approach, relies solely on the temporal dynamics of task losses. However, our method is grounded in the underlying assumptions of UW that tasks with lower uncertainty should receive higher weights. Impartial Multi-Task Learning (**IMTL-L**) (Liu, Li, Kuang, Xue, Chen, Yang, Liao, and Zhang, 2021) aims to ensure that all tasks contribute equally to training by learning task-specific scaling factors for the losses. These scaling factors are optimized via gradient descent such that the scaled losses remain constant across tasks throughout training. The brute force method **Scalarization** (Xin, Ghorbani, Garg, Firat, and Gilmer, 2022) searches all possible combinations of fixed loss weights. While Scalarization has empirically demonstrated strong performance compared to current automated MTO methods, recent theoretical work shows that the approach is generally incapable of fully tracing out the Pareto front over the individual task losses (Hu, Xian, Wu, Fan, Yin, and Zhao, 2023). Random Sampling of Loss Weights (**RLW**) is a stochastic weighting strategy in which task-specific weights are sampled per batch, typically from a Gaussian distribution and normalized, e.g., via softmax. We consider it a baseline as proposed by Baijiong et al. (2021). Other loss weighting methods are proposed in (Lakkapragada, Sleiman, Surabhi, and Wall, 2023; Lin, Jiang, Ye, Zhang, Chen, Chen, Liu, and Kwok, 2023; Fan, Chen, Tian, Li, He, and Jin, 2022; Guo, Haque, Huang, Yeung, and Fei-Fei, 2018; Vasu, Saxena, and Tuzel, 2021).

Gradient-based methods leverage task-specific gradients to either compute scaling factors applied directly to each task's gradient (Chen, Badrinarayanan, Lee, and Rabinovich, 2018; Sener & Koltun, 2018; Liu, Li, Kuang, Xue, Chen, Yang, Liao, and Zhang, 2021; Navon, Shamsian, Achituve, Maron, Kawaguchi, Chechik, and Fetaya, 2022; Mao, Wang, Liu, Lin, and Xie, 2022; Senushkin, Patakin, Kuznetsov, and Konushin, 2023), or to manipulate gradients in order to mitigate conflicts in their directions (Liu, Li, Kuang, Xue, Chen, Yang, Liao, and Zhang, 2021; Chen, Ngiam, Huang, Luong, Kretzschmar, Chai, and Anguelov, 2020; Javaloy & Valera, 2022; Shi, Li, Zhang, Chen, and Wu, 2023; Liu, Feng, Stone, and Liu, 2023). These methods vary in how they handle gradient conflicts, for example by projecting gradients onto the normal plane (Yu, Kumar, Gupta, Levine, Hausman, and Finn, 2020), or optimizing for trade-offs between average and worst-case task performance (Liu, Liu, Jin, Stone, and Liu, 2021). A key drawback of gradient-based approaches is their computational overhead, as they require computing task-wise gradients during training. In this work, we focus on loss weighting methods instead, as they have been shown to outperform gradient-based alternatives in practice (Xin, Ghorbani, Garg, Firat, and Gilmer, 2022; Kurin, Palma, Kostrikov, Whiteson, and Mudigonda, 2022), with notably shorter training times (Chen, Ngiam, Huang, Luong, Kretzschmar, Chai, and Anguelov, 2020).

3 Background

In this section, we first formally introduce the general MTL problem statement, which we use in this work. Next, we describe the UW method, which serves as the foundation for our work, providing a comprehensive discussion of its potential limitations.

3.1 MTL Problem Statement

In MTL, we aim to resolve K tasks for some input data point $x \in \mathcal{X}$. For this, x is mapped to labels $\{y_k \in \mathcal{Y}_k\}_{k\in[1,K]}$ simultaneously using specific mappings $\{f_k : \mathcal{X} \to \mathcal{Y}_k\}$. Thus, we consider the typical scenario of a single input domain $\mathbb{X}$ for multiple tasks $\mathcal{K}$. We assume hard task-shared parameters θ in a hydra-like neural network architecture. This means all tasks receive the same intermediate feature $z = f(x;\theta)$ from the shared backbone and each task head yields the output $f_k(x) = f'_k(z;\theta_k)$ with task-specific parameters θ_k (Ruder, 2017).

The network is trained by considering all tasks' losses L_k. Naively summing up these losses (the EW method) typically leads to imbalanced learning as tasks with high gradient magnitude might dominate the training. The goal is thus to find optimal (dynamic) loss weights ω_k for all tasks to optimize the loss $L = \sum_k \omega_k L_k$ in a way that all tasks benefit w.r.t. their final performance metrics.

3.2 Uncertainty-Based Loss Weighting

Uncertainty-based weighting of task losses in MTL, as proposed by the UW method (Kendall, Gal, and Cipolla, 2018), has emerged as one of the most widely adopted approaches. Its popularity is largely attributed to its simplicity, computational efficiency, and strong empirical performance (see section 5). The core assumption underlying UW is that task losses can be adaptively weighted based on its aleatoric

homoscedastic uncertainty. To provide the necessary context, we begin by briefly reviewing key concepts and assumptions of uncertainty in deep learning, after which we detail the UW method and critically examine its potential limitations.

3.2.1 Uncertainties in Bayesian Deep Learning

In Bayesian deep learning, uncertainty quantification plays a critical role in enhancing model robustness and interpretability, particularly in safety-critical applications such as autonomous driving or medical imaging. There are two principal types of uncertainty that can be modeled: *epistemic uncertainty* and *aleatoric uncertainty* (Kendall & Gal, 2017).

Epistemic uncertainty It captures uncertainty in the model parameters and reflects the model's ignorance due to limited training data. It can be resolved with the availability of more data. Epistemic uncertainty is particularly significant in regions of the input space that are underrepresented during training and is typically modeled by placing a prior distribution over the model's weights.

Aleatoric uncertainty In contrast, aleatory uncertainty captures the inherent noise in the observations—uncertainty that persists even with unlimited data. This type of uncertainty arises from factors such as sensor noise or intrinsic variability in the data-generating process. Aleatoric uncertainty can be further subdivided into two categories:

- **Heteroscedastic** (data-dependent) uncertainty, which varies with the input data and is typically modeled as an additional output of the network.
- **Homoscedastic** (task-dependent) uncertainty, which is independent of the input data but may differ between tasks. This form of uncertainty is not input-dependent and may be modeled using a single parameter per task.

3.2.2 Uncertainty Weighting

Building upon the concept of homoscedastic uncertainty, Kendall et al. (2018) proposed the Uncertainty Weighting (UW) method as a principled approach for balancing task losses in MTL. Rather than relying on manually tuned weights, UW leverages homoscedastic uncertainty to adaptively learn the weighting of each task's loss during training. The key idea is to model the likelihood of each task output using a probabilistic formulation, where, e.g., regression tasks that are evaluated by the L_2 loss follow a Gaussian distribution, such that

$$p(y_k \mid f_k(x)) = \mathcal{N}(f_k(x), \sigma_k^2), \tag{1}$$

where σ_k^2 is assumed to represent the task-specific homoscedastic uncertainty. Minimizing the negative log-likelihood for the above regression task yields the task loss

$$L_k = \frac{1}{2\sigma_k^2} L_k + \log \sigma_k. \tag{2}$$

The term $\frac{1}{2\sigma_k^2}$ acts as an adaptive weight, reducing the influence of noisier tasks, while $\log \sigma_k$ serves as a regularizer to avoid the trivial solution $\sigma \to \infty$. Here, L_k refers to an L_2^2 loss $||f_k(x;\theta) - y||_2^2$.

It is important to note that the exact loss formulation in the UW method depends on the nature of the task, or more precisely, on the associated loss function. While the derivation presented above holds for regression tasks evaluated using the L_2 loss, it does not directly apply to regression tasks employing the L_1 loss, or to classification tasks, which are typically optimized using the cross-entropy loss. For example, an L_1-based regression task corresponds naturally to a Laplace likelihood, which leads to a modified task loss of the form:

$$L_k = \frac{1}{\sigma_k} L_k + \log \sigma_k, \tag{3}$$

where L_k refers to an L_1 loss $||f_k(x;\theta) - y||_1$. Having outlined the UW method, we now proceed to identify and critically examine potential limitations of the approach and its underlying assumptions.

Probabilistic interpretation A key assumption of UW is that each task loss naturally corresponds to some likelihood. Thus, UW requires each task loss L_k to be interpreted as the negative log-likelihood under a specific noise model. For instance, following the derivation of Kendall et al. (2018), we showed above that in a regression setting, there is a natural connection between the Gaussian distribution and the L_2 loss, thus providing a link between loss and task uncertainty. The same was shown to hold for the Laplace distribution and the L_1 loss.

However, many loss functions used in deep learning do not naturally correspond to such likelihoods. As a result, the probabilistic interpretation—central to the UW method—is not naturally applicable to such losses. This mismatch can lead to suboptimal weighting, because the underlying assumptions about the data distribution and its noise properties may not accurately reflect the behavior of the task. For instance, *cosine similarity loss* has no natural link to a distribution, making it inapplicable to be used in the uncertainty-based weighting formulation (Liu, Li, Kuang, Xue, Chen, Yang, Liao, and Zhang, 2021). Furthermore, Kendall et al. (2018) extend their UW method to classification tasks by interpreting task uncertainty as a temperature scaling of the softmax function, which however requires a simplifying assumption to derive an uncertainty-based weight formulation for classification tasks with *cross-entropy*

loss. Specifically, the authors define the likelihood as a scaled softmax function

$$p(y \mid f(x), \sigma) = \text{Softmax}(\frac{1}{\sigma^2} f(x)). \tag{4}$$

This scaling controls the confidence of the model's predictions: larger σ^2 (higher uncertainty) yields a flatter and more evenly distributed softmax (lower confidence), and vice versa. The corresponding log-likelihood is given by:

$$\log p(y = c \mid f(x), \sigma) = \frac{1}{\sigma^2} f_c(x) - \log \sum_{c'} \exp\left(\frac{1}{\sigma^2} f'_c(x)\right), \tag{5}$$

where f_c is the logit corresponding to the true class. This expression depends on σ in both terms, making it nonlinear and complex to optimize, especially because the log-sum-exp term is difficult to handle analytically when scaled. Therefore, the authors introduce a simplifying assumption

$$\sum_{c'} \exp\left(\frac{1}{\sigma^2} f'_c(x)\right) \approx \left(\sum_{c'} \exp(f'_c(x))\right)^{1/\sigma^2}. \tag{6}$$

Introducing this approximation, the loss formulation can be derived as follows:

$$L_k = \frac{1}{\sigma_k^2} L_k + \log \sigma_k. \tag{7}$$

However, this approximation only holds true for $\sigma \to 1$, and it is more valid when the logits $f'_c(x)$ are close in value, i.e., when the softmax distribution is close to uniform. In this case, raising the sum of exponentials to a power is close to summing the individually scaled exponentials. However, the approximation fails to hold in cases where σ differs significantly from 1, especially when large or very small, or the logits $f'_c(x)$ are highly variable, i.e., when one class has a dominant logit, and others are near zero or negative. In such cases, the nonlinearity of the exponential function amplifies errors introduced by approximating the sum of scaled exponentials with a powered sum. This can cause significant deviations in the estimated loss values.

In summary, the probabilistic foundation of UW—which assumes that each task loss corresponds naturally to a specific likelihood function—does not extend to many commonly used loss functions in deep learning. As a result, the resulting task weights may be suboptimal or theoretically inconsistent.

Homoscedastic assumption In UW, the aleatoric uncertainty is assumed to be homoscedastic, meaning it is constant for all data points within a given task $k \in \mathcal{K}$:

$$\sigma_k(x) = \sigma_k \quad \forall x. \tag{8}$$

However, in real-world applications, noise is often *hetero*scedastic, i.e., data dependent. For example, correctly identifying a traffic sign in sunny weather is much more certain than detecting it under foggy conditions. Furthermore, given that UW assumes task uncertainty to be purely aleatoric, it ignores epistemic uncertainty, which captures the model's confidence in its own predictions. In practice, both types of uncertainty are intertwined - limited data or model capacity can lead to uncertainty that is epistemic in nature but gets absorbed into the estimated $\hat{\sigma}$. A single global parameter per task may therefore fail to capture the true variability in the uncertainty.

To show this, Figure 1 depicts the results of a simple toy experiment, where a neural network learns a function $f(x) = x + \epsilon$, with $\varepsilon \sim \mathcal{N}(0, \sigma)$ accounting for homoscedastic uncertainty in the data. The gray hyperplane in the plot represents the optimal prediction for which $\hat{\sigma} = \sigma$. We can observe that $\hat{\sigma}$ deviates significantly from the true value σ in settings with few data points or small networks, for which the epistemic uncertainty is high.

Focus on tasks with low uncertainty

Under UW, tasks with smaller estimated σ are prioritized because their losses receive higher weight due to inverse scaling. However, if the epistemic uncertainty is high, the model may compensate by learning a larger σ when it struggles to fit the data. Consequently, tasks perceived as having higher noise—due to a mismatch between the model's assumptions and actual noise—get down-weighted and effectively neglected during optimization. This imbalance can degrade overall performance, especially when noise varies systematically across the input domain.

Task weight independence In UW, the uncertainties across different tasks are assumed to be independent. This assumption allows the joint likelihood to be factorized as

$$p(\mathbf{y}_1, \ldots, \mathbf{y}_K \mid \mathbf{f}_\theta(\mathbf{x})) = \prod_{k=1}^{K} p(\mathbf{y}_k \mid \mathbf{f}_\theta(\mathbf{x})), \tag{9}$$

which leads to a total loss as a weighted sum over individual task losses (here tasks with L_1 loss; see Equation 3):

$$L = \sum_{k \in \mathcal{K}} \frac{1}{\sigma_k} L_k + \log \sigma_k. \tag{10}$$

However, in real-world MTL scenarios, this independence assumption is often violated. Tasks commonly share intermediate representations and structural cues, leading to correlated predictions and errors. For instance, in computer vision, the tasks of surface normal prediction and depth estimation are inherently linked: surface normals can be computed from the depth map, and both rely on similar geometric

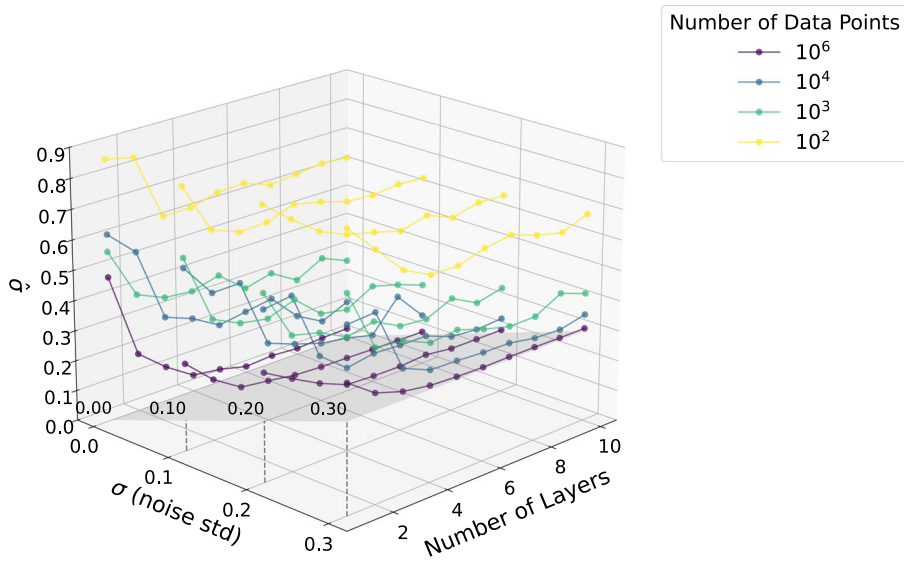


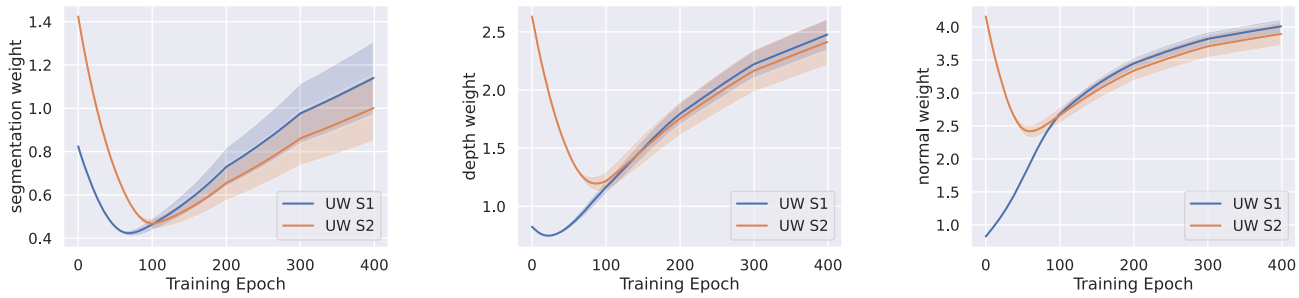
Fig. 1 Learned UW $\hat{\sigma}$ values (last training epoch) for different noise std deviations σ across different sized datasets and neural networks. With increasing amount of data and growing network capacity, $\hat{\sigma}$ converges towards σ. The gray hyperplane defines the area for which $\hat{\sigma} = \sigma$

features in the image. Consequently, an error in depth estimation (e.g., due to occlusions or ambiguous depth cues) can propagate to surface normal predictions.

Caruana's seminal work on MTL (Caruana, 1997) highlights that task interdependence can also serve as a beneficial inductive bias, improving generalization through auxiliary tasks. Ignoring such correlations by modeling tasks with independent noise can yield suboptimal weighting in the UW scheme. In particular, if two tasks are strongly correlated but modeled independently, the task with a smaller estimated uncertainty σ_k will receive a disproportionately high weight. This can dominate the optimization process and suppress learning signals from related tasks, even though joint training would ideally balance their influence.

Inertia While the limitations discussed above were derived from a rather theoretical perspective, we now examine the limitation of *inertia* from an empirical standpoint. We refer to inertia as the phenomenon of slow update steps of the learned uncertainty parameters. Specifically, when uncertainty-based task weights are initialized equally across tasks, this can hinder the convergence to optimal values for each task and epoch during gradient descent, causing a slow adaptation to the true value of σ. This issue is particularly pronounced when the ideal task weights differ by orders of magnitude. To empirically illustrate this phenomenon, we analyze the evolution of task weights ω_k on the NYUv2 dataset using two different initialization strategies (Figure 2). In the first setting (blue line, UW S1), we apply the standard initialization $\omega_k = 0.8$ (Lin & Zhang, 2023). In the second setting (orange line, UW S2), we initialize each task weight to the final value of ω_k obtained from a previous training run. The results show that task weight trajectories differ significantly depending on the initialization. Notably, it takes approximately 100 epochs—around one-quarter of the total training duration—before the weights converge, at which point the blue and orange curves begin to behave similarly. Since both experiments are exposed to identical unweighted task losses from the beginning of the training, we would expect the weights ω_k to rapidly adapt to the task-specific losses, effectively overriding any suboptimal initialization. However, this is not the case. This delay in adaptation highlights the impact of non-optimal initialization on training dynamics, driven by the update inertia of the task weights. In contrast, our UW-SO method is not affected by update inertia as it computes the task weights directly from the losses instead of learning it via gradient descent.

Overfitting In our experiments, we find that UW is prone to *overfitting*, as indicated by a substantial gap between training and test losses (see Section 5). We attribute this behavior to several factors. First, as discussed above, UW relies on the strong assumption of *homoscedastic* task uncertainty, i.e., task uncertainty is constant across all inputs. This assumption often fails to capture the inherent variability in data complexity and label noise such as heteroscedasticity, leading to miscalibrated uncertainty estimates. As a result, UW may overfit to the training data by overemphasizing tasks that appear easy to optimize during training, while underperforming on more challenging or noisy examples at test time. A further contributing factor may be the tendency of the learned uncertainty parameters σ_k to shrink during training as the model becomes increasingly confident. This results in disproportionately large task weights—without any explicit upper bound—which causes the model to aggressively optimize certain tasks, typically those with easily minimized losses, at the expense of generalization. The built-in regularization term $\log \sigma_k$ may be too weak to sufficiently counteract this over-confidence, leading to loss weights that are overly tailored to the training data and ineffective for unseen inputs.



(a) Segmentation (b) Depth (c) Normal

Fig. 2 Comparison of the learning procedure of task weights for a) semantic segmentation, b) depth estimation, and c) surface normals on NYUv2 using SegNet for two different initializations of σ_k for UW. Equal starting parameter values in blue (UW S1) as in (Lin & Zhang, 2023); higher starting values (values of last epoch from a previous run) in orange (UW S2). The plots do not show σ_k values, but actual task weights $\omega_k = \frac{1}{2\sigma_k^2}$. We plot the mean task weight of 5 random seeds with the standard deviation as shaded area

Having discussed uncertainty-based loss weighting and its limitations, we now introduce a novel loss weighting method that builds upon UW while addressing several of the aforementioned shortcomings.

4 Our Approach

In this section, we outline our proposed loss weighting method *Soft Optimal Uncertainty Weighting* (UW-SO). For this, we first derive the analytically optimal solution of UW in Section 4.1, before describing its softmax extension in Section 4.2.

4.1 UW-O Derivation

In Section 3, we outlined that UW estimates homoscedastic task uncertainty via gradient descent to adaptively weigh each task $k \in \mathcal{K}$. However, as discussed, this learning-based approach introduces several limitations, including *update inertia*—i.e., a delayed or suboptimal adaptation of task weights during training—and susceptibility to overfitting due to miscalibrated uncertainty estimates. To address these issues, we propose to derive the *analytical solution* for UW, thereby replacing the learned uncertainty parameters with closed-form expressions.

Recall Equation 3 where we defined the uncertainty-based weighted loss for tasks captured by the L_1 loss. We can write the optimization objective as

$$\min_{\sigma_k} \frac{1}{\sigma_k} L_k + \log \sigma_k, \tag{11}$$

where we minimize the UW loss function with respect to σ_k. Taking the derivative and solving for σ_k results in an analytically optimal solution:

$$\frac{\partial}{\partial \sigma_k} \frac{1}{\sigma_k} L_k + \log \sigma_k = -\frac{1}{\sigma_k^2} L_k + \frac{1}{\sigma_k} \tag{12}$$

$$-\frac{1}{\sigma_k^2} L_k + \frac{1}{\sigma_k} \stackrel{!}{=} 0 \tag{13}$$

$$\sigma_k = L_k \tag{14}$$

We assume σ_k to be positive and therefore only allow for positive losses. Note that this limitation comes from UW which only works for losses that are positive and are based on a location scale distribution.

Replacing σ_k with its analytical solution L_k in the total loss function gives the following loss:

$$L = \sum_{k \in \mathcal{K}} \frac{1}{sg[L_k]} L_k + \log sg[L_k], \tag{15}$$

where we denote sg as the stopgradient operator to avoid zero gradients of the network updates. Since we do not compute any gradient of the second part of the loss, we can simplify the term, such that each loss is effectively weighted by its inverse:

$$L = \sum_{k \in \mathcal{K}} \frac{1}{sg[L_k]} L_k. \tag{16}$$

For later reference, we name this intermediate result *Optimal Uncertainty Weighting* (UW-O), where optimal refers to the analytical loss minimum. Note that although the analytical solution for tasks using L_2 or cross-entropy loss differs by a factor of 2 in the denominator (see Kirchdorfer et al. (2024)), we adopt Equation 16 as the weighting scheme for all tasks for the sake of simplicity and empirically observed robustness. Interestingly, we found that there exist three

approaches that introduce related concepts: 1) **IMTL-L** (Liu, Li, Kuang, Xue, Chen, Yang, Liao, and Zhang, 2021) aims to have each weighted loss $\omega_k L_k$ scaled to 1, though, they learn ω_k using gradient descent. 2) **Dual-balancing** (Lin, Jiang, Ye, Zhang, Chen, Chen, Liu, and Kwok, 2023) transforms the loss to $\log L$ to normalize over different scales. Taking the gradient of $\log L$ is equivalent to taking the gradient of $\frac{1}{sg[L_k]} L_k$ (Lin, Jiang, Ye, Zhang, Chen, Chen, Liu, and Kwok, 2023, Sec. 3.1) which is 1 for all tasks $k \in \mathcal{K}$; for the $L = ||\cdot||_1$ loss it is equivalent to Equation 16. Thus, in dual-balancing the gradient is scaled whereas we scale the loss L. 3) **EMA** (Lakkapragada, Sleiman, Surabhi, and Wall, 2023) scales the loss by the Exponential Moving Average which is identical to the *inverse loss* in UW-O when the hyperparameter $\beta = 1$.

4.2 UW-SO

While UW-O offers a simple and cost-efficient scheme by scaling each task loss to 1, we propose to enhance it by applying a *softmax* normalization, yielding **UW-SO**. Inspired by the findings of Xin et al. (2022), who demonstrate that manually tuned task weights ω_k in an exhaustive grid-search such that $\sum_{k\in\mathcal{K}} \omega_k = 1$ (a method known as *Scalarization*) can outperform existing MTO methods, UW-SO introduces a single *temperature parameter* T to control the smoothness of the weight distribution. This enables improved performance, as shown in our experimental results in Section 5, while avoiding the computational cost of tuning K separate weights, as required by Scalarization.

Formally, UW-SO computes the task weights using a tempered softmax over the analytically derived inverse losses:

$$L = \sum_{k\in\mathcal{K}} \omega_k L_k = \sum_{k\in\mathcal{K}} \frac{exp(\frac{1}{sg[L_k]}/T)}{\sum_{j\in\mathcal{K}} exp(\frac{1}{sg[L_j]}/T)} L_k. \tag{17}$$

The temperature T controls the sharpness of the weighting: higher values produce more uniform weights, while lower values yield more selective weighting. Unlike Scalarization, which becomes impractical for a large number of tasks, UW-SO is scalable to arbitrary many tasks.

The (tempered) softmax function is widely used in the field of MTL. For instance, the MTO methods RLW and DWA also use softmax normalization with $T = 1$ and $T = 2$, respectively. Other domains related to discrete selection also make use of the tempered softmax: Caccia et al. (2019) uses it to control the quality-diversity trade-off of generated samples in GANs. Hinton et al. (2015) use it in knowledge distillation to provide softer outputs from the teacher model. Other usages are out-of-distribution detection (Liang, Li, and Srikant, 2018) or confidence calibration (Guo, Pleiss, Sun, and Weinberger, 2017). Works in hash learning (Tan, Liu, Zhao, Yang, Zhou, and Hu, 2020) and Neural-Architecture search (Chang, Zhang, Guo, Meng, Xiang, and Pan, 2019) both use the Gumbel-Softmax with temperature (Jang, Gu, and Poole, 2016).

Beyond the intuitive motivation for applying a softmax, as discussed above, we now examine several properties of UW-SO, which by help of the softmax function address key limitations of the UW method outlined in Section 3.

4.2.1 Properties of UW-SO

Modeling task interdependence The softmax normalization inherently couples the tasks together by enforcing the weights to sum up to 1:

$$\sum_{k\in\mathcal{K}} \omega_k = 1. \tag{18}$$

As task weights need to be in [0, 1], the optimization of one task's weight directly influences the others, creating a trade-off among tasks. This constraint enforces a dynamic, where increasing the importance of one task necessitates decreasing the importance of others. By introducing this normalized weighting scheme, UW-SO alleviates a key limitation of UW—namely, the independent treatment of task weights—and instead better models interdependencies between tasks, which is more aligned with the core motivation of MTL: improving generalization through shared learning.

Balancing task prioritization via T The temperature parameter T controls the sensitivity of the weight allocation to task losses: lower values of T cause the weights to concentrate more heavily on a few tasks, while higher values spread them more uniformly. A higher value of T can therefore mitigate the limitation of UW where tasks with low uncertainty are prioritized. Moreover, T can be adjusted dynamically during training, drawing an analogy to the exploration–exploitation trade-off in Reinforcement Learning or the shifting focus characteristic of Curriculum Learning.

Mitigating the homoscedasticity assumption of UW UW assumes homoscedastic task uncertainty, modeling each task's noise level with a single learnable parameter and thereby ignoring variability across individual samples. UW-SO mitigates this limitation by determining task weights on the actual observed losses within a batch, rather than relying on static, task-specific parameters. Since the loss function aggregates information across individual samples, variations in sample difficulty or noise can influence the task weight indirectly. This introduces a form of data-driven adaptivity that reflects changing uncertainty patterns without requiring explicit modeling of per-sample heteroscedasticity.

4.2.2 Strategies for Determining the Softmax Temperature T

The temperature parameter T plays a critical role in the performance of UW-SO and, as we will demonstrate in our experiments, its optimal value varies significantly across datasets. In the following, we identify and discuss several strategies for selecting an appropriate temperature.

Strategy 1: Tune as hyperparameter The most straightforward approach to selecting the temperature T is inspired by the *Scalarization* method, as described above: performing a grid search over a predefined range. In this case, T is treated as a static hyperparameter, determined prior to training and kept fixed throughout. As our experiments will demonstrate, this approach is the most robust one by yielding superior performance across various datasets and architectures. While it substantially reduces the computational burden of Scalarization—by requiring the tuning of only one single hyperparameter—it remains more costly than UW, which involves no hyperparameter at all. To address this, we discuss two alternative strategies for deriving the value of T.

Strategy 2: Solve analytically The first cost-efficient alternative we explore is an analytical solution to the optimization problem for T. The corresponding loss function under the UW-SO weighting scheme is given by:

$$L(T) = \sum_{k\in\mathcal{K}} \omega_k(T) L_k, \tag{19}$$

where $\omega_k(T) = \frac{\exp\left(\frac{a_k}{T}\right)}{\sum_{j\in\mathcal{K}} \exp\left(\frac{a_j}{T}\right)}$ and $a_k = \frac{1}{\text{sg}[L_k]}$. To find the optimal temperature T, we differentiate $L(T)$ with respect to T:

$$\frac{dL}{dT} = \sum_{k\in\mathcal{K}} \frac{d\omega_k}{dT} L_k. \tag{20}$$

Using the chain rule and the definition of $\omega_k(T)$, we obtain:

$$\frac{d\omega_k}{dT} = \omega_k(T) \frac{1}{T^2} \left[\sum_{j\in\mathcal{K}} a_j w_j(T) - a_k \right]. \tag{21}$$

Thus, the derivative of the loss is:

$$\frac{dL}{dT} = \frac{1}{T^2} \sum_{k\in\mathcal{K}} \omega_k(T) \left[\alpha(T) - a_k\right] L_k, \tag{22}$$

where $\alpha(T) = \sum_{j\in\mathcal{K}} a_j w_j(T)$. Setting $\frac{dL}{dT} = 0$ gives the necessary condition:

$$\sum_{k\in\mathcal{K}} \omega_k(T) \left[\alpha(T) - a_k\right] L_k = 0. \tag{23}$$

Since $\omega_k(T)$ depends on $\exp(a_k/T)$ and appears in the denominator of a sum, solving for T explicitly is not possible with elementary functions. The resulting equation is *transcendental*, meaning that T cannot be solved in a closed-form.

Strategy 3: Learn via gradient descent Since T does not admit a closed-form solution, we also investigate the possibility of learning it dynamically via gradient descent. To this end, we treat T as an additional learnable parameter of the neural network, initialized to its default value of 1. However, directly optimizing the UW-SO formulation from Equation 17 can lead to degenerate solutions where $T \to 0$, concentrating all weight on the task with the smallest loss magnitude and effectively ignoring the others. To mitigate this, we extend the UW-SO loss with a regularization term that penalizes excessively small temperatures:

$$L = \sum_{k\in\mathcal{K}} \frac{exp(\frac{1}{sg[L_k]}/T)}{\sum_{j\in\mathcal{K}} exp(\frac{1}{sg[L_j]}/T)} L_k + \frac{1}{T}, \tag{24}$$

where the additive term $\frac{1}{T}$ serves as a regularizer to discourage the collapse of the temperature and promote balanced task weighting.

5 Experiments and Results

This section presents the experiments used to evaluate the performance of our UW-SO method for weighting tasks in MTL scenarios. In the remainder, Section 5.1 describes the experimental setup, followed by the discussion of the results in Section 5.2, and various ablation studies in Section 5.3.

5.1 Experimental Setup

In the following, we provide details about the datasets, network architectures, metrics, evaluation setups, and hyperparameters employed in our experiments.

Datasets We use three common computer vision MTL datasets: two datasets for scene understanding—*NYUv2* (Nathan Silberman et al., 2012) and *Cityscapes* (Cordts, Omran, Ramos, Rehfeld, Enzweiler, Benenson, Franke, Roth, and Schiele, 2016)—and a binary attribute dataset *CelebA* (Liu, Luo, Wang, and Tang, 2015).

NYUv2 contains 464 indoor scenes recorded in three different cities, resulting in 636 images for training, 159 for

validation, and 654 images for testing. The dataset comprises three tasks: 13-class semantic segmentation, depth estimation, and surface normal prediction. For training, we follow previous work Liu et al. (2021, 2019) and resize the images to 288x384 and apply image augmentations. The augmentations include randomly scaling the images (ratio=1.0, 1.2, 1.5) and randomly flipping them.

Cityscapes consists of urban street scenes from 50 different cities yielding 2380 images for training, 595 for validation, and 500 images for testing. The dataset has 2 tasks: 7-class semantic segmentation and depth estimation. We resize the images to 128x256 and apply the same augmentation techniques used for NYUv2.

CelebA is a dataset of celebrity faces with different attributes of 10,177 identities. We cast it as an MTL dataset by viewing the available 40 binary attributes, e.g., glasses and smiling as individual classification tasks. The train, validation, and test set contain 162,770, 19,867, and 19,962 samples, respectively.

Architectures For NYUv2, we use a SegNet (Badrinarayanan, Kendall, and Cipolla, 2017), ImageNet pretrained ResNet-50 / ResNet-101 with a DeepLabHead, and the MTAN on top of the SegNet (Liu, Johns, and Davison, 2019). For Cityscapes, we use a SegNet, a DeepLabV3+ (Chen, Zhu, Papandreou, Schroff, and Adam, 2018) network with pre-trained ResNet-50 / ResNet-101 backbones, and again the MTAN/SegNet. All Single-Task Learning (**STL**) baselines are trained with the SegNet. For CelebA, we use a ResNet-18, also for STL.

Metrics To compare models, we use task-specific metrics and the established Δ_m-metric (Maninis, Radosavovic, and Kokkinos, 2019), which measures the average relative performance gain of the multi-task model M_m w.r.t. a single-task baseline M_b:

$$\Delta_m = \frac{1}{K}\sum_{k=1}^{K}(-1)^{l_k}(M_{m,k} - M_{b,k})/M_{b,k}, \tag{25}$$

where l_k is 1/0 if a higher / lower value is better for criterion M_k.

Two evaluation setups Several papers (e.g., Liu et al. (2021, 2019); Yu et al. (2020); Navon et al. (2022)) have used a fixed training protocol for NYUv2 and Cityscapes, with no hyperparameter tuning, averaging results over the last 10 test epochs. In contrast, other studies (e.g., Xin et al. (2022); Kurin et al. (2022); Sener and Koltun (2018)) advocate for method-specific hyperparameter tuning, which is more relevant for practitioners. Following Xin et al. (2022), we perform a thorough hyperparameter search for all methods, selecting the best combination based on the Δ_m score and using early stopping on the validation set. For final evaluation, we train on 5 random seeds and report the mean test performance. However, acknowledging other works, we also provide results using the fixed protocol with MTAN/SegNet on NYUv2 and Cityscapes.

All models are trained with the Adam optimizer which has been shown to perform advantageous on MTL setups (Elich, Kirchdorfer, Koehler, and Schott, 2024). Similar to Liu et al. (2021), we decay the learning rate after 100 epochs by a factor of 2 for both NYUv2 and Cityscapes. In CelebA, we halve it after 30 epochs. As batch size, we use 2 / 8 / 256 for NYUv2 / Cityscapes / CelebA. Compared to Liu et al. (2021), we increase the number of epochs for NYUv2 / Cityscapes to 400 / 600 epochs. CelebA remains at 100 epochs as there is no further improvement.

Hyperparameters For tuning the Learning Rate (LR) and Weight Decay (WD), and the hyperparameters of the MTO method, we employ a sequential line search by first tuning the LR together with the MTO hyperparameters using a fixed WD, followed by tuning the WD. For NYUv2 and Cityscapes, we use the following search space for the LR γ and WD λ: $\gamma \in [10^{-3}, 5*10^{-4}, 10^{-4}, 5*10^{-5}, 10^{-5}]$ and $\lambda \in [0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$. For CelebA, the search space looks as follows: $\gamma \in [5*10^{-2}, 10^{-2}, 5*10^{-3}, 10^{-3}, 5*10^{-4}, 10^{-4}]$ and $\lambda \in [0, 10^{-5}, 10^{-4}, 10^{-3}]$. In our main benchmark, we employ the hyperparameter tuning strategy for determining T in UW-SO. We search T with a step size of 5 and then employ a finer search around the optimum. Note that in an ablation study, we will compare this to the strategy of learning T via gradient descent. For Scalarization, we first test each possible combination of task weights with a step size of 0.1. If no proper result could be achieved (e.g., for Cityscapes), we further test values around the previously found optimum with a step size of 0.02. For DWA, we follow Liu et al. (2019) and set $T = 2$.

5.2 Benchmark of Common Loss Weighting Methods

We compare our method UW-SO to the most common loss weighting approaches. Overall, UW-SO consistently performs best or second-best across all datasets and architectures w.r.t. the Δ_m metric. This holds for the hyperparameter-tuned experiments as well as for those with the fixed training protocol. Occasionally, our method gets beaten by the computationally expensive Scalarization approach, especially when using the SegNet architecture. Although differences between MTO algorithms decrease for larger networks, UW-SO consistently outperforms UW in all experiments.

Cityscapes On Cityscapes, UW-SO achieves the best Δ_m score when trained on both ResNet architectures as well as on the MTAN/SegNet with the fixed hyperparameters, and the second-best behind Scalarization when trained on the SegNet (see Table 1). In contrast to NYUv2 (see Table 2), the performance of Scalarization decreases for larger networks due to weak results on the difficult and highly sensitive relative depth error. While an even more fine-grained search of task

Table 1 Test data results on Cityscapes with SegNet, ResNet-50, ResNet-101, and MTAN. Our method UW-SO is underlined. For the first 3 architectures, we report the average over 5 runs, for Δ_m including $\pm$ one std. dev. The best score is in bold, the second best is underlined. We report the best LR and WD values for each experiment. WD and LR are abbreviated as: a: 0.0, b: 10^{-6}, c: 5×10^{-6}, d: 10^{-5}, e: 5×10^{-5}, f: 10^{-4}, g: 5×10^{-4}, h: 10^{-3}, i: 5×10^{-3}, j: 10^{-2}. The best softmax temperature for the 4 architectures is $T = 20/28/48/22$. For Scalarization, best weights are equal for the first 3 architectures with $\omega = [0.02, 0.98]$ and for MTAN with $\omega = [0.04, 0.96]$ for segm. and depth. Additionally, we report results over 3 seeds based on the training protocol that uses MTAN+SegNet with fixed hyperparameters.

A.	E.	Method	lr	wd	Segmentation ↑		Depth ↓		Ep	$\Delta_m\%$ ↓
					mIoU	PixAcc	AbsErr	RelErr		
		STL	g	f,d	0.723	0.927	0.0124	24.5		
SegNet	early stopping	Scalar	f	d	0.707	0.919	**0.0123**	25.7	531.2	**1.9 ± 0.9**
		EW	g	d	**0.731**	**0.928**	0.0157	76.7	560.8	59.9 ± 5.0
		RLW	f	f	0.722	0.926	0.0170	107.1	529.6	93.8 ± 14.8
		DWA	f	f	0.729	0.927	0.0158	85.9	584.4	69.6 ± 9.7
		GLS	g	d	0.717	0.923	0.0129	27.5	564.2	4.5 ± 0.6
		IMTL-L	h	d	0.715	0.922	0.0128	33.5	568.6	10.6 ± 3.6
		UW	h	d	0.718	0.923	0.0128	32.1	550.6	9.0 ± 3.0
		UW-SO	g	d	0.674	0.907	0.0130	**25.2**	544.0	4.4 ± 0.9
ResNet-50	early stopping	Scalar	h	a	0.738	0.929	0.0117	28.3	417.2	2.1 ± 1.4
		EW	h	a	0.757	**0.936**	0.0119	31.9	448.4	5.3 ± 0.7
		RLW	g	a	0.757	**0.936**	0.0119	32.7	486.6	6.1 ± 1.1
		DWA	g	a	**0.758**	**0.936**	0.0118	31.4	404.6	4.4 ± 0.7
		GLS	h	b	0.755	0.934	0.0115	29.1	495.8	1.7 ± 1.2
		IMTL-L	g	a	0.754	0.935	0.0113	28.6	350.8	0.7 ± 0.9
		UW	h	a	0.752	0.934	0.0113	28.3	415.0	0.5 ± 0.8
		UW-SO	g	a	0.748	0.933	**0.0112**	**28.0**	363.4	**0.3 ± 1.3**
ResNet-101	early stopping	Scalar	f	d	0.740	0.931	0.0115	31.6	462.4	4.8 ± 4.7
		EW	h	a	0.749	0.933	0.0121	32.8	427.0	7.0 ± 0.7
		RLW	g	a	0.752	0.934	0.0119	32.7	482.0	6.3 ± 0.8
		DWA	g	b	**0.753**	**0.935**	0.0118	32.2	457.6	5.6 ± 0.9
		GLS	g	a	0.743	0.931	0.0116	28.9	408.6	2.1 ± 0.6
		IMTL-L	h	b	0.743	0.931	0.0116	**28.2**	374.2	1.6 ± 1.4
		UW	h	b	0.744	0.931	0.0116	28.5	337.8	1.8 ± 1.0
		UW-SO	g	a	0.749	0.933	**0.0113**	28.7	357.6	**1.1 ± 0.9**
MTAN/SegNet	avg. last 10 ep.	Scalar	f	a	0.721	0.927	0.0130	27.7	avg	−0.7 ± (1.2)
		EW	f	a	0.743	0.932	0.0158	44.4	avg	17.8 ± (3.1)
		RLW	f	a	0.736	0.931	0.0159	45.3	avg	19.1 ± (1.8)
		DWA	f	a	0.743	0.933	0.0159	45.6	avg	19.1 ± (2.7)
		GLS	f	a	0.729	0.930	0.0136	29.7	avg	1.9 ± (0.7)
		IMTL-L	f	a	0.744	**0.934**	0.0146	33.8	avg	6.5 ± (0.3)
		UW	f	a	**0.746**	**0.934**	0.0145	34.9	avg	7.2 ± (1.9)
		UW-SO	f	a	0.711	0.920	**0.0127**	**26.9**	avg	**−1.4 ± (0.6)**

weights might yield better results, we argue that our weight search with step size going down to 0.02 was performed adequately well to keep the computational cost feasible.

NYUv2 We observe comparable results on NYUv2 in Table 2, where UW-SO is again always best or second-best behind Scalarization. Both methods perform particularly well on the normal task, while still achieving strong results on the other two tasks. The fixed training protocol with MTAN (see last block in Table 2) leads to a slightly different order of goodness, especially IMTL-L ranks lower compared to the SegNet results.

CelebA Considering a more challenging setup with 40 tasks, UW-SO is clearly exceeding the performance of all other methods with a Δ_m score of −4.0 and an average error of 8.95 (see Table 3). In contrast to the other two datasets, we did not include Scalarization due to the infeasibility of per-

Table 2 Test data results on NYUv2 with SegNet, ResNet-50, ResNet-101, and MTAN. Our method is underlined. For the first 3 architectures, we report the average over 5 runs, for Δ_m including $\pm$ one std. dev. Best score is in bold, second best underlined. LR and WD follow the schema as in Table 1. The best softmax temperature for the 4 architectures is $T = 3/2/3/2$, for Scalarization best weights for all 4 architectures are $\omega = [0.8, 0.1, 0.1]$ for segm., depth, and normal. Additionally, we report results over 3 seeds based on the training protocol that uses MTAN+SegNet with fixed hyperparameters

A.	E.	Method	lr	wd	Segmentation ↑		Depth ↓		Surface Normal Ang Dist ↓		Within $t°$ ↑			Ep	
					mIoU	PixAcc	AbsErr	RelErr	Mean	Med	11.25	22.5	30		$\Delta_m\%$ ↓
		STL	e/f	d	0.361	0.625	0.601	0.252	24.8	18.3	0.312	0.585	0.701		
SegNet	early stopping	Scalar	e	f	**0.419**	**0.675**	0.502	0.204	**24.9**	**19.0**	**0.299**	**0.573**	**0.694**	360.4	**−5.3 ± 0.7**
		EW	e	h	0.411	0.666	0.512	0.207	28.1	23.2	0.231	0.487	0.618	380.0	4.6 ± 2.5
		RLW	e	f	0.394	0.655	0.523	0.208	28.9	24.3	0.220	0.467	0.599	388.8	7.7 ± 2.3
		DWA	e	h	0.412	0.666	0.509	0.207	28.0	23.0	0.234	0.490	0.621	376.4	4.2 ± 2.1
		GLS	f	f	0.380	0.640	0.516	0.211	26.8	21.4	0.261	0.521	0.649	372.0	2.4 ± 1.8
		IMTL-L	e	h	0.395	0.656	**0.498**	0.204	26.4	20.8	0.269	0.534	0.660	363.0	−0.3 ± 1.5
		UW	e	h	0.394	0.655	0.499	**0.201**	26.4	20.8	0.269	0.535	0.661	386.0	−0.3 ± 2.6
		UW-SO	e	f	0.405	0.666	0.508	0.205	25.7	20.0	0.282	0.551	0.675	363.8	−2.3 ± 1.1
ResNet-50	early stopping	Scalar	f	a	0.480	0.717	0.431	0.168	**25.4**	19.6	**0.298**	0.557	0.677	357.6	−9.7 ± 0.2
		EW	e	h	0.478	0.718	0.428	0.166	26.5	20.9	0.282	0.531	0.652	334.2	−7.1 ± 0.4
		RLW	e	f	0.482	0.720	0.433	0.168	26.6	21.1	0.279	0.527	0.649	387.0	−6.6 ± 0.3
		DWA	f	b	0.478	0.717	0.431	0.168	26.5	21.0	0.280	0.528	0.650	329.2	−6.6 ± 0.3
		GLS	f	f	0.469	0.711	0.425	0.166	25.8	19.9	0.296	0.550	0.668	347.4	−8.7 ± 0.4
		IMTL-L	f	h	0.466	0.708	0.434	0.166	26.0	20.2	0.290	0.544	0.664	362.0	−7.7 ± 0.3
		UW	e	b	**0.485**	**0.722**	0.433	0.169	26.4	20.8	0.282	0.532	0.654	324.4	−7.2 ± 0.2
		UW-SO	f	a	0.477	0.713	**0.424**	**0.165**	**25.4**	**19.5**	**0.298**	**0.558**	**0.678**	333.8	**−9.8 ± 0.2**
ResNet-101	early stopping	Scalar	e	b	0.500	0.732	0.417	0.159	**24.9**	**19.0**	**0.306**	**0.570**	**0.689**	341.8	**−12.5 ± 0.3**
		EW	e	f	0.499	**0.734**	0.415	0.158	25.9	20.1	0.291	0.546	0.666	389.0	−10.1 ± 0.2
		RLW	e	a	0.499	0.731	0.415	0.158	26.1	20.5	0.286	0.539	0.661	372.6	−9.4 ± 0.2
		DWA	e	a	0.498	0.733	0.416	0.158	25.9	20.2	0.290	0.544	0.665	364.2	−9.8 ± 0.3
		GLS	e	f	0.491	0.727	**0.410**	**0.157**	25.3	19.4	0.303	0.560	0.678	329.8	−11.6 ± 0.4
		IMTL-L	e	b	**0.503**	0.733	0.415	0.158	25.8	20.1	0.292	0.546	0.667	306.0	−10.3 ± 0.3
		UW	e	b	0.502	0.733	0.414	0.158	25.8	20.1	0.292	0.547	0.667	325.2	−10.4 ± 0.2
		UW-SO	e	d	0.494	0.725	0.412	**0.157**	**24.9**	**19.0**	**0.306**	0.569	0.687	299.2	−12.3 ± 0.3
MTAN/SegNet	avg. last 10 ep.	Scalar	f	a	**0.400**	**0.660**	0.534	0.225	25.7	20.6	0.267	0.539	0.671	avg	**−1.5 ± 0.9**
		EW	f	a	0.387	0.648	0.575	0.247	28.0	23.6	0.225	0.479	0.615	avg	7.0 ± 1.2
		RLW	f	a	0.382	0.637	0.584	0.248	28.5	24.3	0.216	0.465	0.602	avg	9.0 ± 2.8
		DWA	f	a	0.391	0.649	0.586	0.250	27.8	23.4	0.228	0.483	0.618	avg	6.8 ± 0.7
		GLS	f	a	0.384	0.649	**0.528**	**0.224**	26.7	21.9	0.249	0.511	0.644	avg	1.9 ± 1.1
		IMTL-L	f	a	0.376	0.637	0.572	0.248	26.4	21.3	0.261	0.523	0.653	avg	2.7 ± 1.3
		UW	f	a	0.382	0.646	0.550	0.236	26.4	21.3	0.259	0.524	0.655	avg	1.6 ± 1.3
		UW-SO	f	a	0.351	0.626	0.556	0.229	**25.4**	**19.8**	**0.285**	**0.555**	**0.680**	avg	−0.7 ± 0.7

Table 3 Test data results on CelebA with ResNet-18. We show the average test error (5 runs) over all 40 tasks. The chosen softmax temperature for UW-SO is $T = 100$. We exclude GLS and Scalarization due to infeasibility, see details in the text

A.	E.	Method	lr	wd	Avg Err ↓	Ep	$\Delta_m\%$ ↓
		STL	g	f	9.24		
ResNet-18	early stop.	EW	j	f	9.00	33.2	-2.4 ± 0.4
		RLW	h	f	9.01	54.4	-2.5 ± 0.6
		DWA	j	f	9.01	39.2	-2.5 ± 0.7
		IMTL-L	h	a	9.18	5.0	-1.2 ± 0.9
		UW	g	h	9.26	6.0	-0.1 ± 0.7
		UW-SO	g	f	**8.95**	61.2	$\mathbf{-4.0 \pm 0.2}$

forming a grid search over 40 task weights. While we tried to run 50 different random weight combinations, we were not able to beat the EW performance and thus omit to report these results. Furthermore, GLS is also not reported as the losses diverge due to numerical instabilities for a large number of tasks. In contrast to our previous results on other datasets, EW, RLW, and DWA show strong performance compared to UW and IMTL-L. We attribute this observation to UW and IMTL-L being prone to overfitting on some tasks, as indicated by how early the validation Δ_m score reaches its minimum (e.g., epoch 6 for UW). We analyze the overfitting behavior of UW in comparison to UW-SO later in this section. As indicated by the negative Δ_m score, one achieves a positive transfer by training on multiple tasks simultaneously. Examining the task-level performance to verify that the enhanced average performance of UW-SO is not solely attributable to a limited set of tasks, it is noteworthy that UW-SO surpasses UW/UW-O/IMTL-L/RLW/EW/DWA in 34/34/31/27/24/24 out of 40 tasks.

Overfitting of UW Following our results in Table 3, UW achieves the worst Δ_m score for CelebA. We investigate the reasons: Figure 3 shows the train and test loss as well as the weight ratio of the *Bald* task for UW and UW-SO. UW is subject to strong overfitting, as indicated by the huge gap between train and test loss. Contrary, UW-SO steadily decreases its training loss on the bald task, achieving its best test loss of 0.026 at epoch 31 whereas UW has its best test loss of 0.028 at epoch 5. For CelebA, for 34 out of 40 tasks, UW-SO achieves a lower test loss than UW and we assume this is due to overfitting—for most tasks, the train loss for UW drops to nearly 0. Related to this, we observe that UW distributes much of the relative weight to only a few tasks, as can be seen for the *Bald* task in Figure 3a.

Stronger networks and MTO approaches In Tables 1 and 2, our empirical analysis reveals an interesting trend: as network architectures increase in capacity, the influence of the MTL weighting method diminishes, e.g., the difference between best and worst Δ_m score on NYUv2 is 13.0 on the SegNet, but only 3.4/3.1 on ResNet-50 / ResNet-101. This finding raises the question about the necessity of loss weighting methods for networks with large capacity. Further research in this direction is necessary.

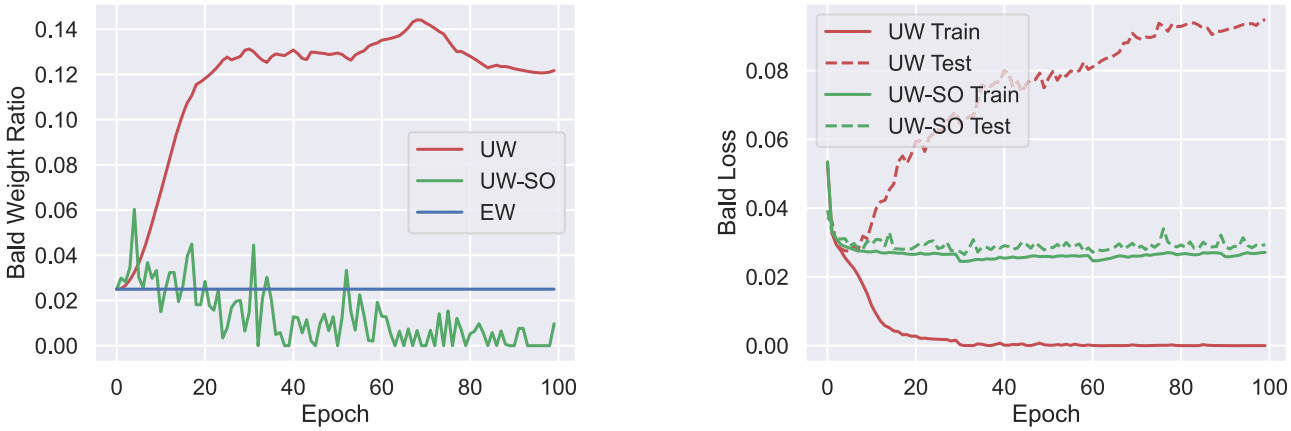
5.3 Ablations

In the following, we present various ablation studies further analyzing our method UW-SO and providing some general insights about MTL methods.

Influence of softmax To demonstrate the influence of the softmax function on the inverse loss weights, we compare the Δ_m scores using UW-O and UW-SO across all datasets and architectures in Table 4. UW-SO outperforms UW-O in all experiments, indicating the performance gain provided by the tempered softmax function. However, we want to emphasize that this is not due to a significantly worse performance of UW-O compared to other MTO methods. Therefore, we also present the results using IMTL-L, which, like UW-O, aims to scale each weighted task loss to 1, but unlike UW-O, it learns rather than computing the weights. Interestingly, none of the two methods can outperform the other one, indicating that despite UW-O's simplicity, it still provides reasonable results compared to existing methods.

Influence of temperature T While our approach UW-SO achieves consistently strong results, its performance depends on the value of T. In Figure 4, we show how the performance changes when the temperature T is tuned for Cityscapes with a SegNet. The best run was achieved for $T = 20$, but values close to it are also performing well. We conducted a line search for T in steps of 5 and further with a step size of 2 around the optimum. We conclude that it is possible with acceptable tuning effort to find a good value for T.

Development of validation metric for different T During analyses of the data, we observed that tuning of the hyperparameter T can be eased by the following finding: Figure 4bshows the validation Δ_m score for different T values over all epochs. Non-optimal T values are clearly identifiable after around 100 epochs as having constantly a higher Δ_m score compared to favorable T values. For instance, in this setup, it is reasonable to stop the runs for all initial T values with step size of 5 except for $T = 5$ and $T = 10$, and then proceed with a finer search around the optimum of

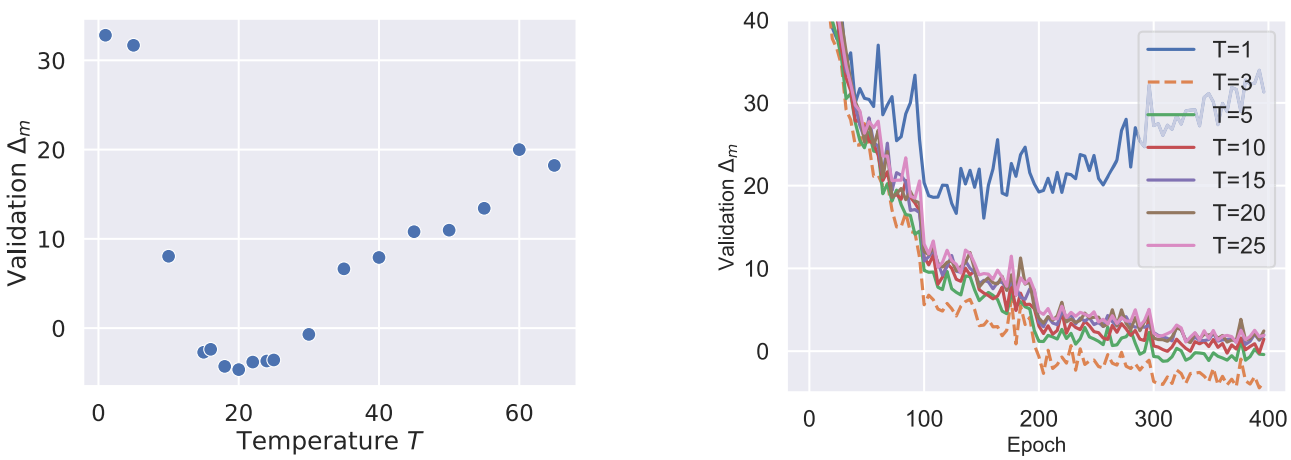


(a) Weight ratio (b) Loss

Fig. 3 Comparison of weight ratio and loss development of UW and UW-SO for the *Bald* task of CelebA. While UW shows superior training performance caused by putting a high weight on the task, it fails to generalize to unseen data (increasing test loss). UW-SO puts less weight on the task and alleviates the overfitting.

Table 4 Comparison of test Δ_m scores of UW-O, UW-SO, and IMTL-L across all evaluated datasets.

Method	NYUv2			Cityscapes			CelebA
	SN	RN-50	RN-101	SN	RN-50	RN-101	RN-18
IMTL-L	–0.3	–7.7	–10.3	10.6	0.7	1.6	–1.2
UW-O	0.0	–9.1	–11.9	5.5	2.1	2.8	–0.6
UW-SO	**–2.3**	**–9.8**	**–12.3**	**4.4**	**0.3**	**1.1**	**–4.0**



(a) Cityscapes SegNet (b) NYUv2 SegNet

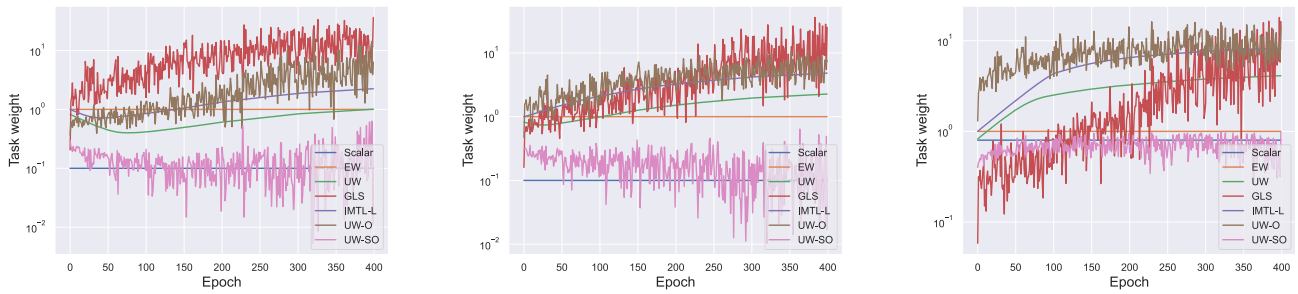
Fig. 4 Performance of UW-SO for different choices of T on the validation data. a) shows a clear, reasonably flat minimum for Cityscapes that eases the optimization of T. b) shows the Δ_m development for different T values for NYUv2, indicating the optimal configuration already after around 100 epochs

which $T = 3$ is best. This reduces computational resources by a large amount.

Learning T via gradient descent While we determined the value of T in our benchmark experiments by means of a grid search, we now want to investigate whether learning T via gradient descent—thereby heavily reducing the computational complexity—can achieve comparable results for different datasets. Table 5 compares the results of UW-SO for the two strategies for determining T. Notably, learning the temperature via *gradient descent* yields better results

Table 5 Comparison of test Δ_m scores for UW-SO with T being determined by *grid search* vs. T being learned via *gradient descent*. For the latter, we show the learned T of the last training epoch. We use the ResNet-50 for NYUv2 and Cityscapes, and ResNet-18 for CelebA.

Method	NYUv2		Cityscapes		CelebA	
	Δ_m	T	Δ_m	T	Δ_m	T
grid search	-9.8 ± 0.2	2.0	0.3 ± 1.3	28.0	-4.0 ± 0.2	100.0
gradient descent	-10.0 ± 0.4	3.8	3.0 ± 1.0	26.3	31.7 ±4.3	13.1



(a) Segmentation (b) Depth (c) Normal

Fig. 5 Weight development of various MTO algorithms for the NYUv2 tasks using the SegNet

than using a *grid search* on the NYUv2 dataset, and only slightly worse results for Cityscapes. Remarkably, the final learned value of T closely matches the value found through grid search for both datasets. However, for CelebA, we can observe a substantial performance decrease when learning T compared to tuning it. We suspect that CelebA is the most difficult of the three datasets—indicated by a large number of tasks (40) and the large optimal value of T (100). Since the learned T value does not come above 13.1, the resulting task weights are far from optimal, yielding a bad overall Δ_m value. In summary, learning T via gradient descent shows promising results but fails in certain settings, especially when a very large T is required.

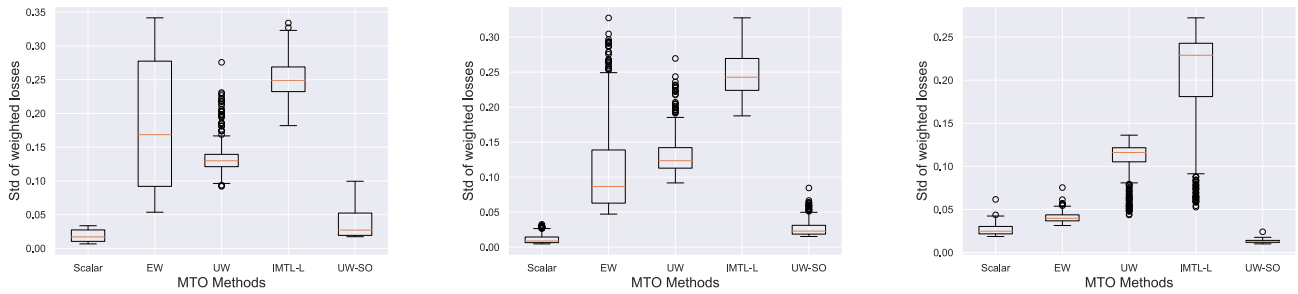
Oscillation of MTO methods The authors of the IMTL-L approach argue that weighting by the inverse of the loss results in "severe oscillations" (Liu, Li, Kuang, Xue, Chen, Yang, Liao, and Zhang, 2021, sec. 3.2). Our experiments confirm that the gradient-based methods IMTL-L and UW have smoother loss weight updates than UW-O and UW-SO (see Figure 5). However, we argue that oscillations of the task weight ω_k itself are not problematic as it is the weighted loss $\omega_k L_k$, which determines the parameter update. Looking at the standard deviation over weighted losses from all batches within one epoch (Figure 6), it turns out that UW-SO, and Scalarization are less affected by oscillations than IMTL-L (UW-O has a standard deviation of 0 by design).

Influence of batch size Since UW-O and UW-SO compute task weights based on the loss of each individual batch, we examine how batch size influences test performance on Cityscapes, NYUv2, and CelebA (see Table 6). Across all datasets, UW, UW-O, and UW-SO show sensitivity to batch size, with UW-SO being the most affected. For example, on CelebA and Cityscapes, UW-SO exhibits substantially larger performance fluctuations compared to UW and UW-O. In contrast, UW-O is less susceptible to changes in batch size, indicating that the softmax-based weighting in UW-SO may amplify this sensitivity. Importantly, LR, WD, and T were kept constant across all batch sizes, which may further contribute to the observed sensitivity.

Influence of learning rate and weight decay Building on prior work in MTL—where LR and WD were often not tuned individually per method (e.g., Liu et al. (2021, 2019); Yu et al. (2020); Navon et al. (2022))—we empirically highlight the importance of tuning both hyperparameters to ensure a fair comparison. As shown in Figure 7, different loss weighting strategies require distinct learning rates to reach optimal performance, even on the same dataset. This effect is particularly pronounced on Cityscapes and CelebA, where the optimal LR varies significantly across weighting methods. Figure 8 further illustrates the interaction between LR and WD on the NYUv2 dataset. While both hyperparameters influence performance, we observe that for sufficiently small WD values (i.e., ≤ 0.0001), the variation in performance due to changes in WD is minimal when LR is fixed. In summary, LR plays a crucial role in determining performance, whereas WD has a comparatively minor effect when appropriately constrained.

6 Conclusion

In this work, we investigated Uncertainty Weighting (UW), examining its properties, underlying assumptions, and limita-

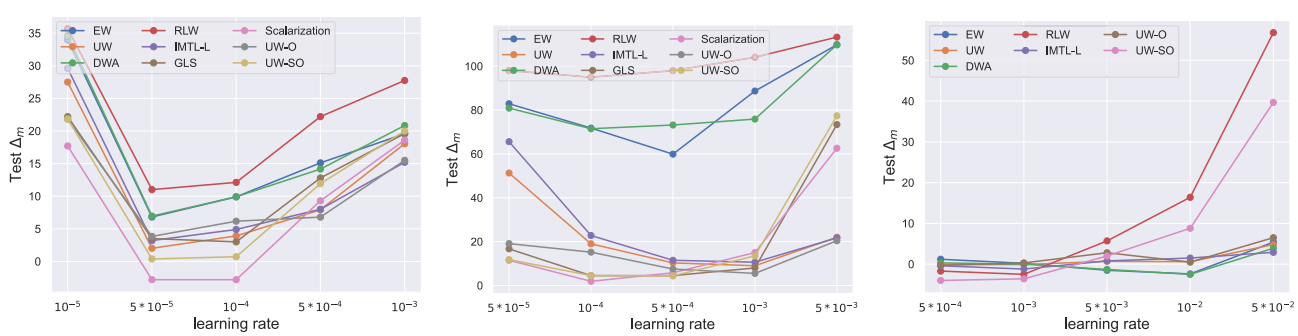


(a) Segmentation (b) Depth (c) Normal

Fig. 6 Boxplots over the std. dev. of the weighted NYUv2 (SegNet) task losses $\omega_k L_k$ of all batches from one epoch. Std. dev. over one epoch is one observation

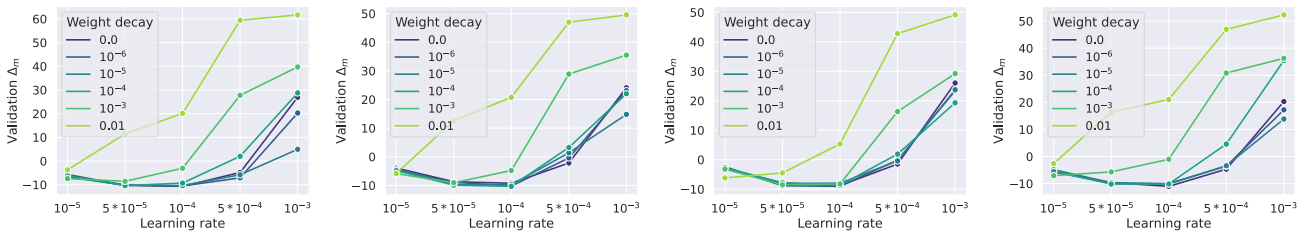
Table 6 Δ_m scores across different batch sizes—averaged over three seeds—on the Cityscapes, NYUv2, and CelebA test datasets. The experiments are conducted for a) Cityscapes with SegNet, b) NYUv2 with ResNet-50, and c) CelebA with ResNet-18, with the respective hyperparameters specified in Table 1, Table 2, and Table 3. Test performance is reported at the epoch with the best validation performance, selected independently for each method, batch size, and seed. The *Variation* indicates the performance range (max-min) across batch sizes for each method, reflecting its sensitivity to this hyperparameter.

Dataset	Batch Size	UW	UW-O	UW-SO
Cityscapes	2	15.14 ± 3.27	**9.88 ± 3.55**	17.42 ± 2.47
	4	12.30 ± 2.76	**10.48 ± 3.16**	11.83 ± 3.77
	8	8.90 ± 5.81	**4.45 ± 2.54**	5.49 ± 0.59
	16	5.66 ± 0.79	4.06 ± 1.44	**2.48 ± 0.79**
	32	8.69 ± 1.58	4.91 ± 0.83	**2.79 ± 0.38**
	Variation	9.48	6.42	14.94
NYUv2	2	-6.94 ± 0.32	-9.32 ± 0.09	**-9.57 ± 0.32**
	4	-9.37 ± 0.09	-11.84 ± 0.06	**-12.78 ± 0.38**
	8	-9.12 ± 0.07	-12.58 ± 0.03	**-12.96 ± 0.52**
	16	-7.58 ± 0.32	-12.00 ± 0.09	**-12.26 ± 0.10**
	Variation	2.43	3.26	3.39
CelebA	32	**-2.15 ± 0.80**	2.21 ± 0.96	13.66 ± 2.82
	64	**-1.66 ± 0.91**	1.44 ± 1.52	4.43 ± 2.31
	128	-1.71 ± 0.59	1.61 ± 0.72	**-1.90 ± 0.41**
	256	0.25 ± 1.69	0.33 ± 0.87	**-3.72 ± 1.49**
	512	2.85 ± 5.37	1.34 ± 1.96	**-2.48 ± 1.14**
	Variation	5.00	1.88	17.38



(a) NYUv2 (b) Cityscapes (c) CelebA

Fig. 7 Δ_m scores on the test data for different choices of the learning rate with a fixed weight decay (for (a) and (b): WD $= 10^{-5}$; for (c): WD $= 10^{-4}$). We show results for a) NYUv2 with SegNet, b) Cityscapes with SegNet, and c) CelebA with ResNet-18



(a) Scalarization (b) GLS (c) UW (d) UW-SO

Fig. 8 Results on NYUv2 with ResNet-50 backbone when varying learning rate and weight decay

tions. Building on these insights, we introduced *Soft-Optimal Uncertainty Weighting* (UW-SO), a novel method for loss weighting in Multi-Task Learning (MTL). UW-SO is derived from the analytical solution of UW and leverages the tempered softmax function applied to the inverse of task losses to determine task weights. This approach addresses several of UW's limitations, such as mitigating overfitting and explicitly modeling task interdependencies. Through an extensive benchmark involving three datasets, up to four architectures per dataset, and eight loss weighting strategies (focusing exclusively on pure loss weighting rather than gradient-based methods), we demonstrated that UW-SO consistently outperforms existing methods. The only approach that occasionally matches UW-SO is brute-force Scalarization; however, its reliance on extensive hyperparameter tuning renders it impractical for settings with many tasks. Additionally, our experiments highlight the critical role of temperature selection in the tempered softmax. Our proposed method for learning the temperature parameter T via gradient descent—instead of tuning T via grid search—reveals already promising results.

To further improve performance, future work could explore dynamic strategies for adjusting the temperature parameter T based on the model's current training behavior. Inspired by concepts from Reinforcement Learning and Curriculum Learning (Bengio, Louradour, Collobert, and Weston, 2009), T could be adapted over time to balance exploitation—using lower values of T to prioritize a few low-loss tasks—and exploration—using higher values to assign more uniform weights across tasks regardless of their loss. It is also important to note that UW-SO does not fully address all identified limitations of UW. For instance, the lack of theoretical justification for the cosine similarity loss remains an open issue in both methods. Nevertheless, UW-SO applies a consistent weighting scheme across tasks in practice. Therefore, to further validate its broader applicability, future research should investigate the generalization capabilities of UW-SO on more real-world datasets involving diverse types of task loss functions.

Acknowledgements This work is partially supported by the Federal Ministry of Research, Technology and Space (BMFTR) and by the Federal Ministry for Economic Affairs and Energy (BMWE) of Germany. We thank Cathrin Elich and Andrej Tschalzev for their helpful feedback.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability The datasets analyzed during the current study are available at: Cityscapes, NYUv2,CelebA

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Badrinarayanan, V., Kendall, A. & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 39(12), pp. 2481–2495. IEEE.

Baijiong, L., Feiyang, Y. & Z. Yu. (2021). A closer look at loss weighting in multi-task learning. *ArXiv preprint* arXiv: abs/2111.10603 .

Bengio, Y., Louradour, J., Collobert, R. & J. Weston (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48.

Caccia, M., Caccia, L., Fedus, W., Larochelle, H., Pineau, J., & Charlin, L. (2019). Language gans falling short.

Caruana, R. (1997). *Multitask learning. Machine learning, 28*(1), 41–75.

Chang, J., Zhang, X., Guo, Y., Meng, G., Xiang, S. & Pan, C. (2019). Differentiable architecture search with ensemble gumbel-softmax. *arXiv preprint* arXiv:1905.01786 .

Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F. & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818.

Chen, Z., Badrinarayanan, V., Lee, C., & Rabinovich, A. (2018). Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In J. G. Dy and A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, Volume 80 of *Proceedings of Machine Learning Research*, pp. 793–802. PMLR.

Chen, Z., Ngiam, J., Huang, Y., Luong, T., Kretzschmar, H., Chai, Y., & Anguelov, D. (2020). Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Chennupati, S., Sistu, G., Yogamani, S., & A Rawashdeh, S. (2019). Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 0–0.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. & Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 3213–3223. IEEE Computer Society.

Duong, L., Cohn, T., Bird, S., & Cook, P. (2015). Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Beijing, China, pp. 845–850. Association for Computational Linguistics.

Elich, C., Kirchdorfer, L., Koehler, J.M. & Schott, L. (2024). Examining common paradigms in multi-task learning. In *Pattern Recognition - 46th DAGM German Conference, DAGM GCPR 2024, Munich, Germany, September 10-13, 2024, Proceedings*, Volume 15297 of *Lecture Notes in Computer Science*. Springer.

Fan, C., Chen, W., Tian, J., Li, Y., He, H., & Jin, Y. (2022). Maxgnr: A dynamic weight strategy via maximizing gradient-to-noise ratio for multi-task learning. Lecture Notes in Computer ScienceIn L. Wang, J. Gall, T. Chin, I. Sato, & R. Chellappa (Eds.), *Computer Vision - ACCV 2022–16th Asian Conference on Computer Vision, Macao, China, December 4–8, 2022, Proceedings, Part I* (Vol. 13841, pp. 523–538). Springer.

Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., & Finn, C. (2021). Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems, 34*, 27503–27516.

Guo, C., Pleiss, G., Sun, Y. & Weinberger, K.Q. (2017). On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR.

Guo, M., Haque, A., Huang, D.A., Yeung, S., & Fei-Fei, L. (2018). Dynamic task prioritization for multitask learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Heuer, F., Mantowsky, S., Bukhari, S.S. & Schneider, G. (2021). Multitask-centernet (MCN): efficient and diverse multitask learning using an anchor free approach. In *IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, BC, Canada, October 11-17, 2021*, pp. 997–1005. IEEE.

Hinton, G., Vinyals, O. & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint* arXiv:1503.02531 .

Hu, R. & Singh, A. (2021). Unit: Multimodal multitask learning with a unified transformer. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 1419–1429. IEEE.

Hu, Y., Xian, R., Wu, Q., Fan, Q., Yin, L., & Zhao, H. (2023). Revisiting scalarization in multi-task learning: A theoretical perspective. *Advances in Neural Information Processing Systems, 36*, 48510–48533.

Ishihara, K., Kanervisto, A., Miura, J., & Hautamäki, V. (2021). Multi-task learning with attention for end-to-end autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021, virtual, June 19-25, 2021*, pp. 2902–2911. Computer Vision Foundation / IEEE.

Jang, E., Gu, S., & Poole, B. (2016). Categorical reparameterization with gumbel-softmax.

Javaloy, A. & Valera, I. (2022). Rotograd: Gradient homogenization in multitask learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Kendall, A. & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5574–5584.

Kendall, A., Gal, Y. & Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 7482–7491. IEEE Computer Society.

Kirchdorfer, L., Elich, C., Kutsche, S., Stuckenschmidt, H., Schott, L. & Köhler, J.M. (2024). Analytical uncertainty-based loss weighting in multi-task learning. In *Pattern Recognition - 46th DAGM German Conference, DAGM GCPR 2024, Munich, Germany, September 10-13, 2024, Proceedings*, Volume 15297 of *Lecture Notes in Computer Science*. Springer.

Kurin, V., Palma, A.D., Kostrikov, I., Whiteson, S. & Mudigonda, P.K. (2022). In defense of the unitary scalarization for deep multi-task learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, December, virtual*.

Lakkapragada, A., Sleiman, E., Surabhi, S., & Wall, D. P. (2023). Mitigating negative transfer in multi-task learning with exponential moving average loss weighting strategies (student abstract). In B. Williams, Y. Chen, & J. Neville (Eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7–14, 2023* (pp. 16246–16247). AAAI Press.

Li, D., Ju, H., Sharma, A. & Zhang, H.R. (2023). Boosting multitask learning on graphs through higher-order task affinities. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1213–1222.

Li, D., A. Sharma, & H.R. Zhang (2024). Scalable multitask learning using gradient-based estimation of task affinity. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1542–1553.

Liang, S., Li, Y., & Srikant, R. (2018). Enhancing the reliability of out-of-distribution image detection in neural networks.

Lin, B., Jiang, W., Ye, F., Zhang, Y., Chen, P., Chen, Y.C., Liu, S. & Kwok, J.T. (2023). Dual-balancing for multi-task learning. *arXiv preprint* arXiv:2308.12029 .

Lin, B. & Zhang, Y. (2023). LibMTL: A python library for multi-task learning. *Journal of Machine Learning Research* .

Liu, B., Feng, Y., Stone, P. & Liu, Q. (2023). FAMO: fast adaptive multitask optimization. *CoRR* arXiv: abs/2306.03792.

Liu, B., Liu, X., Jin, X., Stone, P. & Liu, Q. (2021). Conflict-averse gradient descent for multi-task learning. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems 34:*

Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, pp. 18878–18890.

Liu, L., Li, Y., Kuang, Z., Xue, J., Chen, Y., Yang, W., Liao, Q. & Zhang, W. (2021). Towards impartial multi-task learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Liu, S., E. Johns, & A.J. Davison (2019). End-to-end multi-task learning with attention. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 1871–1880. Computer Vision Foundation / IEEE.

Liu, Z., Luo, P., Wang, X. & Tang, X. (2015). Deep learning face attributes in the wild. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 3730–3738. IEEE Computer Society.

Maninis, K., Radosavovic, I. & Kokkinos, I. (2019). Attentive single-tasking of multiple tasks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 1851–1860. Computer Vision Foundation / IEEE.

Mao, Y., Wang, Z., Liu, W., Lin, X. & Xie, P. (2022). MetaWeighting: Learning to weight tasks in multi-task learning. In *Findings of the Association for Computational Linguistics: ACL 2022*, Dublin, Ireland, pp. 3436–3448. Association for Computational Linguistics.

Misra, I., Shrivastava, A., Gupta, A. & Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 3994–4003. IEEE Computer Society.

Nathan Silberman, Derek Hoiem, P.K. & R. Fergus (2012). Indoor segmentation and support inference from rgbd images. In *Proceedings of the European conference on computer vision (ECCV)*.

Navon, A., Shamsian, A., Achituve, I., Maron, H., Kawaguchi, K., Chechik, G. & Fetaya, E. (2022). Multi-task learning as a bargaining game. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato (Eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, Volume 162 of *Proceedings of Machine Learning Research*, pp. 16428–16446. PMLR.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *ArXiv preprint* arXiv:abs/1706.05098 .

Sener, O. & Koltun, V. (2018). Multi-task learning as multi-objective optimization. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 525–536.

Senushkin, D., Patakin, N., Kuznetsov, A. & Konushin, A. (2023). Independent component alignment for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20083–20093.

Shi, G., Li, Q., Zhang, W., Chen, J., & Wu, X.M. (2023). Recon: Reducing Conflicting Gradients From the Root For Multi-Task Learning. In *The Eleventh International Conference on Learning Representations*.

Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J. & Savarese, S. (2020). Which tasks should be learned together in multi-task learning? In *International conference on machine learning*, pp. 9120–9132. PMLR.

Tan, Q., Liu, N., Zhao, X., Yang, H., Zhou, J., & Hu, X. (2020). Learning to hash with graph neural networks for recommender systems. *In Proceedings of The Web Conference, 2020*, 1988–1998.

Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., & Van Gool, L. (2021). *Multi-task learning for dense prediction tasks: A survey*. In IEEE transactions on pattern analysis and machine intelligence: IEEE.

Vasu, P.K.A., Saxena, S. & Tuzel, O. (2021). Instance-level task parameters: A robust multi-task weighting framework. *arXiv preprint* arXiv:2106.06129 .

Xin, D., Ghorbani, B., Garg, A., Firat, O., & Gilmer, J. (2022). Do current multi-task optimization methods in deep learning even help? In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, December, virtual*.

Xu, D., Ouyang, W., Wang, X. & Sebe, N. (2018). Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 675–684. IEEE Computer Society.

Yang, Y. & Hospedales, T.M. (2016). Trace norm regularised deep multi-task learning. *ArXiv preprint* arXiv:abs/1606.04038 .

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K. & Finn, C. (2020). Gradient surgery for multi-task learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.