CVPR
#8347

Towards Calibrated Gradient-based Multi-Task Learning

Anonymous CVPR submission

Paper ID 8347

Abstract

Multi-task learning (MTL) enhances generalization and efficiency by jointly training multiple related tasks within a shared model. In recent years, numerous MTL paradigms have been proposed, among which gradient-based approaches have gained increasing popularity due to their strong performance and direct access to the optimization process. However, a key challenge in gradient-based MTL, i.e., gradient conflict, often hinders balanced learning. Importantly, we observe that this issue can be indeterministic due to gradient variance, which arises from factors such as small batch sizes, etc. In this paper, we provide the first study on how gradient variance influences the performance of gradient-based MTL methods. Our empirical analysis reveals that elevated gradient variance results in unstable updates and significantly impairs MTL performance. To mitigate this, we propose `VarGrad`*, a lightweight and general framework that reduces gradient instability through iterative gradient correction and selectively schedules joint updates based on task-specific loss dynamics. Extensive experiments across multiple mainstream MTL benchmarks demonstrate that* `VarGrad` *achieves an average performance gain of 24.3%, while remaining broadly compatible with existing MTL methods.*

1. Introduction

Multi-task learning (MTL) aims to improve generalization and training efficiency by jointly optimizing multiple related tasks within a shared model. By leveraging shared representations, MTL has achieved notable success across domains such as computer vision [35], natural language processing [2], and speech [6]. MTL enables leveraging complementary task information and reduces reliance on large single-task datasets, improving robustness and efficiency.

A core challenge in MTL is the presence of gradient conflicts, where gradients from different tasks may point in conflicting directions, which can partially cancel each other out during parameter updates, reducing optimization effectiveness. To address this, numerous gradient-based methods have been developed, including gradient projection [33], Pareto optimization [30], Nash bargaining [27] and fair allocation [1], etc. These approaches fundamentally rely on the assumption that the gradients from each task can be accurately estimated—particularly their directions—so that appropriate multi-task updates can be computed.

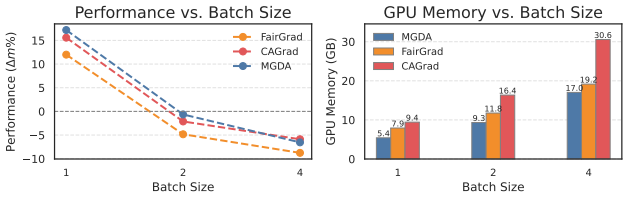

Figure 1. **Task performance and GPU memory usage on NYUv2 across batch sizes.** Left: Performance $\Delta m\%$ (Eqn. 16, lower is better) for MGDA, FairGrad, and CAGrad. Right: Corresponding memory usage. Larger batches yield better performance but significantly higher memory costs. It is important to note that all models are trained for 200 epochs, regardless of the batch size. This implies that models with larger batch sizes undergo fewer update steps during training.

However, this assumption does not always hold in practice. Gradient estimates are often noisy and may deviate from the true descent directions. Such inaccuracies in gradient estimation can reduce optimization stability and limit MTL performance. To study this, we manipulate gradient variance by controlling the batch size and empirically analyze how the variance of gradient estimates across training iterations affects optimization dynamics and final task performance (Section 3).

Limitations of Existing Methods. A straightforward solution to reduce gradient variance is to increase the batch size, which typically enhances gradient stability and improves training outcomes. As shown in Figure 1, enlarging the batch size yields notable performance gains across several representative MTL methods, e.g., MGDA [30], CAGrad [20], and FairGrad [1]. This confirms that more stable gradient estimates—enabled by larger batches—facilitate better MTL optimization. However, such improvements come at the cost of significantly increased GPU memory

consumption. Even modest batch size increments introduce substantial memory overhead. This trade-off highlights a critical limitation of current gradient-based MTL methods: although larger batches can mitigate gradient noise, they are often infeasible due to hardware constraints. A commonly adopted alternative is gradient accumulation [17], which alleviates memory usage by simulating large-batch training over multiple iterations. Nevertheless, this technique slows convergence and suffers from the same limitation of fewer parameter update steps, thereby reducing training efficiency and overall effectiveness.

Proposed Method. Motivated by the need for more reliable and efficient gradient estimation in MTL, we propose `VarGrad`, a lightweight and general framework that enhances the stability of gradient-based optimization. Rather than relying on large batch sizes to reduce gradient variance, `VarGrad` explicitly calibrates gradient directions by variance reduction across iterations to mitigate estimation noise. In addition, it adaptively schedules joint optimization updates based on task-specific loss dynamics to further improve efficiency. `VarGrad` is algorithm-agnostic and can be seamlessly integrated with a wide range of existing gradient-based MTL methods. Our main contributions are as follows:

- We propose `VarGrad`, a lightweight and algorithm-agnostic framework that mitigates inter-iteration gradient instability via temporal calibration, and empirically show—through the first systematic study—that reveal why existing methods underperform in resource-constrained settings.
- We introduce an adaptive scheduling mechanism that selectively triggers joint optimization based on task loss dynamics, reducing computational overhead by up to 55% while maintaining performance quality.
- We conduct comprehensive experiments across diverse benchmarks (classification, regression, dense prediction) and demonstrate `VarGrad`'s compatibility with major MTL methods. Results show consistent improvements with an average performance gain of 24.3% and enhanced training efficiency.

2. Related Work

2.1. Multi-Task Learning

Current MTL approaches can be broadly classified into two main categories: architecture-based and optimization-based methods. Architecture-based methods include a range of paradigms such as hard parameter sharing [13, 18], soft parameter sharing [11, 32], modulation and adapters [12, 23], and mixture of experts (MoE)[4, 8], among others. These methods primarily focus on designing shared or partially shared model structures to facilitate knowledge transfer across tasks.

In contrast, optimization-based approaches focus on the learning dynamics rather than architectural design. These methods aim to optimize all tasks jointly by managing gradient interactions, thereby promoting shared representations across tasks. A classical example is MGDA [30], which formulates MTL as a multi-objective optimization problem and solves it via the Frank-Wolfe algorithm [14] to obtain a common descent direction with minimal norm. PCGrad [33] mitigates gradient conflicts by projecting each task's gradient to remove conflicting components. CAGrad [20] introduces a weighted objective to balance global convergence and Pareto efficiency. Nash-MTL [27] models task interaction as a bargaining game, aiming to find a mutually beneficial update direction that ensures fair progress across tasks. MoCo [9] refines gradient aggregation by correcting biased directions using historical gradient tracking. Finally, FairGrad introduces the conception of fair resource allocation in network to MTL to facilitate the balance progress of individuals.

Despite their methodological differences, these approaches share a key assumption: that task gradients are sufficiently accurate and stable to guide reliable optimization. However, in practice, various factors—including limited computational resources and weak supervision—can cause gradient estimates to be noisy or inaccurate, which undermines the effectiveness of gradient correction and weighting strategies.

2.2. Gradient Variance Reduction

As gradient variance plays a crucial role in the effectiveness of gradient-based optimization, numerous techniques have been developed to reduce this variance and stabilize training. These variance reduction methods broadly fall into two categories, each employing distinct strategies to improve gradient estimates.

History-based Periodic Correction. Methods in this category reduce variance by periodically computing or storing accurate gradient information to correct noisy stochastic gradients. For example, SAG [29] maintains a running average of past gradients to stabilize updates, while SVRG [15] intermittently calculates the full gradient and uses it as a control variate to adjust mini-batch gradients, thereby lowering variance.

Recursive Momentum Estimation. Alternatively, recursive momentum-based methods continuously smooth gradient estimates without the overhead of full gradient computations. STORM [7] leverages a recursive momentum estimator to adaptively reduce gradient variance during training, and MARS [34] builds upon this idea by combining momentum with bias-variance tradeoff mechanisms, offering

CVPR
#8347

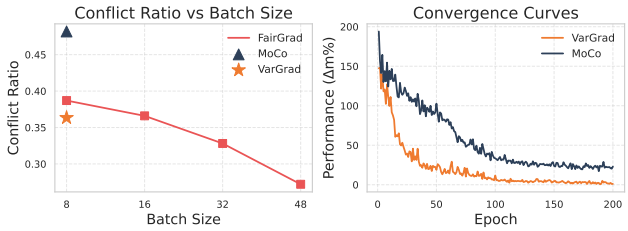

Figure 2. **Conflict Ratio vs. Batch Size on Cityscapes.** Left: Conflict decreases with batch size under FairGrad. MoCo increases conflict, while `VarGrad` effectively suppresses it. Right: Training curves reveal that `VarGrad`, by suppressing conflict, converges substantially faster than MoCo.

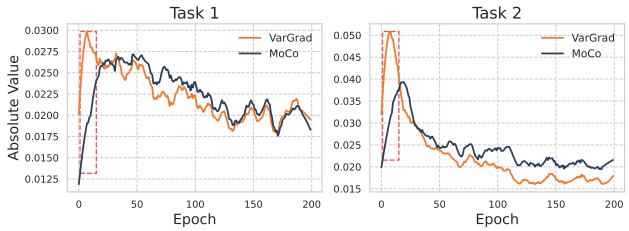

Figure 3. **Task loss progression over training epochs on Cityscapes.** Orange curves correspond to `VarGrad`, and blue curves to MoCo. Left and right subplots display results for Task 1 and Task 2, respectively. The red dashed box highlights the early training phase (*Epochs 1–15*), where `VarGrad` shows more pronounced improvements.

an efficient and scalable solution for large-scale optimization.

While these variance reduction techniques have proven effective in standard single-task optimization, they are typically designed to operate on the final gradients used for parameter updates—often after gradient aggregation in multi-task settings. In the context of multi-task learning, most existing approaches apply optimizers like Adam directly to the aggregated or modified gradients, without explicitly considering the variance or misalignment in task-specific gradients prior to aggregation.

2.3. Discussion with Counterparts

To the best of our knowledge, MoCo [9] is the most closely related work to our proposed method, as it also addresses the issue of gradient bias in MTL. However, our approach differs from MoCo in three key aspects: (1) Motivation: We empirically identify the presence of gradient variance and establish its impact on MTL performance, a perspective not explicitly explored by MoCo. (2) Methodology: While MoCo leverages first-order momentum (as in SGD with Momentum) to calibrate gradients, it does not explicitly reduce gradient variance. In contrast, our method incorporates temporal gradient differences to directly mitigate variance and further introduces an adaptive update schedule to improve training efficiency. (3) Empirical Results: Extensive experiments validate the effectiveness and plug-and-play nature of our approach, while MoCo demonstrates limited competitiveness across various benchmarks.

3. Motivation and Observation

Gradient-based MTL methods assume that task gradients accurately reflect true optimization directions. However, this assumption fails under high variance—particularly with small batch sizes—rendering gradient-based conflict resolution or reweighting unreliable due to noisy signals. Momentum methods like MoCo smooth gradients temporally but do not reduce variance and can even amplify noise, according to our observations. We argue that reducing gradient variance is a prerequisite for effective multi-task optimization. To support this, we analyze the impact of variance from two perspectives: (1) the gradient conflict ratio under varying variance levels, and (2) the effect of variance on per-task optimization progress.

3.1. Gradient Conflict Reflects Gradient Instability

To assess the role of gradient variance in inter-task interference, we analyze the gradient conflict ratio (CR) —the proportion of steps where gradients from two tasks point in opposing directions:

$$\mathrm{CR}(t_i, t_j) = \frac{1}{K} \sum_{k=1}^{K} \mathbb{I}\left[\cos\left(\mathbf{g}_{t_i}^k, \mathbf{g}_{t_j}^k\right) < 0\right], \qquad (1)$$

where $\mathbf{g}_{t_i}^k$ and $\mathbf{g}_{t_j}^k$ denote the gradients of tasks t_i and t_j at the k-th training step, and $\mathbb{I}[\cdot]$ is the indicator function.

We vary the batch size to create a controlled setting for analyzing gradient variance, as larger batches typically produce more accurate gradient estimates. As shown in Figure 2, the baseline FairGrad exhibits a decreasing CR as batch size increases. This trend supports the intuition that high-variance gradients—arising from smaller batches—are more likely to point in conflicting directions, thereby amplifying task interference and destabilizing optimization. In other words, gradient variance is a potential contributor to gradient conflict. These findings validate our core motivation: gradient variance may artificially inflate conflict levels that do not reflect the true learning dynamics. As a result, MTL algorithms often strike a delicate balance to avoid sacrificing any individual task (as illustrated in Figure 3), which can lead to overly conservative updates and limited overall progress.

Moreover, we investigate the effects of MoCo and `VarGrad` under a standard small-batch setting (batch size = 8). We find that MoCo further increases the CR, suggesting that naive momentum may amplify gradient noise

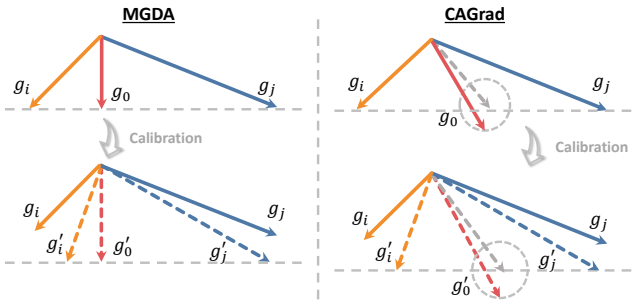

Figure 4. **Illustration of `VarGrad`'s effect.** Task-specific gradients are calibrated and then combined to produce a multi-task update.

rather than mitigate conflicts. In contrast, `VarGrad` consistently reduces CR—even in low-batch regimes—without incurring additional computational cost.

3.2. Variance Reduction and Task Progress

To substantiate the importance of accurate gradient estimation for MTL, we further explore the relative improvement rate of each task's loss over training epochs [21], defined as

$$r = \frac{\ell_{t_i}^k - \ell_{t_i}^{k+1}}{\ell_{t_i}^k}, \tag{2}$$

where $\ell_{t_i}^k$ denotes the loss of task t_i at epoch k. This metric reflects the true task progress and the effectiveness of the optimization direction.

Figure 3 illustrates the per-task relative improvement rates during training. In the early phase—when gradient variance is high—`VarGrad` achieves consistently faster progress than MoCo. This highlights a core issue in MTL: under noisy gradients, the optimization direction may deviate from true task improvement, making balanced learning unreliable. By correcting gradient variance, `VarGrad` enables more stable and coordinated task progress. While both methods eventually converge to similar final losses, `VarGrad` reaches competitive performance much earlier as shown in the right subplot of Figure 2, demonstrating practical gains in training efficiency and robustness.

4. Methodology

4.1. Overview of `VarGrad`

We propose `VarGrad`, a lightweight and general framework aimed at enhancing the accuracy and efficiency of gradient-based MTL. Our method is motivated by two key empirical observations: (1) gradient estimates for individual tasks are often biased or unreliable under practical training conditions; (2) frequent per-step MTL updates may be unnecessary, as they are computationally expensive, and using the mean gradient can suffice in scenarios with moderate

Algorithm 1: `VarGrad`: Calibration for Gradient-based MTL

Input: Model f, data loader $\mathcal{D}$, momentum parameters $\{\beta^k\}$, learning rate η
Output: Updated model parameters θ

Initialize: For each task $t = 1, \ldots, T$, set $\mathbf{g}_t^{k-1} \leftarrow 0$, $\mathbf{m}_t^k \leftarrow 0$
foreach *batch* $(x, \{y_t\}_{t=1}^T)$ *in* $\mathcal{D}$ **do**
  foreach *task* $t = 1$ **to** T **do**
    Compute task loss $\mathcal{L}_t \leftarrow \mathcal{L}(f(x), y_t)$;
    Compute raw gradient $\mathbf{g}_t^k \leftarrow \nabla_\theta \mathcal{L}_t$;
    // Iterative Gradient Correction
    $\mathbf{c}_t^k = \mathbf{g}_t^k + \cdot \frac{\beta^k}{1-\beta^k} \cdot (\mathbf{g}_t^k - \mathbf{g}_t^{k-1})$;
    $\mathbf{m}_t^k \leftarrow \beta^k \cdot \mathbf{m}_t^k + (1-\beta^k) \cdot \mathbf{c}_t^k$;
    // Store gradient for next iteration
    $\mathbf{g}_t^{k-1} \leftarrow \mathbf{g}_t^k$;
  // Merge stabilized gradients
  $\mathbf{g}_{\text{merged}}^k \leftarrow \texttt{MTLMerge}(\{\mathbf{m}_t^k\}_{t=1}^T)$;
  $\theta \leftarrow \theta - \eta \cdot \mathbf{g}_{\text{merged}}^k$;

task imbalance. To address these challenges, `VarGrad` introduces two complementary components:

- A ***gradient stabilization*** mechanism (illustrated in Figure 4) that reduces inter-iteration variance by smoothing task gradients prior to multi-task optimization. Instead of using raw stochastic gradients that may be inaccurate or inconsistent, we maintain temporally calibrated gradient estimates that leverage historical information to suppress fluctuations and improve estimation accuracy.
- A ***selective multi-task optimization*** strategy that adaptively determines when to perform joint updates based on task-specific loss dynamics. By monitoring loss variation as the imbalance proxy, the method identifies stable phases where MTL optimization can be safely skipped, thereby reducing computational overhead while maintaining performance.

Together, these components enable `VarGrad` to produce more accurate and stable optimization signals while reducing redundant or noisy multi-task updates. Importantly, `VarGrad` is model- and algorithm-agnostic, allowing seamless integration with a wide range of existing gradient-based MTL methods (e.g., CAGrad, FairGrad, etc). In the following sections, we provide a detailed description of each component.

4.2. Gradient Stabilization

As mentioned above, our counterpart MoCo has employed a momentum-based smoothing approach to calibrate the task

gradients as follow:

$$\mathbf{m}^{k+1} \leftarrow \beta \cdot \mathbf{m}^k + (1-\beta) \cdot \mathbf{c}^k, \tag{3}$$

where $\beta \in [0, 1)$ is the hyper-parameter, $\mathbf{m}^k$ denotes the estimated first-order momentum at step k, and $\mathbf{c}^k$ is the (corrected) gradient of step k after task-wise calibration. Specifically, $\mathbf{c}^k$ is set as $\mathbf{g}^k$ in MoCo. Although such a moving average approach has the potential of noise suppression, it has not explicit guarantee for gradient variance reduction, which is important for the conflict issue and individual progress of MTL. To improve the reliability of task-specific gradients prior to multi-task optimization, we propose an iterative gradient correction mechanism that stabilizes gradients by leveraging information from consecutive iterations. For each task t, given the current gradient $\mathbf{g}_t^k$ and the previous gradient $\mathbf{g}_t^{k-1}$, the corrected gradient $\mathbf{c}_t^k$ is computed as

$$\mathbf{c}_t^k = \mathbf{g}_t^k + \frac{\beta_t}{1-\beta_t} \cdot (\mathbf{g}_t^k - \mathbf{g}_t^{k-1}). \tag{4}$$

The term $(\mathbf{g}_t^k - \mathbf{g}_t^{k-1})$ measures the change in the task-t gradient between consecutive iterations, implicitly encoding local curvature information of the loss landscape—similar to a finite-difference approximation of the Hessian—while avoiding explicit computation of second-order derivatives. Mathematically, assuming the optimization trajectory is smooth (i.e., $\|\theta_t - \theta_{t-1}\|$ is small), the gradient difference can be linearized via the Hessian $H(\theta_{t-1})$ [26]:

$$\mathbf{g}^k - \mathbf{g}^{k-1} \approx H(\theta^{k-1})(\theta^k - \theta^{k-1}), \tag{5}$$

where the approximation error scales quadratically with $\|\theta_t - \theta_{t-1}\|$. This connection to curvature enables implicit second-order adaptation of update directions and step sizes. Additionally, the term acts as a noise-canceling mechanism in stochastic optimization. Under the common assumption that gradient noise $\boldsymbol{\epsilon}^k = \mathbf{g}^k - \nabla\mathcal{L}(\theta^k)$ exhibits short-term positive correlation (e.g., due to data/model continuity), the difference $\mathbf{g}^k - \mathbf{g}^{k-1}$ suppresses shared noise components between consecutive steps.

On the other hand, let's take a close look on how Eqn. 3 and 4, and analyze how it achieves variance reduction effect. Combining Eqn. 3 and 4, we have the following corrected iterative first order momentum:

$$\mathbf{m}^k \leftarrow \beta\mathbf{m}^{k-1} + (1-\beta)\mathbf{g}^k + \beta(\mathbf{g}^k - \mathbf{g}^{k-1}). \tag{6}$$

By rearranging this equation, we have the following:

$$\mathbf{m}^k \leftarrow \mathbf{g}^k - \beta(\mathbf{g}^{k-1} - \mathbf{m}^{k-1}). \tag{7}$$

Therefore, the variance of $\mathbf{m}_t$ is:

$$\mathrm{Var}(\mathbf{m}^k) = \mathrm{Var}(\mathbf{g}^k) + \beta^2\,\mathrm{Var}(\mathbf{z}) - 2\beta\,\mathrm{Cov}(\mathbf{g}^k, \mathbf{z}), \tag{8}$$

where $\mathrm{Var}(\cdot)$ is the variance function, while $\mathrm{Cov}(\cdot)$ is the covariance function, and $\mathbf{z} = \mathbf{g}^{k-1} - \mathbf{m}^{k-1}$. To demonstrate the variance reduction effect, we should proof $\beta^2\,\mathrm{Var}(\mathbf{z}) - 2\beta\,\mathrm{Cov}(\mathbf{g}^k, \mathbf{z}) \leq 0$. Now let $V(\beta) = \beta^2\,\mathrm{Var}(\mathbf{z}) - 2\beta\,\mathrm{Cov}(\mathbf{g}^k, \mathbf{z})$, we have the optimal β as follow:

$$\frac{\partial V(\beta)}{\partial \beta} = 2\beta\,\mathrm{Var}(\mathbf{z}) - 2\,\mathrm{Cov}(\mathbf{g}^k, \mathbf{z}) \tag{9}$$

Thus, we have the optimal $\beta^* = \frac{\mathrm{Cov}(\mathbf{g}^k, \mathbf{z})}{\mathrm{Var}(c)}$. By inserting β^* into Eqn. 8, we have:

$$\mathrm{Var}(\mathbf{m}^k) = \mathrm{Var}(\mathbf{g}^k) + \left(\frac{\mathrm{Cov}(\mathbf{g}^k, \mathbf{z})^2}{\mathrm{Var}(\mathbf{z})^2}\right)\mathrm{Var}(\mathbf{z}) \tag{10}$$
$$-2\left(\frac{\mathrm{Cov}(\mathbf{g}^k, \mathbf{z})}{\mathrm{Var}(\mathbf{z})}\right)\mathrm{Cov}(\mathbf{g}^k, \mathbf{z})$$

$$\mathrm{Var}(\mathbf{m}^k) = \mathrm{Var}(\mathbf{g}^k) - \frac{\mathrm{Cov}(\mathbf{g}^k, \mathbf{z})^2}{\mathrm{Var}(\mathbf{z})} \tag{11}$$

Considering the square of correlation coefficient $\rho^2 = \frac{\mathrm{Cov}(\mathbf{g}^k, \mathbf{z})^2}{\mathrm{Var}(\mathbf{z})\,\mathrm{Var}(\mathbf{g}^k)} \leq 1$, we have:

$$\mathrm{Var}(\mathbf{m}^k) = \mathrm{Var}(\mathbf{g}^k)(1-\rho^2) \tag{12}$$

As demonstrated, the variance reduction is achieved when β is properly set.

4.3. Selective Multi-task Optimization

We propose an acceleration strategy to reduce the computational cost of MTL. As shown in recent work [36], full MTL updates at every step are often unnecessary—especially under mild task imbalance, where simple averaging suffices. To address this, we introduce a mechanism that adaptively determines whether joint optimization is needed. By monitoring inter-task dynamics using task losses as a lightweight proxy, our method triggers multi-task updates only when imbalance exceeds a predefined threshold, and defaults to mean-gradient updates otherwise.

Imbalance Proxy. Gradient-based MTL typically requires multiple backward passes to obtain task-specific gradients before aggregation, incurring significant computation. To avoid this, we use per-task losses $\{\mathcal{L}_i\}_{i=1}^T$ as a proxy for gradient imbalance, eliminating the need for repeated backpropagation. Specifically, we apply FAMO [21] to derive task weights:

$$\{\omega_t\}_{t=1}^T = \mathrm{FAMO}(\{\mathcal{L}_t\}_{t=1}^T). \tag{13}$$

These weights are not used to scale losses, but rather to reflect relative task difficulty and inform the decision to activate multi-task optimization.

Imbalance Signal Estimation. We estimate inter-task imbalance using the ratio:

$$r = \frac{\max_t \omega_t}{\min_t \omega_t}. \tag{14}$$

A high r indicates significant imbalance—some tasks dominate or lag—where joint optimization may help. A low r suggests balanced progress, where simpler updates are sufficient.

Conditional Optimization Rule. Given a threshold $\tau > 1$, the optimization rule is defined as:

$$\mathcal{L}_{\text{step}} = \begin{cases} \texttt{MTLMethod}(\{\mathcal{L}_t\}_{t=1}^T) & \text{if } r > \tau, \\ \sum_{t=1}^T \mathcal{L}_t & \text{otherwise.} \end{cases} \tag{15}$$

Here, `MTLMethod` refers to any existing MTL optimizer (e.g., CAGrad, FairGrad). When $r \leq \tau$, we simply sum task losses, avoiding unnecessary coordination overhead.

4.4. Theoretical Analysis

We analyze the convergence of `VarGrad` under smoothness and statistical control over stabilized gradient estimators. Let the weighted objective be defined as:

$$F(\mathbf{x}) := \sum_{t=1}^T w_t F_t(\mathbf{x}),$$

where each F_t is a task-specific loss and $w_t \geq 0$, $\sum_{t=1}^T w_t = 1$. To proceed, we make the following standard assumption:

Assumption 1 (Smoothness) *Each task loss F_t is differentiable and L-smooth, i.e., for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,*

$$F_t(\mathbf{y}) \leq F_t(\mathbf{x}) + \langle \nabla F_t(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|^2.$$

Lemma 1 (Controlled Bias) *Let $\mathbf{m}_t^k$ denote the stabilized gradient estimator for task t at iteration k. Then,*

$$\mathbb{E}[\mathbf{m}_t^k] = \nabla F_t(\mathbf{x}_k) + \bar{\delta}_t^k, \quad \textit{with} \quad \|\bar{\delta}_t^k\| \leq \frac{C_1}{k^{1/3}}.$$

Lemma 2 (Controlled Variance) *The variance of $\mathbf{m}_t^k$ is bounded as:*

$$\mathbb{E}\left\|\mathbf{m}_t^k - \nabla F_t(\mathbf{x}_k) - \bar{\delta}_t^k\right\|^2 \leq \frac{C_2}{k^{2/3}}.$$

Define the aggregated update direction at iteration k:

$$\mathbf{d}^k := \sum_{t=1}^T w_t \mathbf{m}_t^k, \quad \text{and} \quad \nabla F(\mathbf{x}_k) := \sum_{t=1}^T w_t \nabla F_t(\mathbf{x}_k).$$

Let $\Delta^k := \mathbf{d}^k - \nabla F(\mathbf{x}_k)$ denote the total gradient error. From the assumptions, it follows that:

$$\left\|\mathbb{E}[\Delta^k]\right\| \leq \frac{C_1}{k^{1/3}}, \quad \mathbb{E}\left\|\Delta^k - \mathbb{E}[\Delta^k]\right\|^2 \leq \frac{C_2}{k^{2/3}}.$$

Method	CelebA	QM9
	$\Delta\mathbf{m}\% \downarrow$	$\Delta\mathbf{m}\% \downarrow$
LS	4.15	177.6
SI	7.20	77.8
RLW [19]	1.46	203.8
DWA [25]	2.40	175.3
UW [16]	3.23	108.0
MGDA [30]	14.85	120.5
PCGrad [33]	3.17	125.7
CAGrad [20]	2.48	112.8
IMTL-G [24]	0.84	77.2
Nash-MTL [27]	2.84	<u>62.0</u>
FairGrad-R [1]	<u>0.37</u>	68.6
`VarGrad`	0.29	61.8

Table 1. Results on *CelebA* and *QM9* datasets.

Theorem 1 (Convergence of VarGrad) *Under Assumption 1 and Lemma 2, let the step size be $\eta_k = \eta_0 k^{-1/3}$ for some constant $\eta_0 > 0$. Then there exists a constant $C > 0$ such that:*

$$\min_{1 \leq k \leq K} \mathbb{E}\|\nabla F(\mathbf{x}_k)\|^2 \leq \frac{C}{K^{1/3}}.$$

The proof is based on a descent argument leveraging the smoothness of F and bounding the error introduced by the biased and noisy update direction. Full derivations are provided in Appendix.

5. Evaluation

5.1. Experimental Setup

5.1.1. Tasks and Datasets.

We evaluate `VarGrad` on three MTL benchmarks spanning classification, regression, and dense prediction, each chosen to reflect key MTL challenges like task imbalance, gradient interference, and scalability. All experimental settings, including data preprocessing, model architectures, and hyperparameter configurations, follow prior work to ensure fair and reproducible comparison [1, 10, 21, 27].

Classification. (CelebA[22]): 200K+ facial images annotated with 40 binary attributes. We follow [21] to treat this as a 40-task binary classification problem.

Regression. (QM9[28]): Molecular dataset with 134K samples and 19 regression targets. We adopt the standard 110K/10K/10K train/val/test split from [10].

Dense Prediction.(NYU-v2[31], Cityscapes[5]): NYU-v2 includes semantic segmentation, depth, and normals; Cityscapes includes segmentation and depth. We use a SegNet+MTAN architecture following [21].

Method	Segmentation ↑		Depth ↓		Δm% ↓
	mIoU	Pix. Acc.	Abs. Err.	Rel. Err.	
Independent	74.01	93.16	0.0125	27.77	-
LS	75.18	93.49	0.0155	46.77	22.60
RLW	74.57	93.41	0.0158	47.79	24.37
DWA	75.24	93.52	0.0160	44.37	21.43
Uncertainty	72.02	92.85	0.0140	30.13	5.88
MGDA	68.84	91.54	0.0309	33.50	44.14
GradDrop	75.27	93.53	0.0157	47.54	23.67
PCGrad	75.13	93.48	0.0154	42.07	18.21
CAGrad	75.16	93.48	0.0141	37.60	11.58
IMTL	75.33	93.49	0.0135	38.41	11.04
MoCo	75.42	93.55	0.0149	34.19	9.90
Nash-MTL	75.41	93.66	0.0129	35.02	6.82
FAMO	74.54	93.29	0.0145	32.59	8.13
FairGrad	75.72	93.68	0.0134	32.25	5.18
VarGrad	74.75	93.34	0.0129	30.04	2.60

Table 2. **Scene understanding** (*CityScapes*, 2 tasks).

5.1.2. Evaluation Metrics.

Following prior work [21, 27], we use the mean relative performance change across tasks as our **primary** metric:

$$\Delta m\% = \frac{1}{T}\sum_{t=1}^{T}(-1)^{\delta_t}\cdot\frac{M_{m,t}-M_{b,t}}{M_{b,t}}\times 100, \qquad (16)$$

where T is the number of tasks, and $M_{m,t}$, $M_{b,t}$ denote multi-task and single-task performance on task t, respectively. $\delta_t = 1$ if lower is better (e.g., error), and 0 if higher is better (e.g., accuracy). Negative $\Delta m\%$ indicates MTL outperforms STL on average. We emphasize this aggregate measure, as the objective of MTL is to **enhance overall task synergy and generalization under shared capacity, not to overfit or specialize to a single task**.

5.2. Baselines

We compare VarGrad against strong MTL baselines spanning five key categories: (1) *Static weighting*, e.g., LS and SI, which combine task losses without gradient treatment; (2) *Dynamic weighting*, including RLW [19], DWA [25], and UW [16], which adapt weights via training dynamics or uncertainty; (3) *Gradient-based methods*, such as MGDA [30], PCGrad [33], CAGrad [20], and IMTL-G [24], which manipulate or align gradients to reduce interference; (4) *Fairness-aware optimization*, including Nash-MTL [27], FAMO [21], and FairGrad [1], which promote balanced performance via constrained or game-theoretic formulations; and (5) *Dense prediction*, such as GradDrop [3] and MoCo [9]—the latter being the most closely related to VarGrad—which address pixel-level gradient conflicts in structured output tasks.

5.3. Experimental Results

We evaluate VarGrad on four representative MTL benchmarks, covering both vision and molecular property prediction domains. For FairGrad, we report the published results on CelebA, Cityscapes, and NYUv2, as our reproduced results closely match those reported in prior work. However, for QM9, we present our own reproduction results, denoted as **FairGrad-R**, due to noticeable discrepancies between reported and reproduced performance, likely arising from implementation details or random seed effects. In the result tables, all results are averaged over three random seeds for statistical robustness. **The best scores** are highlighted in gray, and **the second-best scores** are underlined.

5.3.1. Multi-task Classification and Regression.

On CelebA with 40 binary classification tasks, VarGrad achieves the best overall performance with a $\Delta m\%$ of 0.29, outperforming all prior baselines including FairGrad-R (0.37) (Table 1). This result demonstrates the scalability and effectiveness of VarGrad in large-scale multi-task scenarios.

On QM9, VarGrad achieves a $\Delta m\%$ of 61.8, comparable to FairGrad-R (68.6) and outperforming most baselines. Since QM9 is trained with larger batch sizes than other benchmarks, the gradient variance is intuitively lower. That VarGrad still yields competitive performance under such conditions further supports our motivation that variance reduction contributes to stable multi-task optimization.

5.3.2. Dense Prediction.

On Cityscapes (Table 2), which includes semantic segmentation and depth estimation, VarGrad achieves the best depth accuracy (0.0129 and 30.04) while maintaining competitive segmentation results. The core metric $\Delta m\%$ of 2.60, the lowest among all methods, indicates effective task balancing and supports its ability to reduce dominance in joint optimization.

On NYUv2 (Table 3), a more heterogeneous three-task setting, VarGrad delivers the best surface normal prediction and remains strong on the other tasks. It achieves a $\Delta m\%$ of –5.85, outperforming both FairGrad (–4.66) and Nash-MTL (–4.04). These results reinforce that VarGrad remains superior consistency and stability in multi-task optimization, even in the presence of heterogeneous task types and learning dynamics

5.3.3. Efficiency Analysis.

We compare FairGrad with VarGrad + Selective Multi-Task Optimization (SMO) on Cityscapes. While FairGrad performs joint updates at every iteration, VarGrad + SMO reduces this to only 44.84% of steps, achieving nearly the same performance. As shown in Figure 5, this suggests that many joint updates are redundant under high gradient variance. SMO improves training efficiency by adaptively skip-

Method	Segmentation ↑		Depth ↓		Surface Normal					Δm% ↓
	mIoU	Pix. Acc.	Abs. Err.	Rel. Err.	Angle Distance ↓		Within t° ↑			
					Mean	Median	11.25	22.5	30	
Independent	38.30	63.76	0.68	0.28	25.01	19.21	30.14	57.20	69.15	-
LS	39.29	65.33	0.55	0.23	28.15	23.96	22.09	47.50	61.08	5.46
RLW	37.17	63.77	0.58	0.24	28.27	24.18	22.26	47.05	60.62	7.67
DWA	39.11	65.31	0.55	0.23	27.61	23.18	24.17	50.18	62.39	3.49
Uncertainty	36.87	63.17	0.54	0.23	27.04	22.61	23.54	49.05	63.65	4.01
MGDA	30.47	59.90	0.61	0.26	24.88	19.45	29.18	56.88	69.36	1.47
GradDrop	39.39	65.12	0.55	0.23	27.48	22.96	23.38	49.44	62.87	3.61
PCGrad	38.06	64.64	0.56	0.23	27.41	22.80	23.86	49.83	63.14	3.83
CAGrad	39.79	65.49	0.55	0.23	26.31	21.58	25.61	52.36	65.58	0.29
IMTL	39.35	65.60	0.54	0.23	26.02	21.19	26.20	53.13	66.24	-0.59
MoCo	40.30	66.07	0.5575	0.22	26.67	21.83	25.61	51.78	64.85	0.16
Nash-MTL	40.13	65.93	0.53	0.22	25.26	20.08	28.40	55.47	68.15	-4.04
FAMO	40.30	66.07	0.56	0.21	26.67	21.83	25.61	51.78	64.85	0.16
FairGrad	39.74	66.01	0.54	0.22	24.84	19.60	29.26	56.58	69.16	-4.66
VarGrad	39.10	65.38	0.54	0.22	24.44	18.89	30.72	58.05	70.20	-5.85

Table 3. **Scene understanding** (*NYUv2*, 3 tasks).

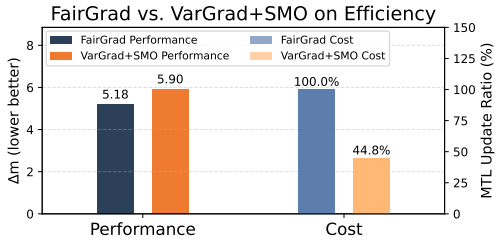

Figure 5. **Efficiency Comparison on Cityscapes.**

Method	Segmentation ↑		Depth ↓		Δm% ↓
	mIoU	Pix. Acc.	Abs. Err.	Rel. Err.	
CAGrad	75.79	93.68	0.0138	38.9	11.87
CAGrad + VarGrad	75.71	93.66	0.0137	36.57	9.61
MGDA-R	72.15	92.08	0.0136	31.64	6.64
MGDA-R + VarGrad	73.34	92.92	0.0138	30.36	5.10
FairGrad	75.72	93.68	0.0134	32.25	5.18
FairGrad + VarGrad	74.75	93.34	0.0129	30.04	2.60

Table 4. Plug-and-play verification on *CityScapes*.

ping updates during stable training phases, yielding significant computational savings with minimal accuracy loss.

5.3.4. Plug-and-Play Verification.

To assess the generality and compatibility of VarGrad, we integrate it into three representative gradient-based MTL algorithms: CAGrad, MGDA-R, and FairGrad. Table 4 reports the performance on the Cityscapes dataset. In all cases, incorporating VarGrad consistently enhances overall performance. For instance, CAGrad combined with VarGrad reduces the depth prediction error and decreases the task imbalance metric $\Delta m\%$ from 11.87% to 9.61%. Similarly, VarGrad improves MGDA-R and FairGrad by 1.54% and 2.58% in $\Delta m\%$, respectively. Importantly, these improvements are achieved without altering the core algorithmic structures, demonstrating VarGrad's modularity and ease of integration. These results confirm that VarGrad acts as a versatile plug-and-play module, effectively stabilizing optimization and promoting better task balance across diverse MTL methods with minimal modification.

5.3.5. Hyper-parameter Sensitivity Analysis.

We conducted a grid search over $\beta \in [0.75, 0.95]$ on the Cityscapes dataset and observed that $\beta = 0.85$ consistently led to the best overall performance. Detailed sensitivity analysis is provided in Supplementary Materials.

6. Conclusion

This work identifies gradient variance as a critical bottleneck in gradient-based MTL, leading to unstable updates and impaired performance. To address this, we propose VarGrad, a lightweight and plug-and-play framework that stabilizes training by iteratively task-specific gradients correction. Besides, VarGrad also supports a simple adaptive scheduling mechanism that selectively applies joint optimization based on task loss dynamics. Extensive experiments across diverse MTL benchmarks show that VarGrad consistently improves both training stability and final task performance, without incurring additional memory or architectural cost.

References

[1] Hao Ban and Kaiyi Ji. Fair resource allocation in multi-task learning. *arXiv preprint arXiv:2402.15638*, 2024. 1, 6, 7

[2] Shijie Chen, Yu Zhang, and Qiang Yang. Multi-task learning in natural language processing: An overview. *ACM Computing Surveys*, 56(12):1–32, 2024. 1

[3] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33:2039–2050, 2020. 7

[4] Zitian Chen, Yikang Shen, Mingyu Ding, Zhenfang Chen, Hengshuang Zhao, Erik G Learned-Miller, and Chuang Gan. Mod-squad: Designing mixtures of experts as modular multi-task learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11828–11837, 2023. 2

[5] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 6

[6] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020. 1

[7] Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd. *Advances in neural information processing systems*, 32, 2019. 2

[8] Zhiwen Fan, Rishov Sarkar, Ziyu Jiang, Tianlong Chen, Kai Zou, Yu Cheng, Cong Hao, Zhangyang Wang, et al. M^3vit: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design. *Advances in Neural Information Processing Systems*, 35:28441–28457, 2022. 2

[9] Heshan Fernando, Han Shen, Miao Liu, Subhajit Chaudhury, Keerthiram Murugesan, and Tianyi Chen. Mitigating gradient bias in multi-objective learning: A provably convergent approach. In *ICLR*, 2023. 2, 3, 7

[10] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019. 6

[11] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3205–3214, 2019. 2

[12] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021. 2

[13] Falk Heuer, Sven Mantowsky, Saqib Bukhari, and Georg Schneider. Multitask-centernet (mcn): Efficient and diverse multitask learning using an anchor free approach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 997–1005, 2021. 2

[14] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International conference on machine learning*, pages 427–435. PMLR, 2013. 2

[15] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013. 2

[16] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. 6, 7

[17] Jaehee Kim, Yukyung Lee, and Pilsung Kang. A gradient accumulation method for dense retriever under memory constraint, 2024. 2

[18] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6129–6138, 2017. 2

[19] Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *arXiv preprint arXiv:2111.10603*, 2021. 6, 7

[20] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021. 1, 2, 6, 7

[21] Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. *Advances in Neural Information Processing Systems*, 36:57226–57243, 2023. 4, 5, 6, 7

[22] Huiyun Liu and Yongxiang Xia. Optimal resource allocation in complex communication networks. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(7):706–710, 2015. 6

[23] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022. 2

[24] Liyang Liu, Yi Li, Zhanghui Kuang, J Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *ICLR*, 2021. 6, 7

[25] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019. 6, 7

[26] James Martens et al. Deep learning via hessian-free optimization. In *Icml*, pages 735–742, 2010. 5

[27] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*, 2022. 1, 2, 6, 7

[28] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1): 1–7, 2014. 6

[29] Nicolas Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. *Advances in neural information processing systems*, 25, 2012. 2

[30] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018. 1, 2, 6, 7

[31] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, pages 746–760. Springer, 2012. 6

[32] Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. *arXiv preprint arXiv:1605.06391*, 2016. 2

[33] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020. 1, 2, 6, 7

[34] Huizhuo Yuan, Yifeng Liu, Shuang Wu, Xun Zhou, and Quanquan Gu. Mars: Unleashing the power of variance reduction for training large models. *arXiv preprint arXiv:2411.10438*, 2024. 2

[35] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering*, 34 (12):5586–5609, 2021. 1

[36] Zhipeng Zhou, Liu Liu, Peilin Zhao, and Wei Gong. Injecting imbalance sensitivity for multi-task learning. *arXiv preprint arXiv:2503.08006*, 2025. 5