Appendix

A. Experimental Setup of Ablation Studies

In the main manuscript, we conducted an ablation study on TaskForce by incrementally integrating five key components: (1) Gram matrix observation ($\mathbf{g}\mathbf{g}^\top$), (2) Multi-Agent (MA), (3) Centralized Training (CT), (4) Decentralized Execution (DE), and (5) adding gradient-based reward $r_\mathcal{G}$. This section provides a detailed description of the experimental configurations for each ablation stage, supplementing the explanations in the main manuscript:

1. **Gram matrix ($\mathbf{g}\mathbf{g}^\top$)**: If we construct the agent observations $\mathcal{O}$ using the raw task gradient set $\mathbf{g}$ and the empirical loss set $\boldsymbol{\mathcal{L}}(\theta)$, the observations are as follows:

$$\begin{aligned}\mathcal{O} &= \left\{ \begin{array}{c} o_1 \\ \vdots \\ o_T \end{array} \right\} = \{\mathbf{g} | \boldsymbol{\mathcal{L}}(\theta)\} \\ &= \left\{ \begin{array}{c|c} g_1 & \mathcal{L}_1(\theta) \\ \vdots & \vdots \\ g_T & \mathcal{L}_T(\theta) \end{array} \right\} \in \mathbb{R}^{T(|\theta|+1)},\end{aligned} \tag{14}$$

   where $g_t = \nabla_\theta \mathcal{L}_t(\theta)$ for each task $t \in \{1, \cdots, T\}$. Since the number of shared parameters $|\theta|$ is exceedingly large compared to the number of tasks T ($T \ll |\theta|$), this formulation results in an observation space with prohibitive spatial and temporal complexity. Thus, as reported in Tab. 4, rendering the multi-task optimization process computationally infeasible.
   Therefore, we propose TaskForce that designs the compact observation based on the Gram matrix of the task gradient set as follows:

$$\begin{aligned}\mathcal{O} &= \left\{ \begin{array}{c} o_1 \\ \vdots \\ o_T \end{array} \right\} = \{\mathbf{g}\mathbf{g}^\top | \boldsymbol{\mathcal{L}}(\theta)\} \\ &= \left\{ \begin{array}{ccc|c} g_1 \cdot g_1 & \cdots & g_1 \cdot g_T & \mathcal{L}_1(\theta) \\ \vdots & \ddots & \vdots & \vdots \\ g_T \cdot g_1 & \cdots & g_T \cdot g_T & \mathcal{L}_T(\theta) \end{array} \right\} \in \mathbb{R}^{T(T+1)}.\end{aligned} \tag{15}$$

   Note that in setups where Decentralized Execution (DE) is not yet applied, the policy uses the entire observation matrix $\mathcal{O}$ as an input.
2. **Multi-agent training (MA)**: The single-agent baseline utilizes the DDPG [21]. Its policy network, $\mu(\mathcal{O}; \phi)$, takes the entire observation $\mathcal{O}$ and directly outputs the joint action vector $\mathcal{A}$. Conversely, the multi-agent configuration employs MADDPG [29] as introduced in the main manuscript. We introduce task-specific policies, where each policy $\mu_t(\mathcal{O}; \phi_t)$ takes the shared observation $\mathcal{O}$ but outputs only its corresponding action a_t as follows:

$$\mathcal{A} = [a_1, \cdots, a_T] = [\mu_t(\mathcal{O}; \phi_t)]_{t=1}^T. \tag{16}$$

   In this configuration, each critic Q_t^μ is still decentralized in its action evaluation; it takes the shared observation $\mathcal{O}$ and only the action a_t from its own agent to estimate the action value $\mathcal{Q}$ as follows:

$$\mathcal{Q} = Q_t^\mu(\mathcal{O}, a_t; \psi_t). \tag{17}$$

3. **Centralized Training (CT)**: A key contribution of MADDPG is its centralized critic, which we introduce in this step. Unlike the previous setup, each critic $Q_t^{\boldsymbol{\mu}}$ now evaluates its action value $\mathcal{Q}$ from the entire action $\mathcal{A}$ of all policies $\boldsymbol{\mu} = \{\mu_1, \cdots, \mu_T\}$, in addition to the shared observation $\mathcal{O}$. This allows for more effective credit assignment during training:

$$\mathcal{Q} = Q_t^\mu(\mathcal{O}, \mathcal{A}; \psi_t). \tag{18}$$

4. **Decentralized Execution (DE)**: To accelerate inference and to improve practicality, many MARL algorithms utilize decentralized execution. In this paradigm, each policy network μ_t uses only its local, task-specific observation o_t as input, rather than the full observation matrix $\mathcal{O}$. The critic, however, continues centralized training manner during training, utilizing the global observation $\mathcal{O}$ and the joint action $\mathcal{A}$ as follows:

$$\mathcal{A} = [a_1, \cdots, a_T] = [\mu_t(o_t; \phi_t)]_{t=1}^T, \tag{19}$$

$$\mathcal{Q} = Q_t^\mu(\mathcal{O}, \mathcal{A}; \psi_t). \tag{20}$$

5. **Gradient-Based Reward ($r_\mathcal{G}$)**: In the final configuration, we incorporate the proposed gradient-based reward term, $r_\mathcal{G}$. While previous setups relied solely on a loss-based reward, $\mathcal{R} = \lambda_\mathcal{L} r_\mathcal{L}$, the final reward function is a weighted sum of both components:

$$\mathcal{R} = \lambda_\mathcal{L} r_\mathcal{L} + \lambda_\mathcal{G} r_\mathcal{G}. \tag{21}$$

B. Hyperparameter and RL Agent Configuration of TaskForce

As aforementioned, in our work, we adapt the MADDPG [29] algorithm, where the number of actor and critic networks corresponds to the number of agents (the number of tasks in our method). Each actor and critic network is implemented as a three-layer MLP with a hidden dimension of 64. The agents are trained using the Adam optimizer [19] with a learning rate of 5×10^{-4} and a transition batch size of 64. The experience replay buffer stores up to 100,000 transitions, allowing the agents to perform off-policy learning effectively. We employ soft target updates [21] for stabilizing the critic network, using an exponential moving average

coefficient of $\tau = 0.01$. The discount factor for delayed rewards is set to $\gamma = 0.95$. To encourage exploration, we utilize Ornstein-Uhlenbeck (OU) noise [21] during training. The noise scale linearly decays from 0.3 to 0.05 over the first 10,000 training iterations. We use the log-transformed empirical loss to ensure scale-invariant conditions across tasks. Also, gradient normalization [34] is applied to prevent instability due to the scale dominance problem in the gradient space.

C. Compatibility of Gradient-based Reward with Other Convex Minimization

Beyond the gradient-based reward $r_{\mathcal{G}}$ utilized in this paper, the multi-objective optimization literature presents various approaches for solving convex minimization problems, often grounded in different hypotheses [26, 32]. To investigate whether our gradient-based reward framework is compatible with these alternative formulations, we adapt the minimization problem from a strong baseline, IMTL [26].

IMTL encourages a balanced update by equalizing the projection of the aggregated gradient onto each normalized task gradient. This is also framed as a minimization objective, negated into a reward as follows:

$$\begin{aligned} \underset{w_1,\cdots,w_T}{\text{minimize}} \quad & \sum_{t=2}^{T} \|\mathbf{G}(g_1/\|g_1\|_2 - g_t/\|g_t\|_2)^\top\|_2^2, \\ \text{subject to} \quad & \sum_{t=1}^{T} w_t = 1,\ w_t \geq 0, \end{aligned} \tag{22}$$

$$r_{\mathcal{G}}^{\text{IMTL}} = -\sum_{t=2}^{T} \|\mathbf{G}(g_1/\|g_1\| - g_t/\|g_t\|)^\top\|_2^2. \tag{23}$$

We conduct additional experiments to validate the performance of TaskForce using this IMTL-based reward $r_{\mathcal{G}}^{\text{IMTL}}$, with results presented in Tab. 8-10. Across all experiments in the appendix, we set $\lambda_{\mathcal{L}} = 1.0$ for the loss-based reward, $\lambda_{\mathcal{G}} = 1 \times 10^{-3}$ for the original gradient-based reward, and $\lambda_{\mathcal{G}}^{\text{IMTL}} = 1.0$ for the IMTL-based reward. As shown in Table 8, when the gradient-based reward is replaced with $r_{\mathcal{G}}^{\text{IMTL}}$, TaskForce maintains its high performance and continues to outperform strong baselines. This result demonstrates the potential extensibility of our methodology, indicating that the gradient-based reward can be effectively constructed from various gradient-level convex minimization problems proposed in the multi-objective optimization literature.

D. Sensitivity of Reward Weight Parameters of the TaskForce.

Beyond the standard hyperparameters of MADDPG (e.g., replay buffer size, discount factor), our proposed TaskForce introduces two tunable reward weights $\lambda_{\mathcal{L}}$ and $\lambda_{\mathcal{G}}$.

Table 7. Grid search of reward weight parameters $\lambda_{\mathcal{L}}$ and $\lambda_{\mathcal{G}}$ on the Cityscapes dataset. We report PSPNet [48] model performance averaged over 3 random seeds.

Reward Ratio ($\lambda_{\mathcal{L}} : \lambda_{\mathcal{G}}$)	$\Delta\mathbf{m}$ ($\lambda_{\mathcal{G}}$)	$\Delta\mathbf{m}$ ($\lambda_{\mathcal{G}}^{\text{IMTL}}$)
1:0	-0.33 %	-0.33 %
1:0.01	-0.65 %	-0.41 %
1:1	$+0.31$ %	-0.66 %
0:1	$+10.42$ %	$+6.10$ %

Due to the MADDPG utilizing the reward normalization for training stability [29], the relative proportion of $\lambda_{\mathcal{L}}$ and $\lambda_{\mathcal{G}}$ only significantly influences the learning outcome. As specified in the upper implementation details section, we used the same reward weights across all experiments. The relative proportions of these lambda values were determined experimentally to ensure a comparable average scale between the loss reward and gradient reward (i.e., $\lambda_{\mathcal{L}} = 1.0, \lambda_{\mathcal{G}} = 0.001, \lambda_{\mathcal{G}}^{IMTL} = 1.0$). The consistent performance improvements observed across all datasets experimentally demonstrate that our chosen lambda values can generalize well in practice.

Additionally, we conducted a grid search based on the ratio of the reward weight parameters, as shown in Tab. 7. We incorporate the original gradient-based reward $r_{\mathcal{G}}$ and other gradient-based reward $r_{\mathcal{G}}^{\text{IMTL}}$ introduced in Sec. C for this grid search. The results on the Cityscapes dataset show that our method outperforms the current state-of-the-art method, Aligned-MTL [35] ($\Delta\mathbf{m} = -0.01\%$), in most cases except for experiments where loss-based rewards were not leveraged, or where the gradient reward scale became excessively large (i.e., $\lambda_{\mathcal{L}} : \lambda_{\mathcal{G}} = 1 : 1$).

E. Statistical Analysis of Evaluation Results

Given that our methodology employs a Reinforcement Learning agent, discovering a suitable policy may fail. Consequently, we summarize the mean and variance of evaluation results across three trials for all experiments in the main manuscript in Tab. 8. Our approach, when relying solely on a loss-based reward, exhibits a non-negligible level of variance across trials. However, our method, incorporating the provably convergent gradient-based rewards $r_{\mathcal{G}}$ and $r_{\mathcal{G}}^{\text{IMTL}}$, demonstrates a comparatively low variance across the three trials. This suggests that gradient-based rewards can contribute to enhanced training stability.

Performing statistical analysis only on our method makes direct comparisons with other methods challenging. Therefore, we've summarized the results in Tab. 9 (including both reported and reproduced values with confidence intervals) for several existing methods and ours on the QM9 dataset below, as it represents the most challenging scenario with the highest number of tasks. Note that previous literature often

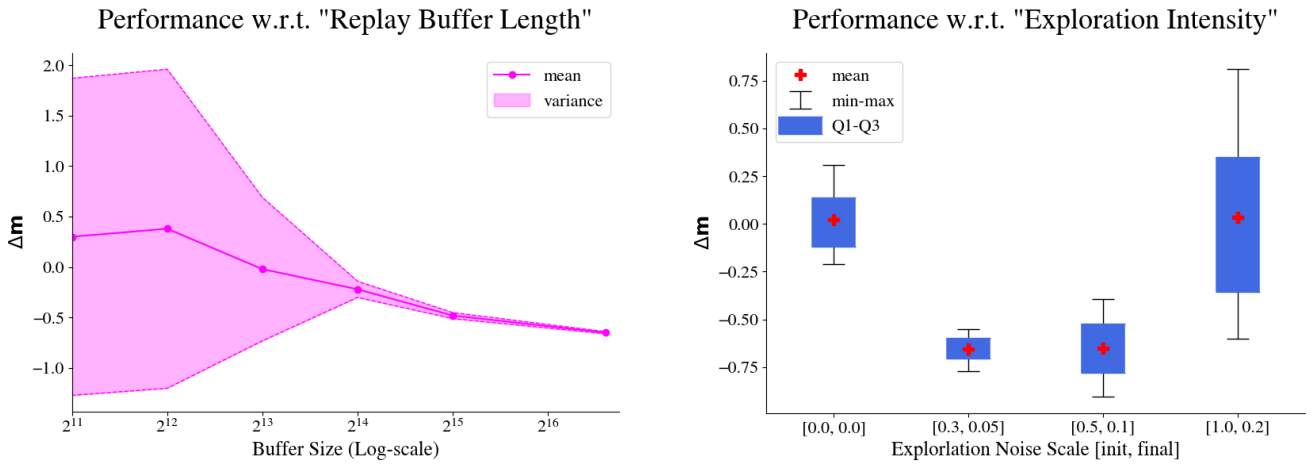

Figure 3. Performance trade-off of TaskForce on Cityscapes w.r.t. replay buffer size and exploration intensity. We report PSPNet [48] model performance averaged over 3 random seeds.

Table 8. Summary of statistical analysis of evaluation results of main manuscripts. We report the 3-run mean and variance of our method under the different configurations.

Method	NYU-v2	Cityscapes	QM9
STL	0.00 %	0.00 %	0.00 %
Ours ($r_{\mathcal{L}}$)	−5.46±1.79 %	−0.33±0.06 %	+64.2±3.8 %
Ours ($r_{\mathcal{L}}$ & $r_{\mathcal{G}}$)	−6.47±0.51 %	−0.65±0.01 %	+59.0±0.8 %
Ours ($r_{\mathcal{L}}$ & $r_{\mathcal{G}}^{\text{IMTL}}$)	−6.23±0.83 %	−0.66±0.02 %	+61.5±1.7 %

Table 9. Statistical summary of evaluation results on the QM9 dataset. We present the 3-run mean and variance of our proposed method and baseline models, including metrics reported in their original papers and our own reproduced metrics.

Method	$\Delta\mathbf{m}$ (reported)	$\Delta\mathbf{m}$ (reproduced)
STL (single-task learning)	0.00 %	0.00 %
LS	+177.6±3.4 %	+177.1±3.5 %
MGDA	+120.5±2.0 %	+113.1±4.3 %
IMTL	+77.2±9.3 %	+77.9±4.1 %
NashMTL	+62.0±1.4 %	+63.1±1.6 %
Aligned-MTL	N/A	+81.9±2.3 %
Ours ($r_{\mathcal{L}}$)	+64.2±3.8 %	+64.2±3.8 %
Ours ($r_{\mathcal{L}}$ & $r_{\mathcal{G}}$)	+59.0±1.0 %	+59.0±1.0 %
Ours ($r_{\mathcal{L}}$ & $r_{\mathcal{G}}^{\text{IMTL}}$)	+61.5±1.7 %	+61.5±1.7 %

tends to report higher values than what can be reproduced. For this reason, we used reported values for the main paper, and the statistical information for these reported values was sourced from the NashMTL [32] and Aligned-MTL [35].

E.1. Robustness of the TaskForce w.r.t. Hyperparameters

TaskForce, which leverages multi-agent reinforcement learning, can exhibit varying performance depending on the agents' hyperparameters. Therefore, we conduct comprehensive experiments on the Cityscapes dataset, systematically varying the values of key components that can influence the agents' learning to observe the resulting performance changes. We focus on two primary components: (1) replay buffer length and (2) exploration noise scale, evaluating the impact of different values on the overall performance metric $\Delta\mathbf{m}$ (where lower is better). Note that, for agent exploration, we progressively decrease the scale of the Ornstein-Uhlenbeck (OU) noise [21] applied to the policy during the exploration period, transitioning from an initial scale to a final scale.

Fig. 3 illustrates the statistical analysis of evaluation results under different configurations across these key components. Firstly, as the replay buffer length increases, the sampling efficiency of stored transitions improves, leading to enhanced stability. These also result in both improved performance and reduced variance.

Regarding the exploration noise scale, we observe that introducing appropriate exploration leads to performance gains compared to configurations without exploration (i.e., noise scale: [0.0, 0.0]). However, excessive exploration intensity results in decreased performance and increased variance, indicating instability.

Theoretically, prior work such as RLW [1] has demonstrated that introducing extra randomness through random loss weighting significantly increases the probability of escaping sharp local minima compared to deterministic linear scalarization (refer to *Theorem 2* of [1]). In a similar vein, the exploration noise of our TaskForce framework serves as a dynamic, learned source of stochasticity at the task-weight level. Unlike simple random weighting, our MARL-driven stochasticity provides a principled exploratory pressure that not only prevents premature convergence to suboptimal solutions but also enables a more robust traversal of complex,

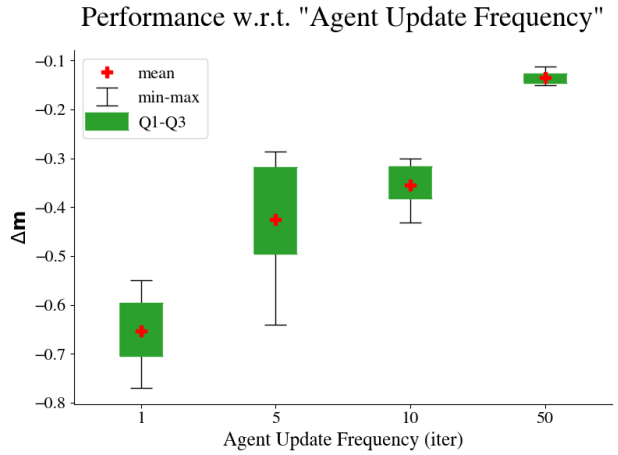

Figure 4. Performance trade-off of TaskForce on Cityscapes w.r.t. agent update frequency. We report PSPNet [48] model performance averaged over 3 random seeds.

non-convex optimization landscapes.

E.2. Practical Speedup of TaskForce

Our methodology employs highly compact agents (e.g., a 2-layer MLP with a hidden dimension of 64) to achieve a fast and practical Multi-Task Learning training process. However, the necessity to update the parameters of both agents and critics at each iteration poses an issue, as it still leads to a linear increase in training cost with respect to the number of tasks. We can mitigate this scalability issue by reducing the agent update frequency within TaskForce, which practically accelerates the Multi-task optimization process. Consequently, we investigate the performance trade-off resulting from variations in this update frequency.

We observe that reducing the agent update frequency leads to an increase in the optimization cost efficiency of TaskForce (a speedup of ×8.73 for an update frequency of 10). As shown in Fig. 4, while increasing the update frequency tends to decrease performance, our method still outperforms existing approaches (e.g., Δm=0.01 for Aligned-MTL [35]), suggesting the potential for efficient optimization even in scenarios with a larger number of tasks.

E.3. Alternative Formulation of the TaskForce

As mentioned previously in our main manuscript, there are two strategies for interpreting the MTL training process as a cooperative Markov game:

Strategy 1: Viewing the entire MTL training process as a single episode (the methodology employed in the main manuscript).

Strategy 2: Alternatively, considering each gradient descent step as an individual episode (where transitions are derived using the same data point after a model update).

Algorithm 2 Alternative Training Process of the TaskForce.

Input: data point number N, task number T, data points $\mathbf{X}, \{\mathbf{Y}_t\}_{1\leq t\leq T}$, MTL model $\mathcal{F}(\cdot;\theta)$, agents $\{\mu_t(\cdot;\phi_t), Q_t^{\mu}(\cdot,\cdot;\psi_t)\}_{1\leq t\leq T}$, replay buffer $\mathbf{D}$, batch size of agents b_{agent}.

Output: trained MTL model $\mathcal{F}(\cdot;\theta^*)$, trained agents $\{\mu_t(\cdot;\phi_t^*), Q_t^{\mu}(\cdot,\cdot;\psi_t^*)\}_{1\leq t\leq T}$.

initialize $\mathcal{O}_{\text{prev}}, \mathcal{A}_{\text{prev}}, \mathcal{R}_{\text{prev}}$ as null matrix or 0.
for $i = 1$ **to** N **do**
  sample data point $\{\mathbf{x}^i, \mathbf{y}_1^i, \cdots, \mathbf{y}_T^i\}$.
  `# compute the current observation` $\mathcal{O}$, `and action` $\mathcal{A}$.
  compute empirical loss set $\boldsymbol{\mathcal{L}}(\theta)$ and task gradient set $\mathbf{g}$ from data point.
  generate observation $\mathcal{O}$ from $\boldsymbol{\mathcal{L}}(\theta), \mathbf{g}$ by Eq. 5.
  compute action $\mathcal{A} = \{\mu_1(o_1;\phi_1), \cdots, \mu_T(o_T;\phi_T)\}$ from $\mathcal{O}$.
  compute aggregated gradient $\mathbf{G}$ from $\mathbf{g}, \mathcal{A}$ by Eq. 6.
  update MTL model $\mathcal{F}(\cdot|\theta)$ with $\mathbf{G}$ by $\theta \leftarrow \theta - \eta\mathbf{G}$.
  `# compute the next observation` $\mathcal{O}'$, `and reward` $\mathcal{R}$.
  compute next empirical loss set $\boldsymbol{\mathcal{L}}_{\text{next}}(\theta)$ and next task gradient set $\mathbf{g}_{\text{next}}$ from same data point.
  generate next observation $\mathcal{O}_{\text{next}}$ from $\boldsymbol{\mathcal{L}}_{\text{next}}(\theta), \mathbf{g}_{\text{next}}$ by Eq. 5.
  compute reward $\mathcal{R}$ from $\boldsymbol{\mathcal{L}}(\theta), \boldsymbol{\mathcal{L}}_{\text{next}}$ by Eq. 7-10.
  push transition $(\mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{O}_{\text{next}})$ to replay buffer $\mathbf{D}$.
  if $i > b_{\text{agent}}$ **then**
    sample b_{agent} transitions $\mathcal{T}$ from replay buffer $\mathbf{D}$.
    update actor $\mu_i(\cdot;\phi_i)$ and critic $Q_i^{\mu}(\cdot,\cdot;\psi_i)$ with transitions $\mathcal{T}$ by Eq. 11-13.
  end if
end for

While *Strategy 1* treats data points as part of a non-stationary environment, *Strategy 2* allows for the measurement of the loss reduction rate on the same data point, potentially leading to more stable reward measurement. However, a significant drawback of *Strategy 2* is its requirement for nearly twice the training cost, as it necessitates measuring the loss and gradients of the network both before and after each update on the same data point. We provide the algorithmic description for this alternative formulation in Alg. 2 and its experimental results on the Cityscapes dataset in Tab. 10. The approach utilizing Alg. 2 enables the agent to learn more rapidly due to the stable reward signal obtained from the same data point, resulting in a slight performance improvement of $(r_{\mathcal{L}})$ and $(r_{\mathcal{L}}$ & $r_{\mathcal{G}}^{\text{IMTL}})$ settings. Nevertheless, given the marginal performance gain in Tab. 10 compared to the doubled computational cost, we adopt the first strategy in the main body of our work.

E.4. Visualize Weight Dynamics

In Fig. 5, we visualize the task weight dynamics observed during our experiments across the two datasets utilized: NYU-v2, Cityscapes, as similar to Fig. 2 in the main

Table 10. Comparison between original TaskForce and Alternative formulation. We evaluate both methods on the Cityscapes 3-task setup. We report PSPNet [48] model performance averaged over 3 random seeds. We report the relative runtime of MTL training from the original TaskForce using the same computing resources.

Method	Semseg. mIoU (%) ↑	Instseg. L1 (px.) ↓	Disparity MSE ↓	$\Delta\mathbf{m}\downarrow$	runtime
STL	66.73	10.55	0.33	0.00%	×0.83
TaskForce ($r_{\mathcal{L}}$)	66.10	10.49	0.33	−0.33%	×1.00
TaskForce ($r_{\mathcal{L}}$ & $r_{\mathcal{G}}$)	66.63	10.55	**0.32**	−0.65%	×1.00
TaskForce ($r_{\mathcal{L}}$ & $r_{\mathcal{G}}^{\text{IMTL}}$)	66.31	**10.48**	**0.32**	−0.66%	×1.00
TaskForce-ALT ($r_{\mathcal{L}}$)	**66.81**	10.51	0.33	−0.41%	×1.86
TaskForce-ALT ($r_{\mathcal{L}}$ & $r_{\mathcal{G}}$)	66.70	10.54	**0.32**	−0.58%	×1.86
TaskForce-ALT ($r_{\mathcal{L}}$ & $r_{\mathcal{G}}^{\text{IMTL}}$)	66.50	10.49	**0.32**	**−0.74%**	×1.86

manuscript. Our method demonstrates the ability to adaptively determine appropriate weights for optimization across these datasets, including NYU-v2 and Cityscapes, as well as the relatively complex QM9 dataset, which presents a scale dominance problem and comprises a large number of tasks.

F. Discussion: Advantages of Cooperative Multi-Agent RL in Multi-Task Optimization

In our main manuscript, we conduct ablation studies on key components of Cooperative MARL to evaluate each component's contribution. These results underscore the significant contribution of MARL's modular and cooperative architecture, especially when fully integrated, to improving both the stability of convergence and the overall performance in multi-task learning settings. Based on these results, this section aims to discuss the characteristics of cooperative MARL that may contribute to the observed performance improvements.

Problem Decomposition and Specialization. Unlike Single-agent RL, which attempts to learn a unified policy to handle all task dynamics, MARL assigns each task to a dedicated agent. This modular design enables agents to specialize in task-specific behaviors, potentially allowing the learning process to capture diverse optimization dynamics without interference. Such decomposition might be particularly beneficial in multi-task settings where task objectives may be partially conflicting.

Improved Exploration through Distributed Policies. In MARL, agents can explore the optimization space in parallel, which not only enhances exploration coverage but also reduces the risk of premature convergence to suboptimal joint strategies. MARL leverages this by allowing task-specific agents to independently probe different gradient accumulation strategies, ultimately potentially leading to more effective joint optimization.

Robustness and Redundancy. Multi-agent systems inherently offer robustness, as the failure or poor performance of one agent could be compensated by others. This redundancy leads to more stable learning dynamics, especially in noisy or partially observable environments. In contrast, a

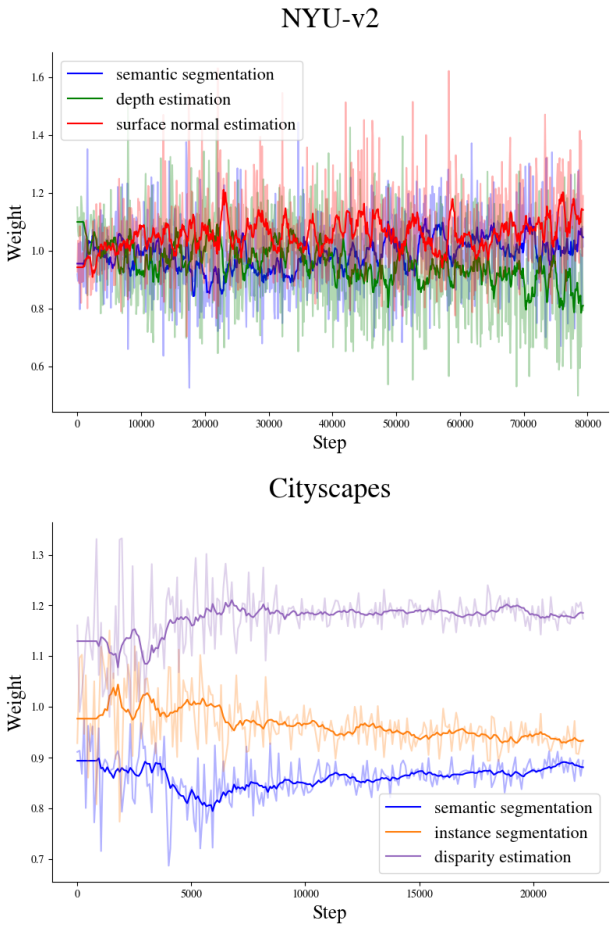

Figure 5. Task weight dynamics of NYU-v2, and Cityscapes datasets. For improved visualization, we smooth the weight dynamics by using a moving average with a window size of 10.

single-agent approach lacks the granularity and flexibility to adapt to individual task needs. It must implicitly learn to balance conflicting gradients and loss scales, often resulting in suboptimal convergence or instability in complex multi-task scenarios.

Taken together, our findings highlight the suitability of cooperative MARL, particularly MADDPG [29], as a powerful optimization backbone for MTL frameworks. By decomposing task responsibilities and leveraging centralized training signals, MARL could offer both stability and efficiency, especially when facing task-level gradient conflicts and scale imbalances.

G. Limitation & Future Works

Our method employs MADDPG, an early approach in Multi-Agent Reinforcement Learning (MARL). Consequently, the robustness of our optimization framework with more advanced MARL techniques remains unverified, as the optimization outcome can vary depending on the agents' performance. We believe that future research could further enhance multi-task optimization performance by applying more sophisticated MARL methodologies, a direction we leave for future investigators. Furthermore, while designing TaskForce, we consider a simultaneous training scenario and thus engineer very compact agents in terms of size and observation space. Nevertheless, a scalability issue persists, as the number of required agents still increases linearly with the number of tasks.