# Multi-Task Learning as a Bargaining Game

**Aviv Navon** [*1] **Aviv Shamsian** [*1] **Idan Achituve** [1] **Haggai Maron** [2] **Kenji Kawaguchi** [3]
**Gal Chechik** [1 2] **Ethan Fetaya** [1]

## Abstract

In Multi-task learning (MTL), a joint model is trained to simultaneously make predictions for several tasks. Joint training reduces computation costs and improves data efficiency; however, since the gradients of these different tasks may conflict, training a joint model for MTL often yields lower performance than its corresponding single-task counterparts. A common method for alleviating this issue is to combine per-task gradients into a joint update direction using a particular heuristic. In this paper, we propose viewing the gradients combination step as a bargaining game, where tasks negotiate to reach an agreement on a joint direction of parameter update. Under certain assumptions, the bargaining problem has a unique solution, known as the *Nash Bargaining Solution*, which we propose to use as a principled approach to multi-task learning. We describe a new MTL optimization procedure, Nash-MTL, and derive theoretical guarantees for its convergence. Empirically, we show that Nash-MTL achieves state-of-the-art results on multiple MTL benchmarks in various domains.

## 1. Introduction

In many real-world applications, one needs to solve several tasks simultaneously using limited computational or data resources. For example, perception for autonomous vehicles requires lane detection, object detection, and free-space estimation, which must all run in parallel and in real-time. This is normally solved via multi-task learning (MTL), where one model is jointly trained on several learning tasks (Caruana, 1997; Ruder, 2017; Crawshaw, 2020). Multi-task learning was also shown to improve generalization in theory (Baxter, 2000) and in practice (e.g., auxiliary learning, Liu et al., 2019a; Achituve et al., 2021; Navon et al., 2021a).

Unfortunately, MTL often causes performance degradation compared to single-task models (Standley et al., 2020). A main reason for such degradation is gradients conflict (Yu et al., 2020a; Wang et al., 2020; Liu et al., 2021a). These per-task gradients may have conflicting directions or a large difference in magnitudes, with the largest gradient dominating the update direction. The degraded performance of MTL due to poor training, compared with its potential to improve performance due to better generalization, has a major impact on many real-world systems. Improving MTL optimization algorithms is therefore an important task with significant implications to many systems.

Currently, most MTL optimization algorithms (Sener & Koltun, 2018; Yu et al., 2020a; Liu et al., 2021a) follow a general scheme. First, compute the gradients for all tasks $g_1, ..., g_K$. Next, combine those gradients into a joint direction, $\Delta = \mathcal{A}(g_1, ..., g_K)$ using an aggregation algorithm $\mathcal{A}$. Finally, update model parameters using a single-task optimization algorithm, replacing the gradients with $\Delta$. Multiple heuristics were proposed for the aggregation algorithm $\mathcal{A}$. However, to the best of our knowledge, a principled, axiomatic, approach to gradient aggregation is still missing.

Here we address the gradient combination step by viewing it as a cooperative bargaining game (Thomson, 1994). Each task represents a player, whose utility is derived from its gradient, and players negotiate to reach an agreed direction. This formulation allows us to use results from game theory literature that analyze this problem from an axiomatic perspective. In his seminal paper, Nash (1953) presented an axiomatic approach to the bargaining problem and showed that under certain axioms, the bargaining problem has a unique solution known as the *Nash Bargaining Solution*. This solution is known to be proportionally fair, where any alternative will have a negative average relative change. This proportionally fair update allows us to find a solution that works for all tasks without being dominated by a single large gradient.

Building on Nash's results, we propose a novel MTL optimization algorithm, named *Nash-MTL*, where the gradients are combined at each step using the Nash bargaining so-

[*]Equal contribution [1]Bar-Ilan University, Ramat Gan, Israel [2]Nvidia, Tel-Aviv, Israel [3]National University of Singapore. Correspondence to: Aviv Navon <aviv.navon@biu.ac.il>, Aviv Shamsian <aviv.shamsian@live.biu.ac.il>.

*Proceedings of the $39^{th}$ International Conference on Machine Learning*, Baltimore, Maryland, USA, PMLR 162, 2022. 

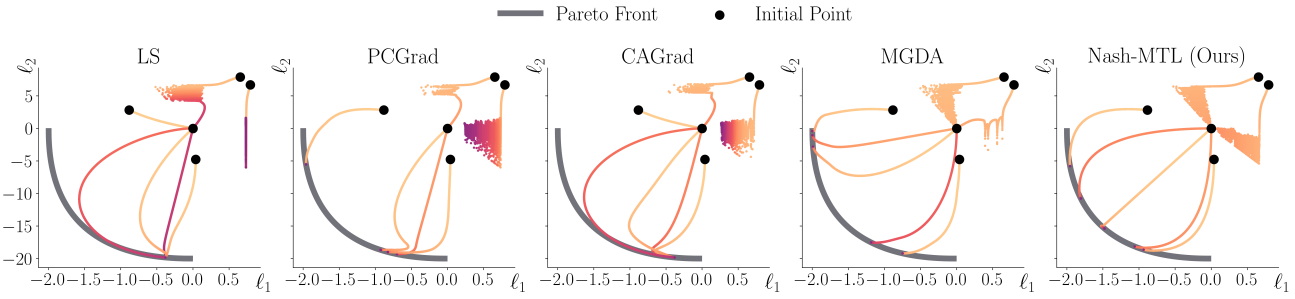



*Figure 1. Illustrative example*: Optimization trajectories in loss space. Shown are 5 different initializations (black dots •), and their trajectories are colored from orange to purple. Losses have a large difference in scale. See Appendix B for details. For linear scalarization (LS), PCGrad, and CAGrad, the optimization process is controlled by the gradient of $\ell_2$, since it has a larger magnitude, resulting in imbalanced solutions between tasks (mostly ending at the bottom right). These three methods also fail to converge to an optimal solution for the rightmost initialization points. In contrast, MGDA is inclined towards the task with the smallest gradient magnitude ($\ell_1$). Our method, *Nash-MTL*, is invariant to changes in loss scale and produces solutions that are well balanced across the Pareto front.

lution. We first characterize the Nash bargaining solution for MTL and derive an efficient algorithm to approximate its value. Then, we analyze our approach theoretically and establish convergence guarantees in the convex and non-convex cases. Finally, we show empirically that our Nash-MTL approach achieves state-of-the-art results on four MTL benchmarks on a variety of challenges ranging from computer vision and quantum chemistry to reinforcement learning. To support future research and the reproducibility of the results, we make our source code publicly available at: https://github.com/AvivNavon/nash-mtl.

## 2. Background

### 2.1. Pareto Optimality

Optimization for MTL is a specific case of multiple-objective optimization (MOO). Given objective functions $\ell_1, ..., \ell_K$, the performance of solution a $x$ is measured by the vector of objective values $(\ell_1(x), ..., \ell_K(x))$. One main property of MOO is that since there is no natural linear ordering on vectors it is not always possible to compare solutions so there is no clear optimal value.

We say that a solution $x$ dominates $x'$ if it is better on one or more objectives and not worse on any other objectives. A solution that is not dominated by any other is called *Pareto optimal*, and the set of all such solutions is called the *Pareto front*. It is important to note that there is no clear way to select between different Pareto optimal solutions without additional assumptions or prior about the user preferences (Navon et al., 2021b). For non-convex problems, a point is defined as local Pareto optimal if it is Pareto optimal in some open set containing it. Further, a point is called *Pareto stationary* if there exists a convex combination of the gradients at this point that equals zero. Pareto stationarity is a necessary condition for Pareto optimality.

### 2.2. Nash Bargaining Solution

We provide a brief background on cooperative bargaining games and the Nash bargaining solution, see Thomson (1994) for more details. In a bargaining problem, we have $K$ players, each with their own utility function $u_i : A \cup \{D\} \to \mathbb{R}$, which they wish to maximize. $A$ is the set of possible agreements and $D$ is the disagreement point which the players default to if they fail to reach an agreement. We define the set of possible payoffs as $U = \{(u_1(x), ..., u_K(x)) : x \in A\} \subset \mathbb{R}^K$ and $d = (u_1(D), ..., u_K(D))$. We assume $U$ is convex, compact and that there exists a point in $U$ that strictly dominates $d$, namely there exists a $u \in U$ such that $\forall i : u_i > d_i$.

Nash (1953) showed that for such payoff set $U$, the two-player bargaining problem has a unique solutionthat satisfies the following properties or axioms: Pareto optimality, symmetry, independence of irrelevant alternatives, and invariant to affine transformations. This was later extended to multiple players (Szép & Forgó, 1985).

**Axiom 2.1. Pareto optimality:** The agreed solution must not be dominated by another option, i.e. there cannot be any other agreement that is better for at least one player and not worse for any of the players.

As it is a cooperative game, it makes little sense that the players will curtail another player without any personal gains, so it is natural to assume the agreed solution will not be dominated by another.

**Axiom 2.2. Symmetry:** The solution should be invariant to permuting the order of the players.

**Axiom 2.3. Independence of irrelevant alternatives (IIA):** If we enlarge the of possible payoffs to $\tilde{U} \supseteq U$, and the solution is in the original set $U$, $u^* \in U$, then the agreed point when the set of possible payoffs is $U$ will stay $u^*$.

**Axiom 2.4. Invariance to affine transformation:** If we

transform each utility function $u_i(x)$ to $\tilde{u}_i(x) = c_i \cdot u_i(x) + b_i$ with $c_i > 0$ then if the original agreement had utilities $(y_1, ..., y_k)$ the agreement after the transformation has utilities $(c_1 y_1 + b_1, ..., c_k y_k + b_k)$

We argue that in the MTL setting, it is natural to require axioms 2.1-2.3. Axiom 2.4, in our mind, is the only non-natural assumption used by the Nash bargaining solution in the context of MTL. We argue that indeed it is a desired property that is helpful for MTL. Axiom 2.4 means that the solution does not take into account the gradients' norms but rather treats all of them the same, as if they were normalized. Without enforcing this assumption, the solution can easily be dominated by a single direction (see Figure 1). We further validate the importance of this assumption by investigating a scale-invariant baseline in Section 6.

The unique point satisfying all these axioms is called the Nash bargaining solution and is given as

$$u^* = \arg\max_{u \in U} \sum_i \log(u_i - d_i) \quad (1)$$
$$s.t.\ \forall i:\ u_i > d_i$$

## 3. Method

We now describe our Nash-MTL method in detail. We first formalize the gradient combination step as a bargaining game and analyze the Nash bargaining solution for this game. We then describe our algorithm to approximate the solution efficiently. We note that the computational cost of that approximation is critical because this approximation is executed for each gradient update. To simplify the notation, we do not distinguish between shared and task-specific parameters. We note, however, that task-specific parameters have no contribution to the Nash bargaining solution calculation.

### 3.1. Nash Bargaining Multi-Task Learning

Given an MTL optimization problem and parameters $\theta$, we search for an update vector $\Delta\theta$ in the ball of radius $\epsilon$ centered around zero, $B_\epsilon$. We frame this as a bargaining problem with the agreement set $B_\epsilon$ and the disagreement point at 0, i.e., staying at the current parameters $\theta$. We define the utility function for each player as $u_i(\Delta\theta) = g_i^\top \Delta\theta$ where $g_i$ is the gradient of the loss of task $i$ at $\theta$. We note that since the agreement set is compact and convex and the utilities are linear then the set of possible payoffs is also compact and convex.

Our main assumption, besides the ones used by Nash, is that if $\theta$ is not Pareto stationary then the gradients are linearly independent (see further discussion on this assumption in Section 5). Under this assumption, we also have that the disagreement point, $\Delta\theta = 0$ is dominated by another in $B_\epsilon$.

We now show that if $\theta$ is not on the Pareto front, the unique Nash bargaining solution has the following form:

**Claim 3.1.** *Let $G$ be the $d \times K$ matrix whose columns are the gradients $g_i$. The solution to $\arg\max_{\Delta\theta \in B_\epsilon} \sum_i \log(\Delta\theta^\top g_i)$ is (up to scaling) $\sum_i \alpha_i g_i$ where $\alpha \in \mathbb{R}_+^K$ is the solution to $G^\top G\alpha = 1/\alpha$ where $1/\alpha$ is the element-wise reciprocal.*

*Proof.* The derivative of this objective is $\sum_{i=1}^K \frac{1}{\Delta\theta^\top g_i} g_i$. For all vectors $\Delta\theta$ such that $\forall i : \Delta\theta^T g_i > 0$ the utilities are monotonically increasing with the norm of $\Delta\theta$. Thus, from the Pareto optimality assumption by Nash, the optimal solution has to be on the boundary of $B_\epsilon$. From this we see that the gradient at the optimal point $\sum_{i=1}^K \frac{1}{\Delta\theta^\top g_i} g_i$ must be in the radial direction, i.e., $\sum_{i=1}^K \frac{1}{\Delta\theta^\top g_i} g_i \parallel \Delta\theta$ or $\sum_{i=1}^K \frac{1}{\Delta\theta^\top g_i} g_i = \lambda\Delta\theta$. Since the gradients are independent we must have $\Delta\theta = \sum_i \alpha_i g_i$ and $\forall i : \frac{1}{\Delta\theta^\top g_i} = \lambda\alpha_i$ or $\forall i : \Delta\theta^\top g_i = \frac{1}{\lambda\alpha_i}$. As the inner product must be positive for a descent direction we can conclude $\lambda > 0$; we set $\lambda = 1$ to ascertain the direction of $\Delta\theta$ (the norm might be larger then $\epsilon$). Now finding the bargaining solution is reduced to finding $\alpha \in \mathbb{R}^K$ with $\alpha_i > 0$ such that $\forall i : \Delta\theta^\top g_i = \sum_j \alpha_j g_j^\top g_i = \frac{1}{\alpha_i}$. This is equivalent to requiring that $G^\top G\alpha = 1/\alpha$ where $1/\alpha$ is the element-wise reciprocal. □

We now provide some intuition for this solution. First, if all $g_i$ are orthogonal we get $\alpha_i = 1/||g_i||$ and $\Delta\theta = \sum \frac{g_i}{||g_i||}$ which is the obvious scale invariant solution. When they are not orthogonal, we get

$$\alpha_i ||g_i||^2 + \sum_{j \neq i} \alpha_j g_j^\top g_i = 1/\alpha_i \quad (2)$$

We can consider $\sum_{j \neq i} \alpha_j g_j^\top g_i = \left(\sum_{j \neq i} \alpha_j g_j\right)^\top g_i$ as the interaction between task $i$ and the other tasks; If it is positive there is a positive interaction and the other gradients aid the $i$'th task, and if it is negative they hamper it. When there is a negative interaction, the LHS of Eq. 2 decreases and as a result, $\alpha_i$ increases to compensate for it. Conversely, where there is a positive interaction $\alpha_i$ will decrease.

### 3.2. Solving $\mathbf{G}^\top \mathbf{G}\alpha = 1/\alpha$

Here we describe how to efficiently approximate the optimal solution for $G^\top G\alpha = 1/\alpha$ through a sequence of convex optimization problems. We define a $\beta_i(\alpha) = g_i^\top G\alpha$, and wish to find $\alpha$ such that $\alpha_i = 1/\beta_i$ for all $i$, or equivalently $\log(\alpha_i) + \log(\beta_i(\alpha)) = 0$. Denote $\varphi_i(\alpha) = \log(\alpha_i) + \log(\beta_i)$ and $\varphi(\alpha) = \sum_i \varphi_i(\alpha)$. With that, our goal is to find a non-negative $\alpha$ such that $\varphi_i(\alpha) = 0$ for all $i$. We can

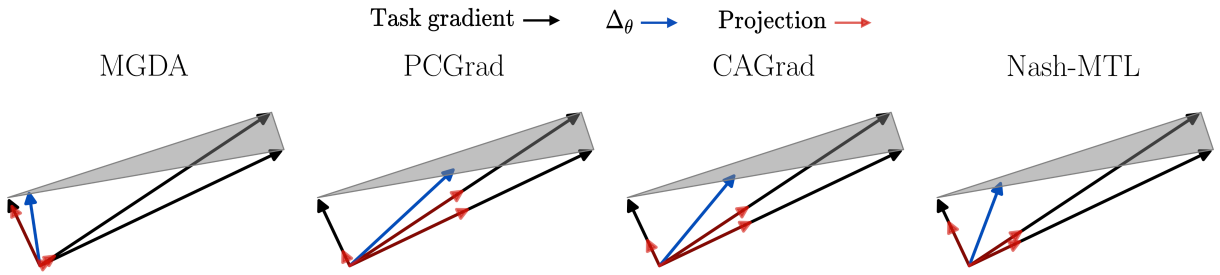



*Figure 2. Visualization of the update direction*: We show the update direction (blue) obtained by various methods on three gradients in $\mathbb{R}^3$. We rescaled the returned vectors for better visibility, showing only the direction. We further show the size of the projection (red) of the update to each gradient direction (black). Nash-MTL produce an update direction with the most balanced projections.

**Algorithm 1** Nash-MTL

**Input:** $\theta^{(0)}$ – initial parameter vector, $\{\ell_i\}_{i=1}^K$ – differentiable loss functions, $\eta$ – learning rate
**for** $t = 1, ..., T$ **do**
  Compute task gradients $g_i^{(t)} = \nabla_{\theta^{(t-1)}} \ell_i$
  Set $G^{(t)}$ the matrix with columns $g_i^{(t)}$
  Solve for $\alpha$: $(G^{(t)})^\top G^{(t)} \alpha = 1/\alpha$ to obtain $\alpha^{(t)}$
  Update the parameters $\theta^{(t)} = \theta^{(t)} - \eta G^{(t)} \alpha^{(t)}$
**end for**
**Return:** $\theta^{(T)}$

write this as the following optimization problem

$$\begin{aligned} \min_{\alpha} & \sum_i \varphi_i(\alpha) \\ \text{s.t.} \forall i, & \quad -\varphi_i(\alpha) \leq 0 \\ & \alpha_i > 0 \quad . \end{aligned} \tag{3}$$

The constraints in this problem are convex and linear and the objective is concave. We first try to solve the following convex surrogate objective

$$\begin{aligned} \min_{\alpha} & \sum_i \beta_i(\alpha) \\ \text{s.t.} \forall i, & \quad -\varphi_i(\alpha) \leq 0 \\ & \alpha_i > 0 \quad . \end{aligned} \tag{4}$$

Here, we minimize $\sum_i \beta_i$ under the constraint $\beta_i = g_i^\top G \alpha \geq 1/\alpha_i$. While this objective is not equivalent to the original problem, we found it very useful. In many cases, it produces exact solutions with $\varphi(\alpha) = 0$ as required.

To further improve our approximation, we considered the following problem,

$$\begin{aligned} \min_{\alpha} & \sum_i \beta_i(\alpha) + \varphi(\alpha) \\ \text{s.t.} \forall i, & \quad -\varphi_i(\alpha) \leq 0 \\ & \alpha_i > 0 \quad . \end{aligned} \tag{5}$$

Adding $\varphi(\alpha)$ to the objective may further reduce it, moving it closer to zero; however, it renders the problem to be non-convex. Despite that, our solution can be improved iteratively by replacing the concave term $\varphi(\alpha)$ with its first-order approximation $\tilde{\varphi}_\tau(\alpha) = \varphi(\alpha^{(\tau)}) + \nabla\varphi(\alpha^{(\tau)})^\top(\alpha - \alpha^{(\tau)})$. Where, $\alpha^{(\tau)}$ is the solution at iteration $\tau$. Note that we replace $\varphi$ with $\tilde{\varphi}$ only in the objective and keep $\varphi(\alpha)$ as is in the constraint: i.e., $\min_\alpha \sum_i \beta_i(\alpha) + \tilde{\varphi}_\tau(\alpha)$ s.t. $-\varphi_i(\alpha) \leq 0$ and $\alpha_i > 0$ for all $i$. This sequential optimization approach is a variation of the concave-convex procedure (CCP) (Yuille & Rangarajan, 2003; Lipp & Boyd, 2016). Therefore the sequence $\{\alpha^{(\tau)}\}_\tau$ converges to a critical point of the original non-convex problem in Eq. 5 based on previous theory of CCP by Sriperumbudur & Lanckriet (2009). Moreover, since we do not modify the constraint, $\alpha^{(\tau)}$ always satisfies the constraint of the original problem for any $\tau$. Finally, the following proposition shows that original objective monotonically decreases with $\tau$:

**Proposition 3.2.** *Denote the objective for the optimization problem in Eq. 5 by* $\phi(\alpha) = \sum_i \beta_i(\alpha) + \varphi(\alpha)$. *Then,* $\phi\left(\alpha^{(\tau+1)}\right) \leq \phi\left(\alpha^{(\tau)}\right)$ *for all* $\tau \geq 1$.

We provide proof and further discussion in Appendix A. In practice, we limit the sequence of CCP to 20 in all experiments, with the exception of Section 6.3 for which we use a single step. We found the improved solution to have a limited effect on the MTL performance (see Appendix D.2).

### 3.3. Practical Speedup

One shortcoming of many leading MTL methods is that all task gradients are required for obtaining the joint update direction. When the number of tasks $K$ becomes large, this may be too computationally expensive as it requires one to perform $K$ backward passes through the shared backbone to compute the $K$ gradients. Prior work suggested using a subset of tasks (Liu et al., 2021a) or replacing the task gradients with the feature level gradient (Sener & Koltun, 2018; Liu et al., 2021b; Javaloy & Valera, 2021) as potential practical speedups. We emphasize that this issue is not

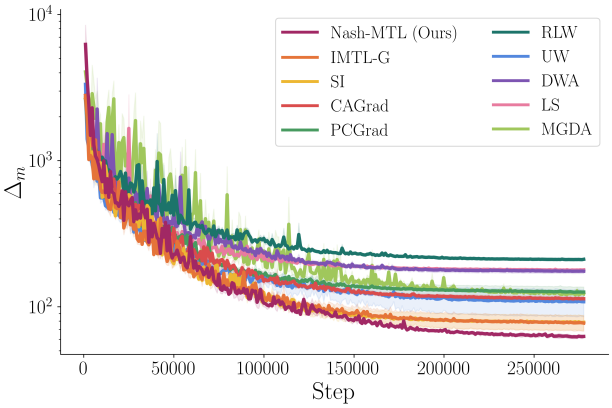



*Figure 3. QM9.* Test $\Delta_m$ throughout the training process averaged over 3 random seeds.

*Table 1. QM9.* Test performance averaged over 3 random seeds.

| | **MR** ↓ | $\mathbf{\Delta_m\%}$ ↓ |
|---|---|---|
| LS | 6.8 | 177.6 ± 3.4 |
| SI | 4.0 | 77.8 ± 9.2 |
| RLW | 8.2 | 203.8 ± 3.4 |
| DWA | 6.4 | 175.3 ± 6.3 |
| UW | 5.3 | 108.0 ± 22.5 |
| MGDA | 5.9 | 120.5 ± 2.0 |
| PCGrad | 5.0 | 125.7 ± 10.3 |
| CAGrad | 5.7 | 112.8 ± 4.0 |
| IMTL-G | 4.7 | 77.2 ± 9.3 |
| Nash-MTL | **2.5** | **62.0 ± 1.4** |

unique to our method, but rather is shared to all methods that compute all gradients for all tasks.

In practice, we found that using feature-level gradients as a surrogate to the gradient of the shared parameters dramatically degrades the performance of our method. See Appendix C for empirical results and further discussion. As an alternative, we suggest updating the gradient weights $\alpha^{(t)}$ once every few iterations instead of every iteration. This simple yet effective solution greatly reduces the runtime (up to $\sim \times 10$ for QM9 and $\sim \times 5$ for MT10) while maintaining high performance. In Section 6.4 we provide experimental results while varying the frequency of task weights update on the QM9 dataset and the MT10 benchmark. Our results show that Nash-MTL runtime can be reduced to about the same as linear scalarization (or STL) while maintaining competitive results compared to other baselines; However, in some cases, we do see a noticeable drop in performance compared with our standard approach.

## 4. Related Work

In multitask learning (MTL), one simultaneously solves several learning problems while sharing information among tasks (Caruana, 1997; Ruder, 2017), commonly through a joint hidden representation (Zhang et al., 2014; Dai et al., 2016; Pinto & Gupta, 2017; Zhao et al., 2018; Liu et al., 2019b). Studies in the literature proposed several explanations for the difficulty in the optimization process of MTL, such as conflicting gradients (Wang et al., 2020; Yu et al., 2020a), or plateaus in the loss landscape (Schaul et al., 2019). Other studies aimed at improving multitask learning by proposing novel architectures (Misra et al., 2016; Hashimoto et al., 2017; Liu et al., 2019b; Chen et al., 2020). We focus on weighting the gradients of the tasks via an axiomatic approach that is agnostic to the architecture used. Studies in a similar vein proposed to weigh the task losses with various approaches, such as the uncertainty of the tasks (Kendall et al., 2018), the norm of the gradients (Chen et al., 2018), random weights (Lin et al., 2021), and similarity of the gradients (Du et al., 2018; Suteu & Guo, 2019). These methods are mostly heuristic and can have unstable performance (Liu et al., 2021a). Recently, several studies proposed MTL approaches based on the multiple-gradient descent algorithm (MGDA) for multi-objective optimization (Désidéri, 2012). This is an appealing approach since, under mild conditions, convergence to a Pareto stationary point is guaranteed. Sener & Koltun (2018) cast the multi-objective problem to multi-task problem and suggest task weighting based on the Frank-Wolfe algorithm (Jaggi, 2013). Liu et al. (2021a) searches for an update direction in a neighborhood of the average gradient that maximizes the worst improvement of any task. Unlike these studies, we propose an MTL approach based on a Bargaining game that can find solutions that are Pareto optimal and proportionally fair.

The closest work to our approach, to the best of our knowledge, is Liu et al. (2021b). There, the authors propose to look for a fair gradient direction where all the cosine similarities are equal. We note that this update direction satisfies all of the Nash axioms except for Pareto optimally. Thus, unlike our proportionally fair approach, it can settle for a sub-optimal solution for the sake of fairness.

Finally, we note that the Nash bargaining solution was effectively applied to problems in various fields such as communication (Zhang et al., 2008; Leshem & Zehavi, 2011; Shi et al., 2018), economics (Dagan & Volij, 1993), and computing (Grosu et al., 2002), and to several learning setups, such as reinforcement learning (Qiao et al., 2006), Bayesian optimization (Binois et al., 2020), clustering (Rezaee et al., 2021), federated learning (Kim, 2021), and multi-armed bandits (Baek & Farias, 2021).

*Table 2. NYUv2*. Test performance for three tasks: semantic segmentation, depth estimation, and surface normal. Values are averages over 3 random seeds.

| | Segmentation | | Depth | | Surface Normal | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Angle Distance ↓ | | Within $t^\circ$ ↑ | | | **MR** ↓ | $\mathbf{\Delta m}\%$ ↓ |
| | mIoU ↑ | Pix Acc ↑ | Abs Err ↓ | Rel Err ↓ | Mean | Median | 11.25 | 22.5 | 30 | | |
| STL | 38.30 | 63.76 | 0.6754 | 0.2780 | 25.01 | 19.21 | 30.14 | 57.20 | 69.15 | | |
| LS | 39.29 | 65.33 | 0.5493 | 0.2263 | 28.15 | 23.96 | 22.09 | 47.50 | 61.08 | 8.11 | 5.59 |
| SI | 38.45 | 64.27 | 0.5354 | 0.2201 | 27.60 | 23.37 | 22.53 | 48.57 | 62.32 | 7.11 | 4.39 |
| RLW | 37.17 | 63.77 | 0.5759 | 0.2410 | 28.27 | 24.18 | 22.26 | 47.05 | 60.62 | 10.11 | 7.78 |
| DWA | 39.11 | 65.31 | 0.5510 | 0.2285 | 27.61 | 23.18 | 24.17 | 50.18 | 62.39 | 6.88 | 3.57 |
| UW | 36.87 | 63.17 | 0.5446 | 0.2260 | 27.04 | 22.61 | 23.54 | 49.05 | 63.65 | 6.44 | 4.05 |
| MGDA | 30.47 | 59.90 | 0.6070 | 0.2555 | **24.88** | **19.45** | **29.18** | **56.88** | **69.36** | 5.44 | 1.38 |
| PCGrad | 38.06 | 64.64 | 0.5550 | 0.2325 | 27.41 | 22.80 | 23.86 | 49.83 | 63.14 | 6.88 | 3.97 |
| GradDrop | 39.39 | 65.12 | 0.5455 | 0.2279 | 27.48 | 22.96 | 23.38 | 49.44 | 62.87 | 6.44 | 3.58 |
| CAGrad | 39.79 | 65.49 | 0.5486 | 0.2250 | 26.31 | 21.58 | 25.61 | 52.36 | 65.58 | 3.77 | 0.20 |
| IMTL-G | 39.35 | 65.60 | 0.5426 | 0.2256 | 26.02 | 21.19 | 26.2 | 53.13 | 66.24 | 3.11 | −0.76 |
| Nash-MTL | **40.13** | **65.93** | **0.5261** | **0.2171** | 25.26 | 20.08 | 28.4 | 55.47 | 68.15 | **1.55** | **−4.04** |

## 5. Analysis

We now analyze the convergence of our method in the convex and non-convex cases. As even single-task non-convex optimization might only converge to a stationary point, we will prove convergence to a Pareto stationary point, i.e., a point where some convex combination of the gradients is zero. As stated, we also assume that the gradients are independent while not at a Pareto stationary point. Independence of the gradients is a slightly stronger assumption than Pareto stationarity but is needed to exclude degenerate edge cases such as two identical tasks.

We note that by substituting local Pareto optimality for Pareto stationarity in Assumption 5.1 we can show convergence to a local Pareto optimal point. However, this assumption has strong implications, as it implies we avoid local maxima and saddle points of any specific task. Since our update rule is a descent direction for all tasks, we can reasonably assume that our algorithm avoids local maxima points. Furthermore, it was shown that first-order methods avoid saddle points (Panageas et al., 2019), giving credence to this stronger assumption. Nevertheless, we take a conservative approach and state our results with the weaker assumption.

We formally make the following assumptions:

**Assumption 5.1.** We assume that for a sequence $\{\theta^{(t)}\}_{t=1}^{\infty}$ generated by our algorithm, the set of the gradient vectors $g_1^{(t)}, ..., g_K^{(t)}$ at any point on the sequence and at any partial limit are linearly independent unless that point is a Pareto stationary point.

**Assumption 5.2.** We assume that all loss functions are differentiable, bounded below and that all sub-level sets are bounded. The input domain is open and convex.

**Assumption 5.3.** We assume that all the loss functions are L-smooth,

$$||\nabla \ell_i(x) - \nabla \ell_i(y)|| \leq L||x - y|| \quad . \tag{6}$$

**Theorem 5.4.** *Let $\{\theta^{(t)}\}_{t=1}^{\infty}$ be the sequence generated by the update rule $\theta^{(t+1)} = \theta^{(t)} - \mu^{(t)}\Delta\theta^{(t)}$ where $\Delta\theta^{(t)} = \sum_{i=1}^{K} \alpha_i^{(t)} g_i^{(t)}$ is the Nash bargaining solution $(G^{(t)})^\top G^{(t)} \alpha^{(t)} = 1/\alpha^{(t)}$. Set $\mu^{(t)} = \min_{i\in[K]} \frac{1}{LK\alpha_i^{(t)}}$. Then, the sequence $\{\theta^{(t)}\}_{t=1}^{\infty}$ has a subsequence that converges to a Pareto stationary point $\theta^*$. Moreover all the loss functions $(\ell_1(\theta^{(t)}), ..., \ell_K(\theta^{(t)}))$ converge to $(\ell_1(\theta^*), ..., \ell_K(\theta^*))$.*

*Proof sketch.* We can show that $\mu^{(t)} = \min_i \frac{1}{\alpha_i^{(t)}} \to 0$ so $||\alpha^{(t)}|| \to \infty$. We also show that $||1/\alpha^{(t)}||$ is bounded. As $(G^{(t)})^\top G^{(t)} \alpha^{(t)} = 1/\alpha^{(t)}$ this means that the smallest singluar value of $(G^{(t)})^\top G^{(t)}$ must converge to zero. From compactness $\{\theta^{(t)}\}_{t=1}^{\infty}$ has a converging subsequence whose limit we denote as $\theta^*$. From continuity we get that the gradients Gram matrix $G^\top G$ computed at $\theta^*$ must have a zero singular value and therefore the gradients are linearly dependent. From our assumption this means that $\theta^*$ is Pareto stationary. As the losses are monotonically decearsing and bounded below they must converge and to the subsequence limit of $(\ell_1(\theta^*), ..., \ell_K(\theta^*))$. □

If we also assume convexity, we can strengthen our claim

**Theorem 5.5.** *Let $\{\theta^{(t)}\}_{t=1}^{\infty}$ be the sequence generated by the update rule $\theta^{(t+1)} = \theta^{(t)} - \mu^{(t)}\Delta\theta^{(t)}$ where $\Delta\theta^{(t)} = \sum_{i=1}^{K} \alpha_i^{(t)} g_i^{(t)}$ is the Nash bargaining solution $(G^{(t)})^\top G^{(t)} \alpha^{(t)} = 1/\alpha^{(t)}$. Set $\mu^{(t)} = \min_{i\in[K]} \frac{1}{LK\alpha_i^{(t)}}$. If*

*Table 3. CityScapes.* Test performance for two tasks: semantic segmentation and depth estimation. Value are averages over 3 random seeds.

| | Segmentation | | Depth | | | |
|---|---|---|---|---|---|---|
| | mIoU ↑ | Pix Acc ↑ | Abs Err ↓ | Rel Err↓ | **MR** ↓ | $\mathbf{\Delta_m}\%$ ↓ |
| STL | 74.01 | 93.16 | 0.0125 | 27.77 | | |
| LS | 75.18 | 93.49 | 0.0155 | 46.77 | 6.12 | 22.60 |
| SI | 70.95 | 91.73 | 0.0161 | 33.83 | 8.00 | 14.11 |
| RLW | 74.57 | 93.41 | 0.0158 | 47.79 | 9.25 | 24.38 |
| DWA | 75.24 | 93.52 | 0.0160 | 44.37 | 6.00 | 21.45 |
| UW | 72.02 | 92.85 | 0.0140 | **30.13** | 5.25 | **5.89** |
| MGDA | 68.84 | 91.54 | 0.0309 | 33.50 | 8.75 | 44.14 |
| PCGrad | 75.13 | 93.48 | 0.0154 | 42.07 | 6.37 | 18.29 |
| GradDrop | 75.27 | 93.53 | 0.0157 | 47.54 | 5.50 | 23.73 |
| CAGrad | 75.16 | 93.48 | 0.0141 | 37.60 | 5.37 | 11.64 |
| IMTL-G | 75.33 | 93.49 | 0.0135 | 38.41 | 3.62 | 11.10 |
| Nash-MTL | **75.41** | **93.66** | **0.0129** | 35.02 | **1.75** | 6.82 |

*we assume that all the loss functions are convex, then the sequence* $\{\theta^{(t)}\}_{t=1}^{\infty}$ *converges to a Pareto optimal point* $\theta^*$.

See the full proofs in the appendix Sec. A.

## 6. Experiments

We evaluate Nash-MTL on diverse multi-task learning problems. The experiments show the superiority of Nash-MTL over previous MTL methods. To support future research and the reproducibility of the results, we will make our source code publicly available. Additional experimental results and details are provided in Appendix B.

**Compared methods:** We compare the following approaches: **(1)** Our proposed Nash-MTL algorithm described in Section 3; **(2)** Single task learning (STL), training an independent model for each task; **(3)** Linear scalarization (LS) baseline which minimizes $\sum_k \ell_k$; **(4)** Scale-invariant (SI) baseline which minimizes $\sum_k \log \ell_k$. This baseline is invariant to rescaling each loss with a positive number; **(5)** Dynamic Weight Average (DWA) (Liu et al., 2019b) adjusts task weights based on the rates of loss changes over time; **(6)** Uncertainty weighting (UW) (Kendall et al., 2018) uses task uncertainty quantification to adjust task weights; **(7)** MGDA (Sener & Koltun, 2018) finds a convex combination of gradients with a minimal norm; **(8)** Random loss weighting (RLW) with normal distribution, scales the losses according to randomly sampled task weights (Lin et al., 2021); **(9)** PCGrad (Yu et al., 2020a) removes conflicting components of each gradient w.r.t the other gradients; **(10)** GradDrop (Chen et al., 2020) randomly drops components of the task gradients based on how much they conflict; **(11)** CAGrad (Liu et al., 2021a) optimizes for the average loss while explicitly controlling the minimum decrease rate across tasks; **(12)** IMTL-G (Liu et al., 2021b) uses an update direction with equal projections on task gradients. IMTL-G is applied to the feature-level gradients, as was suggested by the authors. We also tried applying IMTL-G to the shared-parameters gradient for a fair comparison, but its performance was even worse.

**Evaluation.** For each experiment, we report the common evaluation metrics for each task. Since naturally MTL does not carry a single objective and since the scale of per-task metrics often varies significantly, we report two metrics that capture the overall performance: **(1)** $\mathbf{\Delta_m}\%$, the average per-task performance drop of method $m$ relative to the STL baseline denoted $b$. Formally, $\Delta_m\% = \frac{1}{K}\sum_{k=1}^{K}(-1)^{\delta_k}(M_{m,k} - M_{b,k})/M_{b,k}$, where $M_{b,k}$ is the value of metric $M_k$ obtained by the baseline and $M_{m,k}$ by the compared method. $\delta_k = 1$ if a higher value is better for a metric $M_k$ and 0 otherwise (Maninis et al., 2019; Liu et al., 2021a). **(2) Mean Rank (MR):** The average rank of each method across the different tasks (lower is better). A method receives the best value, MR $= 1$, if it ranks first in all tasks.

### 6.1. Multi-Task Regression for QM9

We evaluate Nash-MTL on predicting 11 properties of molecules from the QM9 dataset (Ramakrishnan et al., 2014), a widely used benchmark for graph neural networks. QM9 consists of $\sim 130K$ molecules represented as graphs annotated with both node and edge features. We used the QM9 example in PyTorch Geometric (Fey & Lenssen, 2019), and use 110K molecules for training, 10K for validation, and 10K as a test set. As each task target range is at a different scale, this could be an issue for other methods that are not scale-invariant like ours. For fairness, we normalized each task target to have zero mean and unit standard deviation. We use the popular GNN model from

Gilmer et al. (2017), a network comprised of several concatenated message passing layers, which update the node features based on both node and edge features, followed by the pooling operator from Vinyals et al. (2015). Specifically, we used the implementation from Fey & Lenssen (2019). We train each method for 300 epochs and search for the best learning-rate (lr) given by the $\Delta_m$ performance on the validation set. We use a learning-rate scheduler to reduce the lr once the validation $\Delta_m$ metric has stopped improving. The validation set is also used for early stopping.

Predicting molecular properties in QM9 poses a significant challenge for MTL methods because the number of tasks is large and because the loss scales vary significantly. The scale issue is only partially resolved by normalization because some tasks are easier to learn than others. Prior work found that single-task learning significantly improves performance on all targets compared to MTL methods (Maron et al., 2019; Klicpera et al., 2020).

Results are shown in Figure 3 and Table 1. Nash-MTL achieves the best performance in terms of both MR and $\Delta_m$. Interestingly, most MTL methods fall short compared to the simple scale-invariant baseline, which ignores gradient interaction, except for IMTL-G whose performance is on par with this baseline. This result shows that the scale-invariant property of our approach can be beneficial. See Appendix D.1 for the per-task evaluation results.

### 6.2. Scene Understanding

We follow the protocol of (Liu et al., 2019b) and evaluate Nash-MTL on the NYUv2 and Cityscapes datasets (Silberman et al., 2012; Cordts et al., 2016). NYUv2 is an indoor scene dataset that consists of 1449 RGBD images and dense per-pixel labeling with 13 classes. We use this dataset as a multitask learning benchmark for semantic segmentation, depth estimation, and surface normal prediction.

The CityScapes dataset (Cordts et al., 2016) contains 5000 high-resolution street-view images with dense per-pixel annotations. We use this dataset as a multitask learning benchmark for semantic segmentation and depth estimation. To speed up the training phase, all images were resized to $128 \times 256$. The original dataset contains 19 categories for pixel-wise semantic segmentation, together with ground-truth depth maps. For segmentation, we used a coarser version of the labels with 7 classes.

For all MTL methods, we train a Multi-Task Attention Network (MTAN) (Liu et al., 2019b) which adds an attention mechanism on top of the SegNet architecture (Badrinarayanan et al., 2017). We follow the training procedure from Liu et al. (2019b); Yu et al. (2020a); Liu et al. (2021a). Each method is trained for 200 epochs with the Adam optimizer (Kingma & Ba, 2015) and an initial learning-rate of $1e-4$. The learning-rate is halved to $5e-5$ after 100 epochs. As in (Liu et al., 2021a) The STL baseline refers to training task-specific SegNet models.

*Table 4. MT10.* Average success over 10 random seeds.

| | Success $\pm$ SEM |
|---|---|
| STL SAC | $0.90 \pm 0.032$ |
| MTL SAC | $0.49 \pm 0.073$ |
| MTL SAC + TE | $0.54 \pm 0.047$ |
| MH SAC | $0.61 \pm 0.036$ |
| SM | $0.73 \pm 0.043$ |
| CARE | $0.84 \pm 0.051$ |
| PCGrad | $0.72 \pm 0.022$ |
| CAGrad | $0.83 \pm 0.045$ |
| Nash-MTL | $\mathbf{0.91 \pm 0.031}$ |

The results are presented in Table 2 and Table 3. Our method, Nash-MTL, achieves the best MR in both datasets, the best $\Delta_m$ in NYUv2 and the seconds to best $\Delta_m$ in the CityScapes experiment. Nash-MTL performance is well balanced across tasks. MGDA is primarily focused on the task of predicting surface normals and achieves poor performance on the other two tasks. The inherent biasedness of MGDA towards the task with the smallest gradient magnitude was previously discussed in Liu et al. (2021b). We note that the optimal solution under Nash-MTL for the two tasks case is equivalent to independently normalizing each gradient and summing with equal weights. While this is a fairly simple approach for MTL, we show that it outperforms almost all the compared MTL methods on the two-tasks CityScapes benchmark.

### 6.3. Multi-Task Reinforcement Learning

We consider a multi-task RL problem and evaluate Nash-MTL on the MT10 environment from the Meta-World benchmark (Yu et al., 2020b). This benchmark involves a simulated robot trained to perform actions like pressing a button and opening a window, each action treated as a task, for a total of 10 tasks. The goal is to learn a policy that can succeed across all the diverse sets of manipulation tasks. Following previous works on MTL-RL (Yu et al., 2020a; Liu et al., 2021a; Sodhani et al., 2021), we use Soft Actor-Critic (SAC) (Haarnoja et al., 2018) as the base RL algorithm. Along with the MTL methods (1) CAGrad (Liu et al., 2021a) and (2) PCGrad (Yu et al., 2020a) applied to a shared model SAC, we evaluate the following methods: (3) STL, one SAC model per task; (4) MTL SAC with a shared model; (5) Multi-task SAC with task encoder (MTL SAC + TE, Yu et al. (2020b)); (6) Multi-headed SAC (MH SAC) with task-specific heads (Yu et al., 2020b); (7) Soft Modularization (SM, Yang et al. (2020)) which estimates per-task routes for different tasks in a shared model, and;

*Table 5.* Training runtime per episode and average success for the MT10 benchmark, computed over 10 random seeds while varying the frequency of task weights updates in Nash-MTL.

| | Success $\pm$ SEM | Runtime[Sec.] |
|---|---|---|
| MTL-SAC | $0.49 \pm 0.073$ | 7.3 |
| PCGrad | $0.72 \pm 0.022$ | 9.7 |
| CAGrad | $0.83 \pm 0.045$ | 20.9 |
| Nash-MTL | $0.91 \pm 0.031$ | 40.7 |
| Nash-MTL-50 | $0.85 \pm 0.022$ | 8.6 |
| Nash-MTL-100 | $0.87 \pm 0.033$ | 7.9 |

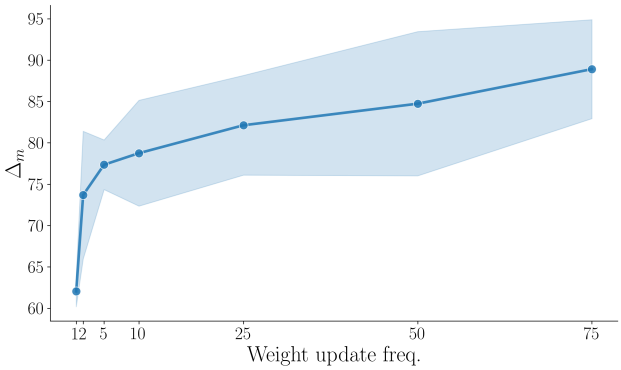



*Figure 4.* Test $\Delta_m$ for the QM9 dataset, averaged over 3 random seeds, for different intervals of task weights update.

(8) CARE (Sodhani et al., 2021) which utilizes language metadata and employs a mixture of encoders. We follow the same experiment setup from Sodhani et al. (2021); Liu et al. (2021a) to train all methods over 2 million steps and report the mean success over 10 random seeds with fixed evaluation frequency. The results are presented in Table 4.

Nash-MTL achieves the best performance by a large margin. In addition, Nash-MTL is the only MTL method to reach the same performance as the per-task SAC STL baseline.

### 6.4. Scaling-up Nash-MTL

One of the major drawbacks of the SOTA MTL methods is that they require access to all task gradients to compute the optimal update direction (Sener & Koltun, 2018; Yu et al., 2020a; Liu et al., 2021b;a). This requires one to perform $K$ backward passes at each optimization step, thus scales poorly with the number of tasks. Previous works suggested using a subset of tasks (Liu et al., 2021a) or replacing the task gradients with the feature-level gradient (Sener & Koltun, 2018; Liu et al., 2021b; Javaloy & Valera, 2021) as potential speedups. In our experiments, we found that using the feature-level gradients can greatly reduce Nash-MTL performance (Appendix C). However, here we show that the simple solution of updating task weights less frequently maintains good performance while dramatically reducing the training time.

One approach to alleviate this issue is to update the task weights less frequently, and use these weights in subsequent steps. We evaluate this approach using the QM9 dataset and the MT10 benchmark and present the result in Figure 4 and Table 5. We denote Nash-MTL with task weight update every $T$ optimization steps with Nash-MTL-$T$.

The results show that Nash-MTL is fairly robust to varying intervals between weights updates. While this simple approach results in a small degradation in performance, it can dramatically decrease the training time of our method. For example, on the QM9, updating the weights every 5/50 steps results in a $\times 3.7/9.8$ speedup w.r.t updating the weights at each step. On the MT10 environment, updating the weights every 100 steps result in $\sim \times 10$ speedup (only $\sim \times 1.1$ slower than the fastest baseline) while outperforming all other MTL baseline method (Table 5).

## 7. Conclusion

In this work, we present Nash-MTL, a novel and principled approach for multitask learning. We frame the gradient combination step in MTL as a bargaining game and use the Nash bargaining solution to find the optimal update direction. We highlight the importance of the scale invariance approach for multitask learning, specifically for setups with varying loss scales and gradient magnitudes. We provide a theoretical convergence analysis for Nash-MTL, showing that it converges to a Pareto optimal and Pareto stationary points in the convex and non-convex settings, respectively. Finally, our experiments show that Nash-MTL achieves state-of-the-art results on various benchmarks across multiple domains.

## 8. Acknowledgements

This work was funded by the Israeli innovation authority through the AVATAR consortium; by the Israel Science Foundation (ISF grant 737/2018); and by an equipment grant to GC and Bar Ilan University (ISF grant 2332/18).

## References

Achituve, I., Maron, H., and Chechik, G. Self-supervised learning for domain adaptation on point clouds. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 123–133, 2021.

Badrinarayanan, V., Kendall, A., and Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

Baek, J. and Farias, V. F. Fair exploration via axiomatic bargaining. *arXiv preprint arXiv:2106.02553*, 2021.

Baxter, J. A model of inductive bias learning. *J. Artif. Intell. Res.*, 2000.

Binois, M., Picheny, V., Taillandier, P., and Habbal, A. The Kalai-Smorodinsky solution for many-objective Bayesian optimization. *J. Mach. Learn. Res.*, 21(150):1–42, 2020.

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

Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pp. 794–803. PMLR, 2018.

Chen, Z., Ngiam, J., Huang, Y., Luong, T., Kretzschmar, H., Chai, Y., and Anguelov, D. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *ArXiv*, abs/2010.06808, 2020.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Crawshaw, M. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

Dagan, N. and Volij, O. The bankruptcy problem: a cooperative bargaining approach. *Mathematical Social Sciences*, 26(3):287–297, 1993.

Dai, J., He, K., and Sun, J. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3150–3158, 2016.

Désidéri, J.-A. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

Du, Y., Czarnecki, W. M., Jayakumar, S. M., Farajtabar, M., Pascanu, R., and Lakshminarayanan, B. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Grosu, D., Chronopoulos, A. T., and Leung, M.-Y. Load balancing in distributed systems: An approach using cooperative games. In *Proceedings 16th International Parallel and Distributed Processing Symposium*, pp. 10–pp. IEEE, 2002.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement, 2018.

Hashimoto, K., Xiong, C., Tsuruoka, Y., and Socher, R. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1923–1933, 2017.

Jaggi, M. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pp. 427–435. PMLR, 2013.

Javaloy, A. and Valera, I. Rotograd: Dynamic gradient homogenization for multi-task learning. *arXiv preprint arXiv:2103.02631*, 2021.

Kendall, A., Gal, Y., and Cipolla, R. 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*, pp. 7482–7491, 2018.

Kim, S. Cooperative federated learning-based task offloading scheme for tactical edge networks. *IEEE Access*, 9: 145739–145747, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. *ArXiv*, abs/2003.03123, 2020.

Leshem, A. and Zehavi, E. Smart carrier sensing for distributed computation of the generalized nash bargaining solution. In *2011 17th International Conference on Digital Signal Processing (DSP)*, pp. 1–5. IEEE, 2011.

Lin, B., Ye, F., and Zhang, Y. A closer look at loss weighting in multi-task learning. *arXiv preprint arXiv:2111.10603*, 2021.

Lipp, T. and Boyd, S. Variations and extension of the convex–concave procedure. *Optimization and Engineering*, 17(2):263–287, 2016.

Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34, 2021a.

Liu, L., Li, Y., Kuang, Z., Xue, J.-H., Chen, Y., Yang, W., Liao, Q., and Zhang, W. Towards impartial multi-task learning. In *International Conference on Learning Representations*, 2021b.

Liu, S., Davison, A., and Johns, E. Self-supervised generalisation with meta auxiliary learning. *Advances in Neural Information Processing Systems*, 32, 2019a.

Liu, S., Johns, E., and Davison, A. J. End-to-end multi-task learning with attention. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1871–1880, 2019b.

Maninis, K.-K., Radosavovic, I., and Kokkinos, I. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1851–1860, 2019.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. *arXiv preprint arXiv:1905.11136*, 2019.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3994–4003, 2016.

Nash, J. Two-person cooperative games. *Econometrica*, 21 (1):128–140, 1953. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1906951.

Navon, A., Achituve, I., Maron, H., Chechik, G., and Fetaya, E. Auxiliary learning by implicit differentiation. In *International Conference on Learning Representations (ICLR)*, 2021a.

Navon, A., Shamsian, A., Chechik, G., and Fetaya, E. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=NjF772F4ZZR.

Panageas, I., Piliouras, G., and Wang, X. First-order methods almost always avoid saddle points: The case of vanishing step-sizes. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Pinto, L. and Gupta, A. Learning to push by grasping: Using multiple tasks for effective learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2161–2168. IEEE, 2017.

Qiao, H., Rozenblit, J., Szidarovszky, F., and Yang, L. Multi-agent learning model with bargaining. In *Proceedings of the 2006 winter simulation conference*, pp. 934–940. IEEE, 2006.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Rezaee, M. J., Eshkevari, M., Saberi, M., and Hussain, O. GBK-means clustering algorithm: An improvement to the K-means algorithm based on the bargaining game. *Knowledge-Based Systems*, 213:106672, 2021.

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

Schaul, T., Borsa, D., Modayil, J., and Pascanu, R. Ray interference: a source of plateaus in deep reinforcement learning. *arXiv preprint arXiv:1904.11455*, 2019.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pp. 527–538, 2018.

Shi, C., Wang, F., Salous, S., Zhou, J., and Hu, Z. Nash bargaining game-theoretic framework for power control in distributed multiple-radar architecture underlying wireless communication system. *Entropy*, 20(4):267, 2018.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pp. 746–760. Springer, 2012.

Sodhani, S., Zhang, A., and Pineau, J. Multi-task reinforcement learning with context-based representations. *arXiv preprint arXiv:2102.06177*, 2021.

Sriperumbudur, B. K. and Lanckriet, G. R. On the convergence of the concave-convex procedure. In *Nips*, volume 9, pp. 1759–1767. Citeseer, 2009.

Standley, T., Zamir, A. R., Chen, D., Guibas, L. J., Malik, J., and Savarese, S. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning ICML*, 2020.

Suteu, M. and Guo, Y. Regularizing deep multi-task networks using orthogonal gradients. *arXiv preprint arXiv:1912.06844*, 2019.

Szép, J. and Forgó, F. *Introduction to the Theory of Games*. Springer, 1985.

Thomson, W. Chapter 35 cooperative models of bargaining. volume 2 of *Handbook of Game Theory with Economic Applications*, pp. 1237–1284. Elsevier, 1994.

Vinyals, O., Bengio, S., and Kudlur, M. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Wang, Z., Tsvetkov, Y., Firat, O., and Cao, Y. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations*, 2020.

Yang, R., Xu, H., Wu, Y., and Wang, X. Multi-task reinforcement learning with soft modularization. *ArXiv*, abs/2003.13661, 2020.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, 2020a.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pp. 1094–1100. PMLR, 2020b.

Yuille, A. L. and Rangarajan, A. The concave-convex procedure. *Neural computation*, 15(4):915–936, 2003.

Zhang, Z., Shi, J., Chen, H.-H., Guizani, M., and Qiu, P. A cooperation strategy based on nash bargaining solution in cooperative relay networks. *IEEE Transactions on Vehicular Technology*, 57(4):2570–2577, 2008.

Zhang, Z., Luo, P., Loy, C. C., and Tang, X. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pp. 94–108. Springer, 2014.

Zhao, X., Li, H., Shen, X., Liang, X., and Wu, Y. A modulation module for multi-task learning with applications in image retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 401–416, 2018.

# A. Proofs

**Lemma A.1.** *If $\mathcal{L}$ is differential and L-smooth (assumption 5.3) then $\mathcal{L}(\theta') \leq \mathcal{L}(\theta) + \nabla\mathcal{L}(\theta)^\top(\theta' - \theta) + \frac{L}{2}\|\theta' - \theta\|^2$.*

*Proof.* Fix $\theta, \theta' \in \mathrm{dom}(\mathcal{L}) \subseteq \mathbb{R}^d$. Since $\mathrm{dom}(\mathcal{L})$ is a convex and open set, there exists $\epsilon > 0$ such that $\theta + t(\theta' - \theta) \in \mathrm{dom}(\mathcal{L})$ for all $t \in [-\epsilon, 1+\epsilon]$. Set $\epsilon > 0$ to be such a number. Thus, we can define a function $\bar{\mathcal{L}} : [-\epsilon, 1+\epsilon] \to \mathbb{R}$ by $\bar{\mathcal{L}}(t) = \mathcal{L}(\theta + t(\theta' - \theta))$. With this, $\bar{\mathcal{L}}(1) = \mathcal{L}(\theta')$, $\bar{\mathcal{L}}(0) = \mathcal{L}(\theta)$, and $\nabla\bar{\mathcal{L}}(t) = \nabla\mathcal{L}(\theta + t(\theta' - \theta))^\top(\theta' - \theta)$ for $t \in [0,1] \subset (-\epsilon, 1+\epsilon)$. From Assumption 5.3, $\|\nabla\mathcal{L}(\theta') - \nabla\mathcal{L}(\theta)\| \leq L\|\theta' - \theta\|$, therefore

$$\begin{aligned}
\|\nabla\bar{\mathcal{L}}(t') - \nabla\bar{\mathcal{L}}(t)\| &= \|[\nabla\mathcal{L}(\theta + t'(\theta' - \theta)) - \nabla\mathcal{L}(\theta + t(\theta' - \theta))^\top(\theta' - \theta)\| \\
&\leq \|\theta' - \theta\|\|\nabla\mathcal{L}(\theta + t'(\theta' - \theta)) - \nabla\mathcal{L}(\theta + t(\theta' - \theta))\| \\
&\leq L\|\theta' - \theta\|\|(t' - t)(\theta' - \theta)\| \\
&\leq L\|\theta' - \theta\|^2\|t' - t\|.
\end{aligned}$$

Hence, $\nabla\bar{\mathcal{L}} : [0,1] \to \mathbb{R}$ is Lipschitz continuous, and therefore continuous. By using the fundamental theorem of calculus with the continuous function $\nabla\bar{\mathcal{L}} : [0,1] \to \mathbb{R}$,

$$\begin{aligned}
\mathcal{L}(\theta') &= \mathcal{L}(\theta) + \int_0^1 \nabla\mathcal{L}(\theta + t(\theta' - \theta))^\top(\theta' - \theta)dt \\
&= \mathcal{L}(\theta) + \nabla\mathcal{L}(\theta)^\top(\theta' - \theta) + \int_0^1 \left(\nabla\mathcal{L}(\theta + t(\theta' - \theta)) - \nabla\mathcal{L}(\theta)\right)^\top(\theta' - \theta)dt \\
&\leq \mathcal{L}(\theta) + \nabla\mathcal{L}(\theta)^\top(\theta' - \theta) + \int_0^1 \|\nabla\mathcal{L}(\theta + t(\theta' - \theta)) - \nabla\mathcal{L}(\theta)\|\|\theta' - \theta\|dt \\
&\leq \mathcal{L}(\theta) + \nabla\mathcal{L}(\theta)^\top(\theta' - \theta) + \int_0^1 tL\|\theta' - \theta\|^2 dt \\
&= \mathcal{L}(\theta) + \nabla\mathcal{L}(\theta)^\top(\theta' - \theta) + \frac{L}{2}\|\theta' - \theta\|^2. \qquad (7)
\end{aligned}$$

□

**Theorem (5.4).** *Let $\{\theta^{(t)}\}_{t=1}^\infty$ be the sequence generated by the update rule $\theta^{(t+1)} = \theta^{(t)} - \mu^{(t)}\Delta\theta^{(t)}$ where $\Delta\theta^{(t)} = \sum_{i=1}^K \alpha_i^{(t)} g_i^{(t)}$ is the Nash bargaining solution $(G^{(t)})^\top G^{(t)}\alpha^{(t)} = 1/\alpha^{(t)}$. Set $\mu^{(t)} = \min_{i\in[K]} \frac{1}{LK\alpha_i^{(t)}}$. The sequence $\{\theta^{(t)}\}_{t=1}^\infty$ has a subsequence that converges to a Pareto stationary point $\theta^*$. Moreover all the loss functions $(\ell_1(\theta^{(t)}), ..., \ell_K(\theta^{(t)}))$ converge to $(\ell_1(\theta^*), ..., \ell_K(\theta^*))$.*

*Proof.* We first note that if for some step we reach a Pareto stationary solution the algorithm halts and sequence stays fixed at that point and therefore converges; Next, we assume that we never get to an exact Pareto stationary solution at any finite step.

We note that the norm of $\Delta\theta^{(t)}$ is $\sqrt{K}$ as $||\Delta\theta^{(t)}||^2 = \sum_{i=1}^K \alpha_i g_i^\top \Delta\theta^{(t)} = \sum_{i=1}^K \alpha_i \cdot 1/\alpha_i = K$. For each loss $\ell_i$ we have using Lemma A.1

$$\ell_i(\theta^{(t+1)}) \leq \ell_i(\theta^{(t)}) - \mu^{(t)}\nabla\ell_i(\theta^{(t)})^\top\Delta\theta^{(t)} + \frac{L}{2}||\mu^{(t)}\Delta\theta^{(t)}||^2 = \qquad (8)$$

$$\ell_i(\theta^{(t)}) - \mu^{(t)}\frac{1}{\alpha_i^{(t)}} + \frac{(\mu^{(t)})^2 LK}{2} \qquad (9)$$

$$= \ell_i(\theta^{(t)}) - \frac{\mu^{(t)}}{\alpha_i^{(t)}} + \frac{\mu^{(t)}}{2}\min_j \frac{1}{\alpha_j^{(t)}} \leq \ell_i(\theta^{(t)}) - \frac{\mu^{(t)}}{2\alpha_i^{(t)}} < \ell_i(\theta^{(t)}) \qquad (10)$$

This shows that our update decreases all the loss functions. We can average over inequality 9 over all losses and get for $\mathcal{L}(\theta) = \frac{1}{K}\sum_{i=1}^K \ell_i(\theta)$:

$$\mathcal{L}(\theta^{(t+1)}) \leq \mathcal{L}(\theta^{(t)}) - \mu^{(t)} \frac{1}{K} \sum_{i=1}^{K} \frac{1}{\alpha_i^{(t)}} + \frac{(\mu^{(t)})^2 LK}{2} \leq \mathcal{L}(\theta^{(t)}) - LK(\mu^{(t)})^2 + \frac{(\mu^{(t)})^2 LK}{2} = \mathcal{L}(\theta^{(t)}) - \frac{LK(\mu^{(t)})^2}{2}. \tag{11}$$

From this we can conclude that $\sum_{\tau=1}^{t} \frac{LK(\mu^{(\tau)})^2}{2} \leq \mathcal{L}(\theta_1) - \mathcal{L}(\theta^{(t+1)})$. As $\mathcal{L}(\theta^{(t)})$ is bounded below we must have that the infinite series $\sum_{t=1}^{\infty} \frac{LK(\mu^{(t)})^2}{2} < \infty$, and also $\mu^{(t)} \to 0$. It follows that $\min_{i\in[K]} 1/\alpha_i^{(t)} \to 0$ and therefore $||\alpha^{(t)}|| \to \infty$.

We will now show that $||1/\alpha^{(t)}||$ is bounded for $t \to \infty$. As the sequence $\mathcal{L}(\theta^{(t)})$ is decreasing we have that the sequence $\theta^{(t)}$ is in the sublevel set $\{\theta : \mathcal{L}(\theta) \leq \mathcal{L}(\theta_0)\}$ which is closed and bounded and therefore compact. If follows that there exists $M < \infty$ such that $||g_i^{(t)}|| \leq M$ for all $t$ and $i \in [K]$. We have for all $i$ and $t$, $|1/\alpha_i^{(t)}| = |(g_i^{(t)})^T \theta^{(t)}| \leq \sqrt{K}||g_i^{(t)}|| \leq \sqrt{K}M < \infty$, and so $||1/\alpha^{(t)}||$ is bounded. Combining these two results we have $||1/\alpha^{(t)}|| \geq \sigma_K((G^{(t)})^\top G^{(t)})||\alpha^{(t)}||$ where $\sigma_K((G^{(t)})^\top G^{(t)})$ is the smallest singular value of $(G^{(t)})^\top G^{(t)}$. Since the norm of $\alpha^{(t)}$ goes to infinity and the norm $1/\alpha^{(t)}$ is bounded, it follows that $\sigma_K((G^{(t)})^\top G^{(t)}) \to 0$.

Now, since $\{\theta : \mathcal{L}(\theta) \leq \mathcal{L}\theta_0\}$ is compact there exists a subsequence $\theta^{(t_j)}$ that converges to some point $\theta^*$. As $\sigma_K((G^{(t)})^T G^{(t)}) \to 0$ we have from continuity that $\sigma_K(G_*^\top G_*) = 0$ where $G_*$ is the matrix of gradients at $\theta^*$. This means that the gradients at $\theta$ are linearly dependent and therefore $\theta^*$ is Pareto stationary by assumption 5.1. As for all $i$ the sequence $\{\ell_i(\theta^{(t)})\}_{t=1}^{\infty}$ is monotonically decreasing and bounded below they all converges. Since $\ell_i(\theta^*)$ is the limit of a subsequence we get that $\ell_i(\theta^{(t)}) \xrightarrow{t\to\infty} \ell_i(\theta^*)$.

□

We now show that if we add a convexity assumption then we can prove convergence to the Pareto front.

**Theorem (5.5).** *Let $\{\theta^{(t)}\}_{t=1}^{\infty}$ be the sequence generated by the update rule $\theta^{(t+1)} = \theta^{(t)} - \mu^{(t)}\Delta\theta^{(t)}$ where $\Delta\theta^{(t)} = \sum_{i=1}^{K} \alpha_i^{(t)} g_i^{(t)}$ is the Nash bargaining solution $(G^{(t)})^\top G^{(t)} \alpha^{(t)} = 1/\alpha^{(t)}$. Set $\mu^{(t)} = \min_{i\in[K]} \frac{1}{LK\alpha_i^{(t)}}$. If we also assume that all the loss functions are convex then the sequence $\{\theta^{(t)}\}_{t=1}^{\infty}$ converges to a Pareto optimal point $\theta^*$.*

*Proof.* We note that this proof uses intermediate results from the proof of theorem 5.4. Given theorem 5.4 it suffices to prove that the sequence $\{\theta^{(t)}\}_{t=1}^{\infty}$ converges, that would mean it converges to the partial limit $\theta^*$ that is Pareto stationary, and from convexity it would be Pareto optimal (as the optimizer of the convex combination of losses). For a convex and differential loss function, we have

$$\ell(\theta') \geq \ell(\theta) + \nabla\ell(\theta)^\top(\theta' - \theta) \tag{12}$$

We can bound

$$\begin{aligned}
||\theta^{(t+1)} - \theta^*||^2 &= ||\theta^{(t)} - \mu^{(t)}\Delta\theta^{(t)} - \theta^*||^2 && (13)\\
&= ||\theta^{(t)} - \theta^*||^2 + (\mu^{(t)})^2||\Delta\theta^{(t)}||^2 - 2\mu^{(t)}(\Delta\theta^{(t)})^\top(\theta^{(t)} - \theta^*) && (14)\\
&= ||\theta^{(t)} - \theta^*||^2 + (\mu^{(t)})^2 K - 2\mu^{(t)} \sum_i \alpha_i^{(t)} (g_i^{(t)})^\top(\theta^{(t)} - \theta^*) && (15)\\
&\leq ||\theta^{(t)} - \theta^*||^2 + (\mu^{(t)})^2 K + 2\mu^{(t)} \sum_i \alpha_i^{(t)} (\ell_i(\theta^*) - \ell_i(\theta^{(t)})) && (16)\\
&\leq ||\theta^{(t)} - \theta^*||^2 + (\mu^{(t)})^2 K + 2\mu^{(t)} \sum_i \alpha_i^{(t)} (\ell_i(\theta^{(t+1)}) - \ell_i(\theta^{(t)})) && (17)\\
&\leq ||\theta^{(t)} - \theta^*||^2 + (\mu^{(t)})^2 K - 2\mu^{(t)} \sum_i \alpha_i^{(t)} \frac{\mu^{(t)}}{2\alpha_i^{(t)}} && (18)\\
&= ||\theta^{(t)} - \theta^*||^2 && (19)
\end{aligned}$$

In Eq. 15 we use the definition of $\Delta\theta^{(t)}$ and the fact that its norm equals $\sqrt{K}$. In Eq. 16 we use convexity and Eq. 12. Eq. 17 uses the fact that we show the losses are monotonically decreasing and converging to $\ell_i(\theta^*)$. In Eq. 18 we use Eq. 10.

We have that the sequence $||\theta^{(t)} - \theta^*||$ is monotonically decreasing and bounded below by zero. Also, it has a subsequence that converges to zero, and so it must hold that the sequence $||\theta^{(t)} - \theta^*||$ also converge to zero, or equivalently $\theta^{(t)} \to \theta^*$.

□

**Proposition (3.1).** *Denote the objective for the optimization problem in Eq. 5 by* $\phi(\alpha) = \sum_i \beta_i(\alpha) + \varphi(\alpha)$. *Then,* $\phi\left(\alpha^{(\tau+1)}\right) \leq \phi\left(\alpha^{(\tau)}\right)$ *for all* $\tau \geq 1$.

*Proof.* In our concave-convex procedure, we use the following linearization at the $\tau$-th iteration:

$$\tilde{\varphi}_\tau(\alpha) = \varphi(\alpha^{(\tau)}) + \nabla\varphi(\alpha^{(\tau)})^\top(\alpha - \alpha^{(\tau)}).$$

Then,

$$\tilde{\varphi}_\tau(\alpha^{(\tau)}) = \varphi(\alpha^{(\tau)}). \tag{20}$$

Moreover, since $\varphi$ is concave and differentiable, we have that

$$\varphi(\alpha^{(\tau+1)}) \leq \varphi(\alpha^{(\tau)}) + \nabla\varphi(\alpha^{(\tau)})^\top(\alpha^{(\tau+1)} - \alpha^{(\tau)}) = \tilde{\varphi}_\tau(\alpha^{(\tau+1)}). \tag{21}$$

Furthermore, since we minimize the convex objective $\sum_i \beta_i(\alpha) + \tilde{\varphi}(\alpha)$ at each iteration of our concave-convex procedure (in the convex feasible set),

$$\sum_i \beta_i(\alpha^{(\tau)}) + \tilde{\varphi}_\tau(\alpha^{(\tau)}) \geq \sum_i \beta_i(\alpha^{(\tau+1)}) + \tilde{\varphi}_\tau(\alpha^{(\tau+1)}). \tag{22}$$

Using Eq. 20–Eq. 22, we have that

$$\begin{aligned}\phi(\alpha^{(\tau)}) = \sum_i \beta_i(\alpha^{(\tau)}) + \varphi(\alpha^{(\tau)}) = \sum_i \beta_i(\alpha^{(\tau)}) + \tilde{\varphi}_\tau(\alpha^{(\tau)}) &\geq \sum_i \beta_i(\alpha^{(\tau+1)}) + \tilde{\varphi}_t(\alpha^{(\tau+1)}) \\ &\geq \sum_i \beta_i(\alpha^{(\tau+1)}) + \varphi(\alpha^{(\tau+1)}) = \phi(\alpha^{(\tau+1)}).\end{aligned}$$

This proves the statement.

□

## B. Experimental Details

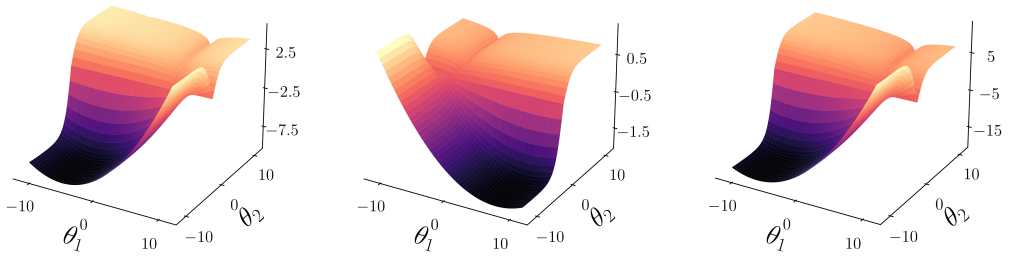
We provide here full experimental details for all experiments described in the main text.

**Implementation Details.** We apply all gradient manipulation methods to the gradients of the shared weights, with the exception of IMTL-G, which was applied to the feature-level gradients, as was originally proposed by the authors. We also tried applying IMTL-G to the shared-parameters gradient for a fair comparison, but it did not perform as well. We set the CAGrad's $c$ hyperparameter to 0.4, which was reported to yield the best performance for NYUv2 and Cityscapes (Liu et al., 2021a). For DWA (Liu et al., 2019b) we set the temperature hyperparameter to 2 which was found empirically to be optimum across all architectures. For RLW (Lin et al., 2021) we sample the weights from a normal distribution.

**QM9.** We adapt the QM9 example in PyTorch Geometric (Fey & Lenssen, 2019), and train the popular GNN model from Gilmer et al. (2017). We use the publicly available[1] implementation, the implementation is provided by Fey & Lenssen (2019). We use 110K molecules for training, 10K for validation, and 10K as a test set. Each task's targets are normalized to have zero mean and unit standard deviation. We train each method for 300 epochs with batch-size of 120 and search for learning-rate (lr) in $\{1e-3, 5e-4, 1e-4\}$. We use a ReduceOnPlateau scheduler to decrease the lr when the validation $\Delta_m$ metric stops improving. Additionally, we use the validation $\Delta_m$ for early stopping.

**Scene Understanding.** We follow the training and evaluation procedure used in previous work on MTL (Liu et al., 2019b; Yu et al., 2020a; Liu et al., 2021a). However, unlike (Liu et al., 2019b), we add data augmentations (DA) during training for

[1] https://github.com/pyg-team/pytorch_geometric/blob/master/examples/qm9_nn_conv.py



(a) Average loss (b) $\ell_1$ (c) $\ell_2$

*Figure 5. Illustrative example*. Visualization of the loss surfaces in our illustrative example of Figure 1

all the compared methods, similar to (Liu et al., 2021a;b). We train each method for 200 epochs with an initial learning-rate of $1e-4$. The learning-rate is reduced to $5e-5$ after 100 epochs. For MTL methods, we train a Multi-Task Attention Network (MTAN) (Liu et al., 2019b) built upon SegNet (Badrinarayanan et al., 2017). Similar to previous works (Liu et al., 2021a), the STL baseline refers to training task-specific SegNet models. We use a batch size of 2 and 8 for NYUv2 and CityScapes respectively. To align with previous work on MTL Liu et al. (2019b); Yu et al. (2020a); Liu et al. (2021a) we report the test performance averaged over the last 10 epochs.

**MT10.** Following previous works (Yu et al., 2020a; Liu et al., 2021a; Sodhani et al., 2021), we use multitask Soft Actor-Critic (SAC) (Haarnoja et al., 2018) as the base RL algorithm for PCGrad, CAGrad, and Nash-MTL. We follow the same experiment setup from and evaluation protocol as in Sodhani et al. (2021); Liu et al. (2021a). Each method is trained over 2 million steps with a batch size of 1280. The agent is evaluated once every 10K environment steps to obtain the average success over tasks. The reported success rate for the agent is the best average performance over all evaluation steps. We repeat this procedure over 10 random seeds, and the performance of each method is obtained by averaging the mean success over all random seeds. For all Nash-MTL experiments, we use a single CCP step in order to speed up computation.

**Illustrative Example.** We provide here the details for the illustrative example of Figure 1. We use a slightly modified version of the illustrative example in (Liu et al., 2021a). We first present the learning problem from (Liu et al., 2021a): Let $\theta = (\theta_1, \theta_2) \in \mathbb{R}^2$, and consider the following objectives:

$$
\begin{aligned}
&\tilde{\ell}_1(\theta) = c_1(\theta) f_1(\theta) + c_2(\theta) g_1(\theta) \quad \text{and} \quad \tilde{\ell}_2(\theta) = c_1(\theta) f_2(\theta) + c_2(\theta) g_2(\theta), \text{where} \\
&f_1(\theta) = \log(\max(|0.5(-\theta_1 - 7) - \tanh(-\theta_2)|, 5e-6)) + 6, \\
&f_2(\theta) = \log(\max(|0.5(-\theta_1 + 3) - \tanh(-\theta_2) + 2|, 5e-6)) + 6, \\
&g_1(\theta) = ((-\theta_1 + 7)^2 + 0.1 \cdot (-\theta_2 - 8)^2)/10 - 20, \\
&g_2(\theta) = ((-\theta_1 - 7)^2 + 0.1 \cdot (-\theta_2 - 8)^2)/10 - 20, \\
&c_1(\theta) = \max(\tanh(0.5\theta_2), 0) \quad \text{and} \quad c_2(\theta) = \max(\tanh(-0.5\theta_2), 0)
\end{aligned}
$$

We now set $\ell_1 = 0.1 \cdot \tilde{\ell}_1$ and $\ell_2 = \tilde{\ell}_2$ as our objectives, see Figure 5. We use five different initialization points $\{(-8.5, 7.5), (0.0, 0.0), (9.0, 9.0), (-7.5, -0.5), (9, -1.0)\}$. We use the Adam optimizer and train each method for 35K iteration with learning rate of $1e-3$.

## C. Computing Task Gradient at the Features-Level

One common approach for speeding and scaling up MTL methods is using feature-level gradients (from the representation layer) as a surrogate for the task-level gradients computed over the entire shared backbone (Sener & Koltun, 2018; Liu et al., 2021b; Javaloy & Valera, 2021). In this section we evaluate Nash-MTL while using the feature-level gradients for computing the Nash bargaining solution. On the QM9 dataset, we found this approach to accelerate training by $\sim \times 6$. However, this acceleration method greatly hurts the performance of Nash-MTL, yielding a test $\Delta_m$ of 179.2 (compared to 62.0 when using full gradients). This result is not surprising, since we are mainly interested in the inner products of gradients. Consider

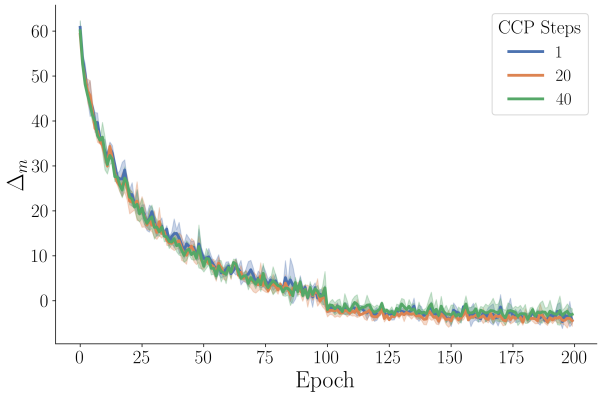



*Figure 6. NYUv2.* The mean and standard divination of test $\Delta_m$ throughout the training process, for Nash-MTL with 1, 20, and 40 CCP steps.

$g_i^\top g_j = (\nabla_\theta z \nabla_z \ell_i)^\top \nabla_\theta z \nabla_z \ell_j$, where $z$ is the feature representation and $\theta$ the shared parameters vector. We see that for $\nabla_z \ell_i^\top \nabla_z \ell_j$ to accurately approximate $g_i^\top g_j$ we need $\nabla_\theta z^\top \nabla_\theta z \approx I$ which is a strong and restricting requirement.

## D. Additional Experiments

### D.1. Full Results for Multi-task Regression

We provide here the full results for the QM9 experiment of Section 6.1. The results for all methods over all 11 tasks are presented in Table 6. Nash-MTL achieves the best $\Delta_m$ and MR performance. Despite being a simple approach, *SI* performs well compared to more sophisticated baselines. It achieves the third/second best $\Delta_m$ and MR respectively. The other scale-invariant method, *IMTL-G*, also performs well in this learning setup.

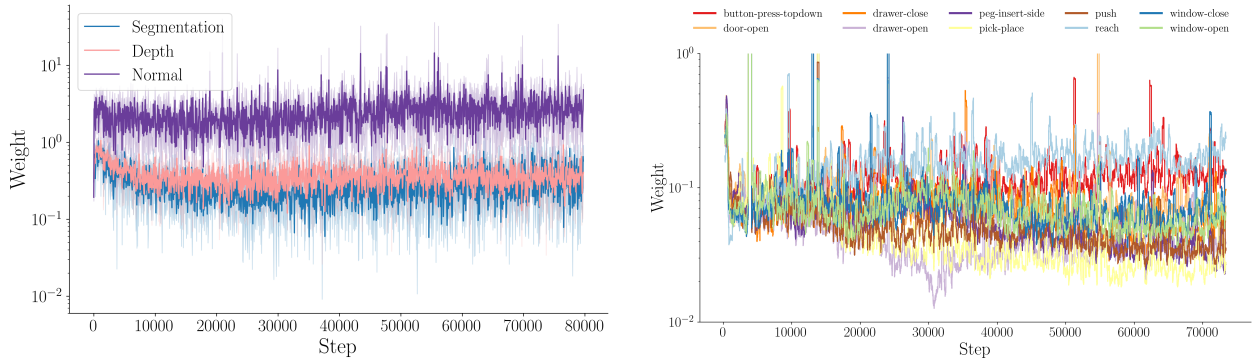
*Table 6. QM9.* Test performance averaged over 3 random seeds.

| | $\mu$ | $\alpha$ | $\epsilon_{\text{HOMO}}$ | $\epsilon_{\text{LUMO}}$ | $\langle R^2 \rangle$ | ZPVE | $U_0$ | $U$ | $H$ | $G$ | $c_v$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAE ↓ | | | | | | | | | | | **MR** ↓ | $\mathbf{\Delta_m}\%$ ↓ |
| STL | 0.067 | 0.181 | 60.57 | 53.91 | 0.502 | 4.53 | 58.8 | 64.2 | 63.8 | 66.2 | 0.072 | | |
| LS | 0.106 | 0.325 | **73.57** | 89.67 | 5.19 | 14.06 | 143.4 | 144.2 | 144.6 | 140.3 | 0.128 | 6.8 | 177.6 |
| SI | 0.309 | 0.345 | 149.8 | 135.7 | **1.00** | **4.50** | **55.3** | **55.75** | **55.82** | **55.27** | 0.112 | 4.0 | 77.8 |
| RLW | 0.113 | 0.340 | 76.95 | 92.76 | 5.86 | 15.46 | 156.3 | 157.1 | 157.6 | 153.0 | 0.137 | 8.2 | 203.8 |
| DWA | 0.107 | 0.325 | 74.06 | 90.61 | 5.09 | 13.99 | 142.3 | 143.0 | 143.4 | 139.3 | 0.125 | 6.4 | 175.3 |
| UW | 0.386 | 0.425 | 166.2 | 155.8 | 1.06 | 4.99 | 66.4 | 66.78 | 66.80 | 66.24 | 0.122 | 5.3 | 108.0 |
| MGDA | 0.217 | 0.368 | 126.8 | 104.6 | 3.22 | 5.69 | 88.37 | 89.4 | 89.32 | 88.01 | 0.120 | 5.9 | 120.5 |
| PCGrad | 0.106 | 0.293 | 75.85 | 88.33 | 3.94 | 9.15 | 116.36 | 116.8 | 117.2 | 114.5 | 0.110 | 5.0 | 125.7 |
| CAGrad | 0.118 | 0.321 | 83.51 | 94.81 | 3.21 | 6.93 | 113.99 | 114.3 | 114.5 | 112.3 | 0.116 | 5.7 | 112.8 |
| IMTL-G | 0.136 | 0.287 | 98.31 | 93.96 | 1.75 | 5.69 | 101.4 | 102.4 | 102.0 | 100.1 | 0.096 | 4.7 | 77.2 |
| Nash-MTL | **0.102** | **0.248** | 82.95 | **81.89** | 2.42 | 5.38 | 74.5 | 75.02 | 75.10 | 74.16 | **0.093** | **2.5** | **62.0** |

### D.2. Effect of the Number of CCP steps

In this section, we investigate the effect of varying the number of CCP steps in our efficient approximation to $G^\top G\alpha = 1/\alpha$ (presented in Section 3.2). We use the NYUv2 dataset and train Nash-MTL with CCP sequences of 1, 20, and 40 steps at each (parameters) optimization step.

We found that increasing the CCP sequence improves the approximation to the optimal $\alpha$. Using a single CCP iteration results with $G^\top G\alpha \approx 1/\alpha$ in $91.5\%$ of the optimization steps, whereas increasing the number of iterations to 20 increases the proportion of optimal solutions to $93.5\%$. However, we found the improved solution to have no significant improvement




(a) NYUv2 (b) MT10

*Figure 7. Task Weights*. Task weights obtained from Nash-MTL throughout the optimization process, for **(a)** NYUv2, and; **(b)** MT10 with weight update frequency of 100. For better visualization, each point corresponds to a moving average with window size 200.

*Table 7. QM9*. Runtime per epoch in minutes.

| | Runtime [Min.] |
|---|---|
| LS | 0.54 |
| MGDA | 7.25 |
| PCGrad | 7.47 |
| CAGrad | 6.85 |
| Nash-MTL | 6.76 |
| Nash-MTL-5 | 1.81 |
| Nash-MTL-50 | 0.69 |

in MTL performance. Figure 6 presents the test $\Delta_m$ throughout the training process.

### D.3. Modifying the CCP Objective

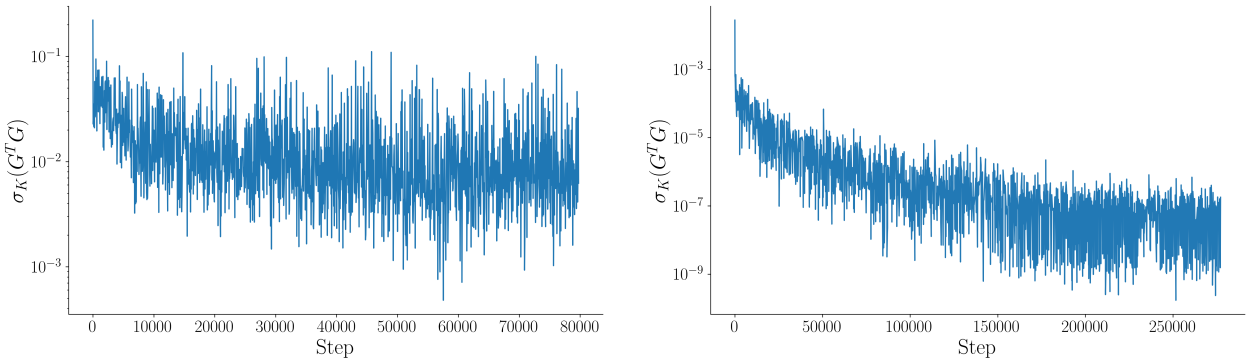
In this section we examine the effect of changing the objective of the CCP procedure described in 3.2 (Eq. 5). Here we first solve the convex optimization problem of Eq. 4 to obtain $\alpha_0$. If $G^\top G\alpha_0 \approx 1/\alpha_0$ we stop. Else we use the CCP procedure with objective $\varphi(\alpha)$, starting at $\alpha_0$ (dropping the addition $\sum_i \beta_i$ term from Eq. 5). While this objective is more natural, in practice we observe a performance degradation in terms of MTL performance. We obtain $\Delta_m = 64.4$ for the QM9 dataset (vs. 62 reported in the paper), $\Delta_m = -3.5$ (vs. $-4$) for NYUv2 and $\Delta_m = 8.8$ (vs. 6.8) for Cityscapes.

### D.4. Visualizing Task Weights

Our method, Nash-MTL, can essentially be viewed as a principled approach for producing dynamic task weights. Here we visualize these task weights throughout the training process using the NYUv2 dataset (Figure 7) and the MT10 dataset (Figure 7(b)).

### D.5. Verifying the Task Independence Assumption

Here we provide an empirical justification for our assumption in Section 3 which we state here once again: we assume that the task gradients are linearly independent for each point $\theta$ that is not Pareto stationary. To investigate whether this assumption holds in our experiments, we observe the smallest singular value of gradients Gram matrix $\sigma_K(G^\top G)$. The results are presented in Figure 8. We see that for both datasets the $\sigma_k$ decreases as the learning progresses. For the NYUv2 experiment, the smallest singular value remains fairly large throughout the entire training process. On the QM9 dataset, $\sigma_K$ decreases more significantly, to around $\sim 1e-8$.




(a) NYUv2 (b) QM9

*Figure 8. Smallest singular value of $G^\top G$ throughout the training process.*