Auxiliary Learning as an Asymmetric Bargaining Game

Aviv Shamsian [*1] **Aviv Navon** [*1 2] **Neta Glazer** [1 2] **Kenji Kawaguchi** [3]
Gal Chechik [1 4] **Ethan Fetaya** [1]

Abstract

Auxiliary learning is an effective method for enhancing the generalization capabilities of trained models, particularly when dealing with small datasets. However, this approach may present several difficulties: (i) optimizing multiple objectives can be more challenging, and (ii) how to balance the auxiliary tasks to best assist the main task is unclear. In this work, we propose a novel approach, named *AuxiNash*, for balancing tasks in auxiliary learning by formalizing the problem as a generalized bargaining game with asymmetric task bargaining power. Furthermore, we describe an efficient procedure for learning the bargaining power of tasks based on their contribution to the performance of the main task and derive theoretical guarantees for its convergence. Finally, we evaluate AuxiNash on multiple multi-task benchmarks and find that it consistently outperforms competing methods.

1. Introduction

When training deep neural networks with limited labeled data, generalization can be improved by adding auxiliary tasks. In this approach, called *Auxiliary learning* (AL), the auxiliary tasks are trained jointly with the main task, and their labels can provide a signal that is useful for the main task. AL is beneficial because auxiliary annotations are often easier to obtain than annotations for the main task. This is the case when the auxiliary tasks use self-supervision (Oliver et al., 2018; Hwang et al., 2020; Achituve et al., 2021), or their annotation process is faster. For example, learning semantic segmentation of an image may require careful and costly annotation, but can be improved if learned jointly with a depth prediction task, whose annotations can be obtained at scale (Standley et al., 2020).

Auxiliary learning has a large potential to improve learning in the low data regime, but it gives rise to two main challenges: Defining the joint optimization problem and performing the optimization efficiently. (1) First, given a main task at hand, it is not clear *which* auxiliary tasks would benefit the main task and how tasks should be *combined* into a joint optimization objective. For example, Standley et al. (2020) showed that depth estimation is a useful auxiliary task for semantic segmentation but not the opposite. In fact, adding semantic segmentation as an auxiliary task harmed depth estimation performance. This suggests that even close-related tasks may interfere with each other. (2) Second, training with auxiliary tasks involves optimizing multiple objectives simultaneously; While training with multiple tasks can potentially improve performance via better generalization, it often underperforms compared to single-task models.

Previous auxiliary learning research focused mainly on the first challenge: namely, weighting and combining auxiliary tasks (Lin et al., 2019). The second challenge, optimizing the main task in the presence of auxiliary tasks, has been less explored. Luckily, this problem can be viewed as a case of optimization in *Multi-Task Learning* (MTL). In MTL, there is extensive research on controlling optimization such that every task would benefit from the others. Specifically, several studies proposed algorithms to aggregate gradients from multiple tasks into a coherent update direction (Yu et al., 2020; Liu et al., 2021a; Navon et al., 2022). We see large potential in bringing MTL optimization ideas into auxiliary learning to address the optimization challenge.

Here we propose a novel approach named AuxiNash that takes inspiration from recent advances in MTL optimization as a cooperative bargaining game (Nash-MTL, Navon et al., 2022). The idea is to view a gradient update as a shared resource, view each task as a player in a game, and have players compete over making the joint gradient similar to their own task gradient. In Nash-MTL, tasks play a symmetric role, since no task is particularly favorable. This leads to a bargaining solution that is proportionally fair across tasks. In contrast, task symmetry no longer holds in aux-

[*]Equal contribution [1]Bar-Ilan University, Ramat Gan, Israel [2]Aiola, Herzliya, Israel [3]National University of Singapore [4]Nvidia, Tel-Aviv, Israel. Correspondence to: Aviv Shamsian <aviv.shamsian@live.biu.ac.il>, Aviv Navon <aviv.navon@biu.ac.il>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

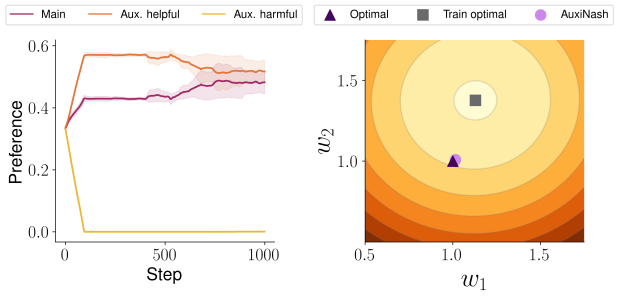

Figure 1. Illustrative example: A regression problem in $\mathbb{R}^2$ with two auxiliary tasks, one helpful and one harmful. AuxiNash succeeds in using the helpful auxiliary task and disregards the harmful one, as demonstrated by how it learns to weigh different tasks: The left panel shows the preference vector p during optimization. As a result, AuxiNash converges to a solution with large proximity to the optimal solution, far superior to that obtained from optimizing with the main tasks alone (right panel, darker colors indicate larger loss values). See Section 6.1 for further details.

iliary learning, where there is a clear distinction between the primary task and the auxiliary ones. As such, we propose to view auxiliary learning as an *asymmetric bargaining game*. Specifically, we consider gradient aggregation as a cooperative bargaining game where each player represents a task with varying bargaining power. We formulate gradient update using asymmetric Nash bargaining solution which takes into account varying task preferences. By generalizing Nash-MTL to asymmetric games with AuxiNash, we can efficiently direct optimization solution towards various areas of the Pareto front.

Determining the task preferences that result in optimal performance is a challenging problem for several reasons. First, the relationship between tasks can change during the optimization process, making it difficult to know in advance what preferences to use. This means that the process of preference tuning needs to be automated during training. Second, using a grid search to find the optimal preferences can be computationally expensive, and the complexity of the search increases exponentially with the number of tasks. To overcome these limitations, we propose a method for efficiently differentiating through the optimization process and using this information to automatically optimize task preferences during training. This can improve the performance of the primary task, and make it more efficient to find the optimal preferences.

We theoretically analyze the convergence of AuxiNash and show that even if the preference changes during the optimization process we are still guaranteed to converge to a Pareto stationary point. Finally, we show empirically on several benchmarks that AuxiNash achieves superior results to previous auxiliary learning and multi-task learning approaches.

Contributions: This paper makes the following contributions: (1) We introduce AuxiNash - a novel approach for auxiliary learning based on principles from asymmetric bargaining games. (2) We describe an efficient method to dynamically learn task preferences during training. (3) We theoretically show that AuxiNash is guaranteed to converge to a Pareto stationary point. (4) We conduct extensive experiments to demonstrate the superiority of AuxiNash against multiple baselines from MTL and auxiliary learning.

2. Related Work

Auxiliary Learning. Learning with limited amount of training data is challenging since deep learning models tend to overfit to the training data and as a result can generalize poorly (Ying, 2019). One approach to overcome this limitation is using auxiliary learning (Chen et al., 2022; Kung et al., 2021). Auxiliary learning aims to improve the model performance on primary task by utilizing the information of related auxiliary tasks (Dery et al., 2022; Chen et al.). Most auxiliary learning approaches use a linear combination of the main and auxiliary losses to form a unified loss (Zhai et al., 2019; Wen et al., 2020). Fine-tuning the weight of each task loss may be challenging as the search space of the grid search grows exponentially with the number of tasks. To find the beneficial auxiliary tasks, recent studies utilized the auxiliary task gradients and measure their similarity with the main task gradients (Lin et al., 2019; Du et al., 2018; Shi et al., 2020). Navon et al. (2021) proposed to learn a non-linear network that combines all losses into a single coherent objective function.

Multi-task Learning. In multi-task learning (MTL) we aim to solve multiple tasks by sharing information between them (Caruana, 1997; Ruder, 2017), usually through joint hidden representation (Zhang et al., 2014; Dai et al., 2016; Pinto & Gupta, 2017; Liu et al., 2019b). Previous studies showed that optimizing a model using MTL helps boost performances while being computationally efficient (Sener & Koltun, 2018; Chen et al., 2018). However, MTL presents a number of optimization challenges such as: conflicting gradients (Wang et al., 2020; Yu et al., 2020), and flatten areas in the loss landscape (Schaul et al., 2019). These challenges may result with performance degradation compare with single task learning. Recent studies proposed novel architectures (Misra et al., 2016; Hashimoto et al., 2017; Liu et al., 2019b; Chen et al., 2020) to improve MTL while others focused on aggregating the gradients of the tasks such that it is agnostic to the optimized model (Liu et al., 2021b; Javaloy & Valera, 2021). Yu et al. (2020) proposed to overcome the conflicting gradients problem by subtracting normal projection of conflicted task before forming an

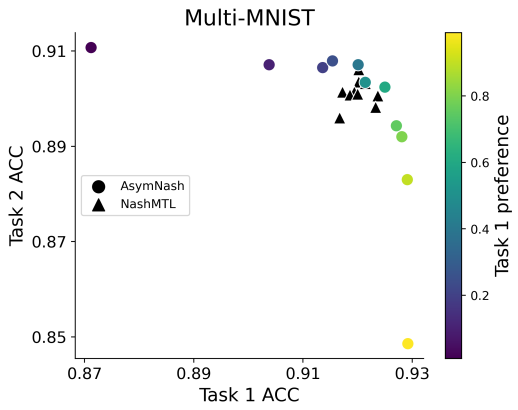

Figure 2. Task preferences: By varying the preference vector p, we show that AuxiNash can control the trafe-off between tasks. Compared with Nash-MTL, AuxiNash achieves a wider range of diverse solutions, an important property for auxiliary learning. See Section 6.2 for further details.

update direction. Most gradient based methods aim to minimize the average loss function. Liu et al. (2021a) suggested an approach that will decrease every task loss in addition to the average loss function. The closest work to our approach is Nash-MTL (Navon et al., 2022). The authors proposed a principled approach to dynamically weight the losses of different tasks by incorporating concepts from game theory.

Bi-level Optimization. Bi-Level Optimization (BLO) consists of two nested optimization problems (Liao et al., 2018; Liu et al., 2021c; Vicol et al., 2022). The outer optimization problem is commonly referred to as the upper-level problem, while the inner optimization problem is referred to as the lower-level problem (Sinha et al., 2017). BLO is widely used in a variety of deep learning applications, spanning hyper-parameter optimization (Foo et al., 2007; MacKay et al., 2019), meta learning (Franceschi et al., 2018), reinforcement learning (Zhang et al., 2020; Yang et al., 2019), and multi-task learning (Liu et al., 2022; Navon et al., 2021). A common practice to derive the gradients of the upper-level task is using the implicit function theorem (IFT). However, applying IFT involves the calculation of an inverse-Hessian vector product which is infeasible for large deep learning models. Therefore, recent studies proposed diverse approaches to approximate the inverse-Hessian vector product. Luketina et al. (2016) proposed approximating the Hessian with the identity matrix, where other line of works used conjugate gradient (CG) to approximate the product (Foo et al., 2007; Pedregosa, 2016; Rajeswaran et al., 2019). We use a truncated Neumann series and efficient vector-Jacobian products, as it was empirically shown to be more stable than CG (Liao et al., 2018; Lorraine et al., 2020; Raghu et al., 2020).

3. Background

3.1. Nash Bargaining Solution

We will first give a quick introduction to cooperative bargaining games. In a bargaining game, a set of K players jointly decide on an agreed-upon point in the set A of all agreement points. If failing to reach an agreement, the game default to the disagreement point D. Each player $i \in [K] := \{1, ..., K\}$ is equipped with a utility function $u_i : A \cup \{D\} \to \mathbb{R}$, which they wish to maximize. Intuitively, each player has a different objective, and each tries to only maximize their own personal utility. However, we generally assume that there are points in the agreement set that are mutually beneficial to all players, compared to the disagreement point, and as such the players are incentivized to cooperate. The main question is on which point in the agreement set will they decide upon.

Denote by $U = \{(u_1(x), ..., u_K(x)) : x \in A\} \subset \mathbb{R}^K$ the set of the utilities of all possible agreement points and $d = (u_1(D), ..., u_K(D))$. The set U is assumed to be convex and compact. Furthermore, we assume that there exists a point $u \in U$ for which $\forall i : u_i > d_i$. Nash (1953) showed that under these assumptions, there exists a unique solution to the bargaining game, which satisfies the following properties: Pareto optimality, symmetry, independence of irrelevant alternatives, and invariant to affine transformations (see Szép & Forgó, 1985, and the supplementary material for more details). This unique solution, referred to as the Nash Bargaining solution (NBS), is given by

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

As shown in Navon et al. (2022), NBS properties are suitable for the multi-task learning setup, even if the invariance to affine transformations implies that gradient norms are ignored. The symmetry assumption, however, implies that each player is interchangeable which is not the case for auxiliary learning. Naturally, our main concern is the main task and the auxiliaries are there to support it, not compete with it. Thus, we wish to discard the symmetry assumption for the auxiliary learning setup.

3.2. Generalized Bargaining Game

Kalai (1977) generalized the NBS to the asymmetric case. First, define a preference vector to control the relative trade-off between tasks $p \in \mathbb{R}^K$ with $p_i > 0$ and $\sum_i p_i = 1$ (see Figure 2). Similar to the symmetric case, the Generalized Nash Bargaining Solution (GNBS) maximizes a weighted

product of utilities,

$$u^* = \arg\max_{u \in U} \sum_i p_i \log(u_i - d_i) \quad (2)$$
$$s.t.\ \forall i:\ u_i > d_i \quad .$$

The symmetric case is a special case of GNBS with uniform preferences $p_i = 1/K,\ \forall i \in [K]$.

3.3. Bargaining Game for Multi-task learning

Recently, Navon et al. (2022) formalized multi-task learning as a bargaining game as follows. Let $\theta \in \mathbb{R}^d$ denote the parameters of a network $f(\cdot;\theta)$. At each MTL optimization step, we search for an update direction $\Delta\theta$. Define the agreement set $U = \{\Delta\theta \mid \|\Delta\theta\| \leq 1\}$ as the set of all possible update directions. The disagreement point d is defined to be equal to zero, i.e., to stay with the current parameters and terminate the optimization process. Let g_i denote the gradient of θ w.r.t. the loss of task i (for each $i \in [K]$). The utility function for task i is defined as $u_i(\Delta\theta) = g_i^\top \Delta\theta$, i.e., the directional derivative in direction $\Delta\theta$. Navon et al. (2022) assumed that the gradients $g_1, ..., g_K$ are linearly independent if θ is not Pareto stationary, and we adopt that assumption in our analysis. Under this assumption, Navon et al. (2022) show that the solution for the bargaining game at any non-Pareto stationary point θ is given by $\Delta\theta = \sum_i \alpha_i g_i$, where the weight vector α satisfies

$$G^\top G\alpha = 1/\alpha. \quad (3)$$

Here, G is the $d \times K$ matrix whose i-th column is the i-th task gradient g_i, and $1/\alpha$ is the element-wise reciprocal.

4. Generalized Bargaining Game for Auxiliary Learning

In this section, we first extend the result from Navon et al. (2022) to the asymmetric case. Next, we describe a method to learn the preference vector p.

4.1. Generalized Bargaining Solution

We prove the following claim, which generalizes the claim of Navon et al. (2022) to asymmetric games.

Claim 4.1. *Let $p \in \mathbb{R}_+^K$ with $\sum_i p_i = 1$. The solution to the generalized bargaining problem $\Delta\theta^* = \arg\max_{\Delta\theta \in U} \sum_i p_i \log(\Delta\theta^\top g_i)$ is given by (up to scaling) $\sum_i \alpha_i g_i$ at any non-Pareto stationary point θ, where $\alpha \in \mathbb{R}_+^K$ is the solution to $G^\top G\alpha = p/\alpha$ where p/α is the element-wise reciprocal.*

Proof. We define $F(\Delta\theta) = \sum_i p_i \log(\Delta\theta^\top g_i)$ and have $\nabla F = \sum_{i=1}^K \frac{p_i}{\Delta\theta^T g_i} g_i$. Note that for all $\Delta\theta$ with $u_i(\Delta\theta) > 0$ for all $i \in [K]$ the utilities are monotonically increasing in $\|\Delta\theta\|$, hence the optimal solution lies on the boundary of U, and $\nabla F|_{\Delta\theta^*}$ is parallel to $\Delta\theta^*$. This implies that $\sum_{i=1}^K \frac{p_i}{\Delta\theta^\top g_i} g_i = \lambda\Delta\theta$ for some $\lambda > 0$. From the linear independence assumption, we have for the optimal solution $\Delta\theta = \sum_i \alpha_i g_i$, thus $\forall i, \Delta\theta^\top g_i = \frac{p_i}{\lambda\alpha_i}$. Setting $\lambda = 1$ (as we ignore scale), the solution to the bargaining game is reduced to finding $\alpha \in \mathbb{R}_+^K$ for which $\forall i,\ \Delta\theta^\top g_i = \sum_j \alpha_j g_j^\top g_j = p_i/\alpha_i$. Equivalently, the optimal solution is given by $\alpha \in \mathbb{R}_+^K$ such that $G^\top G\alpha = p/\alpha$ where p/α is the element-wise reciprocal. □

Given a preference vector p, we solve $G^\top G\alpha = p/\alpha$ by expressing it as the solution to an optimization problem. We first solve a convex relaxation which we follow by a concave-convex procedure (CCP) (Yuille & Rangarajan, 2003; Lipp & Boyd, 2016), similar to Navon et al. (2022) for solving $G^\top G\alpha = p/\alpha$ w.r.t. α. See Appendix C for full details.

4.2. Optimizing the Preference Vector

The derivation in the previous section allows us to learn using a known preference vector p. Unfortunately, in most cases, the preference vector is not known in advance. One simple solution is to treat the preferences p_i as hyperparameters and set them via grid search. However, this approach has two significant limitations. First, as the number of hyperparameters increases, grid search becomes computationally expensive as it scales exponentially. Second, it is possible that the optimal preference vector would vary during optimization, hence using a fixed p would be sub-optimal.

To address these issues, we develop an approach for dynamically learning the task preference vector during training. This reduces the number of hyperparamters the user needs to tune to one (the preference update rate) and dynamically adjusts the preference to improve generalization. We do this by formulating the problem as bi-level optimization, which we discuss next.

Let $\mathcal{L}_T$ denote the training loss and $\mathcal{L}_V$ denote the *generalization* loss, given by the loss of the main task on unseen data, i.e., $\mathcal{L}_V = \ell_{main}^{val}$. In the auxiliary learning setup, we wish to optimize p such that a network $f(\cdot;\theta)$ optimized with $\mathcal{L}_T(\cdot;p)$ would minimize $\mathcal{L}_V$. Formally,

$$p^* = \arg\min_p \mathcal{L}_V(\theta^*(p)), \ \text{ s.t. } \theta^*(p) = \arg\min_\theta \mathcal{L}_T(\theta, p).$$

Using the chain rule to get the derivative of the outer problem, we get

$$\frac{\partial \mathcal{L}_V(p, \theta^*(p))}{\partial p} = \underbrace{\frac{\partial \mathcal{L}_V}{\partial p}}_{=0} + \frac{\partial \mathcal{L}_V}{\partial \theta}\frac{\partial \theta^*}{\partial p} = \frac{\partial \mathcal{L}_V}{\partial \theta}\frac{\partial \theta^*}{\partial \alpha(p)}\frac{\partial \alpha(p)}{\partial p}.$$

As we can compute $\frac{\partial \mathcal{L}_V}{\partial \theta}$ by simple backpropagation, the main challenge is to compute $\frac{\partial \theta^*}{\partial \alpha(p)}$ and $\frac{\partial \alpha(p)}{\partial p}$.

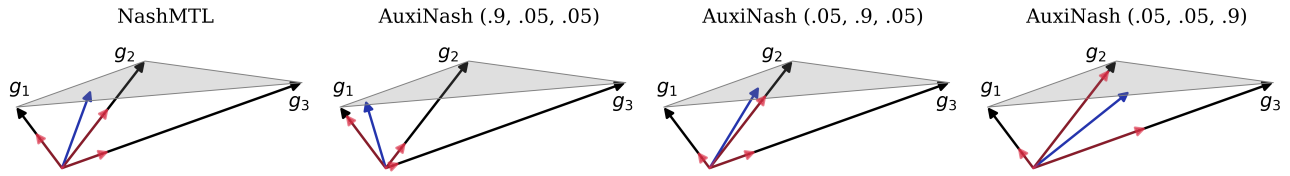

Figure 3. Visualization of the update direction: We show the update direction (blue) obtained by AuxiNash on three gradients in $\mathbb{R}^3$. We rescaled the update directions for better visibility, showing only the direction. We further show the size of the projection (red) of the update to each gradient direction (**black**). By varying the preference vector, we observe the change in the obtained update direction. Importantly, we note that the effect on the update direction is non-trivial, as p only affects the update implicitly through the bargaining solution α.

To compute $\frac{\partial \theta^*}{\partial \alpha(p)}$ we can (indirectly) differentiate through the optimization process using the implicit function theorem (IFT) (Liao et al., 2018; Lorraine et al., 2020; Navon et al., 2021):

$$\frac{\partial \theta^*}{\partial \alpha(p)} = -\left[\frac{\partial^2 \mathcal{L}_T}{\partial \theta \partial \theta^\top}\right]^{-1} \frac{\partial^2 \mathcal{L}_T}{\partial \theta \partial \alpha(p)^\top}. \tag{4}$$

Since computing the inverse Hessian directly is intractable, we use the algorithm proposed by Lorraine et al. (2020) to efficiently estimate the inverse-Hessian vector product. This approach uses the Neumann approximation with an efficient vector-Jacobian product. Thus, we can efficiently approximate the first term, $\frac{\partial \mathcal{L}_V}{\partial \theta}\frac{\partial \theta^*}{\partial \alpha(p)}$. We note that in practice, as in customary, we do not optimize till convergence but perform a few gradient updates from the previous value. For further details, see Vicol et al. (2022) that recently examined how this affects the bi-level optimization process.

For the second term, $\frac{\partial \alpha(p)}{\partial p}$, we derive a simple analytical expression using the IFT in the following proposition:

Proposition 4.2. *For any (p, α) satisfying $G^\top G\alpha = p/\alpha$, there exists an open set $U \subseteq \mathbb{R}^K$ containing p such that there exists a continuously differentiable function $\hat{\alpha} : U \to \mathbb{R}^K$ satisfying all of the following properties: (1) $\hat{\alpha}(p) = \alpha$, (2) $G^\top G\hat{\alpha}(\bar{p}) = \bar{p}/\hat{\alpha}(\bar{p})$ for all $\bar{p} \in U$, and (3)*

$$\frac{\partial \hat{\alpha}(p)}{\partial p} = \left[G^\top G + \Lambda_0\right]^{-1} \Lambda_1. \tag{5}$$

Here $\Lambda_0, \Lambda_1 \in \mathbb{R}^{K\times K}$ are the diagonal matrices defined by $(\Lambda_0)_{ii} = p_i/\alpha_i^2 \in \mathbb{R}$ and $(\Lambda_1)_{ii} = 1/\alpha_i \in \mathbb{R}$ for $i \in [K]$.

We refer the readers to Appendix B for the proof.

Putting everything together, we obtain the following efficient approximation,

$$\frac{\partial \mathcal{L}_V(p, \theta^*(p))}{\partial p} = -\frac{\partial \mathcal{L}_V}{\partial \theta}\left[\frac{\partial^2 \mathcal{L}_T}{\partial \theta \partial \theta^\top}\right]^{-1} \frac{\partial^2 \mathcal{L}_T}{\partial \theta \partial \alpha(p)^\top} \times \left[G^\top G + \Lambda_0\right]^{-1} \Lambda_1. \tag{6}$$

Algorithm 1 AuxiNash

Input: θ – initial parameter vector, p – initial preference vector, $\{\ell_i\}_{i=1}^K$ – differentiable loss functions, η, η_p – learning rates
for $T = 1, ..., N$ **do**
  for $t = 1, ..., N_p$ **do**
    Compute task gradients $g_i = \nabla_\theta \ell_i$
    Set G the matrix with columns g_i
    Solve for α: $G^\top G\alpha = p/\alpha$
    Update the parameters $\theta \leftarrow \theta - \eta G\alpha$
  end for
  Evaluate $\nabla_p \mathcal{L}_V$ using Eq. 6
  Update $p \leftarrow p - \eta_p \nabla_p \mathcal{L}_V$
end for
Return: θ.

We note that this approximation can be computed in a relatively efficient manner, with the cost of only several back-propagation operations to estimate the vector-Jacobian product (we use 3 in our experiments). We also note that the matrix $\left(G^\top G + \Lambda_0\right)$ that we invert is of size $K \times K$, where K is the number of tasks that is generally relatively small.

In practice, we use a separate batch from the training set to estimate the generalization loss $\mathcal{L}_V$. We further discuss this design choice and provide an empirical evaluation in Section 6.3. During the optimization process, we alternate between optimizing θ and optimizing p. Specifically, we update p once every N_p optimization steps over θ. We set $N_p = 25$ in our experiments. The AuxiNash algorithm is summarized in Alg. 1.

5. Analysis

We analyze the convergence properties of our proposed method in nonconvex optimization. We adopt the following three assumptions from Navon et al. (2022):

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.

Table 1. 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° ↑			$\mathbf{\Delta}\%$ ↓
	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	38.43	64.36	0.5472	0.2184	29.57	25.42	20.50	44.85	58.20	8.69
PCGrad	39.25	64.95	0.5389	0.2141	28.66	24.26	21.99	47.00	60.31	5.66
CAGrad	39.25	65.15	0.5385	0.2155	26.11	20.95	26.96	53.66	66.37	-1.46
Nash-MTL	39.83	66.00	0.5235	0.2075	25.32	19.87	28.86	55.87	68.27	-4.76
GCS	38.96	64.35	0.5769	0.2293	29.57	25.53	20.64	44.68	57.99	9.54
OL-AUX	40.51	65.49	0.6652	0.2614	**24.65**	**18.72**	**30.92**	**58.37**	**70.12**	-2.88
AuxiLearn	38.63	64.20	0.5415	0.2173	29.98	25.29	20.03	43.94	57.17	9.15
AuxiNash (ours)	**40.79**	**66.79**	**0.5092**	**0.2042**	24.90	19.31	29.83	57.07	69.27	$\mathbf{-6.80}$

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,

$$\|\nabla \ell_i(x) - \nabla \ell_i(y)\| \leq L\|x - y\|. \tag{7}$$

Since even single-task non-convex optimization might only admits convergence to a stationary point, the following theorem proves convergence to a Pareto stationary point when both θ and p are optimized concurrently:

Theorem 5.4. *Suppose that Assumptions 5.1, 5.2, and 5.3 hold. 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 weighted Nash bargaining solution $(G^{(t)})^\top G^{(t)}\alpha^{(t)} = p^{(t)}/\alpha^{(t)}$ where $p^{(t)}$ can be any discrete distribution. Set $\mu^{(t)} = \frac{1}{K}\sum_{i=1}^{K} p_i^{(t)}(L\alpha_i^{(t)})^{-1}$. The sequence $\{\theta^{(t)}\}_{t=1}^{\infty}$ has a subsequence that converges to a Pareto stationary point θ^*. Moreover all the loss functions $(\ell_1(\theta^{(t)}), ..., \ell_K(\theta^{(t)}))$ converge to $(\ell_1(\theta^*), ..., \ell_K(\theta^*))$.*

See full proof in Appendix B.

6. Experiments

In this section, we compare AuxiNash with different approaches from multi-task and auxiliary learning. We use variety of datasets and learning setups to demonstrate the superiority of AuxiNash. To encourage future research and reproducibility, we make our source code publicly available https://github.com/AvivSham/auxinash. Additional experimental results and details are provided in Appendix D.

Baselines. We compare AuxiNash with natural baselines from recent auxiliary and multi-task learning works. The compared methods includes (1) Single-task learning (STL), which trains a model using the main task only; (2) Linear scalarization (LS) that minimizes the sum of losses $\sum_k \ell_k$; (3) GCS (Du et al., 2018), an auxiliary learning approach that uses gradient similarity between primary and auxiliary tasks; (4) OL-AUX (Lin et al., 2019), an auxiliary learning approach that adaptively changes the loss weight based on the gradient inner product w.r.t the main task; (5) AuxiLearn (Navon et al., 2021), an auxiliary learning approach that dynamically learns non-linear combinations of different tasks; (6) PCGrad (Yu et al., 2020), an MTL method that removes gradient components that conflict with other tasks; (7) CAGrad (Liu et al., 2021a), an MTL method that optimizes for the average loss while explicitly controlling the minimum decrease rate across tasks; (8) Nash-MTL (Navon et al., 2022), an MTL approach that is equivalent to AuxiNash but with a fixed $p_i = 1/K$ weighting.

Evaluation We report the common evaluation metrics for each task. Since MTL methods treat each task equally, and these may vary in scale, we also report the overall relative multi-task performance $\Delta\%$. $\Delta\%$ is defined as the performance drop compared to the STL performance. Formally, $\Delta\% = \frac{1}{K}\sum_{k=1}^{K}(-1)^{\delta_k}(M_{m,k} - M_{b,k})/M_{b,k}$. We denote $M_{b,k}$ and $M_{m,k}$ as the performance of STL and the compared method on task k, respectively. $\delta_k = 0$ if a lower value is better for the metric M_k and 1 otherwise (Maninis et al., 2019). In all experiments, we report the mean value based on 3 random seeds.

It is important to note that for MTL models, we present the results of a single model trained on all tasks. For auxiliary learning methods, we trained a unique model per task, treating it as the main task and using the remaining tasks as

Table 2. Cityscapes. Test performance for three tasks: 19-class semantic segmentation, 10-class part segmentation, and disparity.

	Semantic Seg.		Part Seg.		Disparity	
	mIoU ↑	Pix Acc ↑	mIoU ↑	Pix Acc ↑	Abs Err ↓	$\Delta\%$ ↓
STL	48.64	91.01	53.60	97.62	1.108	
LS	37.66	88.63	40.92	96.98	1.105	9.84
PCGrad	39.10	89.31	41.71	97.14	1.133	9.28
CAGrad	39.45	89.04	51.95	97.54	1.098	4.66
Nash-MTL	51.14	91.59	56.99	97.87	1.066	−3.23
GCS	37.45	88.62	41.14	96.97	1.124	10.19
OL-AUX	27.63	89.34	51.12	97.52	1.397	15.16
AuxiLearn	36.18	88.24	40.51	96.95	1.141	11.3
AuxiNash	**52.52**	**91.91**	**58.53**	**97.93**	**1.027**	**−5.15**

auxiliaries.

6.1. Illustrative Example

We start with an illustrative example, showing that AuxiNash can utilize helpful auxiliaries while ignoring harmful ones.

We adopt a similar problem setup as in Navon et al. (2021) and consider a regression problem with parameters $W^T = (w_1, w_2) \in \mathbb{R}^2$, fully shared among tasks. The optimal parameters for the main and helpful auxiliary tasks are $W^\star$, while the optimal parameters for the harmful auxiliary are $\tilde{W} \neq W^\star$. The main task is sampled from a Normal distribution $N(W^{\star T}x, \sigma_{\text{main}})$, with $\sigma_{\text{main}} > \sigma_{\text{h}}$ where σ_{h} denotes the standard deviation for the noise of the helpful auxiliary.

The change in the task preference throughout the optimization process is depicted in the left panel of Figure 1. AuxiNash identify the helpful tasks and fully ignore the harmful ones. In addition, Figure 1 right panel presents the main task's loss landscape, along with the optimal solution ($W^\star$, marked ▲), the optimal training set solution of the main task alone (■) and the solution obtained by AuxiNash (marked ●). While using the training data alone with no auxiliary information yields a solution that generalizes poorly, AuxiNash converges to a solution with large proximity to the optimal solution $W^\star$,

6.2. Controlling Task Preference

In this section, we wish to better understand the relationship between the preference p and the obtained solution. We note the preference vector only implicitly affects the optimization solution through the bargaining solution α.

Here, we show that controlling the preference vector can be used to steer the optimization outcome to different parts of the Pareto front, compared to the NashMTL baseline. We consider MTL setup with 2 tasks and use the Multi-MNIST (Sabour et al., 2017) dataset. In Multi-MNIST two images from the original MNIST dataset are merged into one by placing one at the top-left corner and the other at the bottom-right corner. The tasks are defined as image classification of the merged images. We run AuxiNash 11 times with varying preference vector values p and fix it throughout the training. For both tasks we report the classification accuracy. For Nash-MTL we run the experiments with different seed values. For both methods we train a variant of LeNet model for 50 epochs with Adam optimizer and $1e-4$ as the learning rate.

Figure 2 shows the results. AuxiNash reaches a diverse set of solutions across the Pareto front while Nash-MTL solutions are all relatively similar due to its symmetry property.

6.3. Analyzing the Effect of Auxiliary Set

Table 3. The effect of auxiliary set: We report the mean IoU, along with the % change w.r.t STL performance.

	Cityscapes		NYUv2	
	mIoU ↑	Change % ↑	mIoU ↑	Change % ↑
STL	48.64		38.30	
STL Partial	45.97	−5.48	36.54	−4.59
AuxiNash	52.52	7.97	40.79	6.50
AuxiNash Aux. Set	51.81	6.51	38.94	1.67

One important question is on what data to evaluate the generalization loss $\mathcal{L}_V$. It would seem intuitive that one would need a separate validation set for that since estimating $\mathcal{L}_V$ on the training data may be biased. In practice, some previous works use a held-out auxiliary set (Navon et al., 2021), while others use a separate batch from the training set (Liu et al., 2019a; 2022). While using an auxiliary set might be more intuitive, it requires reducing the available amount of training data which can be detrimental in the low-data regime we are interested in.

We empirically evaluate this using the NYUv2 (Silberman et al., 2012) and Cityscapes (Cordts et al., 2016) datasets. See Section 6.4 for more details. We choose semantic segmentation as the main task for both datasets. We compare the following methods (i) *STL*: single task learning using the main task only, (ii) *STL Partial*: STL using only 90% of the training data, (iii) *AuxiNash*: our proposed method where we optimize the preference vector using the entire training set, (iv) *AuxiNash Aux. Set*: our proposed method, where we optimize the preference vector using 10% of the data, allocated from the training set.

We report the mean-IoU metric (higher is better) along with the relative change from the performance of the STL method. The results suggest that the drawback of sacrificing some of the training data overweighs the benefit of using an auxiliary set. This result aligns with the observation in Liu et al. (2022).

6.4. Scene Understanding

We follow the setup from Liu et al. (2019b; 2022); Navon et al. (2022) and evaluate AuxiNash on the NYUv2 and Cityscapes datasets (Silberman et al., 2012; Cordts et al., 2016). The indoor scene NYUv2 dataset (Silberman et al., 2012) contains 3 tasks: 13 classes semantic segmentation, depth estimation, and surface normal prediction. The dataset consists of 1449 RGBD images captured from diverse indoor scenes.

We also use the Cityscapes dataset (Cordts et al., 2016) with 3 tasks (similar to Liu et al. (2022)): 19-class semantic segmentation, disparity (inverse depth) estimation, and 10-class part segmentation (de Geus et al., 2021). To speed up the training phase, all images and label maps were resized to 128×256.

For all methods, we train SegNet (Badrinarayanan et al., 2017), a fully convolutional model based on VGG16 architecture. We follow the training and evaluation procedure from Liu et al. (2019b; 2022); Navon et al. (2022) and train the network for 200 epochs with Adam optimizer (Kingma & Ba, 2015). We set the learning rate to $1e-4$ and halved it after 100 epochs.

The results are presented in Table 1 and Table 2. Observing the results, we can see our approach AuxiNash outperforms other approaches by a significant margin. It is also important to note that several methods achieve a positive $\%\Delta$ score, meaning they performed worse than simply ignoring the auxiliary tasks and training on the main task alone. We believe this is due to the difficulties presented by optimizing with multiple objectives.

6.5. Semi-supervised Learning with SSL Auxiliaries

Table 4. CIFAR10-SSL. Test performance for classification with a varying number of labeled data. Values are averages over 3 random seeds.

	CIFAR10-SSL-5K	CIFAR10-SSL-10K
STL	79.31 ± 0.31	83.75 ± 0.18
LS	83.17 ± 0.54	86.16 ± 0.39
PCGrad	82.71 ± 0.16	86.17 ± 0.34
CAGrad	85.89 ± 0.63	87.82 ± 0.28
Nash-MTL	$\mathbf{86.69 \pm 0.14}$	$\mathbf{88.68 \pm 0.14}$
GCS	83.09 ± 0.34	86.47 ± 0.56
OL-AUX	81.44 ± 1.06	85.49 ± 0.73
AuxiLearn	82.83 ± 0.57	85.52 ± 0.57
AuxiNash (ours)	$\mathbf{87.01 \pm 0.52}$	$\mathbf{88.81 \pm 0.34}$

In semi-supervised learning one generally trains a model with a small amount of labeled data, while utilizing self-supervised tasks as auxiliaries to be optimized using unlabeled training data.

We follow the setup from Shi et al. (2020) and evaluate AuxiNash on a Self-supervised Semi-supervised Learning setting (Zhai et al., 2019). We use CIFAR-10 dataset to form 3 tasks. We set the supervised classification as the main task along with two self-supervised learning (SSL) tasks used as auxiliaries: (i) Rotation (ii) Exempler-MT. In Rotation, we randomly rotate each image by $[0°, 90°, 180°, 270°]$ and optimize the network to predict the angle. In Exempler-MT we apply a combination of three transformations: horizontal flip, gaussian noise, and cutout. Similarly to contrastive learning, the model is trained to extract invariant features by encouraging the original and augmented images to be close in their feature space. For the supervised task we randomly allocate samples from the training set. We repeat this experiment twice with $5K$ and $10K$ labeled training examples. The results are presented in Table 4. AuxiNash significantly outperforms most baselines.

6.6. Audio Classification

Table 5. Speech Commands. Test accuracy for speech classification, for models trained with 1000 and 500 training examples.

	SC-500	SC-1000
STL	95.8 ± 0.1	96.4 ± 0.1
LS	95.7 ± 0.2	96.7 ± 0.1
PCGrad	95.7 ± 0.1	96.7 ± 0.1
CAGrad	95.7 ± 0.2	95.7 ± 0.1
Nash-MTL	95.7 ± 0.2	96.6 ± 0.3
GCS	96.3 ± 0.1	96.9 ± 0.1
OL-AUX	96.2 ± 0.2	96.9 ± 0.1
AuxiLearn	96.0 ± 0.1	97.0 ± 0.1
AuxiNash (ours)	$\mathbf{96.4 \pm 0.1}$	$\mathbf{97.2 \pm 0.1}$

We evaluate AuxiNash on the speech commands (SC) dataset (Warden, 2018), which consists of $\sim$ 50K speech samples of specific keywords. The data contains 30 different keywords, and each speech sample is one second long. We use a subset of the SC containing audio samples for only the 10 numbering keywords (zero to nine). As a pre-processing step, we use a short-time Fourier transform (STFT) to extract a spectrogram for each example, which we then fed to a convolutional neural network (CNN). We evaluate AuxiNash on 10 one-vs-all binary classification tasks. We repeat the experiment with a training set of sizes 500 and 1000. The results are presented in Table 5.

6.7. Learned Preferences

To better understand the learned preference and its dynamics, we observe the change in preference vectors throughout

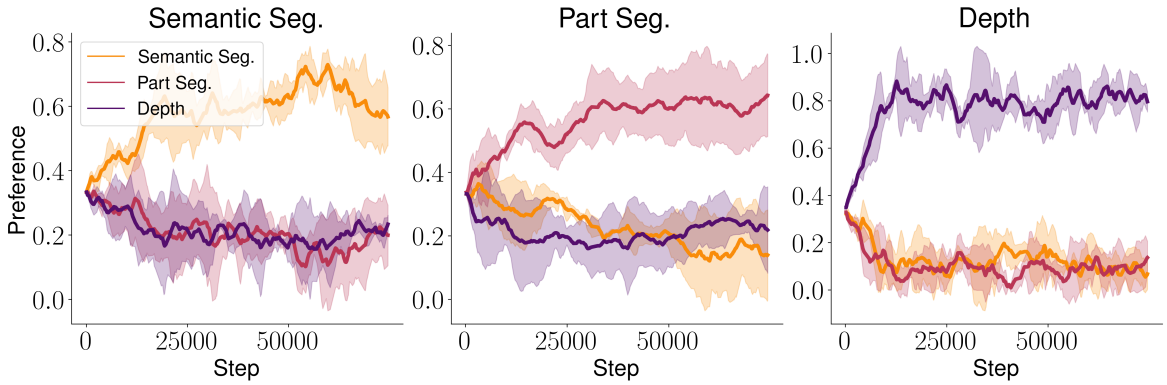

Figure 4. Learned task preference for Cityscapes dataset. The title of each panel indicates the main task.

the optimization process. We use the scene understanding experiment of Section 6.4, specifically the Cityscapes dataset. The learned preferences are presented in Figure 4. The title of each panel indicates the main task. Here, AuxiNash learns preferences that are aligned with our intuition, i.e., the main task's preference is the largest.

7. Limitations

Auxiliary learning methods, while valuable in various domains, are not without their limitations. One potential limitation is designing effective auxiliary tasks that truly capture the underlying structure and knowledge of the main task can be challenging. Another limitation is the potential for negative transfer. If the auxiliary task is not sufficiently aligned with the main task, the learned representations may not be beneficial for improving performance. Additionally, auxiliary learning methods may suffer from scalability issues when dealing with large-scale datasets or complex problems. Finally, auxiliary learning methods might be sensitive to the choice of hyperparameters, architecture, or training procedures, making them less robust and generalizable across different tasks and datasets. Overcoming these limitations remains an ongoing research endeavor to unlock the full potential of auxiliary learning methods.

8. Conclusion and Future Work

In this work, we formulate auxiliary learning as an asymmetric bargaining game and use game-theoretical tools to derive an efficient algorithm. We adapt and generalize recent advancements in multi-task learning to auxiliary learning and show how they can be automatically tuned to get a significant improvement in performance.

We evaluated AuxiNash on multiple datasets with different learning setups and show that it outperforms previous approaches by a significant margin. Across all experiments, it is noticeable that MTL methods perform better than auxiliary learning ones although the former treat equally the primary task and the auxiliary tasks. We suspect that this is caused by conflicting gradients and by the fact that gradient norms may vary significantly across tasks. These results emphasize the connection between auxiliary learning and multi-task optimization. In many examples, the benefit of the auxiliary task was diminished or even completely negated by poor optimization. Thus, we suggest that auxiliary learning research should be closely aligned with MTL optimization research to effectively utilize auxiliary tasks.

9. Acknowledgements

This study was funded by a grant to GC from the Israel Science Foundation (ISF 737/2018), and by an equipment grant to GC and Bar-Ilan University from the Israel Science Foundation (ISF 2332/18). AN and AS are supported by a grant from the Israeli higher-council of education, through the Bar-Ilan data science institute (BIU DSI).

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.

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

Chen, H., Wang, X., Liu, Y., Zhou, Y., Guan, C., and Zhu, W. Module-aware optimization for auxiliary learning. In *Advances in Neural Information Processing Systems*.

Chen, H., Wang, X., Guan, C., Liu, Y., and Zhu, W. Aux-

iliary learning with joint task and data scheduling. In *International Conference on Machine Learning*, pp. 3634–3647. PMLR, 2022.

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.

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.

de Geus, D., Meletis, P., Lu, C., Wen, X., and Dubbelman, G. Part-aware panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5485–5494, 2021.

Dery, L. M., Michel, P., Khodak, M., Neubig, G., and Talwalkar, A. Aang: Automating auxiliary learning. *arXiv preprint arXiv:2205.14082*, 2022.

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.

Foo, C.-s., Ng, A., et al. Efficient multiple hyperparameter learning for log-linear models. *Advances in neural information processing systems*, 20, 2007.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pp. 1568–1577. PMLR, 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.

Hwang, D., Park, J., Kwon, S., Kim, K., Ha, J., and Kim, H. J. Self-supervised auxiliary learning with meta-paths for heterogeneous graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Javaloy, A. and Valera, I. Rotograd: Gradient homogenization in multitask learning. *arXiv preprint arXiv:2103.02631*, 2021.

Kalai, E. Nonsymmetric nash solutions and replications of 2-person bargaining. *International Journal of Game Theory*, 6(3):129–133, 1977.

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

Kung, P.-N., Yin, S.-S., Chen, Y.-C., Yang, T.-H., and Chen, Y.-N. Efficient multi-task auxiliary learning: Selecting auxiliary data by feature similarity. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 416–428, 2021.

Liao, R., Xiong, Y., Fetaya, E., Zhang, L., Yoon, K., Pitkow, X., Urtasun, R., and Zemel, R. Reviving and improving recurrent back-propagation. In *International Conference on Machine Learning*, pp. 3082–3091. PMLR, 2018.

Lin, X., Baweja, H., Kantor, G., and Held, D. Adaptive auxiliary task weighting for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

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., Chen, Y., Yang, W., Liao, Q., and Zhang, W. Towards impartial multi-task learning. ICLR, 2021b.

Liu, R., Gao, J., Zhang, J., Meng, D., and Lin, Z. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021c.

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.

Liu, S., James, S., Davison, A. J., and Johns, E. Autolambda: Disentangling dynamic task relationships. *arXiv preprint arXiv:2202.03091*, 2022.

Lorraine, J., Vicol, P., and Duvenaud, D. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pp. 1540–1552. PMLR, 2020.

Luketina, J., Berglund, M., Greff, K., and Raiko, T. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International conference on machine learning*, pp. 2952–2960. PMLR, 2016.

MacKay, M., Vicol, P., Lorraine, J., Duvenaud, D., and Grosse, R. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. *arXiv preprint arXiv:1903.03088*, 2019.

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.

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)*, 2021.

Navon, A., Shamsian, A., Achituve, I., Maron, H., Kawaguchi, K., Chechik, G., and Fetaya, E. Multi-task learning as a bargaining game. In *International Conference on Machine Learning*, 2022.

Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., and Goodfellow, I. Realistic evaluation of deep semi-supervised learning algorithms. *Advances in neural information processing systems*, 31, 2018.

Pedregosa, F. Hyperparameter optimization with approximate gradient. In *International conference on machine learning*, pp. 737–746. PMLR, 2016.

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.

Raghu, A., Raghu, M., Kornblith, S., Duvenaud, D., and Hinton, G. Teaching with commentaries. *arXiv preprint arXiv:2011.03037*, 2020.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *Neural Information Processing Systems*, 2019.

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

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. *Advances in neural information processing systems*, 30, 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, B., Hoffman, J., Saenko, K., Darrell, T., and Xu, H. Auxiliary task reweighting for minimum-data learning. *Advances in Neural Information Processing Systems*, 33: 7148–7160, 2020.

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.

Sinha, A., Malo, P., and Deb, K. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

Standley, T., Zamir, A., 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.

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

Vicol, P., Lorraine, J. P., Pedregosa, F., Duvenaud, D., and Grosse, R. B. On implicit bias in overparameterized bilevel optimization. In *International Conference on Machine Learning*, pp. 22234–22259. PMLR, 2022.

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.

Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Wen, H., Zhang, J., Wang, Y., Lv, F., Bao, W., Lin, Q., and Yang, K. Entire space multi-task modeling via post-click behavior decomposition for conversion rate prediction.

In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 2377–2386, 2020.

Yang, Z., Chen, Y., Hong, M., and Wang, Z. Provably global convergence of actor-critic: A case for linear quadratic regulator with ergodic cost. *Advances in neural information processing systems*, 32, 2019.

Ying, X. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, volume 1168, pp. 022022. IOP Publishing, 2019.

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*, 2020.

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

Zhai, X., Oliver, A., Kolesnikov, A., and Beyer, L. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1476–1485, 2019.

Zhang, H., Chen, W., Huang, Z., Li, M., Yang, Y., Zhang, W., and Wang, J. Bi-level actor-critic for multi-agent coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7325–7332, 2020.

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.

A. Nash Bargaining Solution Details

The set U is assumed to be convex and compact. Furthermore, we assume that there exists a point $u \in U$ for which $\forall i : u_i > d_i$. Nash (1953) showed that under these assumptions, 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 A.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.

Axiom A.2. Symmetry: The solution should be invariant to permuting the order of the players.

Axiom A.3. Independence of irrelevant alternatives (IIA): If we enlarge the of possible payoffs to $\tilde{U} \supsetneq U$, 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 A.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 $(c_1 y_1 + b_1, ..., c_k y_k + b_k)$

B. Proofs

Proof of Proposition 4.2. Define a function $F(\bar{p}, \bar{\alpha}) = G^\top G\bar{\alpha} - \bar{p}/\bar{\alpha} \in \mathbb{R}^K$ where and $\bar{p} \in \mathbb{R}_+^K$ and $\bar{\alpha} \in \mathbb{R}_+^K$ are independent variables of F with $\mathbb{R}_+$ being the set of strictly positive real numbers. Here, $\bar{p}/\bar{\alpha}$ represents the coordinate-wise operation, i.e., $\bar{p}/\bar{\alpha} \in \mathbb{R}^K$ with $(\bar{p}/\bar{\alpha})_i = \bar{p}_i/\bar{\alpha}_i \in \mathbb{R}$ for $i \in [K]$. Then, we have

$$\frac{\partial F(\bar{p}, \bar{\alpha})}{\partial \bar{\alpha}}\Big|_{(\bar{p},\bar{\alpha})=(p,\alpha)} = G^\top G - \frac{\partial(\bar{p}/\bar{\alpha})}{\partial \bar{\alpha}}\Big|_{(\bar{p},\bar{\alpha})=(p,\alpha)} = G^\top G + \Lambda_0,$$

where the last equality follows from $\frac{\partial \bar{p}/\bar{\alpha}}{\partial \bar{\alpha}} = -\Lambda_0$ since

$$\frac{\partial(\bar{p}/\bar{\alpha})_i}{\partial \bar{\alpha}_j} = \begin{cases} \frac{\partial(\bar{p}_j/\bar{\alpha}_j)}{\partial \bar{\alpha}_j} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} = \begin{cases} -\bar{p}_i/\bar{\alpha}_i^2 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Similarly,

$$\frac{\partial F(\bar{p}, \bar{\alpha})}{\partial \bar{p}}\Big|_{(\bar{p},\bar{\alpha})=(p,\alpha)} = -\frac{\partial(\bar{p}/\bar{\alpha})}{\partial \bar{p}}\Big|_{(\bar{p},\bar{\alpha})=(p,\alpha)} = -\Lambda_1,$$

since

$$\frac{\partial(\bar{p}/\bar{\alpha})_i}{\partial \bar{p}_j} = \begin{cases} \frac{\partial(\bar{p}_j/\bar{\alpha}_j)}{\partial \bar{p}_j} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1/\bar{\alpha}_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Thus, F is continuously differentiable and $\frac{\partial F(\bar{p},\bar{\alpha})}{\partial \bar{\alpha}}|_{(\bar{p},\bar{\alpha})=(p,\alpha)} = G^\top G + \Lambda_0$ is invertible since $G^\top G$ is positive semi-definite and Λ_0 is positive definite due to the condition of $p_i > 0$ for all $i \in [K]$. Therefore, the function F at (p, α) satisfies the condition of the implicit function theorem, which implies the statement of this proposition as

$$\frac{\partial \hat{\alpha}(p)}{\partial p} = -\left[\frac{\partial F(\bar{p}, \bar{\alpha})}{\partial \bar{\alpha}}\right]^{-1} \frac{\partial F(\bar{p}, \bar{\alpha})}{\partial \bar{p}}\Big|_{(\bar{p},\bar{\alpha})=(p,\alpha)} = \left[G^\top G + \Lambda_0\right]^{-1} \Lambda_1. \tag{8}$$

□

B.1. Proof of Theorem 5.4

We adopt the following implicit assumption (or the design of algorithm) from (Navon et al., 2022): if we reach a Pareto stationary solution at some step, the algorithm halts. We will also use the following Lemma.

Lemma B.1. *(Lemma A.1 of Navon et al., 2022) 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. 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. Since $\sum_{i=1}^{K} p_i^{(t)} = 1$ and $\Delta\theta^{(t)} = G^{(t)}\alpha^{(t)}$,

$$||\Delta\theta^{(t)}||^2 = (\alpha^{(t)})^\top (G^{(t)})^\top G^{(t)}\alpha^{(t)} = (\alpha^{(t)})^\top (p^{(t)}/\alpha^{(t)}) = \sum_{i=1}^{K} \alpha_i^{(t)} \cdot (p_i^{(t)}/\alpha_i^{(t)}) = 1.$$

For each loss ℓ_i for $i \in [K]$, using Lemma B.1 and $(g_i^{(t)})^\top \Delta\theta^{(t)} = (g_i^{(t)})^\top G^{(t)}\alpha^{(t)} = p_i^{(t)}/\alpha_i^{(t)}$,

$$\ell_i(\theta^{(t+1)}) \leq \ell_i(\theta^{(t)}) - \mu^{(t)} \nabla \ell_i(\theta^{(t)})^\top \Delta\theta^{(t)} + \frac{L}{2}||\mu^{(t)}\Delta\theta^{(t)}||^2 \tag{9}$$

$$= \ell_i(\theta^{(t)}) - \mu^{(t)} \frac{p_i^{(t)}}{\alpha_i^{(t)}} + \frac{(\mu^{(t)})^2 L}{2}. \tag{10}$$

We average over the above inequality over all losses and get for $\mathcal{L}(\theta) = \frac{1}{K}\sum_{i=1}^{K} \ell_i(\theta)$:

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

By rearranging, this shows that $\mathcal{L}(\theta^{(t+1)}) \leq \mathcal{L}(\theta^{(t)})$ and $\frac{L(\mu^{(t)})^2}{2} \leq \mathcal{L}(\theta^{(t)}) - \mathcal{L}(\theta^{(t+1)})$. From the first inequality, the sequence $(\mathcal{L}(\theta^{(t)}))_t$ is non-increasing. As $\mathcal{L}(\theta^{(t)}) \in \mathbb{R}$ is bounded below and $(\mathcal{L}(\theta^{(t)}))_t$ is non-increasing, the monotone convergence theorem concludes that the sequence $(\mathcal{L}(\theta^{(t)}))_t$ converges to a finite limit. Since a convergent sequence is a Cauchy sequence, $\frac{L(\mu^{(t)})^2}{2} \leq \mathcal{L}(\theta^t) - \mathcal{L}(\theta^{t+1}) \to 0$ as $t \to \infty$. This implies that $\mu^{(t)} \to 0$ as $t \to \infty$. It follows that $\frac{1}{K}\sum_{i=1}^{K} p_i^{(t)}/\alpha_i^{(t)} \to 0$ as $t \to \infty$. Since $p_i^{(t)} > 0$ and $\alpha_i^{(t)} > 0$, it implies that $\max_i p_i^{(t)} \geq 1/K$. If we define $j_t = \arg\max_i p_i^{(t)}$ we will get that $p_{j_t}^{(t)}/\alpha_{j_t}^{(t)} \to 0$ and therefore $\alpha_{j_t}^{(t)} \to \infty$. We can conclude that $||\alpha^{(t)}|| \to \infty$.

We will now show that $||p^{(t)}/\alpha^{(t)}||$ 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, $|p_i^{(t)}/\alpha_i^{(t)}| = |(g_i^{(t)})^T \Delta\theta^{(t)}| \leq ||g_i^{(t)}|| \leq M < \infty$, and so $||p^{(t)}/\alpha^{(t)}||$ is bounded. Combining these two results we have $||p^{(t)}/\alpha^{(t)}|| \geq \sigma_K((G^{(t)})^\top G^{(t)})||\alpha^{(t)}||$ 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 $p^{(t)}/\alpha^{(t)}$ 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 θ^*. 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 θ^*. This means that the gradients at θ^* are linearly dependent and therefore θ^* is Pareto stationary by assumption 5.1. Since all loss functions ℓ_i are the are differentiable, all loss functions ℓ_i are continuous. Thus, we have from the continuity of ℓ_i that $\ell_i(\theta^{(t)}) \xrightarrow{t\to\infty} \ell_i(\theta^*)$. □

C. Solving $\mathbf{G}^\top\mathbf{G}\alpha = p/\alpha$

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

$$\begin{aligned} \min_\alpha \sum_i \varphi_i(\alpha) \\ \text{s.t.} \forall i, \quad -\varphi_i(\alpha) \leq 0 \\ \alpha_i > 0 \quad . \end{aligned} \tag{12}$$

Table 6. Speech Commands. Test performance on each one of the digit keywords. The train data-set consist of 1000 samples, uniformly distributed between the 10 classes. Values are averaged over 3 random seeds.

Speech Commands - 1000											
	0	1	2	3	4	5	6	7	8	9	Mean
STL	97.0	96.7	95.8	96.3	96.4	96.3	96.7	96.6	96.0	96.6	96.4
LS	97.0	97.2	96.2	97.1	97.0	96.2	96.9	97.3	95.8	96.8	96.7
PCGrad	97.0	97.3	96.2	96.6	96.7	96.7	96.8	97.3	95.8	96.9	96.7
CAGrad	97.2	97.7	95.6	97.0	97.0	96.6	97.0	97.2	95.8	97.0	96.8
Nash-MTL	97.1	96.8	96.4	96.6	96.9	96.6	96.8	96.9	96.0	96.8	96.6
GCS	97.3	97.8	96.2	97.0	**97.4**	96.2	96.8	97.3	96.2	97.1	96.9
OL-AUX	97.2	97.7	96.0	97.2	96.9	96.5	97.0	97.5	96.2	97.1	96.9
AuxiLearn	97.3	97.7	96.4	97.1	97.2	96.6	97.0	97.3	96.3	97.1	97.0
AuxiNash	**97.4**	**98.1**	**96.5**	**97.2**	97.1	**97.0**	**97.1**	**97.6**	**96.3**	**97.6**	**97.2**

One can see that the constraints of this problem are convex and linear while the objective is concave. To overcome this challenge we follow Navon et al. (2022) and use the CCP to modify the concave objective into sequence of convex optimization problems. For further details please refer to Section 3.2 in Navon et al. (2022).

D. Experimental Details

Unless stated otherwise for AuxiNash we update the preference vector p every 25 optimization steps using SGD optimizer with momentum of 0.9 and learning rate of $5e-3$.

D.1. Illustrative Example

We follow a similar setup as in Navon et al. (2021). We consider a regression problem with parameters $W^T = (w_1, w_2) \in \mathbb{R}^2$, shared among tasks, with no task-specific parameters. The optimal parameters for the main and helpful auxiliary tasks are $W^{\star T} = (1, 1)$, while the optimal parameters for the harmful auxiliary are $\tilde{W}^T = (-1, -4)$. The main task is sampled from a Normal distribution $N(W^{\star T}x, \sigma_{\text{main}})$, with $\sigma_{\text{main}} = 20 \cdot \sigma_{\text{h}}$ where $\sigma_{\text{h}} = 0.25$ denotes the standard deviation for the noise of the helpful auxiliary. We use 1000 training and train the model using Adam optimizer and learning-rate $1e-2$ and batch-size of 256 for 1000 epochs.

D.2. CIFAR10-SSL

We follow the training and evaluation procedure proposed by previous works (Shi et al., 2020). We use the CIFAR10 dataset and divide the dataset to train/val/test splits each containing $45K$/$5K$/$10K$ respectively. We allocate $5K$/$10K$ samples as our labeled dataset for the supervised main task. Following Shi et al. (2020) we use the wide resnet (WRN-28-2) architecture which is ResNet with depth 28 and width 2. We train WRN for $50K$ iterations using Adam optimizer with learning rate of $5e-3$ and 256 batch size. Lastly, we use the main task performance on the validation set for early stopping.

D.3. Speech Commands

We used the Speech Command dataset, an audio dataset of spoken words designed to train and evaluate keyword spotting systems. We repeat the experiment twice with 1000 and 500 training samples distributed uniformly across 10 classes. For validation and test we used the original dataset split containing 3643 and 4107 samples respectively. For all methods we use a CNN network with 3 Convolution layers with a linear layer as the classifier. We train the model for 200 epochs with Adam optimizer, and a learning rate of $1e-3$. We present the per-task results for both experiments in Table 7 and Table 6.

D.4. Scene Understanding

We follow the training and evaluation protocol presented by previous studies (Liu et al., 2019b; 2022; Navon et al., 2022). For all methods we train SegNet (Badrinarayanan et al., 2017) model for 200 epochs with Adam optimizer. We use learning

Table 7. Speech Commands. Test performance on each one of the digit keywords. The train data-set consist of 500 samples, uniformly distributed between the 10 classes. Values are averaged over 3 random seeds.

Speech Commands - 500											
	0	1	2	3	4	5	6	7	8	9	Mean
STL	96.4	97.0	95.0	95.7	96.3	95.5	95.9	95.8	95.3	95.4	95.8
LS	95.9	96.9	94.8	95.8	96.0	95.3	95.9	96.4	95.3	95.6	95.7
PCGrad	96.0	97.1	95.0	95.7	95.9	95.8	95.8	96.0	94.7	95.3	95.7
CAGrad	96.4	97.1	95.0	95.9	96.3	95.9	96.0	96.1	95.0	95.4	95.9
Nash-MTL	96.6	97.4	94.1	95.7	96.3	95.4	95.9	96.2	94.8	95.2	95.7
GCS	96.4	97.6	**95.4**	96.3	96.5	95.8	96.1	96.5	96.0	97.0	96.3
OL-AUX	96.2	97.2	94.6	96.0	96.4	95.7	95.9	96.4	**96.1**	**97.5**	96.2
AuxiLearn	96.2	97.5	94.8	96.3	96.3	95.8	96.1	96.3	95.3	95.8	96.0
AuxiNash	**96.6**	**97.8**	95.3	**96.7**	**97.2**	**96.0**	**96.2**	**96.7**	95.6	95.6	**96.4**

Table 8. NYUv2. Test performance using fixed p. The preference for the main task, p_{main}, is in parentheses. Values are averaged over 3 random seeds.

	Segmentation		Depth		Surface Normal					
					Angle Distance ↓		Within t° ↑			$\mathbf{\Delta}\%$ ↓
	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	
AuxiNash (0.7)	**41.36**	66.48	0.5214	0.2138	24.46	18.78	30.80	58.33	70.42	$\mathbf{-7.62}$
AuxiNash (0.8)	41.27	66.47	0.5322	0.2172	24.34	18.67	31.05	58.53	70.64	-7.56
AuxiNash (0.9)	41.35	66.32	0.5493	0.2197	**24.39**	**18.61**	**31.23**	**58.64**	**70.68**	-7.28
AuxiNash	40.79	**66.79**	**0.5092**	**0.2042**	24.90	19.31	29.83	57.07	69.27	-6.80

rate of $1e-4$ for the first 100 epochs, then reduced it to $5e-5$ for the remaining epochs. We use a batch size of 2 and 8 for NYUv2 and CityScapes respectively. During training we apply data augmentations for all compared methods, more specifically, we use random scale and random horizontal flip as augmentations. Following Liu et al. (2019b; 2022); Navon et al. (2022) we report the test performance averaged over the last 10 epochs.

D.5. Controlling Task Preference

We use the Multi-MNIST dataset that consits of $120K$ training samples and $20K$ testing samples. We allocate $12K$ training examples to form validation set. We use a variant of LeNet as the trained model. Specifically, the model contains 3 CNN layers channels followed by 2 fully connected layers. We train all methods using Adam optimizer with learning rate $1e-4$ for 50 epochs and 256 batch size. For AuxiNash we repeat this experiment 11 times with varying preferences. More specifically, we select the first task preference from $p_1 \in \{0.01, 0.1, 0.2, 0.25, 0.4, 0.5, 0.6, 0.75, 0.8, 0.9, 0.99\}$ and set $p_2 = 1 - p_1$. We run the experiment equal amount of time with randomly selected seeds for Nash-MTL.

E. Additional Results

E.1. Fixed Preference Vector

Our method, AuxiNash, dynamically adjusts the preference vector throughout the optimization process. It is possible, however, to fix the preference vector to its initial value and train a model over a grid of such preferences. As discussed in the main text, this procedure does not scale well as the number of grid search values grows exponentially with the number of tasks. We also note that it is possible that the optimal preference changes during the optimization process, depending on the optimization dynamics.

Here we present an ablation study of optimizing AuxiNash with fixed preference p. We use the NYUv2 dataset and train the model with $p_{\text{main}} \in \{0.9, 0.8, 0.7\}$. The preference for the two auxiliary tasks is equal, e.g., $p_{\text{aux},i} = 0.1$ when $p_{\text{main}} = 0.8$. The results are presented in Table 8.

E.2. Sensitivity to Hyperparameters in Bi-level Optimization

Bi-level optimization may be sensitive to hyperparameters (HPs), however, we show that this is not the case for AuxiNash. Generally, we did not tune HPs related to the inverse Hessian vector product approximation but used the ones reported by (Lorraine et al., 2020). We find that these HPs generalize well to the various tasks that we studied without further tuning, suggesting that (good) HPs could transfer across tasks in our approach. For the remaining HP, namely the learning rate for updating the preference vector p, we fixed it to a single value of $5e-3$ throughout the experiments without tunning. Here we investigate the effect of the outer learning rate on the performance of AuxiNash. Specifically, we re-run the scene understanding experiment using NUYv2 dataset with 3 outer learning rate values $\{1e-2, 5e-3, 1e-3\}$. The results are presented in Table 9 and averaged over 3 seeds. The results in Table 9 suggest that AuxiNash is not sensitive to the choice of learning rates, and significantly outperforms all baseline methods for a range of outer loop learning rate choices.

Table 9. Outer learning rate ablation. Test performance using varying outer learning rates. Values are averaged over 3 random seeds.

	Segmentation		Depth		Surface Normal					
					Angle Distance ↓		Within t° ↑			$\boldsymbol{\Delta}\%$ ↓
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Mean	Median	11.25	22.5	30	
AuxiNash ($lr = 5e-3$)	40.79	66.79	0.5092	0.2042	25.01	19.21	30.14	57.20	69.15	-6.80
AuxiNash ($lr = 1e-2$)	40.75	66.74	0.5175	0.2055	24.90	19.31	29.83	57.07	69.27	-6.76
AuxiNash ($lr = 1e-3$)	41.03	66.55	0.5100	0.2043	25.08	19.58	29.34	56.55	68.83	-6.22