arXiv:1810.04650v2 [cs.LG] 11 Jan 2019

Multi-Task Learning as Multi-Objective Optimization

Ozan Sener
Intel Labs

Vladlen Koltun
Intel Labs

Abstract

In multi-task learning, multiple tasks are solved jointly, sharing inductive bias between them. Multi-task learning is inherently a multi-objective problem because different tasks may conflict, necessitating a trade-off. A common compromise is to optimize a proxy objective that minimizes a weighted linear combination of per-task losses. However, this workaround is only valid when the tasks do not compete, which is rarely the case. In this paper, we explicitly cast multi-task learning as multi-objective optimization, with the overall objective of finding a Pareto optimal solution. To this end, we use algorithms developed in the gradient-based multi-objective optimization literature. These algorithms are not directly applicable to large-scale learning problems since they scale poorly with the dimensionality of the gradients and the number of tasks. We therefore propose an upper bound for the multi-objective loss and show that it can be optimized efficiently. We further prove that optimizing this upper bound yields a Pareto optimal solution under realistic assumptions. We apply our method to a variety of multi-task deep learning problems including digit classification, scene understanding (joint semantic segmentation, instance segmentation, and depth estimation), and multi-label classification. Our method produces higher-performing models than recent multi-task learning formulations or per-task training.

1 Introduction

One of the most surprising results in statistics is Stein's paradox. Stein (1956) showed that it is better to estimate the means of three or more Gaussian random variables using samples from all of them rather than estimating them separately, even when the Gaussians are independent. Stein's paradox was an early motivation for multi-task learning (MTL) (Caruana, 1997), a learning paradigm in which data from multiple tasks is used with the hope to obtain superior performance over learning each task independently. Potential advantages of MTL go beyond the direct implications of Stein's paradox, since even seemingly unrelated real world tasks have strong dependencies due to the shared processes that give rise to the data. For example, although autonomous driving and object manipulation are seemingly unrelated, the underlying data is governed by the same laws of optics, material properties, and dynamics. This motivates the use of multiple tasks as an inductive bias in learning systems.

A typical MTL system is given a collection of input points and sets of targets for various tasks per point. A common way to set up the inductive bias across tasks is to design a parametrized hypothesis class that shares some parameters across tasks. Typically, these parameters are learned by solving an optimization problem that minimizes a weighted sum of the empirical risk for each task. However, the linear-combination formulation is only sensible when there is a parameter set that is effective across all tasks. In other words, minimization of a weighted sum of empirical risk is only valid if tasks are not competing, which is rarely the case. MTL with conflicting objectives requires modeling of the trade-off between tasks, which is beyond what a linear combination achieves.

An alternative objective for MTL is finding solutions that are not dominated by any others. Such solutions are said to be Pareto optimal. In this paper, we cast the objective of MTL in terms of finding Pareto optimal solutions.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

The problem of finding Pareto optimal solutions given multiple criteria is called multi-objective optimization. A variety of algorithms for multi-objective optimization exist. One such approach is the multiple-gradient descent algorithm (MGDA), which uses gradient-based optimization and provably converges to a point on the Pareto set (Désidéri, 2012). MGDA is well-suited for multi-task learning with deep networks. It can use the gradients of each task and solve an optimization problem to decide on an update over the shared parameters. However, there are two technical problems that hinder the applicability of MGDA on a large scale. (i) The underlying optimization problem does not scale gracefully to high-dimensional gradients, which arise naturally in deep networks. (ii) The algorithm requires explicit computation of gradients per task, which results in linear scaling of the number of backward passes and roughly multiplies the training time by the number of tasks.

In this paper, we develop a Frank-Wolfe-based optimizer that scales to high-dimensional problems. Furthermore, we provide an upper bound for the MGDA optimization objective and show that it can be computed via a single backward pass without explicit task-specific gradients, thus making the computational overhead of the method negligible. We prove that using our upper bound yields a Pareto optimal solution under realistic assumptions. The result is an exact algorithm for multi-objective optimization of deep networks with negligible computational overhead.

We empirically evaluate the presented method on three different problems. First, we perform an extensive evaluation on multi-digit classification with MultiMNIST (Sabour et al., 2017). Second, we cast multi-label classification as MTL and conduct experiments with the CelebA dataset (Liu et al., 2015b). Lastly, we apply the presented method to scene understanding; specifically, we perform joint semantic segmentation, instance segmentation, and depth estimation on the Cityscapes dataset (Cordts et al., 2016). The number of tasks in our evaluation varies from 2 to 40. Our method clearly outperforms all baselines.

2 Related Work

Multi-task learning. We summarize the work most closely related to ours and refer the interested reader to reviews by Ruder (2017) and Zhou et al. (2011b) for additional background. Multi-task learning (MTL) is typically conducted via hard or soft parameter sharing. In hard parameter sharing, a subset of the parameters is shared between tasks while other parameters are task-specific. In soft parameter sharing, all parameters are task-specific but they are jointly constrained via Bayesian priors (Xue et al., 2007; Bakker and Heskes, 2003) or a joint dictionary (Argyriou et al., 2007; Long and Wang, 2015; Yang and Hospedales, 2016; Ruder, 2017). We focus on hard parameter sharing with gradient-based optimization, following the success of deep MTL in computer vision (Bilen and Vedaldi, 2016; Misra et al., 2016; Rudd et al., 2016; Yang and Hospedales, 2016; Kokkinos, 2017; Zamir et al., 2018), natural language processing (Collobert and Weston, 2008; Dong et al., 2015; Liu et al., 2015a; Luong et al., 2015; Hashimoto et al., 2017), speech processing (Huang et al., 2013; Seltzer and Droppo, 2013; Huang et al., 2015), and even seemingly unrelated domains over multiple modalities (Kaiser et al., 2017).

Baxter (2000) theoretically analyze the MTL problem as interaction between individual learners and a meta-algorithm. Each learner is responsible for one task and a meta-algorithm decides how the shared parameters are updated. All aforementioned MTL algorithms use weighted summation as the meta-algorithm. Meta-algorithms that go beyond weighted summation have also been explored. Li et al. (2014) consider the case where each individual learner is based on kernel learning and utilize multi-objective optimization. Zhang and Yeung (2010) consider the case where each learner is a linear model and use a task affinity matrix. Zhou et al. (2011a) and Bagherjeiran et al. (2005) use the assumption that tasks share a dictionary and develop an expectation-maximization-like meta-algorithm. de Miranda et al. (2012) and Zhou et al. (2017b) use swarm optimization. None of these methods apply to gradient-based learning of high-capacity models such as modern deep networks. Kendall et al. (2018) and Chen et al. (2018) propose heuristics based on uncertainty and gradient magnitudes, respectively, and apply their methods to convolutional neural networks. Another recent work uses multi-agent reinforcement learning (Rosenbaum et al., 2017).

Multi-objective optimization. Multi-objective optimization addresses the problem of optimizing a set of possibly contrasting objectives. We recommend Miettinen (1998) and Ehrgott (2005) for surveys of this field. Of particular relevance to our work is gradient-based multi-objective optimization, as developed by Fliege and Svaiter (2000), Schäffler et al. (2002), and Désidéri (2012). These methods

use multi-objective Karush-Kuhn-Tucker (KKT) conditions (Kuhn and Tucker, 1951) and find a descent direction that decreases all objectives. This approach was extended to stochastic gradient descent by Peitz and Dellnitz (2018) and Poirion et al. (2017). In machine learning, these methods have been applied to multi-agent learning (Ghosh et al., 2013; Pirotta and Restelli, 2016; Parisi et al., 2014), kernel learning (Li et al., 2014), sequential decision making (Roijers et al., 2013), and Bayesian optimization (Shah and Ghahramani, 2016; Hernández-Lobato et al., 2016). Our work applies gradient-based multi-objective optimization to multi-task learning.

3 Multi-Task Learning as Multi-Objective Optimization

Consider a multi-task learning (MTL) problem over an input space $\mathcal{X}$ and a collection of task spaces $\{\mathcal{Y}^t\}_{t\in[T]}$, such that a large dataset of i.i.d. data points $\{\mathbf{x}_i, y_i^1, \dots, y_i^T\}_{i\in[N]}$ is given where T is the number of tasks, N is the number of data points, and y_i^t is the label of the tth task for the ith data point.[1] We further consider a parametric hypothesis class per task as $f^t(\mathbf{x}; \boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) : \mathcal{X} \to \mathcal{Y}^t$, such that some parameters ($\boldsymbol{\theta}^{sh}$) are shared between tasks and some ($\boldsymbol{\theta}^t$) are task-specific. We also consider task-specific loss functions $\mathcal{L}^t(\cdot,\cdot) : \mathcal{Y}^t \times \mathcal{Y}^t \to \mathbb{R}^+$.

Although many hypothesis classes and loss functions have been proposed in the MTL literature, they generally yield the following empirical risk minimization formulation:

$$\min_{\substack{\boldsymbol{\theta}^{sh},\\ \boldsymbol{\theta}^1,\dots,\boldsymbol{\theta}^T}} \quad \sum_{t=1}^{T} c^t \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) \tag{1}$$

for some static or dynamically computed weights c^t per task, where $\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)$ is the empirical loss of the task t, defined as $\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) \triangleq \frac{1}{N}\sum_i \mathcal{L}\big(f^t(\mathbf{x}_i; \boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t), y_i^t\big)$.

Although the weighted summation formulation (1) is intuitively appealing, it typically either requires an expensive grid search over various scalings or the use of a heuristic (Kendall et al., 2018; Chen et al., 2018). A basic justification for scaling is that it is not possible to define global optimality in the MTL setting. Consider two sets of solutions $\boldsymbol{\theta}$ and $\bar{\boldsymbol{\theta}}$ such that $\hat{\mathcal{L}}^{t_1}(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^{t_1}) < \hat{\mathcal{L}}^{t_1}(\bar{\boldsymbol{\theta}}^{sh}, \bar{\boldsymbol{\theta}}^{t_1})$ and $\hat{\mathcal{L}}^{t_2}(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^{t_2}) > \hat{\mathcal{L}}^{t_2}(\bar{\boldsymbol{\theta}}^{sh}, \bar{\boldsymbol{\theta}}^{t_2})$, for some tasks t_1 and t_2. In other words, solution $\boldsymbol{\theta}$ is better for task t_1 whereas $\bar{\boldsymbol{\theta}}$ is better for t_2. It is not possible to compare these two solutions without a pairwise importance of tasks, which is typically not available.

Alternatively, MTL can be formulated as multi-objective optimization: optimizing a collection of possibly conflicting objectives. This is the approach we take. We specify the multi-objective optimization formulation of MTL using a vector-valued loss $\mathbf{L}$:

$$\min_{\substack{\boldsymbol{\theta}^{sh},\\ \boldsymbol{\theta}^1,\dots,\boldsymbol{\theta}^T}} \mathbf{L}(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^1, \dots, \boldsymbol{\theta}^T) = \min_{\substack{\boldsymbol{\theta}^{sh},\\ \boldsymbol{\theta}^1,\dots,\boldsymbol{\theta}^T}} \big(\hat{\mathcal{L}}^1(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^1), \dots, \hat{\mathcal{L}}^T(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^T)\big)^\intercal. \tag{2}$$

The goal of multi-objective optimization is achieving Pareto optimality.

Definition 1 (Pareto optimality for MTL)

(a) *A solution $\boldsymbol{\theta}$ dominates a solution $\bar{\boldsymbol{\theta}}$ if $\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) \leq \hat{\mathcal{L}}^t(\bar{\boldsymbol{\theta}}^{sh}, \bar{\boldsymbol{\theta}}^t)$ for all tasks t and $\mathbf{L}(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^1, \dots, \boldsymbol{\theta}^T) \neq \mathbf{L}(\bar{\boldsymbol{\theta}}^{sh}, \bar{\boldsymbol{\theta}}^1, \dots, \bar{\boldsymbol{\theta}}^T)$.*

(b) *A solution $\boldsymbol{\theta}^\star$ is called Pareto optimal if there exists no solution $\boldsymbol{\theta}$ that dominates $\boldsymbol{\theta}^\star$.*

The set of Pareto optimal solutions is called the Pareto set ($\mathcal{P}_{\boldsymbol{\theta}}$) and its image is called the Pareto front ($\mathcal{P}_{\mathbf{L}} = \{\mathbf{L}(\boldsymbol{\theta})\}_{\boldsymbol{\theta}\in\mathcal{P}_{\boldsymbol{\theta}}}$). In this paper, we focus on gradient-based multi-objective optimization due to its direct relevance to gradient-based MTL.

In the rest of this section, we first summarize in Section 3.1 how multi-objective optimization can be performed with gradient descent. Then, we suggest in Section 3.2 a practical algorithm for performing multi-objective optimization over very large parameter spaces. Finally, in Section 3.3 we propose an efficient solution for multi-objective optimization designed directly for high-capacity deep networks. Our method scales to very large models and a high number of tasks with negligible overhead.

[1] This definition can be extended to the partially-labelled case by extending $\mathcal{Y}^t$ with a null label.

3.1 Multiple Gradient Descent Algorithm

As in the single-objective case, multi-objective optimization can be solved to local optimality via gradient descent. In this section, we summarize one such approach, called the multiple gradient descent algorithm (MGDA) (Désidéri, 2012). MGDA leverages the Karush-Kuhn-Tucker (KKT) conditions, which are necessary for optimality (Fliege and Svaiter, 2000; Schäffler et al., 2002; Désidéri, 2012). We now state the KKT conditions for both task-specific and shared parameters:

- There exist $\alpha^1, \ldots, \alpha^T \geq 0$ such that $\sum_{t=1}^{T} \alpha^t = 1$ and $\sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) = 0$
- For all tasks t, $\nabla_{\boldsymbol{\theta}^t} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) = 0$

Any solution that satisfies these conditions is called a Pareto stationary point. Although every Pareto optimal point is Pareto stationary, the reverse may not be true. Consider the optimization problem

$$\min_{\alpha^1,\ldots,\alpha^T} \left\{ \left\| \sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) \right\|_2^2 \middle| \sum_{t=1}^{T} \alpha^t = 1, \alpha^t \geq 0 \quad \forall t \right\} \tag{3}$$

Désidéri (2012) showed that either the solution to this optimization problem is 0 and the resulting point satisfies the KKT conditions, or the solution gives a descent direction that improves all tasks. Hence, the resulting MTL algorithm would be gradient descent on the task-specific parameters followed by solving (3) and applying the solution ($\sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}}$) as a gradient update to shared parameters. We discuss how to solve (3) for an arbitrary model in Section 3.2 and present an efficient solution when the underlying model is an encoder-decoder in Section 3.3.

3.2 Solving the Optimization Problem

The optimization problem defined in (3) is equivalent to finding a minimum-norm point in the convex hull of the set of input points. This problem arises naturally in computational geometry: it is equivalent to finding the closest point within a convex hull to a given query point. It has been studied extensively (Makimoto et al., 1994; Wolfe, 1976; Sekitani and Yamamoto, 1993). Although many algorithms have been proposed, they do not apply in our setting because the assumptions they make do not hold. Algorithms proposed in the computational geometry literature address the problem of finding minimum-norm points in the convex hull of a large number of points in a low-dimensional space (typically of dimensionality 2 or 3). In our setting, the number of points is the number of tasks and is typically low; in contrast, the dimensionality is the number of shared parameters and can be in the millions. We therefore use a different approach based on convex optimization, since (3) is a convex quadratic problem with linear constraints.

Before we tackle the general case, let's consider the case of two tasks. The optimization problem can be defined as $\min_{\alpha \in [0,1]} \|\alpha \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^1(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^1) + (1-\alpha) \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^2(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^2)\|_2^2$, which is a one-dimensional quadratic function of α with an analytical solution:

$$\hat{\alpha} = \left[\frac{\left(\nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^2(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^2) - \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^1(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^1)\right)^\intercal \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^2(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^2)}{\|\nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^1(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^1) - \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^2(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^2)\|_2^2} \right]_{+,\frac{1}{\intercal}} \tag{4}$$

where $[\cdot]_{+,\frac{1}{\intercal}}$ represents clipping to $[0,1]$ as $[a]_{+,\frac{1}{\intercal}} = \max(\min(a,1),0)$. We further visualize this solution in Figure 1. Although this is only applicable when $T = 2$, this enables efficient application of the Frank-Wolfe algorithm (Jaggi, 2013) since the line search can be solved analytically. Hence, we use Frank-Wolfe to solve the constrained optimization problem, using (4) as a subroutine for the line search. We give all the update equations for the Frank-Wolfe solver in Algorithm 2.

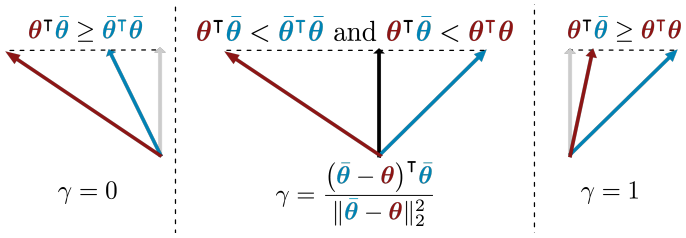

Figure 1: Visualisation of the min-norm point in the convex hull of two points ($\min_{\gamma\in[0,1]} \|\gamma\boldsymbol{\theta} + (1-\gamma)\bar{\boldsymbol{\theta}}\|_2^2$). As the geometry suggests, the solution is either an edge case or a perpendicular vector.

Algorithm 1
$\min_{\gamma\in[0,1]} \|\gamma\boldsymbol{\theta} + (1-\gamma)\bar{\boldsymbol{\theta}}\|_2^2$

if $\boldsymbol{\theta}^\intercal\bar{\boldsymbol{\theta}} \geq \boldsymbol{\theta}^\intercal\boldsymbol{\theta}$ **then**
  $\gamma = 1$
else if $\boldsymbol{\theta}^\intercal\bar{\boldsymbol{\theta}} \geq \bar{\boldsymbol{\theta}}^\intercal\bar{\boldsymbol{\theta}}$ **then**
  $\gamma = 0$
else
  $\gamma = \frac{(\bar{\boldsymbol{\theta}}-\boldsymbol{\theta})^\intercal\bar{\boldsymbol{\theta}}}{\|\boldsymbol{\theta}-\bar{\boldsymbol{\theta}}\|_2^2}$
end if

Algorithm 2 Update Equations for MTL

for $t = 1$ **to** T **do**
  $\boldsymbol{\theta}^t = \boldsymbol{\theta}^t - \eta\nabla_{\boldsymbol{\theta}^t}\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)$ ▷ Gradient descent on task-specific parameters
end for
$\alpha^1, \ldots, \alpha^T$ = FRANKWOLFESOLVER($\boldsymbol{\theta}$) ▷ Solve (3) to find a common descent direction
$\boldsymbol{\theta}^{sh} = \boldsymbol{\theta}^{sh} - \eta\sum_{t=1}^T \alpha^t\nabla_{\boldsymbol{\theta}^{sh}}\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)$ ▷ Gradient descent on shared parameters

procedure FRANKWOLFESOLVER($\boldsymbol{\theta}$)
  Initialize $\boldsymbol{\alpha} = (\alpha^1, \ldots, \alpha^T) = (\frac{1}{T}, \ldots, \frac{1}{T})$
  Precompute $\mathbf{M}$ st. $\mathbf{M}_{i,j} = \left(\nabla_{\boldsymbol{\theta}^{sh}}\hat{\mathcal{L}}^i(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^i)\right)^\intercal\left(\nabla_{\boldsymbol{\theta}^{sh}}\hat{\mathcal{L}}^j(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^j)\right)$
  repeat
    $\hat{t} = \arg\min_r \sum_t \alpha^t\mathbf{M}_{rt}$
    $\hat{\gamma} = \arg\min_\gamma \left((1-\gamma)\boldsymbol{\alpha} + \gamma\boldsymbol{e}_{\hat{t}}\right)^\intercal\mathbf{M}\left((1-\gamma)\boldsymbol{\alpha} + \gamma\boldsymbol{e}_{\hat{t}}\right)$ ▷ Using Algorithm 1
    $\boldsymbol{\alpha} = (1-\hat{\gamma})\boldsymbol{\alpha} + \hat{\gamma}\boldsymbol{e}_{\hat{t}}$
  until $\hat{\gamma} \sim 0$ **or** Number of Iterations Limit
  return $\alpha^1, \ldots, \alpha^T$
end procedure

3.3 Efficient Optimization for Encoder-Decoder Architectures

The MTL update described in Algorithm 2 is applicable to any problem that uses optimization based on gradient descent. Our experiments also suggest that the Frank-Wolfe solver is efficient and accurate as it typically converges in a modest number of iterations with negligible effect on training time. However, the algorithm we described needs to compute $\nabla_{\boldsymbol{\theta}^{sh}}\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)$ for each task t, which requires a backward pass over the shared parameters for each task. Hence, the resulting gradient computation would be the forward pass followed by T backward passes. Considering the fact that computation of the backward pass is typically more expensive than the forward pass, this results in linear scaling of the training time and can be prohibitive for problems with more than a few tasks.

We now propose an efficient method that optimizes an upper bound of the objective and requires only a single backward pass. We further show that optimizing this upper bound yields a Pareto optimal solution under realistic assumptions. The architectures we address conjoin a shared representation function with task-specific decision functions. This class of architectures covers most of the existing deep MTL models and can be formally defined by constraining the hypothesis class as

$$f^t(\mathbf{x}; \boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) = (f^t(\cdot; \boldsymbol{\theta}^t) \circ g(\cdot; \boldsymbol{\theta}^{sh}))(\mathbf{x}) = f^t(g(\mathbf{x}; \boldsymbol{\theta}^{sh}); \boldsymbol{\theta}^t) \tag{5}$$

where g is the representation function shared by all tasks and f^t are the task-specific functions that take this representation as input. If we denote the representations as $\mathbf{Z} = \left(\mathbf{z}_1, \ldots, \mathbf{z}_N\right)$, where $\mathbf{z}_i = g(\mathbf{x}_i; \boldsymbol{\theta}^{sh})$, we can state the following upper bound as a direct consequence of the chain rule:

$$\left\|\sum_{t=1}^T \alpha^t\nabla_{\boldsymbol{\theta}^{sh}}\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)\right\|_2^2 \leq \left\|\frac{\partial\mathbf{Z}}{\partial\boldsymbol{\theta}^{sh}}\right\|_2^2\left\|\sum_{t=1}^T \alpha^t\nabla_{\mathbf{Z}}\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)\right\|_2^2 \tag{6}$$

where $\left\|\frac{\partial\mathbf{Z}}{\partial\boldsymbol{\theta}^{sh}}\right\|_2$ is the matrix norm of the Jacobian of $\mathbf{Z}$ with respect to $\boldsymbol{\theta}^{sh}$. Two desirable properties of this upper bound are that (i) $\nabla_{\mathbf{Z}}\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)$ can be computed in a single backward pass for all

tasks and (ii) $\left\|\frac{\partial \mathbf{Z}}{\partial \boldsymbol{\theta}^{sh}}\right\|_2^2$ is not a function of $\alpha^1, \ldots, \alpha^T$, hence it can be removed when it is used as an optimization objective. We replace the $\left\|\sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)\right\|_2^2$ term with the upper bound we have just derived in order to obtain the approximate optimization problem and drop the $\left\|\frac{\partial \mathbf{Z}}{\partial \boldsymbol{\theta}^{sh}}\right\|_2^2$ term since it does not affect the optimization. The resulting optimization problem is

$$\min_{\alpha^1,\ldots,\alpha^T} \left\{ \left\| \sum_{t=1}^{T} \alpha^t \nabla_{\mathbf{Z}} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) \right\|_2^2 \middle| \sum_{t=1}^{T} \alpha^t = 1, \alpha^t \geq 0 \quad \forall t \right\} \tag{MGDA-UB}$$

We refer to this problem as MGDA-UB (Multiple Gradient Descent Algorithm – Upper Bound). In practice, MGDA-UB corresponds to using the gradients of the task losses with respect to the representations instead of the shared parameters. We use Algorithm 2 with only this change as the final method.

Although MGDA-UB is an approximation of the original optimization problem, we now state a theorem that shows that our method produces a Pareto optimal solution under mild assumptions. The proof is given in the supplement.

Theorem 1 *Assume $\frac{\partial \mathbf{Z}}{\partial \theta^{sh}}$ is full-rank. If $\alpha^{1,\ldots,T}$ is the solution of MGDA-UB, one of the following is true:*

(a) $\sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) = 0$ *and the current parameters are Pareto stationary.*

(b) $\sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)$ *is a descent direction that decreases all objectives.*

This result follows from the fact that as long as $\frac{\partial \mathbf{Z}}{\partial \boldsymbol{\theta}^{sh}}$ is full rank, optimizing the upper bound corresponds to minimizing the norm of the convex combination of the gradients using the Mahalonobis norm defined by $\frac{\partial \mathbf{Z}}{\partial \boldsymbol{\theta}^{sh}}^{\mathsf{T}} \frac{\partial \mathbf{Z}}{\partial \boldsymbol{\theta}^{sh}}$. The non-singularity assumption is reasonable as singularity implies that tasks are linearly related and a trade-off is not necessary. In summary, our method provably finds a Pareto stationary point with negligible computational overhead and can be applied to any deep multi-objective problem with an encoder-decoder model.

4 Experiments

We evaluate the presented MTL method on a number of problems. First, we use MultiMNIST (Sabour et al., 2017), an MTL adaptation of MNIST (LeCun et al., 1998). Next, we tackle multi-label classification on the CelebA dataset (Liu et al., 2015b) by considering each label as a distinct binary classification task. These problems include both classification and regression, with the number of tasks ranging from 2 to 40. Finally, we experiment with scene understanding, jointly tackling the tasks of semantic segmentation, instance segmentation, and depth estimation on the Cityscapes dataset (Cordts et al., 2016). We discuss each experiment separately in the following subsections.

The baselines we consider are (i) **uniform scaling:** minimizing a uniformly weighted sum of loss functions $\frac{1}{T}\sum_t \mathcal{L}^t$, (ii) **single task:** solving tasks independently, (iii) **grid search:** exhaustively trying various values from $\{c^t \in [0,1] | \sum_t c^t = 1\}$ and optimizing for $\frac{1}{T}\sum_t c^t \mathcal{L}^t$, (iv) **Kendall et al. (2018):** using the uncertainty weighting proposed by Kendall et al. (2018), and (v) **GradNorm:** using the normalization proposed by Chen et al. (2018).

4.1 MultiMNIST

Our initial experiments are on MultiMNIST, an MTL version of the MNIST dataset (Sabour et al., 2017). In order to convert digit classification into a multi-task problem, Sabour et al. (2017) overlaid multiple images together. We use a similar construction. For each image, a different one is chosen uniformly in random. Then one of these images is put at the top-left and the other one is at the bottom-right. The resulting tasks are: classifying the digit on the top-left (task-L) and classifying the digit on the bottom-right (task-R). We use 60K examples and directly apply existing single-task MNIST models. The MultiMNIST dataset is illustrated in the supplement.

We use the LeNet architecture (LeCun et al., 1998). We treat all layers except the last as the representation function g and put two fully-connected layers as task-specific functions (see the

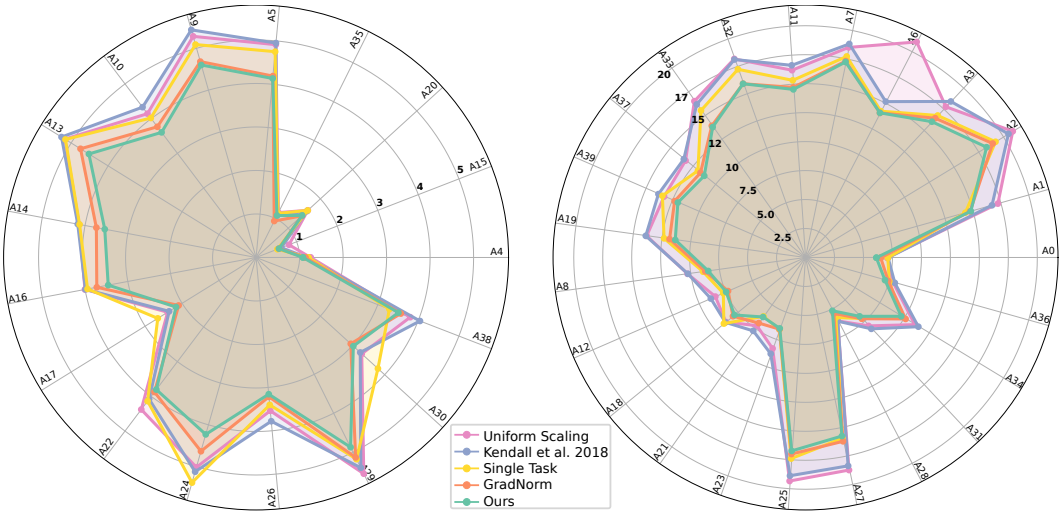

Figure 2: Radar charts of percentage error per attribute on CelebA (Liu et al., 2015b). Lower is better. We divide attributes into two sets for legibility: easy on the left, hard on the right. Zoom in for details.

supplement for details). We visualize the performance profile as a scatter plot of accuracies on task-L and task-R in Figure 3, and list the results in Table 3.

In this setup, any static scaling results in lower accuracy than solving each task separately (the single-task baseline). The two tasks appear to compete for model capacity, since increase in the accuracy of one task results in decrease in the accuracy of the other. Uncertainty weighting (Kendall et al., 2018) and GradNorm (Chen et al., 2018) find solutions that are slightly better than grid search but distinctly worse than the single-task baseline. In contrast, our method finds a solution that efficiently utilizes the model capacity and yields accuracies that are as good as the single-task solutions. This experiment demonstrates the effectiveness of our method as well as the necessity of treating MTL as multi-objective optimization. Even after a large hyper-parameter search, *any* scaling of tasks does not approach the effectiveness of our method.

4.2 Multi-Label Classification

Next, we tackle multi-label classification. Given a set of attributes, multi-label classification calls for deciding whether each attribute holds for the input. We use the CelebA dataset (Liu et al., 2015b), which includes 200K face images annotated with 40 attributes. Each attribute gives rise to a binary classification task and we cast this as a 40-way MTL problem. We use ResNet-18 (He et al., 2016) without the final layer as a shared representation function, and attach a linear layer for each attribute (see the supplement for further details).

We plot the resulting error for each binary classification task as a radar chart in Figure 2. The average over them is listed in Table 1. We skip grid search since it is not feasible over 40 tasks. Although uniform scaling is the norm in the multi-label classification literature, single-task performance is significantly better. Our method outperforms baselines for significant majority of tasks and achieves comparable performance in rest. This experiment also shows that our method remains effective when the number of tasks is high.

Table 1: Mean of error per category of MTL algorithms in multi-label classification on CelebA (Liu et al., 2015b).

	Average error
Single task	8.77
Uniform scaling	9.62
Kendall et al. 2018	9.53
GradNorm	8.44
Ours	**8.25**

4.3 Scene Understanding

To evaluate our method in a more realistic setting, we use scene understanding. Given an RGB image, we solve three tasks: semantic segmentation (assigning pixel-level class labels), instance

Table 2: Effect of the MGDA-UB approximation. We report the final accuracies as well as training times for our method with and without the approximation.

	Scene understanding (3 tasks)				Multi-label (40 tasks)	
	Training time	Segmentation mIoU [%]	Instance error [px]	Disparity error [px]	Training time (hour)	Average error
Ours (w/o approx.)	38.6	66.13	10.28	2.59	429.9	8.33
Ours	**23.3**	**66.63**	**10.25**	**2.54**	**16.1**	**8.25**

segmentation (assigning pixel-level instance labels), and monocular depth estimation (estimating continuous disparity per pixel). We follow the experimental procedure of Kendall et al. (2018) and use an encoder-decoder architecture. The encoder is based on ResNet-50 (He et al., 2016) and is shared by all three tasks. The decoders are task-specific and are based on the pyramid pooling module (Zhao et al., 2017) (see the supplement for further implementation details).

Since the output space of instance segmentation is unconstrained (the number of instances is not known in advance), we use a proxy problem as in Kendall et al. (2018). For each pixel, we estimate the location of the center of mass of the instance that encompasses the pixel. These center votes can then be clustered to extract the instances. In our experiments, we directly report the MSE in the proxy task. Figure 4 shows the performance profile for each pair of tasks, although we perform all experiments on all three tasks jointly. The pairwise performance profiles shown in Figure 4 are simply 2D projections of the three-dimensional profile, presented this way for legibility. The results are also listed in Table 4.

MTL outperforms single-task accuracy, indicating that the tasks cooperate and help each other. Our method outperforms all baselines on all tasks.

4.4 Role of the Approximation

In order to understand the role of the approximation proposed in Section 3.3, we compare the final performance and training time of our algorithm with and without the presented approximation in Table 2 (runtime measured on a single Titan Xp GPU). For a small number of tasks (3 for scene understanding), training time is reduced by 40%. For the multi-label classification experiment (40 tasks), the presented approximation accelerates learning by a factor of 25.

On the accuracy side, we expect both methods to perform similarly as long as the full-rank assumption is satisfied. As expected, the accuracy of both methods is very similar. Somewhat surprisingly, our approximation results in slightly improved accuracy in all experiments. While counter-intuitive at first, we hypothesize that this is related to the use of SGD in the learning algorithm. Stability analysis in convex optimization suggests that if gradients are computed with an error $\hat{\nabla}_{\boldsymbol{\theta}}\mathcal{L}^t = \nabla_{\boldsymbol{\theta}}\mathcal{L}^t + \mathbf{e}^t$ ($\boldsymbol{\theta}$ corresponds to $\boldsymbol{\theta}^{sh}$ in (3)), as opposed to $\mathbf{Z}$ in the approximate problem in (MGDA-UB), the error in the solution is bounded as $\|\hat{\alpha} - \alpha\|_2 \leq \mathcal{O}(\max_t \|\mathbf{e}^t\|_2)$. Considering the fact that the gradients are computed over the full parameter set (millions of dimensions) for the original problem and over a smaller space for the approximation (batch size times representation which is in the thousands), the dimension of the error vector is significantly higher in the original problem. We expect the l_2 norm of such a random vector to depend on the dimension.

In summary, our quantitative analysis of the approximation suggests that (i) the approximation does not cause an accuracy drop and (ii) by solving an equivalent problem in a lower-dimensional space, our method achieves both better computational efficiency and higher stability.

5 Conclusion

We described an approach to multi-task learning. Our approach is based on multi-objective optimization. In order to apply multi-objective optimization to MTL, we described an efficient algorithm as well as specific approximations that yielded a deep MTL algorithm with almost no computational overhead. Our experiments indicate that the resulting algorithm is effective for a wide range of multi-task scenarios.

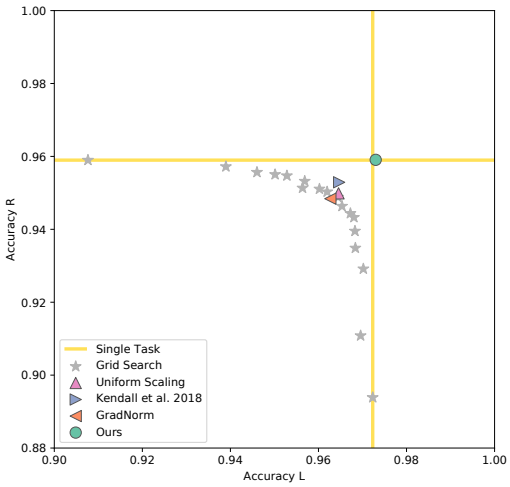

Figure 3: **MultiMNIST accuracy profile.** We plot the obtained accuracy in detecting the left and right digits for all baselines. The grid-search results suggest that the tasks compete for model capacity. Our method is the only one that finds a solution that is as good as training a dedicated model for each task. Top-right is better.

Table 3: Performance of MTL algorithms on MultiMNIST. Single-task baselines solve tasks separately, with dedicated models, but are shown in the same row for clarity.

	Left digit accuracy [%]	Right digit accuracy [%]
Single task	**97.23**	**95.90**
Uniform scaling	96.46	94.99
Kendall et al. 2018	96.47	95.29
GradNorm	96.27	94.84
Ours	**97.26**	**95.90**

Table 4: Performance of MTL algorithms in joint semantic segmentation, instance segmentation, and depth estimation on Cityscapes. Single-task baselines solve tasks separately but are shown in the same row for clarity.

	Segmentation mIoU [%]	Instance error [px]	Disparity error [px]
Single task	60.68	11.34	2.78
Uniform scaling	54.59	10.38	2.96
Kendall et al. 2018	64.21	11.54	2.65
GradNorm	64.81	11.31	**2.57**
Ours	**66.63**	**10.25**	**2.54**

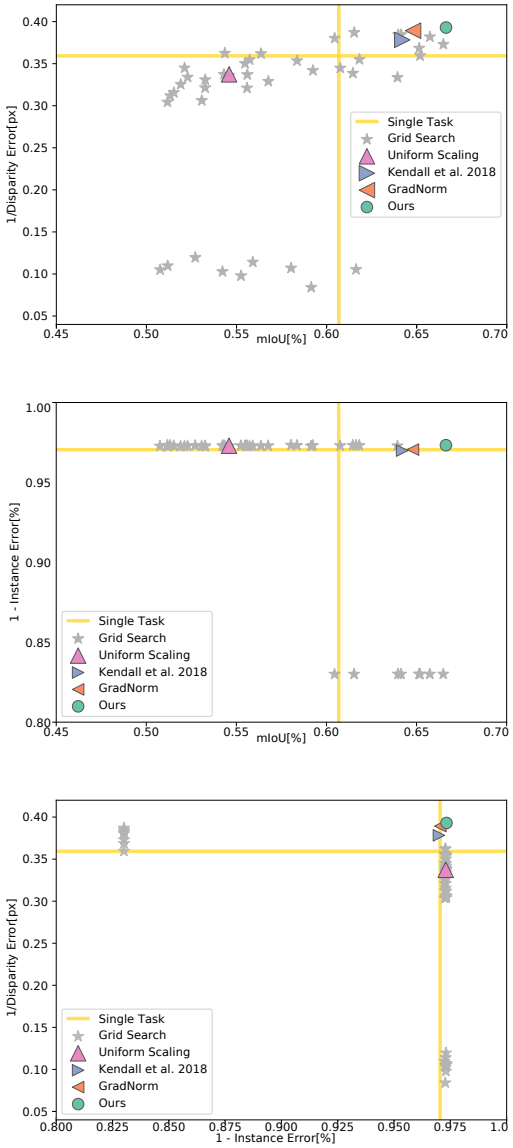

Figure 4: **Cityscapes performance profile.** We plot the performance of all baselines for the tasks of semantic segmentation, instance segmentation, and depth estimation. We use mIoU for semantic segmentation, error of per-pixel regression (normalized to image size) for instance segmentation, and disparity error for depth estimation. To convert errors to performance measures, we use $1 -$ instance error and 1/disparity error. We plot 2D projections of the performance profile for each pair of tasks. Although we plot pairwise projections for visualization, each point in the plots solves all tasks. Top-right is better.

References

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *NIPS*, 2007.

A. Bagherjeiran, R. Vilalta, and C. F. Eick. Content-based image retrieval through a multi-agent meta-learning framework. In *International Conference on Tools with Artificial Intelligence*, 2005.

B. Bakker and T. Heskes. Task clustering and gating for Bayesian multitask learning. *JMLR*, 4:83–99, 2003.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

H. Bilen and A. Vedaldi. Integrated perception with recurrent multi-task neural networks. In *NIPS*, 2016.

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

Z. Chen, V. Badrinarayanan, C. Lee, and A. Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, 2018.

R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.

M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

P. B. C. de Miranda, R. B. C. Prudêncio, A. C. P. L. F. de Carvalho, and C. Soares. Combining a multi-objective optimization approach with meta-learning for SVM parameter selection. In *International Conference on Systems, Man, and Cybernetics*, 2012.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

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

D. Dong, H. Wu, W. He, D. Yu, and H. Wang. Multi-task learning for multiple language translation. In *ACL*, 2015.

M. Ehrgott. *Multicriteria Optimization (2. ed.)*. Springer, 2005.

J. Fliege and B. F. Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000.

S. Ghosh, C. Lovell, and S. R. Gunn. Towards Pareto descent directions in sampling experts for multiple tasks in an on-line learning paradigm. In *AAAI Spring Symposium: Lifelong Machine Learning*, 2013.

K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. In *EMNLP*, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

D. Hernández-Lobato, J. M. Hernández-Lobato, A. Shah, and R. P. Adams. Predictive entropy search for multi-objective bayesian optimization. In *ICML*, 2016.

J.-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *ICASSP*, 2013.

Z. Huang, J. Li, S. M. Siniscalchi, I.-F. Chen, J. Wu, and C.-H. Lee. Rapid adaptation for deep neural networks through multi-task learning. In *Interspeech*, 2015.

M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML*, 2013.

L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit. One model to learn them all. *arXiv:1706.05137*, 2017.

A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

I. Kokkinos. UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, 2017.

H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, Calif., 1951. University of California Press.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

C. Li, M. Georgiopoulos, and G. C. Anagnostopoulos. Pareto-path multi-task multiple kernel learning. *arXiv:1404.3190*, 2014.

X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL HLT*, 2015a.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015b.

M. Long and J. Wang. Learning multiple tasks with deep relationship networks. *arXiv:1506.02117*, 2015.

M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser. Multi-task sequence to sequence learning. *arXiv:1511.06114*, 2015.

N. Makimoto, I. Nakagawa, and A. Tamura. An efficient algorithm for finding the minimum norm point in the convex hull of a finite point set in the plane. *Operations Research Letters*, 16(1):33–40, 1994.

K. Miettinen. *Nonlinear Multiobjective Optimization*. Springer, 1998.

I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016.

S. Parisi, M. Pirotta, N. Smacchia, L. Bascetta, and M. Restelli. Policy gradient approaches for multi-objective sequential decision making. In *IJCNN*, 2014.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Workshops*, 2017.

S. Peitz and M. Dellnitz. Gradient-based multiobjective optimization with uncertainties. In *NEO*, 2018.

M. Pirotta and M. Restelli. Inverse reinforcement learning through policy gradient minimization. In *AAAI*, 2016.

F. Poirion, Q. Mercier, and J. Désidéri. Descent algorithm for nonsmooth stochastic multiobjective optimization. *Computational Optimization and Applications*, 68(2):317–331, 2017.

D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

C. Rosenbaum, T. Klinger, and M. Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv:1711.01239*, 2017.

E. M. Rudd, M. Günther, and T. E. Boult. MOON: A mixed objective optimization network for the recognition of facial attributes. In *ECCV*, 2016.

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

S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *NIPS*, 2017.

S. Schäffler, R. Schultz, and K. Weinzierl. Stochastic method for the solution of unconstrained vector optimization problems. *Journal of Optimization Theory and Applications*, 114(1):209–222, 2002.

K. Sekitani and Y. Yamamoto. A recursive algorithm for finding the minimum norm point in a polytope and a pair of closest points in two polytopes. *Mathematical Programming*, 61(1-3):233–249, 1993.

M. L. Seltzer and J. Droppo. Multi-task learning in deep neural networks for improved phoneme recognition. In *ICASSP*, 2013.

A. Shah and Z. Ghahramani. Pareto frontier learning with expensive correlated objectives. In *ICML*, 2016.

C. Stein. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. Technical report, Stanford University, US, 1956.

P. Wolfe. Finding the nearest point in a polytope. *Mathematical Programming*, 11(1):128–149, 1976.

Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with dirichlet process priors. *JMLR*, 8:35–63, 2007.

Y. Yang and T. M. Hospedales. Trace norm regularised deep multi-task learning. *arXiv:1606.04038*, 2016.

A. R. Zamir, A. Sax, W. B. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.

Y. Zhang and D. Yeung. A convex formulation for learning task relationships in multi-task learning. In *UAI*, 2010.

H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017a.

D. Zhou, J. Wang, B. Jiang, H. Guo, and Y. Li. Multi-task multi-view learning based on cooperative multi-objective optimization. *IEEE Access*, 2017b.

J. Zhou, J. Chen, and J. Ye. Clustered multi-task learning via alternating structure optimization. In *NIPS*, 2011a.

J. Zhou, J. Chen, and J. Ye. MALSAR: Multi-task learning via structural regularization. *Arizona State University*, 2011b.

A Proof of Theorem 1

Proof. We begin by showing that if the optimum value of (MGDA-UB) is 0, so is the optimum value of (3). This shows the first case of the theorem. Then, we will show the second part.

If the optimum value of (MGDA-UB) is 0,

$$\sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) = \frac{\partial \mathbf{Z}}{\partial \theta^{sh}} \sum_{t=1}^{T} \alpha^t \nabla_{\mathbf{Z}} \hat{\mathcal{L}}^t = \sum_{t=1}^{T} \alpha^t \nabla_{\theta^{sh}} \hat{\mathcal{L}}^t = 0 \tag{7}$$

Hence $\alpha^1, \ldots, \alpha^T$ is the solution of (3) and the optimal value of (3) is 0. This proves the first case of the theorem. Before we move to the second case, we state a straightforward corollary. Since $\frac{\partial \mathbf{Z}}{\partial \theta^{sh}}$ is full rank, this equivalence is bi-directional. In other words, if $\alpha^1, \ldots, \alpha^T$ is the solution of (3), it is the solution of (MGDA-UB) as well. Hence, both formulations completely agree on Pareto stationarity.

In order to prove the second case, we need to show that the resulting descent direction computed by solving (MGDA-UB) does not increase any of the loss functions. Formally, we need to show that

$$\left(\sum_{t=1}^{T} \alpha^t \nabla_{\theta^{sh}} \hat{\mathcal{L}}^t \right)^{\intercal} \left(\nabla_{\theta^{sh}} \hat{\mathcal{L}}^{t'} \right) \geq 0 \quad \forall\, t' \in \{1, \ldots, T\} \tag{8}$$

This condition is equivalent to

$$\left(\sum_{t=1}^{T} \alpha^t \nabla_{\mathbf{Z}} \hat{\mathcal{L}}^t \right)^{\intercal} \mathbf{M} \left(\nabla_{\mathbf{Z}} \hat{\mathcal{L}}^{t'} \right) \geq 0 \quad \forall\, t' \in \{1, \ldots, T\} \tag{9}$$

where $\mathbf{M} = \left(\frac{\partial \mathbf{Z}}{\partial \theta^{sh}}\right)^{\intercal} \left(\frac{\partial \mathbf{Z}}{\partial \theta^{sh}}\right)$. Since $\mathbf{M}$ is positive definite (following the assumption), this is further equivalent to

$$\left(\sum_{t=1}^{T} \alpha^t \nabla_{\mathbf{Z}} \hat{\mathcal{L}}^t \right)^{\intercal} \left(\nabla_{\mathbf{Z}} \hat{\mathcal{L}}^{t'} \right) \geq 0 \quad \forall\, t' \in \{1, \ldots, T\} \tag{10}$$

We show that this follows from the optimality conditions for (MGDA-UB). The Lagrangian of (MGDA-UB) is

$$\left(\sum_{t=1}^{T} \alpha^t \nabla_{\mathbf{Z}} \hat{\mathcal{L}}^t \right)^{\intercal} \left(\sum_{t=1}^{T} \alpha^t \nabla_{\mathbf{Z}} \hat{\mathcal{L}}^t \right) - \lambda \left(\sum_i \alpha^i - 1 \right) \text{ where } \lambda \geq 0. \tag{11}$$

The KKT condition for this Lagrangian yields the desired result as

$$\left(\sum_{t=1}^{T} \alpha^t \nabla_{\mathbf{z}} \hat{\mathcal{L}}^t\right)^{\intercal} \left(\nabla_{\mathbf{z}} \hat{\mathcal{L}}^t\right) = \frac{\lambda}{2} \geq 0 \tag{12}$$

□

B Additional Results on Multi-label Classification

In this section, we present the experimental results we did not include in the main text.

In the main text, we plotted a radar chart of the binary attribute classification errors. However, we did not include the tabulated results due to the space limitations. Here we list the binary classification error of each attribute for each algorithm in Table 5.

Table 5: Multi-label classification error per attribute for all algorithms.

	Uniform scaling	Single task	Kendall et al.	Grad Norm	Ours		Uniform scaling	Single task	Kendall et al.	Grad Norm	Ours
Attr. 0	7.11	7.16	7.18	6.54	**6.17**	Attr. 5	4.91	4.75	4.95	4.19	**4.13**
Attr. 1	17.30	**14.38**	16.77	14.80	14.87	Attr. 6	20.97	14.24	15.17	**14.07**	14.08
Attr. 2	20.99	19.25	20.56	18.97	**18.35**	Attr. 7	18.53	17.74	18.84	17.33	**17.25**
Attr. 3	17.82	16.79	18.45	16.47	**16.06**	Attr. 8	10.22	8.87	10.19	8.67	**8.42**
Attr. 4	1.25	1.20	1.17	1.13	**1.08**	Attr. 9	5.29	5.09	5.44	4.68	**4.60**
Attr. 10	4.14	4.02	4.33	3.77	**3.60**	Attr. 15	0.81	**0.52**	0.62	0.56	0.56
Attr. 11	16.22	15.34	16.64	14.73	**14.56**	Attr. 16	4.00	3.94	3.99	3.72	**3.46**
Attr. 12	8.42	7.68	8.85	**7.23**	7.41	Attr. 17	2.39	2.66	2.35	**2.09**	2.16
Attr. 13	5.17	5.15	5.26	4.75	**4.52**	Attr. 18	8.79	9.01	8.84	8.00	**7.83**
Attr. 14	4.14	4.13	4.17	3.73	**3.54**	Attr. 19	13.78	12.27	13.86	11.79	**11.29**
Attr. 20	1.61	1.61	1.58	**1.42**	1.43	Attr. 25	27.59	24.82	26.94	24.26	**23.87**
Attr. 21	7.18	**6.20**	7.73	6.91	6.26	Attr. 26	3.54	3.40	3.78	3.22	**3.16**
Attr. 22	4.38	4.14	4.08	3.88	**3.81**	Attr. 27	26.74	22.74	26.21	23.12	**22.45**
Attr. 23	8.32	6.57	8.80	6.54	**6.47**	Attr. 28	6.14	5.82	6.17	5.43	**5.16**
Attr. 24	5.01	5.38	5.12	4.63	**4.23**	Attr. 29	5.55	5.18	5.40	5.13	**4.87**
Attr. 30	3.29	3.79	3.24	**2.94**	3.03	Attr. 35	1.15	1.13	1.08	**0.94**	1.08
Attr. 31	8.05	7.18	8.40	7.21	**6.92**	Attr. 36	7.91	7.56	8.06	7.47	**7.18**
Attr. 32	18.21	17.25	18.15	**15.93**	**15.93**	Attr. 37	13.27	11.90	13.47	11.61	**11.19**
Attr. 33	16.53	15.55	16.19	13.93	**13.80**	Attr. 38	3.80	**3.29**	4.04	3.57	3.51
Attr. 34	11.12	9.76	11.46	10.17	**9.73**	Attr. 39	13.25	13.40	13.78	12.26	**11.95**

C Implementation Details

C.1 MultiMNIST

We use the MultiMNIST dataset, which overlays multiple images together (Sabour et al., 2017). For each image, a different one is chosen uniformly in random. One of these images is placed at the top-left and the other at the bottom-right. We show sample MultiMNIST images in Figure 6.

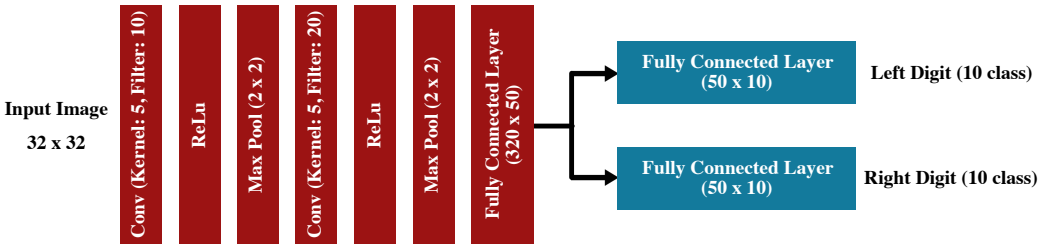

Figure 5: Architecture used for MultiMNIST experiments.

For the MultiMNIST experiments, we use an architecture based on LeNet (LeCun et al., 1998). We use all layers except the final one as a shared encoder. We use the fully-connected layer as a task-specific function for the left and right tasks by simply adding two independent fully-connected layers, each taking the output of the shared encoder as input. As a task-specific loss function, we use the cross-entropy loss with a softmax for both tasks. The architecture is visualized in Figure 5.

The implementation uses PyTorch (Paszke et al., 2017). For all baselines, we searched over the set $LR = \{1\text{e}{-4}, 5\text{e}{-4}, 1\text{e}{-3}, 5\text{e}{-3}, 1\text{e}{-2}, 5\text{e}{-2}\}$ of learning rates and chose the model with the highest validation accuracy. We used SGD with momentum, halving the learning rate every 30 epochs. We use batch size 256 and train for 100 epochs. We report test accuracy.

C.2 Multi-label classification

For multi-label classification experiments, we use ResNet-18 (He et al., 2016) without the final layer as a shared representation function. Since there are 40 attributes, we add 40 separate 2048×2 dimensional fully-connected layers as task-specific functions. The final two-dimensional output is passed through a 2-class softmax to get binary attribute classification probabilities. We use cross-entropy as a task-specific loss. The architecture is visualized in Figure 7.

The implementation uses PyTorch (Paszke et al., 2017). We resize each CelebA image (Liu et al., 2015b) to $64 \times 64 \times 3$. For all experiments, we searched over the set $LR = \{1\text{e}{-4}, 5\text{e}{-4}, 1\text{e}{-3}, 5\text{e}{-3}, 1\text{e}{-2}, 5\text{e}{-2}\}$ of learning rates and chose the model with the highest validation accuracy. We used SGD with momentum, halving the learning rate every 30 epochs. We use batch size 256 and train for 100 epochs. We report attribute-wise binary accuracies on the test set as well as the average accuracy.

C.3 Scene understanding

For scene understanding experiments, we use the Cityscapes dataset (Cordts et al., 2016). We resize all images to resolution 256×512 for computational efficiency. As a shared representation function (encoder), we use the ResNet-50 architecture (He et al., 2016) in fully-convolutional fashion. We take the ResNet-50 architecture and only use layers prior to average pooling that are fully convolutional. As a decoder, we use the pyramid pooling module (Zhao et al., 2017) and set the output sizes to $256 \times 512 \times 19$ for semantic segmentation (19 classes), $256 \times 512 \times 2$ for instance segmentation (one output channel for the x-offset of the center location and another channel for the y-offset), and $256 \times 512 \times 1$ for monocular depth estimation. For instance segmentation, we use the proxy task of estimating the offset for the center location of the instance that encompasses the pixel. We directly estimate disparity instead of depth and later convert it to depth using the provided camera intrinsics. As a loss function, we use cross-entropy with a softmax for semantic segmentation, and MSE for depth and instance segmentation. We visualize the architecture in Figure 8.

We initialize the encoder with a model pretrained on ImageNet (Deng et al., 2009). We use the implementation of the pyramidal pooling network with bilinear interpolation shared by Zhou et al. (2017a). Ground-truth results for the Cityscapes test set are not publicly available. Therefore, we report numbers on the validation set. As a validation set for hyperparameter search, we randomly choose 275 images from the training set. After the best hyperparameters are chosen, we retrain with the full training set and report the metrics on the Cityscapes validation set, which our algorithm never sees during training or hyperparameter search. As metrics, we use mean intersection over union (mIoU) for semantic segmentation, MSE for instance segmentation, and MSE for disparities (depth estimation). We directly report the metric in the proxy task for instance segmentation instead of performing a further clustering operation. For all experiments, we searched over the set $LR = \{1\text{e}{-4}, 5\text{e}{-4}, 1\text{e}{-3}, 5\text{e}{-3}, 1\text{e}{-2}, 5\text{e}{-2}\}$ of learning rates and chose the model with the highest

Figure 6: Sample MultiMNIST images. In each image, one task (task-L) is classifying the digit on the top-left and the second task (task-R) is classifying the digit on the bottom-right.

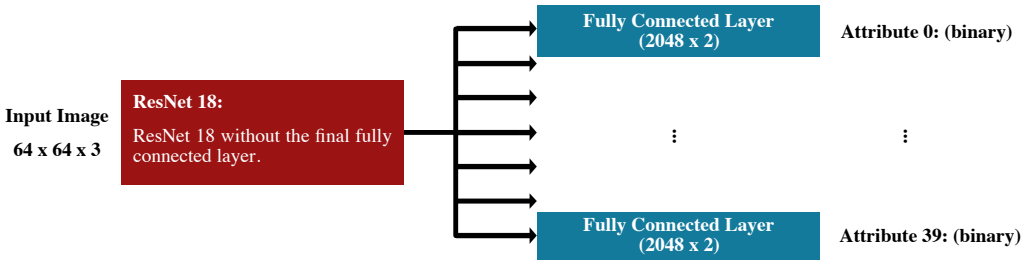

Figure 7: Architecture used for multi-label classification experiments.

validation accuracy. We used SGD with momentum, halving the learning rate every 30 epochs. We use batch size 8 and train for 250 epochs.

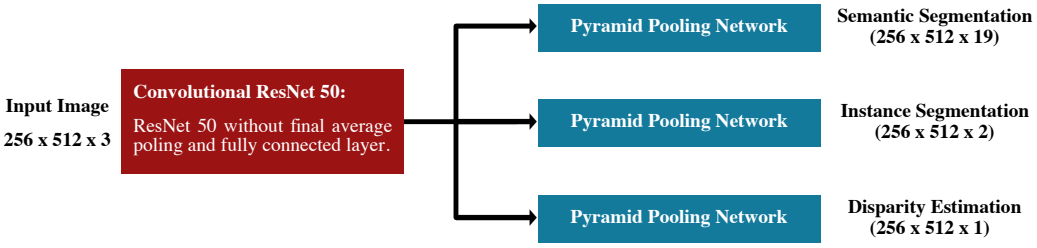

Figure 8: Architecture used for scene understanding experiments.