Can Small Heads Help? Understanding and Improving Multi-Task Generalization

Yuyan Wang, Zhe Zhao, Bo Dai, Christopher Fifty, Dong Lin, Lichan Hong, Li Wei, Ed H. Chi
{yuyanw,zhezhao,bodai,cfifty,dongl,lichan,liwei,edchi}@google.com
Google Research
USA

ABSTRACT

Multi-task learning aims to solve multiple machine learning tasks at the same time, with good solutions being both generalizable and Pareto optimal. A multi-task deep learning model consists of a shared representation learned to capture task commonalities, and task-specific sub-networks capturing the specificities of each task. In this work, we offer insights on the under-explored trade-off between minimizing task training conflicts in multi-task learning and improving *multi-task generalization*, i.e. the generalization capability of the shared presentation across all tasks. The trade-off can be viewed as the tension between multi-objective optimization and shared representation learning: As a multi-objective optimization problem, sufficient parameterization is needed for mitigating task conflicts in a constrained solution space; However, from a representation learning perspective, over-parameterizing the task-specific sub-networks may give the model too many "degrees of freedom" and impedes the generalizability of the shared representation.

Specifically, we first present insights on understanding the parameterization effect of multi-task deep learning models and empirically show that larger models are not necessarily better in terms of multi-task generalization. A delicate balance between mitigating task training conflicts vs. improving generalizability of the shared presentation learning is needed to achieve optimal performance across multiple tasks. Motivated by our findings, we then propose the use of a *under-parameterized self-auxiliary head* alongside each task-specific sub-network during training, which automatically balances the aforementioned trade-off. As the auxiliary heads are small in size and are discarded during inference time, the proposed method incurs minimal training cost and no additional serving cost. We conduct experiments with the proposed self-auxiliaries on two public datasets and live experiments on one of the largest industrial recommendation platforms serving billions of users. The results demonstrate the effectiveness of the proposed method in improving the predictive performance across multiple tasks in multi-task models.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**; **Machine learning algorithms**; **Multi-task learning**.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04.
https://doi.org/10.1145/3485447.3512021

KEYWORDS

neural networks, multi-task learning, Pareto frontier, auxiliary tasks

ACM Reference Format:
Yuyan Wang, Zhe Zhao, Bo Dai, Christopher Fifty, Dong Lin, Lichan Hong, Li Wei, Ed H. Chi. 2022. Can Small Heads Help? Understanding and Improving Multi-Task Generalization. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512021

1 INTRODUCTION

In many machine learning applications, there is more than one task that is of interest. For example, an object detection algorithm may involve predicting both the class of and position of an object [20]; a content recommendation system may optimize for short-term conversion rates as well as long-term satisfaction of its users [55]. These use cases require the prediction of multiple objectives given a common set of features, a problem multi-task learning is well suited to solve [7]. Over the past years, multi-task deep learning has gained popularity through its success in a wide range of applications, including natural language processing [11], computer vision [20, 41], and online recommender systems [4, 33, 34].

A multi-task learning problem usually entails learning from a parameterized model class that shares a subset of parameters across different tasks [42]. The benefits of such a shared architecture are numerous. First, it exploits task relatedness with inductive bias learning [6, 7]. Assuming that tasks share a common hypothesis class, learning a shared representation across tasks is beneficial especially for harder tasks or tasks with limited training examples [31, 35]. Second, by forcing tasks to share model capacity, it introduces a regularization effect. Third, it offers a highly compact and efficient form of modeling which better enables training and serving multiple prediction quantities for large-scale online systems.

At the same time, multi-task learning brings new challenges as it often suffers from task training conflicts. With a single model optimizing multiple objectives that come from potentially conflicting tasks, it is unlikely that all objectives would achieve optimality at the same time [34, 45]. In other words, multi-task learning comes with inherent trade-offs in the performance across different tasks. Given a fixed model capacity, task conflicts can be quantified by computing the per-task training loss in a multi-task learning paradigm and comparing these losses with their respective single-task counterparts. The larger the difference in loss between multi-task and single-task models, the more task conflicts there are [49]. In the rest of the paper, we interchangeably use the terms "task training conflicts" and "task conflicts". Recent research has focused on models and algorithms designed to reduce task training conflicts and improve multi-task performance [32, 46, 47]. Other work has

focused on designing flexible model architecture [42] and efficient optimization algorithms [9, 45].

Large models have attracted a lot of popularity in the past years with its success in numerous applications [15, 48]. Are larger models, especially larger task-specific sub-networks, always better in multi-task learning scenarios? We start with understanding this parameterization effect and point out the largely ignored trade-off between minimizing task training conflicts and improving multi-task generalization. Specifically, larger task-specific sub-networks are not always better. On one hand, similar to [49], we empirically show that by increasing parameterization, multiple training objectives can have less conflicts, which coincides with multi-objective learning theories [43] that sufficient parameterization is needed for properly handling task conflicts in a constrained solution space. However, on the other hand, we find that over-parameterizing task-specific networks can yield solutions which are worse than those derived from smaller models. This is because over-parameterization diminishes the benefit of inductive transfer from multi-task deep learning and can lead to poor per-task performance. In other words, while larger task-specific networks have more flexibility in mitigating task conflicts, they also detract from good multi-task generalization. Note that the *multi-task generalization* we discuss here is different from the traditional notion of generalization: It is specifically in the context of multi-task learning and refers to the generalizability of the learned representation across multiple tasks.

Motivated by the findings, we then propose a simple yet effective approach to automatically balance the trade-off between minimizing task training conflicts and improving multi-task generalization. It can be viewed as an effort to improve the generalizability of the shared representation learning (referred to as *multi-task generalization* in the rest of the work). Specifically, we propose to use *under-parameterized self-auxiliaries*, which are small-capacity sub-networks, alongside the large task-specific sub-networks during training. They are effectively copies of the main task-specific sub-networks and learn the same tasks, but with much lower capacity. They are discarded during inference. So the proposed method only incurs minimal additional training cost and no additional serving cost. We find these under-parameterized self-auxiliaries introduces an *implicit regularization* effect that improves the generalizability of the shared representation learning. Empirical results validate our proposed approach on benchmark datasets and an industrial recommendation platform.

To summarize, our contributions are

- **Understanding:** We provide insights on the under-explored trade-off in multi-task learning, which is the tension between minimizing task training conflicts and improving multi-task generalization. Specifically, over-parameterizing task-specific sub-networks may impedes the generalizability of shared representation learning.
- **Improvements:** We propose the use of *under-parameterized self-auxiliaries* to automatically balance the trade-off via implicitly regularizing the shared representation learning.
- **Validation:** Through experiments on two benchmark datasets and an industrial recommendation platform, we validate the efficacy of the proposed approach in improving multi-task generalization and therefore the performance across all tasks.

2 RELATED WORK

Multi-task learning as multi-objective optimization. Minimizing task losses in multi-task learning can be formulated as a multi-objective optimization problem [45], in which the notion of Pareto optimality was first proposed and studied [43]. In addition to a linear weighting of task losses which is commonly used for multi-task learning problems, examples of other multi-objective optimization methods include constraint methods, goal programming [24], exponential weighted sum [3, 51], population methods [44], preference elicitation [12], and many more [26, 37]. There is also research on multi-objective optimization methods where the objectives are nonconvex [39] or the Pareto frontier is nonconvex [19].

Despite the close relationship between multi-task learning and multi-objective optimization, they also have significant differences. For example, multi-objective optimization barely looks into non-convex optimization of deep neural networks [53], which is a main challenge for multi-task learning problems [9]. An example of the recent works [28, 45] toward bridging this gap is the application of the multiple-gradient descent algorithm [14] to multi-task learning, which is a gradient-based multi-objective optimization method.

Inspired by these explorations, our work starts by empirically investigating the Pareto frontiers of multi-task learning problems. We find that trade-off is a function of parameterization as it lead to different training and generalization difficulties. This is rarely discussed in multi-objective optimization literature. Based on our understanding, we then propose a simple yet effective treatment to *automatically* balance the task conflict mitigation and the generalization benefits from learning multiple tasks jointly.

Improving Pareto efficiency for multi-task deep learning. Recent research on reducing task training conflicts and improving per-task performance for multi-task deep learning can be grouped into three lines of efforts. The first line aims to develop flexible parameter sharing in model architectures. Examples include soft parameter-sharing that encourage more sharing for similar tasks and less for conflicting tasks [22, 34, 38], adaptively deciding which layers to share during the training process [32, 47], or on a macro level, deciding which tasks should be learned together [46]. The second line of research focuses on improving optimization algorithms to better traverse the loss surface. These works mainly focus on adaptive linear weighting approaches [9, 16, 25, 52] that find better solutions than a naive linear weighting method. The third line of research adds auxiliary tasks to enhance the performance of one or more primary tasks. It has been widely adapted to computer vision [54], natural language processing [2] and information retrieval [29]. If related tasks are unavailable, auxiliary tasks can also be constructed using adversarial loss [18], predicting inputs or past labels [7, 8], pseudo-task augmentation [36] or learning representations [40].

Our proposed under-parameterized self-auxiliaries can be viewed as a special case of auxiliary task learning. However, unlike its typical formulation, our method does not require specific domain knowledge on designing auxiliary tasks. In particular, the auxiliary tasks in our case are self-auxiliaries. They operate on the *same tasks* but with *different parameterizations*. A similar concept manifests in knowledge distillation [1, 23], but our method differs from distillation in multi-task learning [30]. Instead of using a smaller

(student) network to learn the predictions of the larger (teacher) network, under-parameterized self-auxiliaries learn concurrently with their primary (and larger) counterparts, collaboratively building a generalizable feature representation within the shared layers.

3 UNDERSTANDING

Suppose there are T tasks sharing an input space $\mathcal{X}$. Each task has its own task space $\{\mathcal{Y}^t\}_{i=1}^T$. A dataset of n i.i.d. examples from the input and task spaces is given by $\{(x_i, y_i^1, ..., y_i^T)\}_{i=1}^n$, where y_i^t is the label of the t-th task for example i. We assume a multi-task model parameterized by $\theta \in \Theta$. $\theta = (\theta_{sh}, \theta_1, ..., \theta_T)$ includes shared-parameters θ_{sh} and task-specific parameters $\theta_1, ..., \theta_T$. Let $f_t(\cdot, \cdot) : \mathcal{X} \times \Theta \rightarrow \mathcal{Y}^t$ be the model function and $\mathcal{L}_t(\cdot, \cdot) : \mathcal{Y}^t \times \mathcal{Y}^t \rightarrow \mathbb{R}^+$ be the loss function for the t-th task. This formulation also includes the more general multi-task learning setting where different tasks have different inputs, in which case $x_i = (x_i^1, ..., x_i^T)^T$ where x_i^t is the input of the t-th task for example i.

Let $\hat{\mathcal{L}}_t(\theta) := \frac{1}{n} \sum_{i=1}^n \mathcal{L}_t(f_t(x_i; \theta_{sh}, \theta_t), y_i^t)$ be the empirical loss for the t-th task, where we drop the dependency on x and y for ease of notation. The optimization for multi-task learning can then be formulated as a joint optimization of a vector-valued loss function:

$$\min_{\theta} (\hat{\mathcal{L}}_1(\theta), ..., \hat{\mathcal{L}}_T(\theta))^\top. \tag{1}$$

It is unlikely that a single θ optimizes all objectives simultaneously. The solution to (1) is therefore a set of points which represent different trade-off preferences. Formally, solution θ_a is said to dominate solution θ_b if $\hat{\mathcal{L}}_t(\theta_a) \leq \hat{\mathcal{L}}_t(\theta_b), \forall t$ and there exists at least one task j such that the inequality is strict. A solution θ is called *Pareto optimal* if there is no solution $\theta' \neq \theta$ such that θ' dominates θ. *Pareto frontier* is the set of all Pareto optimal solutions.

For linear weighting method, the minimization objective is a scalarization of the empirical loss vector $\hat{\mathcal{L}}(\theta) := \sum_{t=1}^T w_t \hat{\mathcal{L}}_t(\theta)$, where $\{w_t\}_{t \in \{1,...,T\}}$ are weights for individual tasks. Note that, although being a popular choice for most existing multi-task learning algorithms, linear weighting methods can only identify solutions of (1) that are in the *convex* region of the Pareto frontier.

Benefits from large task-specific sub-networks. In this section, we present two major reasons for using large multi-task deep learning models, namely: providing justification for the use of linear weighting method; and effectiveness in the mitigation of task training conflicts.

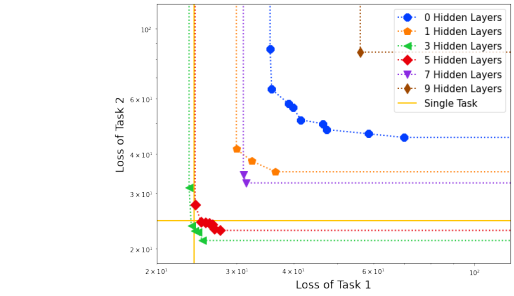
Linear weighting method is known to only be able to obtain solutions that lie on the convex regions of the Pareto frontier [26]. Therefore, we first look into the convexity of the Pareto frontier for multi-task models. Notably, when all objectives are convex in their respective parameters, the Pareto frontier is guaranteed to be convex.

Proposition 1. *Suppose $\mathcal{L}_t(\theta)$ is convex and continuous in θ for all tasks $t \in \{1, ..., T\}$ and Θ is convex. Then the Pareto frontier of $(\hat{\mathcal{L}}_1(\theta), ..., \hat{\mathcal{L}}_T(\theta))^\top$ in problem (1) is convex.*

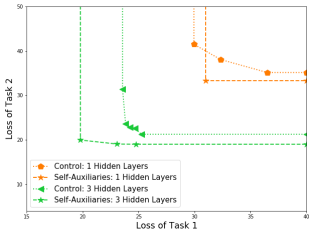
When some or all objectives are nonconvex, we find that over-parameterized multi-task deep learning models imply convex or nearly-convex Pareto frontiers. Details for the discussion as well as the proof for Proposition 1 can be found in Appendix A.1, which provides justification for using linear weighting methods for over-parameterized multi-task learning models.

Another benefit from large multi-task models, and in particular large task-specific sub-networks, is their flexibility in dealing with task conflicts. As a multi-objective optimization problem, conflicts are reflected as the competition among tasks over limited model capacity. Large models therefore enable better handling of task conflicts, as it offers larger solution spaces.

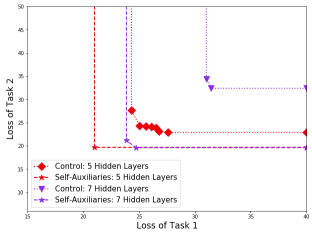
Challenges inherent in large task-specific sub-networks. To understand if larger task-specific networks lead to better multi-task performance, we perform a series of studies on synthetic datasets. Similar to the setup in Finn et al. [17] and Ma et al. [34], we generate a multi-task dataset and define each task as a regression from the input to the output of a combination of sine waves. To introduce task conflicts together with task relatedness, we let the two tasks share a small subset of frequencies and a shared-bottom architecture [42] with task-specific sub-networks is used. A full description of the synthetic dataset and model architectures is available in Appendix A.2. Figure 1a shows that, as the number of hidden layers increases from 0 to 5, the Pareto frontier on test data also improves; Surprisingly, however, as more hidden layers are added (from 5 to 9), the Pareto frontier rapidly deteriorates. We also observe similar trends when increasing the number of shared layers, the number of both shared layers and task-specific layers, or varying layer capacity rather than depth.



(a) Baseline.



(b) 1-3 hidden layers.



(c) 5-7 hidden layers.

Figure 1: Pareto frontiers on synthetic data. (a): Baseline Pareto frontiers with increasing model capacity. The best single task performance across all models is also reported. (b)-(c): Comparison of our method with baselines on different model capacities.

This intriguing observation touches upon the largely ignored trade-off between minimizing task training conflicts and improving multi-task generalization. Multi-objective optimization theory suggests that sufficient model capacity is needed to be able to deal

with task conflicts. However, treating multi-task learning as a multi-objective optimization problem is limiting. Multi-task learning is a more general problem as it leverages parameter sharing and inductive transfer [6] which benefit the generalizability of the learned shared representations. Over-parameterization inevitably undermines the benefit of sharing, which may hurt *multi-task generalization* and ultimately backfire.

4 METHOD

To summarize our insights, for multi-task learning, solutions learned by small task-specific sub-networks generalize well to multiple tasks but suffer from task training conflicts. Large task-specific sub-networks are able to better mitigate those training conflicts, but suffer from loss of multi-task generalization. These observations motivate our design of an *automatic* treatment towards achieving the best of both worlds.

We propose to use *under-parameterized self-auxiliaries* to *automatically* balance task conflict mitigation and generalization for multi-task deep learning models. By adding an under-parameterized small tower for each task in a large multi-task model, we hope to enjoy the benefits of both large and small task-specific sub-networks.

We present the proposed approach under the popular multi-task architecture, which consists of a representation shared by all tasks with task-specific layers built on top of the shared representation. The model family $\mathcal{H}_t$ for each task t is represented as:

$$f_t(x; \theta_{sh}, \theta_t) = f_t(h(x; \theta_{sh}); \theta_t), \forall t, \tag{2}$$

where $h(\cdot;\cdot) : \mathcal{X} \times \Theta \rightarrow \mathcal{R}^M$ is the shared representation.

Now we construct a self-auxiliary tower for every task with the same task labels but a *different* parameterization with significantly less capacity (Figure 2):

$$f_t^a(x; \theta_{sh}, \theta_t^a) = f_t(h(x; \theta_{sh}); \theta_t^a), \forall t, \tag{3}$$

where the superscript stands for auxiliary and θ_t^a is much smaller in size than the original task-specific sub-network θ_t. The empirical loss is defined as

$$\hat{\mathcal{L}}(\theta) = \sum_{t=1}^{T} w_t(\hat{\mathcal{L}}_t(\theta_{sh}, \theta_t) + \gamma \hat{\mathcal{L}}_t(\theta_{sh}, \theta_t^a)) \tag{4}$$

where w_t is the weight for task t, and

$$\hat{\mathcal{L}}_t(\theta_{sh}, \theta_t^a) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_t(f_t(x_i; \theta_{sh}, \theta_t^a), y_i^t) \tag{5}$$

is the loss for task t's self-auxiliary, and $\gamma > 0$ controls the weight of the auxiliary loss. At inference time, the self-auxiliaries are *discarded* and *only* $f_t(\cdot; \theta_{sh}, \theta_t)$ is used as task t's prediction. Because the self-auxiliaries are small in size and only used in training, they incur minimal additional training cost and *no extra cost* at serving time.

On the synthetic datasets in Section 3, under-parameterized self-auxiliaries significantly improve the Pareto frontier on the test dataset for all levels of parameterization (Figure 1b and 1c). Moreover, the improvement is greater for larger task sub-networks, which aligns with our insights that small auxiliary heads improve the generalization of larger models.

Why do small heads help? The fact that under-parameterized self-auxiliaries improve multi-task generalization is not surprising to us. In large models, what can be shared may also be captured well in the large task-specific towers with equal or even smaller training losses; as a consequence, multi-task generalization and the benefits of sharing are likely sacrificed.

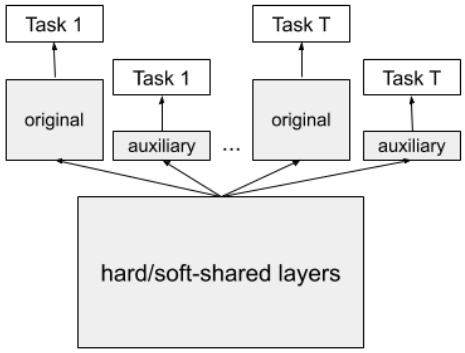


Figure 2: An illustration of under-parameterized self-auxiliaries for multi-task learning.

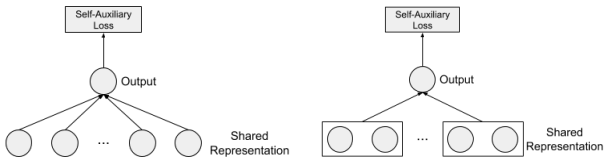
However, adding small towers to large multi-task models shifts the paradigm. By simultaneously training the same task with two towers – once with the full parameterization and the other with the under-parameterization – the shared representation of the multi-task model $h(\cdot; \theta_{sh})$ is "forced" toward learning a representation which suits both fully parameterized and under-parameterized task-specific transformations.

In other words, when a multi-task model has sufficient capacity, it has the "freedom" to allocate the information to either the shared representation $h(\cdot; \theta_{sh})$ or task-specific towers $f_t(\cdot; \theta_t)$. The way it allocates the information *trades off* the model's ability to mitigate task conflicts with the multi-task generalization benefits derived from sharing representations. The proposed under-parameterized self-auxiliaries act as *implicit regularization* in that sharing happens in the shared layers as much as possible. And the original task towers effectively have *more* capacity to learn the task specifics and mitigate conflicts better.

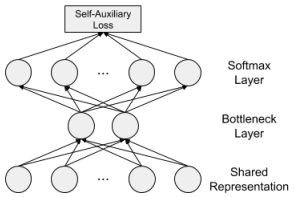
We point out that self-auxiliaries share similar form as pseudo-task augmentation (PTA) by Meyerson and Miikkulainen [36], in which models are trained with multiple towers for each task, and is shown to benefit both single-task and multi-task settings. PTA and other task augmentation methods are theoretically supported by Baxter [5] which shows that additional tasks can lead to implicit data augmentation and better generalization. Our proposed use of *under-parameterized self-auxiliaries*, however, relies more on the implicit regularization brought by the small towers which leads to a better learning dynamics of the shared representations. Unlike PTA which trains multiple towers and significantly increases parameterization, under-parameterized self-auxiliaries incurs very little additional parameterization while efficiently balancing the mitigation of task conflicts with generalization improvements. In the ablation studies in Section 5.3, we confirm the need of under-parameterization for self-auxiliaries to improve multi-task generalization.

Using under-parameterized self-auxiliaries is simple: any tower $f_t^a(x; \theta_{sh}, \theta_t^a)$ that is significantly smaller than the original tower

$f_t(x; \theta_{sh}, \theta_t)$ would work. In this sense our approach is nearly model-agnostic as it is general, adaptive, and can be applied to any model architecture. For example, one can simply use a single fully-connected layer over the shared representation $h(x; \theta_{sh})$ as the self-auxiliary tower (Figure 3a). If the dimension of the shared layer output M is big, we can further reduce the parameterization through pooling (Figure 3b). For multi-class classification tasks, the final layer is a softmax layer with the size equal to the number of classes C. In this case, a single fully-connected layer as self-auxiliaries introduce CM additional parameters, which could be a considerable amount if both C and M are large as in many multi-task applications. We can instead let the self-auxiliaries be a two-layer tower with a bottleneck layer of size $b \ll M, C$ (Figure 3c), in this case the number of additional parameters will be $O(\max(C, M))$ instead of $O(CM)$.



(a) Single fully-connected layer. **(b)** Pooling.



(c) Bottleneck layer.

Figure 3: Example architectures for under-parameterized self-auxiliaries. (a): Single fully-connected layer. (b): Single layer with average pooling. (c): Two-layer tower with bottleneck layer.

It is worth noting that the proposed method of under-parameterized self-auxiliaries for multi-task learning can be combined with prior multi-task learning augmentations such as uncertainty weighting [25], gradient surgery [52] and multi-objective optimization algorithms [28, 45].

5 EXPERIMENTS

We start with describing the setup of the experiments including datasets, architectures and tuning etc. Then we present the results in multi-task scenarios which cover regression and classification tasks in computer vision and recommender systems, including live experiments on an industrial content recommendation platform. In the end we present ablation studies to test the need of "small heads" as self-auxiliaries and provide more insights into the improved multi-task generalization as a result. The link to code is provided in Appendix A.4.

5.1 Experiment Setup

5.1.1 Datasets, architecture and tuning. We introduce the datasets and the model architecture used for each application below. More details can be found in Appendix.

MultiMNIST and MultiFashionMNIST: The MultiMNIST and MultiFashionMNIST datasets are created by extending the MNIST [27] and FashionMNIST [50] to a multi-task setup [28, 45]. Two 32×32 images are chosen at random from MNIST/FashionMNIST. Then one is put at the top-left corner and the other is at bottom-right, overlapping each other with a vertical and horizontal stride of 4 pixels. The multi-task learning problem is to classify the digit on the top-left (task 1) and bottom-right (task 2) for each combined image. We adopt the LeNet architecture [27] as the multi-task model. We consider *three different model sizes*, with an increasing number of shared hidden layers and task-specific hidden layers (Figure 11 in Appendix). The architecture for the self-auxiliary head is as in Fig. 3a. Details on the architecture and hyperparameters can be found in Appendix A.3. Training and test data are randomly split at 80/20. For each method and every model size, we perform 1000 runs for hyperparameters tuning. With the selected hyperparameters, we then perform another 1000 runs with task weights w_t varying from 0 to 1, evaluate each of them on the test dataset, plot the Pareto frontier (i.e. the Pareto optimal solutions from the 1000 runs).

MovieLens: The MovieLens 1M dataset[1] [21] records 1 million ratings from 6000 users on 4000 movies. To formulate a multi-task learning problem, we first augment the dataset by randomly sampling movies for every user that do not have ratings in the original dataset, and label those examples as un-watched movies for that user. For every user, the number of un-watched movies is the same as the number of her watched movies, with leaves us with 2 million examples. We then sample 1.6 million examples from the augmented dataset as training data, and another 0.2 million examples as test data. For every user and movie pair, we construct a binary classification task to predict whether the user watches the movie (task 1), and a regression task to predict the user's rating (1-5) on the movie as a float value (task 2). The design of the tasks as well as the model architecture is similar to what is described in a real-world large-scale recommendation system [13]. Each layer is of size 200 with ReLU activation. Adagrad optimizer with batch size of 100 is used. We also experiment with average pooling (Figure 3b) on the last shared hidden layer as the input for the self-auxiliary towers. The performance of the tasks is measured by computing the error rate for watch prediction and mean squared error (MSE) for rating prediction on the test dataset. For each baseline method, we perform 1000 runs to search for the best learning rate; for our method of self-auxiliaries, we perform 1000 runs to search for the best combination of learning rate, self-auxiliary weight γ, and self-auxiliary pool length (Figure 3b). We fix the number of tuning runs for each method to make sure that we are not over-tuning for our method which has more hyperparameters. We then perform another 1000 runs with the task loss weights varying between 0 and 1, evaluate each of them on the test dataset, and plot the Pareto frontier of the test metrics.

[1] https://grouplens.org/datasets/movielens/1m/

An industrial content recommendation platform: We conduct live experiments on one of the largest industrial content recommendation platforms serving billions of users. For every user who is currently consuming contents on the platform, a recommendation of a list of contents to be consumed next is shown to her. The recommendations are generated from a pool of hundreds of millions of contents on the platform, based on the user's past interaction with the platform as well as content features and contextual features. The modeling framework for the content recommendation system is similar to the existing two-stage architecture [13], where a candidate generation network is followed by a ranking network. There are 8 tasks in total: 4 tasks predicting user satisfaction-related scores on the platform, which are denoted as S_1, S_2, S_3, S_4; and another 4 tasks predicting user short-term and long-term engagement-related behaviors, which are denoted as E_1, E_2, E_3, E_4.

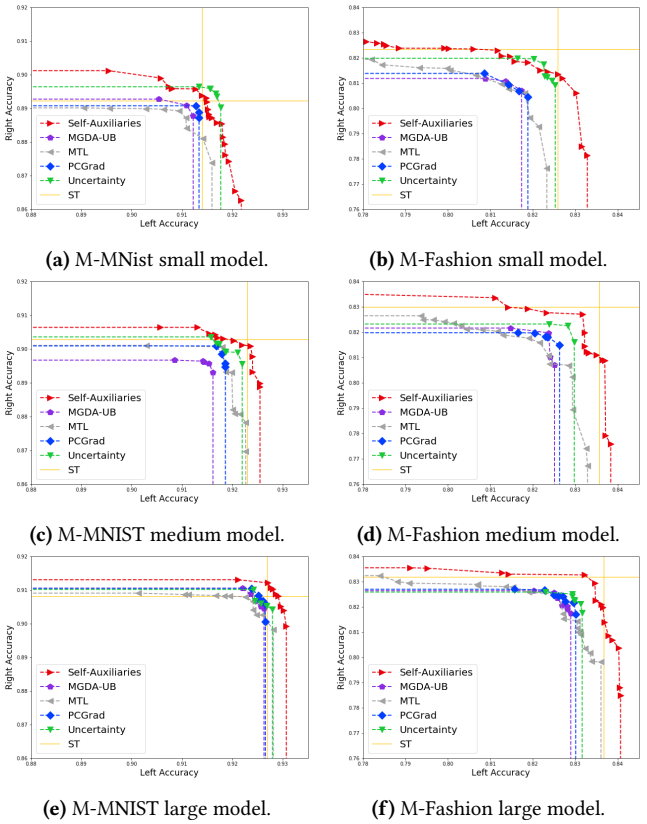
We experiment with the *ranking* network which is a *mixture-of-experts* model [34]. The architecture for the self-auxiliary head is as shown in Fig. 3a. The model is trained continuously with 10% of next day's data as test data. For binary classification tasks, AUC on test data is used as the evaluation metric; for regression tasks, root mean squared error (RMSE) is used.

5.1.2 Baseline methods. We compare our method with state-of-the-art model-agnostic methods for multi-task learning. The methods in experiment are: (1) **self-auxiliaries**: our method of under-parameterized self-auxiliaries; (1) **single task baseline (ST)**: learning each task separately; (2) **linear weighting (MTL)**: linear weighting method $\hat{\mathcal{L}}(\theta) = \sum_{t=1}^{T} w_t \hat{\mathcal{L}}_t(\theta)$ with the weights varying in the simplex $\{w = (w_1, \ldots w_T) | \sum_{t=1}^{T} w_t = 1, w_t \geq 0, \forall t\}$; (3) **uncertainty weighting (Uncertainty)**: learned uncertainty of the tasks are used as loss weights [25]; (4) **multiple-gradient descent algorithm (MGDA-UB)**: a modified multiple-gradient descent algorithm from multi-objective optimization [45]; (5) **gradient surgery (PCGrad)**: a gradient projection method for mitigating task conflicts [52]; (6) **pseudo-task augmentation (PTA-F)**: multiple task-specific sub-networks are trained for the same task [36], and we adopt the variant in which all but one of the sub-networks receive gradient update during training ('**-F**'), which is reported to have best performance on multi-task learning.

5.2 Results

5.2.1 MultiMNIST and MultiFashionMNIST.. Figure 4. The results show that our method achieves similar performance compared with the best baseline method for small models (Figures 4a, 4b), and better than other baselines for medium (Figures 4c, 4d) and large models (Figures 4e, 4f). We note that PTA performs noticeably worse than all baselines on small and medium models and lags behind self-auxiliaries on large models (Figure 6b), therefore we don't report the results here. As a side note, uncertainty reweighting ('Uncertainty') is the strongest baseline which consistently outperforms other baselines in our experiments. Table 4 in Appendix A.3 summarizes numerical comparisons.

We also observe that the larger the model, the greater the improvement our method exhibits over the baselines. As larger models introduce more generalization challenges as discussed in Section 3, these results further suggest the effectiveness of self-auxiliaries in achieving a better trade-off between task conflict mitigation and multi-task generalization.



(a) M-MNist small model. **(b)** M-Fashion small model.

(c) M-MNIST medium model. **(d)** M-Fashion medium model.

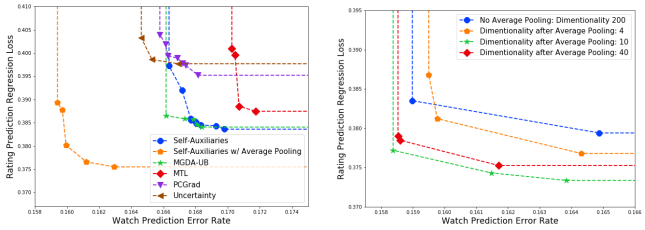
(e) M-MNIST large model. **(f)** M-Fashion large model.

Figure 4: Experiment results on MultiMNIST (M-MNIST) and MultiFashionMNIST (M-Fashion) datasets with different model capacities.

5.2.2 MovieLens. Figure 5a shows the Pareto frontier for the two tasks, indicating that our method with average pooing significantly improves the performance on both tasks. To understand the effectiveness of average pooling, Figure 5b shows the performance of our methods with different input dimensions for the self-auxiliary heads. Numerical results are summarized in Table 1 by reporting the middle points on the Pareto frontier. We find that by reducing the parameterization of self-auxiliary towers with average pooling, we can further improve its performance.

	Watch Error	Rating MSE
MTL	0.172	0.387
Uncertainty	0.165	0.399
MGDA-UB	0.168	0.385
PCGrad	0.167	0.397
Self-Auxiliaries	0.168	0.385
Self-Auxiliaries-pooling	**0.161**	**0.377**

Table 1: Numerical results on MovieLens dataset.



(a) Self-auxiliaries vs. baselines. **(b)** Effect of average pooling.

Figure 5: Experiment results on MovieLens dataset.

5.2.3 An industrial content recommendation platform. In the experiments, we compare both self-auxiliaries and PTA-F against the current production setting (control) where each task is trained with a *single* tower. For self-auxiliaries, we adopt the simple architecture as in Figure 3a and with a neuron dropout rate of 0.2; For PTA-F, we apply two additional towers of the same size as the original task-specific towers.

Offline evaluation results. In Table 2, we report the relative changes of the offline evaluation metrics for all eight tasks at 1 million steps. For AUC metrics, positive changes means better classification accuracy; for RMSE metrics, negative changes means smaller regression error. We see that both PTA-F and our method of under-parameterized self-auxiliaries perform better than the current production model (control) across all tasks. However, our method of self-auxiliaries is able to outperform PTA-F on 4 tasks while PTA-F only out-performs self-auxiliaries on 1 task (Table 2a). We also compute an average AUC across all classification tasks $S_1, S_2, S_3, S_4, E_1, E_2$ and average RMSE across both regression tasks E_3, E_4. Table 2b shows that our method achieves better overall performance on both classification and regression tasks.

Live experiment results. We conduct a series of A/B experiments in the live system serving billions of users to measure the benefits of our approach. A small percentage of user traffic is split into three groups, where the current production model (control), PTA-F and our method serve as the deep ranking model respectively. Experiments are run for three weeks, during which we aggregate user satisfaction metrics and engagement metrics. We track page-specific metrics measured on the page which shows the recommendations, and also site-wide metrics for understanding overall user enjoyment.

Table 3 summarizes the live experiment results. We see while both PTA-F and our method of self-auxiliaries improve *page-specific* satisfaction and engagement metrics where the deep ranking model is at play, our method is able to achieve even bigger improvements than PTA-F. Moreover, our method also improves *site-wide* satisfaction and engagement metrics, all statistically significant at 95% confidence level, while PTA does not.

5.3 Ablation Studies

In this section, we report (1) several ablation studies that confirm the need of small sized heads as self-auxiliaries and (2) understanding the improved generalization through analyzing the shared representation.

Task	Metric Name	PTA-F	Self-Auxiliaries (Ours)
S1	AUC	+0.22%	**+0.55%**
S2	AUC	+0.33%	+0.33%
S3	AUC	+1.13%	**+1.27%**
S4	AUC	+0.12%	+0.12%
E1	AUC	**+0.27%**	+0.14%
E2	AUC	+0.12%	+0.12%
E3	RMSE	-0.00%	**-0.08%**
E4	RMSE	-0.09%	**-0.18%**

(a) Per-task performance.

Metric Name	PTA-F	Self-Auxiliaries (Ours)
Average AUC	+0.351%	**+0.416%**
Average RMSE	-0.072%	**-0.153%**

(b) Average performance for classification (AUC) and regression (RMSE) tasks. Average AUC computes the average AUC over $S_1, S_2, S_3, S_4, E_1, E_2$, and Average RMSE computes the average RMSE over E_3, E_4.

Table 2: Offline evaluation results. Metrics are shown as relative changes compared with the current production model (control) where each task is trained with a single tower. **PTA-F**: 2 additional towers of the same size as original task-specific towers are added for each task; **Self-Auxiliaries**: Self-auxiliaries with neuron dropout of rate 0.2. Metrics are computed on test data and are reported at 1 million training steps.

Metric Name	PTA-F	Self-Auxiliaries (Ours)
Page-specific Satisfaction	+0.15%***	**+0.17%*** **
Site-wide Satisfaction	+0.01%	**+0.06%** **
Page-specific Engagement	+0.13%***	**+0.15%*** **
Site-wide Engagement	0.00%	**+0.05%** **

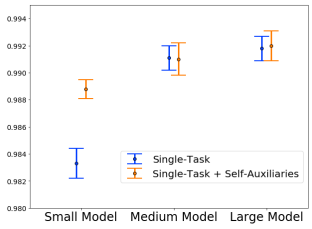
** p-value < 0.05; *** p-value < 0.01.

Table 3: Live experiment results. Metrics are shown in percentage improvement compared with current production model (control).

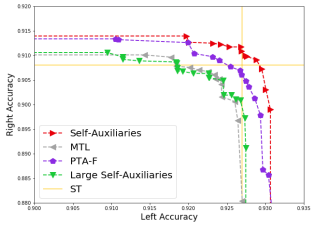
5.3.1 Self-auxiliaries on single-task learning. To understand if self-auxiliaries uniquely help multi-task learning, we conduct experiments on the regular MNIST dataset in a single-task setting, using three different model sizes with an increasing number of fully-connected ReLU layers. Details on the architecture and hyperparameters can be found in Appendix A.3.

Figure 6a shows that self-auxiliaries improve performance on small models, but do *not* significantly influence medium and large models. Compared with Section 5.2.1 above where self-auxiliaries have more advantages on larger models, this confirms that improvements from self-auxiliaries come from improved *multi-task* generalization, instead of benefiting single-task learning.

5.3.2 Large Heads as Self-auxiliaries. We run ablation studies to test the need of "small heads": (1) ablation of self-auxiliaries completely and use a linear weighting of task losses as the training objective ("**MTL**"); (2) large self-auxiliaries with size identical to that of the original task sub-networks ("**Large Self-Auxiliaries**");

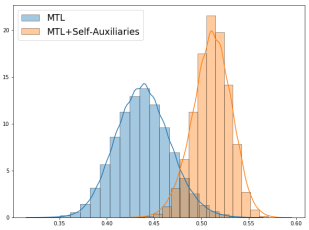


(a) Classification accuracy on MNIST with self-auxiliaries.

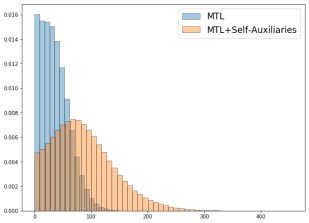


(b) Comparison with big towers as self-auxiliaries.

Figure 6: Ablation studies.



(a) Histogram of nonzero neuron fractions in shared representation.



(b) Histogram of nonzero neuron values in shared representation.

Figure 7: Analysis of shared representation layer on MultiMNIST test dataset.

(3) pseudo-task augmentation (PTA) proposed by Meyerson and Miikkulainen [36], where multiple task-specific sub-networks are trained for the same task, and we adopt the variant in which all but one of the sub-networks receive gradient update during training, which is reported to have best performance on multi-task learning ("**PTA-F**").

Figure 6b shows the Pareto frontier on MultiMNIST, and we can see that the proposed under-parameterized self-auxiliaries (red) wins. This confirms the need for "small heads". It also suggests that the improvement comes from the implicit regularization as discussed in Section 4, in addition to the data augmentation effect as in PTA and other task augmentation methods. The fact that under-parameterized self-auxiliaries does not significantly improve single-task learning (while PTA does) further reflects the difference in our methods. In addition, PTA-F and large self-auxiliaries add significantly more parameters during training. In contrast, under-parameterized self-auxiliaries retain minimal parameterization overhead and little extra training cost, at the same time achieving a better balance between training conflicts and *multi-task generalization*.

5.3.3 Understanding the improved generalization. To gather more insights on the improved multi-task generalization from the self-auxiliaries, we analyze the learning dynamics of the shared model architectures. More specifically, we look at the distribution of the fraction of activated neurons as well as activated neuron values from the shared representation, i.e. $h(\cdot; \theta_{sh})$ in Equation (3), which are believed to correlate with model stability and prediction variations in deep learning models [10].

Figure 7 shows the shared representation of size 100 on the MultiMNIST test data set ($n = 20000$). There is a clear separation in the histograms of layer density (Figure 7a) and activated neuron value distributions (Figure 7b): With self-auxiliaries, the shared representation has more activated neurons on average, and the distribution of those activated values has longer and heavier tails. These analyses shed some light on the improved learning dynamics for shared representation with self-auxiliary towers. With a denser layer and neuron values varying greatly, the shared representation likely lies in a richer subspace induced by $h(\cdot; \theta_{sh})$, which is likely to be able to encode more information across the tasks. Therefore, we believe our observed gains from under-parameterized self-auxiliaries are derived from improving the generalization of the shared representation.

5.4 Discussion

We demonstrate the effectiveness of our proposed method of under-parameterized self-auxiliaries in three multi-task datasets covering different applications including image classification, public recommendation dataset and a real-world large-scale content recommendation platform. Our proposed method is able to improve Pareto efficiency in real-world multi-task problems compared with existing methods, with a negligible training overhead and no additional serving cost. It works well on different model architectures and different types of tasks. The ablation studies suggest the adoption of "small heads" as self-auxiliaries, in addition to shedding some light on the improved multi-task generalization dynamics as a result.

Further, we observe the greatest benefit of our method with high capacity model architectures. As larger models introduce generalization challenges for multi-task learning as discussed in Section 3, this result supports our insights from Section 4 that under-parameterized self-auxiliaries help achieve a better balance between task conflict mitigation and multi-task generalization. We also observe that the performance of self-auxiliaries can be improved by further reducing the parameterization of the self-auxiliaries. Techniques such as average pooling and adding bottleneck layers can be treated as hyperparameters to optimize model performance.

6 CONCLUSION

In this paper, we showed an intriguing trade-off between minimizing task training conflicts and improving *multi-task generalization* in multi-task models. Notably, larger models are not necessarily better than smaller ones in terms of their performance across multiple tasks. Drawing from our findings, we propose the use of under-parameterized self-auxiliaries to automatically balance multi-task generalization with mitigating task conflicts via implicit regularization. By adding small-capacity towers on the same tasks, large models are able to learn a representation that generalizes to multiple tasks better, while maintaining the flexibility and capacity to mitigate task conflicts during training. Experimental results on benchmark datasets and a real-world large-scale content recommendation platform demonstrated the effectiveness of our proposed method in a number of multi-task applications.

REFERENCES

[1] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. 2018. Large scale distributed neural network training

through online distillation. *arXiv preprint arXiv:1804.03235* (2018).
[2] Sercan Ö Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al. 2017. Deep voice: Real-time neural text-to-speech. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 195–204.
[3] Timothy Ward Athan and Panos Y Papalambros. 1996. A note on weighted criteria methods for compromise solutions in multi-objective optimization. *Engineering optimization* 27, 2 (1996), 155–176.
[4] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 107–114.
[5] Jonathan Baxter. 1998. Theoretical models of learning to learn. In *Learning to learn*. Springer, 71–94.
[6] Jonathan Baxter. 2000. A model of inductive bias learning. *Journal of artificial intelligence research* 12 (2000), 149–198.
[7] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.
[8] Rich Caruana and Virginia R De Sa. 1997. Promoting poor features to supervisors: Some inputs work better as outputs. In *Advances in Neural Information Processing Systems*. 389–395.
[9] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2017. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257* (2017).
[10] Zhe Chen, Yuyan Wang, Dong Lin, Derek Zhiyuan Cheng, Lichan Hong, Ed H Chi, and Claire Cui. 2020. Beyond Point Estimate: Inferring Ensemble Prediction Variation from Neuron Activation Strength in Recommender Systems. *arXiv preprint arXiv:2008.07032* (2020).
[11] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. 160–167.
[12] Vincent Conitzer. 2009. Eliciting single-peaked preferences using comparison queries. *Journal of Artificial Intelligence Research* 35 (2009), 161–191.
[13] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.
[14] Jean-Antoine Désidéri. 2012. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique* 350, 5-6 (2012), 313–318.
[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[16] Alexey Dosovitskiy and Josip Djolonga. 2020. You Only Train Once: Loss-Conditional Training of Deep Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HyxY6JHKwr
[17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1126–1135.
[18] Yaroslav Ganin and Victor Lempitsky. 2014. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495* (2014).
[19] A Ghane-Kanafi and E Khorram. 2015. A new scalarization method for finding the efficient frontier in non-convex multi-objective problems. *Applied Mathematical Modelling* 39, 23-24 (2015), 7483–7498.
[20] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.
[21] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.
[22] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587* (2016).
[23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
[24] Dylan Jones, Mehrdad Tamiz, et al. 2010. *Practical goal programming*. Vol. 141. Springer.
[25] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7482–7491.
[26] Mykel J Kochenderfer and Tim A Wheeler. 2019. *Algorithms for optimization*. Mit Press.
[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[28] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. 2019. Pareto Multi-Task Learning. In *Advances in Neural Information Processing Systems*. 12037–12047.
[29] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. (2015).
[30] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482* (2019).
[31] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504* (2019).
[32] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. 2017. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5334–5343.
[33] Jiaqi Ma, Zhe Zhao, Jilin Chen, Ang Li, Lichan Hong, and Ed H Chi. 2019. SNR: Sub-Network Routing for Flexible Parameter Sharing in Multi-task Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 216–223.
[34] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1930–1939.
[35] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730* (2018).
[36] Elliot Meyerson and Risto Miikkulainen. 2018. Pseudo-task Augmentation: From Deep Multitask Learning to Intratask Sharing—and Back. *arXiv preprint arXiv:1803.04062* (2018).
[37] Kaisa Miettinen. 2012. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media.
[38] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3994–4003.
[39] Panos M Pardalos, Antanas Žilinskas, and Julius Žilinskas. 2017. *Non-convex multi-objective optimization*. Springer.
[40] Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. *arXiv preprint arXiv:1704.07156* (2017).
[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.
[42] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).
[43] Yoshikazu Sawaragi, Hirotaka Nakayama, and Tetesuzo Tanino. 1985. *Theory of multiobjective optimization*. Elsevier.
[44] J David Schaffer. 1985. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the first international conference on genetic algorithms and their applications, 1985*. Lawrence Erlbaum Associates. Inc., Publishers.
[45] Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*. 527–538.
[46] Trevor Standley, Amir R Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2019. Which Tasks Should Be Learned Together in Multi-task Learning? *arXiv preprint arXiv:1905.07553* (2019).
[47] Simon Vandenhende, Stamatios Georgoulis, Bert De Brabandere, and Luc Van Gool. 2019. Branched multi-task networks: deciding what layers to share. *arXiv preprint arXiv:1904.02920* (2019).
[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
[49] Sen Wu, Hongyang R Zhang, and Christopher Ré. 2020. Understanding and Improving Information Transfer in Multi-Task Learning. *arXiv preprint arXiv:2005.00944* (2020).
[50] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).
[51] P Le Yu. 1974. Cone convexity, cone extreme points, and nondominated solutions in decision problems with multiobjectives. *Journal of Optimization Theory and Applications* 14, 3 (1974), 319–377.
[52] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782* (2020).
[53] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530* (2016).
[54] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. 2014. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*. Springer, 94–108.
[55] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 43–51.

A APPENDIX

A.1 Convexity of the Pareto Frontier

Proof of Proposition 1. For ease of presentation, we only show the proof for number of tasks $T = 2$. The proof naturally generalizes to $T > 2$. Let $P(\theta) = (\hat{\mathcal{L}}_1(\theta), \hat{\mathcal{L}}_2(\theta))$ denote the feasible point for $\theta \in \Theta$. By definition of a convex curve, we only need to show that for any two points $P(\theta_1), P(\theta_2)$ on the Pareto frontier, the line connecting them, i.e. $\lambda P(\theta_1) + (1-\lambda)P(\theta_2), \forall \lambda \in [0,1]$ is *above* the Pareto frontier. Because $\mathcal{L}_t(\theta)$ is convex in θ, $\hat{\mathcal{L}}_t(\theta)$ is also convex in θ, for $t = 1, 2$. By convexity, we have

$$\begin{aligned} \hat{\mathcal{L}}_1(\lambda\theta_1 + (1-\lambda)\theta_2) &\le \lambda\hat{\mathcal{L}}_1(\theta_1) + (1-\lambda)\hat{\mathcal{L}}_1(\theta_2), \\ \hat{\mathcal{L}}_2(\lambda\theta_1 + (1-\lambda)\theta_2) &\le \lambda\hat{\mathcal{L}}_2(\theta_1) + (1-\lambda)\hat{\mathcal{L}}_2(\theta_2), \end{aligned} \tag{6}$$

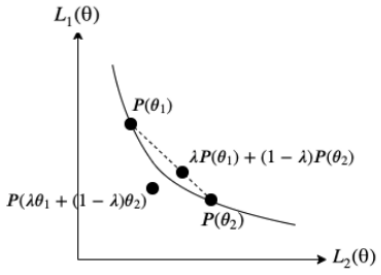


Figure 8: Pareto frontier is convex when objectives are convex.

where $\lambda\theta_1 + (1-\lambda)\theta_2 =: \theta_0 \in \Theta$ by convexity of Θ. This means that for every point $(\lambda\hat{\mathcal{L}}_1(\theta_1)+(1-\lambda)\hat{\mathcal{L}}_1(\theta_2), \lambda\hat{\mathcal{L}}_2(\theta_1)+(1-\lambda)\hat{\mathcal{L}}_2(\theta_2)$, there exists a feasible point $(\hat{\mathcal{L}}_1(\theta_0), \hat{\mathcal{L}}_2(\theta_0))$ dominating it. As shown in Figure 8, this suggests the Pareto frontier between $P(\theta_1)$ and $P(\theta_2)$ is below the line $\lambda P(\theta_1) + (1-\lambda)P(\theta_2), \forall \lambda \in [0,1]$. Therefore the Pareto frontier is convex. □

When some or all objectives are nonconvex, it is unlikely that the Pareto frontier remains convex. But there is still something to say about the shape of the Pareto frontier. We start with the case where all tasks are forced to share all parameters, i.e. $\theta = \theta_{sh}$. Consider square loss for regression tasks for ease of visualization. Let $Y_1 = (y_1^1, ..., y_n^1)^\top$, $Y_2 = (y_1^2, ..., y_n^2)^\top$ and $\hat{f}_\theta = (\hat{f}(x_1), ..., \hat{f}(x_n))^\top$. Then $\hat{\mathcal{L}}_t(\theta)$ is the squared Euclidean distances between Y_t and $\hat{f}_\theta$. Now assume that f_θ is over-parameterized enough so that $\hat{f}_\theta$ is able to *fully* populate the n-dimensional space. In other words, for any $Y = (y_1, ..., y_n)$, there exists $\theta \in \Theta$ such that $\mathcal{L}(f_\theta(x_i), y_i) = 0, \forall i$. In this case, it is obvious to see that Pareto optimality is obtained when $\hat{f}_\theta$ is a linear combination of Y_1 and Y_2, as shown in Figure 9, i.e. where $\sqrt{\hat{\mathcal{L}}_1(\theta)} + \sqrt{\hat{\mathcal{L}}_2(\theta)} = \|Y_1 - Y_2\|_2$. Therefore the Pareto frontier is convex. Similar arguments can be made with other convex loss functions.

When task-specific parameters are allowed, over-parameterized multi-task models can achieve zero training loss. In this case the Pareto frontier is an orthant, which is also convex. Note that the Pareto frontier discussed above is the optimal training loss value

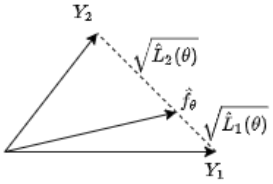


Figure 9: Training loss trade-off for over-parameterized fully-shared multi-task model.

considering all possible $f_\theta \in \mathcal{H}$, without considering optimization error. However this provides some justification for using linear weighting methods for over-parameterized multi-task learning models.

A.2 Experiment Details on Synthetic Data

A.2.1 Details on data generation. Inspired by Finn et al. [17] and Ma et al. [34], we generate a multi-task dataset and define each task as a regression from the input to the output, with the output being a combination of sine waves. To introduce task conflicts together with task correlation, we let the two tasks share a small subset of frequencies. More specifically, the synthetic dataset is generated as follows:

(1) Generate the frequency sets used by the two tasks. $W_1 = \{i \in \mathbb{N} : 0 \le i \le 29 \text{ or } 50 \le i \le 79 \text{ or } 100 \le i \le 129\}$ and $W_2 = \{i \in \mathbb{N} : 25 \le i \le 49 \text{ or } 75 \le i \le 99 \text{ or } 125 \le i \le 149\}$, so that they have overlapping but mostly different frequencies.
(2) Generate shared inputs. Let input dimension $D = 200$ and generate $x_d \sim U[-1/2, 1/2]$ for $1 \le d \le D$.
(3) Generate outputs. Let $x = \sum_{d=1}^{D} x_d$ and generate $e_1, e_2 \sim N(0,1)$. The labels y_1, y_2 for the two regression tasks are defined as:

$$\begin{aligned} y_1 &= \sum_{w_1 \in W_1} (w_1 x + 0.2e_1) \\ y_2 &= \sum_{w_2 \in W_2} (w_2 x + 0.2e_2), \end{aligned} \tag{7}$$

(4) Repeat Step 2-3 $n_{\text{train}} = 100000$ times to generate training dataset, and $n_{\text{test}} = 10000$ times to generate test dataset.

Figure 10 shows the shape of the two tasks as a function of x. We adopt the shared-bottom model architecture with 2 shared hidden layers of size 250 and 125 each with ReLU activation, with the input as $(x_1, ..., x_D)$. For the two task-specific towers for the output y_1 and y_2, we fix the size of hidden ReLU layers to be 100 and vary the number of hidden layers from 0 to 9. The regression loss is the mean-squared error between the prediction and the true value. The resulting Pareto frontier on the test dataset is shown in Figure 1a.

A.3 Experiment Details on MultiMNIST and MultiFashion

A.3.1 Experiment setup. The dataset is randomly split into a training data set of size 100,000 and a test dataset of size 20,000. Figure 11 shows the architectures for the small/medium/large models in the experiments, with an increasing number of shared hidden layers and task-specific hidden layers. SGD optimizer with momentum = 0.9 is used and batch size is fixed at 256.

		Accu-L (%)	Accu-R (%)
Small	MTL	90.95	88.92
	Uncertainty	**91.56**	**89.59**
	MGDA-UB	90.09	89.09
	PCGrad	91.33	88.88
	Self-Auxiliaries	91.49	89.19
Medium	MTL	91.85	89.51
	Uncertainty	91.86	89.92
	MGDA-UB	91.44	89.62
	PCGrad	91.79	89.85
	Self-Auxiliaries	**92.20**	**90.12**
Large	MTL	91.95	90.82
	Uncertainty	92.45	90.68
	MGDA-UB	92.45	90.67
	PCGrad	92.56	90.69
	Self-Auxiliaries	**92.80**	**91.03**

(a) MultiFashionMNIST dataset.

		Accu-L (%)	Accu-R (%)
Small	MTL	81.26	80.96
	Uncertainty	82.46	81.09
	MGDA-UB	81.36	81.08
	PCGrad	81.42	80.94
	Self-Auxiliaries	**82.70**	**81.19**
Medium	MTL	81.90	81.76
	Uncertainty	82.83	82.25
	MGDA-UB	82.38	81.97
	PCGrad	82.30	81.82
	Self-Auxiliaries	**83.06**	**82.38**
Large	MTL	82.72	81.72
	Uncertainty	82.99	82.25
	MGDA-UB	82.68	82.06
	PCGrad	82.78	82.22
	Self-Auxiliaries	**83.37**	**82.67**

(b) MultiFashionMNIST dataset.

Table 4: Left and right accuracies (Accu-L and Accu-R) for **MultiMNIST** and **MultiFashionMNIST** dataset for different model sizes.

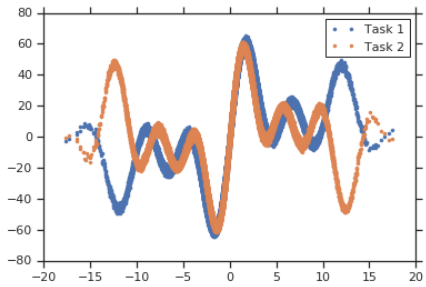


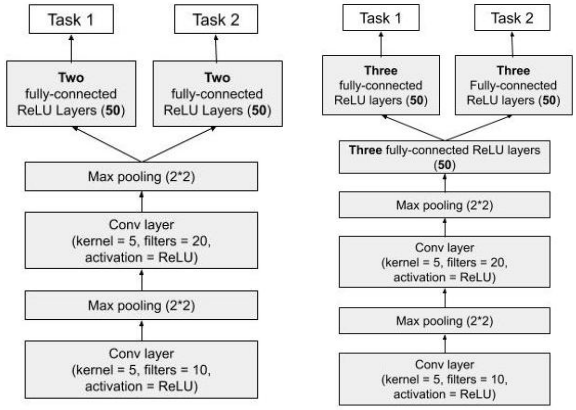
Figure 10: An illustration of two tasks as a function of sum of inputs $x = \sum_{d=1}^{D} x_d$.

For each baseline method and each model architecture, the hyperparameters include learning rate and weight for each task if applicable. For self-auxiliaries, the tuning parameters include weight γ, temperature for the auxiliary tower and width for the bottleneck layer if a bottleneck architecture (Figure 3c) is used. For every method, we first perform 1000 runs for hyperparameters tuning, and then do another 1000 runs with task weights varying from 0 to 1, evaluate each of them on the test dataset, and plot the Pareto frontier (i.e. the Pareto optimal solutions from the 1000 runs).

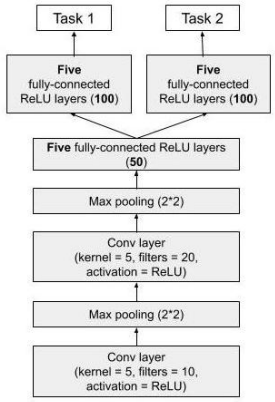
A.3.2 Numerical results. In addition to the Pareto frontier reported in Figure 4, we also present the numerical results here, by reporting the middle point on the Pareto frontier. Table 4a and 4b summarize left and right accuracies for different methods on both datasets with different model sizes. Our proposed method of self-auxiliaries achieves on-par performance with the best baseline methods on small models, and outperforms all baseline methods on medium and large models. The results again confirm our observation that the larger the model, the greater the improvement our method exhibits over the baselines.

A.4 Code

The code for the experiments in Section 5 is available at: https://github.com/double-blind-review-code/self-auxiliaries.



(a) Small model. **(b)** Medium model.



(c) Large model.

Figure 11: Model architectures for MultiMNIST and MultiFashion. (a): Small model. (b): Medium model. (c): Large model.