Task Meta-Transfer from Limited Parallel Labels

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

In this work we introduce a novel meta-learning algorithm that learns to utilize the gradient information of auxiliary tasks to improve the performance of a model on a given primary task. Our proposed method learns to project gradients from the auxiliary tasks to the primary task from a *small* training set with "parallel labels," i.e., examples annotated with respect to both the primary task and the auxiliary tasks. This strategy enables the learning of models with strong performance on the primary task by leveraging a large collection of auxiliary examples and few primary examples. Our scheme differs from methods for transfer learning, multi-task learning or domain adaptation in several ways: unlike naïve transfer learning, our strategy uses auxiliary examples to directly optimize the model with respect to the primary task instead of the auxiliary task; unlike hard-sharing multi-task learning methods, our algorithm devotes the entire capacity of the backbone model to attend the primary task instead of splitting it over multiple tasks; unlike most domain adaptation techniques, our scheme does not require any overlap in labels between the auxiliary and the primary task, thus enabling knowledge transfer between completely disjoint tasks. Experiments on two image analysis benchmarks involving multiple tasks demonstrate the performance improvements of our meta-learning scheme over naïve transfer learning, multi-task learning as well as prior related work.

1 Introduction

Transfer learning [8] has emerged as a powerful methodology to enable the use of deep models in domains with scarce labeled data. The idea is to pretrain the model on a related domain where data is plentiful, and then transfer useful information to the target domain either by finetuning or by training a lightweight model on top of the pretrained features. This simple strategy has enabled remarkable state-of-the-art results to be achieved in small-data regimes across different fields, including computer vision [15], natural language processing [46], and robotics [47]. However, because the pretraining stage optimizes the model solely on the auxiliary task, the effectiveness of this approach lies on the shoulder of the human expert who must find an auxiliary task that makes the pretraining procedure directly beneficial for the final training on the target task.

In this work instead we propose a meta-learning procedure that explicitly models the relationship between tasks and enables the auxiliary examples to be used to optimize the model with respect to the target task rather than the auxiliary task. The relationship between tasks is meta-learned from a small training set with "parallel labels," i.e., examples annotated with respect to both the primary task and the auxiliary tasks. The parallel labels are used to learn how to transform the gradients computed from auxiliary examples in order to make them beneficial for the primary task. The gradient transformation is implemented by a low-rank linear projection that effectively aligns the backpropagation updates of the auxiliary task to those of the primary task. The intuition is that we can harness information from label spaces that are easier to annotate to improve performance on tasks where labels are costly to obtain.

As further discussed in our technical section, our approach adopts the multi-head network architecture commonly used for multi-task learning, where a single backbone common to all tasks splits into separate branches, each addressing a different task. However, differently from multi-task training, the learning capacity of the backbone is entirely devoted to the primary task as the gradients backpropagated from the auxiliary branches are transformed into gradients optimized for the primary task *before* being backward-passed to the backbone. Our approach differs also from domain adaptation methods, which assume different data distributions for the source and the target domain but typically require the two domains to share the same underlying task (we note the exceptions of open-set [32, 37] and partial-set [52, 3] domain adaptation). Instead, our method is applicable in arbitrary settings involving tasks defined on the same input space but having disparate output spaces.

2 Related work

As our approach optimizes a model with respect to a primary task by leveraging one or more auxiliary tasks where training data is abundant, it is closely related to transfer learning methods. In computer vision, the advantages of transfer learning were first demonstrated in the context of visual recognition tasks [8, 51, 31, 39, 48]. Yet, recent research [50, 41] has shown that the advantages of transfer learning extend across a broad range of visual tasks, beyond recognition, even between seemingly distant tasks. These papers present also approaches that find the best source tasks for a given primary task. We note that our method is complementary to this line of work, as our focus is on *how* to best leverage auxiliary tasks. Our approach differs from methods that aim at preserving good performance on the original source tasks when adapting the network to the target task [27, 26, 36, 38], since our technique is specifically designed to optimize performance on the primary task using the auxiliary tasks as mere means towards this goal.

While pretraining followed by finetuning is the natural approach for transfer learning, our experiments demonstrate that multi-task learning [5, 35] gives in certain settings superior performance on the target task. A variety of strategies have been investigated in prior work for the purpose of boosting the performance of multi-task learning, including task or gradient weighting [17, 6], gradient surgery [49], attention [22], as well as specialized sharing units [28]. Most multi-task learning methods can be categorized into two classes: hard-sharing versus soft-sharing [35] of parameters. Models of the former class have a single backbone shared across all tasks and a specialized network head for each individual task. The downside of these approaches is that the performance is inherently limited by the capacity of the backbone that must address all tasks simultaneously and thus it is destined to decay as the number of tasks is grown beyond a certain value [18]. Reducing the number of shared parameters decreases this problem but this also decreases the potential benefit of shared supervision from all tasks. Conversely, multi-task models that rely on soft parameter sharing allow each task to have distinct parameter but use regularizations to encourage the parameters to be similar. The disadvantage is that under this strategy the number of parameters grows linearly with the number of tasks. Our approach is most similar to hard parameter sharing models, since it uses a single backbone. However, unlike these models, it effectively allocates the entire capacity of the backbone to the primary task, since the gradients from the auxiliary branches are transformed to optimize performance on the primary task and thus the auxiliary tasks do not consume the learning capacity of the network. Yet, because it has a single backbone, it does not suffer from the problem of linear growth of parameters with the number of tasks that affects soft-sharing models.

We note that the setting considered in our work differs from that typically considered in domain adaptation [23, 24, 12, 43, 25, 4, 45, 53, 19], where the source domain and the target domain involve the *same* task, albeit under different data distributions. Within this area, the work that is closest to our own is the GradMix approach of [19] which transfers across tasks in addition to domains. Similarly to our method, GradMix uses gradients from the auxiliary tasks for optimization. GradMix weighs the auxiliary gradients according to their cosine similarity to gradients computed from a validation set of primary examples. A similar approach was also studied in the multi-task setting [9]. Although such weighting was shown to be empirically beneficial, there is no guarantee to aid optimization. Instead, our approach meta-learns a transformation of the auxiliary gradients that is explicitly optimized to reduce the loss.

Our meta-learning approach uses a backward-on-backward auto differentiation to compute the second order derivative. This meta-learning strategy has been adopted in prior work, mostly for the purpose of few-shot learning [34, 42] and fast model adaption [11] but also in the context of learning to optimize [1]. Within the genre of meta-learning methods, the work of [21] is more closely related to our own, since it leverages auxiliary tasks to improve the performance of a primary task. Meta-learning is in this case used to automatically generate labels for self-supervised auxiliary tasks whereas in our work we adopt it for making use of supervised auxiliary gradients in the optimization.

[20] studies how to speed up training for reinforcement learning of a primary task by meta-learning adaptive weights for the auxiliary losses. Different from this work, we adopt a more complex projection of gradients than a linear combination, and apply our approach to supervised learning instead of reinforcement learning. We show in experiments that our projection model outperforms scalar weighting of auxiliary gradients.

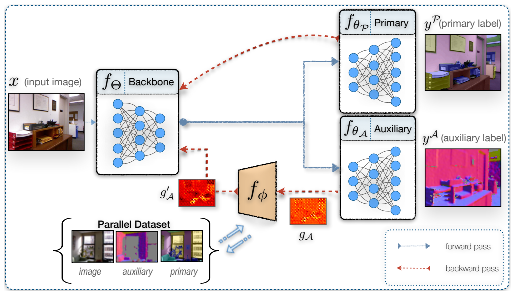

Figure 1: Overview of our proposed approach

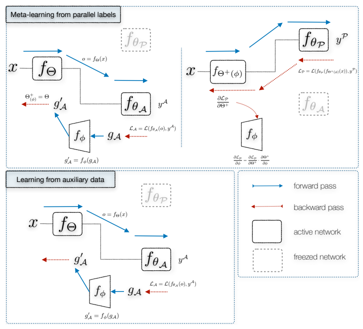
Meta-learning from parallel labels
Learning from auxiliary data
forward pass
backward pass
active network
freezed network

Figure 2: Illustration of the graph workflow of our approach

3 Technical Approach

3.1 Problem Statement

In this section, we introduce our method, which we name *"Task Meta-Transfer"* (TMT). We first formally introduce the problem of Task Transfer. Let $\mathcal{X}$ and $\mathcal{X}^\mathcal{A}$ be two sets of training input examples, such that $\mathcal{X} \subset \mathcal{X}^\mathcal{A}$. We assume that all examples in $\mathcal{X}$ are labeled with respect to both the primary task and the auxiliary task. Thus, each example $x_i \in \mathcal{X}$ has both a primary-task label $y_i^\mathcal{P} \in \mathcal{Y}^\mathcal{P}$ and an auxiliary-task label $y_i^\mathcal{A} \in \mathcal{Y}^\mathcal{A}$. Instead, each input example $x_j \in \mathcal{X}^\mathcal{A} - \mathcal{X}$ is labeled with an auxiliary-task label $y_j^\mathcal{A} \in \mathcal{Y}^\mathcal{A}$ only. Note that while the primary task and the auxiliary task are defined on the same input space (e.g., the space of all natural images, or the space of all English documents), they need not share any labels, i.e., $\mathcal{Y}^\mathcal{P}$ and $\mathcal{Y}^\mathcal{A}$ may be disjoint. In fact, we will present results for cases where one task involves regression to a continuous space (e.g., scene depth estimation from a single photo) while the other is classification (e.g., image categorization). Our only underlying assumption is that the tasks are loosely related such that gradients computed for the auxiliary task may be "transformed" into gradients useful to optimize the model with respect to the primary task. We stress that the sole objective of this work is to obtain a model that performs well on the primary task.

3.2 Approach Overview

We adopt a two-headed network with hard parameter sharing [35] as our architecture. The first part of the network is shared between the two tasks and computes a feature tensor $f_\Theta(x)$ from input x. We refer to this part of the model as the backbone. The features computed from the backbone are passed as input to two separate branches, an auxiliary network $f_{\theta_\mathcal{A}}$ and a primary network $f_{\theta_\mathcal{P}}$, which compute separate predictions for each task. While this type of architecture is commonplace in multi-task learning, the novelty in our approach lies in the training procedure, which is illustrated in Figure 1 and Figure 2. Let $g_\mathcal{A}$ be the gradient backpropagated from the auxiliary branch at the bifurcation (i.e., the layer where the backbone splits into the task-specific branches). Directly backward-passing $g_\mathcal{A}$ (as traditionally done in multi-task learning) would update the backbone to benefit the auxiliary task. Instead, we propose to transform the auxiliary gradient into a new gradient vector $g'_\mathcal{A}$ that yields improvements for the primary task. The projected gradient $g'_\mathcal{A}$ is computed from $g_\mathcal{A}$ using a meta-learned model gradient projection f_ϕ, i.e., $g'_\mathcal{A} = f_\phi(g_\mathcal{A})$. The meta-learning step leverages the small set of example $\mathcal{X}$ that are annotated with respect to both tasks. This step explicitly optimizes gradient projection f_ϕ to 'transform" (or "project") gradients computed for the auxiliary task into gradients optimizing the backbone for the primary task. We discuss the meta-learning step in Subsection: 3.3. The gradient projection f_ϕ makes it possible to use the large set of examples in $\mathcal{X}^\mathcal{A}$ to learn a good backbone representation for the primary task. We describe this second step in Subsection: 3.4. In practice, during optimization we alternate between one meta-learning step (using a mini-batch sampled from $\mathcal{X}$) and k consecutive update steps from the auxiliary task (using k mini-batches sampled from $\mathcal{X}^\mathcal{A}$). The hyper-parameter k is chosen via hold-out validation and is further discussed in our experimental section. Each update from the auxiliary task modifies the auxiliary branch and, more importantly, the backbone to benefit the primary task. We note that our approach can be used with any arbitrary choice of parameterized mapping gradient projection f_ϕ preserving the dimensionality of the input. In Subsection: 3.5 we discuss our specific choice of parameterized function f_ϕ.

3.3 Meta-Learning from Parallel Labels

The learning of gradient projection f_ϕ is through a meta-loss on the set of examples $\mathcal{X}$ for which parallel labels are available. Let us consider a single mini-batch sample $x \in \mathcal{X}$, with corresponding labels $y^\mathcal{P} \in \mathcal{Y}^\mathcal{P}$ and $y^\mathcal{A} \in \mathcal{Y}^\mathcal{A}$. We pass x forward through the backbone f_Θ to compute the backbone activation $o = f_\Theta(x)$ at the bifurcation. We then forward o into auxiliary network $f_{\theta_\mathcal{A}}$ to calculate the loss $\mathcal{L}_\mathcal{A}$ with respect to the auxiliary task: $\mathcal{L}_\mathcal{A} = \mathcal{L}(f_{\theta_\mathcal{A}}(o), y^\mathcal{A})$

Let $g_\mathcal{A}$ denote the gradient of $\mathcal{L}_\mathcal{A}$ with respect to the activation o: $g_\mathcal{A} = \nabla_o \mathcal{L}_\mathcal{A}$

This gradient is computed via traditional backpropagation by backward-passing the gradients from the loss $\mathcal{L}_\mathcal{A}$ through the layers of auxiliary network $f_{\theta_\mathcal{A}}$ until reaching the bifurcation layer. We refer to this gradient as the auxiliary gradient.

Using this gradient would cause the backbone to be optimized with respect to the auxiliary task. Instead, we transform the auxiliary gradient into a new gradient vector $g'_\mathcal{A} = f_\phi(g_\mathcal{A})$ before backward-passing it to update the parameters Θ of the backbone:

$$\Theta^+ = \Theta - \alpha g'_\mathcal{A} \frac{\partial o}{\partial \Theta} = \Theta - \alpha f_\phi(\nabla_o \mathcal{L}_\mathcal{A}) \nabla_\Theta o \tag{1}$$

Note that Θ^+ is now a function of ϕ, i.e., $\Theta^+ = \Theta^+(\phi)$. We then start a second forward pass propagating the same mini-batch into backbone f_{Θ^+} and primary network $f_{\theta_\mathcal{P}}$ to calculate the loss $\mathcal{L}_\mathcal{P}$ for the primary task:

$$\mathcal{L}_\mathcal{P} = \mathcal{L}(f_{\theta_\mathcal{P}}(f_{\Theta^+(\phi)}(x)), y^\mathcal{P}) \tag{2}$$

Finally, we update gradient projection f_ϕ with respect to this loss:

$$\begin{aligned} \phi &\leftarrow \phi - \beta \nabla_\phi \mathcal{L}_\mathcal{P} && (3) \\ &\leftarrow \phi - \beta \nabla_{\Theta^+} \mathcal{L}_\mathcal{P} \nabla_\phi \Theta^+ && (4) \end{aligned}$$

Updating the Network with Primary Labels. The meta-learning step is carried out over a mini-batch of examples in $\mathcal{X}$, which include annotations with primary labels in $\mathcal{Y}^\mathcal{P}$. Thus, while carrying out the meta-learning step, we keep primary network $f_{\theta_\mathcal{P}}$ and backbone f_Θ updated with traditional backpropagation using the primary loss from Eq. (2).

3.4 Learning from Auxiliary Data

This step uses *(i)* a mini-batch sampled for the auxiliary task (i.e., defined over examples $x \in \mathcal{X}^\mathcal{A}$ with corresponding labels $y^\mathcal{A} \in \mathcal{Y}^\mathcal{A}$) and *(ii)* the current estimate of the gradient projection mapping f_Θ, in order to update the backbone in a way that is beneficial for the *primary* task. First, a standard forward pass is performed through backbone f_Θ to compute activation $o = f_\Theta(x)$, and auxiliary network $f_{\theta_\mathcal{A}}$ to calculate the loss with respect to the auxiliary task: $\mathcal{L}_\mathcal{A} = \mathcal{L}(f_{\theta_\mathcal{A}}(o), y^\mathcal{A})$

The subsequent backward pass is split into two parts. Traditional backpropagation on the auxiliary branch to keep auxiliary network updated using the auxiliary loss: $\theta_\mathcal{A} \leftarrow \theta_\mathcal{A} - \alpha \frac{\partial \mathcal{L}_\mathcal{A}}{\partial \theta_\mathcal{A}}$

However, at the split the gradient is transformed using the current projection f_ϕ in order to update the backbone with respect to the primary task. In other words, instead of updating the backbone with respect to the auxiliary task:

$$\Theta \leftarrow \Theta - \alpha g_{\mathcal{A}} \frac{\partial o}{\partial \Theta}, \qquad g_{\mathcal{A}} = \nabla_o \mathcal{L}_{\mathcal{A}} \tag{5}$$

Our procedure backward-passes the projected gradient $g'_{\mathcal{A}} = f_\phi(g_{\mathcal{A}})$: $\quad \Theta \leftarrow \Theta - \alpha g'_{\mathcal{A}} \frac{\partial o}{\partial \Theta}$

3.5 Gradient Projection Model

The gradient projection f_ϕ can be any learnable parametric mapping preserving the input dimensionality. Our experiments show that simple *linear* projections can transform the auxiliary gradients so as to benefit the primary task (see Figure 3, which shows that increasing further the model complexity does not produce gains on most tasks. Thus, we consider a generic projection matrix $\phi \in \mathbb{R}^{C \times C}$ such that $g'_{\mathcal{A}} = f_\phi(g_{\mathcal{A}}) = \phi \cdot g_{\mathcal{A}}$ where C is the dimensionality (i.e., the number of neurons) of the bifurcation layer. In the case of a convolutional network, we define C to be number of convolutional feature maps of the bifurcation layer and implement the linear projection via 1×1 point-wise convolution.

This projection model involves C^2 parameters to be meta-learned via second-order differentiation. As demonstrated in our empirical evaluation, this optimization is quite challenging. In practice we found advantageous to constrain the form of the projection and further reduce the number of parameters to be meta-learned. We achieve this goal by parameterizing the projection matrix ϕ as $\phi = \Lambda + UU^\top$ where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_C)$ is a diagonal matrix and $U \in \mathbb{R}^{C \times r}$. This effectively constrains the matrix $UU^\top$ to have rank $r < C$ and reduces the number of meta-parameters to $C + rC$.

In the case of multiple auxiliary tasks (say, $T > 1$ auxiliary tasks), we define a separate projection model ϕ_t for each auxiliary task $\mathcal{A}_t$ and compute the gradient $g'_{\mathcal{A}}$ by summing the projected gradients of all auxiliary tasks:

$$g'_{\mathcal{A}} = f_{\phi_1, \ldots, \phi_T}(g_{\mathcal{A}_1}, \ldots, g_{\mathcal{A}_T}) = \sum_{t=1}^{T} \phi_t \cdot g_{\mathcal{A}_t} \tag{6}$$

We note that, despite its simplicity, this strategy enables learning the relative importance of the auxiliary gradients that are fused into the projected gradient $g'_{\mathcal{A}}$.

4 Experiments

4.1 Datasets

NYUv2 [30] is an indoor dataset where each image is annotated with respect to three tasks: the 13-class semantic segmentation from [2], as well as the depth estimation and surface normal estimation from [10]. We report the results for all the possible primary-auxiliary tasks. We use 795 images for training, 327 for validation, and 327 for testing. We simulate a scenario where limited labels for the primary task are available by assuming that the set of parallel examples $\mathcal{X}$ is a small subset of the entire training set, while the auxiliary training set $\mathcal{X}^{\mathcal{A}}$ consists of all 795 examples. ***CityScapes*** [7] is mainly used for semantic urban scene understanding. We use 2975 images for training, 250 for validation and 250 for testing. Our experiments include auxiliary-primary task-pairs obtained from the tasks of inverse depth estimation, 7-class segmentation and 19-class segmentation [22]. More details can be found in supplementary material.

4.2 Network Architectures

We utilize two different encoder-decoder architectures for pixel-level prediction: ***SplitVGG*** and ***SplitRes***, based on VGG16 [40] and ResNet18 [14], respectively. To construct a two-headed model, we introduce a split at a particular block of the decoder, and create two separate copies of all subsequent layers (i.e., two branches), one branch devoted to the auxiliary task and the other to the primary task. Thus, the part of the network before the split constitutes the backbone f_Θ. We report results obtained for different positions of the split in the network. We denote with ***SplitVGG-***x and ***SplitRes-***x the models obtained by splitting the network at the x-th block of the decoder. For example,

SplitVGG-3 indicates a network where the backbone consists of the encoder and the first 2 blocks of the decoder. Further details about the architectures and training can be found in the supplementary material.

Table 1: Test performance of different models (average of 3 runs) on *NYUv2* dataset using different pairs of primary and auxiliary tasks with two distinct architectures (SplitVGG-3 and SplitRes-3). The statistical variance and hyper-parameters learned via validation is provided in supplementary material. The best performing models are shown in bold, the second best are underlined. Our *TMT* outperforms other methods on 8 out of the 12 cases, and ranks second best in 2 cases. In the remaining 2 cases, *TMT* achieves performance comparable to the best methods.

	Tasks		*Primary Only*	*Transfer Learning*	*MTL*	*Kendall et al. [17]*	*Du et al. [9]*	*Lin et al. [20]*	*TMT (ours)*	*Oracle*
	Primary	*Auxiliary*								
SplitVGG-3	Semantic (`IoU` $\uparrow$)	Depth	12.87	13.48	$\underline{14.11}$	13.77	13.72	13.70	**15.13**	18.23
		Normal		14.43	14.04	$\underline{14.60}$	13.68	13.80	**16.74**	
	Depth ($10^2 \times$ `AbsError` $\downarrow$)	Semantic	81.89	70.55	$\underline{70.03}$	71.55	71.59	70.06	**69.41**	64.14
		Normal		71.62	71.18	73.54	74.09	$\underline{70.59}$	**68.74**	
	Normal (`AngleDist` $\downarrow$)	Semantic	40.51	37.70	35.79	36.39	**35.60**	36.40	$\underline{35.71}$	32.97
		Depth		37.65	**36.60**	36.97	37.19	$\underline{36.83}$	36.92	
SplitRes-3	Semantic (`IoU` $\uparrow$)	Depth	11.14	12.20	13.07	12.16	12.89	$\underline{13.15}$	**13.41**	15.94
		Normal		12.15	$\underline{13.96}$	12.58	12.75	13.09	**14.10**	
	Depth ($10^2 \times$ `AbsError` $\downarrow$)	Semantic	81.34	74.48	73.24	77.62	74.48	$\underline{73.03}$	**72.18**	68.52
		Normal		$\underline{74.61}$	75.15	79.95	75.76	74.68	**73.17**	
	Normal (`AngleDist` $\downarrow$)	Semantic	41.24	40.83	$\underline{38.21}$	40.03	**38.12**	38.67	38.45	36.61
		Depth		40.28	**38.58**	40.67	38.86	39.04	$\underline{38.78}$	

4.3 Baselines

Primary Only is single-task learning with primary labels in $\mathcal{X}$. The method does not utilize any auxiliary information.

Transfer Learning consists of first pre-training the model on the large auxiliary dataset $\mathcal{X}^{\mathcal{A}}$ and then fine-tuning it on the primary task using the labels in $\mathcal{X}$. We use early-stopping for both pre-training and fine-tuning, by measuring validation performance on the auxiliary task for pre-training and validation performance on the primary task for finetuning.

Multi-Task Learning (MTL) is the joint training of the entire model with respect to both the primary and the auxiliary tasks. We consider two hard-sharing MTL baselines [17, 9], since their architectures are closer to our model compared to those of soft-sharing models. In addition, we include a weighted hard sharing MTL model [35]. The mini-batches sampled from $\mathcal{X}^{\mathcal{A}}$ are used to update only the auxiliary and the backbone networks using the auxiliary loss, while the mini-batches sampled from $\mathcal{X}$ (for which parallel labels are available) induce backpropagation through both branches (using both losses) as well as the backbone. Because the subset of examples with parallel labels is much smaller than the auxiliary dataset, we found experimentally beneficial to alternate one mini-batch from $\mathcal{X}$ every k consecutive mini-batches sampled from $\mathcal{X}^{\mathcal{A}}$. The hyper-parameter $k \in \{1, 2, 3, \ldots, 10\}$ is chosen via hold-out validation with respect to validation performance on the primary task. This sampling strategy effectively implements a form of task weighting, which is quite common in multi-task settings. Furthermore, this renders this approach more similar to our own method, with the difference being that our TMT uses a meta-learned gradient projection matrix to transform the gradient from the auxiliary branch to the backbone, while multi-task learning is equivalent to using TMT with a fixed identity matrix for gradient projection. This direct comparison makes it possible to assess the true value of the meta-learning step.

Uncertainty Task Weighting [17] is a MTL method that treats the weighting of each task as a learnable parameter. This work shows strong performance on the full NYUv2 dataset. Instead, our setting involves few primary examples and many auxiliary examples. Thus, we adapt this method to our setting as follows: we alternate the training between (1) training with Uncertainty Task Weighting using examples that have labels for both the primary and the auxiliary task and (2) standard training using auxiliary examples. To get the best results, we search over the values of hyper-parameter k to control how often we alternate between these two steps.

Du et al. [9] aim at minimizing the primary loss by weighting the auxiliary loss using cosine similarity between the gradients of tasks. Using cosine similarity to weight gradients has been also explored in by GardMix in domain adaptation [19], and multi-task learning [49]. Due to the imbalance of $\mathcal{X}$ and $\mathcal{X}^{\mathcal{A}}$, we find it is beneficial to modify their method by alternating one mini-batch from $\mathcal{X}$ every k consecutive mini-batches sampled from $\mathcal{X}^{\mathcal{A}}$, as we do for Multi-Task Learning described above.

Meta Task Weighting has been studied for speeding up reinforcement learning in [20]. The approach effectively learns a scalar weighting for the individual tasks. We apply this approach to the supervised learning setting considered in this paper.

Oracle is single-task learning using a dataset with *primary* labels for *all* examples in $\mathcal{X}^{\mathcal{A}}$. This represents an upper bound performance, as the method has access to primary supervision for all examples instead of for only a subset.

4.4 Results

Table 2: Performance of various models (average of 3 runs) on *CityScapes* dataset using different pairs of primary and auxiliary tasks with SplitVGG-3

Primary	class-7 (IoU $\uparrow$)	class-19 (IoU $\uparrow$)	Depth ($10^2\times$AbsError$\downarrow$)	
Auxiliary	Depth		class-7	class-19
Primary Only	37.42	18.30	2.31	2.31
Transfer Learning	38.00	19.11	2.02	2.01
MTL	39.21	**21.52**	1.83	1.83
Kendall et al. [17]	39.25	20.90	2.25	2.23
Du et al. [9]	38.53	20.87	1.82	1.82
Lin et al. [20]	39.08	20.96	1.85	1.86
TMT (ours)	**39.48**	21.28	**1.80**	**1.78**
Oracle	45.90	24.83	1.74	

Table 3: Primary-task performance on the test set of *NYUv2* using *two* auxiliary tasks with SplitVGG-3 architecture

Primary	Semantic (S) (IoU $\uparrow$)	Depth (D) ($10^2\times$AbsError$\downarrow$)	Normal (N) (AngleDist$\downarrow$)
Auxiliary	D + N	S + N	S + D
Transfer Learning	15.30	69.25	35.79
MTL	15.47	67.32	**34.74**
TMT (ours)	**16.59**	**66.84**	35.01

Single Auxiliary Task. We study the performance of SplitVGG-3 and SplitRes-3 on NYUv2 with a parallel dataset set to be 10% of the auxiliary dataset. We report the test results in Table 1 with the error of each method averaged over three separate training runs. As described in Subsection: 3.5, the rank r controls the complexity of our gradient projection f_ϕ. The reported results are computed by selecting the best r for each primary-auxiliary pair according to the performance on the validation set. Similarly, MTL are multi-task learning models trained where the hyper-parameter k is optimized based on the validation performance. [17] and [9] are two advanced MTL methods that work well on datasets where the number of examples of the different tasks are balanced, but suffer in the scenario where the primary task labels are scarce. Our TMT outperforms other methods on 8 out of the 12 cases, and ranks second best in 2 cases. In the remaining 2 cases, TMT is close to the best methods. Note that the comparison between TMT and MTL shows the value of a learnable projection matrix over a fixed identity projection. Instead, the comparison between TMT and Meta Task Weighting in [20], TMT shows the advantage of a more complex meta-learnable projection matrix compared to a simple weighting scalar. We stress that our setup is different from that considered by most prior works evaluated on these benchmarks [29, 13, 16, 33, 44], since the methods in our experiments have access to only 10% of the primary labels in the datasets. As a result the absolute performances are not comparable to those reported in prior works that use 100% of the labels.

The improvement of performance compared to multi-task learning is particularly large on depth using normal, and semantic using normal. These results underscore the importance of transforming the auxiliary gradient using the meta-learned projection as opposed to directly backpropagating it, as done in multi-task learning. [9] produces larger errors in 10 out of 12 cases compared to multi-task learning. This is because when auxiliary tasks are only loosely related to the primary task, the weighting of auxiliary loss by cosine similarity between auxiliary and primary tasks is either too small or negative. The small or negative weighting of auxiliary loss in [9] will opt to utilize few auxiliary examples to help the primary training. All 6 methods leveraging auxiliary data outperform by large margins "Primary Only" which learns the model using only primary data. Table 2 reports analogous results on CityScapes. Even on CityScapes, TMT does overall better than multi-task learning and consistently improves over transfer learning and training from only primary data.

Multiple Auxiliary Tasks. Our method can be applied in scenarios involving multiple auxiliary tasks by simply averaging the projected gradients (see Eq. (6)). We similarly adapt the baseline of multi-task learning to perform joint training over the multiple auxiliary tasks and the primary task. Table 3 shows a comparison between these two approaches using two auxiliary tasks to improve performance on the primary task. It can be noticed that, compared to using a single auxiliary task (see Table 1), leveraging two auxiliary tasks yields consistently better performance for both TMT and multi-task learning.

4.5 Robustness Study of the Experiments

Different Ranks of Projection Model. The number of parameters in the gradient projection f_ϕ is controlled by the rank r of the matrix U. When f_ϕ contains too few parameters, the projection matrix may not be complex enough to properly transform the auxiliary gradients. Conversely, we found that when f_ϕ is over-parameterized, the optimization in the meta-learning step becomes difficult given the small size of the parallel dataset (see supplementary material). We study the effect of different ranks r on NYUv2. The output tensor of the backbone f_Θ for this network has 128 channels. Thus, we vary the ranks from 5 to 100. In Figure 3 for nearly all primary-auxiliary pairs the performance of TMT remains relatively unperturbed by the choice of rank. This implies that we need to choose the number of parameters in the gradient projection f_ϕ wisely, depending on the complexity of the transformation required to adapt the auxiliary gradients.

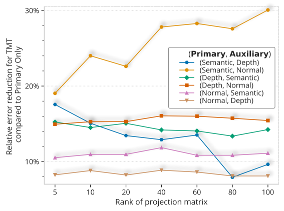

Figure 3: Varying the rank r of the matrix defining the gradient transformation f_ϕ. The y-axis shows the relative-error reduction for TMT compared to Primary Only on the test set of NYUv2 with SplitVGG-3.

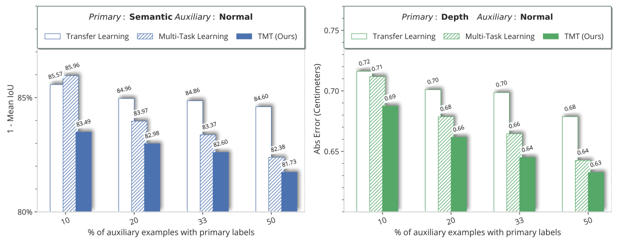

Figure 4: Studying how the number of parallel labels affects the error on the test set of NYUv2 using SplitVGG-3. TMT outperforms multi-task learning under all different training settings.

Different Splitting Blocks. The splitting position in the network affects dramatically the number of parameters shared by the tasks. For example, while in SplitVGG-2 only 98% of the parameters reside in the backbone, in SplitVGG-5 50% of the parameters are devoted to the backbone. Figure 5 shows how the position of the split affects the performance of transfer learning, multiple-task learning, and TMT.

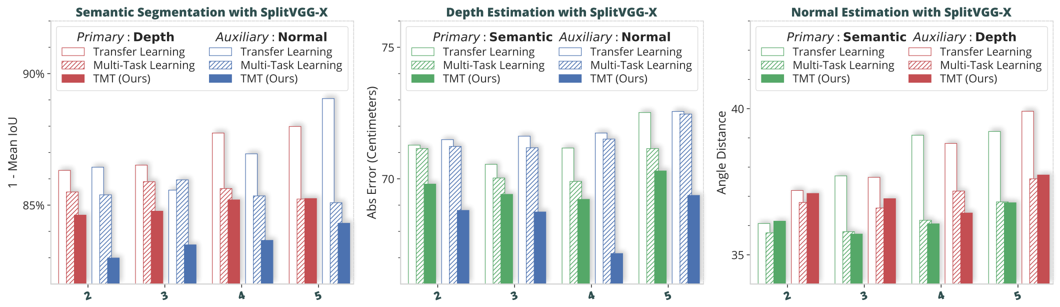

Figure 5: Bar plots show the primary-task errors obtained on NYUv2 for different positions of the splitting between the backbone and the task-specific branches. For the primary tasks of depth estimation and semantic segmentation, TMT does consistently better than the baselines.

Different Amounts of Parallel Labels. We now consider how performance varies as we set the the number of parallel labels $\mathcal{X}$ to be 10%, 20%, 33% or 50% of the auxiliary dataset $\mathcal{X}^{\mathcal{A}}$. Figure 4 shows a comparison between multi-task learning (weighted) and TMT, highlighting the lower error achieved with TMT under all these different experimental settings.

References

[1] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989, 2016.

[2] C. Camille, F. Clément, N. Laurent, and L. Yann. Indoor semantic segmentation using depth information. In *International Conference on Learning Representations, ICLR*, 2013.

[3] Z. Cao, L. Ma, M. Long, and J. Wang. Partial adversarial domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018.

[4] F. M. Cariucci, L. Porzi, B. Caputo, E. Ricci, and S. R. Bulo. Autodial: Automatic domain alignment layers. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5077–5085. IEEE, 2017.

[5] R. Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on International Conference on Machine Learning*, 1993.

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

[7] 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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

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

[10] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[11] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[12] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[13] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[16] S. Hickson, K. Raveendran, A. Fathi, K. Murphy, and I. Essa. Floors are flat: Leveraging semantics for real-time surface normal prediction. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019.

[17] A. Kendall, Y. Gal, and R. Cipolla. 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 (CVPR)*, 2018.

[18] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6129–6138, 2017.

[19] J. Li, Z. Xu, W. Yongkang, Q. Zhao, and M. Kankanhalli. Gradmix: Multi-source transfer across domains and tasks. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[20] X. Lin, H. Baweja, G. Kantor, and D. Held. Adaptive auxiliary task weighting for reinforcement learning. In *Advances in Neural Information Processing Systems 32*, 2019.

[21] S. Liu, A. Davison, and E. Johns. Self-supervised generalisation with meta auxiliary learning. In *Advances in Neural Information Processing Systems*, pages 1677–1687, 2019.

[22] S. Liu, E. Johns, and A. J. Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1871–1880, 2019.

[23] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pages 97–105, 2015.

[24] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in neural information processing systems*, pages 136–144, 2016.

[25] Z. Luo, Y. Zou, J. Hoffman, and L. F. Fei-Fei. Label efficient learning of transferable representations acrosss domains and tasks. In *Advances in Neural Information Processing Systems*, pages 165–177, 2017.

[26] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[27] A. Mallya, D. Davis, and S. Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[28] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[29] A. Mousavian, H. Pirsiavash, and J. Košecká. Joint semantic segmentation and depth estimation with deep convolutional networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, 2016.

[30] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[31] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[32] P. Panareda Busto and J. Gall. Open set domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 754–763, 2017.

[33] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia. Geonet: Geometric neural network for joint depth and surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[34] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

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

[36] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[37] K. Saito, S. Yamamoto, Y. Ushiku, and T. Harada. Open set domain adaptation by backpropagation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 153–168, 2018.

[38] J. Serra, D. Suris, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557, 2018.

[39] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[41] T. Standley, A. R. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese. Which tasks should be learned together in multi-task learning? *arXiv preprint arXiv:1905.07553*, 2019.

[42] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele. Meta-transfer learning for few-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[43] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[44] P. Wang, X. Shen, B. Russell, S. Cohen, B. Price, and A. L. Yuille. Surge: Surface regularized geometry estimation from a single image. In *Advances in Neural Information Processing Systems 29*, pages 172–180, 2016.

[45] R. Xu, Z. Chen, W. Zuo, J. Yan, and L. Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[46] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, 2019.

[47] L. Yen-Chen, A. Zeng, S. Song, P. Isola, and T.-Y. Lin. Learning to see before learning to act: Visual pre-training for manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[48] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[49] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

[50] A. R. Zamir, A. Sax, W. B. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[51] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *In Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[52] J. Zhang, Z. Ding, W. Li, and P. Ogunbona. Importance weighted adversarial nets for partial domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8156–8164, 2018.

[53] H. Zhao, S. Zhang, G. Wu, J. M. Moura, J. P. Costeira, and G. J. Gordon. Adversarial multiple source domain adaptation. In *Advances in neural information processing systems*, pages 8559–8570, 2018.