Knowledge Distillation for Multi-task Learning

Wei-Hong Li and Hakan Bilen

VICO Group, University of Edinburgh, United Kingdom
{w.h.li, hbilen}@ed.ac.uk

Abstract. Multi-task learning (MTL) is to learn one single model that performs multiple tasks for achieving good performance on all tasks and lower cost on computation. Learning such a model requires to jointly optimize losses of a set of tasks with different difficulty levels, magnitudes, and characteristics (*e.g.* cross-entropy, Euclidean loss), leading to the imbalance problem in multi-task learning. To address the imbalance problem, we propose a knowledge distillation based method in this work. We first learn a task-specific model for each task. We then learn the multi-task model for minimizing task-specific loss and for producing the same feature with task-specific models. As the task-specific network encodes different features, we introduce small task-specific adaptors to project multi-task features to the task-specific features. In this way, the adaptors align the task-specific feature and the multi-task feature, which enables a balanced parameter sharing across tasks. Extensive experimental results demonstrate that our method can optimize a multi-task learning model in a more balanced way and achieve better overall performance.

1 Introduction

The objective of multi-task learning (MTL) [3,26] is to develop methods that can tackle a large variety of tasks within a single model. MTL has multiple practical benefits. First, learning shared parameters across multiple tasks leads to representations that can be more data-efficient to train and also generalize better to unseen data. Second, sharing parameters and computations across tasks can significantly reduce both training and inference time over running multiple individual models, which is especially important in platforms with limited computational resources such as mobile devices. Therefore there is a growing interest in developing MTL methods and MTL has been successfully applied to machine learning problems in several fields including natural language processing [5], computer vision [2,14] and speech recognition [27].

There are at least two challenges to achieve better performance and efficiency with MTL. The first one is to design a multi-task deep neural network architecture that shares only the relevant parameters across the tasks and keeps the remaining ones task-specific. This is in contrast to the standard MTL methods that share all the layers except the last few ones across all the tasks. This heuristic is possibly suboptimal when the tasks have different characteristics and goals (*e.g.* semantically low and high-level tasks), however, searching for an optimal

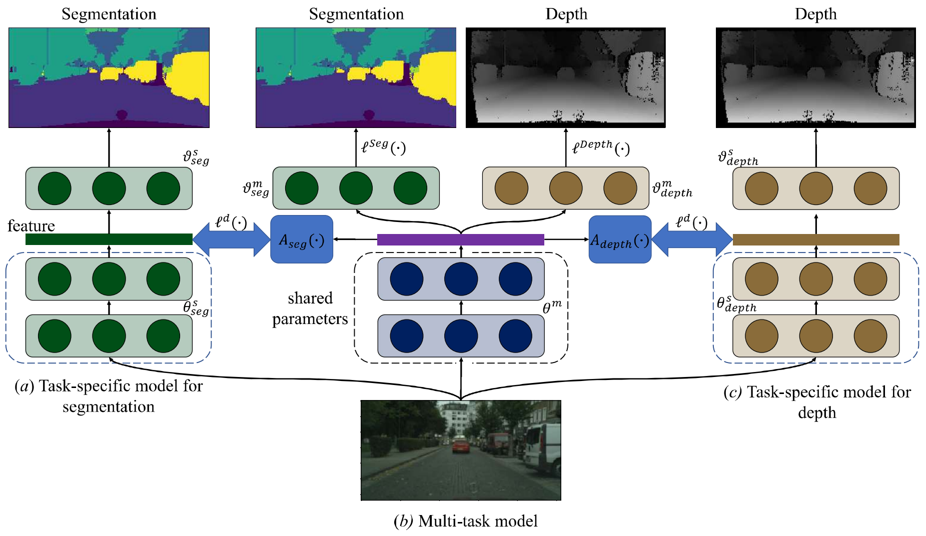

Fig. 1. Diagram of our method. We first train a task-specific model for each task in an offline stage and freeze their parameters (*i.e.* (a), (c)). We then optimize the parameters of the multi-task network for minimizing a sum of task-specific losses and also for producing similar features with the single-task networks (*i.e.* (b)). Best seen in color.

architecture in an exponential configuration space is extremely expensive. The second one is to develop MTL training algorithms that achieve good performance not only in one of the tasks but in all of them. This problem is especially important when MTL involves jointly minimizing a set of loss functions for various problems with different difficulty levels, magnitudes, and characteristics (*e.g.* cross-entropy, Euclidean loss). Thus a naive strategy of uniformly weighing multiple losses can lead to sub-optimal performances and searching for optimal weights in a continuous hyperparameter space can be prohibitively expensive.

Concerned with the second problem, previous work [4,28,12,10,16] addresses the *unbalanced loss optimization* problem with balanced loss weighting and parameter updating strategies. Kendall *et al.* [12] weigh loss functions by considering the task-dependent uncertainty of the model at training time. Sener *et al.* [28] pose the MTL as a multiple objective optimization problem and propose an approximate Pareto optimization method that uses Frank-Wolfe algorithm to solve the constrained optimization. Yu *et al.* [32] project the gradients for each loss function to a space where conflicting gradient components are removed to eliminate the disturbance between the tasks. Although the previous work improves over the uniform weighing loss strategy in MTL, they still suffer from the problem of one task dominating the remaining ones and lower task performance than the single task models in standard multi-task benchmarks.

In this paper, we approach the unbalanced MTL problem from a different point and propose a knowledge distillation based method inspired from [25,11]. As weighing the individual loss functions (*e.g.* [12]) or modifying the gradients for the loss functions by simple transformations (*e.g.* [4,32]) provide a limited control on the learned parameters and are thus limited to prevent one task dominating the rest, we propose a more strict control on the parameters of the multi-task network. Given that single-task networks often perform well with sufficient training data, we hypothesize that the solution of the multi-task network should be close to the single task ones' and lie in the intersection of the single-task solutions. To this end, we first train a task-specific model for each task in an offline stage and freeze their parameters; then optimize the parameters of the multi-task network for minimizing a sum of task-specific losses and also for producing similar features with the single-task networks. As each task-specific network can compute different features, we introduce small task-specific adaptors that map multi-task features to the task-specific one's. The adaptors align the features of the single-task and multi-task networks, and enables a balanced parameter sharing across multiple tasks.

In the remainder of this paper, we first discuss how our method relates the previous MTL and data distillation methods in Section 2, formulate our method in Section 3, demonstrate that our method outperforms the state-of-the-art MTL methods in two standard benchmarks in Section 4 and conclude the paper with future remarks in Section 5.

2 Related work

2.1 Multi-task Learning

Multi-task learning (MTL) is one of the long-standing problems in machine learning and has been used broadly [3,23,26,7,18,18,33,12,19,14]. In computer vision, MTL has been used for image classification [23], facial landmark regression [33], segmentation and depth estimation [12] and so on. In this work, we specifically focus on tackling the unbalance in the optimization of multi-task networks to achieve good performance not only in a few tasks but in all tasks.

In recent years, several methods have been proposed for solving the imbalance problem in MTL by either designing loss weighting schemes [4,12,28,16,10] to weigh each task-specific loss or modifying parameter updates [32]. Chen *et al.* [4] develop a training strategy, namely GradNorm, that looks at the gradient's norm of each task and learns the weight to normalize each task's gradient so as to balance the losses for MTL. In [28], Sener *et al.* formulate the MTL as a multiple objectives optimization problem and proposed an approximation Pareto optimization method using Frank-Wolfe algorithm to learn weights of losses. Kendall *et al.* [12] propose to weigh multiple loss functions by considering the homoscedastic uncertainty of each task during training. To design the weighting scheme, Guo *et al.* [10] observe that the imbalances in task difficulty can lead to an unnecessary emphasis on easier tasks, thus neglecting and slowing progress on difficult tasks. Based on the observation, they introduce dynamic task prioritization for MTL, which allows the model to dynamically prioritize difficult tasks during training, where the difficulty is inversely proportional to performance. Rather than weighing the losses, Yu *et al.* [32] propose a form of gradient "surgery" that projects each task's gradient onto the normal plane of the gradient of any other task and modifies the gradients for each task so as to minimize negative conflict with other task gradients during the MTL optimization.

Unlike existing methods, we propose a knowledge distillation based MTL method to solve the unbalanced loss optimization problem from a different angle. To this end, we first train a task-specific model for each task in an offline stage and freeze their parameters. We then train the MTL network for minimizing task-specific loss and also for producing the same features with the task-specific networks. As the task-specific network encodes different features, we introduce small task-specific adaptors to project multi-task features to the task-specific features. In this way, the adaptors align the task-specific feature and the multi-task feature, which enables a balanced parameter sharing across tasks.

2.2 Knowledge Distillation

Our work is also related to knowledge distillation [11,25,31,17,22]. Hinton *et al.* [11] show that distilling the knowledge of the whole ensemble of models to a neural network can achieve better performance and avoid an expensive computation. Romero *et al.* [25] introduce the knowledge distillation to training a small student network to achieve better performance than the teacher network. Apart from the success in single-task learning, knowledge distillation has also been shown to be effective in MTL. Parisotto *et al.* [21] exploits the use of deep reinforcement learning and model compression techniques to train a single policy network that learns to perform in multiple tasks by using the guidance of several expert teachers. In the contrast, in [5], Clark *et al.* extends the Born-Again network [9] to MTL setting for NLP. More specifically, they apply the knowledge distillation loss proposed in [11] on each task's predictions and propose a weight annealing strategy to update the weight of the distillation losses and multiple tasks losses.

Different from these methods, we aim at solving the unbalanced loss optimization problem in MTL. Aligning the predictions from the multi-task network and the task-specific networks would still result in unbalance as the dimension of tasks' predictions is usually different and we need to use different loss functions for matching different tasks' predictions [5], *e.g.*, a kl-divergence loss for classification and l2-norm loss for regression. In this work, we first introduce a task-specific adaptor for each task to transform features from the multi-task network and we apply the same loss function to align the transformed multi-task feature and the task-specific networks' features. We train the MTL network for minimizing task-specific loss and for producing the same feature with task-specific networks. This enables the MTL to share the parameters in a balanced way.

3 Methodology

3.1 Single-task Learning (STL)

Consider that we are given a dataset $\mathcal{D}$ that contains N training images $\boldsymbol{x}^i$ and their labels $\boldsymbol{y}_1^i, \ldots, \boldsymbol{y}_T^i$ for T tasks (*e.g.* semantic segmentation, depth estimation, surface normals). In case of the STL, we wish to learn T convolutional neural networks, one for each task, each maps the input $\boldsymbol{x}$ to the target label $\boldsymbol{y}_\tau$, *i.e.* $f(\boldsymbol{x}; \theta^s_\tau, \vartheta^s_\tau) = \boldsymbol{y}_\tau$ where the superscript s indicates the single-task, θ^s_τ and ϑ^s_τ are the parameters of the network. Each single-task network is composed of two parts: i) a feature encoder $\phi(\cdot; \theta^s_\tau)$ that takes in an image and outputs a high-dimensional encoding $\phi(\boldsymbol{x}; \theta^s_\tau) \in \mathbb{R}^{C\times H\times W}$ where C, H, W are the number channels, height and width of the feature map; ii) a predictor $\psi(\cdot; \vartheta^s_\tau)$ that takes in the encoding $\phi(\boldsymbol{x}; \theta^s_\tau)$ and predicts the output for the task τ, *i.e.* $\hat{\boldsymbol{y}}_\tau = \psi(\cdot; \vartheta^s_\tau) \circ \phi(\boldsymbol{x}; \theta^s_\tau)$ where θ^s_τ and ϑ^s_τ denote the parameters of the feature encoder and predictor respectively. The parameters for the network can be learned for each task independently by optimizing a task-specific loss function $\ell_\tau(\hat{\boldsymbol{y}}, \boldsymbol{y})$ (*e.g.* Cross-Entropy loss function for classification) over the training samples that measure the mismatch between the ground-truth label and prediction as following:

$$\min_{\theta^s_\tau, \vartheta^s_\tau} \sum_{\boldsymbol{x}, \boldsymbol{y}_\tau \in \mathcal{D}} \ell_\tau(\psi(\cdot; \vartheta^s_\tau) \circ \phi(\boldsymbol{x}; \theta^s_\tau), \boldsymbol{y}_\tau). \tag{1}$$

3.2 Multi-task Learning (MTL)

In the case of MTL, we would like to learn one network that shares the majority of its parameters across the tasks and solves all the tasks simultaneously. Similar to STL, the multi-task network can be decomposed into two parts: i) a feature encoder $\phi(\cdot; \theta^m)$ that encodes the input image into a high-dimensional encoding, now its parameters θ^m are shared across all the tasks; ii) a task-specific predictor $\psi(\cdot; \vartheta^m_\tau)$ for each task that takes in the shared encoding $\phi(\boldsymbol{x}; \theta^m)$ and outputs its prediction for task τ, *i.e.* $\psi(\cdot; \vartheta^m_\tau) \circ \phi(\boldsymbol{x}; \theta^m)$. Note that we use superscript m to denote MTL. The multi-task network can be learned by optimizing a linear combination of task-specific losses:

$$\min_{\theta^m, \vartheta^m_1, \ldots, \vartheta^m_T} \sum_{\tau=1}^{T} \sum_{\boldsymbol{x}, \boldsymbol{y}_\tau \in \mathcal{D}} w_\tau \ell_\tau(\psi(\cdot; \vartheta^m_\tau) \circ \phi(\boldsymbol{x}; \theta^m), \boldsymbol{y}_\tau) \tag{2}$$

where w_τ is a scaling hyperparameter for task τ that is used for balancing the loss functions among the tasks.

In contrast to the STL optimization in Equation (1), optimizing Equation (2) involves a joint learning of all the task-specific and shared parameters which is typically more challenging when the task-specific loss functions ℓ_τ have different characteristics such as their magnitude and dynamics (*e.g.* logarithmic, quadratic). One solution to balance the loss terms is to search for the best scaling hyperparameters w_τ by a cross-validation which has two shortcomings. First, the hyperparameter search in a continuous space is computationally expensive, especially when the number of tasks is large, as each validation step requires the training of the model. Second, even when the optimal hyperparameters can be found, it may be sub-optimal to use the same fixed ones throughout the optimization.

3.3 Knowledge Distillation for Multi-task Learning

Motivated by these challenges, the previous work [4,12,28] propose dynamic weighing strategies that can adjust them at each training iteration. Here we argue that these hyperparameters provide a limited control on the parameters of the network for preventing the unbalanced MTL and thus we propose a different view on this problem inspired by the knowledge distillation methods [25,11].

To this end, we first train a task-specific model $f(\cdot;\theta^s_\tau,\vartheta^s_\tau)$ for each task τ by optimizing Equation (1) in an offline stage, freeze their parameters and use only their feature encoders $\phi(\cdot;\theta^s_\tau)$ to regulate the multi-task network at train time by minimizing the distance between the features of task-specific networks and multi-task network for given training samples (see Figure 1). As the outputs of the task-specific encoders can differ significantly and the feature encoder of the multi-task network cannot match all of them simultaneously. Instead, we project the output of the multi-task feature encoder into each task-specific one via a task-specific adaptor $A_\tau : \mathbb{R}^{C\times H\times W} \rightarrow \mathbb{R}^{C\times H\times W}$ where H, W and C are the height, width and depth (number of channels) of the features. In our experiments, we use a linear layer that consists of a $1\times 1\times C\times C$ convolution for each adaptor. These adaptors are jointly learned along the parameters of the multi-task network to align its features with the single-task feature encoders.

$$\mathcal{L}^d = \sum_{\tau=1}^{T} \sum_{\boldsymbol{x},\boldsymbol{y}_\tau\in\mathcal{D}} \ell^d(A_\tau(\phi(\boldsymbol{x};\theta^m)),\phi(\boldsymbol{x};\theta^s_\tau)) \tag{3}$$

where ℓ^d is the Euclidean distance function between the L2 normalized feature maps:

$$\ell^d(\boldsymbol{a},\boldsymbol{b}) = \left\| \frac{\boldsymbol{a}}{||\boldsymbol{a}||_2} - \frac{\boldsymbol{b}}{||\boldsymbol{b}||_2} \right\|_2^2. \tag{4}$$

Now we can write the optimization formulation that is employed to learn the multi-task model as a linear combination of Equation (2) and Equation (3):

$$\min_{\theta^m,\vartheta^m_1,\ldots,\vartheta^m_T} \sum_{\tau=1}^{T} \sum_{\boldsymbol{x},\boldsymbol{y}_\tau\in\mathcal{D}} w_\tau\ell_\tau(\psi(\cdot;\vartheta^m_\tau)\circ\phi(\boldsymbol{x};\theta^m),\boldsymbol{y}_\tau)+\lambda_\tau\ell^d(A_\tau(\phi(\boldsymbol{x};\theta^m)),\phi(\boldsymbol{x};\theta^s_\tau)) \tag{5}$$

where λ_τ is the task-specific tradeoff hyperparameter.

Discussion. Alternatively, the inverse of each adaptor function can be thought as a mapping from each task-specific representation to a shared representation across all the tasks. The assumption here is that a large portion of encodings in the task-specific models is common to all the models up to a simple linear transformation. While the assumption of linear relations between the features of highly non-linear networks may be surprising, such linear relations have also been observed in multi-domain [24] and multi-task problems [30].

4 Experiments

4.1 Datasets

We evaluate our method on three multi-task computer vision benchmarks, including SVHN & Omniglot, NYU-v2, and Cityscapes.[1]

SVHN & Omniglot consists of two datasets, *i.e.* SVHN [20] and Omniglot [15] where SVHN is a dataset for digital number classification and Omniglot is the one for characters classification. Specifically, SVHN contains 47,217 training images and 26,040 validation images of 10 classes. Omniglot consists of 19,476 training and 6492 validation samples of 1623 categories. As the testing labels are not provided, we evaluate all methods on the validation images and report the accuracy of both tasks. Note that in contrast to the NYU-V2 and Cityscapes datasets where each image is associated with multiple labels, each image in this benchmark is labeled only for one task. Thus the goal is to learn a multi-task network that can learn both tasks from SVHN and Omniglot.

NYU-V2 [**29**] contains RGB-D indoor scene images, where we evaluate performances on 3 tasks, including 13-class semantic segmentation, depth estimation, and surface normals estimation. We use the true depth data recorded by the Microsoft Kinect and surface normals provided in [8] for depth estimation and surface normal estimation. All images are resized to 288×384 resolution as [16].

Cityscapes [**6**] consists of street-view images, each labeled for two tasks: 7-class semantic segmentation[2] and depth estimation. We resize the images to 128×256 to speed up the training.

4.2 Baselines

In this work, we use the hard parameters sharing architecture for all methods where the early layers of the network are shared across all tasks and the last layers are task-specific (See Figure 1). We compare our method with two baselines:

- **STL** learns a task-specific model for each task.
- **Uniform**: This vanilla MTL model is trained by minimizing the uniformly weighted loss Equation (2).

We also compare our method to the state-of-the-art MTL methods which are proposed for solving the unbalanced MTL, including Uncert [12], MGDA [28], GradNorm [4] and a knowledge distillation based method, namely BAM [5], that applies knowledge distillation to network's prediction. On NYU-v2 and Cityscapes, we also compare our method with Gradient Surgery (GS) [32] and Dynamic Weight Average (DWA) [16] with using different architectures, *i.e.* SegNet [1] and MTAN [16] which is the extension of SegNet by introducing task-specific attention modules for each task.

[1] The implementation of our method is available at `https://github.com/VICO-UoE/KD4MTL`.

[2] The original version of Cityscapes provides labels 19-class semantic segmentation. We follow the evaluation protocol in [16], we use labels of 7-class semantic segmentation. Please refer to [16] for more details.

4.3 Comparison to the State-of-the-art

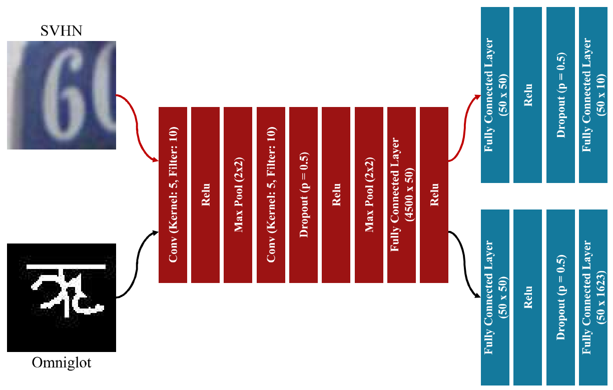
SVHN
Omniglot
Conv (Kernel: 5, Filter: 10)
Relu
Max Pool (2x2)
Conv (Kernel: 5, Filter: 10)
Dropout (p = 0.5)
Relu
Max Pool (2x2)
Fully Connected Layer (4500 x 50)
Relu
Fully Connected Layer (50 x 50)
Relu
Dropout (p = 0.5)
Fully Connected Layer (50 x 10)
Fully Connected Layer (50 x 50)
Relu
Dropout (p = 0.5)
Fully Connected Layer (50 x 1623)

Fig. 2. Network architecture used in SVHN & Omniglot.

Results on SVHN & Omniglot. First, we evaluate all methods on SVHN & Omniglot. We extend the LeNet to MTL setting (See Figure 2) and use the extended network for all methods. We set the batch size of the mini-batch as 512 where 256 samples from SVHN and 256 images from Omniglot. We use Adam [13] for optimizing the networks and adaptors. The learning rate of all task-specific adaptors is 0.01. We train all methods for 300 epochs in total where we scaled the learning rate by 0.85 every 15 epochs. In our method, weights of task-specific losses (*i.e.* w in Equation (5)) are set uniformly. As a validation set for hyperparameter search (λ), we randomly pick 10% of training data. After the best hyperparameters are chosen, we retrain with the full training set and report the median validation accuracy of the last 20 epochs in Table 1. We search over the set $\lambda = \{1, 5, 10, 20\}$ of λ and we chose $\lambda = 10$.

Table 1. Testing accuracy on SVHN & Omniglot.

Type	Methods	SVHN	Omniglot	avg
STL	- -	88.84	65.76	- -
MTL	Uniform	85.88	66.91	76.40
	Uncert [12]	85.43	64.1	74.77
	MGDA [28]	83.93	66.8	75.37
	GradNorm [4]	84.48	65.55	75.02
	BAM [5]	86.57	66.08	76.33
	Ours	**88.05**	**70.12**	**79.09**

Table 1 shows that MTL with uniform loss weights (Uniform) obtains worse results on SVHN while it achieves better performance on Omniglot than STL. The

Table 2. Testing results on NYU-v2. * Results of the 'GS' method are from [32].

Architecture Type		Methods	Segmentation		Depth		Surface Normal				
			(Higher Better ↑)		(Lower Better ↓)		Angle Distance (Lower Better ↓)		Within t° (Higher Better ↑)		
			mIoU	Pix Acc	Abs Err	Rel Err	Mean	Median	11.25	22.5	30
SegNet	STL	- -	17.32	55.70	0.6577	0.2828	29.99	23.81	24.31	48.06	60.05
	MTL	Uniform	18.14	55.82	0.5841	0.2490	31.89	26.81	20.61	42.95	55.33
		Uncert [12]	16.79	52.51	0.6183	0.2612	32.44	27.27	20.81	42.41	54.47
		MGDA [28]	15.57	49.05	0.6716	0.2710	29.90	24.13	23.81	47.58	59.75
		GradNorm [4]	18.08	55.76	0.5819	0.2495	30.65	25.38	22.64	45.34	57.59
		Ours	**18.75**	**58.02**	**0.5780**	**0.2467**	**29.40**	**23.71**	**24.33**	**48.22**	**60.45**
MTAN [16]	STL	- -	16.38	53.89	0.6792	0.2963	30.66	24.26	23.35	47.34	59.33
	MTL	Uniform	17.72	55.32	0.5906	0.2577	31.44	25.37	23.17	45.65	57.48
		DWA [16]	17.52	55.76	0.5869	0.2549	31.75	25.64	22.60	45.12	57.05
		Uncert [12]	17.67	55.61	0.5927	0.2592	31.25	25.57	22.99	45.83	57.67
		MGDA [28]	15.60	52.36	0.6215	0.2767	30.26	24.01	23.98	47.77	59.85
		GradNorm [4]	17.37	55.92	0.5924	0.2630	31.20	24.91	23.11	46.27	58.20
		GS* [32]	20.17	56.65	0.5904	0.2467	30.01	24.83	22.28	46.12	58.77
		Ours	**20.75**	**57.90**	**0.5816**	**0.2445**	**29.97**	**23.96**	**24.24**	**47.78**	**59.78**

state-of-the-art methods which dynamically weigh the task-specific losses cannot achieve a good trade-off between these two tasks. More specifically, Uncert and GradNorm obtain worse overall performance than STL while MGDA improves the performance on Omniglot and obtains worse performance on SVHN. Though BAM obtains better overall performance on both tasks, the improvement is achieved mainly because of more informative information provided by the continuous predictions of the teacher network (task-specific models). The unbalanced problem in MTL when we use BAM is still unsolved as BAM applies knowledge distillation on network predictions, which would have similar problems with the vanilla MTL (Uniform). In contrast, our approach achieves significantly better performance than any other MTL methods *i.e.* our method obtains 88.05 % accuracy on SVHN and 70.12 % accuracy on Omniglot. Compared with STL, our method obtains comparable results on SVHN and significant gains on Omniglot. These results strongly verify that our method is able to alleviate the unbalanced problem in this benchmark and to outperform STL as it enables the MTL model to learn more informative features.

Results on NYU-V2. We follow the training and evaluation protocol in [16]. We use cross-entropy loss for semantic segmentation, l1-norm loss for depth estimation, and cosine similarity loss for surface normal estimation. We train all methods using Adam [13] with the learning rate initialized at 1e-4 and halved at the 100-th epoch for 200 epochs in total. The learning rate of all task-specific adaptors is 0.1 and the batch size is 2. In our method, weights of task-specific losses (*i.e.* w in Equation (5)) are set uniformly. As a validation set for hyperparameter search (λ), we randomly pick 10% of training data. After the best hyperparameters are chosen, we retrain with the full training set and report the results of three tasks on the validation set in Table 2. We search over the set $\lambda = \{1, 2, 3, 4, 5, 6\}$ of λ and the distillation loss' weight of Segmentation, depth, and surface normal are set to 1, 1 and 2, respectively.

Table 3. Testing results on Cityscapes.

Architecture	Type	Methods	Segmentation (Higher Better ↑)		Depth (Lower Better ↓)	
			mIoU	Pix Acc	Abs Err	Rel Err
SegNet	STL	- -	51.85	91.08	0.0136	22.68
	MTL	Uniform	50.73	90.76	0.0151	40.81
		Uncert [12]	51.09	90.85	0.0143	27.66
		MGDA [28]	51.69	90.99	**0.0130**	**24.04**
		GradNorm [4]	50.06	90.82	0.0143	28.61
		Ours	**52.18**	**91.24**	0.0140	28.90
MTAN	STL	- -	51.24	91.16	0.0137	24.80
	MTL	Uniform	52.56	91.33	0.0152	24.64
		DWA [16]	51.95	91.33	0.0141	30.03
		Uncert [12]	50.37	91.11	0.0142	31.78
		MGDA [28]	52.32	**91.59**	**0.0138**	30.35
		GradNorm [4]	51.88	91.40	0.0148	31.43
		Ours	**52.71**	91.54	0.0139	**27.33**

From the results shown in Table 2, we can see that it is possible to tackle multiple tasks within a network and achieve performance improvement on some tasks, *e.g.* when we use SegNet as the based network, the vanilla MTL (Uniform) achieves better performance on semantic segmentation and depth estimation though it causes a drop on surface normal estimation in comparison with STL. Though we see the benefits of using MTL, it is also clear that the unbalanced problem exists. We then apply existing methods that introduce loss weighting strategies for addressing the unbalanced loss optimization problem. From the results of using SegNet, GradNorm performs the best among all compared methods. However, it provides limited control on the learned parameters (*e.g.* it achieves similar results to the MTL model using a uniformly weighting scheme) and still suffers from lower task performance than the single task models.

In comparison with these methods, our method obtains significant gains over all tasks and achieves better results than single task learning models. This strongly verifies our hypothesis that the solution of the multi-task network should be close to the single task ones' and lie in the intersection of the single-task solutions and our method that applies stricter control on the parameters of the multi-task network can better address the unbalanced loss problem.

Results on Cityscapes. Similar to NYU-V2, we use cross-entropy loss for semantic segmentation and l1-norm loss for depth estimation in Cityscapes as in [16]. We train all methods using Adam [13] with the learning rate initialized at 1e-4 and halved at the 100-th epoch for 200 epochs in total. The learning rate of all task-specific adaptors is 0.1 and the batch size is 8. In our method, weights of task-specific losses (*i.e.* w in Equation (5)) are set uniformly. As a validation set for hyperparameter search (λ), we randomly pick 10% of training data. After the best hyperparameters are chosen, we retrain with the full training set and report the results of two tasks on the validation set in Table 3. We search over

Table 4. Ablation study on NYU-v2. Here, '#layer' means which layers are selected for computing distillation loss. 'adaptors' indicates which kind of adaptors is used and '✗' means no adaptors are used.

Method			Segmentation		Depth		Surface Normal				
			(Higher Better ↑)		(Lower Better ↓)		Angle Distance (Lower Better ↓)		Within t° (Higher Better ↑)		
backbone	#layers	adaptors	mIoU	Pix Acc	Abs Err	Rel Err	Mean	Median	11.25	22.5	30
SegNet	last	linear	18.66	57.78	0.5813	**0.2375**	30.17	24.74	22.96	46.44	58.76
	mid + last	linear	18.75	58.02	**0.5780**	0.2467	**29.40**	**23.71**	**24.33**	**48.22**	**60.45**
	mid + last	non-linear	**19.00**	**58.12**	0.5853	0.2398	29.74	24.25	23.06	47.16	59.53
	mid + last	✗	17.11	54.60	0.6060	0.2586	30.32	24.74	22.66	46.36	58.72
MTAN [16]	last	linear	18.52	56.81	**0.5756**	0.2489	31.13	24.93	23.52	46.29	58.18
	mid + last	linear	**20.75**	**57.90**	0.5816	**0.2445**	**29.97**	**23.96**	**24.24**	**47.78**	**59.78**
	mid + last	non-linear	20.42	56.93	0.5975	0.2589	30.32	24.86	22.98	46.21	58.55
	mid + last	✗	19.30	53.85	0.6064	0.2567	31.43	27.61	17.61	40.86	54.63

the set $\lambda = \{1, 2, 3, 4, 5, 6\}$ of λ and the distillation loss' weight of Segmentation and depth are set to 2 and 6, respectively.

As shown in Table 3, in overall, MTL obtains worse performance than STL. It is clear that GradNorm obtains worse performance on semantic segmentation while it improves the performance on depth estimation. However, MGDA assigns much larger weight on depth estimation task and this enables the MTL model to achieve better performance on both tasks. Our method also achieves significant gains on both tasks. The results again demonstrate that our method is able to optimize MTL model in a more balanced way and to achieve better overall results.

4.4 Ablation Study

To better analyze the effect of the distillation loss, we conduct an ablation study on NYU-V2. We first evaluate the effect of applying distillation loss to more layer's features. On NYU-V2, we report the results of applying distillation loss to the last shared layer's feature only and the results of applying distillation loss to features of both the middle layer and the last layer. From the results presented in Table 4, adding more layers' features for computing distillation loss boosts the performance on NYU-V2 in general. However, results on SVHN & Omniglot and Cityscapes indicate that using the last layers obtains the best performance. We argue that adding more layers can enhance the distillation loss and a more strict control on the parameters of the multi-task network. This is not necessary for those tasks that use a small network, *e.g.* the network we used in SVHN & Omniglot and would be useful for tasks using a large network, *e.g.* the SegNet used in NYU-V2.

Analysis on Adaptors. We evaluate our method using different types of adaptors. As mentioned in Section 3, we use a linear layer that consists of a $1 \times 1 \times C \times C$ convolution for each adaptor and denote as 'linear' in Table 4. We also evaluate our method without any adaptors (*i.e.* indicated as '✗') and with 'non-linear 'adaptors (*i.e.* each adaptors consists of a $1 \times 1 \times C \times 2C$ convolution,

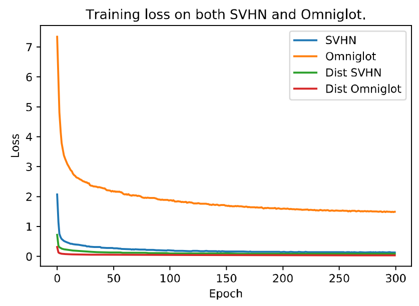

Fig. 3. Task-specific loss and distillation loss on training set of SVHN & Omniglot. Best view in color.

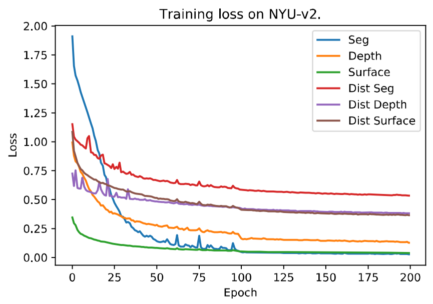

Fig. 4. Task-specific loss and distillation loss on training set of NYU-V2. Best view in color.

a Relu activation layer and a $1 \times 1 \times 2C \times C$ convolution). From the results shown in Table 4, it is clear to see that the adaptors help to align the features of multi-task and single-task as each single-task model produce different features. Compared with our method using 'non-linear' adaptors, which have much larger capacity of mapping features, the results indicate that using 'linear' adaptors is sufficient as we discuss in Section 3.

Further Analysis. We also plot the task-specific loss and distillation loss on the training set of two benchmarks for analyzing our method. In both Figure 3 and Figure 4, it is clear that distillation loss is more balanced than task-specific loss. More specifically, the task-specific loss of SVHN and Omniglot converge at around 0.137 and 1.492, respectively. In contrast, the distillation loss of SVHN and Omniglot converge at around 0.089 and 0.041, respectively. On NYU-V2, the task-specific loss of semantic segmentation, depth estimation, and surface normal estimation end up at 0.027, 0.127, and 0.039 while the distillation loss ends up at around 0.534, 0.381, and 0.364. These results again verify that our method can optimize the MTL method in a more balanced way.

5 Conclusion

In this work, we proposed a knowledge distillation based multi-task method that learns to produce the same features with the single-task networks to address the unbalanced multi-task learning problem with the hypothesis that the solution of the multi-task network should be close to the single task ones' and lie in the intersection of the single solutions. We demonstrated that our method achieves significant performance gains over the state-of-the-art methods, on challenging benchmarks for image classification and scene understanding (semantic segmentation, depth estimation, and surface normal estimation). As future work, we plan to extend our method to multi-task network architecture searching.

References

1. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. Transactions on Pattern Analysis and Machine Intelligence **39**(12), 2481–2495 (2017)
2. Bilen, H., Vedaldi, A.: Integrated perception with recurrent multi-task neural networks. In: Advances in Neural Information Processing Systems. pp. 235–243 (2016)
3. Caruana, R.: Multitask learning. Machine learning **28**(1), 41–75 (1997)
4. Chen, Z., Badrinarayanan, V., Lee, C.Y., Rabinovich, A.: Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. arXiv preprint arXiv:1711.02257 (2017)
5. Clark, K., Luong, M.T., Khandelwal, U., Manning, C.D., Le, Q.V.: Bam! born-again multi-task networks for natural language understanding. arXiv preprint arXiv:1907.04829 (2019)
6. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Computer Vision and Pattern Recognition. pp. 3213–3223 (2016)
7. Du, Y., Czarnecki, W.M., Jayakumar, S.M., Pascanu, R., Lakshminarayanan, B.: Adapting auxiliary losses using gradient similarity. arXiv preprint arXiv:1812.02224 (2018)
8. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: IEEE International Conference on Computer Vision. pp. 2650–2658 (2015)
9. Furlanello, T., Lipton, Z.C., Tschannen, M., Itti, L., Anandkumar, A.: Born again neural networks. arXiv preprint arXiv:1805.04770 (2018)
10. Guo, M., Haque, A., Huang, D.A., Yeung, S., Fei-Fei, L.: Dynamic task prioritization for multitask learning. In: European Conference on Computer Vision. pp. 270–287 (2018)
11. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
12. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 7482–7491 (2018)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
14. Kokkinos, I.: Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 6129–6138 (2017)
15. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: Human-level concept learning through probabilistic program induction. Science **350**(6266), 1332–1338 (2015)
16. Liu, S., Johns, E., Davison, A.J.: End-to-end multi-task learning with attention. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1871–1880 (2019)
17. Ma, J., Mei, Q.: Graph representation learning via multi-task knowledge distillation. arXiv preprint arXiv:1911.05700 (2019)
18. Meyerson, E., Miikkulainen, R.: Pseudo-task augmentation: From deep multitask learning to intratask sharing—and back. arXiv preprint arXiv:1803.04062 (2018)
19. Misra, I., Shrivastava, A., Gupta, A., Hebert, M.: Cross-stitch networks for multi-task learning. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 3994–4003 (2016)

20. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: Advances in Neural Information Processing Systems Workshop on Deep Learning and Unsupervised Feature Learning (2011)
21. Parisotto, E., Ba, J.L., Salakhutdinov, R.: Actor-mimic: Deep multitask and transfer reinforcement learning. arXiv preprint arXiv:1511.06342 (2015)
22. Phuong, M., Lampert, C.: Towards understanding knowledge distillation. In: International Conference on Machine Learning. pp. 5142–5151 (2019)
23. Rebuffi, S.A., Bilen, H., Vedaldi, A.: Learning multiple visual domains with residual adapters. In: Advances in Neural Information Processing Systems. pp. 506–516 (2017)
24. Rebuffi, S.A., Bilen, H., Vedaldi, A.: Efficient parametrization of multi-domain deep neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 8119–8127 (2018)
25. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550 (2014)
26. Ruder, S.: An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098 (2017)
27. Seltzer, M.L., Droppo, J.: Multi-task learning in deep neural networks for improved phoneme recognition. In: IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 6965–6969. IEEE (2013)
28. Sener, O., Koltun, V.: Multi-task learning as multi-objective optimization. In: Advances in Neural Information Processing Systems. pp. 527–538 (2018)
29. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: European conference on computer vision. pp. 746–760. Springer (2012)
30. Stickland, A.C., Murray, I.: Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. arXiv preprint arXiv:1902.02671 (2019)
31. Tian, Y., Krishnan, D., Isola, P.: Contrastive representation distillation. arXiv preprint arXiv:1910.10699 (2019)
32. Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., Finn, C.: Gradient surgery for multi-task learning. arXiv preprint arXiv:2001.06782 (2020)
33. Zhang, Z., Luo, P., Loy, C.C., Tang, X.: Facial landmark detection by deep multi-task learning. In: European conference on computer vision. pp. 94–108. Springer (2014)