Online Knowledge Distillation for Multi-task Learning

Geethu Miriam Jacob Vishal Agarwal Björn Stenger
Rakuten Institute of Technology, Rakuten Group, Inc.
{geethu.jacob, vishal.agarwal, bjorn.stenger}@rakuten.com

Abstract

Multi-task learning (MTL) has found wide application in computer vision tasks. We train a backbone network to learn a shared representation for different tasks such as semantic segmentation, depth- and normal estimation. In many cases negative transfer, i.e. *impaired performance in the target domain, causes the MTL accuracy to be lower than training the corresponding single-task networks. To mitigate this issue, we propose an online knowledge distillation method, where single-task networks are trained simultaneously with the MTL network to guide the optimization process. We propose selectively training layers for each task using an adaptive feature distillation (AFD) loss with an online task weighting (OTW) scheme. This task-wise feature distillation enables the MTL network to be trained in a similar way to the single-task networks. On the NYUv2 and Cityscapes datasets we show improvements over a baseline MTL model by 6.22% and 9.19%, respectively, outperforming recent MTL methods. We validate the design choices in ablative experiments, including the use of online task weighting and the adaptive feature distillation loss.*

1. Introduction

Multi-task learning (MTL) helps scaling real-world applications, where multiple tasks need to be solved simultaneously. MTL has been used extensively in the domain of computer vision [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13], NLP [14, 15, 16] and speech recognition [17, 18]. Specifically, we focus on pixel-wise prediction tasks, such as semantic segmentation, depth estimation, and surface normal estimation from images. Most existing MTL techniques rely on a branched architecture, where the majority of parameters are shared between all tasks, with fewer parameters allocated exclusively to each task. Such a shared network architecture is memory-efficient and increases inference speed. However, MTL models often exhibit worse performance compared to the corresponding single task models. This problem is referred to as "negative transfer", in which the performance improvement in one task leads to the performance degradation of another. A number of techniques have been proposed to mitigate this issue, including task weighting [10, 19, 20], feature fusion [1, 2], feature selection [21], task affinity [22] and knowledge distillation [8, 9, 23, 24].

Vision transformers (ViT) have emerged as a successful technique for many tasks, such as image classification [25, 26], object detection [3, 27] and pixel-wise prediction problems such as depth estimation [28, 29] and semantic segmentation [30]. Transformer-based techniques for multi-task scene understanding have been proposed recently [31, 32, 33]. Owing to their excellent performance across different tasks, we also use a ViT-based MTL architecture with a shared backbone and task-specific heads to learn multiple tasks simultaneously.

Knowledge distillation methods have shown promising results when applied to multi-task learning [8, 9], where current state-of-the-art methods use pre-trained models as teacher models. An online distillation method ('rocket launching') was proposed by Zhou *et al*. [34] for learning light-weight models,where simultaneous training of booster and light networks is performed. By simultaneous training, the booster network transfers knowledge and guides the light-weight model via a hint loss throughout the training process. Inspired by this, we propose training an MTL network with multiple single task networks, where the single task networks guide the MTL network throughout the training process. This leads to reduced inference time and memory requirements, while achieving comparable accuracy to single-task models. The difference of the proposed method to alternative knowledge distillation approaches [8, 9] is the simultaneous training of single task and multi-task models with the help of adaptive feature distillation (AFD) and online task weighting (OTW), compared to the use of pre-trained models.

The AFD component selectively distills feature knowledge from single-task models to the multi-task model in each iteration during training. The idea of adaptive feature distillation is motivated by the observation that different layers of the shared backbone contribute differently to each

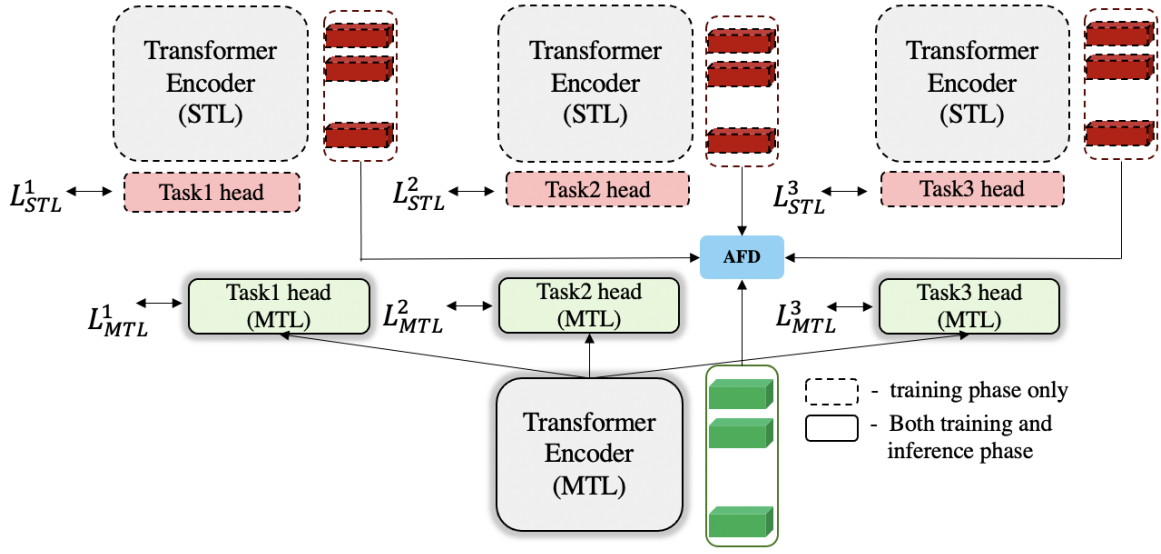

Figure 1: Model Overview. *Multiple single-task models are trained simultaneously along with the multi-task model. Task-specific losses are used to train the task heads of both the single task (L^i_{STL}) and multi-task models (L^i_{MTL}). An adaptive feature distillation loss (AFD) is used between intermediate features of single models and the multi-task model. At inference time, only the MTL model is used.*

task [21]. Since the single-task and MTL models are trained together, the training of the MTL model for each task can be monitored with respect to that of single-task models. In addition to feature weighting, we also introduce an online task weighting (OTW) scheme, where weights are based on the loss values of both single and multi-task models at each iteration. The overall framework is shown in Fig. 1.

In summary, the contributions in this paper are (1) a novel simultaneous training strategy of single- and multi-task models with online knowledge distillation, (2) using task weights computed from single-task and multi-task loss values, and (3) using an adaptive feature distillation strategy. We demonstrate the effectiveness of this approach in experiments on two public datasets, and evaluate the design choices in ablation studies.

2. Prior Work

Our method majorly explores the use of Vision Transformers for Multi-task Learning (MTL) with the help of online knowledge distillation. Here, we discuss related work on multi-task networks and knowledge distillation.

Multi-task learning. The aim of multi-task learning (MTL) is to exploit information in the training data of related tasks to learn a shared representation [1, 19, 35, 36, 37, 38, 39, 40]. A recent survey paper discusses various approaches to multi-task learning [41]. In computer vision, MTL has been used for a wide range of tasks, including image classification [36], facial landmark localization [39], scene understanding tasks [19].

Several MTL techniques for scene understanding have been proposed. Task weighting schemes estimate suitable weights for combining the loss function of each task [6, 7, 10, 11, 19, 20]. Gradient-based methods, such as GradNorm [4], change the gradient magnitudes of tasks to regularize training. PCGrad [42] modifies the gradient in order to project a task gradient onto the normal plane of the gradient of any other task that has a conflicting gradient. This helps to mitigate the negative transfer, commonly found in MTL [4, 42]. Another approach taken in [1, 2] is to first train separate single-tasks models and then fuse the intermediate features to obtain better generalization. Attention-based methods, *e.g.*, [20], as well as other feature weighting methods [21, 43] have been proposed, which compute the importance of feature layers for each task.

Recently, a number of different weighting schemes for task losses have been introduced. Guo *et al*. [5] observed that the imbalances in task difficulty can lead to an undesired emphasis on easier tasks, leading to slower progress on difficult tasks. To allow the model to dynamically prioritize difficult tasks with larger weights, dynamic task prioritization is proposed. Similarly, Kendall *et al*. [19] weigh the losses of multiple tasks using task-dependent homoscedastic uncertainty of model outputs during training. Sener *et al*. [10] formulate MTL as a multi-objective optimization task and propose Pareto optimization using the Frank-Wolfe algorithm to learn weights of losses. A dynamic weight averaging (DWA) strategy based on the progression of loss for each task is computed in [20]. As an extension to DWA, the method in [11] divides the learning process into three phases. The first phase uses equal weights for all tasks, the

second phase computes weights based on the task-level loss and overall loss, and the third phase uses a difficulty factor based on the loss weights of the previous two phases. The magnitude of the loss gradient in each iteration is considered as the task weight in [6]. A low gradient magnitude signifies that the task is being learned correctly, and the corresponding weight may be decreased. On the other hand, a large gradient magnitude suggests abrupt training for the task and requires further attention. Other work proposed cross-affinity patterns [44] or pattern structures using graphlets [22], and propagate them among and across tasks. A few recent works [23, 24, 45, 22, 33] perform multi-task learning in stages, where a multi-task network is first employed for initial predictions, the features from which are used to obtain predictions for other tasks. We propose an online knowledge distillation based task weighting strategy, where the MTL network is encouraged to have similar loss to that of the STL network.

Knowledge distillation (KD). Our work is related to Knowledge distillation [46, 47, 48, 49, 50]. Hinton *et al*. [46] show that knowledge from ensemble networks can be distilled to a neural network, thus helping in achieving better performance with lower inference time. Romero *et al*. [47] extended this work by introducing various hint losses for knowledge distillation. Knowledge distillation has also been used in various MTL techniques as well. The works by Parisotto *et al*. [51], Clark *et al*. [52] use deep reinforcement learning and model compression to train a single network that learns to perform multiple tasks using knowledge distillation. Rocket launching, which simultaneously trains a lighter network along with large (booster) network, is proposed in [34]. The work in [34] proposed joint training of booster and lighter networks with knowledge distillation, for model compression. In this paper we propose online knowledge distillation for training a multi-task network, with similar computation and memory requirements as that of single task networks. We also design task weights based on the loss values of single tasks and that of MTL.

KD methods have recently been shown to perform well for multi-task learning on scene understanding tasks [8, 9, 53, 54]. KD-MTL [8] and multi-teacher knowledge distillation [53] trains in two stages, where the first stage trains single-task models, and the second stage trains the MTL model by distilling knowledge from the penultimate feature layer of each single-task model. The method in [54] uses a self-distillation loss, where knowledge is distilled from the same network for MTL. A self-coordinated knowledge amalgamation network is proposed in [9], where the student learns from the heterogeneous pre-trained teachers. Some work proposed to distill information from the initial predictions of other tasks of the same multi-task network at a single scale [23] or multiple scales [24]. In contrast to these papers, which employ pre-trained models or use predictions of auxiliary tasks for KD to train a multi-task model, our method trains both single and multi-task networks jointly.

Vision Transformers. Transformers have been proposed for the tasks of image classification [25, 26], visual question answering (VQA) [55, 56], object detection [3, 27], semantic segmentation [30, 57], and depth estimation [28, 29]. Multi-task models with transformers have been proposed in the field of natural language processing. Recently, transformer-based methods have been introduced for multi-modal tasks, involving both language and vision [12, 13]. This is concurrent with recent work [33, 31, 32], which explores multi-task learning with transformers. While the method [33] incorporates multi-scale aggregation and self-attention message passing to produce task-specific prediction at a high resolution, the method [31] uses task-specific queries and cross-task attention module in transformers for multi-task learning. Meanwhile, [32] explores the problem exploring the generalization of multitask transformers to unseen domains. We propose a backbone agnostic knowledge distillation training method for multi-task learning ad focus on the design of the transformer architecture for image prediction tasks, owing to the success of the transformer models.

3. Method

Our framework, shown in Fig. 1, consists of a multi-task network with a shared Vision Transformer (ViT) [25] backbone and separate heads for N tasks. The architecture also consists of single task networks with single heads and ViT backbone, specific to one task. We propose a training strategy in which we train single-task models and a multi-task model simultaneously on N_t tasks. The single-task networks guide the optimization of the multi-task network throughout the training process. The multi-task network weights are tied to the single-task networks through distillation losses on intermediate features. The model is trained in an end-to-end fashion by minimizing the following loss function:

$$L = L_{\mathrm{AFD}} + \sum_{i=1}^{N_t} \left(L^i_{\mathrm{STL}} + \lambda_i L^i_{\mathrm{MTL}} \right), \qquad (1)$$

where L^i_{STL} denotes the task-specific loss for the i^{th} single-task network. L^i_{MTL} is the task-specific loss for the i^{th} head of the multi-task network (sec. 3.2) and L_{AFD} denotes the adaptive knowledge distillation loss (sec. 3.3.1) between features of the single-task and multi-task networks. The loss weights, $\lambda_i, i = 1, 2, ..., N_t$, are calculated at each training iteration for each task based on the loss values of single- and multi-task models (sec. 3.3.2).

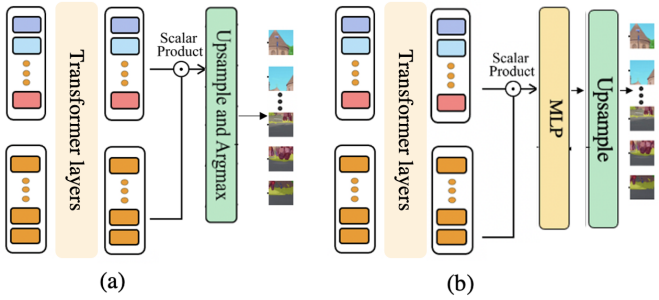

Figure 2: Task head details. *Architectural details of (a) classification head and (b) regression head. Patch embeddings from the transformer encoder along with class/regression queries are passed through two transformer layers. The classification head upsamples and takes the argmax of the embeddings, whereas the regression head passes the embeddings through MLP and upsamples them to obtain predictions.*

3.1. Model Architecture

The pretrained ViT-tiny [25] model is used as the backbone for transformer models. The multi-task model consists of a shared backbone with N heads, and the single models each have their separate backbone network and a single head. The input images are partitioned into P patches of 16×16 pixels. Each patch is mapped via a patch embedding network and passed to the ViT encoder. After dividing an image of size $W \times H$ into patches, the size of the patch grid is $W/16 \sim W_{patch} \times H/16 \sim H_{patch}$. The patches are passed to a linear embedding network and the $N_{patch} = W_{patch} \times H_{patch}$ embeddings are input to the transformer encoders of both single and multi-task networks.

As part of a system for visual scene understanding we consider multiple pixel-wise classification and regression tasks. We adopt the design of the Mask Transformer [30] for the pixel-wise classification task (Fig. 2 (a)). Semantic segmentation is an example of a pixel-wise classification task. Let E be the embedding size of patches and tokens passed to the transformer layers. The patches extracted from the ViT encoder (of dimension $N_{patch} \times E$) backbone are passed to the pixel-wise classification head. Class queries (of dimension $N_{cls} \times E$) are introduced along with the patch embeddings and passed to two transformer layers. The number of class queries (N_{cls}) are taken as the number of classes for classification task. A scalar product of class queries and patch embeddings is computed (of output dimension, $N_{patch} \times N_{cls}$). The output is then upsampled and reshaped to the image size. We obtain N_{cls} class maps, each of the same size as the input image. Pixel labels are estimated as the argmax of the N_{cls} class maps.

Pixel-wise depth and surface normal prediction are formulated as regression tasks. We use a similar design as that of the classification head, see Fig 2 (b). Let N_{reg} (empirically set as 128) be the number of query embeddings passed along with N_{patch} embeddings, each of dimension E, to the regression head with two transformer layers. A scalar product of the output queries and patches from transformer layers is computed (of output dimension, $N_{patch} \times N_{reg}$) and passed through an MLP block. We use a sequence of linear layers in the MLP block, with the output dimension of the last linear layer according to the task. The output dimension for depth estimation is $N_{patch} \times 1$, whereas that of surface normal estimation is $N_{patch} \times 3$. The output is upsampled and reshaped to the image size.

3.2. Task-specific losses

Given the different nature of the tasks, we use task-specific loss functions. We use cross-entropy loss for semantic segmentation (L^1_{MTL}), L1 loss for depth estimation [58] (L^2_{MTL}), and cosine similarity loss for surface normal estimation (L^3_{MTL}), similar to that of [7, 8, 20]. Both multi-task (L^i_{MTL}) and single task (L^i_{STL}) networks are trained using the same loss functions.

3.3. Online Knowledge Distillation

A well-known challenge in multi-task learning is 'negative transfer'. To mitigate this problem, we propose online knowledge distillation. The intuitive idea behind this is that the optimization of multiple single task models guide the multi-task model during the training phase. The knowledge of features of the single task transformer encoders are distilled to that of the multi-task model in each iteration. We propose two components for online distillation, Adaptive Feature Distillation (AFD) and Online Task Weighting (OTW). To provide a good starting point for our MTL network, we use a warm-up training phase of the single-task models for 5 epochs each.

3.3.1 Adaptive feature distillation (AFD)

The first component of the proposed online knowledge distillation is adaptive feature distillation (AFD), a method for sharing intermediate features of the backbone models. We perform online weighted knowledge distillation on intermediate features from the shared backbone of the multi-task network. Let L denote the number of layers in the shared transformer encoder, ω^l_i the learnable parameter for the i^{th} task in the l^{th} layer. The AFD loss, L_{AFD}, is defined as:

$$L_{\mathrm{AFD}} = \sum_{l=1}^{L} \left|\left| \mathbf{f}_{\mathrm{MTL}}(l) - \sum_{i=1}^{N_t} \omega^l_i \mathbf{f}^i_{\mathrm{STL}}(l) \right|\right|^2 \qquad (2)$$

where $\mathbf{f}_{MTL}(l)$ are features extracted from the l^{th} layer of the shared MTL backbone, $\mathbf{f}^i_{STL}(l)$ are the l-layer features from the i^{th} single-task model. The degree of alignment of the STL features of each task towards the MTL features

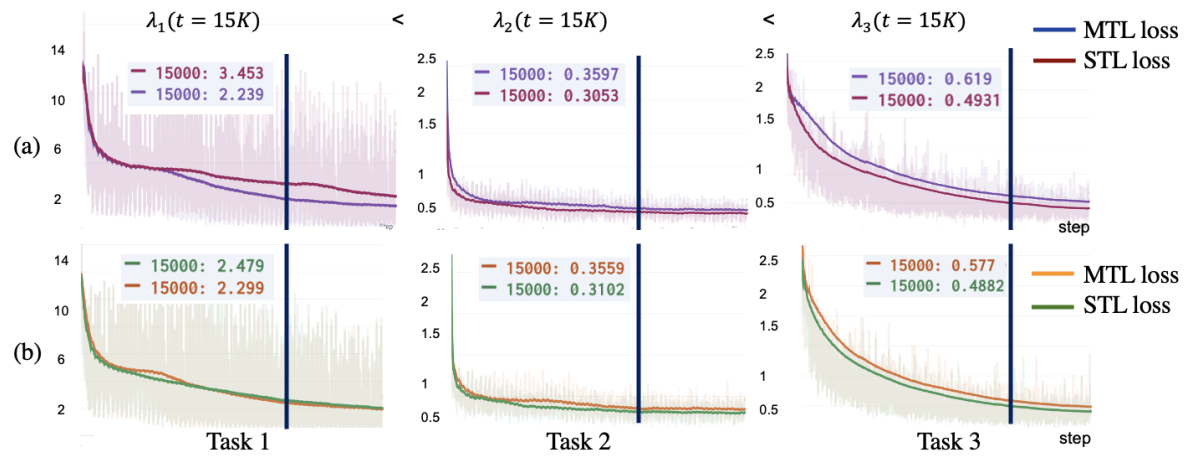

Figure 3: Illustration of Online Task Weighting (OTW) *showing smoothed task loss functions of MTL and STL networks for (a) online distillation without OTW, (b) online distillation with OTW. Loss function values at a specific iteration ($t = 15K$) are shown, where the STL loss value is higher than the MTL loss value for task 1, whereas for both task 2 and task 3 the STL loss remains lower than the MTL loss. Task 3 has the largest difference at this time step, resulting in the largest task weight. Online task weighting reduces the gap between the MTL and STL losses.*

is decided by the parameter, ω_i^l. The AFD function ensures that the feature space of the MTL network is aligned with that of STL networks. This function ensures cross-task learning from the STL features. The parameter ω_i^l is learned along with the network parameters using the same optimizer and scheduler. Note that we stop the gradients from the AFD function from being back-propagated to the STL networks, as suggested in [34]. Otherwise the STL network performance may be impaired by the MTL parameters. We therefore detach the single tasks tensors ($\mathbf{f}_{STL}^i$) from the computational graph, while computing the AFD function.

3.3.2 Online task weighting (OTW)

Secondly, we propose the use of task weights during simultaneous learning of single and multi-task networks. The multi-task network is trained using a linear combination of task-specific losses, where task weights are based on the performance of the multi-task model with respect to the single task models. Let the multi-task model loss at any iteration t be $L_{\text{MTL}}^i(t)$ and the single task loss $L_{\text{STL}}^i(t)$ for the i^{th} task. The task weight for the i^{th} task at iteration t is computed as a temperature-scaled softmax function of the ratio of multi-task to single-task loss:

$$\lambda_i(t) = N_t \frac{\exp(\frac{m_t^i}{\mathcal{T}})}{\sum_{j=1}^{N_t} \exp(\frac{m_t^j}{\mathcal{T}})}, \quad m_t^i = \frac{L_{\text{MTL}}^i(t)}{L_{\text{STL}}^i(t)}. \tag{3}$$

Higher weights are given to tasks for which the multi-task loss is larger than the corresponding single-task loss. $\mathcal{T}$ represents a temperature term which controls the task weighting [46]. A large $\mathcal{T}$ results in a more evenly distributed weights for different tasks. N_t is number of tasks in the softmax function and ensures that $\sum_{i=1}^{N_t} \lambda_i = N_t$. Empirically, we set $\mathcal{T} = 0.1$.

The effect of OTW in training is illustrated in Fig 3. The plots show the task-specific loss STL and MTL functions during the training process. Using online task weighting, the difference between MTL and STL losses during training is reduced.

4. Results

The proposed method is evaluated on two public datasets, *NYUv2* [59] and *Cityscapes* [60]. The NYUv2 dataset [59] contains 1,449 densely labeled images, collected from a variety of indoor scenes with an RGBD sensor. The images were hand selected from 435,103 video frames to ensure diverse scene content. We use the same training and test split as in the original work and evaluate performance on three learning tasks: semantic segmentation (13 labels), depth estimation, and surface normal estimation. The Cityscapes [60] dataset contains high-resolution street-view images for semantic segmentation and depth estimation. It contains 2,975 images for training and 500 for testing, respectively. The images are sparsely sampled from video clips. This dataset is used for evaluating performance on two tasks: semantic segmentation (7 labels) and depth estimation.

4.0.1 Implementation details.

We use the pre-processed datasets for NYUv2 and Cityscapes provided in the code repository of MTAN [20].

Table 1: Comparison with implemented SOTA on 3-task NYUv2 dataset. *Performance evaluation of state-of-the-art methods and the proposed method, implemented on ViT-tiny architecture. The last column shows the average performance improvement.*

	Sem. Segm.		Depth estimation		Surface Normal Prediction					
Method	mIoU↑	pAcc↑	abs↓	rel↓	<11° ↑	<22.5° ↑	<30° ↑	mean↓	median↓	Δ ↑
Single Task	51.29	72.23	0.4423	0.1789	**34.88**	**55.19**	**63.27**	**31.65**	**21.14**	4.19
Baseline (MTL)	50.47	73.20	0.4431	0.1904	29.98	50.83	59.94	33.20	23.87	0
DWA [20]	51.58	**73.76**	0.4137	0.1743	32.56	53.49	62.03	32.33	22.25	4.79
GradNorm [4]	51.28	73.63	0.4179	0.1757	33.17	53.83	62.33	32.26	22.02	4.67
UW [19]	51.21	73.64	0.4126	0.1724	32.27	52.97	61.59	32.50	22.46	4.57
RLW [7]	50.85	73.63	0.4158	0.1744	32.16	53.34	61.98	32.30	22.28	4.32
Cross Stitch [1]	47.98	71.14	0.4310	0.1824	31.46	51.86	60.55	33.17	23.35	0.58
KD-MTL [8]	51.07	73.82	**0.4102**	0.1717	32.47	53.31	61.87	32.35	22.29	4.92
OKD-MTL	**51.99**	73.75	0.4112	**0.1701**	33.58	54.74	63.20	31.82	21.50	**6.22**

We use the ViT-Tiny [25] model fine-tuned on ImageNet [61] as the backbone network. Images are resized to 384×384 pixels before passing them as input to ViT. The patch size for the input is taken as 16×16 and the embedding dimension, E, is 192. We train all models with the AdamW optimizer [62] and the OneCycleLR scheduler [63]. The initial learning rate is set to 10^{-3} and models are trained for 200 epochs for each dataset.

4.0.2 Baseline models

We compare our approach with the following baselines. First, we consider a single-task baseline, where we train networks for each task separately, using a task-specific backbone and a task-specific head. Second, we use a multi-task baseline, in which all tasks share the same vision transformer [25] backbone network, but have separate task-specific heads. We use uniform task losses as the baseline. We compare several recent multi-task learning approaches, including task-weighting schemes such as DWA [20], RLW [7] and Uncertainty Weighting (UW) [19], feature fusion methods such as Cross-Stitch Networks [1], gradient-based methods such as Gradnorm [4], and knowledge distillation based techniques KD-MTL [8]. Similar to the evaluation in [8], we use the same backbone and task-specific heads in all methods for a fair comparison. To separately evaluate the performance of the proposed online knowledge distillation scheme, we compare with previous methods using CNNs instead of transformers. We also compare the performances on the MTAN baseline [20] on CNN models. Similar to the experiments in [7, 8] we compare the methods with both MTAN (DeepLabV3-MTAN) as well as the baseline CNN (DeeplabV3 [64]). The MTAN [20] baseline comparison was not performed with transformer models, as they require fusing of CNN layers with the transformer layers.

Table 2: Comparison with implemented SOTA 2-task Cityscapes dataset. *Performance evaluation of state-of-the-art methods and the proposed method, implemented on ViT-tiny architecture. The last column shows the average performance improvement.*

	Sem. Segm.		Depth estimation		
Method	mIoU↑	pAcc↑	abs↓	rel↓	Δ ↑
Single Task	74.93	93.03	0.0092	0.1422	7.56
Baseline (MTL)	72.12	92.76	0.0110	0.1575	0
DWA [20]	73.90	93.46	0.0092	0.1432	7.17
GradNorm [4]	73.65	93.35	0.0099	0.1483	4.65
UW [19]	74.86	93.72	0.0095	0.1413	7.19
RLW [7]	73.84	93.36	0.0092	0.1396	7.69
Cross Stitch [1]	71.09	92.56	0.0101	0.1551	2.02
KD-MTL [8]	73.81	93.35	0.0098	0.1418	5.96
OKD-MTL	**75.40**	**93.97**	**0.0091**	**0.1360**	**9.19**

4.0.3 Evaluation Metrics

In both NYUv2 and Cityscapes, semantic segmentation is evaluated via mean intersection over union (mIoU) and pixel accuracy (pAcc). For surface normal prediction, we use mean and median angle distances between the prediction and ground truth of all pixels. We also measure the percentage of pixels whose angle prediction error is within $11.25°$, $22.5°$ and $30°$. For depth prediction, we compute absolute and relative errors as the evaluation metrics. The performance of a method relative to the baseline is computed as follows [7]:

$$\Delta = \frac{1}{N_t}\sum_{n=1}^{N_t}\frac{1}{N_n}\sum_{m=1}^{N_n}\frac{(-1)^{p_{n,m}}(M_{n,m} - M^B_{n,m})}{M^B_{n,m}} \times 100\% \ . \tag{4}$$

Here, N_t is the number of tasks, and N_n the number of metrics for the n^{th} task. $M_{n,m}$ and $M^B_{n,m}$ denote the metric of current method and the baseline, respectively, for metric m of task n. The sign is controlled via $p_{n,m}$, which is set to 1

Table 3: Comparison with implemented SOTA on 3-task NYUv2 dataset with DeeplabV3 and DeeplabV3-MTAN backbone. *Performance evaluation of state-of-the-art methods and the proposed method, implemented on DeeplabV3 and DeeplabV3-MTAN architectures. The last column for each architecture shows the average performance improvement.*

	DeepLabV3 backbone							DeepLabV3-MTAN backbone						
	Sem. Segm.		Depth estimation		SN Prediction			Sem. Segm.		Depth estimation		SN Prediction		
Method	mIoU↑	pAcc↑	abs↓	rel↓	mean↓	median↓	Δ ↑	mIoU↑	pAcc↑	abs↓	rel↓	mean↓	median↓	Δ ↑
Single Task	**49.57**	72.88	0.5052	0.1962	27.15	22.11	0.15	**48.69**	**72.87**	0.6228	0.2344	**26.41**	**21.07**	0.15
Baseline	48.11	72.38	0.4792	0.1859	28.63	23.60	0	46.25	72.01	0.5314	0.2151	28.28	23.59	0
DWA [20]	48.21	72.29	0.4703	**0.1817**	28.54	23.54	0.81	46.58	72.23	0.5337	0.2079	27.79	23.10	1.39
GradNorm [4]	48.14	72.49	0.4816	0.1842	28.54	23.59	0.14	46.76	72.26	0.5304	0.2072	27.81	23.11	1.64
UW [19]	48.17	72.39	0.4773	0.1844	28.52	23.43	0.42	46.72	72.05	0.5351	0.2136	28.23	23.62	0.36
RLW [7]	48.39	72.43	0.4756	0.1871	28.67	23.57	0.18	46.24	71.64	0.5371	0.2050	28.03	23.57	0.48
Cross Stitch [1]	48.20	72.86	0.4789	0.1834	28.57	23.87	0.11	-	-	-	-	-	-	-
KD-MTL [8]	48.78	**73.07**	**0.4605**	0.1841	28.08	23.04	1.96	47.35	72.50	0.5148	0.2031	27.66	22.94	3.04
OKD-MTL	49.06	72.91	0.4880	0.1883	**27.04**	**21.52**	**2.45**	48.30	72.58	**0.4957**	**0.1971**	27.36	22.33	**5.18**

Table 4: Component ablation study on NYUv2. *In this experiment we compare models with different key components: Online knowledge distillation (OKD), adaptive feature distillation (AFD), and online task weighting (OTW).*

Model	OKD	AFD	OTW	mIoU↑	abs↓	rel↓	mean↓	median↓	Δ ↑
STL	-	-	-	51.29	0.4423	0.1789	31.65	**21.14**	4.19
Baseline (MTL)	-	-	-	50.47	0.4431	0.1904	33.20	23.87	0
OKD-MTL-AFD	✓	✓	-	51.60	0.4125	0.1735	**31.43**	21.36	5.75
OKD-MTL-OTW	✓	-	✓	51.23	0.4200	0.1770	31.80	21.54	4.96
MTL-pretrained-AFD	-	✓	-	51.18	0.4204	0.1771	31.82	21.50	4.94
OKD-MTL-AFD-OTW	✓	✓	✓	**51.99**	**0.4112**	**0.1701**	31.82	21.50	**6.22**

Table 5: Ablation study on AFD component (left) and hyper-parameter $\mathcal{T}$ (right). *We compare models with different configurations of AFD: AFD for last layer only (AFD-last layer), equal weighting (AFD-equal), softmax weighting (AFD-softmax), select/skip policy (AFD-select/skip) and the proposed random initialization (AFD-ours) for knowledge distillation on intermediate features. The right table shows an ablation study on the different values of $\mathcal{T}$ (eqn. 3).*

Model	Δ ↑
AFD-last layer	4.89
AFD-equal	5.28
AFD-softmax	5.68
AFD-select/skip	4.39
AFD-ours	**6.22**

$\mathcal{T}$	Δ ↑
0.01	3.11
0.05	5.55
0.10	**6.22**
0.30	5.32
1.00	2.11

if a higher value indicates better performance and -1 otherwise. A larger Δ value indicates a larger improvement over the baseline. The number of metrics taken for the semantic segmentation, depth and surface normal estimation are 2 ($mIoU$, $pAcc$), 2 (abs, rel) and 4 ($< 11°$, $< 22.5°$, $< 30°$, $mean$, $median$) in all the tables.

4.1. Experimental settings for comparing methods

We re-implemented a subset of prior methods and replaced the CNN with a ViT-tiny model [25] for a fair comparison. The techniques, DWA [20], Gradnorm [4], UW [19] and RLW [7] are task weighting schemes. The respective task weights are computed and used in the loss function, similar to our proposed OTW function. Cross-Stitch [1] performs fusion of single task network features. This feature fusion is performed in each block. While the original method combines convolution layers, we take layers of the transformer encoder to form a block. In the Cross-Stitch network implementation, we use a linear combination of single-task network features using learnable parameters.

4.2. Results on NYUv2 and Cityscapes datasets

Table 1 shows the results on the NYUv2 dataset. The last column shows the performance of each method relative to the baseline. The proposed method shows a 6.22% improvement over the baseline. The second best performing method is KD-MTL [8] with an improvement of 4.92%. Table 2 shows the results on the Cityscapes dataset. Our method shows an improvement of 9.19% over the baseline. The Random Loss Weighting (RLW) method [7] comes a close second, with a 7.69% improvement.

4.3. Ablation Study

We analyze the performance of proposed online distillation framework on CNN models as well. Table 3 shows the performance using a DeepLabV3 [64] backbone. We also show the performance of various methods with the popular MTAN model as backbone [20]. The training configuration is unchanged to [7, 8, 20]. Similar to the transformer model, the proposed online distillation method performs the best for the CNN model (with and without MTAN) as well, with an improvement of 2.45% and 5.18% over the baseline.

Table 4 shows the contribution of the different components, measured on the NYUv2 dataset. We evaluated various components, including adaptive feature distillation (AFD), pretrained AFD, and online task weighting (OTW). In the table, 'OKD' indicates whether the method is performing simultaneous training of single and multi-task networks, 'STL' indicates single-task performance, 'Baseline (MTL)' indicates the vanilla MTL network. Other rows show the performance of our method using different components. The row 'MTL-pretrained-AFD' indicates offline knowledge distillation with features of single tasks taken from pre-trained models. The components AFD and OTW leads to an improvement of 5.75% and 4.96% over the MTL baseline, respectively. Offline knowledge distillation, 'MTL-pretrained-AFD', provides an improvement of 4.94%. The combination of both AFD and OTW yields the largest improvement of 6.22%.

We evaluate different feature weighting approaches for adaptive knowledge distillation on the NYUv2 dataset, see Table 5 on NYU dataset. The first row apply feature distillation on the penultimate layer only, similar to that of [8]. Learnable weights (AFD-last layer) was applied for distillation from STL networks. The proposed method uses learnable weights, randomly initialized, for feature distillation (AFD-ours). A softmax function on the weights for each layer (AFD-softmax) ensures that the layer-wise weights are normalized across tasks. A select/skip policy (AFD-select/skip) distills knowledge from only one task, which is decided by the argmax of the softmax weighting. In the equal weighting approach (AFD-equal), ω_i^l is set to 1, where distillation of MTL features is performed equally across all the tasks. As seen in Table 5, AFD-equal provides an improvement of 5.28% over the MTL baseline. AFD-softmax and AFD-select/skip techniques show an improvement of 5.68% and 4.39%. AFD-ours performs the best among the compared weighting techniques. We also show the ablation study on the parameter $\mathcal{T}$ for NYU dataset, where the performance is the best for $\mathcal{T} = 0.1$.

Figure 4 shows the variation of ω values for a 5-layer network trained on the NYU dataset for 3 tasks. Red stars in each box plot shows the final value after 100 epochs. Interestingly, the contribution from layers is different for each task. For the semantic segmentation task, contributions are largest from layers 2,4, and 5. Information from layers 3 and 1 are the most vital for the depth and surface normal tasks, respectively.

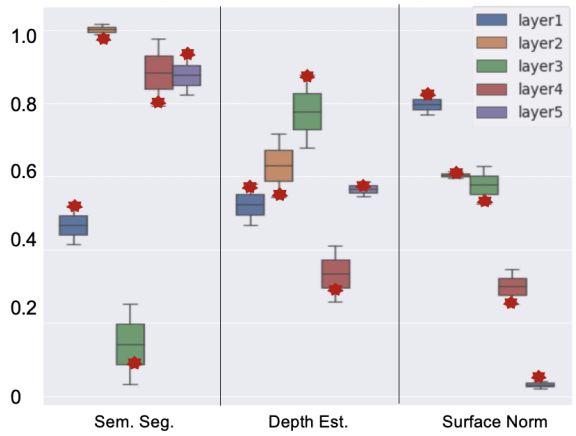

Figure 4: Ablation study on ω. *Box plots of ω with 5 layers of DeeplabV2 backbone. The plots show the variation in the values for each task across 100 epochs, with the final value plotted as a red star.*

Discussion. Online knowledge distillation is able to train multi-task networks with similar accuracy and inference time of STL networks. Single-task networks with the ViT-tiny architecture consist of $26M$ parameters each, whereas the multi-task model uses $33M$ parameters. The simultaneous training of single and multi-task networks on three tasks involves 111M (33M+3$\times$26M) parameters and takes approximately 6 hours on a single V100 GPU. The MTL model only has $33M$ parameters, representing a 57% size reduction and a 60% inference time reduction compared to using separate STL models. Inference for a single image takes approximately 2ms running ViT-tiny model on a V100 GPU. Note that training requires N_t STL networks along with the MTL network, so there is a trade-off of resources spent during training *vs*. inference.

5. Conclusion

We proposed a multi-task learning framework using online knowledge distillation, demonstrated for jointly learning scene understanding tasks. We simultaneously train single-task and multi-task networks and use knowledge distillation of intermediate features of the single task networks. Additionally, we introduce a novel online task weighting scheme analyzing the single- and multi-task network losses. Experiments show improvements on two benchmark datasets over baseline networks, achieving comparable accuracy to single-task models.

References

[1] Ishan Misra et al. “Cross-stitch networks for multi-task learning”. In: *CVPR*. 2016, pp. 3994–4003.

[2] Yuan Gao et al. “Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction”. In: *CVPR*. 2019, pp. 3205–3214.

[3] Nicolas Carion et al. “End-to-end object detection with transformers”. In: *ECCV*. 2020, pp. 213–229.

[4] Zhao Chen et al. “Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks”. In: *ICML*. 2018, pp. 794–803.

[5] Michelle Guo et al. “Dynamic task prioritization for multitask learning”. In: *ECCV*. 2018, pp. 270–287.

[6] Ankit Jha et al. “AdaMT-Net: An Adaptive Weight Learning Based Multi-Task Learning Model For Scene Understanding”. In: *CVPR Workshops*. 2020, pp. 706–707.

[7] Baijiong Lin, Feiyang Ye, and Yu Zhang. “A Closer Look at Loss Weighting in Multi-Task Learning”. In: *arXiv:2111.10603* (2021).

[8] Wei-Hong Li and Hakan Bilen. “Knowledge distillation for multi-task learning”. In: *ECCV Workshops*. 2020, pp. 163–176.

[9] Sihui Luo et al. “Collaboration by competition: Self-coordinated knowledge amalgamation for multi-talent student learning”. In: *ECCV*. 2020, pp. 631–646.

[10] Ozan Sener and Vladlen Koltun. “Multi-task learning as multi-objective optimization”. In: *NeurIPS*. 2018, pp. 525–536.

[11] Jae-Han Lee, Chul Lee, and Chang-Su Kim. “Learning Multiple Pixelwise Tasks Based on Loss Scale Balancing”. In: *ICCV*. 2021, pp. 5107–5116.

[12] Ronghang Hu and Amanpreet Singh. “UniT: Multimodal Multitask Learning with a Unified Transformer”. In: *arXiv:2102.10772* (2021).

[13] Jiasen Lu et al. “12-in-1: Multi-task vision and language representation learning”. In: *CVPR*. 2020, pp. 10437–10446.

[14] Ronan Collobert and Jason Weston. “A unified architecture for natural language processing: Deep neural networks with multitask learning”. In: *ICML*. 2008, pp. 160–167.

[15] Daxiang Dong et al. “Multi-task learning for multiple language translation”. In: *IJCNLP*. 2015, pp. 1723–1732.

[16] Minh-Thang Luong et al. “Multi-task sequence to sequence learning”. In: *arXiv:1511.06114* (2015).

[17] Jui-Ting Huang et al. “Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers”. In: *ICASSP*. 2013, pp. 7304–7308.

[18] Michael L Seltzer and Jasha Droppo. “Multi-task learning in deep neural networks for improved phoneme recognition”. In: *ICASSP*. 2013, pp. 6965–6969.

[19] Alex Kendall, Yarin Gal, and Roberto Cipolla. “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics”. In: *CVPR*. 2018, pp. 7482–7491.

[20] Shikun Liu, Edward Johns, and Andrew J Davison. “End-to-end multi-task learning with attention”. In: *CVPR*. 2019, pp. 1871–1880.

[21] Ximeng Sun et al. “Adashare: Learning what to share for efficient deep multi-task learning”. In: *NeurIPS* 33 (2020).

[22] Ling Zhou et al. “Pattern-structure diffusion for multi-task learning”. In: *CVPR*. 2020, pp. 4514–4523.

[23] Dan Xu et al. “Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing”. In: *CVPR*. 2018, pp. 675–684.

[24] Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. “Mti-net: Multi-scale task interaction networks for multi-task learning”. In: *ECCV*. 2020, pp. 527–543.

[25] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *ICLR* (2021).

[26] Hugo Touvron et al. “Training data-efficient image transformers amp; distillation through attention”. In: *ICML*. Vol. 139. July 2021, pp. 10347–10357.

[27] Zhiqing Sun et al. “Rethinking transformer-based set prediction for object detection”. In: *ICCV*. 2021, pp. 3611–3620.

[28] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. “Adabins: Depth estimation using adaptive bins”. In: *CVPR*. 2021, pp. 4009–4018.

[29] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. “Vision Transformers for Dense Prediction”. In: *ICCV*. Oct. 2021, pp. 12179–12188.

[30] Robin Strudel et al. “Segmenter: Transformer for Semantic Segmentation”. In: *ICCV*. Oct. 2021, pp. 7262–7272.

[31] Yangyang Xu et al. “Multi-Task Learning with Multi-query Transformer for Dense Prediction”. In: *arXiv:2205.14354* (2022).

[32] Deblina Bhattacharjee et al. “MulT: An End-to-End Multitask Learning Transformer”. In: *CVPR*. 2022, pp. 12031–12041.

[33] Hanrong Ye and Dan Xu. “Inverted Pyramid Multi-task Transformer for Dense Scene Understanding”. In: 2022.

[34] Guorui Zhou et al. “Rocket launching: A universal and efficient framework for training well-performing light net”. In: *32nd AAAI Conf. AI*. 2018.

[35] Rich Caruana. “Multitask Learning”. In: *Machine Learning* 1.28 (1997), pp. 41–75.

[36] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. “Learning multiple visual domains with residual adapters”. In: *NeurIPS*. 2017, pp. 506–516.

[37] Sebastian Ruder. “An overview of multi-task learning in deep neural networks”. In: *arXiv:1706.05098* (2017).

[38] Elliot Meyerson and Risto Miikkulainen. “Pseudo-task augmentation: From deep multitask learning to intratask sharing—and back”. In: *ICML*. 2018, pp. 3511–3520.

[39] Zhanpeng Zhang et al. “Facial landmark detection by deep multi-task learning”. In: *ECCV*. 2014, pp. 94–108.

[40] Iasonas Kokkinos. “Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory”. In: *CVPR*. 2017, pp. 6129–6138.

[41] Simon Vandenhende et al. “Multi-task learning for dense prediction tasks: A survey”. In: *PAMI* (2021).

[42] Tianhe Yu et al. “Gradient surgery for multi-task learning”. In: *arXiv:2001.06782* (2020).

[43] Lucas Pascal et al. “Maximum Roaming Multi-Task Learning”. In: *AAAI*. Vol. 35. 10. 2021, pp. 9331–9341.

[44] Zhenyu Zhang et al. “Pattern-affinitive propagation across depth, surface normal and semantic segmentation”. In: *CVPR*. 2019, pp. 4106–4115.

[45] Zhenyu Zhang et al. “Joint task-recursive learning for semantic segmentation and depth estimation”. In: *ECCV*. 2018, pp. 235–251.

[46] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv:1503.02531* (2015).

[47] Adriana Romero et al. “Fitnets: Hints for thin deep nets”. In: *arXiv:1412.6550* (2014).

[48] Yonglong Tian, Dilip Krishnan, and Phillip Isola. “Contrastive Representation Distillation”. In: *ICLR*. 2019.

[49] Jiaqi Ma and Qiaozhu Mei. “Graph Representation Learning via Multi-task Knowledge Distillation”. In: *NeurIPS Workshops*. 2019.

[50] Mary Phuong and Christoph Lampert. “Towards understanding knowledge distillation”. In: *ICML*. 2019, pp. 5142–5151.

[51] Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. “Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning”. In: *ICLR*. 2016.

[52] Kevin Clark et al. “BAM! Born-Again Multi-Task Networks for Natural Language Understanding”. In: *ACL*. 2019, pp. 5931–5937.

[53] Ze Meng, Xin Yao, and Lifeng Sun. “Multi-Task Distillation: Towards Mitigating the Negative Transfer in Multi-Task Learning”. In: *ICIP*. 2021, pp. 389–393.

[54] Ankit Jha et al. “SD-MTCNN: Self-Distilled Multi-Task CNN.” In: *BMVC*. 2020.

[55] Aisha Urooj Khan et al. “Mmft-bert: Multimodal fusion transformer with bert encodings for visual question answering”. In: *arXiv:2010.14095* (2020).

[56] Yiyi Zhou et al. “TRAR: Routing the Attention Spans in Transformer for Visual Question Answering”. In: *ICCV*. 2021, pp. 2074–2084.

[57] Jieneng Chen et al. “Transunet: Transformers make strong encoders for medical image segmentation”. In: *arXiv:2102.04306* (2021).

[58] David Eigen, Christian Puhrsch, and Rob Fergus. “Depth map prediction from a single image using a multi-scale deep network”. In: *NeurIPS* 27 (2014).

[59] Nathan Silberman et al. “Indoor Segmentation and Support Inference from RGBD Images”. In: *ECCV*. 2012.

[60] Marius Cordts et al. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *CVPR*. 2016.

[61] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *CVPR*. 2009, pp. 248–255.

[62] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *ICLR*. 2018.

[63] Leslie N Smith. “A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay”. In: *arXiv:1803.09820* (2018).

[64] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv:1706.05587* (2017).