Mitigating Task Interference in Multi-Task Learning via Explicit Task Routing with Non-Learnable Primitives

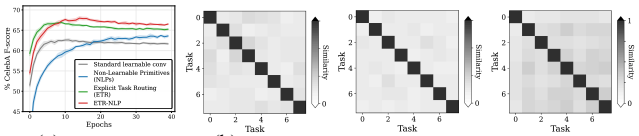
Chuntao Ding[1*] Zhichao Lu[2†] Shangguang Wang[3] Ran Cheng[4] Vishnu N. Boddeti[5]
[1] Beijing Jiaotong University [2] Sun Yat-sen University [3] Beijing University of Posts and Telecommunications
[4] Southern University of Science and Technology [5] Michigan State University
chuntaoding@163.com {luzhichaocn, ranchengcn}@gmail.com sgwang@bupt.edu.cn vishnu@msu.edu

Abstract

Multi-task learning (MTL) seeks to learn a single model to accomplish multiple tasks by leveraging shared information among the tasks. Existing MTL models, however, have been known to suffer from negative interference among tasks. Efforts to mitigate task interference have focused on either loss/gradient balancing or implicit parameter partitioning with partial overlaps among the tasks. In this paper, we propose ETR-NLP to mitigate task interference through a synergistic combination of non-learnable primitives (NLPs) and explicit task routing (ETR). Our key idea is to employ non-learnable primitives to extract a diverse set of task-agnostic features and recombine them into a shared branch common to all tasks and explicit task-specific branches reserved for each task. The non-learnable primitives and the explicit decoupling of learnable parameters into shared and task-specific ones afford the flexibility needed for minimizing task interference. We evaluate the efficacy of ETR-NLP networks for both image-level classification and pixel-level dense prediction MTL problems. Experimental results indicate that ETR-NLP significantly outperforms state-of-the-art baselines with fewer learnable parameters and similar FLOPs across all datasets. Code is available at this URL.

1. Introduction

Multi-task learning (MTL) is commonly employed to improve learning efficiency and performance of multiple tasks by using supervised signals from other related tasks [6, 25, 37]. These models have led to impressive results across numerous tasks. However, there is well-documented evidence [14,21,32,39] that these models are suffering from *task interference* [39], thereby limiting multi-task networks (MTNs) from realizing their full potential.



(a) ResNet18 (b) Layer 1, 4, and 8 of ResNet18 (from left to right)

Figure 1. (a) Learning progression of multi-task networks (MTNs) on CelebA for eight tasks. Hard-sharing models with fully learnable parameters (gray) learn rapidly and then suffer from performance degradation due to conflicting gradients from task interference. Networks with non-learnable primitives (NLPs; blue) do not suffer from task interference by design, while explicit task routing (ETR; green), and ETR with NLPs (red) do not eliminate but suffer less from task interference. (b) Gradient correlations measured via CKA [15] across all pairs of tasks for different layers of a standard MTN at the end of training. Observe the acute lack of correlation between tasks (low off-diagonal magnitude).

For instance, consider the learning progression of an MTN with a standard learnable convolutional layer in Figure 1a (blue curve). Observe that the model learns rapidly, we posit, by exploiting all the shared information between the tasks, i.e., gradients pointing in similar directions. However, the performance starts degrading on further training since the model needs to exploit dissimilar information between the tasks for further improvement, i.e., gradients point in different directions. The latter can be verified by observing the similarity (centered kernel alignment [15]), or the lack thereof, between the gradients for each pair of tasks in Figure 1b.

Several approaches were proposed for mitigating task interference in MTNs, including loss/gradient balancing [13, 17, 18, 26, 38], parameter partitioning [2, 21, 23, 29] and architectural design [7, 14, 22]. Despite the diversity of these approaches, they share two common characteristics, (i) all parameters are learned, either for a pre-trained task or for the multiple tasks at hand, (ii) the learned parameters are either fully shared across all tasks or are shared across a partial set of tasks through implicit partitioning, i.e., with no direct control over which parameters are shared across which tasks. Both of these features limit the flexibility of existing

*Work done as a visiting scholar at Michigan State University.
†Corresponding author

multi-task network designs from mitigating the deleterious effects of task interference on their predictive performance.

Relaxing the above design choices is the primary goal of this paper. We propose two complementary design principles, namely *explicit task routing (ETR)*, and *non-learnable primitives (NLPs)*, that explicitly seek to mitigate task interference. Through extensive empirical evaluation, we demonstrate that these two complementary ideas, individually and jointly, help mitigate task interference and consistently improve the performance of MTNs. As can be observed in Figure 1a, compared to a hard-sharing MTN with a standard learnable convolutional layer (gray curve), an MTN with NLP (blue curve) has better learning characteristics, i.e., learn more steadily and not suffer from performance degradation. Similarly, MTN with ETR (green curve) and MTN with ETR-NLP (red curve) does not eliminate task interference but reduce it to an extent. Figure 2 shows an overview of the proposed ETR-NLP networks.

From a network topological perspective, we propose explicit task-routing (ETR), a parameter allocation strategy that partitions the parameters into *shared* and *task-specific* branches. More explicitly, it comprises one branch shared across all tasks and task-specific branches, one for each task. Unlike existing parameter partitioning methods, ETR is designed to offer precise and fine-grained control over *which* and *how many* parameters are *shared* or *not shared* across the tasks. Additionally, ETR is flexible enough to allow existing implicit parameter partitioning methods [23, 29] to be incorporated into its shared branch.

From a layer design perspective, we propose using non-learnable primitives (NLPs) to extract task-agnostic features and allow each task to choose optimal combinations of these features adaptively. There is growing evidence that features extracted from NLPs can be very effective for single-task settings, including for image classification [12, 24, 34–36], reinforcement learning [8] and modeling dynamical systems [20]. NLPs are attractive for mitigating task interference in MTL. Since they do not contain learnable parameters, the task-agnostic features extracted from such layers alleviate the impact of conflicting gradients, thus implicitly addressing task interference. However, the utility and design of NLPs for multi-task networks have not been explored. We summarize our key contributions below:

– We introduce the concept of *non-learnable primitives (NLPs)* and *explicit task routing (ETR)* to mitigate task interference in multi-task learning. We systematically study the effect of different design choices to determine the optimal design of ETR and NLP.

– We demonstrate the effectiveness of ETR and NLP through MTNs constructed with only NLPs (MTN-NLPs) and only ETR (MTN-ETR) for both image-level classification and pixel-level dense prediction tasks.

– We evaluate the effectiveness of ETR-NLP networks across three different datasets and compare them against a wide range of baselines for both image-level classification and pixel-level dense prediction tasks. Results indicate that ETR-NLP networks consistently improve performance by a significant amount.

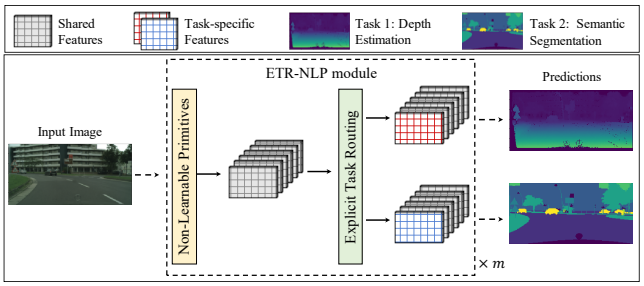


Figure 2. **ETR-NLP Networks** comprise non-learnable primitives to extract diverse task-agnostic features, followed by explicit task routing to control the parameters/features that are shared across all tasks and those that are exclusive to every single task.

2. Related Work

We briefly review prior work on non-learnable primitives and mitigating task interference for multi-task learning (MTL), from which our work draws inspiration. Due to space constraints, we refer the readers to the supplementary material for more discussion of related work. We also encourage readers to refer to multiple excellent reviews [3, 32, 37] for a comprehensive overview of MTL.

Non-Learnable Primitives (NLPs): The notion of NLP for feature extraction was explored for single-task learning motivated either by scientific curiosity or in the quest for computational efficiency [4, 8, 12, 34, 35]. Xu et al. used non-learnable sparse binary convolutional filters referred to as *LBConv* [12]. Wu et al. proposed randomly adjusting the spatial alignments of data, referred to as *shift* [34]. Xu et al. applied non-learnable additive noises sampled from a uniform distribution to data, referred to as *perturbation* [35]. Yu et al. replaced the attention-based token mixer with a non-parametric pooling primitive in vision transformers [36]. As a common practice, a follow-up 1×1 convolution was used to learn a weighted linear combination of features extracted by non-learnable primitives. These methods generally perform as well as those with standard learnable layers but with much fewer parameters required to optimize. However, the utility of NLPs for MTL is yet to be explored.

The NLP-based feature extraction proposed in this work is notably different in three key respects: (i) We expand the scope of NLP from single-task image classification to MTL, including image-level and pixel-level prediction tasks. (ii) We consider multiple types of NLPs (i.e., pooling, shift, perturbation, convolution without learnable weights) and demonstrate that a single type of NLP does not benefit MTL. (iii) We design an MTL-specific NLP by exploring

various combinations of NLPs under a diverse set of hyper-parameters (e.g., pooling/kernel size, sparsity, perturbation strength, real, binary, depth-wise separable weights, etc.).

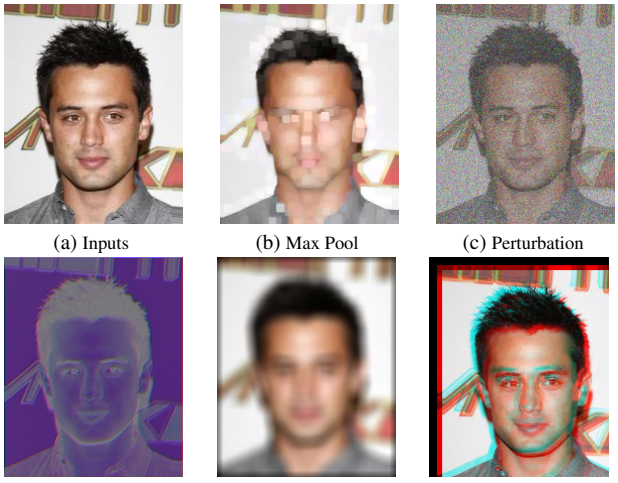
Task Interference in MTL: The success of MTL in computer vision has led to many solutions for mitigating task interference in MTL. The approaches fall into three main categories, (i) loss/gradient balancing [13, 17, 18, 26, 38], (ii) parameter partitioning [2, 21, 23, 29], and (iii) architectural design [7, 14, 22] (supplementary materials). A brief overview is provided below.

–Loss/Gradient balancing: Kendall *et al.* [13] utilized homoscedastic uncertainty as task-dependent weights to balance the losses of various tasks. Chen *et al.* [38] adaptively balanced the training of deep MTL models by dynamically adjusting the magnitudes of gradients computed w.r.t different tasks. Liu *et al.* [18] proposed task-specific loss functions to maintain balance among tasks. Finally, Sener and Koltun [26] applied multi-objective optimization to find Pareto-optimal gradients for multiple tasks. The primary goal of this class of methods is to control the contribution of the loss/gradient of each task to the overall loss/gradient, which in turn implicitly helps mitigate task interference.

–Parameter partitioning: Attention mechanisms have been widely used to allow networks to focus on different regions of the feature maps adaptively [33]. Attention mechanisms have been employed [18] for MTL at the filter level, allowing each task to select a subset of parameters (i.e., partition) at each layer. Maninis *et al.* [21] used a task-specific squeeze-and-excitation module (i.e., channel attention) for soft parameter partitioning. Strezoski *et al.* [29] introduced a task routing module as a hard parameter partitioning strategy to alleviate interference among tasks by randomly assigning a sub-network to each task. Once assigned, the parameter partitioning remained unchanged. Maximum roaming improved task routing by adaptively updating the parameter partition assignments during training [23]. In contrast to the overlapped parameter partitioning in the aforementioned work, our task routing strategy explicitly reserves a task-specific branch of parameters exclusive for each task, leading to more precise control over the partitioning of parameters among the tasks.

3. ETR-NLP Network Design

We first introduce non-learnable primitives (NLPs) based feature extraction and explicit task routing (ETR). Then, we integrate both into a single module, dubbed ETR-NLP, that can be incorporated into modern MTL architectures (e.g., ResNets [10], VGGs [28], SegNet [1], etc.) in a straightforward manner. Lastly, we describe the network's training and inference process with ETR-NLP modules.



(a) Inputs (b) Max Pool (c) Perturbation

(d) Conv (binary) (e) Avg. Pool (f) Shift

Figure 3. **Visualization of features extracted by various NLPs:** Given an input image (a), different types of NLP extract diversely different features (b) - (f).

3.1. NLP based Feature Extraction

NLPs afford several attractive properties that render them well-suited for MTL. *Primarily*, NLPs do not have any learnable parameters. Hence, the extracted features are agnostic to any particular task, alleviating the impact of disparate gradients. As such, NLPs implicitly mitigate task interference and lead to better predictive performance. A *secondary benefit* is from a computational perspective. Our proposed NLP design offers computation benefits in terms of fewer learned parameters or lower FLOPs. However, as we demonstrate in §4.2.1, obtaining parameter efficiency in MTL is challenging since directly employing existing efficiency-oriented convolutional layer designs (which work very well on standard problems) leads to performance loss on MTL.

We hypothesize that extracting features from non-learnable primitives (NLPs) that are neither biased nor adaptable to the tasks at hand can mitigate task interference in MTL. A plethora of NLPs are available, including both non-parametric (e.g., average/max pooling, identity mapping, etc.) and weight-agnostic ones (e.g., LBConv [12], perturbation [35], shift [4, 34], etc.). Furthermore, one can tune each type of NLP by adjusting its hyperparameters, such as pooling size, perturbation strength, the sparsity of the non-learnable weights, etc.

However, we demonstrate in Table 1 directly employing a single type of non-learnable primitive degrades the performance of the corresponding MTL model. Since different tasks benefit from different kinds of features, a single NLP is sub-optimal for MTL. And, as we observe in Figure 3, different NLPs extract different features. Therefore, to facilitate extracting a dictionary of diverse features, we place various types of NLPs across different hyperparameter combinations in parallel, similar to an Inception structure [30].

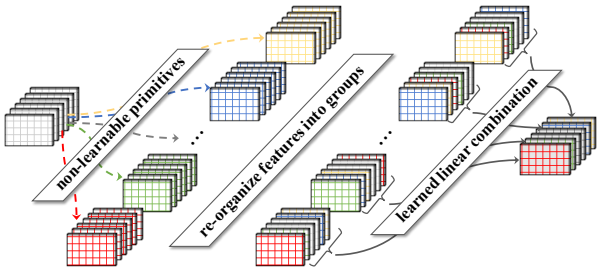


Figure 4. **Design of our NLP-based feature extraction:** It first uses various types of non-learnable primitives (e.g., pooling, shift [4, 34], perturbation [35], etc.) with different hyperparameters (e.g., kernel size, sparsity, perturbation strength, etc.) to extract a dictionary of diverse features. Then, these features are re-arranged into groups by taking one output feature from each primitive. Subsequently, a linear combination is learned to compress features in a group into a single output feature. Finally, these compressed features from different groups are concatenated together.

Next, we re-arrange the extracted features into groups to enhance diversity by taking one feature map from each primitive. Then, a linear combination among features within a group is learned via group-wise 1×1 convolutions. A pictorial illustration of this process is provided in Figure 4.

Based on the schema described above, we first determine the optimal combination of NLPs for MTL. Specifically, we consider five types of NLPs, i.e., average pooling, max pooling, convolution with fixed weights [12], shift [34], and perturbation [35], for a total of $\sum_{i=1}^{5} \binom{5}{i} = 31$ possible variations. Then, we evaluate each variation on both CelebA multi-attribute classification [19] and Cityscapes dense prediction (semantic segmentation and depth estimation) [5] MTL problems, and perform five repetitions to account for performance fluctuations. A representative subset of results is presented in Table 1 (see supplementary material for full results). We observed that using more NLPs for extracting features generally leads to better MTL performance. In particular, the combination of averaging pool, convolution with non-learnable weights, and perturbation in parallel emerges as the top choice, i.e., our final configuration of NLPs. The effects of the hyperparameters of NLPs are presented in §5.

3.2. Explicit Task Routing (ETR)

Despite the extracted features being agnostic to any particular task, a standalone application of NLPs does not proactively address task interference. Therefore, to complement NLP-based feature extraction, we present a novel parameter partitioning method, dubbed *explicit task routing* (ETR), to provide precise and fine-grained control over the sharing of parameters among tasks.

Figure 5 provides a pictorial illustration of ETR (along with the de-facto hard parameter sharing and existing parameter partitioning methods [23, 29]) for a three-task scenario. Parameters associated with the shared branch are shared across all tasks and absorb supervised signals common to all tasks. On the other hand, the parameters related to the task-specific branch are exclusive to each task and learn task-specific features. This explicit separation of parameters helps mitigate mutual interference among tasks. Additionally, to provide direct control and flexibility in terms of the number of shared parameters vs. task-specific parameters, we also introduce a hyperparameter, $\gamma \in [0, 1]$, that controls the ratio of shared parameters over total available parameters. This ratio can be independently varied for each task. Note that $\gamma = 0$ indicates the absence of the shared branch, which is equivalent to single-task learning. In contrast, $\gamma = 1$ corresponds to all features shared among tasks, equal to a standard hard parameter sharing MTL. The effect of γ is studied in §4.2.2.

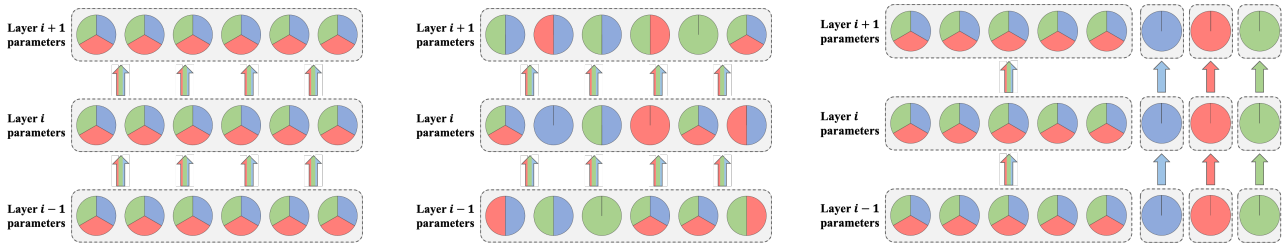
Table 1. **Effect of different configurations of NLPs:** Relative improvements/degradation over standard learnable convolution on CelebA multi-attribute classification are highlighted in color.

#Types	Non-Learnable Primitives					CelebA F-score (↑)	Δ_p (↑)
	Avg. pool	Max pool	Conv	Shift	Perturb		
1			✓			$61.1_{\pm 0.2}$	-3.0%
		✓				$61.3_{\pm 0.1}$	-2.7%
2	✓	✓				$61.6_{\pm 0.1}$	-2.3%
			✓	✓		$62.2_{\pm 0.2}$	-1.3%
	✓			✓		$62.4_{\pm 0.1}$	-1.0%
3	✓		✓	✓		$62.4_{\pm 0.1}$	-0.9%
		✓		✓	✓	$64.5_{\pm 0.2}$	+2.4%
	✓		✓		✓	$66.3_{\pm 0.3}$	+5.2%
4	✓		✓	✓	✓	$65.0_{\pm 0.1}$	+3.2%
	✓	✓	✓		✓	$64.1_{\pm 0.1}$	+1.8%
5	✓	✓	✓	✓	✓	$64.1_{\pm 0.2}$	+1.7%
Standard learnable convolution						$63.0_{\pm 0.2}$	0.0%

3.3. ETR-NLP Module

We incorporate the proposed NLPs-based feature extraction and ETR parameter partitioning into a single module, dubbed ETR-NLP, and form MTL networks by replacing the standard convolutional layers in modern MTL architectures with ETR-NLP modules. In each ETR-NLP module, since interactions between the tasks happen through the shared branch, the shared features are obtained by recombining task agnostic features through learned via group-wise 1×1 convolutions. Furthermore, since the task-specific weights are exclusive to each task and do not interfere with other tasks, the task-specific features are obtained by directly applying standard 3×3 convolutions to features from the previous layer, i.e., task-specific features are not extracted for the task-specific branch. A pictorial illustration of ETR-NLP and its corresponding pseudocode is shown in Figure 1 and Algorithm 1, respectively.

Training and inference: Similar to prior parameter partitioning-based methods [21, 23, 29], only one task is ac-



(a) Hard parameter sharing (b) Existing parameter partitioning methods (c) Parameter partitioning through explicit task routing (ours)

Figure 5. **Illustration of Explicit Task Routing (ETR):** (a) The de-facto hard parameter sharing method assigns all parameters to all tasks. (b) Existing parameter partitioning methods assign a partial and overlapped set of parameters for each task, and the assignment is either kept fixed [29] or updated/learned iteratively [21, 23]. (c) Our task explicit task routing separates parameters into a common branch shared by all tasks and a task-specific branch reserved exclusively for each task, providing more precise control over parameter partitioning among tasks. Note that only one task is activated during a forward pass for both the existing and our proposed parameter partitioning methods.

Algorithm 1 ETR-NLP: PyTorch-like Pseudocode

```
# C_in, C_out: number of input/output channels
# gamma: sharing ratio of explicit task routing
# prims: settings of non-learnable primitives
# T: No. of tasks
class NLP(nn.Module):
    def __init__(self, C_in, C_out, prims, **kwargs):
        # define non-learnable primitives
        for i, op in enumerate(prims):
            self.add_module(str(i), op(C_in, **kwargs))
        k = len(prims) # No. of NLPs
        # group-wise linear combination
        conv1x1 = nn.Conv2d(C_in * k, C_out, ks=1,
            groups=C_in)

    def forward(self, x):
        # extract features by NLPs
        y = torch.cat([op(x) for i, op in enumerate(
            self.values())], dim=1)
        # re-arrange features via channel shuffling
        y = torch.channel_shuffle(y, groups=k)
        return conv1x1(y)

class ETR-NLP(nn.Module):
    def __init__(self, C_in, C_out, gamma, prims, T,
        **kwargs):
        C_shared = int(gamma * C_out)
        C_specif = int(C_out - self.C_shared)
        # define a shared branch
        shared_branch = NLP(C_in, C_shared, prims, **
            kwargs)
        # define task-specific branches
        for i in range(T):
            specif_branch = nn.Conv2d(C_in, C_specif,
                ks=3, s=1, p=1) # standard Conv
            self.add_module(``task_{}''.format(i),
                specif_branch)
        self.task = 0 # set an active task

    def get_layer(self, name):
        return getattr(self, name)

    def forward(self, x):
        shared = shared_branch(x)
        specif = self.get_layer(``task_{}''.format(
            self.task))(x)
        return torch.cat([shared, specif], dim=1)
```

tivated at a time during a forward pass of our ETR-NLP-based MTL networks. The training process for ETR-NLP-based networks proceeds as follows. The shared branch and one task-specific branch are activated during a forward pass. As shown in Figure 5c, when task i ($i \in [1, \mathcal{T}]$) is active, features for the i-th task will be extracted through the shared and the active i-th task-specific parameters. After training the current task, the parameters of the shared branch are updated immediately for image-level classification MTL problems (e.g., CelebA). While for dense prediction MTL problems (e.g., Cityscapes, NYU-v2), we wait until all tasks are forwarded before updating the parameters of the shared branch. These decisions are driven by an ablative analysis shown in §5. During inference, a separate per-task evaluation is required as the input propagates through the shared and task-specific branches.

4. Experimental Evaluation

In this section, we first describe our experimental setup. Then, we independently demonstrate the effectiveness of NLPs and ETR for MTL. Finally, we compare our ETR-NLP to a wide range of MTL baselines for both image-level classification and pixel-level prediction problems.

4.1. Experimental Setup

Datasets. We conduct experiments on three widely used MTL benchmarks: *CelebA* [19] is a large-scale face attributes dataset containing more than 200K celebrity images, each with 40 binary attribute annotations that can be grouped into eight categories. Accordingly, we can define an eight or 40-task MTL problem by considering each group or attribute as an individual binary classification task. *Cityscapes* [5] is a large-scale dataset for the semantic understanding of urban street scenes. It is split into training, validation, and test sets, with 2975, 500, and 1525 images. Following [18, 23], we resize all images to 128 by 256 and use the median level segmentation comprising seven semantic categories. Together with depth estimation, we define an eight-task MTL problem by treating the segmentation of each semantic category separately. *NYU-v2* [27] is a video sequence dataset composed of 1449 indoor images recorded over 464 scenes from a Microsoft Kinect camera. Following [18], we resize all images to 288 by 384 resolution. It supports the segmentation of 13 semantic cate-

gories, depth estimation, and surface normal estimation for 15 tasks. More details are available in the supplementary.

Implementation Details. We implement ETR-NLP in ResNet18 [10] and VGG16 [28] architectures for image-level classification problems (e.g., CelebA), and in SegNet [1] architecture for pixel-level dense prediction problems (e.g., Cityscapes and NYU-v2). For training on CelebA, we use Adam optimizer with a learning rate of 10^{-4} and a batch size of 256 images for 40 epochs. For training on Cityscapes and NYU-v2, we also use Adam optimizer with a learning rate of 10^{-4}, but with a batch size of 8 images for 500 epochs. We repeat each experiment five times.

Evaluation Metrics. To evaluate the performance on image-level classification MTL problems, we consider *precision*, *recall*, and *F-Score*. *Precision* measures how precise a method is regarding how many predicted true instances are true positives. *Recall* estimates how well a method has adapted to each task by measuring how much of the actual positive instances are recognized. *F-Score* provides a composite measurement derived from precision and recall. This work reports mean precision, recall, and F-Score averaged over all tasks. For evaluating the performance on pixel-level dense prediction MTL problems, we track the *mean Intersection over Union* (mIoU), and *pixel accuracy* (Pix. Acc.) averaged over all segmentation tasks, and the *mean absolute* (Abs. Err.) and *relative error* (Rel. Err.) for the depth estimation task. Lastly, following prior work [3, 16, 32, 37], we also report the average relative improvement Δ_{p} (defined below) w.r.t. a chosen baseline.

$$\Delta_{\mathrm{p}} = 100\% \times \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{n=1}^{N_t}\frac{(-1)^{p_{t,n}}(M_{t,n} - M_{t,n}^{\mathrm{baseline}})}{M_{t,n}^{\mathrm{baseline}}}$$

where N_t is the number of metrics in task t, $M_{t,n}$ is the performance of a task balancing method for the n-th metric in task t, $M_{t,n}^{\mathrm{baseline}}$ is defined similarly for the baseline method, and $p_{t,n}$ is one if a higher value indicates better performance for the n-th metric in task t and zero otherwise.

4.2. Experimental Results

4.2.1 Effectiveness of Non-Learnable Primitives

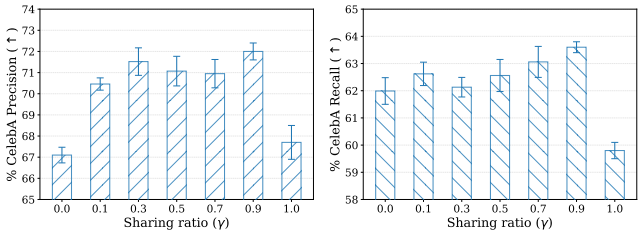
Table 2 compares our proposed NLPs with other alternative operations. We make the following observations, (i) All standalone instantiations of NLPs lead to performance degradation. The lack of diversity in the features is detrimental to predictive performance. (ii) Unlike standalone NLPs, our proposed NLP-based networks achieve higher precision and recall while requiring fewer learnable parameters and FLOPs. Since NLPs extract task-agnostic and diverse features, they can prevent the network parameters from being dominated by one or more tasks and mitigate mutual interference between tasks. Compared to the baseline architecture with standard learned convolution, NLPs-based networks significantly improve performance.

Table 2. Comparison of NLPs with alternative designs on CelebA image-level classification problems. 2× means a width multiplier of 2. Our results are highlighted with shading.

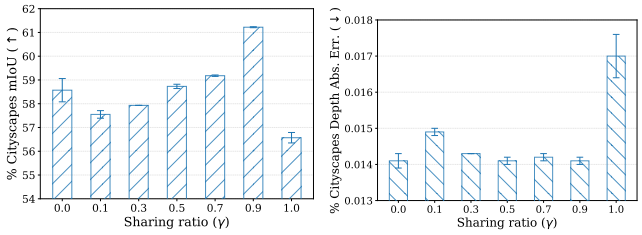
	Method	#P	#F	Prec. (↑)	Recall (↑)	Δ_{p} (↑)
ResNet18	Conv	11.2M	148M	$67.7_{\pm 0.8}$	$59.8_{\pm 0.3}$	0.0%
	LBConv [12]	1.61M	165M	$65.1_{\pm 0.8}$	$53.2_{\pm 0.4}$	-7.4%
	Shift [4]	2.81M	42M	$67.5_{\pm 1.1}$	$58.4_{\pm 0.6}$	-1.3%
	Depth-wise Conv [11]	2.91M	45M	$65.7_{\pm 0.4}$	$51.5_{\pm 0.4}$	-8.4%
	Ghost module [9]	5.77M	46M	$67.6_{\pm 1.5}$	$57.9_{\pm 0.6}$	-1.7%
	NLPs	2.05M	43M	$\mathbf{72.8_{\pm 0.3}}$	$59.2_{\pm 0.4}$	+3.3%
	NLPs (2×)	8.11M	148M	$71.3_{\pm 0.6}$	$\mathbf{62.2_{\pm 0.1}}$	+4.7%
VGG16	Conv	14.7M	1.25G	$71.1_{\pm 0.8}$	$63.8_{\pm 0.6}$	0.0%
	LBConv [12]	1.84M	1.43G	$67.1_{\pm 0.6}$	$58.0_{\pm 0.4}$	-7.4%
	Shift [4]	1.67M	0.14G	$68.9_{\pm 0.2}$	$59.5_{\pm 0.2}$	-4.9%
	Depth-wise Conv [11]	3.57M	0.34G	$65.8_{\pm 0.7}$	$51.6_{\pm 0.8}$	-13.3%
	Ghost module [9]	7.45M	0.32G	$68.8_{\pm 0.8}$	$61.6_{\pm 0.4}$	-3.3%
	NLPs	2.48M	0.22G	$\mathbf{74.2_{\pm 0.8}}$	$64.5_{\pm 1.1}$	+2.7%
	NLPs (2×)	14.3M	1.23G	$72.5_{\pm 0.7}$	$\mathbf{68.7_{\pm 0.9}}$	+4.8%

4.2.2 Effectiveness of Explicit Task Routing

Figure 6 shows the effect of sharing ratio γ on explicit task routing (ETR) performance over the CelebA and Cityscapes datasets. The results show that $\gamma = 0.9$, i.e., 90% of the features are shared among tasks, leads to the best performance across both image-level and pixel-level tasks. These results suggest that both cases benefit by sharing a significant number of parameters while still needing a small fraction of task-specific parameters. It is worth noting that when $\gamma = 1$, i.e., all tasks share all the parameters of the multi-task network, the performance is impaired due to mutual interference between tasks. Accordingly, we set $\gamma = 0.9$ for all experiments shown in the main results section.



(a) CelebA Image-Level Classification



(b) Cityscapes Pixel-Level Prediction

Figure 6. Effects of sharing ratio γ on image-level classification and pixel-level prediction MTL problems.

Table 3. Comparison of ETR with baselines on image-level classification and pixel-level prediction MTL problems. Our results are highlighted with shading.

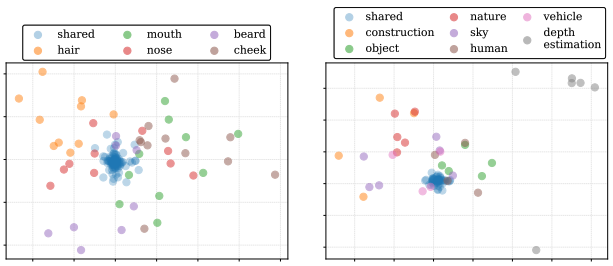
(a) Image-Level Classification

	Method (ResNet18)	#P (M)	Prec. (↑)	Recall (↑)	Δ_p (↑)
CelebA	Hard sharing	11.2	$67.7_{\pm 0.8}$	$59.8_{\pm 0.3}$	0.0%
	Attentive hard sharing [21]	12.9	$71.1_{\pm 0.3}$	$62.6_{\pm 0.5}$	+4.9%
	Task routing ($\gamma = 0.9$) [29]	11.2	$71.7_{\pm 0.1}$	$61.7_{\pm 0.5}$	+4.5%
	Max roaming ($\gamma = 0.9$) [23]	11.2	$71.2_{\pm 0.4}$	$63.0_{\pm 0.6}$	+5.3%
	ETR ($\gamma = 0.9$)	11.2	$\mathbf{72.0}_{\pm \mathbf{0.4}}$	$\mathbf{63.6}_{\pm \mathbf{0.2}}$	**+6.4%**

(b) Pixel-Level Prediction

	Method (SegNet)	#P (M)	Segm. mIoU (↑)	Depth Abs. Err. (↓)	Δ_p (↑)
Cityscapes	Hard sharing	25.1	$56.57_{\pm 0.22}$	$0.0170_{\pm 0.0006}$	0.0%
	Attentive hard sharing [21]	28.2	$55.45_{\pm 1.03}$	$0.0160_{\pm 0.0006}$	+2.0%
	Task routing ($\gamma = 0.6$) [29]	25.1	$56.52_{\pm 0.41}$	$0.0155_{\pm 0.0003}$	+4.4%
	Max roaming ($\gamma = 0.6$) [23]	25.1	$57.93_{\pm 0.20}$	$0.0143_{\pm 0.0001}$	+9.1%
	ETR ($\gamma = 0.9$)	25.1	$\mathbf{61.22}_{\pm \mathbf{0.16}}$	$\mathbf{0.0141}_{\pm \mathbf{0.0001}}$	**+12.6%**

Table 3 compares our explicit task routing with other parameter partitioning methods. All methods use regular convolution. We make the following observations, (i) All parameter partitioning methods improve performance over hard sharing. (ii) Our explicit task routing strategy is more effective for alleviating task interference and improving performance on both image-level classification and pixel-level dense prediction tasks. Additionally, to further understand the utility of explicit task routing, we visualize the features extracted by the shared branch and task-specific branches using t-SNE [31] in Figure 7. We observe that the shared branch extracts similar features across tasks, while the task-specific stems extract individualized features for specific tasks. We also notice that the features extracted by task-specific branches are dissimilar for different tasks, which shows that our explicit task routing can obtain task-specific features and mitigate mutual interference between tasks.



(a) CelebA Image-Level Classification (b) Cityscapes Pixel-Level Prediction

Figure 7. **t-SNE visualization of feature activations** from ETR's shared branch and task-specific branches of a single image on (a) CelebA and (b) Cityscapes. Note that features from the shared branch are clustered, while task-specific branches are spread-out.

4.2.3 ETR-NLP Networks

Table 4 presents the performance of various methods on the CelebA image-level classification problems. We consider two experimental settings, one with eight group-level tasks and another with 40 binary tasks. We make the following observations, (i) Our ETR-NLP is consistently better than the baselines methods while having fewer learnable parameters. (ii) We observe that the performance gains become more prominent as the number of tasks increases due to the inherent minimization of interference between tasks in ETR-NLP. (iii) We also observe that parameter partitioning methods (i.e., task routing, max roaming, and ETR-NLP) scale better than loss/gradient balancing methods (i.e., GradNorm and MGDA-UB) to a higher number of tasks. For instance, when the number of tasks increases from 8 to 40, the F-score of MGDA-UB decreases by 1.3%, while the F-score of our proposed ETR-NLP improves by 0.3% while having 28.5% fewer (11.2M to 8M) learnable parameters.

Table 5 and Table 6 show the experimental results for pixel-level dense prediction problems on NYUv2 and Cityscapes datasets, respectively. Again, we observe that ETR-NLP significantly outperforms the baselines. Furthermore, it is worth mentioning that our ETR-NLP is the highest on all metrics. For instance, as shown in Table 6, ETR-NLP obtains 61.49 mIoU for semantic segmentation, an improvement of +3.56 mIoU over the previous state-of-the-art results, while having 13.5% less learnable parameters. A similarly significant improvement is observed on the NYUv2 dataset across *all* tasks.

5. Ablation Analysis

NLP Hyperparameters: Figure 8 shows the effect of different settings for individual NLPs for image-level classification tasks. We observe that (i) A combination of different kernel sizes can improve performance even for the same type of NLP (e.g., avg pooling), as kernels of different sizes can extract diverse features. These results also indicate that multi-task learning benefits from operating on diverse features. (ii) For each type ofLP, the choice of parameters greatly impacts performance. For instance, for avg pooling, the F-score with a kernel size of 9 is 3% higher than that with a kernel size of 3.

The final design of our proposed NLP for image-level classification tasks was guided by the observations that we summarize as follows: (1) average pooling with larger kernels outperforms max pooling; (2) depth-wise convolutions outperform full convolutions while there is no appreciable difference between using real-valued or binary weights; (3) smaller convolution kernels outperform larger ones; (4) perturbation helps improve performance.

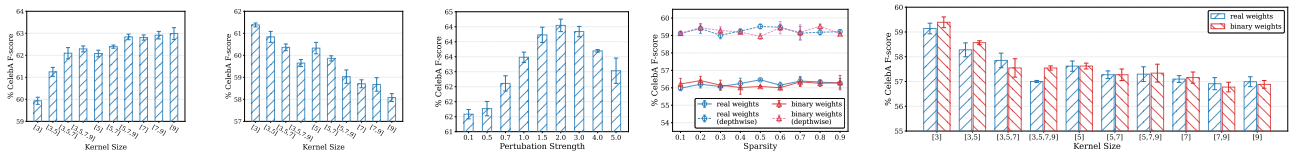
Training strategy for ETR-NLP: Table 7 shows the experimental results of "Steady-state" and "Synchronized" training strategies on the CelebA image-level and Cityscapes pixel-level prediction tasks. "Steady-state" refers to up-

Table 4. Comparison of ETR-NLP with baselines on CelebA image-level classification problems. Our results are highlighted with shading.

Method (ResNet18)	#P (M)	8 grouped facial attributes (tasks)				40 facial attributes (tasks)			
		Precision (↑)	Recall (↑)	F-score (↑)	Δ_p (↑)	Precision (↑)	Recall (↑)	F-score (↑)	Δ_p (↑)
Hard sharing	11.2	$67.7_{\pm 0.8}$	$59.8_{\pm 0.3}$	$63.0_{\pm 0.2}$	0.0%	$70.8_{\pm 0.9}$	$60.0_{\pm 0.3}$	$64.2_{\pm 0.1}$	0.0%
GradNorm ($\alpha = 0.5$) [38]	11.2	$70.4_{\pm 0.1}$	$59.5_{\pm 0.6}$	$63.6_{\pm 0.5}$	+1.5%	$70.7_{\pm 0.8}$	$60.0_{\pm 0.3}$	$64.1_{\pm 0.3}$	-0.1%
MGDA-UB [26]	11.2	$68.6_{\pm 0.1}$	$60.2_{\pm 0.3}$	$63.6_{\pm 0.3}$	+1.0%	$71.8_{\pm 0.9}$	$57.4_{\pm 0.3}$	$62.3_{\pm 0.2}$	-2.0%
Atten. hard sharing [21]	12.9	$71.1_{\pm 0.3}$	$62.6_{\pm 0.5}$	$65.9_{\pm 0.2}$	+4.8%	$73.2_{\pm 0.1}$	$63.6_{\pm 0.2}$	$67.5_{\pm 0.1}$	+4.8%
Task routing [29]	11.2	$71.7_{\pm 0.1}$	$61.7_{\pm 0.5}$	$65.5_{\pm 0.3}$	+4.4%	$72.1_{\pm 0.8}$	$63.4_{\pm 0.3}$	$66.8_{\pm 0.2}$	+3.9%
Max roaming [23]	11.2	$71.2_{\pm 0.4}$	$63.0_{\pm 0.6}$	$66.2_{\pm 0.2}$	+5.2%	$73.0_{\pm 0.4}$	$63.6_{\pm 0.1}$	$67.3_{\pm 0.1}$	+4.6%
ETR-NLP	8.0	$\mathbf{72.7}_{\pm 0.4}$	$\mathbf{64.8}_{\pm 0.3}$	$\mathbf{67.8}_{\pm 0.1}$	+7.8%	$\mathbf{73.2}_{\pm 0.2}$	$\mathbf{64.8}_{\pm 0.3}$	$\mathbf{68.1}_{\pm 0.1}$	+5.8%

Table 5. Comparison of ETR-NLP with baselines on NYU-v2 pixel-level prediction problems. Our results are highlighted with shading.

Method (SegNet)	#P (M)	Segm. mIoU (↑)	Depth Estimation (Lower better ↓)		Surface Normal Estimation Angle distance (↓)		Within t^o (↑)			Δ_p (↑)
			Abs. Err.	Rel. Err.	Mean Err.	Median Err.	11.25	22.5	30	
Hard sharing	25.1	$15.98_{\pm 0.56}$	$0.6095_{\pm 0.0041}$	$0.2554_{\pm 0.0007}$	$32.43_{\pm 0.19}$	$27.43_{\pm 0.35}$	$20.66_{\pm 0.19}$	$42.84_{\pm 0.19}$	$55.02_{\pm 0.19}$	0.0%
GradNorm [38]	25.1	$16.13_{\pm 0.23}$	$0.7626_{\pm 0.0034}$	$0.3208_{\pm 0.0050}$	$34.45_{\pm 0.52}$	$30.98_{\pm 0.80}$	$18.96_{\pm 0.60}$	$40.85_{\pm 0.92}$	$53.34_{\pm 0.24}$	-10.6%
Cross-Stitch [22]	75.3	$14.71_{\pm 0.23}$	$0.6481_{\pm 0.0034}$	$0.2871_{\pm 0.0050}$	$33.56_{\pm 0.52}$	$28.58_{\pm 0.80}$	$20.08_{\pm 0.80}$	$40.54_{\pm 0.80}$	$51.97_{\pm 0.80}$	-6.0%
MTAN [18]	44.4	$17.72_{\pm 0.23}$	$0.5960_{\pm 0.0034}$	$0.2577_{\pm 0.0050}$	$31.44_{\pm 0.52}$	$25.37_{\pm 0.80}$	$23.17_{\pm 0.80}$	$45.65_{\pm 0.80}$	$57.48_{\pm 0.80}$	+5.7%
Atten. [21]	25.1	$16.02_{\pm 0.12}$	$0.5988_{\pm 0.0112}$	$0.2630_{\pm 0.0058}$	$32.22_{\pm 0.02}$	$26.12_{\pm 0.02}$	$20.44_{\pm 0.09}$	$42.86_{\pm 0.34}$	$55.14_{\pm 0.67}$	+0.5%
Task routing [29]	25.1	$16.54_{\pm 0.02}$	$0.6354_{\pm 0.0085}$	$0.2786_{\pm 0.0090}$	$30.93_{\pm 0.19}$	$25.51_{\pm 0.28}$	$22.52_{\pm 0.36}$	$45.41_{\pm 0.82}$	$57.46_{\pm 0.37}$	+2.7%
Max roaming [23]	25.1	$17.40_{\pm 0.31}$	$0.6082_{\pm 0.0023}$	$0.2750_{\pm 0.0015}$	$30.58_{\pm 0.04}$	$24.67_{\pm 0.08}$	$23.74_{\pm 0.61}$	46.75 ± 0.41	$58.84_{\pm 0.28}$	+6.0%
ETR-NLP	25.1	$\mathbf{20.37}_{\pm 0.32}$	$\mathbf{0.5790}_{\pm 0.0067}$	$\mathbf{0.2510}_{\pm 0.0090}$	$\mathbf{28.92}_{\pm 0.05}$	$\mathbf{23.22}_{\pm 0.16}$	$\mathbf{25.38}_{\pm 0.11}$	$\mathbf{49.11}_{\pm 0.27}$	$\mathbf{61.22}_{\pm 0.23}$	+13.6%



(a) Avg. Pooling (b) Max Pooling (c) Perturbation (d) Convolution (sparsity) (e) Convolution (kernel size)

Figure 8. Effect of different hyperparameters of individual NLP.

Table 6. Comparison of ETR-NLP with baselines on Cityscapes pixel-level predictions. Our results are highlighted with shading.

Method (SegNet)	#P (M)	Segm. mIoU (↑)	Depth Estimation		Δ_p (↑)
			Abs. Err. (↓)	Rel. Err. (↓)	
Hard sharing	25.1	$56.57_{\pm 0.22}$	$0.0170_{\pm 0.0006}$	$43.99_{\pm 5.53}$	0.0%
GradNorm [38]	25.1	$56.77_{\pm 0.08}$	$0.0199_{\pm 0.0004}$	$68.13_{\pm 4.48}$	-23.9%
Cross-Stitch [22]	75.3	$50.08_{\pm 0.23}$	$0.0154_{\pm 0.0001}$	$34.49_{\pm 1.24}$	+6.5%
MTAN [18]	44.4	$53.04_{\pm 0.32}$	$0.0144_{\pm 0.0001}$	$33.63_{\pm 1.51}$	+10.9%
Atten. [21]	25.1	$55.45_{\pm 1.03}$	$0.0160_{\pm 0.0006}$	$35.72_{\pm 1.62}$	+7.6%
Task routing [29]	25.1	$56.52_{\pm 0.41}$	$0.0155_{\pm 0.0003}$	$31.47_{\pm 0.55}$	+12.4%
Max roaming [23]	25.1	$57.93_{\pm 0.20}$	$0.0143_{\pm 0.0001}$	$29.38_{\pm 1.66}$	+17.2%
ETR-NLP	22.1	$\mathbf{61.49}_{\pm 0.29}$	$\mathbf{0.0136}_{\pm 0.0001}$	$\mathbf{29.16}_{\pm 1.30}$	+20.8%

Table 7. Comparison of different training strategies for ETR on image-level classification and pixel-level prediction problems.

Method	CelebA		Cityscapes	
	Precision (↑)	Recall (↑)	mIoU (↑)	Abs. Err. (↓)
steady-state	$72.0_{\pm 0.4}$	$63.6_{\pm 0.2}$	$59.70_{\pm 0.38}$	$0.0139_{\pm 0.0002}$
synchronized	$23.5_{\pm 2.4}$	$42.1_{\pm 3.7}$	$61.22_{\pm 0.16}$	$0.0141_{\pm 0.0001}$

dating the parameters of the shared branch immediately after forwarding on one task, while "Synchronized" refers to waiting until all tasks are forwarded before updating the parameters of the shared branch. For image-level classification tasks, "steady-state" training is better, while for pixel-level prediction tasks, "synchronized" training is better.

6. Conclusion

In this paper, we present the Explicit Task Routing with Non-Learnable Primitives (ETR-NLP) module for multi-task learning. The ETR-NLP module introduces non-learnable primitives for extracting task-agnostic and diverse features and explicit task routing to extract task-specific features for each task. Both non-learnable primitives and explicit task routing can provide the flexibility needed to minimize task interference. Experiments on the CelebA dataset with multiple image-level classification tasks and on the NYU-v2 and Cityscapes datasets with multiple pixel-level prediction tasks show that our ETR-NLP method significantly outperforms state-of-the-art baselines with fewer learnable parameters and similar FLOPs across all datasets.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 62202039, 62106097, 62032003) and the National Key Research and Development Program of China (No. 2022ZD0118502).

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. 3, 6

[2] Felix JS Bragman, Ryutaro Tanno, Sebastien Ourselin, Daniel C Alexander, and Jorge Cardoso. Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels. In *International Conference on Computer Vision (ICCV)*, pages 1385–1394, 2019. 1, 3

[3] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997. 2, 6

[4] Weijie Chen, Di Xie, Yuan Zhang, and Shiliang Pu. All you need is a few shifts: Designing efficient convolutional neural networks for image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7241–7250, 2019. 2, 3, 4, 6

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016. 4, 5

[6] Michael Crawshaw. Multi-task learning with deep neural networks: A survey, 2020. CoRR abs/2009.09796. 1

[7] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *International Conference on Computer Vision (ICCV)*, pages 2650–2658, 2015. 1, 3

[8] Adam Gaier and David Ha. Weight agnostic neural networks. In *Neural Information Processing Systems (NeurIPS)*, pages 5365–5379, 2019. 2

[9] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1577–1586, 2020. 6

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 3, 6

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 6

[12] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4284–4293, 2017. 2, 3, 4, 6

[13] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018. 1, 3

[14] Iasonas 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 5454–5463, 2017. 1, 3

[15] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019. 1

[16] Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W. Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *arXiv preprint arXiv: 2111.10603*, 2022. 6

[17] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. In *Advance in Neural Information Processing Systems (NeurIPS)*, pages 12037–12047, 2019. 1, 3

[18] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1871–1880, 2019. 1, 3, 5, 8

[19] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision (ICCV)*, pages 3730–3738, 2015. 4, 5

[20] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002. 2

[21] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1851–1860, 2019. 1, 3, 4, 5, 7, 8

[22] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *IEEE Conference on Computer Vision and pattern Recognition (CVPR)*, pages 3994–4003, 2016. 1, 3, 8

[23] Lucas Pascal, Pietro Michiardi, Xavier Bost, Benoit Huet, and Maria A Zuluaga. Maximum roaming multi-task learning. In *AAAI Conference on Artificial Intelligence*, pages 9331–9341, 2021. 1, 2, 3, 4, 5, 7, 8

[24] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11890–11899, 2020. 2

[25] Sebastian Ruder. An overview of multi-task learning in deep neural networks, 2017. CoRR abs/1706.05098. 1

[26] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 525–536, 2018. 1, 3, 8

[27] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision (ECCV)*, pages 746–760, 2012. 5

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

In *International Conference on Learning Representations (ICLR)*. 3, 6

[29] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Many task learning with task routing. In *International Conference on Computer Vision (ICCV)*, pages 1375–1384, 2019. 1, 2, 3, 4, 5, 7, 8

[30] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 3

[31] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008. 7

[32] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3614–3633, 2022. 1, 2, 6

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, 2017. 3

[34] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9127–9135, 2018. 2, 3, 4

[35] Felix Juefei Xu, Vishnu Naresh Boddeti, and Marios Savvides. Perturbative neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3310–3318, 2018. 2, 3, 4

[36] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10809–10819, 2022. 2

[37] Yu Zhang and Qiang Yang. A survey on multi-task learning, 2017. CoRR abs/1707.08114. 1, 2, 6

[38] Chen Zhao, Badrinarayanan Vijay, Lee Chen-Yu, and Rabinovich Andrew. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning (ICML)*, pages 793–802, 2018. 1, 3, 8

[39] Xiangyun Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. A modulation module for multi-task learning with applications in image retrieval. In *European Conference on Computer Vision (ECCV)*, pages 415–432, 2018. 1