InterroGate: Learning to Share, Specialize, and Prune Representations for Multi-task Learning

Babak Ehteshami Bejnordi 1 Gaurav Kumar 2 Amelie Royer 1 Christos Louizos 1 Tijmen Blankevoort $^{\dagger\,1}$ Mohsen Ghafoorian 3

Abstract

Jointly learning multiple tasks with a unified model can improve accuracy and data efficiency, but it faces the challenge of task interference, where optimizing one task objective may inadvertently compromise the performance of another. A solution to mitigate this issue is to allocate taskspecific parameters, free from interference, on top of shared features. However, manually designing such architectures is cumbersome, as practitioners need to balance between the overall performance across all tasks and the higher computational cost induced by the newly added parameters. In this work, we propose InterroGate, a novel MTL architecture designed to mitigate task interference while optimizing inference computational efficiency. We employ a learnable gating mechanism to automatically balance the shared and task-specific representations while preserving the performance of all tasks. Crucially, the patterns of parameter sharing and specialization dynamically learned during training, become fixed at inference, resulting in a static, optimized MTL architecture. Through extensive empirical evaluations, we demonstrate SoTA results on three MTL benchmarks using convolutional as well as transformer-based backbones on CelebA, NYUDv2, and PASCAL-Context.

1. Introduction

Multi-task learning (MTL) involves learning multiple tasks concurrently with a unified architecture. By leveraging the shared information among related tasks, MTL has the po-

tential to improve accuracy and data efficiency. In addition, learning a joint representation reduces the computational and memory costs of the model at inference as visual features relevant to all tasks are extracted only once: This is crucial for many real-life applications where a single device is expected to solve multiple tasks simultaneously under strict compute constraints (e.g. mobile phones, extended reality, self-driving cars, etc.). Despite these potential benefits, in practice, MTL is often met with a key challenge known as negative transfer or task interference (Zhao et al., 2018), which refers to the phenomenon where the learning of one task negatively impacts the learning of another task during joint training. While characterizing and solving task interference is an open issue (Wang et al., 2019; Royer et al., 2023), there exist two major lines of work to mitigate this problem: (i) Multi-task Optimization (MTO) techniques aim to balance the training process of each task, while (ii) architectural designs carefully allocate shared and task-specific parameters to reduce interference.

Specifically, MTO approaches balance the losses/gradients of each task to mitigate the extent of gradient conflicts while optimizing the shared features. However, the results may still be compromised when the tasks rely on inherently different visual cues, making sharing parameters difficult. For instance, text detection and face recognition require learning very different texture information and object scales. An alternative and orthogonal research direction is to allocate additional task-specific parameters, on top of shared generic features, to bypass task interference. In particular, several state-of-the-art methods have proposed task-dependent selection and adaptation of shared features (Guo et al., 2020; Sun et al., 2020; Wallingford et al., 2022; Rahimian et al., 2023). However, the dynamic allocation of task-specific features is usually performed one task at a time, and solving all tasks still requires multiple forward passes. Alternatively, Mixture of Experts (MoE) have also been employed to reduce the computational cost of MTL by dynamically routing inputs to a subset of experts (Ma et al., 2018; Hazimeh et al., 2021; Fan et al., 2022; Chen et al., 2023). However, the input-dependent routing of MoE is typically hard to efficiently deploy at inference, particularly with batched execution (Sarkar et al., 2023; Yi et al., 2023).

¹Qualcomm AI Research*, Amsterdam, The Netherlands.
²Qualcomm AI Research, Hyderabad, India.
³XR Labs, Qualcomm Technologies Inc., Amsterdam, The Netherlands.

^{*} Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc. † Work was completed while an employee at Qualcomm AI Research.
Preprint. Under review.

In contrast to previous dynamic architectures, we learn to balance shared and task-specific features jointly for all tasks, which allows us to predict all task outputs in a single forward pass. In addition, we propose to regulate the expected inference computational cost through a budget-aware regularization during training. By doing so, we aim to depart from a common trend in MTL that heavily focuses on accuracy while neglecting computational efficiency (Misra et al., 2016; Vandenhende et al., 2020).

In this paper, we introduce InterroGate, a novel MTL architecture to mitigate task interference while optimizing computational efficiency during inference. Our method learns the per-layer optimal balance between sharing and specializing representations for a desired computational budget. In particular, we leverage a shared network branch which is used as a general communication channel through which the task-specific branches interact with each other. This communication is enabled through a novel gating mechanism which learns for each task and layer to select parameters from either the shared branch or their task-specific branch. To enhance the learning of the gating behaviour, we harness single task baseline weights to initialize task-specific branches.

InterroGate primarily aims to optimize efficiency in the inference phase, crucial in real-world applications. While the gate dynamically learns to select between a large pool of task-specific and shared parameters during training, at inference, the learned gating patterns are static and thus can be used to prune the unselected parameters in the shared and task-specific branches: As a result, InterroGate collapses to a simpler, highly efficient, static architecture at inference time, suitable for batch processing. We control the trade-off between inference computational cost and multi-task performance, by regularizing the gates using a sparsity objective. In summary, our contributions are as follows:

- We propose a novel multi-task learning framework that learns the optimal parameter sharing and specialization patterns for all tasks, in tandem with the model parameters, enhancing multi-task learning efficiency and effectiveness.
- We enable a training mechanism to control the trade-off between multi-task performance and inference compute cost. Our proposed gating mechanism finds the optimal balance between selecting shared and specialized parameters for each task, within a desired computational budget, controlled with a sparsity objective. This, subsequently, enables a simple process for creating a range of models on the efficiency/accuracy trade-off spectrum, as opposed to most other MTL methods.
- Our proposed method is designed to optimize inference-time efficiency. Post-training, the unselected

parameters by the gates are pruned from the model, resulting in a simpler, highly efficient neural network. In addition, our feature fusion strategy allows to predict all tasks in a single forward pass, critical in many real-world applications.

 Through extensive empirical evaluations, we report SoTA results consistently on three multi-tasking benchmarks with various convolutional and transformerbased backbones. We then further investigate the proposed framework through ablation experiments.

2. Related Work

Multi-task Optimization (MTO) methods aim to automatically balance the different tasks when optimizing shared parameters to maximize average performance. Loss-based methods (Kendall et al., 2018; Liu et al., 2022) are usually scalable and adaptively scale task losses based on certain statistics (e.g. task output variance); Gradient-based methods (Sener & Koltun, 2018; Chen et al., 2018b; 2020; Liu et al., 2021; Javaloy & Valera, 2021) are more costly in practice as they require storing a gradient per task, but usually yield higher performance. Orthogonal to these optimization methods, several research directions investigate how to design architectures that inherently mitigate task-interference, as described below.

Task Grouping approaches investigate which groups of tasks can safely share encoder parameters without task interference. For instance (Standley et al., 2020; Fifty et al., 2021) identify "task affinities" as a guide to isolate parameters of tasks most likely to interfere with one another. Similarly, (Guo et al., 2020) apply neural architecture search techniques to design MTL architectures. However, exploring these large architecture search spaces is a costly process.

Hard Parameter Sharing works such as Cross-Stitch (Misra et al., 2016), MTAN (Liu et al., 2019) or MuIT (Bhattacharjee et al., 2022) propose to learn the task parameter sharing design alongside the model features. However, most of these works mainly focus on improving the accuracy of the model while neglecting the computational cost: For instance, (Misra et al., 2016; Gao et al., 2020) require a full network per task and improve MTL performance through lightweight adapters across task branches. (Liu et al., 2019; Bhattacharjee et al., 2022) use task-specific attention module on top of a shared feature encoder, but the cost of the task-specific decoder heads often dominates the final architecture.

Conditional Compute approaches learn task-specific gating of model parameters: For instance (Sun et al., 2020; Wallingford et al., 2022) learn to select a subset of the most relevant layers when adapting the network to a new downstream task. Piggyback (Mallya et al., 2018) adapts a pretrained network

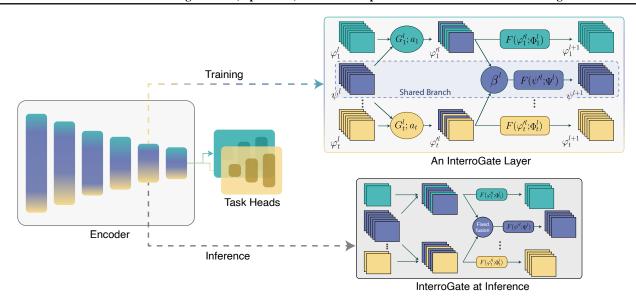


Figure 1. Overview of the proposed InterroGate framework: The original encoder layers are substituted with InterroGate layers. The input to the layer is t+1 feature maps, one shared representation and t task-specific representations. To decide between shared ψ^{ℓ} or task-specific φ_t^{ℓ} features, each task relies on its own gating module G_t^{ℓ} . The resulting channel-mixed feature-map $\varphi_t^{\prime\ell}$ is then fed to the next task-specific layer. The input to the shared branch for the next layer is constructed by linearly combining the task-specific features of all tasks using the learned parameter β_t^{ℓ} . During inference, the parameters (shared or task-specific) that are not chosen by the gates are removed from the model, resulting in a plain neural network architecture.

to multiple tasks by learning a set of per-task sparse masks for the network parameters. Similarly, (Berriel et al., 2019) select the most relevant feature channels using learnable gates. Finally, Mixture of Experts (MoE) (Ma et al., 2018; Hazimeh et al., 2021; Fan et al., 2022; Chen et al., 2023) leverage sparse gating to select a subset of experts for each input example.

Nevertheless, due to the dynamic nature of these works at inference, obtaining all task predictions is computationally inefficient as it often requires one forward pass through the model per task. Therefore, these methods are less suited for MTL settings requiring solving all tasks concurrently. Additionally, dynamic expert selection in MoEs requires either storing all expert weights on-chip, increasing memory demands, or frequent off-chip data transfers to load necessary experts, leading to significant overheads, complicating their efficient inference on resource-constrained devices (Sarkar et al., 2023; Yi et al., 2023). In contrast, InterroGate focuses on optimizing efficiency and is capable of addressing all tasks simultaneously, aligning with many practical real-world needs. InterroGate employs learned gating patterns to prune unselected parameters, resulting in a more streamlined and efficient static architecture during inference, well-suited for batch processing. Finally, closest to our work is (Bragman et al., 2019), which proposes a probabilistic allocation of convolutional filters as task-specific or shared. However, this design only allows for the shared features to send information to the task-specific branches. In contrast, our gating mechanism allows for information to

flow in any direction between the shared and task-specific features, thereby enabling cross-task transfer in every layer.

3. InterroGate

Given T tasks, we aim to learn a flexible allocation of shared and task-specific parameters, while optimizing the trade-off between accuracy and efficiency. Specifically, an Interro-Gate model is characterized by task-specific parameters $\{\Phi_t\}_{t=1}^T$ and shared parameters Ψ . In addition, discrete gates (with parameters α) are trained to only select a subset of the most relevant features in both the shared and task-specific branches, thereby reducing the model's computational cost. Under this formalism, the model and gate parameters are trained end-to-end by minimizing the classical MTL objective:

$$\mathcal{L}(\{\Phi_t\}_{t=1}^T, \Psi, \alpha) = \sum_{t=1}^T \omega_t \, \mathcal{L}_t(X, Y_t; \Phi_t, \Psi, \alpha), \quad (1)$$

where X and Y_t are the input data and corresponding labels for task t, \mathcal{L}_t represents the loss function associated to task t, and ω_t are hyperparameter coefficients which allow for balancing the importance of each task in the overall objective. In the rest of the section, we describe how we learn and implement the feature-level routing mechanism characterized by α . We focus on convolutional architectures in Section 3.1, and discuss the case of transformer-based models in Appendix C.

3.1. Learning to Share, Specialize and Prune

Figure 1 presents an overview of the proposed InterroGate framework. Formally, let $\psi^{\ell} \in \mathbb{R}^{C^{\ell} \times W^{\ell} \times H^{\ell}}$ and $\varphi^{\ell}_{t} \in \mathbb{R}^{C^{\ell} \times W^{\ell} \times H^{\ell}}$ are represent the shared and task-specific features at layer ℓ of our multi-task network, respectively. In each layer ℓ , the gating module G^{ℓ}_{t} of task t selects relevant channels from either ψ^{ℓ} and φ^{ℓ}_{t} . The output of this hard routing operation yields features φ'^{ℓ}_{t} :

$$\varphi_t^{\prime\ell} = G_t^{\ell}(\alpha_t^{\ell}) \odot \varphi_t^{\ell} + (1 - G_t^{\ell}(\alpha_t^{\ell})) \odot \psi^{\ell}, \tag{2}$$

where \odot is the Hadamard product and $\alpha_t^\ell \in \mathbb{R}^{C^\ell}$ denotes the learnable gate parameters for task t at layer ℓ and the gating module G_t^ℓ outputs a vector in $\{0,1\}^{C^\ell}$, encoding the binary selection for each channel. These intermediate features are then fed to the next task-specific layer to form the features $\varphi_t^{\ell+1} = F(\varphi_t'^\ell; \Phi_t^\ell)$.

Similarly, we construct the shared features of the next layer $\ell+1$ by mixing the previous layer's task-specific feature maps. However, how to best combine these T feature maps is a harder problem than the pairwise selection described in (2). Therefore, we let the model learn its own soft combination weights and the resulting mixing operation for the shared features is defined as follows:

$$\psi^{\ell} = \sum_{t=1}^{T} \underset{t=1...T}{\operatorname{softmax}} (\beta_t^{\ell}) \odot \varphi_t^{\ell}, \tag{3}$$

where $\beta^\ell \in \mathbb{R}^{C^\ell \times T}$ denotes the learnable parameters used to linearly combine the task-specific features and form the shared feature map of the next layer. Similar to the task-specific branch, these intermediate features are then fed to a convolutional block to form the features $\psi^{\ell+1} = F(\psi^{\prime\ell}; \Psi^\ell)$. Finally, note that there is no direct information flow between the shared features of one layer to the next, i.e., ψ^ℓ and $\psi^{\ell+1}$: Intuitively, the shared feature branch can be interpreted as a general communication channel which the task-specific branches communicate with one another.

3.2. Implementing the Discrete Routing Mechanism

During **training**, the model features and gates are trained jointly and end-to-end. In (2), the gating modules G_t^ℓ each output a binary vector over channels in $\{0,1\}^C$, where 0 means choosing the shared feature at this channel index, while 1 means choosing the specialized feature for the respective task t. In practice, we implement G as a sigmoid operation applied to the learnable parameter α , followed by a thresholding operation at 0.5. Due to the non-differentiable nature of this operation, we adopt the straight-through estimation (STE) during training (Bengio et al., 2013): In the backward pass, STE approximates the gradient flowing through the thresholding operation as the identity function.

At **inference**, since the gate modules do not depend on the input data, our proposed InterroGate method converts to a static neural network architecture, where feature maps are pruned following the learned gating patterns: To be more specific, for a given layer ℓ and task t, we first collect all channels for which the gate $G_t^\ell(\alpha_t^\ell)$ outputs 0; Then, we simply prune the corresponding task-specific weights in $\Phi_t^{\ell-1}$. Similarly, we can prune away weights from the shared branch $\Psi^{\ell-1}$ if the corresponding channels are never chosen by any of the tasks in the mixing operation of (2). The pseudo-code for the complete unified encoder forward-pass is detailed in Appendix B.

3.3. Sparsity Regularization

During training, we additionally control the proportion of shared versus task-specific features usage by regularizing the gating module G. This allows us to reduce the computational cost, as more of the task-specific weights can be pruned away at inference. We implement the regularizer term as a hinge loss over the gating activations for task-specific features:

$$\mathcal{L}_{\text{sparsity}}(\alpha) = \frac{1}{T} \sum_{t=1}^{T} \max \left(0, \ \frac{1}{L} \sum_{\ell=1}^{L} \sigma(\alpha_t^{\ell}) - \tau_t \right), \quad (4)$$

where σ is the sigmoid function and τ_t is a task-specific hinge target. The parameter τ allows to control the proportion of active gates at each specific layer by setting a soft upper limit for active task-specific parameters. A lower hinge target value encourages more sharing of features while a higher value gives more flexibility to select task-specific features albeit at the cost of higher computational cost.

Our final training objective is a combination of the multitask objective and sparsity regularizer:

$$\mathcal{L} = \mathcal{L}(\{\Phi_t\}_{t=1}^T, \Psi, \alpha, \beta) + \lambda_s \mathcal{L}_{\text{sparsity}}(\alpha),$$
 (5)

where λ_s is a hyperparameter balancing the two losses.

4. Experiments

4.1. Experimental Setup

Datasets and Backbones. We evaluate the performance of InterroGate on three popular datasets: CelebA (Liu et al., 2015), NYUD-v2 (Silberman et al., 2012), and PASCAL-Context (Chen et al., 2014). CelebA is a large-scale face attributes dataset, consisting of more than 200k celebrity images, each labeled with 40 attribute annotations. We consider the age, gender, and clothes attributes to form three output classification tasks for our MTL setup and use binary cross-entropy to train the model. The NYUD-v2 dataset is designed for semantic segmentation and depth estimation

tasks. It comprises 795 train and 654 test images taken from various indoor scenes in New York City, with pixel-wise annotation for semantic segmentation and depth estimation. Following recent work (Xu et al., 2018; Zhang et al., 2019; Maninis et al., 2019), we also incorporate the surface normal prediction task, obtaining annotations directly from the depth ground truth. We use the mean intersection over union (mIoU) and root mean square error (rmse) to evaluate the semantic segmentation and depth estimation tasks, respectively. We use the mean error (mErr) in the predicted angles to evaluate the surface normals. Finally, the PASCAL-Context dataset is an extension of the PASCAL VOC 2010 dataset (Everingham et al., 2010) and provides a comprehensive scene understanding benchmark by labeling images for semantic segmentation, human parts segmentation, semantic edge detection, surface normal estimation, and saliency detection. The dataset consists of 4,998 train images and 5,105 test images. The semantic segmentation, saliency estimation, and human part segmentation tasks are evaluated using mIoU. Similar to NYUD, mErr is used to evaluate the surface normal predictions.

We use ResNet-20 (He et al., 2016) as the backbone in our CelebA experiments, with simple linear heads for the task-specific predictions. For the NYUD-v2 dataset, we use ResNet-50 with dilated convolutions and HRNet-18 following (Vandenhende et al., 2021). We also present results using a dense prediction transformer (DPT) (Ranftl et al., 2021), with a ViT-base and -small backbone. Finally, on PASCAL-Context, we use a ResNet-18 backbone. For both NYUD and PASCAL, we use dense prediction heads to output the task predictions, as described in Appendix A.

SoTA Baselines and Metrics. To establish upper and lower bounds of MTL performance, we always compare our models to the Single-Task baseline (STL), which is the performance obtained when training an independent network for each task, as well as the uniform MTL baseline where the model's encoder (backbone) is shared by all tasks.

In addition, we compare InterroGate to encoder-based methods including Cross-stitch (Misra et al., 2016) and MTAN (Liu et al., 2019), as well as MTO approaches such as uncertainty weighting (Kendall et al., 2018), DWA (Liu et al., 2019), and Auto- λ (Liu et al., 2022), PCGrad (Yu et al., 2020), CAGrad (Liu et al., 2021), MGDA-UB (Sener & Koltun, 2018), and RLW (Lin et al., 2022). Following (Maninis et al., 2019), our main metric is the multi-task performance $\Delta_{\rm MTL}$ of a model m as the averaged normalized drop in performance w.r.t. the single-task baselines b:

$$\Delta_{\text{MTL}} = \frac{1}{T} \sum_{i=1}^{T} (-1)^{l_i} \left(M_{m,i} - M_{b,i} \right) / M_{b,i}$$
 (6)

where $l_i = 1$ if a lower value means better performance for

metric M_i of task i, and 0 otherwise. Furthermore, similar to (Navon et al., 2022), we compute the mean rank (MR) as the average rank of each method across the different tasks, where a lower MR indicates better performance. All reported results for InterroGate and baselines are averaged across 3 random seeds.

Finally, to generate the trade-off curve between MTL performance and compute cost of InterroGate in Figure 2 and all the tables, we sweep over the gate sparsity regularizer weight, λ_s , in the range of $\{1,3,5,7,10\} \cdot 10^{-2}$. The task-specific targets τ in (4) also impact the computation cost. Intuitively, tasks with significant performance degradation benefit from more task-specific parameters, i.e., from higher values of the hyperparameters τ_t . We analyze the gating patterns for sharing and specialization in section 4.3.2 and, through ablation experiments, we further discuss the impact of sparsity targets $\{\tau_t\}_{t=1}^T$ in Appendix E.

While InterroGate primarily aims at improving the inference cost efficiency, we also measure and report training time comparison between our method and the baselines, on the PASCAL-Context (Chen et al., 2014) dataset in Appendix F.

Training Pipeline. For initialization, we use pretrained ImageNet weights for the single-task and multi-task baseline. For InterroGate, the shared branch is initialized with ImageNet weights while the task-specific branches are with their corresponding single-task weights. Finally, we discover that employing a separate optimizer for the gates improves the convergence of the model. We further describe training hyperparameters in Appendix A. All experiments were conducted on a single NVIDIA V100 GPU and we use the same training setup and hyperparameters across all MTL approaches included in our comparison.

4.2. Results

In this section, we present the results of InterroGate, single-task (STL), multi-task baselines, and the competing MTL approaches, on the CelebA, NYUD-v2, and PASCAL-Context datasets. In Section 4.3, we will also present several ablation studies on (i) how we design the sparsity regularization loss, (ii) on the qualitative sharing/specialization patterns learned by the gate, and finally, (ii) on the impact of the backbone model capacity on MTL performance. As mentioned earlier, we also present an ablation on the impact of the sparsity targets $\{\tau_t\}_{t=1}^T$ in Appendix E.

4.2.1. CELEBA

Figure 2a shows the trade-off between MTL performance and the computational cost (FLOPs) for InterroGate, MTL uniform, and STL baselines on the CelebA dataset, for 3 different widths for the ResNet-20 backbone: quarter, half, and original capacity. We report the detailed results in Ta-

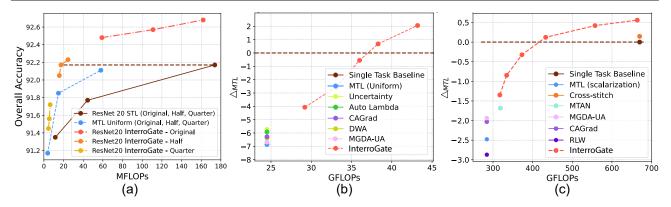


Figure 2. Accuracy vs. floating-point operations (FLOP) trade-off curves for InterroGate and SoTA MTL methods. (a) Results on CelebA using ResNet-20 backbone at three different widths (Original, Half, and Quarter). (b) NYUD-v2 using HRNet-18 backbone, and (c) ResNet-18 on PASCAL-Context. To avoid clutter, we present the six highest-performing MTL baselines in (b) & (c). The single task baseline in (b) has 65.1 GFLOPs.

ble 8, Appendix D. InterroGate outperforms MTL uniform and STL baselines with higher overall accuracy at a much lower computational cost. Most notably, the performance of InterroGate with ResNet-20 half width at only 14.8 MFlops matches the performance of STL with 174 MFlops. Finally, we further discuss the behavior of InterroGate and MTL baselines across different model capacities in the ablation experiment in section 4.3.3.

4.2.2. NYUD-v2

Table 1 and 2 present the results on the NYUD-v2 dataset, using the HRNet-18 and ResNet-50 backbones, respectively. Figure 2b illustrates the trade-off between Δ_{MTL} and computational cost of various methods using the HRNet-18 backbone. We additionally report the parameter count and the standard deviation of the Δ_{MTL} scores in Tables 9 and 10, in Appendix D. As can be seen, most MTL methods improve the accuracy on the segmentation and depth estimation tasks, while surface normal prediction significantly drops. While MTL uniform and MTO strategies operate at the lowest computational cost by sharing the full backbone, they fail to compensate for this drop in performance. MTO approaches often show mixed results across the two backbones. While CAGrad (Liu et al., 2021) and uncertainty weighting (Kendall et al., 2018) are the best performing MTO baselines using ResNet-50 and HRNet-18 backbones, respectively, they show negligible improvement over uniform MTL when applied to the alternate backbones. In contrast, among the encoder-based methods, Cross-stitch largely retains performance on normal estimation and achieves a positive Δ_{MTL} score of +1.66. However, this comes at a substantial computational cost and parameter count, close to that of the STL baseline. In comparison, InterroGate achieves an overall $\Delta_{\rm MTL}$ score of +2.06 and +2.04 using HRNet-18 and ResNet-50, respectively, at a lower computational cost. At an equal parameter count of 92.4 M, InterroGate surpasses MTAN using the ResNet-50 backbone, exhibiting a Δ_{MTL}

score of +1.16, in contrast to MTAN's -0.84.

Table 1. Performance comparison on NYUD-v2 using HRNet-18.

MODEL	Semseg \uparrow	Depth \downarrow	Normals \downarrow	$\Delta_{\mathrm{MTL}} \ (\%) \uparrow$	FLOPS (G)	$MR\!\downarrow$
STL	41.70	0.582	18.89	0	65.1	8.0
MTL (UNI.)	41.83	0.582	22.84	-6.86	24.5	11.0
DWA	41.86	0.580	22.61	-6.29	24.5	8.7
UNCERTAINTY	41.49	0.575	22.27	-5.73	24.5	8.3
Auto- λ	42.71	0.577	22.87	-5.92	24.5	8.0
RLW	42.10	0.593	23.29	-8.09	24.5	11.7
PCGRAD	41.75	0.581	22.73	-6.70	24.5	10.3
CAGRAD	42.31	0.580	22.79	-6.28	24.5	8.7
MGDA-UB	41.23	0.625	21.07	-6.68	24.5	11.3
INTERROGATE	43.58	0.559	19.32	+2.06	43.2	1.3
INTERROGATE	42.95	0.562	19.73	+0.68	38.3	2.3
INTERROGATE	42.36	0.564	20.04	-0.55	36.0	4.0
INTERROGATE	42.73	0.575	21.01	-2.55	33.1	4.0
Interrogate	42.35	0.575	21.70	-4.07	29.2	5.7

Table 2. Performance comparison on NYUD-v2 using ResNet-50.

MODEL	Semseg \uparrow	Depth \downarrow	Normals \downarrow	$\Delta_{\mathrm{MTL}}\left(\%\right)\uparrow$	FLOPS (G)	$MR\!\downarrow$
STL	43.20	0.599	19.42	0	1149	9.0
MTL (UNI.)	43.39	0.586	21.70	-3.04	683	9.7
DWA	43.60	0.593	21.64	-3.16	683	9.7
UNCERTAINTY	43.47	0.594	21.42	-2.95	683	10.0
Auto- λ	43.57	0.588	21.75	-3.10	683	10.0
RLW	43.49	0.587	21.54	-2.74	683	8.3
PCGRAD	43.74	0.588	21.55	-2.66	683	7.3
CAGRAD	43.57	0.583	21.55	-2.49	683	7.0
MGDA-UB	42.56	0.586	21.76	-3.83	683	11.3
MTAN	44.92	0.585	21.14	-0.84	683	4.0
CROSS-STITCH	44.19	0.577	19.62	+1.66	1151	2.7
INTERROGATE	44.38	0.576	19.50	+2.04	916	1.7
INTERROGATE	43.63	0.577	19.66	+1.16	892	3.7
INTERROGATE	43.05	0.589	19.95	-0.50	794	9.7

Table 3 and 4 report the performance of DPT trained models with the ViT-base and ViT-small backbones. The MTL uniform and MTO baselines, display reduced computational cost, yet once again manifesting a performance drop in the normals prediction task. Similar to the trend between HRNet-18 and ResNet-50, the performance drop is more

substantial for the smaller model, ViT-small, indicating that task interference is more prominent in small capacity settings. In comparison, InterroGate consistently demonstrates a more favorable balance between the computational cost and the overall MTL accuracy across varied backbones.

Table 3. Results on NYUD-v2 using DPT with ViT-base.

MODEL	Semseg ↑	ДЕРТН ↓	Normals ↓	$\Delta_{\mathrm{MTL}} (\%) \! \uparrow$	FLOPS (G)	MR↓
STL	51.65	0.548	19.04	0	759	5.0
MTL (UNI.)	51.38	0.539	20.73	-2.57	294	7.3
DWA	51.66	0.536	20.98	-2.66	294	6.0
UNCERTAINTY	51.87	0.5352	20.72	-2.02	294	4.0
INTERROGATE	51.98	0.528	19.10	+1.32	626	1.3
INTERROGATE	51.46	0.536	19.34	+0.08	483	5.0
INTERROGATE	51.66	0.534	20.16	-1.10	387	3.7
INTERROGATE	51.71	0.535	20.38	-1.51	324	3.7

Table 4. Results on NYUD-v2 using DPT with ViT-small.

MODEL	Semseg ↑	Depth \downarrow	Normals \downarrow	$\Delta_{\mathrm{MTL}}\left(\%\right)\uparrow$	FLOPS (G)	$MR\downarrow$
STL	46.58	0.583	21.22	0	248	4.0
MTL (UNI.)	45.32	0.576	22.86	-3.04	118	7.3
DWA	45.74	0.5721	22.94	-2.68	118	5.7
UNCERTAINTY	45.67	0.5737	22.80	-2.60	229	5.7
INTERROGATE	45.96	0.5648	20.77	+1.30		1.7
INTERROGATE	45.34	0.5671	20.96	+0.43	183	4.0
INTERROGATE	45.57	0.5666	21.36	+0.00	168	4.0
INTERROGATE	45.99	0.5713	22.02	-1.01	132	3.7

4.2.3. PASCAL-CONTEXT

Table 5 summarizes the results of our experiments on the PASCAL-context dataset encompassing five tasks. Note that following previous work, we use the task losses' weights ω_t from (Maninis et al., 2019) for all MTL methods, but also report MTL uniform results as a reference. Figure 2c illustrates the trade-off between Δ_{MTL} and the computational cost of all models. The STL baseline outperforms most methods on the semantic segmentation and normals prediction tasks with a score of 14.70 and 66.1, while incurring a computational cost of 670 GFlops. Among the baseline MTL and MTO approaches, there is a notable degradation in surface normal prediction. Finally, as witnessed in prior works (Maninis et al., 2019; Vandenhende et al., 2020; Brüggemann et al., 2021), we observe that most MTL and MTO baselines struggle to reach STL performance. Among competing methods, MTAN and MGDA-UB yield the best MTL performance versus computational cost tradeoff, however, both suffer from a notable decline in normals prediction performance.

At its highest compute budget (no sparsity loss and negligible computational savings), InterroGate outperforms the STL baseline, notably in Saliency and Human parts prediction tasks, and achieves an overall $\Delta_{\rm MTL}$ of +0.56. As we reduce the computational cost by increasing the sparsity loss weight λ_s , we observe a graceful decline in the multi-task performance that outperforms competing methods. Our InterroGate models additionally obtain more favorable MR scores compared to the baselines. This emphasizes our

model's ability to maintain a favorable balance between compute cost and multi-task performance across computational budgets.

Table 5. Performance comparison on PASCAL-Context.

Model	Semseg ↑	Normals ↓	Saliency ↑	Human 1	`Edge↓ ₄	$\Delta_{ m MTL}(\%)$ 1	Flops (G) MR↓
STL	66.1	14.70	0.661	0.598	0.0175	0	670	6.0
MTL (uniform)	65.8	17.03	0.641	0.594	0.0176	-4.14	284	12.0
MTL (Scalar)	64.3	15.93	0.656	0.586	0.0172	-2.48	284	10.6
DWA	65.6	16.99	0.648	0.594	0.0180	-3.91	284	12.0
Uncertainty	65.5	17.03	0.651	0.596	0.0174	-3.68	284	10.2
RLW	65.2	17.22	0.660	0.634	0.0177	-2.87	284	9.2
PCGrad	62.6	15.35	0.645	0.596	0.0174	-2.58	284	12.0
CAGrad	62.3	15.30	0.648	0.604	0.0174	-2.03	284	10.2
MGDA-UB	63.0	15.34	0.646	0.604	0.0174	-1.94	284	10.2
Cross-stitch	66.3	15.13	0.663	0.602	0.0171	+0.14	670	4.0
MTAN	65.1	15.76	0.659	0.590	0.0170	-1.78	319	9.0
InterroGate	65.7	14.71	0.663	0.606	0.0172	+0.56	664	3.2
InterroGate	65.1	14.64	0.663	0.604	0.0172	+0.42	577	4.8
InterroGate	65.2	14.75	0.663	0.600	0.0172	+0.12	435	5.4
InterroGate	64.9	14.72	0.658	0.596	0.0172	-0.28	377	7.6
InterroGate	65.1	15.02	0.655	0.592	0.0172	-0.85	334	8.8

Table 6. Comparing the MTL performance using the L_1 Hinge loss and the standard L_1 loss on PASCAL-Context.

Model $\mathcal{L}_{\text{sparsity}}$	Semseg ↑	Normals ↓	Saliency 1	Human 1	↑Edge ↓∠	Δ _{MTL} (%)	↑Flops (G)MR↓
InterroGate None	65.7	14.71	0.663	0.606	0.0172	+0.56	664	1.8
InterroGate L_1 InterroGate L_1	63.9	14.74	0.664	0.600	0.0172	-0.27	623	2.8
	61.1	15.07	0.663	0.582	0.0172	-2.20	518	4.0
InterroGate Hinge	65.1	14.64	0.648	0.604	0.0171	+0.28	557	2.2
InterroGate Hinge	65.2	14.75	0.644	0.600	0.0172		433	3.4

4.3. Ablation Studies

4.3.1. Sparsity Loss

To study the effect of the sparsity loss defined in Equation 4, we conduct the following two experiments: First, we omit the sparsity regularization loss ($\lambda_s=0$): As can be seen in the first row of Table 6, InterroGate outperforms the single task baseline, but the computational savings are very limited. In the second ablation experiment, we compare the use of the L_1 hinge loss with a standard L_1 loss function as the sparsity regularizer: The results of Table 6 show that the hinge loss formulation consistently yields better trade-offs.

4.3.2. LEARNED SHARING AND SPECIALIZATION PATTERNS

We then investigate the gating patterns that the model converges to. Specifically, we want to observe how much each task contributes to and benefits from the shared representations. To that aim, we monitor (i) the percentage of task-specific representations selected by each task (captured by the gates $G_t(\alpha_t)$), as well as (ii) how much the features specific to each task contribute to the formation of the shared feature bank (captured by the learned combination weights β); We visualize these values for the five tasks of the Pascal-Context dataset in Figure 3 for different sparsity regularizers: with hinge loss (*left*), with L_1 loss at medium (middle) and

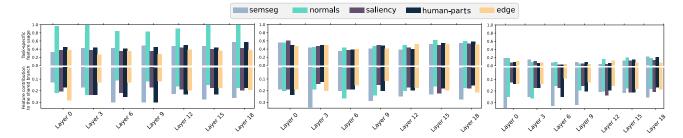


Figure 3. The task-specific representation selection ratio (top) versus proportions of maximum contributions to the shared branch (bottom) for InterroGate with hinge loss (left), L_1 loss with medium pruning (middle) and L_1 loss with high pruning (right).

high pruning levels (*right*). We also present these results for all layers in Figure 4, Appendix D.

In all settings, the semantic segmentation task makes the largest contribution to the shared branch, followed by the normals prediction tasks. It is worth noting that the amount of feature contribution to the shared branch can also be largely influenced by other tasks' loss functions. In this situation, we observe that if the normal task lacks enough task-specific features (as seen in the middle and right models), its performance deteriorates significantly. In contrast, when it acquires sufficient task-specific features, it maintains a high accuracy (*left*). Intriguingly, the features of the normal task become less interesting to other tasks in this scenario possibly due to increased specialization.

4.3.3. IMPACT OF MODEL CAPACITY

In this section, we conduct an ablation study to analyze the relationship between model capacity and multi-task performance. We progressively reduce the width of ResNet-50 and ResNet-20 to half and a quarter of the original sizes for NYUD-v2 and CelebA datasets, respectively. Shrinking the model size, as observed in Table 7, incurs progressively more harmful effect on multi-task performance compared to the single task baseline. In comparison, our proposed InterroGate approach consistently finds a favorable trade-off between capacity and performance and improves over single task performances, across all capacity ranges.

5. Discussion and Conclusion

In this paper, we propose InterroGate, a novel framework to address the fundamental challenges of task interference and computational constraints in MTL. InterroGate leverages a learnable gating mechanism that enables individual tasks to select and combine channels from both a specialized and shared feature set. This framework promotes an asymmetric flow of information, where each task is free to decide how much it contributes to – or takes from – the shared branch. By regularizing the learnable gates, we can strike a balance between task-specific resource allocation and overarching computational costs. InterroGate demonstrates state-

of-the-art performance across various architectures and on notable benchmarks such as CelebA, NYUD-v2, and Pascal-Context. The gating mechanism in InterroGate operates over the channel dimension, or the embedding dimension in the case of ViTs, which in its current form does not support structured pruning for attention matrix computations. Future explorations might integrate approaches like patch-token gating to further optimize computational efficiency.

Table 7. Performance across various model capacities using the ResNet-20 and ResNet-50 backbones on the CelebA (*top*) and NYUD-v2 (*bottom*) tasks.

_	MODEL	GENDER ↑	AGE ↑	CLOTHES ↑	OVERALL ↑	FLOPS (M)	MR↓
ORIGINAL	STL MTL INTERROGATE	97.50 97.28 97.60	86.02 86.70 87.44	93.00 92.35 92.40	92.17 92.11 92.48	174 58 59	2.0 2.7 1.3
HALF	STL MTL Interrogate	96.99 97.02 97.33	85.60 86.41 86.75	92.72 92.11 92.05	91.77 91.85 92.05	44.4 14.8 15.5	2.3 2.0 1.7
QUARTER	STL MTL INTERROGATE	96.64 96.46 96.81	85.22 85.46 86.05	92.19 91.59 91.48	91.35 91.17 91.45	11.6 3.9 4.7	2.0 2.3 1.7
_	Money	Criscopa A I	None I	Montered	A (01)	t Francisco	MD
ORIGINAL	MODEL STL MTL INTERROGATE	43.20 43.39 43.63	0.599 0.586 0.577	19.42 21.70 19.66	0 -3.02 +1.16	1149 683 892	2.3 2.3 1.3
HALF	STL MTL Interrogate	39.72 40.20 39.78	0.613 0.610 0.591	20.06 22.78 20.41	0 -3.98 +0.63	415 296 348	2.3 2.0 1.7
QUARTER	STL MTL InterroGate	35.44 35.68 35.71	0.654 0.632 0.624	21.21 24.57 21.75	0 -4.06 +0.94	177 147 164	2.3 2.3 1.3

Limitations. In this work, we primarily focus on enhancing the trade-off between accuracy and efficiency during inference. However, InterroGate comes with a moderate increase in training time. While we observe that InterroGate demonstrates SoTA results on PASCAL-Context with five tasks, further evaluation on MTL setups with larger number of tasks remains to be explored. Furthermore, although both λ_s and τ_t can control the trade-off between performance and computational cost, effectively approximating the desired FLOPs, we still cannot guarantee a specific target FLOP.

6. Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

- Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- Berriel, R., Lathuilière, S., Nabi, M., Klein, T., Oliveira-Santos, T., Sebe, N., and Ricci, E. Budget-aware adapters for multi-domain learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 382–391, 2019.
- Bhattacharjee, D., Zhang, T., Süsstrunk, S., and Salzmann, M. Mult: An end-to-end multitask learning transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12031–12041, 2022.
- Bragman, F. J., Tanno, R., Ourselin, S., Alexander, D. C., and Cardoso, J. Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1385–1394, 2019.
- Brüggemann, D., Kanakis, M., Obukhov, A., Georgoulis, S., and Van Gool, L. Exploring relational context for multitask dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15869–15878, 2021.
- Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 801–818, 2018a.
- Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., and Yuille, A. Detect what you can: Detecting and representing objects using holistic models and body parts. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1971–1978, 2014.
- Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning (ICML)*, pp. 794–803. PMLR, 2018b.

- Chen, Z., Ngiam, J., Huang, Y., Luong, T., Kretzschmar, H., Chai, Y., and Anguelov, D. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33: 2039–2050, 2020.
- Chen, Z., Shen, Y., Ding, M., Chen, Z., Zhao, H., Learned-Miller, E. G., and Gan, C. Mod-squad: Designing mixtures of experts as modular multi-task learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11828–11837, 2023.
- Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88: 303–338, 2010.
- Fan, Z., Sarkar, R., Jiang, Z., Chen, T., Zou, K., Cheng, Y., Hao, C., Wang, Z., et al. M³vit: Mixture-of-experts vision transformer for efficient multi-task learning with modelaccelerator co-design. *Advances in Neural Information Processing Systems*, 35:28441–28457, 2022.
- Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516, 2021.
- Gao, Y., Bai, H., Jie, Z., Ma, J., Jia, K., and Liu, W. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11543–11552, 2020.
- Guo, P., Lee, C.-Y., and Ulbricht, D. Learning to branch for multi-task learning. In *International conference on machine learning*, pp. 3854–3863. PMLR, 2020.
- Hazimeh, H., Zhao, Z., Chowdhery, A., Sathiamoorthy, M., Chen, Y., Mazumder, R., Hong, L., and Chi, E. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems*, 34:29335–29347, 2021.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
- Javaloy, A. and Valera, I. Rotograd: Gradient homogenization in multitask learning. In *International Conference* on Learning Representations (ICLR), 2021.
- Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7482–7491, 2018.

- Lin, B., Feiyang, Y., Zhang, Y., and Tsang, I. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, 2022.
- Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021.
- Liu, S., Johns, E., and Davison, A. J. End-to-end multi-task learning with attention. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1871–1880, 2019.
- Liu, S., James, S., Davison, A., and Johns, E. Auto-lambda: Disentangling dynamic task relationships. *Transactions on Machine Learning Research*, 2022.
- Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. *Proceedings of the IEEE* international Conference on Computer Vision (ICCV), pp. 3730–3738, 2015.
- Ma, J., Zhao, Z., Yi, X., Chen, J., Hong, L., and Chi, E. H. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. *Proceedings of the 24th* ACM SIGKDD international conference on knowledge discovery & data mining, pp. 1930–1939, 2018.
- Mallya, A., Davis, D., and Lazebnik, S. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 67–82, 2018.
- Maninis, K.-K., Radosavovic, I., and Kokkinos, I. Attentive single-tasking of multiple tasks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1851–1860, 2019.
- Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Crossstitch networks for multi-task learning. In *Proceedings* of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3994–4003, 2016.
- Navon, A., Shamsian, A., Achituve, I., Maron, H., Kawaguchi, K., Chechik, G., and Fetaya, E. Multi-task learning as a bargaining game. *International Conference on Machine Learning*, pp. 16428–16446, 2022.
- Rahimian, E., Javadi, G., Tung, F., and Oliveira, G. Dynashare: Task and instance conditioned parameter sharing for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4534–4542, 2023.
- Ranftl, R., Bochkovskiy, A., and Koltun, V. Vision transformers for dense prediction. *Proceedings of the*

- *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12179–12188, 2021.
- Royer, A., Blankevoort, T., and Ehteshami Bejnordi, B. Scalarization for multi-task and multi-domain learning at scale. *Advances in Neural Information Processing Systems*, 2023.
- Sarkar, R., Liang, H., Fan, Z., Wang, Z., and Hao, C. Edgemoe: Memory-efficient multi-task vision transformer architecture with task-level sparsity via mixture-of-experts. arXiv preprint arXiv:2305.18691, 2023.
- Sener, O. and Koltun, V. Multi-task learning as multiobjective optimization. *Advances in Neural Information Processing Systems*, 31, 2018.
- Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. Indoor segmentation and support inference from rgbd images. *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 746–760, 2012.
- Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., and Savarese, S. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pp. 9120–9132. PMLR, 2020.
- Sun, X., Panda, R., Feris, R., and Saenko, K. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33:8728–8740, 2020.
- Vandenhende, S., Georgoulis, S., and Van Gool, L. Mtinet: Multi-scale task interaction networks for multi-task learning. *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 527–543, 2020.
- Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., and Van Gool, L. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3614–3633, 2021.
- Wallingford, M., Li, H., Achille, A., Ravichandran, A., Fowlkes, C., Bhotika, R., and Soatto, S. Task adaptive parameter sharing for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7561–7570, 2022.
- Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine* intelligence, 43(10):3349–3364, 2020.
- Wang, Z., Dai, Z., Póczos, B., and Carbonell, J. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11293–11302, 2019.

- Xu, D., Ouyang, W., Wang, X., and Sebe, N. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 675–684, 2018.
- Yi, R., Guo, L., Wei, S., Zhou, A., Wang, S., and Xu, M. Edgemoe: Fast on-device inference of moe-based large language models. *arXiv preprint arXiv:2308.14352*, 2023.
- Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33: 5824–5836, 2020.
- Zhang, Z., Cui, Z., Xu, C., Yan, Y., Sebe, N., and Yang, J. Pattern-affinitive propagation across depth, surface normal and semantic segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4106–4115, 2019.
- Zhao, X., Li, H., Shen, X., Liang, X., and Wu, Y. A modulation module for multi-task learning with applications in image retrieval. *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 401–416, 2018.

Appendix

A. Implementation Details

A.1. CelebA

On the CelebA dataset, we use ResNet-20 as our backbone with three task-specific linear classifier heads, one for each attribute. We resize the input images to 32x32 and remove the initial pooling in the stem of ResNet to accommodate the small image resolution. For training, we use the Adam optimizer with a learning rate of 1e-3, weight decay of 1e-4, and a batch size of 128. For learning rate decay, we use a step learning rate scheduler with step size 20 and a multiplicative factor of 1/3. We use SGD with a learning rate of 0.1 for the gates' parameters.

A.2. NYUD and PASCAL-Context

For both NYUD-v2 and PASCAL-Context with ResNet-18 and ResNet-50 backbones, we use the Atrous Spatial Pyramid Pooling (ASPP) module introduced by (Chen et al., 2018a) as task-specific heads. For the HRNet-18 backbone, we follow the methodology of the original paper (Wang et al., 2020): HRNet combines the output representations at four different resolutions and fuses them using 1x1 convolutions to output dense prediction.

We train all convolution-based encoders on the NYUD-v2 dataset for 100 epochs with a batch size of 4 and on the PASCAL-Context dataset for 60 epochs with a batch size of 8. We use the Adam optimizer, with a learning rate of 1e-4 and weight decay of 1e-4. We use the same data augmentation strategies for both NYUD-v2 and PASCAL-Context datasets as described in (Vandenhende et al., 2020). We use SGD with a learning rate of 0.1 to learn the gates' parameters.

In terms of task objectives, we use the cross-entropy loss for semantic segmentation and human parts, L_1 loss for depth and normals, and binary-cross entropy loss for edge and saliency detection tasks, similar to (Vandenhende et al., 2020). For learning rate decay, we adopt a polynomial learning rate decay scheme with a power of 0.9.

The Choice of ω_t . The hyper-parameter ω_t denotes the scalarization weights. We use the weights suggested in prior work but also report numbers of uniform scalarization. For NYUD-v2, we use uniform scalarization as suggested in (Maninis et al., 2019; Vandenhende et al., 2021), and for PASCAL-Context, we similarly use the weights suggested in (Maninis et al., 2019) and (Vandenhende et al., 2021).

A.3. DPT Training

For DPT training, we follow the same training procedure as described by the authors, which employs the Adam optimizer, with a learning rate of 1e-5 for the encoder and 1e-4 for the task heads, and a batch size of 8.

The ViT backbones were pre-trained on ImageNet-21k at resolution 224×224, and fine-tuned on ImageNet 2012 at resolution 384×384. The feature dimension for DPT's task heads was reduced from 256 to 64. We conducted a sweep over a set of weight decay values and chose 1e-6 as the optimal value for our DPT experiments.

B. Forward-pass Pseudo-code

Algorithm 1 illustrates the steps in the forward pass of the algorithm.

C. Generalization to Vision Transformers

As transformers are becoming widely used in the vision literature, and to show the generality of our proposed MTL framework, we also apply InterroGate to vision transformers: We again denote $\varphi_t^\ell \in \mathbb{R}^{N^\ell \times C^\ell}$ and $\psi^\ell \in \mathbb{R}^{N^\ell \times C^\ell}$ as the t-th task-specific and shared representations in layer ℓ , where N^ℓ and C^ℓ are the number of tokens and embedding dimensions, respectively. We first apply our feature selection to the key, query and value linear projections in each self-attention block:

$$q_t^{l+1} = G_t^\ell(\alpha_{q,t}^\ell) \odot f_{q,t}^\ell(\varphi_t^\ell; \Phi_t^\ell) + (1 - G_t^\ell(\alpha_{q,t}^\ell)) \odot f_q^\ell(\psi^\ell; \Psi^\ell), \tag{7}$$

$$k_t^{l+1} = G_t^{\ell}(\alpha_{k,t}^{\ell}) \odot f_{k,t}^{\ell}(\varphi_t^{\ell}; \Phi_t^{\ell}) + (1 - G_t^{\ell}(\alpha_{k,t}^{\ell})) \odot f_k^{\ell}(\psi^{\ell}; \Psi^{\ell}), \tag{8}$$

$$v_t^{l+1} = G_t^{\ell}(\alpha_{v,t}^{\ell}) \odot f_{v,t}^{\ell}(\varphi_t^{\ell}; \Phi_t^{\ell}) + (1 - G_t^{\ell}(\alpha_{v,t}^{\ell})) \odot f_v^{\ell}(\psi^{\ell}; \Psi^{\ell}), \tag{9}$$

Algorithm 1 Pseudo-code for unified representation encoder

Given: • $x \in \mathbb{R}^{3 \times W \times H}$ Input image • $T, L \in \mathbb{R}$ Number of tasks and encoder layers • Ψ , Φ_t Shared and t-th task-specific layer parameters • β , α_t Shared and t-th task-specific gating parameters $\begin{array}{ll} \textbf{Return:} \ [\varphi_1^L,...,\varphi_T^L] \\ \psi^0,\varphi_1^0,...,\varphi_T^0 \leftarrow x \end{array}$ Task-specific encoder representations Set initial shared and task-specific features for $\ell = 1$ to L do for t = 1 to T do $\begin{array}{l} \varphi_t'^\ell \leftarrow G_t^\ell(\alpha_t^\ell) \odot \varphi_t^\ell + (1 - G_t^\ell(\alpha_t^\ell)) \odot \psi^\ell \text{ (2)} \\ \varphi_t^{\ell+1} \leftarrow F(\varphi_t'^\ell; \Phi_t^\ell) \end{array}$ Choose shared and task-specific features Compute task-specific features end for $\psi^{\prime\ell} = \sum_{t=1}^{T} \underset{t=1...T}{\operatorname{softmax}} (\beta_t^{\ell}) \odot \varphi_t^{\prime\ell} \ (3)$ $\psi^{\ell+1} \leftarrow F(\psi^{\prime\ell}; \Psi^{\ell})$ Combine task-specific features to form shared ones Compute shared features end for

where $\alpha_{q,t}^{\ell}$, $\alpha_{k,t}^{\ell}$, $\alpha_{v,t}^{\ell}$ are the learnable gating parameters mixing the task-specific and shared projections for queries, keys and values, respectively. $f_{q,t}^{l}$, $f_{k,t}^{l}$, $f_{v,t}^{l}$ are the linear projections for query, key and value for the task t, while f_{q}^{l} , f_{k}^{l} , f_{v}^{l} are the corresponding shared projections. Once the task-specific representations are formed, the shared embeddings for the next block are computed by a learned mixing of the task-specific feature followed by a linear projection, as described in (3). Similarly, we apply this gating mechanism to the final linear projection of the multi-head self-attention, as well as the linear layers in the feed-forward networks in-between each self-attention block.

D. Additional Experiments

D.1. Full Results on CelebA and NYUD-v2

In Table 8, we report results on the CelebA dataset for different model capacities: Here, InterroGate is compared to the STL and standard MTL methods with different model width: at original, half and quarter of the original model width. In Tables 9 and 10, we report the complete results on NYUD-v2 dataset using HRNet-18 and ResNet-50 backbones, including the parameter count and standard deviation of the $\Delta_{\rm MTL}$ scores.

D.2. Sharing/Specialization Patterns

Table 8. Performance comparison of various MTL models on the CelebA dataset with different model capacities. Different InterroGate models are obtained by varying λ_s

	Model	Gender ↑	Age ↑	Clothes ↑	Overall ↑	Flops (M)	MR↓
nal	STL	97.50	86.02	93.00	92.17	174	3.3
Original	MTL	97.28	86.70	92.35	92.11	58	4.7
Ö	InterroGate	97.60	87.44	92.40	92.48	59	2.7
	InterroGate	97.77	87.39	92.56	92.57	11	2.3
	InterroGate	97.95	87.24	92.85	92.68	162	2.0
J	STL	96.99	85.60	92.72	91.77	44.4	3.7
Half	MTL	97.02	86.41	92.11	91.85	14.8	4.0
_	InterroGate	97.33	86.75	92.05	92.05	15.5	3.7
	InterroGate	97.33	87.05	92.12	92.17	17.4	2.3
	InterroGate	97.46	86.97	92.47	92.23	24.5	1.7
Eer	STL	96.64	85.22	92.19	91.35	11.6	3.3
Quarter	MTL	96.46	85.46	91.59	91.17	3.9	4.3
Ō	InterroGate	96.81	86.05	91.48	91.45	4.7	3.7
	InterroGate	96.92	86.10	91.64	91.56	5.5	2.0
	InterroGate	96.81	86.61	91.74	91.72	6.4	2.0

Table 9. Performance comparison on NYUD-v2 using HRNet-18 backbone. Different InterroGate models are obtained by varying λ_s .

MODEL	Semseg ↑	ДЕРТН ↓	Normals ↓	$\Delta_{\mathrm{MTL}} \ (\%) \uparrow$	FLOPS (G)	Param (M)	MR↓
STL	41.70	0.582	18.89	0 ± 0.12	65.1	28.9	8.0
MTL (UNI.)	41.83	0.582	22.84	-6.86 ± 0.76	24.5	9.8	11.0
DWA	41.86	0.580	22.61	-6.29 ± 0.95	24.5	9.8	8.7
UNCERTAINTY	41.49	0.575	22.27	-5.73 ± 0.35	24.5	9.8	8.3
Auto- λ	42.71	0.577	22.87	-5.92 ± 0.47	24.5	9.8	8.0
RLW	42.10	0.593	23.29	$\textbf{-8.09} \pm 1.11$	24.5	9.8	11.7
PCGRAD	41.75	0.581	22.73	-6.70 ± 0.99	24.5	9.8	10.3
CAGRAD	42.31	0.580	22.79	-6.28 ± 0.90	24.5	9.8	8.7
MGDA-UB	41.23	0.625	21.07	$\textbf{-6.68} \pm 0.67$	24.5	9.8	11.3
InterroGate	43.58	0.559	19.32	+2.06 ± 0.13	43.2	18.8	1.3
INTERROGATE	42.95	0.562	19.73	$+0.68 \pm 0.09$	38.3	16.5	2.3
INTERROGATE	42.36	0.564	20.04	-0.55 ± 0.17	36.0	15.4	4.0
INTERROGATE	42.73	0.575	21.01	-2.55 ± 0.11	33.1	13.7	4.0
INTERROGATE	42.35	0.575	21.70	-4.07 ± 0.38	29.2	11.9	5.7

Table 10. Performance comparison on NYUD-v2 using ResNet-50 backbone. Different InterroGate models are obtained by varying λ_s .

MODEL	Semseg ↑	ДЕРТН ↓	Normals ↓	Δ_{MTL} (%) \uparrow	FLOPS (G)	PARAM (M)	MR↓
STL	43.20	0.599	19.42	0 ± 0.11	1149	118.9	9.0
MTL (UNI.)	43.39	0.586	21.70	-3.04 ± 0.79	683	71.9	9.7
DWA	43.60	0.593	21.64	-3.16 ± 0.39	683	71.9	9.7
UNCERTAINTY	43.47	0.594	21.42	-2.95 ± 0.40	683	71.9	10.0
Auto- λ	43.57	0.588	21.75	-3.10 ± 0.39	683	71.9	10.0
RLW	43.49	0.587	21.54	-2.74 ± 0.09	683	71.9	8.3
PCGRAD	43.74	0.588	21.55	-2.66 ± 0.15	683	71.9	7.3
CAGRAD	43.57	0.583	21.55	-2.49 ± 0.11	683	71.9	7.0
MGDA-UB	42.56	0.586	21.76	-3.83 ± 0.17	683	71.9	11.3
MTAN	44.92	0.585	21.14	-0.84 ± 0.32	683	92.4	4.0
CROSS-STITCH	44.19	0.577	19.62	$+1.66 \pm 0.09$	1151	119.0	2.7
INTERROGATE	44.38	0.576	19.50	+2.04 ± 0.07	916	95.4	1.7
INTERROGATE	43.63	0.577	19.66	$+1.16 \pm 0.10$	892	92.4	3.7
INTERROGATE	43.05	0.589	19.95	$\textbf{-0.50} \pm 0.05$	794	83.3	9.7

Figure 4 illustrates the distribution of the gating patterns across all layers of the ResNet-18 backbone for the PASCAL-Context dataset for 3 models using (a) a Hinge loss, (b) a medium-level pruning using uniform L_1 loss and (c) a high-level pruning with uniform L_1 loss.

E. Ablation: Sparsity Targets

By tuning the sparsity targets τ in Equation 4, we can achieve specific compute budgets of the final network at inference. However, there are multiple choices of $\{\tau_t\}_{t=1}^T$ that can achieve the same budget. In this section, we further investigate the impact of which task we allocate more or less of the compute budget on the final accuracy/efficiency trade-off.

We perform an experiment sweep for different combination of sparsity targets, where each τ_t is chosen from $\{0, 0.25, 0.75, 1.0\}$. Each experiment is run for two different random seeds and two different sparsity loss weights λ_s . Due to the large number of experiments, we perform the ablation experiments for shorter training runs (75% of the training epochs for each setup)

Our take-away conclusions are that (i) we clearly observe that some tasks require more task-specific parameters (hence a higher sparsity target) and (ii) this dichotomy often correlates with the per-task performance gap observed between the STL and MTL baselines, which can thus be used as a guide to set the hyperparameter values for τ .

In the results of NYUD-v2 in Figure 5, we observe a clear hierarchy in terms of task importance: When looking at the points on the Pareto curve, they prefer high values of τ_{normals} , followed by $\tau_{\text{segmentation}}$: In other words, these two tasks, and in particular normals prediction, requires more task-specific parameters than the depth prediction task to obtain the best MTL performance versus compute cost trade-offs.

Then, we conduct a similar analysis for the five tasks of PASCAL-Context in Figure 6. Here we see a clear split in tasks: The graph for the edges prediction and saliency task are very similar to one another and tend to prefer high τ values, i.e. more task-specific parameters, at higher compute budget. But when focusing on a lower compute budget, it is more beneficial to the overall objective for these tasks to use the shared branch. Similarly, the tasks of segmentation and human parts exhibit similar behavior under variations of τ and are more robust to using shared representations (lower values of τ). Finally, the

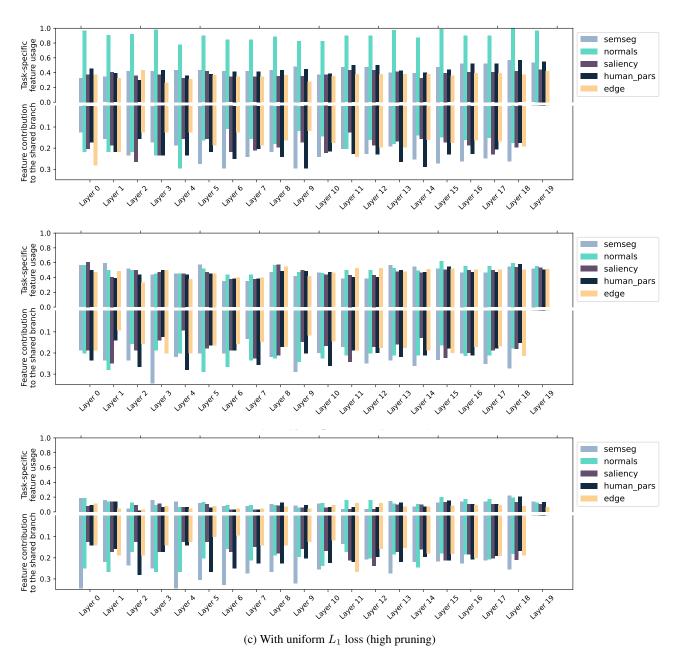


Figure 4. Sharing and specialization patterns on pascal context dataset with ResNet-18 backbone.

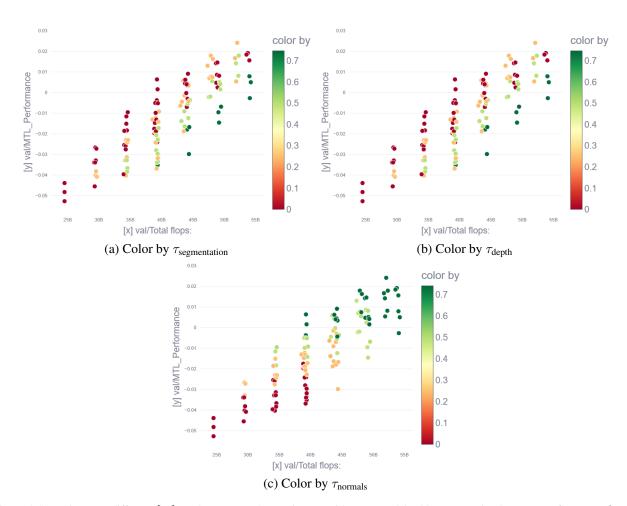


Figure 5. Sweeping over different $\{\tau_t\}$ on the NYUD-v2 experiments with HRNet-18 backbone. We plot the MTL performance Δ_{MTL} against the total number of FLOPs, then color each scatter point by the value of τ_t when the task t is (a) segmentation, (b) depth and (c) normals.

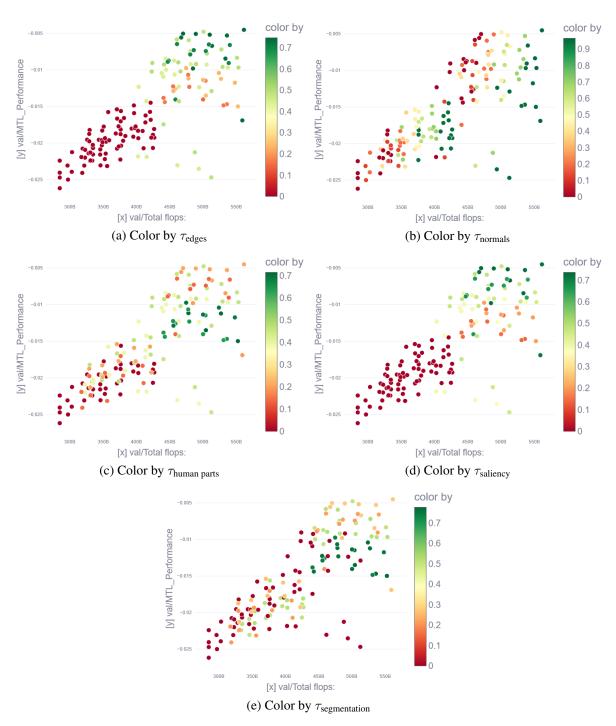


Figure 6. Sweeping over different $\{\tau_t\}$ on the PASCAL-Context. We plot the MTL performance Δ_{MTL} against the total number of FLOPs, then color each scatter point by the value of τ_t when the task t is (a) edges, (b) normals (c) human parts, (d) saliency and (e) segmentation.

task of normals prediction (b) differ from the other four, and in particular exhibit a variance of behavior across different compute budget. In particular, when targetting the intermediate range (350B-450B FLOPs), setting higher τ_{normals} helps the overall objective.

F. Training Time Comparisons

While our method is mainly aiming at improving the inference cost efficiency, we also measure and compare training times between our method and the baselines, on the PASCAL-Context (Chen et al., 2014). The results are shown on Table 11. The forward and backward iterations are averaged over 1000 iterations, after 10 warmup iterations, on a single NvidiaV100 GPU, with a batch size of 4.

Table 11. Training time comparison of various MTL methods

Method	Forward (ms)	Backward (ms)	Training time	(h) Δ_{MTL}
Standard MTL	60	299	7.5	-4.14
MTAN	73	330	8.5	-1.78
Cross-stitch	132	454	12.3	+0.14
MGDA-UB	60	568	13.2	-1.94
CAGrad	60	473	11.1	-2.03
PCGrad	60	495	11.6	-2.58
InterroGate	76	324	8.4	-1.35
InterroGate	102	376	10.1	+0.12
InterroGate	119	426	11.5	+0.42