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

Babak Ehteshami Bejnordi[1]
behtesha@qti.qualcomm.com
Gaurav Kumar[2]
gakum@qti.qualcomm.com
Amelie Royer[1 †]
aroyer@qti.qualcomm.com
Christos Louizos[1]
clouizos@qti.qualcomm.com
Tijmen Blankevoort[1 †]
tijmen@qti.qualcomm.com
Mohsen Ghafoorian[3]
mghafoor@qti.qualcomm.com

[1] Qualcomm AI Research[*]
Science Park 301,
Amsterdam, Netherlands
[2] Qualcomm AI Software
KRC Tower,
Hyderabad, India
[3] Qualcomm XR Labs
Science Park 301,
Amsterdam, Netherlands

Abstract

Jointly learning multiple tasks with a unified model can improve accuracy and data efficiency, but 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 task-specific 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 enhancing computational efficiency during inference. InterroGate features a learnable gating mechanism to automatically balance the shared and task-specific representations while preserving the performance of all tasks. 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.

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 potential

[*]Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc. [†]Work was completed while an employee at Qualcomm AI Research.

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 devices, 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* [44], 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 [30, 39], 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. 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 [13, 28, 35, 38]. 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 [9, 10, 14, 23]. However, the input-dependent routing of MoE is typically hard to efficiently deploy at inference, particularly with batched execution [31, 41].

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 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 [26, 36].

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 transformer-based backbones. We then further investigate the proposed framework through ablation experiments.

2 Related Work

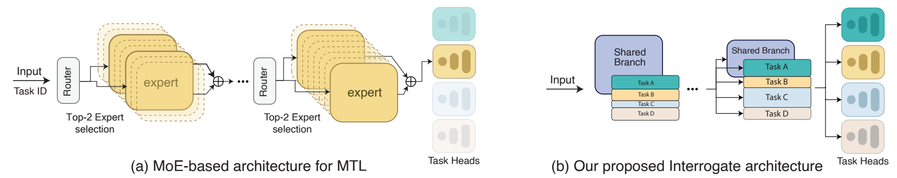
Multi-task Optimization (MTO) methods aim to automatically balance the different tasks when optimizing shared parameters to maximize average performance. Loss-based methods [17, 21] are usually scalable and adaptively scale task losses based on certain statistics (e.g. task output variance); Gradient-based methods [7, 8, 16, 19, 32] 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 [11, 34] identify "task affinities" as a guide to isolate parameters of tasks most likely to interfere with one another. Similarly, [13] 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 [26], MTAN [20] or MuIT [3] 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, [12, 26] require a full network per task and improve MTL performance through lightweight adapters across task branches. [3, 20] 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.

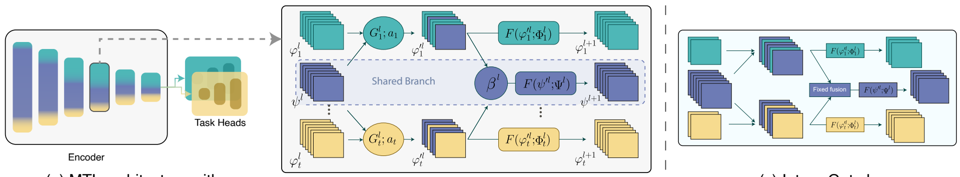
Conditional Compute approaches learn task-specific gating of model parameters: For instance [35, 38] learn to select a subset of the most relevant layers when adapting the network to a new downstream task. Piggyback [24] adapts a pretrained network to multiple tasks by learning a set of per-task sparse masks. Similarly, [2] selects the most relevant feature channels using learnable gates. Finally, **Mixture of Experts (MoE)** [9, 10, 14, 23] 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



(a) MoE-based architecture for MTL (b) Our proposed Interrogate architecture

Figure 1: (a) MoE-based architectures for multi-task learning [9, 10, 14, 23] rely on a router which selects Top-k experts to execute based on the current task and representation. At inference, MoEs can solve one task at a time, requiring T multiple forward passes to solve T tasks. (b) Our proposed architecture, learns the per-layer optimal balance between sharing and specializing representations for each task, and can solve all tasks in one forward pass.



(a) MTL architecture with InterroGate Layers (b) An InterroGate layer (c) InterroGate layer at inference

Figure 2: **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 $\varphi_t'^\ell$ 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.

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 [31, 41]. In contrast, InterroGate focuses on optimizing efficiency and is capable of addressing all tasks simultaneously (See Figure 1), 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 [4], 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 InterroGate 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:

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

where X and Y_t are the input data and corresponding labels for task t, $\mathcal{L}_t$ represents the loss 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, 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 transformer-based models in Appendix C.

3.1 Learning to Share, Specialize and Prune

Figure 2 presents an overview of the proposed InterroGate framework. Formally, let $\psi^\ell \in R^{C^\ell \times W^\ell \times H^\ell}$ and $\varphi_t^\ell \in R^{C^\ell \times W^\ell \times H^\ell}$ represent the shared and task-specific features at layer ℓ of our multi-task network, respectively. In each layer ℓ, the gating module G_t^ℓ of task t selects relevant channels from either ψ^ℓ and φ_t^ℓ. The output of this hard routing operation yields features $\varphi_t'^\ell$:

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

where $\odot$ is the Hadamard product and $\alpha_t^\ell \in R^{C^\ell}$ 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. In particular, we let the model learn its own soft combination weights and the resulting mixing operation for the shared features is defined as follows:

$$\psi'^\ell = \sum_{t=1}^{T} \underset{t=1\ldots T}{\mathrm{softmax}}(\beta_t^\ell) \odot \varphi_t'^\ell, \tag{3}$$

where $\beta^\ell \in R^{C^\ell \times T}$ 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 $\psi^{\ell+1} = F(\psi'^\ell; \Psi^\ell)$. 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 through 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 [1]: 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 D.

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:

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

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 multi-task objective and sparsity regularizer:

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

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 [22], NYUD-v2 [33], and PASCAL-Context [6]. Detailed description of the datasets is presented in Appendix A. We consider the age, gender, and clothes attributes in CelebA to form three output classification tasks for our MTL setup and use ResNet-20 [13] as the backbone. For NYUD-v2, following recent work [25, 40, 43], we also incorporate the surface normal prediction task, obtaining annotations directly from the depth ground truth. We use ResNet-50 with dilated convolutions and HRNet-18 following [37]. We also present results using a dense prediction transformer (DPT) [29], with a ViT-base and -small backbone. Finally, on PASCAL-Context, we use a ResNet-18 backbone. We further describe implementation details and training hyper-parameters, in Appendix B.

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 [26] and MTAN [20], as well as MTO approaches such as uncertainty weighting [17], DWA [20], and Auto-λ [21], PCGrad [42], CAGrad [19], MGDA-UB [32], and RLW [18]. Following [25], our main metric is the multi-task performance Δ_{MTL} of a model m as the averaged normalized drop in performance w.r.t. the single-task baselines b:

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

where $l_i = 1$ if a lower value means better performance for metric M_i of task i, and 0 otherwise. Furthermore, similar to [27], 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 3 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 F.3. 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 [6] dataset in Appendix E.

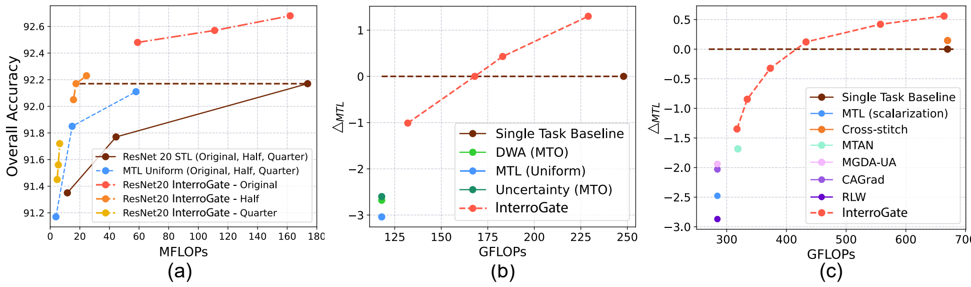


Figure 3: Accuracy vs. floating-point operations (FLOP) trade-off curves for InterroGate and SoTA MTL methods. (a) Results on CelebA using ResNet-20 backbone at three different widths (Original, Half, and Quarter). (b) NYUD-v2 using DPT with ViT-small backbone, and (c) ResNet-18 on PASCAL-Context.

4.2 Results

CelebA. Figure 3a 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 Table 9, Appendix F.1. 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 section 4.3.3.

NYUD-v2. Table 1 & 2 present the results on the NYUD-v2 dataset, using the HRNet-18 and ResNet-50 backbones, respectively. We additionally report the **parameter count** and the standard deviation of the Δ_{MTL} scores in Table 10 & 11, in Appendix F.1. 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 [19] and uncertainty weighting [17] 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 Δ_{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 ResNet-50, exhibiting a Δ_{MTL} score of +1.16, in contrast to MTAN's -0.84 (Table 11 in Appendix F.1).

Table 3 & 4 report the performance of DPT trained models with the ViT-base and ViT-small backbones, and Figure 3b illustrates the trade-off between Δ_{MTL} and computational costs of various methods using the ViT-small backbone. 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 1: Results on NYUD-v2 with HRNet-18.

Model	Semseg ↑	Depth ↓	Normals ↓	Δ_{MTL} (%) ↑	Flops (G)	MR↓
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: Results on NYUD-v2 with ResNet-50.

Model	Semseg ↑	Depth ↓	Normals ↓	Δ_{MTL} (%) ↑	Flops (G)	MR↓
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

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 [25] for all MTL methods, but also report MTL uniform results as a reference. Figure 3c 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 [6, 25, 36], 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 trade-off, however, both suffer from a notable decline in normals prediction performance.

At its highest compute budget (no sparsity loss and negligible computational savings), In-

Table 3: Results on NYUD-v2 using ViT-base.

Model	Semseg ↑	Depth ↓	Normals ↓	Δ_{MTL} (%) ↑	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 ViT-small.

Model	Semseg ↑	Depth ↓	Normals ↓	Δ_{MTL} (%) ↑	Flops (G)	MR↓
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	118	5.7
InterroGate	45.96	**0.5648**	**20.77**	**+1.30**	229	**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

terroGate outperforms the STL baseline, notably in Saliency and Human parts prediction tasks, and achieves an overall Δ_{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. Our InterroGate models consistently 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 ↑	Edge ↓	Δ_{MTL} (%) ↑	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 ↑	Human ↑	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	63.9	14.74	0.664	0.600	0.0172	-0.27	623	2.8
InterroGate	L_1	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	-0.13	433	3.4

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

	Model	Gender ↑	Age ↑	Clothes ↑	Overall ↑	Flops (M)	MR↓
Original	STL	97.50	86.02	**93.00**	92.17	174	2.0
	MTL	97.28	86.70	92.35	92.11	58	2.7
	InterroGate	**97.60**	**87.44**	92.40	**92.48**	59	**1.3**
Half	STL	96.99	85.60	**92.72**	91.77	44.4	2.3
	MTL	97.02	86.41	92.11	91.85	14.8	2.0
	InterroGate	**97.33**	**86.75**	92.05	**92.05**	15.5	**1.7**
Quarter	STL	96.64	85.22	**92.19**	91.35	11.6	2.0
	MTL	96.46	85.46	91.59	91.17	3.9	2.3
	InterroGate	**96.81**	**86.05**	91.48	**91.45**	4.7	**1.7**

	Model	Semseg ↑	Depth ↓	Normals ↓	Δ_{MTL} (%) ↑	Flops (G)	MR↓
Original	STL	43.20	0.599	**19.42**	0	1149	2.3
	MTL	43.39	0.586	21.70	-3.02	683	2.3
	InterroGate	**43.63**	**0.577**	19.66	**+1.16**	892	**1.3**
Half	STL	39.72	0.613	**20.06**	0	415	2.3
	MTL	40.20	0.610	22.78	-3.98	296	2.0
	InterroGate	39.78	**0.591**	20.41	**+0.63**	348	**1.7**
Quarter	STL	35.44	0.654	**21.21**	0	177	2.3
	MTL	35.68	0.632	24.57	-4.06	147	2.3
	InterroGate	**35.71**	**0.624**	21.75	**+0.94**	164	**1.3**

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. Next, 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

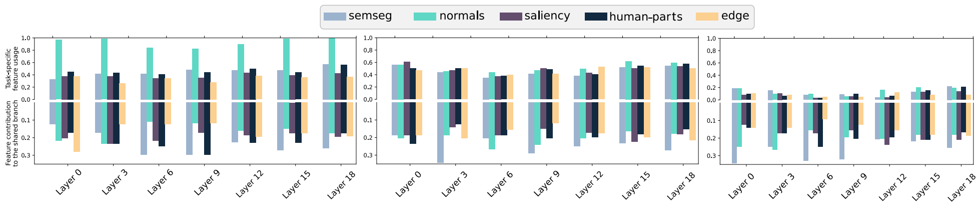


Figure 4: The task-specific representation selection ratio (top) versus proportions of maximum contributions to the shared branch (bottom) for InterroGate with hinge loss (left), L_1 loss with medium pruning (middle) and L_1 loss with high pruning (right).

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 4, for different sparsity regularizers: with hinge loss (*left*), with L_1 loss at medium (middle) and high pruning levels (*right*). In all settings, the semantic segmentation task makes the largest contribution to the shared branch, followed by the normals prediction task. Interestingly, we show that when the normals prediction task acquires sufficient task-specific features, it maintains a high accuracy. We provide more detailed discussion on the learned sharing and specialization patterns in Appendix F.2.

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 MTL framework to address the fundamental challenges of task interference and computational constraints during inference. InterroGate leverages a learnable gating mechanism that enables individual tasks to select and combine channels from both a specialized and shared feature set. 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 work might integrate approaches like token gating to further optimize computational efficiency.

Limitations. While our method is more effective in resolving task-interference via dedicated task-specific parameters, it comes with an increase in the total number of parameters compared to MTO approaches. 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.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[2] Rodrigo Berriel, Stéphane Lathuilière, Moin Nabi, Tassilo Klein, Thiago Oliveira-Santos, N. Sebe, and Elisa Ricci. Budget-aware adapters for multi-domain learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 382–391, 2019.

[3] Deblina Bhattacharjee, Tong Zhang, Sabine Süsstrunk, and Mathieu Salzmann. Mult: An end-to-end multitask learning transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12031–12041, 2022.

[4] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1385–1394, 2019.

[5] David Brüggemann, Menelaos Kanakis, Anton Obukhov, Stamatios Georgoulis, and Luc Van Gool. Exploring relational context for multi-task dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15869–15878, 2021.

[6] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. 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)*, pages 1971–1978, 2014.

[7] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning (ICML)*, pages 794–803. PMLR, 2018.

[8] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33: 2039–2050, 2020.

[9] Zitian Chen, Yikang Shen, Mingyu Ding, Zhenfang Chen, Hengshuang Zhao, Erik G Learned-Miller, and Chuang Gan. 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)*, pages 11828–11837, 2023.

[10] Zhiwen Fan, Rishov Sarkar, Ziyu Jiang, Tianlong Chen, Kai Zou, Yu Cheng, Cong Hao, Zhangyang Wang, et al. M^{3}vit: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design. *Advances in Neural Information Processing Systems*, 35:28441–28457, 2022.

[11] Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516, 2021.

[12] Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia, and Wei Liu. 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)*, pages 11543–11552, 2020.

[13] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In *International conference on machine learning*, pages 3854–3863. PMLR, 2020.

[14] Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. 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.

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

[16] Adrián Javaloy and Isabel Valera. Rotograd: Gradient homogenization in multitask learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[17] Alex Kendall, Yarin Gal, and Roberto Cipolla. 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)*, pages 7482–7491, 2018.

[18] Baijiong Lin, YE Feiyang, Yu Zhang, and Ivor Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, 2022.

[19] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021.

[20] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1871–1880, 2019.

[21] Shikun Liu, Stephen James, Andrew Davison, and Edward Johns. Auto-lambda: Disentangling dynamic task relationships. *Transactions on Machine Learning Research*, 2022.

[22] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *Proceedings of the IEEE international Conference on Computer Vision (ICCV)*, pages 3730–3738, 2015.

[23] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 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*, pages 1930–1939, 2018.

[24] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[25] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1851–1860, 2019.

[26] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003, 2016.

[27] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. *International Conference on Machine Learning*, pages 16428–16446, 2022.

[28] Elahe Rahimian, Golara Javadi, Frederick Tung, and Gabriel Oliveira. Dynashare: Task and instance conditioned parameter sharing for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4534–4542, 2023.

[29] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12179–12188, 2021.

[30] Amelie Royer, Tijmen Blankevoort, and Babak Ehteshami Bejnordi. Scalarization for multi-task and multi-domain learning at scale. *Advances in Neural Information Processing Systems*, 2023.

[31] Rishov Sarkar, Hanxue Liang, Zhiwen Fan, Zhangyang Wang, and Cong Hao. Edge-moe: Memory-efficient multi-task vision transformer architecture with task-level sparsity via mixture-of-experts. *arXiv preprint arXiv:2305.18691*, 2023.

[32] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

[33] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 746–760, 2012.

[34] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020.

[35] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33:8728–8740, 2020.

[36] Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction networks for multi-task learning. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 527–543, 2020.

[37] 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, 2021.

[38] Matthew Wallingford, Hao Li, Alessandro Achille, Avinash Ravichandran, Charless Fowlkes, Rahul Bhotika, and Stefano Soatto. Task adaptive parameter sharing for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7561–7570, 2022.

[39] Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11293–11302, 2019.

[40] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. 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)*, pages 675–684, 2018.

[41] Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. Edgemoe: Fast on-device inference of moe-based large language models. *arXiv preprint arXiv:2308.14352*, 2023.

[42] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

[43] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Yan Yan, Nicu Sebe, and Jian Yang. Pattern-affinitive propagation across depth, surface normal and semantic segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4106–4115, 2019.

[44] Xiangyun Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. A modulation module for multi-task learning with applications in image retrieval. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 401–416, 2018.