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

Qualcomm AI research

BMVC 2024

Babak Ehteshami Bejnordi
Gaurav Kumar
Amélie Royer
Christos Louizos
Tijmen Blankevoort
Mohsen Ghafoorian

Qualcomm Technologies Netherlands, B.V.

{behtesha, gakum, aroyer, clouizos, tijmen, mghafoor}@qti.qualcomm.com

Multi-task learning (MTL): Concerns jointly learning multiple tasks with a unified network, providing:

- Improved accuracy
- Data efficiency
- Reduced computational and memory costs

Application: Crucial for many real-life applications (e.g. XR, self-driving cars, mobile phones, etc.)

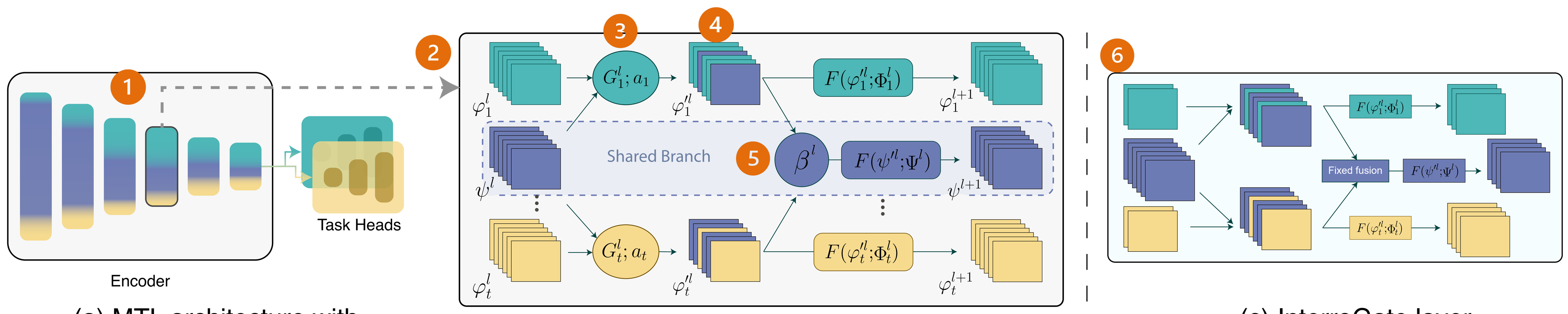
Challenge: Optimizing one task objective may inadvertently compromise the performance of another: This is known as ***task interference***.

Our goal is to design an architecture that carefully allocates shared and task-specific parameters to reduce interference while considering the computational budget.

Solution: We propose a novel MTL framework, ***InterroGate***, to address the fundamental challenges of task interference and computational constraints in MTL.

- InterroGate learns the *optimal balance between shared and specialized representations*
- By leveraging a set of learnable Gates, InterroGate controls the balance between accuracy and inference compute cost
- InterroGate achieves SoTA results on three multi-tasking benchmarks: CelebA, NYUD-v2, and PASCAL-Context

Overview of the proposed InterroGate framework



(a) MTL architecture with InterroGate Layers

(b) An InterroGate layer

(c) InterroGate layer at inference

1. InterroGate layer replaces the original encoder layer
2. Each layer receives $t+1$ feature maps, one shared and t task-specific representations
3. Task-specific gating modules, G_t^l decides between shared ψ^l or task-specific φ_t^l features
4. Selected features are mixed, $\varphi_t'^l$ and passed to the next task-specific layer
5. Shared input is a linear combination of the task-specific features via the learned parameter β_t^l
6. During inference, unselected feature parameters are pruned, simplifying the model to a plain architecture

Algorithm 1: Pseudo-code for unified representation encoder

```
Given:
  • x ∈ R^{3×W×H}          // Input image
  • T, L ∈ 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
Return: [φ_1^L, ..., φ_T^L]     // Task-specific encoder representations
ψ^0, φ_1^0, ..., φ_T^0 ← x      // Set initial shared and task-specific features
for ℓ = 1 to L do
  for t = 1 to T do
    φ_t'^ℓ ← G_t^ℓ(α_t^ℓ) ⊙ φ_t^ℓ + (1 − G_t^ℓ(α_t^ℓ)) ⊙ ψ^ℓ  (Equation 2)
    // Choose shared and task-specific features
    φ_t^{ℓ+1} ← F(φ_t'^ℓ; Φ_t^ℓ)    // Compute task-specific features
  end for
  ψ'^ℓ = Σ_{t=1}^T softmax_{t=1...T}(β_t^ℓ) ⊙ φ_t'^ℓ  (Equation 3)
  // Combine task-specific features to form shared ones
  ψ^{ℓ+1} ← F(ψ'^ℓ; Ψ^ℓ)           // Compute shared features
end for
```

How to train:

The model and gate parameters are trained end-to-end by minimizing the classical MTL objective:

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

Sparsity regularization:

The gating module G is regularized using a hinge loss, controlled by task-specific hinge target τ_t

❖A lower τ_t promotes more feature sharing, while a higher τ_t allows greater task-specific selection at the cost of increased computation

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

Final Objective:

- The overall training objective is a combination of multi-task loss and the sparsity regularizer, balanced by a hyperparameter λ_s

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

Evaluation metric:

- Δ_{MTL} is the averaged normalized drop in performance w.r.t. the single-task baselines.

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

Ablation on model capacity:

- Shrinking the ResNet-50 model size increasingly harms multi-task performance
- InterroGate consistently finds a favorable trade-off between capacity and performance, enhancing multi-task performance across all model sizes

	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**

Results:

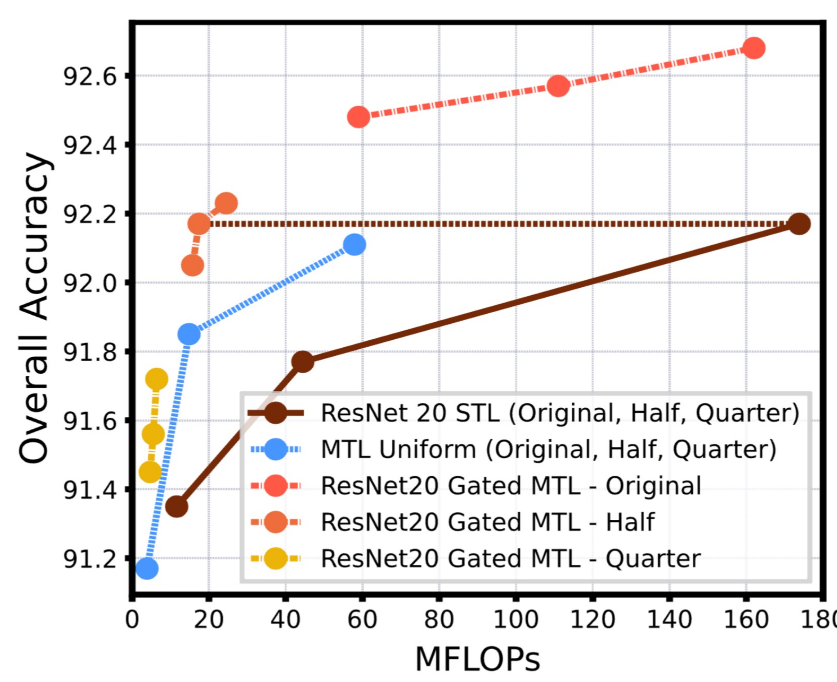
- InterroGate outperforms Cross-stitch and MTAN in ΔMTL scores and computational efficiency, achieving +2.04 compared to Cross-stitch's +1.66 (which comes at a substantial computational cost) and MTAN's -0.84 at equal parameter counts
- On the PASCAL-Context, while most MTL and MTO baselines fall short of STL performance, InterroGate, at its highest compute budget, surpasses the STL baseline, especially in Saliency and Human parts prediction tasks, achieving an overall ΔMTL of +0.56

NYUD-v2 (ResNet-50)

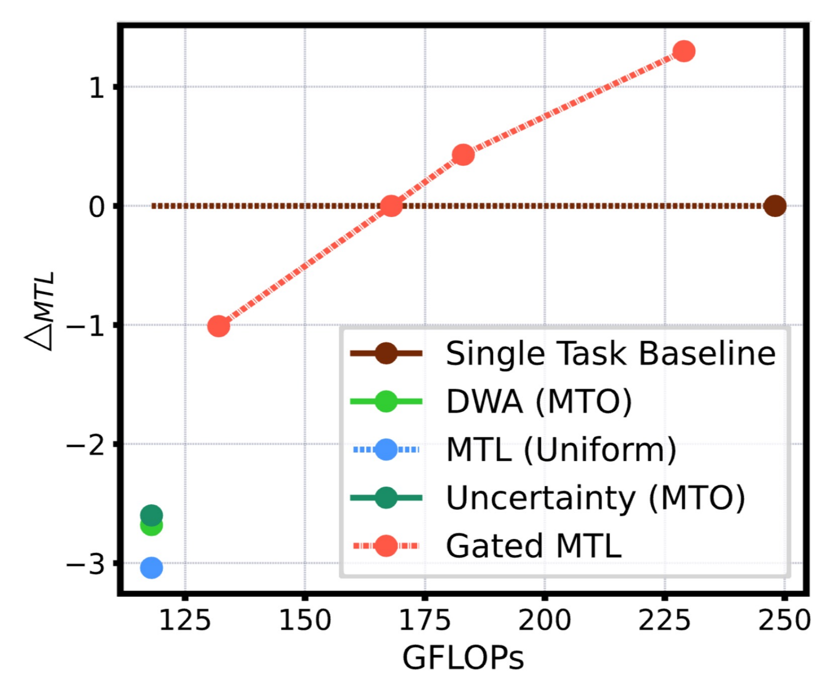
Model	Semseg ↑	Depth ↓	Normals ↓	Δ_{MTL} (%) ↑	Flops (G)	MR↓
STL	43.20	0.599	**19.42**	0	1149	9.0
MTL (Uni.)	43.39	0.586	21.70	-3.04	683	9.7
DWA	43.60	0.593	21.64	-3.16	683	9.7
Uncertainty	43.47	0.594	21.42	-2.95	683	10.0
Auto-λ	43.57	0.588	21.75	-3.10	683	10.0
RLW	43.49	0.587	21.54	-2.74	683	8.3
PCGrad	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 (ResNet18)

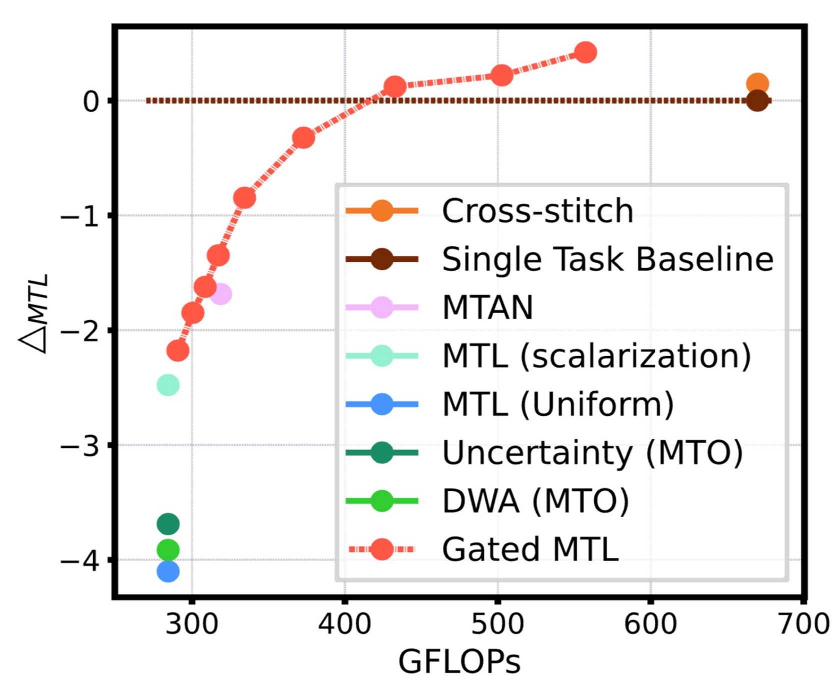
Model	Semseg ↑	Normals ↓	Saliency ↑	Human ↑	Edge ↓	Δ_{MTL} (%) ↑	Flops (G)	MR↓
STL	66.1	14.70	0.661	0.598	0.0175	0	670	6.0
MTL (uniform)	65.8	17.03	0.641	0.594	0.0176	-4.14	284	12.0
MTL (Scalar)	64.3	15.93	0.656	0.586	0.0172	-2.48	284	10.6
DWA	65.6	16.99	0.648	0.594	0.0180	-3.91	284	12.0
Uncertainty	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



CelebA (ResNet20)



NYUD-v2 (DPT with ViT-small)



PASCAL-Context (ResNet18)

Conclusion:

- InterroGate is an effective method in resolving task-interference via dedicated task-specific parameters and provides inference-time computational gains
- It demonstrates state-of-the-art performance across various architectures and on notable benchmarks such as CelebA, NYUD-v2, and PASCAL-Context

Limitations:

- **Increased Parameter Count:** It leads to an increased no. of parameters compared to MTO approaches
- **Performance-Cost Trade-Off:** The hyperparameters λ_s and τ_t help balance performance and computational cost, effectively approximating the desired FLOPs, but cannot guarantee a specific target FLOP