Supplemental Material

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

A Dataset Description

We use the three popular MTL benchmarks, CelebA [4], NYUD-v2 [6], and PASCAL-Context [7], to evaluate the performance of InterroGate. 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 [5, 10, 11], 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 [3] 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.

B Implementation and Training Details

B.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.

B.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 [1] as task-specific heads. For the HRNet-18 backbone, we follow the methodology of the original paper [9]: 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 [7]. 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 [7]. For learning rate decay, we adopt a polynomial learning rate decay scheme with a power of 0.9.

DPT training on NYUD-v2. We follow the same training procedure for DPT, 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.

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 [5, 8], and for PASCAL-Context, we similarly use the weights suggested in [5] and [8].

B.3 Setup

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. 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.

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 $\varphi_t^\ell \in R^{N^\ell \times C^\ell}$ and $\psi^\ell \in R^{N^\ell \times C^\ell}$ 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:

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

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

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

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 Equation 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 Forward-pass Pseudo-code

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

Algorithm 1: Pseudo-code for unified representation encoder

Given:

- $x \in R^{3\times W\times H}$ // Input image
- $T, L \in R$ // 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: $[\varphi_1^L, ..., \varphi_T^L]$ // Task-specific encoder representations

$\psi^0, \varphi_1^0, ..., \varphi_T^0 \leftarrow x$ // Set initial shared and task-specific features

for $\ell = 1$ to L **do**

  for $t = 1$ to T **do**

    $\varphi_t'^{\ell} \leftarrow G_t^{\ell}(\alpha_t^{\ell}) \odot \varphi_t^{\ell} + (1 - G_t^{\ell}(\alpha_t^{\ell})) \odot \psi^{\ell}$ (Equation 2)

    // Choose shared and task-specific features

    $\varphi_t^{\ell+1} \leftarrow F(\varphi_t'^{\ell}; \Phi_t^{\ell})$ // Compute task-specific features

  end for

  $\psi'^{\ell} = \sum_{t=1}^{T} \underset{t=1...T}{\text{softmax}}(\beta_t^{\ell}) \odot \varphi_t'^{\ell}$ (Equation 3)

  // Combine task-specific features to form shared ones

  $\psi^{\ell+1} \leftarrow F(\psi'^{\ell}; \Psi^{\ell})$ // Compute shared features

end for

E 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 [2]. The results are shown on Table 8. The forward and backward passes are averaged over 1000 iterations, after 10 warmup iterations, on a single NvidiaV100 GPU, with a batch size of 4.

F Additional Experiments

F.1 Full Results on CelebA and NYUD-v2

In Table 9, 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:

Table 8: 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

Table 9: 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↓
Original	STL	97.50	86.02	**93.00**	92.17	174	3.3
	MTL	97.28	86.70	92.35	92.11	58	4.7
	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**
Half	STL	96.99	85.60	**92.72**	91.77	44.4	3.7
	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**
Quarter	STL	96.64	85.22	**92.19**	91.35	11.6	3.3
	MTL	96.46	85.46	91.59	91.17	3.9	4.3
	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**

at original, half and quarter of the original model width. In Tables 10 and 11, 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 Δ_{MTL} scores.

Table 10: Performance comparison on NYUD-v2 using HRNet-18 backbone. Different InterroGate models are obtained by varying λ_s.

Model	Semseg ↑	Depth ↓	Normals ↓	Δ_{MTL} (%) ↑	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	-8.09 ± 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	-6.68 ± 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 ± 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 11: Performance comparison on NYUD-v2 using ResNet-50 backbone. Different InterroGate models are obtained by varying λ_s.

Model	Semseg ↑	Depth ↓	Normals ↓	Δ_{MTL} (%) ↑	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 $\pm$ 0.79	683	71.9	9.7
DWA	43.60	0.593	21.64	-3.16 $\pm$ 0.39	683	71.9	9.7
Uncertainty	43.47	0.594	21.42	-2.95 $\pm$ 0.40	683	71.9	10.0
Auto-λ	43.57	0.588	21.75	-3.10 $\pm$ 0.39	683	71.9	10.0
RLW	43.49	0.587	21.54	-2.74 $\pm$ 0.09	683	71.9	8.3
PCGrad	43.74	0.588	21.55	-2.66 $\pm$ 0.15	683	71.9	7.3
CAGrad	43.57	0.583	21.55	-2.49 $\pm$ 0.11	683	71.9	7.0
MGDA-UB	42.56	0.586	21.76	-3.83 $\pm$ 0.17	683	71.9	11.3
MTAN	**44.92**	0.585	21.14	-0.84 $\pm$ 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** $\pm$ 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	-0.50 $\pm$ 0.05	794	83.3	9.7

F.2 Ablation: Sharing/Specialization Patterns

As discussed in the manuscript, in this ablation, we investigate the gating patterns that the model converges to. Specifically, we study how much each task contributes to and benefits from the shared representations. To that aim, we show **(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); 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 with (a) Hinge loss, (b) medium-level pruning using uniform L_1 loss and (c) high-level pruning with uniform L_1 loss.

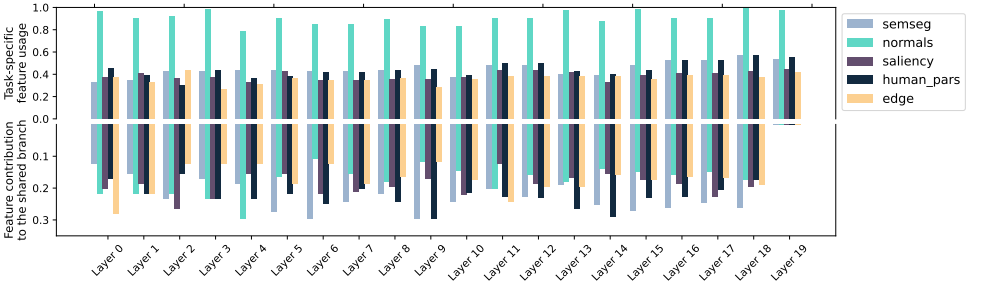
In all settings, the semantic segmentation task makes the largest contribution to the shared branch, followed by the normals prediction tasks. It is worth noting that the amount of feature contribution to the shared branch can also be largely influenced by other tasks' loss functions. In this situation, 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 (a). Intriguingly, the features of the normal task become less interesting to other tasks in this scenario possibly due to increased specialization.

F.3 Ablation: Sparsity Targets

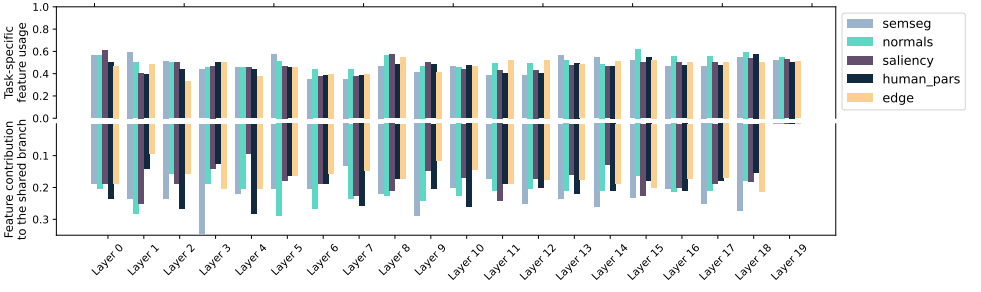
By tuning the sparsity targets τ in Equation 4, we can achieve specific compute budgets of the final network at inference. However, there are multiple choices of $\{\tau_t\}_{t=1}^{T}$ that can achieve the same budget. In this section, we further investigate the impact of which task we allocate more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

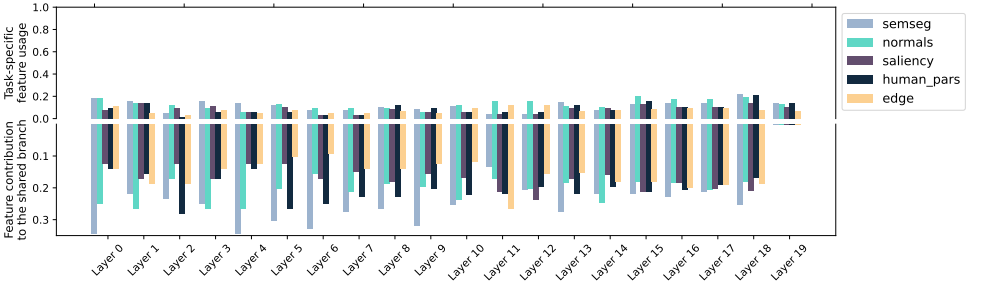
Our take-away conclusions are that **(i)** we clearly observe that some tasks require more



(a) With Hinge sparsity loss



(b) With uniform L_1 loss (medium pruning)

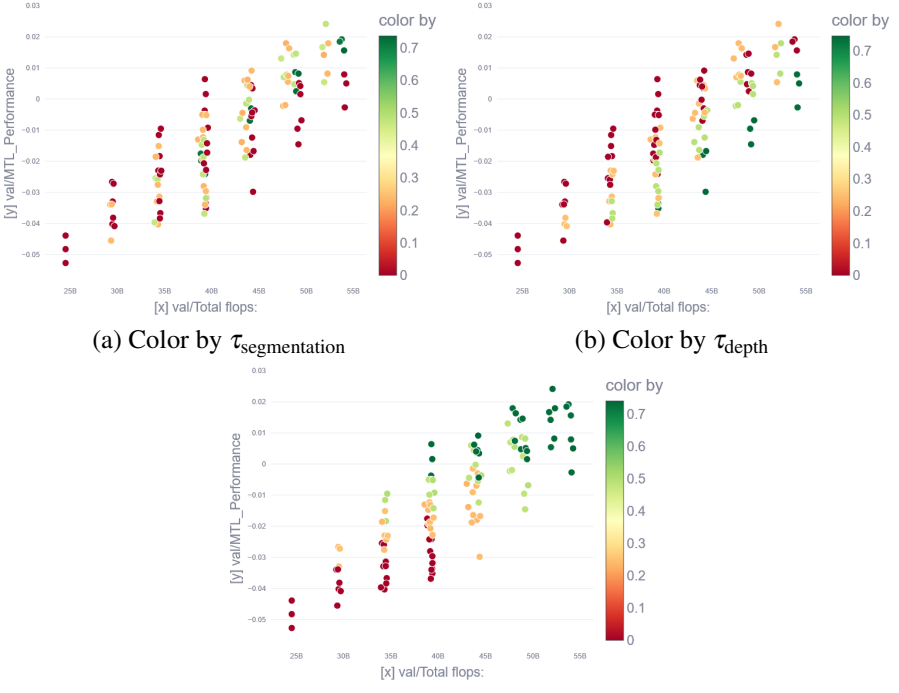


(c) With uniform L_1 loss (high pruning)

Figure 4: Sharing and specialization patterns on pascal context dataset with ResNet-18 backbone for InterroGate with (a) hinge loss, (b) L_1 loss with medium pruning and (c) L_1 loss with high pruning. Each sub-figure shows the task-specific representation selection ratio (top part) versus proportions of maximum contributions to the shared branch (bottom part).

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.



(a) Color by $\tau_{\text{segmentation}}$

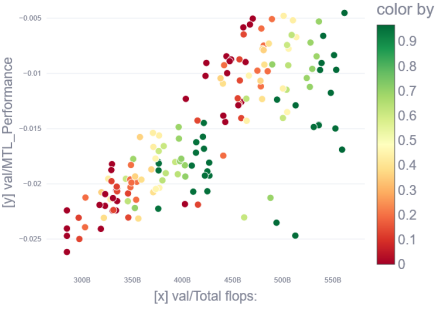
(b) Color by τ_{depth}

(c) Color by τ_{normals}

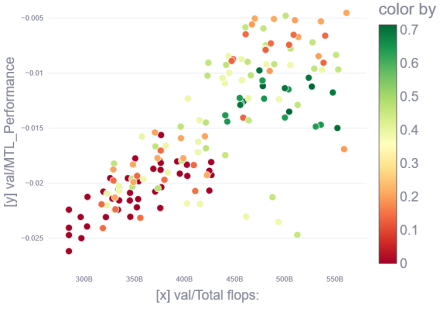
Figure 5: Sweeping over different $\{\tau_t\}$ on the NYUD-v2 experiments with HRNet-18 backbone. We plot the MTL performance Δ_{MTL} against the total number of FLOPs, then color each scatter point by the value of τ_t when the task t is (a) segmentation, (b) depth and (c) normals.

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 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.

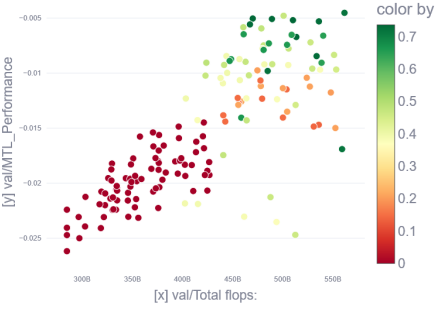
(a) Color by τ_{edges}



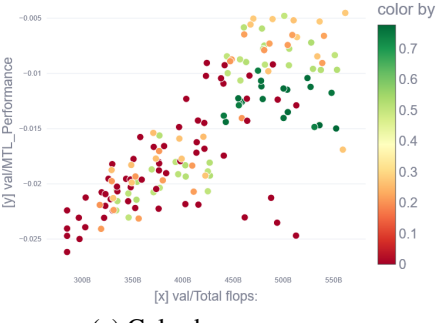
(b) Color by τ_{normals}



(c) Color by $\tau_{\text{human parts}}$



(d) Color by τ_{saliency}



(e) Color by $\tau_{\text{segmentation}}$

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.

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.

[2] 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.

[3] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

[10] 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.

[11] 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.