Supplementary Document for

Controllable Dynamic Multi-Task Architectures

Dripta S. Raychaudhuri[1] Yumin Suh[2] Samuel Schulter[2] Xiang Yu[2] Masoud Faraki[2]
Amit K. Roy-Chowdhury[1] Manmohan Chandraker[2,3]
[1]University of California, Riverside [2]NEC Labs America [3]University of California, San Diego

A. Resource Usage Plots

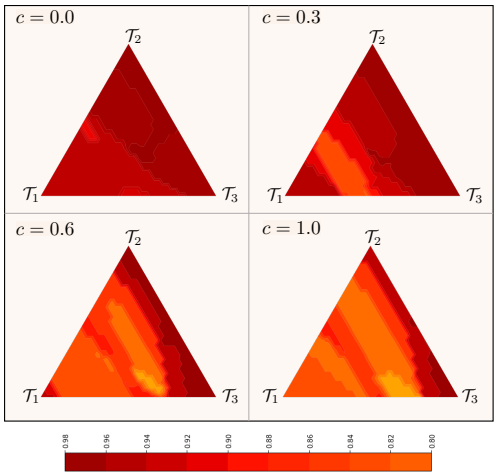


Figure A-1. **Resource usage on removing $\mathcal{L}_{\text{inact}}$**

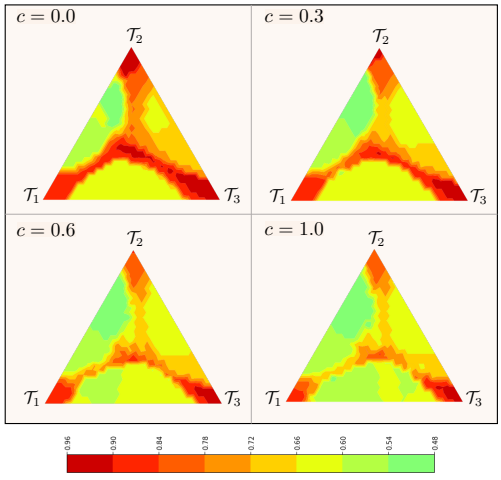


Figure A-2. **Resource usage on removing $\frac{L-l}{L}$**

In Figures A-1-A-4, we plot the resource usage across different preferences on the NYU-v2 dataset to analyze the effect of different parts of our framework. The primary observation in each case is the lack of proper controllability.

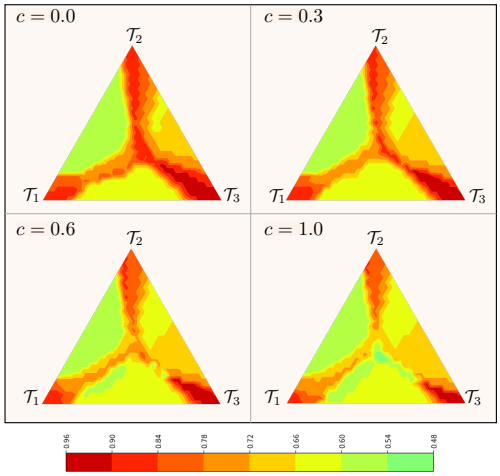


Figure A-3. **Resource usage on removing A**

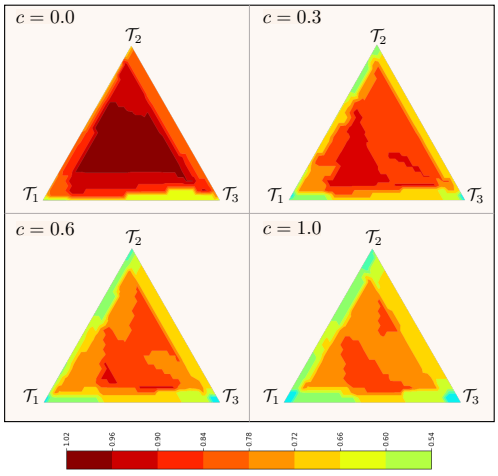
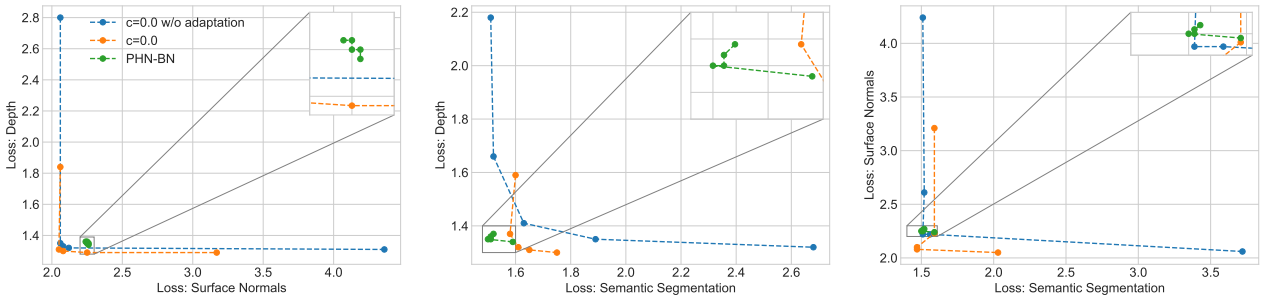


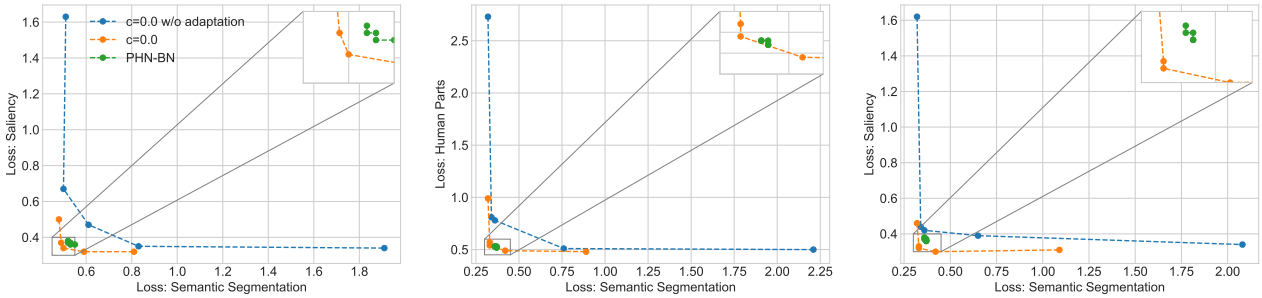
Figure A-4. **Resource usage on removing task dichotomy**

On removing $\mathcal{L}_{\text{inact}}$ or task dichotomy (Figures A-1,A-4), the overall resource usage increases across all preferences. On the other hand, removing the active loss branching weights (Figures A-2,A-3) leads to a misallocation of resources - more compute cost is allocated to skewed resources than the dense ones.



(a) Surface Normal/Depth (b) Semantic Seg./Depth (c) Semantic Seg./Surface Normal

Figure A-5. **Trade-off curves on NYU-v2:** Visualization of performance trade-off between pairs of tasks.



(a) Human Parts/Saliency (b) Semantic Seg./Human Parts (c) Semantic Seg./Saliency

Figure A-6. **Trade-off curves on PASCAL-Context (3 tasks):** Visualization of performance trade-off between pairs of tasks.

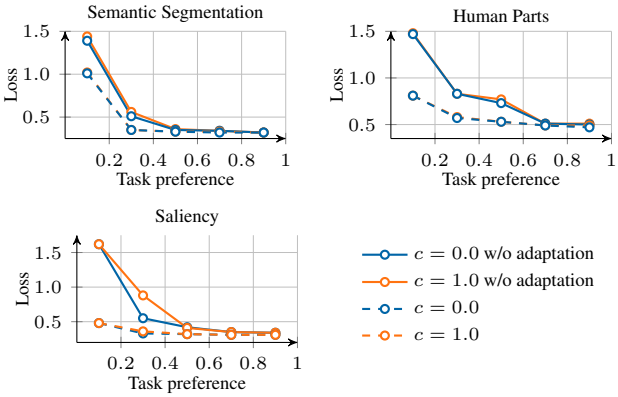


Figure A-7. **Marginal evaluation on PASCAL-Context (3 tasks)**

Method	$\mathcal{T}_1 \uparrow$	$\mathcal{T}_2 \uparrow$	$\mathcal{T}_3 \uparrow$	$\mathcal{T}_4 \downarrow$	$\mathcal{T}_5 \downarrow$	Avg $\Delta_T(\%) \uparrow$	**# Params** $(\%) \downarrow$
Single-Task	63.88	58.08	64.92	13.95	0.018	-	-
LTB	59.45	56.48	65.29	14.19	0.018	-3.61	-36.3
BMTAS	63.27	59.32	64.52	13.87	0.018	+0.37	-53.8
Ours, c=0.0	61.64	56.95	64.55	13.72	0.018	-1.45	-48.5
Ours†, c=0.0	60.03	56.91	64.59	13.95	0.018	-2.84	-48.5
Ours, c=1.0	61.15	56.85	64.67	13.74	0.018	-1.76	-50.1
Ours†, c=1.0	59.74	56.66	64.62	13.98	0.018	-3.20	-50.1

Table A-1. **Architecture evaluation on PASCAL-Context (5 tasks).** We report the mean intersection over union for $\mathcal{T}_1$: Semantic seg., $\mathcal{T}_2$: Parts seg., and $\mathcal{T}_3$: Saliency. We report mean error in angle for $\mathcal{T}_4$: Surface normal and mean loss for $\mathcal{T}_5$: Edge. Presence of † indicates that we train the networks from ImageNet weights, while its absence indicates training from anchor net weights.

B. Additional Results on Controllability

Trade-off curves. In Figures A-5 and A-6, we visualize the task controllability on pairs of tasks while keeping the preference on the rest fixed to zero. Compared to conventional dynamic models, our framework achieves a much larger dynamic range in terms of performance.

Marginal evaluation. We present the marginal evaluation of task controllability on the PASCAL-Context 3 task learning setting in Figure A-7.

C. Architecture Evaluation

Figure A-8 illustrates the architectures predicted by the edge hypernet for uniform task preference on the NYU-v2 3-task setting. As we increase c, the model size decreases via increased sharing of the low-level features. We also visualize the architectures for skewed preferences in Figure A-9. This leads to architectures which are predominantly a single-stream network, with the selected stream corresponding to the primary task. In all cases, Task 1 denotes semantic segmentation, Task 2 denotes surface normals estimation, and Task 3 denotes depth estimation. We report an additional evaluation of the predicted architectures on the Pascal-Context dataset in Table A-1. We choose the architecture

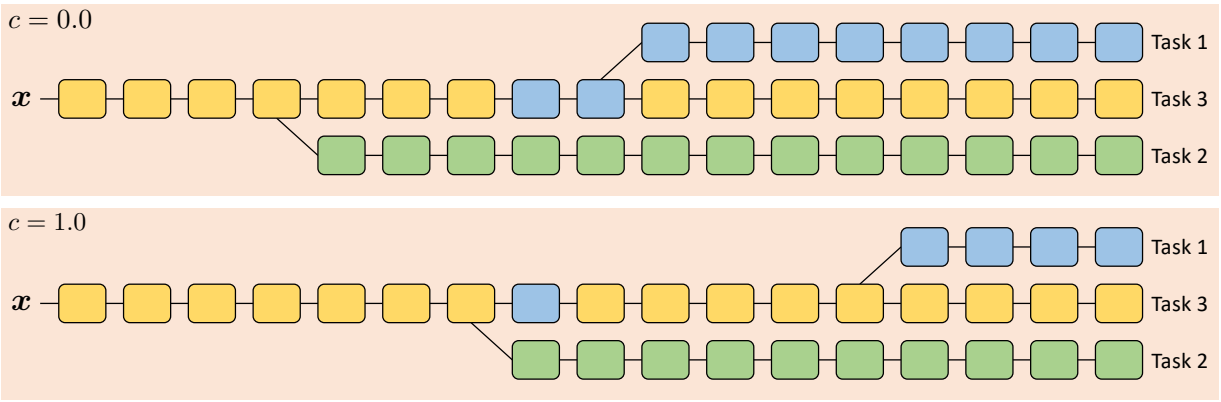


Figure A-8. **Predicted architectures on NYU-v2 for uniform task preference.**

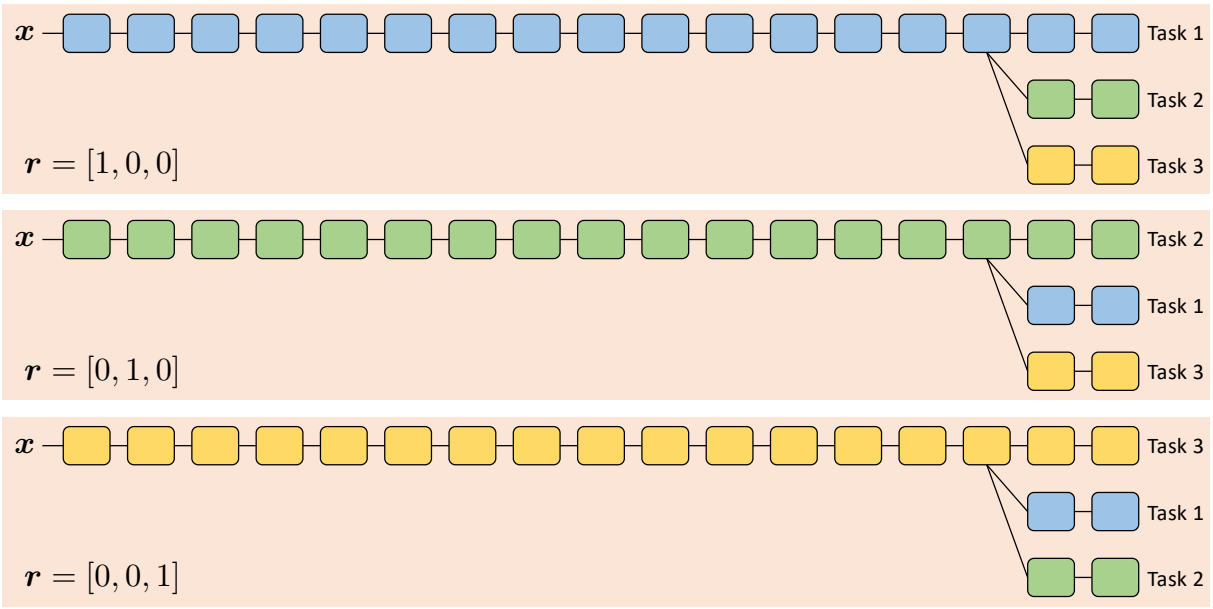


Figure A-9. **Predicted architectures on NYU-v2 for preferences focusing on a single task.**

predicted for a uniform task preference and, similar to LTB, we retrain it for a fair comparison.

D. Calculation of Task Affinity

We adopt *Representational Similarity Analysis* (RSA) [3] to obtain the task affinity scores from the anchor net F. First, using a random subset of K instances from the training set, we extract the features for each of these data points for every task. Let us denote the extracted feature map for instance k at layer l for task i as $\boldsymbol{f}_{[i,l,k]}$. Using these feature maps, we compute the feature similarity tensor $\mathbf{S}_l$ at each layer, of dimensions $N \times K \times K$, as follows,

$$\mathbf{S}_l\left(i,k,k'\right) = \frac{\langle \boldsymbol{f}_{[i,l,k]}, \boldsymbol{f}_{[i,l,k']} \rangle}{\|\boldsymbol{f}_{[i,l,k]}\| \|\boldsymbol{f}_{[i,l,k']}\|}. \qquad \text{(a-1)}$$

The task dissimilarity matrix D_l at layer l, of dimensions $N \times N$, is calculated as follows,

$$D_l(i,j) = \|\mathbf{S}_l\left(i,:,:\right) - \mathbf{S}_l\left(j,:,:\right)\|_F. \qquad \text{(a-2)}$$

The rows of D_l are normalized separately between $[0, 1]$ to get the scaled task dissimilarity matrix $\hat{D}_l$. The task affinity matrix is subsequently calculated as $A_l(i,j) = 1 - \hat{D}_l(i,j)$. The factor $A(i,j)$ is obtained by taking the mean of the affinity scores across layers, i.e., $A(i,j) = \frac{1}{L}\sum_l A_l(i,j)$. Figure A-10 presents an example of A for the PASCAL-Context dataset.

E. Calculation of P_{use}

$P_{use}(l,i)$ represents the probability that the node i at layer l is present in the tree structure that is sampled by the edge hypernet. We follow a dynamic programming approach to calculate this value. Denoting the i^{th} node in the l^{th} layer

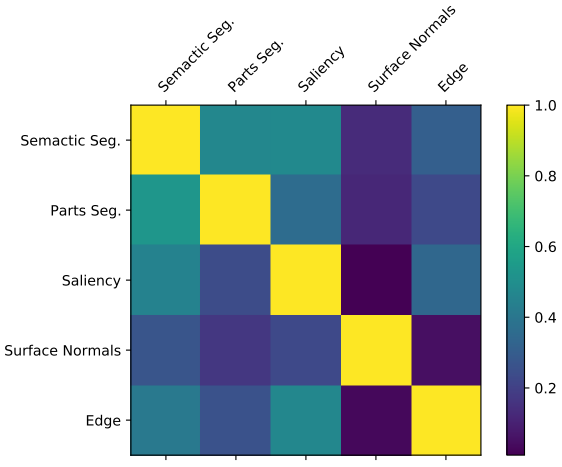


Figure A-10. **Task affinity for PASCAL-Context 5-task setting**

as (l, i), we can define the marginal probability that the node $(l+1, k)$ samples the node (l, i) as its parent as $\nu_k^l(i)$. Thus, the probability that the node (l, i) is used in the sampled tree structure is

$$P_{use}(l,i) = 1 - \prod_k \{1 - P_{use}(l+1,k) \cdot \nu_k^l(i)\}, \quad \text{(a-3)}$$

where $P_{use}(L, i) = 1$ for all $i \in [N]$.

Note that the actual path chosen from hypernet can be different from the ones whose α is penalized in (3). We also tested an alternative, which applies dynamic programming similarly as in (a-3), to penalize exact activated path. Since the performance gain was minor, we adopt (3) for simplicity.

F. Pseudo Code

Algorithm 1 Training controllable dynamic multi-task architecture.

Initialize F with single-task weights ▷ Anchor net
while h not converged **do** ▷ Edge hypernet
  $\boldsymbol{r} \sim \text{Dir}(\eta), c \sim \text{Unif}(0, 1)$
  Sample batch $\{(\boldsymbol{x}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)\}$
  $\hat{\boldsymbol{\alpha}} = h(\boldsymbol{r}, c; \phi)$
  Calculate $\Omega(\boldsymbol{r}, c, \hat{\boldsymbol{\alpha}})$
  Calculate $\mathcal{L}_{\text{task}}(\boldsymbol{r}) = \sum_i r_i \mathcal{L}_i(\boldsymbol{x}, \boldsymbol{y}_i, \text{F}, \hat{\boldsymbol{\alpha}})$
  Update ϕ by back-propagating $\mathcal{L}_{\text{task}} + \Omega$
while $\bar{h}$ not converged **do** ▷ Weight hypernet
  $\boldsymbol{r} \sim \text{Dir}(\eta), c \sim \text{Unif}(0, 1)$
  Sample batch $\{(\boldsymbol{x}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)\}$
  $\hat{\boldsymbol{\alpha}} = h(\boldsymbol{r}, c; \phi)$
  $\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}} = \bar{h}(\boldsymbol{r}, c; \bar{\phi})$
  Calculate $\mathcal{L}_{\text{task}}(\boldsymbol{r}) = \sum_i r_i \mathcal{L}_i(\boldsymbol{x}, \boldsymbol{y}_i, \text{F}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}})$
  Update $\bar{\phi}$ by back-propagating $\mathcal{L}_{\text{task}}$

G. Implementation Details

Hypernet architecture. The edge hypernet is constructed using an MLP with two hidden layers (dimension 100) and L linear heads, (dimension $N \times N$) which output the flattened branching distribution parameters at each layer. For the weight hypernet, we use a similar MLP with three hidden layers (dimension 100) and generate the normalization parameters using linear heads. In both cases we use learnable embeddings for each task (e_i) and the cost (e_c). Given the user preference $(\boldsymbol{r}, c)$, the preference embedding is calculated as $p = \sum_i r_i e_i + c e_c$. This embedding p is then used as input the MLP. The preference dimension is set to 32 in all experiments. PHN-BN and PHN [8] use a similar architecture to the weight hypernet, with PHN employing an additional chunking [4] embedding for scalability.

Task loss scaling. Due to the different scales of the various task losses, we first weight the loss terms with a factor w_i before applying linear scalarization with respect to $\boldsymbol{r}$. This ensures that the relative task importance is not skewed by the different loss scales. Thus, $\mathcal{L}_{\text{task}}(\boldsymbol{r}) = \sum_i r_i w_i \mathcal{L}_i$.

Anchor net architecture. The anchor net comprises N single-task networks, with each stream corresponding to a particular task. For experiments on dense prediction tasks, we use the DeepLabv3+ architecture [2] for each task. The MobileNetV2 [9] backbone is used for experiments on PASCAL-Context [7], while the ResNet-34 [5] backbone is used for experiments on NYU-v2 [10]. For CIFAR-100 [6] experiments we use the ResNet-9 [1] architecture with linear task heads.

Hyperparameters. For experiments on CIFAR-100 we use $\lambda_{\mathcal{A}} = 0.2$ and $\lambda_{\mathcal{I}} = 0.02$. For the rest, we use $\lambda_{\mathcal{A}} = 1$ and $\lambda_{\mathcal{I}} = 0.1$. The task scaling weights are given below:

1. **NYU-v2**:
   - Semantic segmentation: 1
   - Surface normals: 10
   - Depth: 3
2. **PASCAL-Context**:
   - Semantic segmentation: 1
   - Human parts segmentation: 2
   - Saliency: 5
   - Surface normals: 10
   - Edge: 50

For CIFAR-100, all tasks are equally weighted.

The threshold τ is set to 0.02 for CIFAR-100, 0.2 for NYU-v2 and PASCAL-Context (3 task), and 0.1 for PASCAL-Context (5 task).

Training. Hypernetworks are trained using Adam for 30K steps with a learning rate of $1e{-}3$, reduced by a factor of 0.3 every 14K steps. Temperature ζ is initialized to 5 and is decayed by 0.97 every 300 steps. The anchor net is an N

[1] https://github.com/davidcpage/cifar10-fast

η	0.1	0.2	0.5	1.0
c=0.0	4.20	4.26	4.26	4.26
c=1.0	4.18	4.25	4.25	4.21

Table A-2. **Varying parameterization of Dirichlet distribution.**

stream network where each stream is trained using individual task losses initialized from the same ImageNet pre-trained weights. Single-task networks for dense-prediction tasks are trained in accordance to [1]. For CIFAR, we use Adam with a learning rate of $1e{-}3$ and weight decay of $1e{-}5$ for 75 epochs.

Preference sampling. During training we sample preferences $(\boldsymbol{r}, c)$ from the distribution $P_{(\boldsymbol{r},c)} = P_{\boldsymbol{r}} P_c$, where $P_{\boldsymbol{r}}$ is defined as a Dirichlet distribution of order N with parameter $\eta = [\eta_1, \eta_2, \ldots, \eta_N]$ ($\eta_i > 0$) and P_c is defined as a standard uniform distribution $\text{Unif}(0, 1)$. Following [8], we set $\eta = (0.2, 0.2, \ldots, 0.2)$ for all our experiments. As shown in Table A-2, varying this parameter on PASCAL-Context (3 tasks) does not have any significant impact on the hypervolume.

H. Effect of τ

Setting $\tau \gg \frac{1}{N}$ leads to virtually all tasks being treated as inactive. This results in lack of control since there is no active task to be tied to the resource control c. Also, there is no compression since the inactive loss searches for active tasks to share nodes with. Consequently, this mostly leads to a fully branched out network which explains the flat curve in Figure 8. When $\tau = \frac{1}{N} + \epsilon$ ($\epsilon \geq 0$), both losses are in effect, which results in proper controllability, but possible ignorance of the uniform (or near uniform) preferences explains the slightly poorer performance. In our experiments, $\tau = 0.6/N$ works well across all datasets.

I. Computational Resource

Controller overhead. The usage of hypernetworks leads to additional resource usage which we term as controller overhead. In comparison to PHN-BN, while we incur a larger overhead due to the prediction of a higher number of normalization parameters, the effect is minimal due to the controller being active infrequently only during preference change. This overhead has no impact on inference.

Model size. In this work, we consider inference time as a measure of computational resource. We believe fast inference matters in many practical scenarios, whereas reasonable amount of increase in memory overhead is relatively easy to handle. This justifies our design choice to introduce the anchor net.

J. Hypervolume

Given a point set $S \subset \mathbb{R}^N$ and a reference point $p \in \mathbb{R}^N$ in the loss space, the hypervolume of the set S is defined as the size of the region dominated by S and bounded above by p,

$$\text{HV}(S) = \lambda\left(\{a \in \mathbb{R}^N | \exists b \in S : b \preceq a \text{ and } b \preceq p\}\right), \tag{a-4}$$

where λ is the Lebesgue measure. Figure A-11 shows an example in two-dimensions.

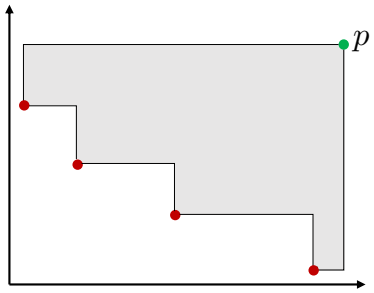


Figure A-11. **Hypervolume calculation.** The shaded area represents the hypervolume obtained by the point set comprising the red points, with respect to the reference green point p.

K. Potential Negative Societal Impact

The requirement of heavy computation makes neural architecture search (NAS) environmentally unfriendly. As pointed out in [11], the CO2 emissions from a NAS process can be comparable to that from 5 cars' lifetime. Since our framework is fundamentally a NAS method for multi-task learning, it shares these drawbacks. However, due to the dynamic nature of our method, it saves energy in the long run by allowing a single network to emulate different models corresponding to various preferences.

References

[1] David Bruggemann, Menelaos Kanakis, Stamatios Georgoulis, and Luc Van Gool. Automated search for resource-efficient branched multi-task networks. *arXiv preprint arXiv:2008.10292*, 2020. A-5

[2] 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*, pages 801–818, 2018. A-4

[3] Kshitij Dwivedi and Gemma Roig. Representation similarity analysis for efficient task taxonomy & transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12387–12396, 2019. A-3

[4] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. A-4

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. A-4

[6] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. A-4

[7] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2014. A-4

[8] Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations*, 2021. A-4, A-5

[9] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. A-4

[10] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of the European Conference on Computer Vision*, 2012. A-4

[11] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019. A-5