Active Learning with Task Consistency and Diversity in Multi-Task Networks (Supplementary Material)

Aral Hekimoglu
Technical University Munich
Munich, Germany
aral.hekimoglu@tum.de

Michael Schmidt
BMW Group
Munich, Germany
michael.se.schmidt@bmw.de

Alvaro Marcos-Ramiro
BMW Group
Munich, Germany
alvaro.marcos-ramiro@bmw.de

1. Implementation details of MMD blocks

For each task, we establish pairwise connections with $K-1$ other tasks using pairwise MMD blocks, resulting in $C(K,2)$ additional blocks, where C denotes combinations. These blocks take the initial task predictions as inputs and produce two predictions for each block, corresponding to the involved tasks. We use the same MMD block used in [4] with an attention mechanism for guiding message passing between the two tasks. As shown in Fig. 8, each block contains a convolutional layer that transforms the input into feature representations (F_i^1, F_i^2), an attention-guided gating mechanism (G) refining task feature representations with information from the other task, and finally, a decoder generating refined task predictions. During the training of these additional MMD blocks, we freeze the backbone and the initial task prediction layers from the PAD-Net and separately train the MMD blocks using the respective tasks' loss functions.

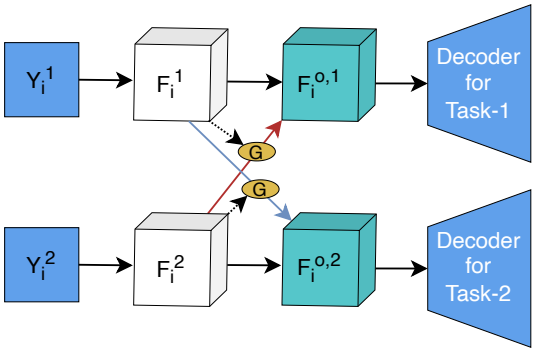

Figure 8. Illustration of the used pairwise MMD block. Y_1^i, Y_2^i represent the initial task predictions for the two tasks. G denotes the generated attention map which is used as guidance in the distillation.

2. Implementation details for the auto-encoder

The architecture of our auto-encoder is designed to process concatenated task-specific feature maps as input. Both the encoder and decoder components of the auto-encoder utilize fully convolutional designs.

Encoder architecture. The encoder segment consists of four sequential blocks, each comprising a convolutional layer followed by batch normalization and a Rectified Linear Unit (ReLU) activation. The convolutional layers make use of 4x4 filters. The specific architecture is visually represented below:

$$\begin{aligned} R^{4C\times H\times W} &\longrightarrow \text{Conv}_{128} \longrightarrow \text{BN} \longrightarrow \text{ReLU} \\ &\longrightarrow \text{Conv}_{256} \longrightarrow \text{BN} \longrightarrow \text{ReLU} \\ &\longrightarrow \text{Conv}_{512} \longrightarrow \text{BN} \longrightarrow \text{ReLU} \\ &\longrightarrow \text{Conv}_{1024} \longrightarrow \text{BN} \longrightarrow \text{ReLU} \longrightarrow \text{FC}_{512} \end{aligned}$$

Decoder architecture. Mirroring the encoder, the decoder also comprises four sequential blocks. However, in place of standard convolutional layers, the decoder utilizes transposed convolutional layers to reconstruct the feature maps. Like the encoder, it employs batch normalization and ReLU activation. The layout is as follows:

$$\begin{aligned} z \in R^{512} &\longrightarrow \text{FC}_{1024\times \frac{H}{4}\times \frac{W}{4}} \\ &\longrightarrow \text{ConvT}_{512} \longrightarrow \text{BN} \longrightarrow \text{ReLU} \\ &\longrightarrow \text{ConvT}_{256} \longrightarrow \text{BN} \longrightarrow \text{ReLU} \\ &\longrightarrow \text{ConvT}_{128} \longrightarrow \text{BN} \longrightarrow \text{ReLU} \longrightarrow \text{ConvT}_{4C} \end{aligned}$$

Notation. Conv_k denotes a convolutional layer with k filters. ConvT_k signifies a transposed convolutional layer with k filters. All the convolutions use a stride of 2 and SAME padding. BN refers to the batch normalization layer, ReLU is used for the Rectified Linear Units, and FC_k indicates a fully connected layer mapping to R^k.

3. Implementation details for the baselines

For the LL4AL [5] baseline, we use task-specific feature representations as input of a separate feed-forward network that predicts the loss of the corresponding task. This network is trained concurrently with the PAD-Net network using the loss function and the optimization strategy utilized in [5].

For EquAL [2] baselines, we apply horizontal flipping to the original image and make predictions for the flipped version. Then, we revert the flipped predictions and compare them with the predictions of the original image. For classification tasks like semantic segmentation, saliency estimation, and edge detection, we calculate the pixel-wise entropy between the original and flipped predictions. For regression tasks like depth and surface normals estimation, we employ differential entropy. We sum pixel-wise scores to form the selection score for the image.

For PartAL [1], which involves partially selecting tasks of a sample for labeling, we set the labeling budget for tasks to be three times the labeling budget for images in the NYU dataset and five times for PASCAL images. Doing so ensures a consistent total number of labeled tasks across all our baselines.

4. Adapting to other MTL architectures

We apply our approach to the MTI-Net architecture [3]. We introduce pairwise MMD connections to tasks using the initial task predictions at the highest scale (Scale 1/4). We construct the same committee of refined predictions to define the inconsistency score. To extract diversity embeddings using the auto-encoder architecture, we input the features from the same scale (Scale 1/4).

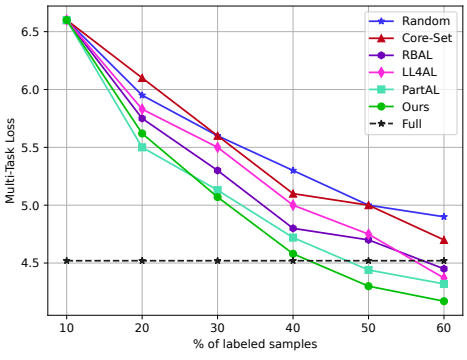

Figure 9. Comparison of our proposed method using MTI-Net with SOTA AL methods on the PASCAL dataset.

Fig. 9 shows similar baseline comparisons as our experiments using the PAD-Net architecture. This suggests that our approach can be easily adapted to alternative multi-task network architectures featuring a similar two-stage structure.

5. Additional inconsistency experiments

In Eq. (1), we aggregate the scores of each committee member into a single-task inconsistency score by selecting the maximum value. To explore various aggregation approaches, we compare this method against averaging and selecting the minimum in Tab. 2. The results indicate that utilizing the maximum score yields the best performance. As a result, we adopt this configuration in our experiments.

	Min	Avg.	Max
Depth	0.82	0.69	**0.67**
Norm.	25.2	22.3	**21.5**
Segm.	29.6	32.9	**33.6**
Loss	8.03	7.37	**7.08**

Table 2. Performance of networks after the final AL iterations, using different task-score aggregation variants on the *NYUD-v2*.

6. Additional diversity experiments

Fig. 10 presents a comparison between two distance functions d to use in defining the diversity score in Eq. (3). We compare Euclidean and cosine distance, across various feature dimension sizes. We observe that the lowest loss is achieved using Euclidean distance with feature vectors of dimension 512. Therefore, we use this configuration in our experiments.

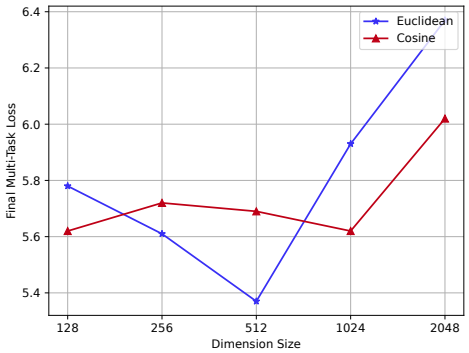

Figure 10. Performance of networks after the final AL iterations, using different distance functions and varying feature embedding sizes on the *NYUD-v2*.

7. Ablation study on individual scores and aggregation

To generate a unified selection score, we combine the inconsistency-based scoring s_k and diversity-based scoring div. Our final selection score is computed as the product of these two metrics. In Fig. 11, we present an ablation study to evaluate different approaches for combining these scores. We examine four distinct scenarios: 1) using the inconsistency score s_k alone, 2) using only the diversity score div alone, 3) averaging the single-task scores, and 4) taking the maximum of the single-task scores. The results indicate that while the inconsistency-based scoring s_k alone yields better performance than the diversity-based score div alone, combining both yields the best overall results. Additionally, when aggregating scores from individual tasks, taking the maximum score slightly outperforms averaging.

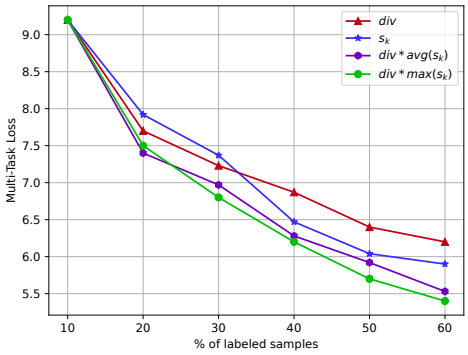

Figure 11. Ablation study on individual scoring metrics (inconsistency-based s_k and diversity-based div) and two aggregation strategies (averaging and taking the maximum)

8. Numerical active learning results

Due to the limited space in the paper, we present our AL comparisons as plots. Tables 3 - 12 provide the numerical values for Figures 4a - 4f and 5a - 5d from the main paper. The mean and variances of three experiments trained with different random initializations are presented.

References

[1] Nikita Durasov, Nik Dorndorf, and Pascal Fua. Partal: Efficient partial active learning in multi-task visual settings. *arXiv preprint arXiv:2211.11546*, 2022. 2

[2] S Alireza Golestaneh and Kris M Kitani. Importance of self-consistency in active learning for semantic segmentation. In *BMVC*, 2020. 2

[3] Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction networks for multi-task learning. In *ECCV*, 2020. 2

[4] 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. In *CVPR*, 2018. 1

[5] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *CVPR*, 2019. 2

	20	30	40	50	60
Random	8.4±0.21	7.8±0.09	7.4±0.13	7.0±0.20	6.9±0.11
Core-Set	8.1±0.12	7.6±0.19	7.2±0.11	6.5±0.06	6.3±0.13
RBAL	7.8±0.18	7.5±0.24	7.0±0.23	6.7±0.21	6.7±0.03
LL4AL	8.0±0.05	7.3±0.05	6.7±0.08	6.2±0.04	6.2±0.11
PartAL	7.8±0.16	7.0±0.13	6.4±0.07	6.0±0.11	5.9±0.12
EquAL*	8.2±0.17	7.7±0.06	7.3±0.21	6.6±0.09	6.4±0.10
Ours	7.5±0.17	6.8±0.10	6.2±0.16	5.7±0.05	5.4±0.08

Table 3. Comparison of multi-task loss with SOTA AL methods on the PASCAL dataset. (Fig. 4a)

	20	30	40	50	60
Random	54.3±0.29	57.3±0.17	57.9±0.35	58.8±0.28	60.0±0.25
Core-Set	55.3±0.22	58.1±0.28	58.5±0.11	59.2±0.14	61.0±0.27
RBAL	55.4±0.30	58.4±0.21	59.5±0.25	60.2±0.31	62.2±0.29
LL4AL	56.4±0.20	59.4±0.22	60.1±0.22	60.8±0.22	61.9±0.16
EquAL-SS	56.9±0.30	59.8±0.17	60.6±0.30	61.1±0.12	62.5±0.27
PartAL	57.8±0.22	60.3±0.28	61.4±0.29	62.3±0.23	64.1±0.14
Ours	57.6±0.13	61.7±0.16	61.9±0.27	62.8±0.18	64.5±0.15

Table 4. Comparison of semantic segmentation (mIoU) with SOTA AL methods on the PASCAL dataset. (Fig. 4b)

	20	30	40	50	60
Random	64.3±0.27	66.0±0.28	68.0±0.15	68.6±0.11	70.5±0.25
Core-Set	65.2±0.07	66.3±0.19	68.8±0.06	69.5±0.19	71.2±0.10
RBAL	66.4±0.17	68.2±0.23	70.8±0.17	71.3±0.21	73.4±0.27
LL4AL	65.7±0.21	66.5±0.22	69.3±0.14	70.5±0.29	72.0±0.25
EquAL-ED	66.1±0.21	69.2±0.29	70.7±0.29	71.7±0.15	74.0±0.21
PartAL	65.8±0.28	68.8±0.21	70.3±0.17	71.3±0.19	73.8±0.29
Ours	67.4±0.11	69.3±0.16	71.7±0.11	72.2±0.29	74.2±0.19

Table 5. Comparison of edge detection (odsF) with SOTA AL methods on the PASCAL dataset. (Fig. 4c)

	20	30	40	50	60
Random	18.7±0.10	18.2±0.03	18.0±0.06	17.8±0.09	17.6±0.03
Core-Set	18.6±0.09	18.1±0.07	17.8±0.08	17.6±0.02	17.5±0.07
RBAL	18.4±0.03	17.9±0.03	17.7±0.03	17.5±0.04	17.3±0.08
LL4AL	18.6±0.08	18.0±0.05	17.9±0.09	17.7±0.06	17.4±0.08
EquAL-SN	18.5±0.09	18.0±0.10	17.9±0.03	17.7±0.09	17.5±0.05
PartAL	18.2±0.04	17.8±0.04	17.6±0.05	17.4±0.05	17.3±0.04
Ours	18.3±0.07	17.6±0.07	17.5±0.02	17.3±0.07	17.1±0.04

Table 6. Comparison of surface normals estimation (mErr) with SOTA AL methods on the PASCAL dataset. (Fig. 4d)

	20	30	40	50	60
Random	50.7±0.16	52.2±0.07	53.0±0.30	54.1±0.19	55.3±0.23
Core-Set	51.2±0.27	52.9±0.21	53.6±0.24	54.7±0.21	55.7±0.25
RBAL	52.3±0.15	53.3±0.15	54.5±0.16	55.3±0.11	56.7±0.17
LL4AL	52.6±0.16	54.2±0.18	55.2±0.24	56.4±0.15	57.1±0.12
EquAL-HPS	54.0±0.20	55.5±0.28	56.5±0.25	57.5±0.15	58.6±0.12
PartAL	53.7±0.07	55.0±0.19	56.0±0.18	56.9±0.27	58.5±0.17
Ours	54.5±0.27	56.0±0.14	57.2±0.08	58.4±0.18	59.2±0.20

Table 7. Comparison of human parts segmentation (mIoU) with SOTA AL methods on the PASCAL dataset. (Fig. 4e)

	20	30	40	50	60
Random	56.3±0.44	60.1±0.24	61.6±0.27	62.2±0.28	62.8±0.31
Core-Set	58.2±0.49	61.1±0.46	62.3±0.23	62.9±0.46	63.2±0.34
RBAL	58.2±0.41	61.1±0.31	62.3±0.37	62.9±0.32	63.2±0.41
LL4AL	60.4±0.50	62.6±0.31	63.7±0.47	64.2±0.25	64.5±0.47
EquAL-SE	59.9±0.47	62.0±0.44	63.1±0.42	63.7±0.37	63.9±0.43
PartAL	59.7±0.36	63.4±0.50	64.3±0.23	65.0±0.40	65.5±0.26
Ours	59.1±0.40	63.3±0.40	64.9±0.33	65.7±0.46	66.5±0.22

Table 8. Comparison of saliency estimation (mIoU) with SOTA AL methods on the PASCAL dataset. (Fig. 4f)

	20	30	40	50	60
Random	9.3±0.12	8.8±0.03	8.6±0.12	8.3±0.04	8.0±0.15
Core-Set	9.2±0.07	8.7±0.16	8.4±0.03	8.2±0.05	7.9±0.05
EquAL*	9.1±0.10	8.5±0.07	8.3±0.14	8.0±0.13	7.8±0.20
RBAL	9.1±0.13	8.6±0.15	8.2±0.16	8.0±0.12	7.8±0.13
LL4AL	9.0±0.14	8.5±0.19	8.2±0.07	7.7±0.20	7.6±0.03
PartAL	8.9±0.08	8.4±0.12	7.9±0.12	7.5±0.08	7.3±0.07
Ours	8.7±0.16	8.2±0.07	7.7±0.04	7.2±0.16	7.1±0.06

Table 9. Comparison of multi-task loss with SOTA AL methods on the NYU dataset. (Fig. 5a)

	20	30	40	50	60
Random	21.2±0.30	24.6±0.30	26.3±0.31	27.9±0.33	30.1±0.35
Core-Set	21.7±0.27	25.0±0.36	26.7±0.35	28.7±0.29	30.9±0.36
RBAL	21.8±0.23	25.2±0.37	27.2±0.22	28.3±0.25	30.5±0.25
EquAL-SS	23.0±0.38	26.5±0.37	28.6±0.28	30.5±0.35	32.9±0.30
LL4AL	22.6±0.23	26.2±0.39	28.0±0.27	29.4±0.31	31.6±0.28
PartAL	23.3±0.31	27.0±0.34	29.3±0.21	31.6±0.40	32.7±0.24
Ours	23.9±0.24	28.0±0.26	30.4±0.34	32.3±0.23	33.6±0.26

Table 10. Comparison of semantic segmentation (mIoU) with SOTA AL methods on the NYU dataset. (Fig. 5b)

	20	30	40	50	60
Random	0.88±0.009	0.85±0.006	0.83±0.003	0.82±0.014	0.81±0.004
Core-Set	0.87±0.018	0.82±0.019	0.79±0.016	0.77±0.009	0.75±0.012
RBAL	0.82±0.007	0.80±0.020	0.76±0.017	0.74±0.012	0.73±0.004
EquAL-DE	0.82±0.012	0.78±0.017	0.75±0.016	0.72±0.017	0.71±0.010
LL4AL	0.83±0.001	0.80±0.009	0.78±0.007	0.74±0.013	0.72±0.009
PartAL	0.79±0.008	0.74±0.007	0.73±0.003	0.72±0.011	0.70±0.019
Ours	0.77±0.018	0.74±0.014	0.71±0.017	0.68±0.018	0.67±0.016

Table 11. Comparison of depth estimation (mse) with SOTA AL methods on the NYU dataset. (Fig. 5c)

	20	30	40	50	60
Random	28.4±0.07	27.3±0.05	26.8±0.09	25.6±0.15	25.0±0.19
Core-Set	28.2±0.18	27.0±0.14	25.8±0.05	25.0±0.17	24.6±0.19
RBAL	25.9±0.11	24.8±0.07	24.0±0.07	23.5±0.11	23.1±0.11
EquAL-SN	25.4±0.14	24.2±0.10	23.5±0.15	23.2±0.10	22.8±0.17
LL4AL	27.0±0.17	25.9±0.08	25.1±0.11	24.3±0.17	23.8±0.08
PartAL	24.5±0.19	23.7±0.13	22.8±0.07	22.5±0.19	22.3±0.10
Ours	24.9±0.12	23.1±0.13	22.5±0.09	21.8±0.14	21.5±0.08

Table 12. Comparison of surface normals estimation (mErr) with SOTA AL methods on the NYU dataset. (Fig. 5d)