Supplemental Material to NeurIPS submission #12872 "Scalarization for Multi-Task and Multi-Domain Learning at Scale"

1 Experimental settings and training hyperparameters

1.1 Multi-Domain

We refer to benchmarks as "multi-domain" when they contain multiple input visual domains with a shared set of output classes (i.e., $\forall i \neq j, X_i \neq X_j$ and $Y_i = Y_j$).

CIFAR-10 and STL-10. CIFAR-10 [26] is a classical benchmark for image classification containing 50k training samples uniformly distributed across 10 classes. STL-10 [9] is a semi-supervised dataset which was designed to resemble CIFAR-10. Specifically, we only use the 5000 annotated images in STL-10, which are also uniformly distributed across the same 10 classes as CIFAR. In STL-10, the images themselves are from the ImageNet [16] dataset, and cropped/resized to 96 pixels. We further resize them to 32 pixels to align with CIFAR. In summary, the key difficulties are (i) the input distribution shift between the two datasets and (ii) the high imbalance in data availability.

In terms of architecture, we use a vision transformer backbone (ViT-S) optimized for small-scale datasets [15] (compared to the original ViT-S, this backbone contains smaller patch sizes, fewer transformer layers and narrower embeddings but a higher number of heads). To control model capacity, we vary the depth (number of transformer layers) in $\{3, 6, 9\}$ and the width (token dimension) in $\{48, 96, 144, 192\}$, Finally, we train each model from scratch on a single NvidiaV100 GPU with a batch size of 256 images for 300 epochs (including 30 epochs of linear learning rate warmup), using a learning rate of 0.001 and weight decay of 0.05 with the AdamW optimizer and cosine learning rate decay.

DomainNet. DomainNet [44] is a classification dataset of 6 visual domains annotated for 345 classes, for a total of roughly 410k training samples. DomainNet was initially introduced for the problem of multi-source domain adaption, in which one or more of the domains does not have training annotations; the key difficulty is thus to learn representations that are aligned across domains. In contrast to the CIFAR+STL example, DomainNet exhibits distribution shifts across both the input domains and output classes, as visualized in Figure A.

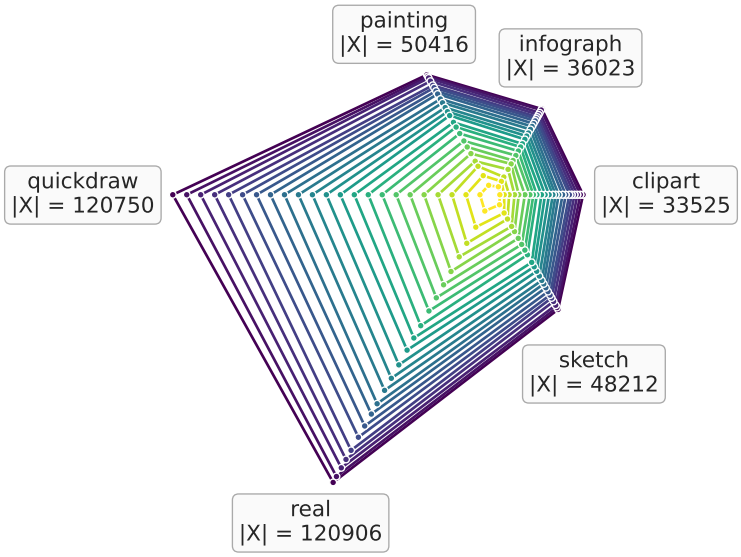


Figure A: Illustrating the data imbalance in DomainNet with a contour plot of the number of samples per class and domains in DomainNet. Each of the corners of the hexagon represents one of the six domains in DomainNet, and the lines (levels of the contour plot) represent the number of samples, drawn every 15 classes. For comparison, a uniformly distributed dataset would yield perfect hexagons.

Following previous literature [44, 29], we use a ResNet-101 as our main backbone. There is no domain-dependent layer in the architecture: the final classifier layer is shared across all domains. We first perform a training sweep over the largest domain (`real`) to select the best-performing learning rate from $\{0.3, 0.03, 0.003, 0.0003\}$ and the number of epochs from $\{30, 60, 90\}$. Following these results, we use a learning rate of 0.03 and train for 30 epochs with a batch size of 512 in subsequent experiments. We train with the AdamW optimizer with a weight decay of $1e-4$. We also apply linear learning rate warm-up during the first five training epochs and use cosine schedule learning rate decay for the rest of the training. Finally, to control model capacity, we vary the depth (backbone) in {ResNet-26, ResNet-50, ResNet-101 } and the width base bottleneck dimension) in $\{16, 32, 64\}$,

Multi-domain, resampling and training length In the multi-domain setting, scalarization weights become resampling probabilities for each dataset, as shown in (2). Consequently, the notion of "epoch" is hard to define compare to the standard mono-dataset setting. To resolve this, we always define epochs with respect to one of the domains. For instance, in the CIFAR+STL case, we use STL as our reference. Therefore, "one epoch" translates to seeing as many samples as in the original STL dataset (5000) using the current batch size (256), i.e. roughly 20 training steps. In the DomainNet case, we define epochs relatively to the `real` domain. This definition has the advantage of not being impacted by the sampling weights $\mathbf{p}$; In particular, this means that both the MDL models and the single dataset (SD) baselines are trained for as many training steps, and see the same amount of training samples, only sampled from different data distributions.

1.2 Multi-task

We define multi-task benchmarks as datasets where every image is fully annotated for multiple output tasks (i.e., $\forall i \neq j, X_i = X_j$ and $Y_i \neq Y_j$). This setting is particularly popular for scene understanding problems where every scene is labelled with multiple dense predictions (e.g. depth, normals, segmentation mas, edges, etc.)

Celeba. CelebA is a binary attribute classification dataset containing 40 attributes and roughly 162k training images. To turn CelebA into a multi-task problem, it is common to consider each attribute as a binary classification task: More specifically, we use a fully shared backbone with a final linear layer of 40 outputs, outputting logits for every task. The model is then trained using 40 binary cross-entropy losses, one for each attribute. To make our comparative analysis more scalable, we define several tasks as subsets of attributes, grouped based on semantic similarity (e.g. all hair colors are in the same subgroup). The 8 resulting subsets of attributes are described in Table A. In the scalarization setting, this simply means that some of the attributes share the same importance weight.

As a backbone, we use the same ViT-S/4 based architecture as for CIFAR/STL. We train for 50 epochs with 5 epochs of learning rate warmup. We use a learning rate of 0.0005 with cosine schedule decay anf train with the AdamW optimizer with a weight decay of 0.05. We use input images of size 32 (with tokens of size 4), a batch size of 256, and RandAugment data augmentations.

Table A: The eight tasks defined as subsets from CelebA attributes used in our main analysis. Attributes in the same subset share a common importance weight p

Hair color	Hairstyle	Facial Hair	Mouth	Clothes	Face Structure	Gender	Age
Black Hair	Bald	5'o'Clock Shadow	Big Lips	Eyeglasses	Big Nose	Male	Young
Blond Hair	Bangs	Mustache	Mouth Slightly Open	Heavy Makeup	Chubby		
Brown Hair	Receding Hairline	No Beard	Smiling	Wearing Earrings	Double Chin		
Gray Hair	Sideburns	Goatee	Wearing Lipstick	Wearing Hat	High Cheekbones		
	Straight Hair			Wearing Necklace	Oval Face		
	Wavy Hair			Wearing Necktie	Pointy Noise		

Taskonomy. Taskonomy [61] is a large dataset containing a variety of dense prediction tasks for indoor scenes. We use the `tiny` split of Taskonomy which contains roughly 275k images. Taskonomy was originally introduced for the problem of task clustering: The original work [61] proposes a task affinity metric to define a taxonomy of tasks. This taxonomy structure is then used to determine which tasks should be trained from scratch and which tasks could benefit from others via transfer learning. Closer to our setting, follow-up works [54, 13] propose to use this taxonomy to determine which tasks should be grouped or not in multi-task learning. Once the groupings are determined, a

separate backbone is trained for each group of tasks, typically using standard uniform scalarization. Instead, for our analysis, we use Taskonomy-tiny in a more standard multi-task framework, where a backbone is shared across tasks.

For training, we follow the methodology of [54]. We use a ResNet-26 backbone (with varying bottleneck width) with a mirrored decoder; By default, only the encoder is shared across tasks and each task receives its own decoder. To vary model capacity, we add the option to share more or fewer layers of the decoders across tasks. We use the same learning rate of 0.1 and training for 100 epochs using a batch size of 256. We train with SGD with a momentum of 0.9 and a weight decay of $1e-4$. Following [54], all output prediction maps are rescaled to have zero mean and unit variance on the training set, and all dense tasks are trained with L1 loss.

2 Additional analysis results

2.1 Complete results and methodology for Figure 2

In Section 4, we perform MDL and MTL experiments on several pairs of datasets, each time comparing to the single dataset (SD) baseline trained for the same model capacity and training length. All results are run for three random seeds on CelebA and CIFAR+STL, and two random seeds for DomainNet and Taskonomy. To present these results in a condensed form in Figure 2, we first find the scalarization weights $\mathbf{p}^* = (p_1^*, 1-p_1^*)$ that yield the best average accuracy across both datasets. Then we report the difference in metrics between MDL trained with weights $\mathbf{p}^*$ and the corresponding SD baseline, for each dataset. Note that for the Taskonomy case, where the tasks are evaluated via L1 loss, we measure the negative difference instead to keep the same interpretation as the other settings where a positive value means MDL improves over SD.

For completeness, we report all results for the CIFAR/STL case as trade-off plots (accuracy on dataset 1 versus accuracy on dataset 2) in Figure B (CIFAR/STL) and in in Figure C (segmentation 2D and depth tasks of Taskonomy). We observe the same trends as summarized in the main paper: First, when increasing model capacity, the MDL performance over the SD baseline increases; This is best seen when width increases (across columns). Second, the optimal weights vary across model sizes: At low width, the best performance is reached for a ratio in the range of $[0.3, 0.4]$. While larger models prefer $p^*_{\text{STL}} \in [0.1, 0.3]$. Finally, it is interesting to note that these weights also differ from heuristics commonly used to set scalarization weights: such as uniform scalarization $p_{\text{STL}} = 0.5$ or for instance setting the weights to match the number of samples in each dataset $p_{\text{STL}} = 0.09$. This further highlights the fact that tuning scalarization weights can make scalarization into a stronger baseline for MDL/MTL.

2.2 MDL helps generalization

When comparing the MDL/MTL and SD performance, we often observe that MDL/MTL improvements over SD are visible at inference but not at training time (as illustrated for instance in Figure D). This indicates that MDL/MTL training helps generalization of the model compared to training on a single dataset. In particular, in the MDL setting, this draws an interesting parallel with data augmentations: In fact, MDL training can be seen as adding additional input data from a new distribution, with a probability given by the scalarization weights $\mathbf{p}$, while sharing the same semantic classes. And similarly to data augmentations, adding this extra data source makes the training distribution harder to fit (hence the SD baseline outperforming MDL at train time) but can greatly benefit generalization performance (hence the inverse trend at inference).

To push the analogy further, the experimental study of [14] suggests that a good data augmentation should be one with a high *affinity* to the original data distribution (i.e., the distribution shift between the original data and augmented one should not be too significant) as well as a high *diversity* (i.e., the added data should be complex enough, which can be measure e.g. with magnitude of the training loss). This hypothesis also matches our observations: For instance adding `infograph` data to the `real` domain leads to a low affinity pairing but with high diversity and yields weaker MDL performance on the `real` images compared to the SD baseline (see Figure 2).

Finally, we note that the problem of finding optimal scalarization weights mirrors the one of finding data augmentation hyperparameters, which has been extensively explored for the task of image

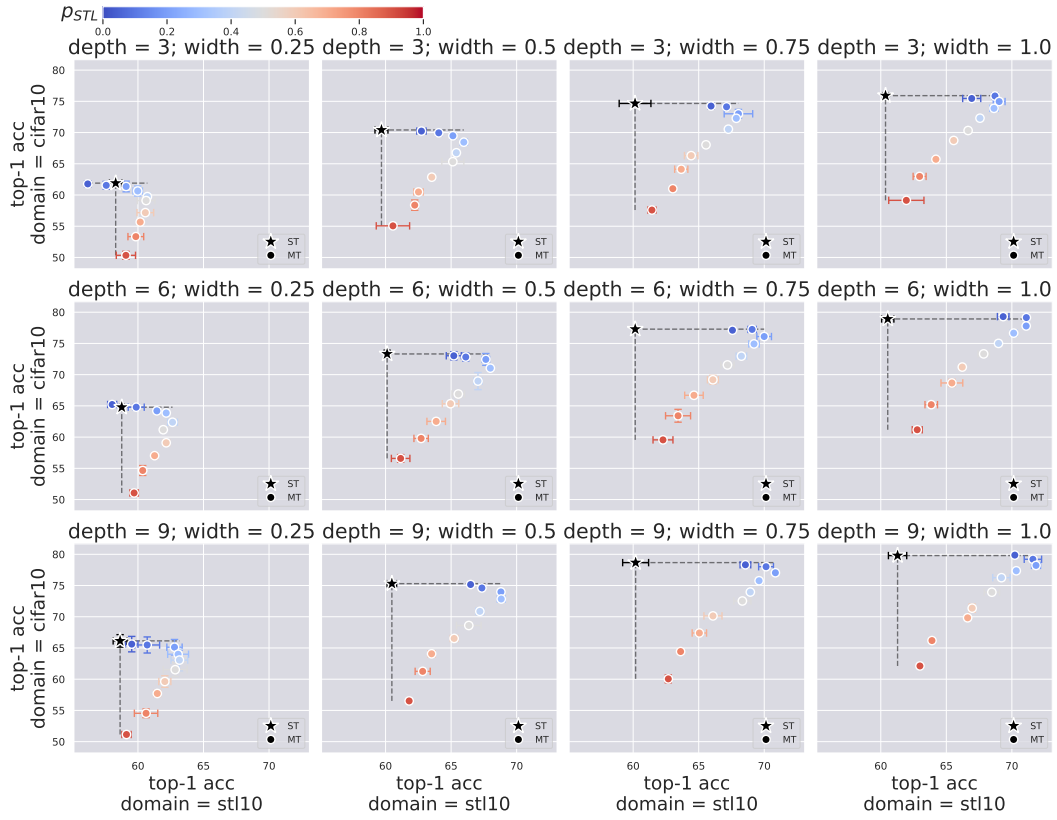


Figure B: Complete analysis results for the CIFAR+STL scenario. Each row corresponds to a different model depth and each column to a model width, in increasing order. In each plot, we plot the model's test accuracy on CIFAR-10 versus the test accuracy on STL-10. The single dataset baseline (SD) is drawn in black and corresponds to the accuracy obtained when training two separate networks, one on each dataset independently. The circle markers correspond to the MDL model trained for different scalarization weights $\mathbf{p}$. The value of p_{STL} is represented as the color of the marker, while the remaining weight is always set to $p_{\text{CIFAR}} = 1 - p_{\text{STL}}$.

classification. It has led to now commonly-adopted augmentation strategies such as RandAugment [3], AutoAugment [2], or PBA [20], which also uses Population-based training to tackle this problem.

3 Methodology for measuring gradients conflicts

In this appendix, we briefly describe our methodology for Section 4.2. We use the same definition of gradient conflicts as in PCGrad [60]: Two task/domain specific gradients are conflicting if and only if the cosine of the angle between them is strictly negative. We train a model using standard uniform scalarization, measure the number of conflicting pairs of task/domain gradients over one epoch of training, and report it as a percentage (of all pairs), for each epoch during training.

We report these results in the main text in Figure 3. Our main observation is that the presence or absence of gradient conflicts does not correlate well with actual MDL/MTL performance. This challenges the assumption underlying many multi-task optimization (MTO) methods [60, 36, 58] that reducing gradient conflicts leads to improved MTL performance. This also aligns well with recent results of [59, 27] showing that MTO methods that reduce gradient conflicts do not outperform simpler scalarization approaches in practice, and with the hypothesis of [27] that most MTO methods are in reality only adding a regularization effect.

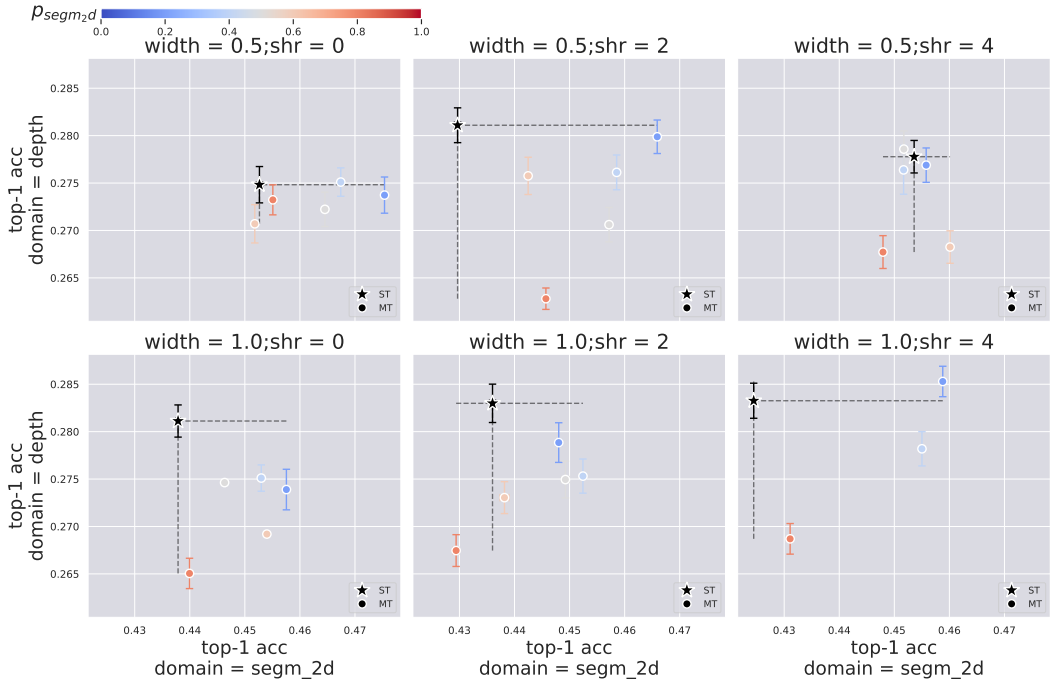


Figure C: Complete analysis results for the segmentation 2D and depth prediction tasks in the Taskonomy scenario. Each row corresponds to a different number of layers shared in the decoder (`shr`) and each column to a model width. In each plot, we plot the model's test Le loss of each task on either axis. The single dataset baseline (SD) is drawn in black and corresponds to the accuracy obtained when training two separate networks, one on each dataset independently. The circle markers correspond to the MDL model trained for different scalarization weights $\mathbf{p}$. The value of $p_{\text{segmentation2D}}$ is represented as the color of the marker.

4 Consistency of optimal scalarization weights

As noted in the analysis from Section 4, the optimal weights $\mathbf{p}^*$ is rather consistent across model depths and widths. For instance, on the CIFAR/STL case, $\mathbf{p}^*$ always falls in the range of [0.2, 0.4].

This can also be seen from the qualitative results of the population-based training search of scalarization weights in Section 5.2: While the history of hyperparameter changes during the search differ, PBT tends to converge to similar distribution for the scalarization weights $\mathbf{p}$ across different model depths and widths.

This suggests that the theoretical search space for $\mathbf{p}$ may be reduced in practice leading to a more computationally efficient search: Performing a rough initial search on a smaller model from the same architecture family can provide a promising range for $\mathbf{p}$, and can then be refined by searching with the larger target architecture.

5 PBT results

Because the search space for scalarization weights $\mathbf{p}$ grows exponentially with the number of tasks, classical hyperparameter search methods such as grid search or random search would struggle to scale as the number of tasks increases. Bayesian optimization (BO) [14] allows for faster results by browsing the search space in a smart way by building and following a probabilistic model of the hyperparameters. However, BO still requires training models to convergence (or until an early stopping criterion is met) which can be computationally expensive. Instead, we experiment with the Population-based Training framework [21] for searching for the optimal scalarization weights. PBT relies on the assumption that the "goodness" of a certain hyperparameter choice can be evaluated in a few epochs during training, rather than having to finish a full run of training.

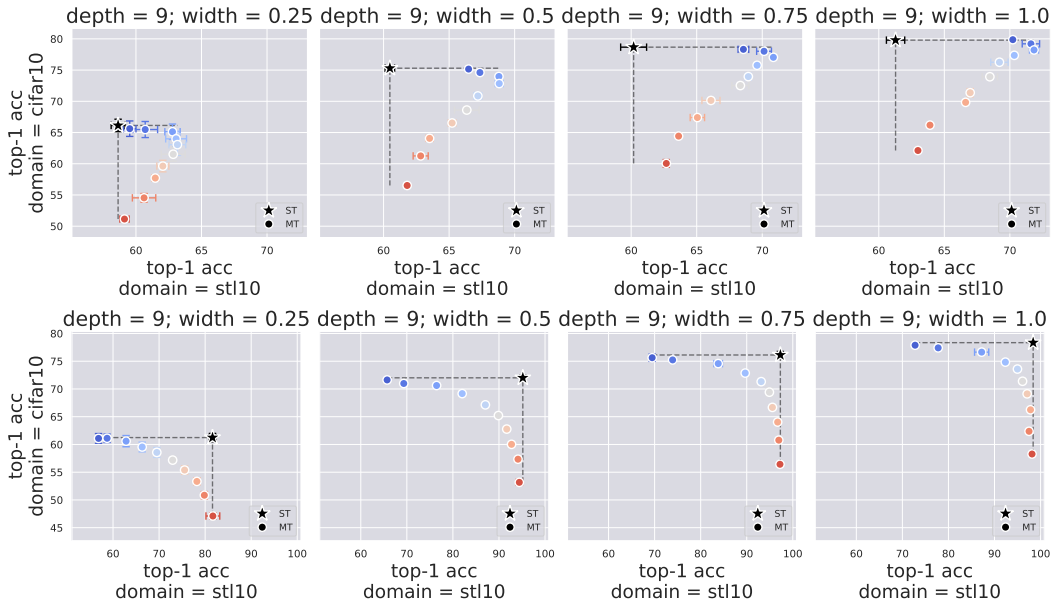


Figure D: Train-test Discrepancy when comparing MDL/MTL improvement over the SD baseline visualized on the CIFAR+STL example. In particular, in the MDL setting, this matches the classical interpretation of data augmentations: Adding additional semantically relevant data from an input distribution may be harder to fit at training time but leads to improved generalization performance at inference.

5.1 Compute resources

As mentioned in the main paper we perform all training runs on a single NVIDIA V100 machine with 32GB of memory. However, both the PBT and MTO models require higher compute resources than a single normal run of training, which we discuss below.

- **PBT** requires training a population of **N** models that are regularly synchronized; then followed by a final training run with the found optimal weights. Following previous works [20], we perform the hyperparameter search on a smaller subset of the data (in our experiments $r = 0.7$ fraction of the training set). In summary, the expected computational cost is roughly $Nr + 1$ times higher than standard training. On the CelebA example, we also observe that using PB2 [43], which combines the benefits of Bayesian Optimisation and PBT, yields better hyperparameters using a smaller population size. In terms of memory usage, PBT is the same as a standard training run: Synchronization is handled via checkpoints saving and loading, such that only one model lives in memory. Finally, we use the publicly available PBT implementation from Raytune[30] which handles all synchronization operations across the population. The implementation would also scale well to more compute resources, as the Ray API allows for easy parallelization.

- **MTO**. As shown in Figure 1, the bottleneck in most gradient-based methods is memory usage. Consequently, this requires us to decrease the batch size to meet memory requirements and compensate with gradient accumulation (or data parallelism if multiple devices are available). For instance, on the DomainNet experiments with all 6 domains, we need to decrease from a batch size of 512 to a batch size of 128 with 4 steps of accumulation to still fit in memory requirements. For some of these methods, this also raises the question of how to handle synchronization across batches: For instance, in PCGrad, the gradient conflicts (and projections) can be computed either **(i)** per local batch, before accumulation: this may lead to noisier updates; or **(ii)** after gradients are accumulated: However this is more memory-intensive as this requires to store the previously accumulated per-task gradient as well as the one being currently computed. In practice we use the implementation of [28] for all MTO methods.

In summary, when comparing different hyperparameter searches for scalarization, PBT allows for much faster exploration of the search space than classical techniques such as grid search.

Comparing PBT+scalarization with MTO is less straightforward as the computational cost depends on many factors (e.g. population size, number of tasks, and impact on memory usage, etc.), but generally, the "scalarization + hyperparameter search" approach is more favorable in case of low memory requirements as it does not change memory costs compared to standard training. However, soTA gradient-based methods are not very costly for a low number of tasks (e.g. 2-3) as shown in Table 1 which makes them appealing in settings with a few tasks. Nevertheless, one of our key takeaways is that allocating extra resources for tuning scalarization weights, to mirror the extra resources needed for MTO training, makes scalarization into a much stronger baseline, on-par or even outperforming MTO methods as shown in Section 5. Finally, another important difference is that hyperparameter search methods directly optimize for the target objective: the optimal hyperparameters are found by maximizing the average task/domain accuracy on a hold-out validation dataset. In contrast, MTO methods optimize for a proxy metric (such as reducing gradients conflicts) that may not always correlate with final performance as shown in Section 4.2.

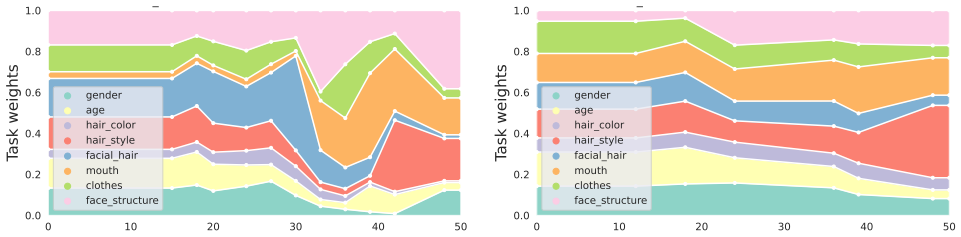
5.2 Qualitative results

In this section, we report some qualitative results of the hyperparameter scheduled found by PBT and PB2. At the end of PBT search, we select the model with the highest validation performance and backtrack its history to backtrack its choices of hyperparameter values during training: This yields the policy of optimal weights found by PBT which is then used to retrain a model on the full training set. We also experimented with retraining a model using the last weights of the policy, but this usually slightly underperform using the whole history of weights in the majority of cases.

In Figure E, we report examples of the policy of weights found by PBT and PB2 search for the parameters $E_{ready} = 3$, $Q = 40\%$ and $N = 12$ for PBT (respectively $N = 8$ for PB2).

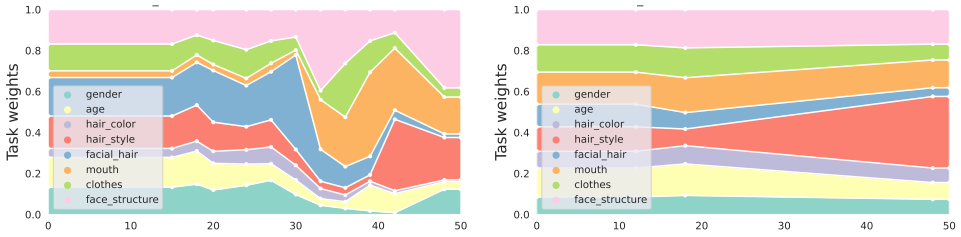
5.3 Complete quantitative results

Here we report additional results for the CelebA and DomainNet experiments of Section 5, in the style of Table 1 and Table 2 in the main text: We include results using a ResNet-50 on DomainNet in Table B, and results for additional widths in CelebA in Table C and Table D.



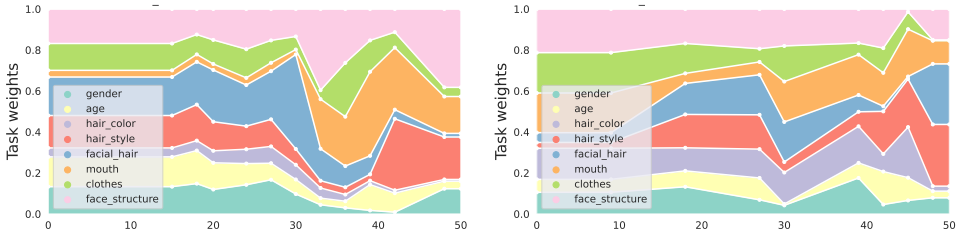
width = 1 width = 0.25

(a) Comparing Population-based training results for a depth of 9 layers with full width (left) and a smaller model with a quarter of the width (right). While both policy are quite different across training epochs, they converge towards similar distribution: For instance the weights for tasks "hair style", "gender" and "age" are significantly smaller than the one for the "mouth" and "hair style" tasks.



depth = 9 depth = 6

(b) Comparing Population-based training results for a depth of 9 layers with full width (left) and a smaller model with a depth of 6 layers (right). Similarly to the results on varying width in **(a)**, both search converge to similar distribution in task weights



Population-based Training (PBT) Population-based Bandit (PB2)

(c) Comparing Population-based training results for a depth of 9 layers with full width (left) and the same search with Population-based bandit (right). The two search algorithms converge to significantly different results in particular regarding weights for the "mouth" and "facial hair" tasks. This suggests that (i) there may be multiple good local minima in the search space of **p** and (ii) the heuristic used in the explore step has a significant impact on how the resulting policy.

Figure E: Qualitative results for Population-based training search on CelebA. The x-axis represents training epochs. The y-axis represents the policy scalarization weights for each task as a cumulative histogram for the run of the population with highest validation accuracy

Table B: Results of MDL when jointly training on all 6 domains of DomainNet for scalarization (uniform and PBT-found weights) and MTO methods with a **ResNet50** backbone. PBT is run with a population size of $N = 12$ models, such that every $E_{ready} = 5$ epochs, $Q = 25\%$ of the population triggers an exploit/explore.

DomainNet (ResNet50 with 0.25 width)	**average**	clipart	infograph	painting	quickdraw	real	sketch
Scalarization							
Uniform	49.69 ± 0.05	59.90 ± 0.15	22.45 ± 0.01	43.90 ± 0.14	63.13 ± 0.15	58.95 ± 0.09	49.79 ± 0.10
PBT	50.69 ± 0.10	61.69 ± 0.45	21.27 ± 0.10	44.72 ± 0.27	63.96 ± 0.13	62.43 ± 0.13	50.06 ± 0.17
MTO - Loss-based							
Uncertainty [23]	40.51 ± 0.19	53.33 ± 0.67	15.70 ± 0.02	34.44 ± 0.41	54.27 ± 0.61	47.86 ± 0.39	37.45 ± 0.37
IMTL-L [36]	37.04 ± 0.17	48.85 ± 0.59	13.93 ± 0.42	30.64 ± 0.36	51.60 ± 0.34	44.12 ± 0.21	33.12 ± 0.50
MTO - Gradient-based							
CAGrad [34]	39.82 ± 0.10	50.68 ± 0.05	16.94 ± 0.04	34.37 ± 0.35	52.16 ± 0.47	46.59 ± 0.00	38.20 ± 0.07
GradDrop [8]	39.18 ± 0.15	49.80 ± 0.04	16.77 ± 0.65	33.95 ± 0.52	51.18 ± 0.06	46.04 ± 0.28	37.36 ± 0.20
PCGrad [60]	39.48 ± 0.31	50.42 ± 0.97	16.83 ± 0.31	34.63 ± 0.92	51.14 ± 0.53	46.37 ± 0.51	37.49 ± 0.98

DomainNet (ResNet50 with original width)	**average**	clipart	infograph	painting	quickdraw	real	sketch
Scalarization							
Uniform	51.53 ± 0.06	61.89 ± 0.12	23.63 ± 0.01	45.87 ± 0.01	64.50 ± 0.08	61.46 ± 0.09	51.83 ± 0.33
PBT	51.83 ± 0.06	62.22 ± 0.06	22.61 ± 0.20	46.61 ± 0.29	64.71 ± 0.05	61.91 ± 0.05	52.93 ± 0.10
MTO - Loss-based							
Uncertainty [23]	42.90 ± 0.20	56.24 ± 0.44	17.36 ± 0.12	36.91 ± 0.82	56.11 ± 0.08	50.33 ± 0.48	40.49 ± 0.51
IMTL-L [36]	39.69 ± 0.13	52.51 ± 0.59	15.51 ± 0.14	33.45 ± 0.14	53.54 ± 0.05	46.37 ± 0.48	36.75 ± 0.17
MTO - Gradient-based							
CAGrad [34]	41.90 ± 0.13	53.32 ± 0.41	17.94 ± 0.42	36.72 ± 0.39	54.15 ± 0.03	48.31 ± 0.25	40.94 ± 0.14
GradDrop [8]	42.15 ± 0.14	53.38 ± 0.54	18.68 ± 0.33	37.00 ± 0.45	53.85 ± 0.11	49.04 ± 0.27	40.95 ± 0.04
PCGrad [60]	41.94 ± 0.20	53.46 ± 0.69	18.20 ± 0.28	36.95 ± 0.66	53.29 ± 0.13	48.88 ± 0.48	40.87 ± 0.49

Table C: (Table best seen zoomed in PDF) Results of MTL when training on all 8 tasks (subset of attributes) of CelebA for a depth of 6 layers For PBT and PB2 we use slightly different parameters than DomainNet to account for the fact that CelebA contains more tasks, and hence has a larger search space: All PBT runs use a population size of $N = 12$ models, such that every $E_{ready} = 3$ epochs, $Q = 40\%$ of the population triggers an exploit/explore step. For PB2 runs we use a population size of $N = 8$ and otherwise the same Q and E_{ready} hyperparameters.

ViT-S/4, 6 layers, 0.25 width	**average**	age	clothes	face structure	facial hair	gender	hair color	hair style	mouth
Scalarization									
Uniform	90.82 ± 2.1e-04	86.96 ± 8.5e-04	92.23 ± 6.8e-04	84.64 ± 6.6e-04	95.30 ± 1.8e-04	97.61 ± 7.1e-04	92.58 ± 3.4e-04	91.03 ± 5.8e-04	86.23 ± 4.4e-04
PBT	90.93 ± 1.8e-04	86.94 ± 7.6e-04	92.37 ± 3.3e-04	84.78 ± 4.5e-04	95.18 ± 2.1e-04	97.58 ± 5.3e-04	92.82 ± 4.3e-04	91.48 ± 3.5e-04	86.25 ± 7.0e-04
PB2	90.90 ± 2.0e-04	87.10 ± 1.2e-03	92.13 ± 3.2e-04	84.82 ± 5.6e-04	95.32 ± 2.8e-04	97.41 ± 4.8e-04	92.67 ± 5.6e-04	91.23 ± 2.3e-04	86.50 ± 2.1e-04
MTO - Loss-based									
Uncertainty [23]	90.82 ± 1.9e-04	86.94 ± 1.1e-03	92.22 ± 7.2e-04	84.63 ± 3.0e-04	95.32 ± 2.7e-05	97.62 ± 2.8e-04	92.57 ± 3.9e-04	91.02 ± 4.1e-04	86.22 ± 1.9e-04
IMTL-L [36]	90.82 ± 2.0e-04	86.94 ± 1.2e-03	92.22 ± 7.3e-04	84.63 ± 3.1e-04	95.32 ± 2.7e-05	97.62 ± 3.2e-04	92.57 ± 3.8e-04	91.02 ± 4.1e-04	86.22 ± 1.6e-04
MTO - Gradient-based									
CAGrad [34]	90.92 ± 2.7e-04	86.96 ± 2.1e-03	92.35 ± 3.0e-05	84.92 ± 1.2e-04	95.38 ± 3.9e-04	97.56 ± 1.4e-04	92.73 ± 3.8e-04	91.30 ± 2.2e-04	86.14 ± 8.9e-05
GradDrop [8]	90.65 ± 2.9e-04	86.76 ± 1.7e-03	92.03 ± 8.1e-05	84.48 ± 1.2e-04	95.23 ± 3.9e-04	97.41 ± 5.7e-04	92.46 ± 2.1e-04	90.83 ± 1.4e-03	85.98 ± 3.5e-05
PCGrad [60]	90.86 ± 1.5e-04	87.04 ± 1.1e-04	92.32 ± 7.3e-04	84.65 ± 4.0e-04	95.27 ± 1.9e-04	97.62 ± 3.2e-04	92.64 ± 3.7e-04	91.16 ± 2.8e-04	86.22 ± 5.9e-04

ViT-S/4, 6 layers, 0.5 width	**average**	age	clothes	face structure	facial hair	gender	hair color	hair style	mouth
Scalarization									
Uniform	91.28 ± 2.8e-04	87.33 ± 1.9e-03	92.71 ± 8.6e-05	85.15 ± 2.8e-04	95.49 ± 2.2e-04	98.03 ± 6.0e-04	92.99 ± 3.2e-04	91.63 ± 5.8e-04	86.87 ± 3.8e-04
PBT	91.29 ± 1.7e-04	87.56 ± 3.3e-04	92.77 ± 1.6e-04	85.34 ± 2.5e-04	95.40 ± 2.9e-04	97.81 ± 6.4e-04	92.99 ± 2.1e-04	91.56 ± 7.2e-04	86.89 ± 7.6e-04
PB2	91.38 ± 2.3e-04	87.71 ± 1.1e-03	92.84 ± 3.3e-04	85.20 ± 6.0e-04	95.54 ± 4.5e-04	98.00 ± 7.3e-04	92.95 ± 6.2e-04	91.73 ± 2.2e-04	87.05 ± 7.0e-04
MTO - Loss-based									
Uncertainty [23]	91.30 ± 3.1e-04	87.52 ± 2.2e-03	92.72 ± 1.4e-04	85.14 ± 4.6e-04	95.48 ± 4.7e-04	98.08 ± 5.3e-04	92.97 ± 3.5e-04	91.65 ± 1.9e-04	86.86 ± 6.4e-04
IMTL-L [36]	91.30 ± 2.6e-04	87.49 ± 1.4e-03	92.74 ± 6.6e-05	85.14 ± 1.0e-04	95.51 ± 1.3e-04	98.06 ± 7.8e-04	92.98 ± 5.8e-04	91.66 ± 6.7e-04	86.85 ± 1.0e-03
MTO - Gradient-based									
CAGrad [34]	91.32 ± 3.2e-04	87.47 ± 2.3e-03	92.86 ± 4.9e-04	85.26 ± 3.1e-04	95.51 ± 4.3e-04	98.03 ± 0.0e+00	93.00 ± 5.9e-04	91.73 ± 1.5e-04	86.72 ± 2.3e-04
GradDrop [8]	91.19 ± 2.3e-04	87.42 ± 8.5e-04	92.65 ± 5.2e-04	85.03 ± 9.5e-04	95.43 ± 2.9e-04	97.96 ± 6.7e-04	92.80 ± 1.8e-04	91.54 ± 8.5e-04	86.70 ± 3.0e-04
PCGrad [60]	91.30 ± 4.6e-04	87.45 ± 2.8e-03	92.72 ± 6.0e-04	85.16 ± 8.9e-04	95.54 ± 5.0e-04	98.08 ± 2.1e-03	92.97 ± 1.6e-04	91.63 ± 1.2e-04	86.82 ± 3.8e-04

ViT-S/4, 6 layers, full width	**average**	age	clothes	face structure	facial hair	gender	hair color	hair style	mouth
Scalarization									
Uniform	91.23 ± 3.6e-04	86.70 ± 2.7e-03	92.79 ± 4.5e-04	85.16 ± 4.8e-04	95.45 ± 4.8e-04	98.10 ± 2.1e-04	92.99 ± 3.6e-04	91.68 ± 4.7e-05	86.95 ± 4.8e-04
PBT	91.21 ± 2.7e-04	86.79 ± 1.1e-03	92.77 ± 5.8e-04	85.18 ± 3.6e-04	95.42 ± 8.9e-04	98.02 ± 4.5e-04	92.92 ± 6.4e-04	91.66 ± 1.0e-03	86.95 ± 6.7e-04
PB2	91.28 ± 2.5e-04	86.96 ± 1.5e-03	92.87 ± 2.7e-04	85.17 ± 2.4e-04	95.47 ± 6.5e-05	98.10 ± 3.9e-04	92.90 ± 5.8e-04	91.81 ± 2.6e-04	86.94 ± 1.1e-03
MTO - Loss-based									
Uncertainty [23]	91.22 ± 2.1e-04	86.60 ± 1.3e-03	92.85 ± 6.4e-04	85.24 ± 1.7e-04	95.43 ± 8.0e-05	98.07 ± 3.5e-04	92.91 ± 7.7e-04	91.72 ± 2.1e-04	86.93 ± 4.4e-05
IMTL-L [36]	91.21 ± 2.0e-04	86.65 ± 7.1e-04	92.83 ± 5.2e-04	85.17 ± 1.8e-04	95.42 ± 5.5e-04	98.01 ± 6.0e-04	92.99 ± 8.9e-04	91.70 ± 6.4e-04	86.93 ± 1.1e-04
MTO - Gradient-based									
CAGrad [34]	91.25 ± 2.2e-04	86.79 ± 1.7e-03	92.88 ± 7.1e-05	85.12 ± 1.4e-04	95.46 ± 4.6e-04	98.17 ± 1.1e-04	92.91 ± 2.1e-04	91.74 ± 3.6e-05	86.88 ± 1.5e-04
GradDrop [8]	91.29 ± 2.3e-04	87.01 ± 5.0e-04	92.80 ± 1.0e-03	85.22 ± 8.0e-04	95.50 ± 2.7e-04	98.15 ± 6.0e-04	93.00 ± 9.1e-04	91.65 ± 1.5e-04	86.97 ± 4.0e-04
PCGrad [60]	91.21 ± 3.9e-04	86.55 ± 2.3e-03	92.81 ± 9.3e-04	85.15 ± 8.4e-04	95.44 ± 9.3e-04	98.16 ± 5.0e-04	92.92 ± 2.8e-04	91.75 ± 1.3e-03	86.87 ± 4.8e-04

Table D: (Table best seen zoomed in PDF) Results of MTL when training on all 8 tasks (subset of attributes) of CelebA for a depth of 9 layers. For PBT and PB2 we use slightly different parameters than DomainNet to account for the fact that CelebA contains more tasks, and hence has a larger search space: All PBT runs use a population size of $N = 12$ models, such that every $E_{ready} = 3$ epochs, $Q = 40\%$ of the population triggers an exploit/explore step. For PB2 runs we use a population size of $N = 8$ and otherwise the same Q and E_{ready} hyperparameters.

ViT-S/4, 9 layers, 0.25 width	**average**	age	clothes	face structure	facial hair	gender	hair color	hair style	mouth
Scalarization									
Uniform	91.00 ± 2.5e-04	87.24 ± 1.6e-03	92.40 ± 7.6e-05	84.85 ± 4.5e-04	95.34 ± 5.7e-04	97.77 ± 5.0e-04	92.68 ± 3.5e-04	91.31 ± 5.8e-04	86.40 ± 8.0e-05
PBT	90.97 ± 1.7e-04	86.96 ± 1.1e-03	92.34 ± 2.5e-04	84.91 ± 8.4e-04	95.30 ± 8.1e-05	97.76 ± 2.9e-05	92.55 ± 1.1e-04	91.35 ± 6.3e-05	86.59 ± 2.1e-04
PB2	91.04 ± 2.7e-04	87.22 ± 1.9e-03	92.40 ± 2.5e-04	84.96 ± 4.2e-04	95.35 ± 3.5e-04	97.70 ± 4.0e-04	92.67 ± 1.4e-04	91.36 ± 3.1e-04	86.67 ± 4.8e-04
MTO - Loss-based									
Uncertainty [23]	91.01 ± 2.3e-04	87.18 ± 1.4e-03	92.40 ± 4.5e-04	84.87 ± 3.5e-04	95.34 ± 6.6e-04	97.81 ± 5.7e-04	92.70 ± 3.9e-04	91.30 ± 4.5e-04	86.44 ± 1.1e-04
IMTL-L [36]	91.00 ± 2.4e-04	87.18 ± 1.5e-03	92.40 ± 4.6e-04	84.87 ± 4.0e-04	95.34 ± 6.6e-04	97.81 ± 5.0e-04	92.70 ± 3.3e-04	91.30 ± 4.7e-04	86.44 ± 6.2e-05
MTO - Gradient-based									
CAGrad [34]	91.07 ± 1.3e-04	87.19 ± 4.3e-04	92.55 ± 3.5e-04	85.03 ± 4.8e-04	95.41 ± 2.7e-05	97.79 ± 6.0e-04	92.82 ± 1.8e-04	91.52 ± 3.5e-04	86.23 ± 2.3e-04
GradDrop [8]	90.83 ± 1.8e-04	86.91 ± 4.6e-04	92.23 ± 1.4e-04	84.66 ± 1.0e-03	95.26 ± 6.1e-04	97.66 ± 3.2e-04	92.58 ± 3.9e-04	91.14 ± 2.8e-04	86.18 ± 4.6e-04
PCGrad [60]	90.95 ± 2.4e-04	87.13 ± 1.2e-03	92.29 ± 4.2e-04	84.77 ± 4.7e-04	95.32 ± 3.6e-04	97.72 ± 3.5e-04	92.70 ± 8.0e-05	91.21 ± 1.6e-04	86.44 ± 1.3e-03

ViT-S/4, 9 layers, 0.5 width	**average**	age	clothes	face structure	facial hair	gender	hair color	hair style	mouth
Scalarization									
Uniform	91.32 ± 3.2e-04	87.26 ± 2.3e-03	92.81 ± 2.4e-04	85.19 ± 9.4e-05	95.51 ± 3.4e-04	98.15 ± 7.1e-04	93.03 ± 8.9e-06	91.66 ± 4.0e-04	86.93 ± 7.3e-04
PBT	91.36 ± 2.3e-04	87.45 ± 1.5e-03	92.86 ± 6.5e-05	85.31 ± 5.2e-04	95.50 ± 2.8e-04	97.98 ± 2.2e-04	93.03 ± 2.4e-04	91.70 ± 5.4e-04	87.06 ± 5.4e-04
PB2	91.36 ± 1.4e-04	87.45 ± 7.8e-04	92.82 ± 2.7e-04	85.17 ± 3.1e-04	95.55 ± 4.4e-04	98.14 ± 3.4e-04	93.06 ± 1.2e-04	91.76 ± 4.3e-04	86.89 ± 1.7e-04
MTO - Loss-based									
Uncertainty [23]	91.33 ± 1.4e-04	87.32 ± 4.3e-04	92.84 ± 2.0e-05	85.19 ± 2.1e-04	95.52 ± 4.9e-04	98.15 ± 7.8e-04	92.96 ± 2.0e-04	91.69 ± 6.5e-05	86.93 ± 2.5e-04
IMTL-L [36]	91.29 ± 1.5e-04	87.25 ± 5.0e-04	92.79 ± 2.0e-04	85.17 ± 7.1e-05	95.50 ± 4.9e-04	98.10 ± 8.5e-04	92.97 ± 1.4e-04	91.66 ± 4.8e-04	86.92 ± 9.7e-05
MTO - Gradient-based									
CAGrad [34]	91.37 ± 1.8e-04	87.42 ± 2.1e-04	92.90 ± 4.1e-05	85.24 ± 1.3e-03	95.53 ± 2.3e-04	98.20 ± 1.1e-04	92.98 ± 2.6e-04	91.78 ± 2.7e-04	86.88 ± 3.4e-04
GradDrop [8]	91.27 ± 1.8e-04	87.50 ± 7.1e-05	92.71 ± 3.5e-05	85.10 ± 1.1e-04	95.46 ± 6.2e-05	98.04 ± 1.3e-03	92.94 ± 4.3e-04	91.57 ± 3.2e-04	86.85 ± 1.9e-04
PCGrad [60]	91.29 ± 1.6e-04	87.10 ± 3.2e-04	92.83 ± 4.3e-04	85.18 ± 6.1e-04	95.47 ± 1.1e-04	98.11 ± 6.4e-04	93.01 ± 8.9e-05	91.67 ± 6.3e-04	86.96 ± 2.6e-04

ViT-S/4, 9 layers, full width	**average**	age	clothes	face structure	facial hair	gender	hair color	hair style	mouth
Scalarization									
Uniform	91.17 ± 1.7e-04	87.33 ± 5.7e-04	92.50 ± 8.1e-04	85.10 ± 4.2e-04	95.45 ± 1.9e-04	97.93 ± 1.1e-04	92.85 ± 2.9e-04	91.39 ± 1.2e-04	86.80 ± 6.9e-04
PBT	91.15 ± 2.7e-04	87.43 ± 3.1e-04	92.51 ± 3.6e-04	85.18 ± 1.1e-03	95.46 ± 3.1e-04	97.78 ± 1.5e-03	92.51 ± 1.4e-04	91.36 ± 8.0e-04	87.00 ± 5.8e-04
PB2	91.25 ± 2.3e-04	86.83 ± 1.2e-03	92.85 ± 4.8e-04	85.12 ± 7.3e-04	95.49 ± 5.6e-04	98.19 ± 6.0e-04	92.90 ± 5.6e-04	91.78 ± 4.5e-04	86.81 ± 3.7e-04
MTO - Loss-based									
Uncertainty [23]	91.17 ± 1.8e-04	87.36 ± 6.4e-04	92.50 ± 8.0e-04	85.11 ± 3.0e-04	95.45 ± 1.6e-04	97.93 ± 1.4e-04	92.84 ± 3.8e-04	91.39 ± 5.3e-05	86.81 ± 8.3e-04
IMTL-L [36]	91.18 ± 1.6e-04	87.37 ± 4.3e-04	92.50 ± 8.0e-04	85.11 ± 2.8e-04	95.45 ± 1.7e-04	97.94 ± 2.1e-04	92.84 ± 3.7e-04	91.39 ± 6.5e-05	86.81 ± 7.8e-04
MTO - Gradient-based									
CAGrad [34]	91.21 ± 1.4e-04	87.34 ± 4.3e-04	92.64 ± 3.7e-04	85.16 ± 4.0e-04	95.43 ± 4.3e-04	97.91 ± 1.1e-04	92.89 ± 4.3e-04	91.58 ± 3.6e-04	86.70 ± 5.5e-04
GradDrop [8]	91.20 ± 1.4e-04	86.55 ± 2.1e-04	92.81 ± 2.6e-04	85.19 ± 4.3e-04	95.40 ± 8.1e-04	98.09 ± 4.6e-04	92.91 ± 2.6e-04	91.71 ± 3.4e-04	86.90 ± 8.9e-06
PCGrad [60]	91.13 ± 3.5e-04	87.35 ± 2.4e-03	92.50 ± 1.1e-03	85.04 ± 5.3e-04	95.40 ± 3.5e-04	97.87 ± 0.0e+00	92.82 ± 2.8e-04	91.37 ± 1.2e-04	86.71 ± 5.0e-04

Appendix References

[9] Adam Coates, A. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. 4

[2] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–123, 2019.

[3] Ekin Dogus Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3008–3017, 2019.

[13] Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2

[14] P. Frazier. A tutorial on bayesian optimization. *ArXiv*, abs/1807.02811, 2018. 8

[15] Hanan Gani, Muzammal Naseer, and Mohammad Yaqub. How to train vision transformer on small-scale datasets? In *British Machine Vision Conference (BMVC)*, 2022. 4

[20] Daniel Ho, Eric Liang, Ion Stoica, P. Abbeel, and Xi Chen. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, 2019. 7

[21] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017. 2, 7

[26] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 4

[27] Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and M. Pawan Kumar. In defense of the unitary scalarization for deep multi-task learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 1, 2, 4, 6

[29] Yunsheng Li, Lu Yuan, Yinpeng Chen, Pei Wang, and Nuno Vasconcelos. Dynamic transfer for multi-source domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10998–11007, 2021. 4

[30] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *ArXiv*, abs/1807.05118, 2018. 8

[36] Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *International Conference on Learning Representations (ICLR)*, 2021. 2, 3

[14] Raphael Gontijo Lopes, Sylvia J. Smullin, Ekin Dogus Cubuk, and Ethan Dyer. Tradeoffs in data augmentation: An empirical study. In *International Conference on Learning Representations (ICLR)*, 2021.

[44] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *International Conference on Computer Vision (ICCV)*, pages 1406–1415, 2018. 4

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[54] Trevor Scott Standley, Amir Roshan Zamir, Dawn Chen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? *ArXiv*, abs/1905.07553, 2019. 1, 2, 4, 5

[58] Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations (ICLR)*, 2021. 2

[59] Derrick Xin, B. Ghorbani, Ankush Garg, Orhan Firat, and Justin Gilmer. Do current multi-task optimization methods in deep learning even help? In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 1, 2, 4, 6, 9

[60] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2, 3, 5

[61] Amir Roshan Zamir, Alexander Sax, Bokui (William) Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 4, 5