

SIMPLIFYING MULTI-TASK ARCHITECTURES THROUGH TASK-SPECIFIC NORMALIZATION

Anonymous authors

Paper under double-blind review

ABSTRACT

Multi-task learning (MTL) aims to leverage shared knowledge across tasks to improve generalization and parameter efficiency, yet balancing resources and mitigating interference remain open challenges. Architectural solutions often introduce elaborate task-specific modules or routing schemes, increasing complexity and overhead. In this work, we show that normalization layers alone are sufficient to address many of these challenges. Simply replacing shared normalization with task-specific variants already yields competitive performance, questioning the need for complex designs. Building on this insight, we propose Task-Specific Sigmoid Batch Normalization ($TS\sigma BN$), a lightweight mechanism that enables tasks to softly allocate network capacity while fully sharing feature extractors. $TS\sigma BN$ improves stability across CNNs and Transformers, matching or exceeding performance on NYUv2, Cityscapes, CelebA, and PascalContext, while remaining highly parameter-efficient. Moreover, its learned gates provide a natural framework for analyzing MTL dynamics, offering interpretable insights into capacity allocation, filter specialization, and task relationships. Our findings suggest that complex MTL architectures may be unnecessary and that task-specific normalization offers a simple, interpretable, and efficient alternative.

1 INTRODUCTION

Multi-task learning (MTL) trains a single model to solve multiple tasks jointly, leveraging shared representations to improve generalization and computational efficiency. Despite many successes, MTL remains difficult to understand and control. Core challenges include task interference, where competing gradients from divergent task requirements disrupt joint training (Zhang et al., 2022); capacity allocation, where shared and task-specific resources must be balanced to avoid dominance (Maziarz et al., 2019; Newell et al., 2019); and task similarity, where the degree of relatedness determines how tasks should interact (Standley et al., 2020). Existing approaches typically address only one of these issues. Optimization-based methods focus on mitigating interference by reweighting losses or modifying gradients (Yu et al., 2020; Navon et al., 2022). Soft-sharing architectures attempt to disentangle capacity by adding task-specific modules on top of a shared backbone, but in doing so often introduce significant design complexity in deciding how modules should interact (Misra et al., 2016; Liu et al., 2019). Neural architecture search methods learn to partition networks based on data-driven estimates of task-relatedness (Guo et al., 2020; Sun et al., 2020).

In this work, we argue that normalization layers and in particular batch normalization (BN) (Ioffe, 2015) are a sufficient and highly effective solution for all the aforementioned challenges in MTL. Our motivation stems from the following observations:

First, while neural networks are heavily over-parameterized, existing approaches struggle to resolve tasks conflicts (Shi et al., 2023), indicating a failure to utilize the available network capacity optimally. Second, BN has proven to be highly expressive - not only does it stabilize and accelerate training (Santurkar et al., 2018; Bjorck et al., 2018), but it also demonstrates remarkable standalone performance when used on random feature extractors (Rosenfeld & Tsotsos, 2019; Frankle et al., 2021) and its ability to leverage features not explicitly optimized for a specific task (Zhao et al., 2024). Third, BN can learn to ignore unimportant features (Frankle et al., 2021) or be explicitly regularized to produce structured sparsity (Liu et al., 2017; Suteu & Guo, 2022). This can be leveraged for MTL when unrelated tasks cannot fully share all features without interference and require disentanglement. Fourth, normalization layers are extremely parameter-efficient, taking up typically less than 0.5% of a

054 model’s size. This makes them particularly suitable as lightweight universal adapters for applications
 055 where models need to scale to multiple tasks (Rebuffi et al., 2017; Bilen & Vedaldi, 2017).
 056

057 Lastly, while conditional BN layers have been explored in settings with domain shift (Wallingford
 058 et al., 2022; Xie et al., 2023; Chang et al., 2019; Deng et al., 2023), these methods focus on the issue
 059 of mismatched normalization statistics and use task-specific BN as a domain-alignment tool. Our
 060 focus is different: we study single-domain MTL, where all tasks share the same input distribution
 061 and normalization does not become a failure mode. In this setting, we show that task-specific BN
 062 can provide a simple way to modulate representations via their affine parameters - turning it from a
 063 normalization module into a lightweight mechanism for capacity allocation and interference reduction.
 064 The extension of BN as the sole mechanism for modulation and interpretability rather than domain
 alignment remains largely unexplored.

065 Motivated by these observations, we propose a minimalist soft-sharing approach to MTL, where
 066 feature extractors are fully shared and only normalization layers are task-specific. Unlike prior soft-
 067 sharing architectures that add complex modules or routing schemes, our design isolates normalization
 068 as the sole mechanism for balancing tasks. Building on σ BN (Suteu & Guo, 2022), we introduce
 069 lightweight task-specific gates that modulate feature usage with negligible overhead, making the
 070 approach broadly compatible, easy to implement, and resilient to task imbalance. Beyond performance
 071 and efficiency, the learned σ BN parameters naturally form a task-filter importance matrix, enabling a
 072 structured analysis of capacity allocation, filter specialization, and task relationships, providing an
 073 interpretable view of MTL that is largely absent in prior work.

074 **Contributions:**

- 075
- 076 • A minimal MTL baseline. We show that simply replacing shared normalization with
 077 task-specific BatchNorm (TSBN) already delivers competitive performance out-of-the-box,
 078 questioning the necessity of elaborate task-specific modules or routing schemes.
- 079 • An extended design with sigmoid normalization. We introduce TS σ BN which improves sta-
 080 bility and scale across CNNs and transformers. This variant achieves superior performance
 081 on nearly all benchmarks while remaining parameter-efficient.
- 082 • An interpretable analysis framework. The use of σ BN further provides a natural lens for
 083 analyzing MTL dynamics. By interpreting learned feature importances, we obtain structured
 084 insights into capacity allocation, filter specialization, and task relationships.
 085

086 2 RELATED WORK

087

088 **Soft parameter sharing** methods tackle MTL interference architecturally by introducing task-specific
 089 modules to a shared backbone. Design options include replicating backbones (Misra et al., 2016;
 090 Ruder et al., 2019), adding attention mechanisms (Liu et al., 2019; Maninis et al., 2019), low-rank
 091 adaptation modules (Liu et al., 2022b; Agiza et al., 2024) or allowing cross-talk at a decoder level
 092 (Xu et al., 2018; Vandenhende et al., 2020b). However, these methods rely on task-specific feature
 093 extractors to avoid negative transfer at the cost of forgoing the multi-task inductive bias. Furthermore,
 094 adding task-specific capacity scales poorly with many tasks (Strezoski et al., 2019), and requires
 095 extensive code modifications that hinder adaptation to new architectures. Although BatchNorm is
 096 present in many of these systems, it is embedded in larger task-specific designs. In contrast, our
 097 method isolates BatchNorm as the sole soft-sharing mechanism, showing that it is a sufficient solution
 098 for competitive MTL while challenging unnecessary complexity.

099 **Neural Architecture Search (NAS)** methods reduce task interference by choosing which parameters
 100 to share among tasks as hard-partitioned sub-networks. Some approaches use probabilistic sampling
 101 (Sun et al., 2020; Bragman et al., 2019; Maziarz et al., 2019; Newell et al., 2019) or explicit
 102 branching/grouping strategies based on task affinities (Vandenhende et al., 2020a; Guo et al., 2020;
 103 Bruggemann et al., 2020; Standley et al., 2020; Fifty et al., 2021). Others use hypernetworks
 104 (Raychaudhuri et al., 2022; Aich et al., 2023) which learn to generate MTL architectures conditioned
 105 on user preferences. While our method also models task relationships and capacity allocation, it does
 106 so without architecture search, relying solely on static modulation via normalization layers.
 107

Mixture-of-Experts (MoE) methods address task interference by dynamically routing inputs to specialized experts, enabling flexible capacity allocation among tasks (Ma et al., 2018; Hazimeh et al., 2021; Tang et al., 2020). More recent work extends MoE designs to large-scale transformer architectures for vision and language tasks (Fan et al., 2022; Chen et al., 2023; Ye & Xu, 2023; Yang et al., 2024). Although effective, these methods rely on dynamic, per-sample routing that increases architectural and training complexity. In contrast, our approach provides a static and lightweight form of soft partitioning, achieving similar benefits with minimal changes to the wrapped backbone.

Parameter-efficient fine-tuning (PEFT) is a popular approach for adapting large pre-trained models without updating the full backbone. Single-task PEFT methods such as Adapters (He et al., 2021), BitFit (Zaken et al., 2022), VPT (Jia et al., 2022), Compacter (Karimi Mahabadi et al., 2021), and LoRA-style updates add small task-specific modules or low-rank layers while keeping most weights frozen. Extending these ideas to MTL requires managing several task-specific adapters at once. Recent PEFT-MTL methods address this by generating adapter weights through hypernetworks or decompositions, as in HyperFormer (Mahabadi et al., 2021), Polyhistor (Liu et al., 2022b), and MTLORA (Agiza et al., 2024). However, these methods still rely on additional task-specific capacity, which parallels traditional soft-parameter sharing and scales poorly with the number of tasks. In contrast, we modulate the shared capacity directly through BN, without adding new feature extractors.

Domain-specific normalization has become a common technique in settings with domain shift, where shared BatchNorm fails because domains have different input distributions. In these cases, separate BN statistics or layers are required to maintain stable normalization (Li et al., 2016; Zajac et al., 2019; Chang et al., 2019). The same motivation appears in several areas: In meta-learning, TaskNorm (Bronskill et al., 2020) adapt BN statistics per episode to handle changes in input distribution. In continual learning, CLBN (Xie et al., 2023) store task-specific BN parameters to avoid catastrophic forgetting from normalization drift. In conditional or multi-modal models, BN and LayerNorm is adjusted to match modality-specific statistics (Michalski et al., 2019; Zhao et al., 2024). In multi-domain MTL (Bilen & Vedaldi, 2017; Mudrakarta et al., 2019; Wallingford et al., 2022; Deng et al., 2023), task-specific BN is used as an adapter for tasks from different domains. In contrast, our work targets single-domain MTL, where all tasks share the same input and normalization does not fail. In this case, task-specific BN is not needed for statistical correction. Instead, we focus on its affine parameters as a basis for task-specific feature modulation, and extend this idea with a reparameterization and optimization scheme tailored to reduce interference and allocate capacity.

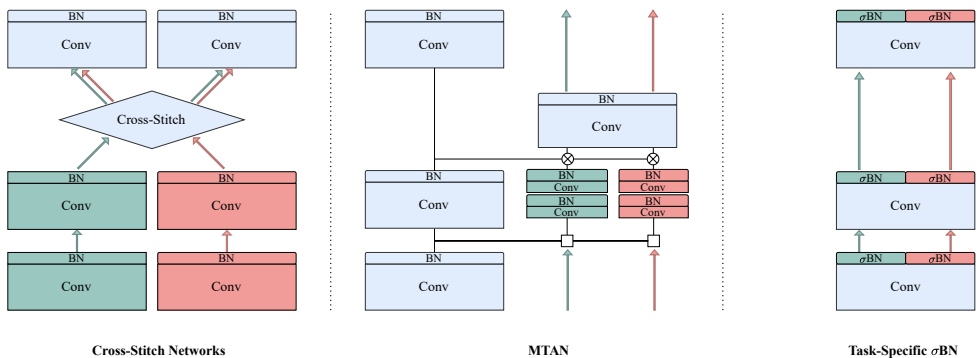


Figure 1: Illustration of soft parameter sharing architectures in a two-task setting. Cross-Stitch Networks (Misra et al., 2016) and MTAN (Liu et al., 2019) incorporate additional feature extractors, which lead to scalability challenges as the number of tasks increases. Task-Specific σ BN Networks introduce only task-specific normalization layers, offering a highly parameter-efficient solution.

3 BATCHNORM AND σ BATCHNORM

Batch normalization is a cornerstone for deep CNNs due to its versatility, efficiency, and wide-ranging benefits, including improved training stability for faster convergence (Santurkar et al., 2018; Bjorck et al., 2018), regularization effects (Luo et al., 2019), and the orthogonalization of representations (Daneshmand et al., 2021). BN operates in two key steps - normalization and affine transformation:

$$BN(x; \gamma, \beta) = \gamma \hat{x} + \beta, \quad \hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (1)$$

The normalization step standardizes input activations using the mini-batch mean μ_B and variance σ_B^2 , while the affine transformation applies channel-specific learnable parameters, γ and β , to re-scale and shift the normalized activations. During inference, BN relies on population statistics collected during training via running estimates. When the test distribution differs from the training set, these statistics can become mismatched and significantly degrade model performance (Summers & Dinneen, 2020). Because of this, many BN variants aim to improve the normalization step itself by adjusting μ and σ to handle distribution changes, domain shift, meta-learning episodes, or multi-modal inputs. For a survey on normalization approaches we refer to Huang et al. (2023).

In single-domain MTL, all tasks share the same input distribution, so the normalization component of BN does not need adjustment. Instead, we focus on the affine transformation post-normalization. These parameters represent only a small fraction of the network, yet they have substantial expressive power, as shown by studies demonstrating high performance when training BN alone (Frankle et al., 2021). In this work, we build on a variation of BN originally introduced to determine feature importance in structured pruning, Sigmoid Batch Normalization (Suteu & Guo, 2022) replaces the affine transformation with a single bounded scaler:

$$\sigma BN(x; \gamma) = \sigma(\gamma) \hat{x}, \quad \sigma(\gamma) = \frac{1}{1 + e^{-\gamma}} \quad (2)$$

Using a single bounded scaler per feature has little impact on performance, but enables targeted regularization and improves interpretability. These properties make σ BN especially attractive for multi-task learning, where understanding how tasks share limited capacity is critical. In this setting, $\sigma(\gamma)$ acts as a static soft gate that can down-weight or disable features. This implicit static gating contrasts with soft-sharing models, which explicitly partition capacity, and MoE methods, which route features dynamically through task-specific gates. Furthermore, this formulation can be extended to other normalization layers (Ba et al., 2016), as we show in experiments on transformers. Using σ BN as the only task-specific components, we create a parameter-efficient framework that sustains performance while providing tools to analyze and influence capacity allocation and task relationships.

4 TASK-SPECIFIC σ BATCHNORM NETWORKS

TS σ BN networks are constructed by replacing every shared Batch Normalization layer with task-specific σ BN layers, as illustrated in Figure 1. This design allows tasks to normalize and modulate the outputs of shared convolutional layers:

$$TS\sigma BN(x; \gamma_t) = \sigma(\gamma_t) \hat{x}, \quad \hat{x} = \frac{x - \mu_{B,t}}{\sqrt{(\sigma_{B,t})^2 + \epsilon}} \quad (3)$$

enabling better disentanglement of representations and reduced task interference. Unlike prior methods introducing additional task-specific capacity, TS σ BN keeps all convolutions shared, preserving the multi-task learning inductive bias toward generalizable representations. While domain-specific BN has been used reactively in domain adaptation (Chang et al., 2019) to handle distribution shifts, our work is the first to use it proactively as a standalone mechanism in single-input scenarios.

Task interference. Conflicting gradient updates between tasks is a central challenge in MTL, often measured by negative cosine similarity (Zhao et al., 2018; Yu et al., 2020; Shi et al., 2023). Figure 2 (left) shows the gradient similarity distribution for shared convolutional parameters: in hard parameter sharing, the distribution is nearly uniform, meaning roughly half of all updates conflict. MTAN

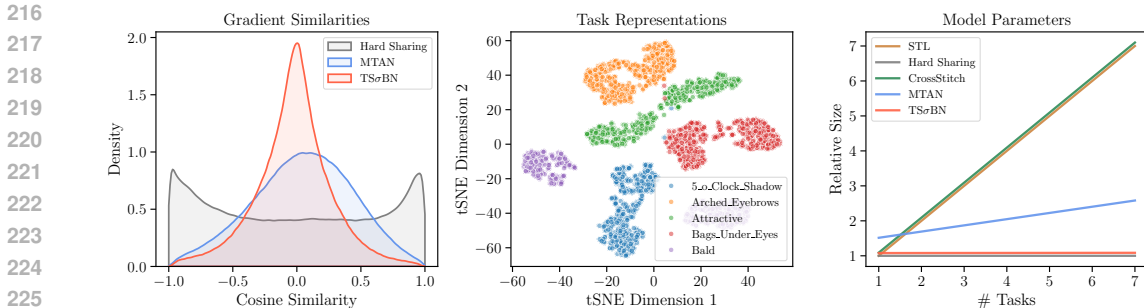


Figure 2: Left: Distribution of cosine similarities between the gradients of NYUv2 tasks over the shared convolutions in the early stages of training. Middle: t-SNE visualization of the encoder representations for the first five CelebA tasks. Right: Encoder parameter count for various numbers of tasks relative to a ResNet50 backbone. Overall, TSσBN has a greater concentration of orthogonal gradients, produces well-separated task representations and has a negligible parameter growth.

(Liu et al., 2019) partially alleviates this issue by introducing task-specific convolutions. In contrast, TSσBN yields a sharp, zero-centered distribution with low variance, indicating gradients are mostly orthogonal. This mirrors optimization-based methods that explicitly enforce orthogonality (Yu et al., 2020; Suteu & Guo, 2019), yet TSσBN achieves it through a lightweight architectural change. Figure 2 (middle) further supports this: on CelebA, task representations form well-separated clusters, illustrating reduced interference. A full analysis across all tasks is provided in Appendix A.

Parameter Efficiency. Task-Specific σBN is highly parameter efficient since it does not introduce additional feature extractors like related soft parameter sharing architectures. At the extreme end, such as Single Task Learning or Cross-Stitch networks, the entire backbone is duplicated for each new task. TSσBN on the other hand duplicates only σBN layers, whose parameters comprise a fraction of the total model size. Figure 2 (right) shows how different approaches scale with additional tasks. TSσBN adds an insignificant amount of new parameters, allowing it to scale to any number of tasks.

Discriminative Learning Rates. We increase the learning rate of σBN parameters by a fixed multiple ($\alpha_{\sigma BN} = 10^2$) relative to other parameters, allowing them to allocate filters before these undergo significant updates. This accelerates specialization and ensures capacity allocation occurs early in training. A further advantage of σBN is its robustness to high learning rates: the sigmoid dampens gradients, making training stable across scales, whereas vanilla BN is more sensitive and requires careful tuning. The approach parallels transfer learning, where deeper layers are updated more aggressively to drive adaptation (Howard & Ruder, 2018; Vlaar & Leimkuhler, 2022). We provide ablations on how higher learning rates improve performance and filter allocation.

5 MTL ANALYSIS WITH TSσBN

A key advantage of the TSσBN design is the ability to quantify filter allocation through task-filter importance matrices. Since each σBN layer introduces a dedicated scaling parameter $\gamma_{t,i}$ per task and filter, we construct a task-filter importance matrix $I \in \mathbb{R}^{T \times F}$, where each entry $I_{t,i}$ captures the importance task t assigns to filter i . Applying the sigmoid function to the raw scaling parameters $I_{t,i} = \sigma(\gamma_{t,i})$ ensures that values remain within $[0, 1]$, facilitating interpretability and comparability across tasks, layers, and models. Using this representation, TSσBN enables a principled analysis of MTL dynamics, including capacity allocation, task relationships, and filter specialization.

5.1 CAPACITY ALLOCATION

One of the central challenges in multi-task learning is understanding how model capacity is allocated among competing tasks. The TSσBN task-filter importance matrix I can directly quantify the total capacity of a task t as the normalized sum of the importances it assigns to filters $C_t = \frac{1}{F} \sum_{i=1}^F \sigma(\gamma_{t,i})$. This measure provides an overall assessment of the resources required for each task; however, it does

not account for task relationships or shared capacity. A task with high absolute capacity does not necessarily imply it monopolizes filters, as it may rely heavily on shared generic filters.

We apply an orthogonal projection-based decomposition to differentiate between task-specific and shared capacity. Given the set of task importance vectors $\{I_1, I_2, \dots, I_T\}$, we decompose each task’s capacity into an independent component and a shared component. Let A be the matrix formed by stacking all task importance vectors except I_t . The projection of I_t onto the subspace spanned by the other tasks is given by the projection matrix P_A :

$$P_A I_t = A(A^T A)^{-1} A^T I_t, \tag{4}$$

The shared $\hat{I}_t = P_A I_t$ and independent $I_t^\perp = I_t - \hat{I}_t$ components of I_t can therefore be defined so that I_t^\perp is orthogonal to the subspace spanned by the other task importance vectors.

To derive a capacity decomposition consistent with the original measure, we define the independent and shared capacities as scaled versions of the total capacity:

$$C_t^{indep} = \frac{\|I_t^\perp\|_2}{\|I_t\|_2} C_t, \quad C_t^{shared} = \frac{\|\hat{I}_t\|_2}{\|I_t\|_2} C_t. \tag{5}$$

Because in this formulation the components are orthogonal, the L_2 norm satisfies the Pythagorean theorem, yielding $C_t^2 = (C_t^{shared})^2 + (C_t^{indep})^2$. This guarantees that a task’s total capacity is preserved while providing an interpretable split between shared and independent resource usage.

Using our framework, we analyze task capacity allocation after training as shown in Figure 3. For both SegNet and DeepLabV3 architectures, we find that most capacity is shared among tasks without a single task dominating. For a more detailed analysis on the effects of task difficulty and similarity on capacity allocation, we refer to Appendix E. Overall, this view offers interpretability into the interaction between tasks and can be a powerful tool in real-world applications where relationships are not known a priori.

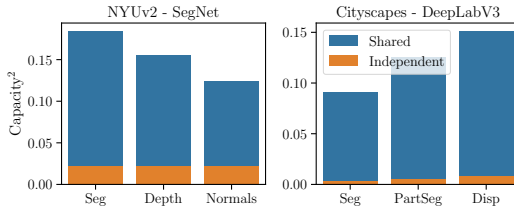


Figure 3: Decomposed task capacity into shared and independent components using the $TS\sigma BN$ framework. In all standard scenarios, tasks share most capacity without signs of dominance.

5.2 TASK RELATIONSHIPS

A desirable feature for any multi-task learning model is the ability to derive task relationships, as this can help gauge interference between tasks and provide insights into the joint optimization process. To showcase this, we use the CelebA dataset, containing 40 binary facial attribute tasks, allowing us to explore complex task relationships and hierarchies via $TS\sigma BN$. Moreover, because these attributes are semantically interpretable (e.g., "Smiling", "Mouth Slightly Open"), they enable meaningful qualitative assessments of the learned relationships.

To derive task relationships we compute the pairwise cosine similarity between the task importance vectors $I_t \in \mathbb{R}^F$, yielding a $T \times T$ similarity matrix, with values ranging from 0 (orthogonal filter usage) to 1 (indicating identical usage). We use this as the basis for constructing distance matrices to identify task clusters and hierarchical relationships that reflect the model’s capacity allocation.

To assess the stability of the task relationships derived from our model, we focus on the consistency of task hierarchies across multiple training runs. Specifically, we evaluate the similarity matrices obtained from seven independently trained models with different intializations. We compute the pairwise Spearman rank correlation between similarity matrices to determine whether the relative task orderings are robust to such variations. Our results show that the task hierarchies are highly stable, with an average Spearman correlation of 0.8 across all model pairs.

We further assess the resulting relationships by aggregating the representative task clusters from the seven runs, via co-occurrence matrices and hierarchical clustering. The identified clusters exhibit

semantic coherence, suggesting a correlation with the spatial proximity of facial attributes. For instance, tasks related to hair characteristics (e.g., Bangs, Blond Hair) form a distinct cluster. In contrast, facial hair attributes (e.g. Goatee, Mustache) are grouped separately. More details about the procedure and resulting task clusters can be found in the Appendix C.

5.3 FILTER GROUPS

A different way to analyze multi-task learning is from an individual filter perspective. Using the task-filter matrix, we can gauge each task’s reliance on a filter to determine if the resource is specialized or generic. We define a filter as specialized for a particular task if its normalized task-filter importance exceeds a threshold τ . We set $\tau = 0.5$ to signify that the filter predominantly contributes to a single task rather than being shared among multiple tasks. Formally, let $\sigma(\gamma_{t,i})$ denote the importance of filter i for task t . A filter i is deemed specialized for task t' if $\sigma(\gamma_{t',i}) / \sum_t \sigma(\gamma_{t,i}) > \tau$.

We prune the top 200 most important filters per task to test our definitions of specialization and importance. If accurate, removing a task’s specialized filters should degrade its performance more than others. Figure 4 (right) confirms this: diagonal elements, representing self-impact, show significantly larger drops than off-diagonals, supporting our hypothesis.

Next, we examine where specialized filters occur across the network. Figure 4 (left) shows the percentage of specialized filters per layer from different runs. Specialization increases with network depth, indicating that early layers are more shared while deeper layers become task-specific. This mirrors findings in single-task learning (Yosinski et al., 2015), where lower layers encode general features, and aligns with branching-based NAS heuristics (Bruggemann et al., 2020; Vandenhende et al., 2020a; Guo et al., 2020), which assign specialized layers to later stages. Our method for quantifying specialization and task similarity offers an alternative perspective for NAS strategies.

6 EXPERIMENTS

We evaluate TS σ BN across a wide range of MTL settings - covering three CNN (from scratch and pretrained) and two vision transformer architectures over four standard MTL datasets: NYUv2 (Silberman et al., 2012), Cityscapes (Cordts et al., 2016), CelebA (Liu et al., 2015) and PascalContext (Chen et al., 2014). We follow established protocols from prior work (Liu et al., 2019; Ban & Ji, 2024; Lin & Zhang, 2023; Yang et al., 2024; Agiza et al., 2024) for training, evaluation, and metric reporting. TS σ BN achieves comparable or superior performance to related and state-of-the-art methods while maintaining better resource efficiency. We refer to Appendix F for additional details on TS σ BN integration, datasets, protocols and baselines.

Convolutional Neural Networks. We evaluate TS σ BN on CNNs in two settings: models trained from scratch and initialized from pretrained backbones. For models trained from scratch, we follow standard protocols on NYUv2 (3-task) using SegNet (Badrinarayanan et al., 2017) as in Liu et al. (2019), and on Cityscapes (3-task) using DeepLabV3 (Chen, 2017) following Liu et al. (2022a). We also evaluate on CelebA, which contains 40 binary classification tasks, and adopt the CNN architecture used in Liu et al. (2024); Ban & Ji (2024). For pretrained CNNs, we integrate TS σ BN into LibMTL (Lin & Zhang, 2023) using DeepLabV3 with a pretrained ResNet50 backbone on NYUv2 (3-task) and Cityscapes (2-task). This allows comparison to a wide range of recent MTL baselines under a consistent framework.

Vision Transformers. We evaluate TS σ BN on two transformer-based MTL setups that reflect current state-of-the-art: MoE-style modulation, and parameter-efficient adapter-based methods. Both settings use pretrained Vision Transformer backbones with CNN based fusion or downsampling modules before task-specific decoders. For recent MoE MTL methods we follow the MLoRE protocol (Yang

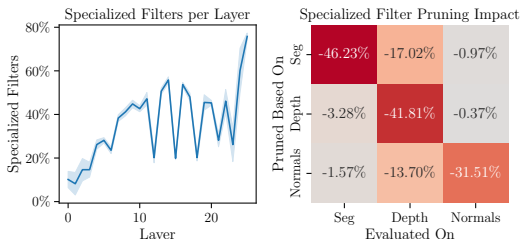


Figure 4: Left: Percentage of specialized filters per layer in a TS σ BN SegNet. Specialization increases in the latter layers. Right: Performance drop across tasks (columns) after pruning filters based on their primary specialization (rows).

et al., 2024) on PascalContext (5-task). We use a pretrained ViT-S backbone (Dosovitskiy et al., 2021) and fine-tune the entire model. We also evaluate TS σ BN on the MTLORA benchmark (Agiza et al., 2024), which focuses on parameter-efficient MTL. This setup uses a partially frozen Swin-T (Liu et al., 2021c) backbone on PascalContext (4-task). We compare against a wide range of LoRA and adapter based models reported in MTLORA. To showcase compatibility we also evaluate TS σ BN with added task-generic (shared) LoRA($r = 16$) adapters.

Multi-task evaluation. Following Maninis et al. (2019) to evaluate a multi-task model, we compute the average per-task performance gain or drop relative to a baseline B specified in the top row of the results tables. $\Delta m\% = \frac{1}{T} \sum_{t=1}^T (-1)^{\delta_t} \frac{M_{m,t} - M_{B,t}}{M_{B,t}} \times 100$, where $M_{m,t}$ is the performance of a model m on a task t , and δ_t is an indicator variable that is 1 if a lower value shows better performance for the metric of task t . All results are presented as an average over three independent runs. Additionally, we report parameters (P) and FLOPs (F) relative to the baseline.

Baselines. Across all experiments we compare TS σ BN to a set of standard and protocol-specific multi-task baselines. The most common reference points are Single-Task Learning (STL), which trains a separate model for each task, and Hard Parameter Sharing (HPS), which shares the entire backbone with equal task weights. We also include TSBN, the multi-task equivalent of domain-specific BN, which simply duplicates BN layers without our reparameterization and optimization changes. Each experimental setting includes additional baselines that follow the protocol and architecture family, reflecting standard practice in prior work and ensuring fair comparisons. For completeness, we also report results for multi-task optimization methods in the Appendix G.

Method	NYUv2					Cityscapes					CelebA		
	#P	Seg \uparrow	Depth \downarrow	Norm \downarrow	$\Delta\%$	#P	Seg \uparrow	P.Seg \uparrow	Disp \downarrow	$\Delta\%$	#P	F1 \uparrow	$\Delta\%$
STL	1.00	41.45	0.580	23.80	0.00	1.00	56.61	53.95	0.841	0.00	1.00	68.21	0.00
HPS	0.33	42.17	0.502	26.63	+1.07	0.60	55.03	51.92	0.796	-0.39	0.03	67.06	-1.69
CS	1.00	41.77	0.492	26.15	+1.98	1.00	56.73	53.89	0.781	+2.43	1.01	65.57	-3.86
MTAN	0.59	43.12	0.508	25.44	+3.14	0.78	55.83	52.61	0.799	+0.39	0.39	59.49	-12.78
TSBN	0.33	43.47	0.494	25.32	+4.42	0.61	56.10	52.82	0.806	+0.40	0.03	67.17	-1.52
TSσBN	0.33	43.75	0.484	24.09	+6.93	0.60	56.45	53.26	0.814	+0.57	0.03	69.45	+1.81

Table 1: Comparison of encoder-based soft-sharing architectures on NYUv2 (3-task SegNet), Cityscapes (3-task DeepLabV3), and CelebA (40-task CNN) trained from random initialization. TS σ BN achieves the best overall performance on NYUv2 and CelebA by a significant margin, and competitive results on Cityscapes, while maintaining the lowest parameter count.

Method	NYUv2						CityScapes				
	#P	#F	Seg \uparrow	Depth \downarrow	Normal \downarrow	$\Delta\%$	#P	#F	Seg \uparrow	Depth \downarrow	$\Delta\%$
HPS	1.00	1.00	53.93	0.3825	23.57	0.00	1.00	1.00	69.81	0.0125	0.00
CS	1.65	1.69	53.44	0.3818	23.15	+0.35	1.42	1.44	69.97	0.0123	+0.55
MMOE	1.35	1.34	53.14	0.3876	23.02	-0.15	1.42	1.44	69.81	0.0126	-0.43
MTAN	1.28	1.56	54.64	0.3771	23.12	+1.55	1.29	1.48	70.62	0.0125	+0.49
CGC	2.01	2.03	53.27	0.3914	22.14	+0.84	1.85	1.88	69.75	0.0125	-0.12
PLE	2.41	2.71	52.75	0.3943	22.10	+0.32	1.95	2.32	69.30	0.0129	-2.02
LTB	1.65	1.69	52.58	0.3828	23.31	-0.49	1.42	1.44	69.81	0.0125	-0.35
DSelect-k	1.38	1.34	53.75	0.3802	23.18	+0.64	1.44	1.44	69.67	0.0124	+0.26
TSBN	1.00	1.69	53.44	0.3761	23.01	+1.04	1.00	1.44	69.89	0.0124	+0.38
TSσBN	1.00	1.69	53.78	0.3735	22.31	+2.48	1.00	1.44	70.17	0.0123	+0.85

Table 2: Comparison of various multi-task architectures within the LibMTL framework using DeepLabV3 with a pre-trained ResNet-50 backbone on NYUv2 (3-task) and CityScapes (2-task). TS σ BN achieves the best overall performance while being the most parameter-efficient.

6.1 RESULTS

Across all experimental settings, $\text{TS}\sigma\text{BN}$ delivers consistent gains in performance while maintaining superior parameter efficiency.

On randomly initialized CNNs in Table 1, $\text{TS}\sigma\text{BN}$ achieves the best results on NYUv2 (+6.93%) and CelebA (+1.81%), with competitive performance on Cityscapes, all at the lowest parameter cost. Notably, soft parameter sharing methods underperform the STL baseline on CelebA, highlighting their poor scalability to many tasks, whereas $\text{TS}\sigma\text{BN}$ remains robust. On pretrained CNNs within LibMTL in Table 2, $\text{TS}\sigma\text{BN}$ achieves the strongest overall performance on both NYUv2 (+2.48%) and Cityscapes (+0.85%), outperforming all MTL baselines, including MoE approaches, while remaining lightweight. On pre-trained transformers with ViT-S in Table 3, $\text{TS}\sigma\text{BN}$ surpasses state-of-the-art methods, such as M^3ViT , Mod-Squad, and MLoRE, while using fewer parameters. Relative to other parameter-efficient fine-tuning approaches in Table 4 $\text{TS}\sigma\text{BN}$ offers the best performance relative to its trainable parameter count. Adding shared capacity via LoRA($r = 16$) adapters further improves performance.

We note that even the simpler TSBN variant (without sigmoid and differential learning rates) delivers competitive performance out of the box, suggesting that complex architectures may be unnecessarily over-engineered. Overall, $\text{TS}\sigma\text{BN}$ achieves the best balance of accuracy, efficiency, and simplicity, consistently outperforming specialized MTL architectures across CNNs and transformers, while scaling to many-task regimes.

7 ABLATIONS

7.1 DISCRIMINATIVE LEARNING RATES

We analyze the impact of different learning rate multipliers applied to the σBN layers, focusing on their effect on the distribution of scaling parameters γ_t and overall model performance. Figure 5 illustrates how varying the α_{BN} multiplier influences the distribution of $\sigma(\gamma_t)$ values across all filters. A more detailed task-wise breakdown is provided in the Appendix. Higher learning rates induce more significant parameter variance, increasing their expressivity. Since $\sigma(\gamma_t)$ is initialized at 0.5, lower learning rates result in minimal divergence, with $\alpha_{\sigma\text{BN}} = 1$ being excluded as it shows almost no differentiation between tasks. At $\alpha_{\sigma\text{BN}} = 100$, we see a substantial spread in $\sigma(\gamma_t)$ values across the full $[0, 1]$ range, allowing tasks to choose and specialize on subsets of filters. However, an extreme learning rate of $\alpha_{\sigma\text{BN}} = 10^3$ leads to a highly polarized distribution, where filter importances collapse to a binary mask, effectively enforcing a hard-partitioning regime. These findings highlight how BN learning rates control the degree of task-specific capacity allocation, influencing both representation disentanglement and network adaptability.

We further analyze the impact of different learning rate multipliers on the MTL performance in Table 6. For TSBN, moderate multipliers yield small gains, but performance collapses at high rates. In contrast, σBN consistently benefits from larger multipliers across values, indicating that sigmoid activation is essential both for unlocking greater improvements and for robustness.

Method	Seg. mIoU \uparrow	Parts. mIoU \uparrow	Sal. maxF \uparrow	Norm. mErr \downarrow	Bdry. odsF \uparrow	#F (G)	#P (M)
M^3ViT	72.80	62.10	66.30	14.50	71.70	420	42
Mod-Squad	74.10	62.70	66.90	13.70	72.00	420	52
TaskExpert	75.04	62.68	84.68	14.22	68.80	204	55
MLoRE	75.64	62.65	84.70	14.43	69.81	72	44
TSBN	75.95	63.33	84.655	14.16	68.05	214	29
$\text{TS}\sigma\text{BN}$	77.12	64.73	85.24	14.04	70.00	214	29

Table 3: PascalContext results for MoE-style models using a pretrained ViT-S backbone. $\text{TS}\sigma\text{BN}$ delivers best results using fewer parameters.

Method	Seg. mIoU \uparrow	Parts. mIoU \uparrow	Sal. mIoU \uparrow	Norm. mErr \downarrow	Δm (%)	#P (M)
STL	67.21	61.93	62.35	17.97	0	112.62
HyperFormer	71.43	60.73	65.54	17.77	2.64	72.77
MTL-Full FT	67.56	60.24	65.21	16.64	2.23	30.06
Adapter	69.21	57.38	61.28	18.83	-2.71	11.24
Polyhistor	70.87	59.15	65.54	17.77	2.34	8.96
MTLoRA($r=16$)	68.19	58.99	64.48	17.03	1.35	4.95
VL-Adapter	70.21	59.15	62.29	19.26	-1.83	4.74
$\text{TS}\sigma\text{BN}(r=16)$	70.00	58.01	63.89	16.85	1.63	4.25
VPT-deep	64.35	52.54	58.15	21.07	-10.85	3.43
MTLoRA+($r=8$)	68.54	58.30	63.57	17.41	0.29	3.15
$\text{TS}\sigma\text{BN}$	69.38	57.46	63.74	17.00	0.91	3.08
LoRA	70.12	57.73	61.90	18.96	-2.17	2.87
BitFit	68.57	55.99	60.64	19.42	-4.60	2.85
Compacter	68.08	56.41	60.08	19.22	-4.55	2.78
Compacter++	67.26	55.69	59.47	19.54	-5.84	2.66
VPT-shallow	62.96	52.27	58.31	20.90	-11.18	2.57

Table 4: Results for PEFT baselines using a Swin-T backbone on PascalContext sorted by number of trainable parameters. $\text{TS}\sigma\text{BN}$ and its combination with LoRA($r = 16$) deliver the best performance relative to their size.

486 7.2 ROBUSTNESS TO LOSS SCALES

487
 488 A well-known challenge in multi-task learning
 489 is the discrepancy in loss scales and, consequently,
 490 gradient magnitudes across tasks, which can lead to
 491 task dominance and suboptimal performance. Many
 492 existing approaches rely on manual tuning or specialized
 493 optimization strategies for dynamic weighting. Our
 494 method is highly robust to perturbations of loss scales
 495 without any additional changes.

496
 497 To evaluate the robustness of our method to loss weight
 498 perturbations, we conduct a series of experiments on
 499 NYUv2 by varying the weight of each task. Specifically,
 500 we scale each task loss by factors of $\{0.5, 1.5, 2.0\}$
 501 while maintaining the default weight of 1.0 for the
 502 remaining tasks. The distribution of relative performances
 503 under these perturbations is visualized in Figure 5. $TS\sigma BN$
 504 shows the lowest variance under loss scale perturbations,
 505 indicating robustness to task dominance and improved
 506 optimization stability.

508 8 CONCLUSION

509
 510 We present $TS\sigma BN$, a simple soft-sharing mechanism for
 511 multi-task learning that relies only on task-specific
 512 normalization layers. Using a sigmoid-gated reparameterization
 513 and differential learning rates, our method turns BN from
 514 a normalization module into a stable and expressive tool for
 515 capacity allocation and interference reduction.

516
 517 Across convolutional and transformer architectures, $TS\sigma BN$
 518 achieves competitive or superior performance while using
 519 substantially fewer parameters. Notably, it matches or
 520 outperforms state-of-the-art MoE-style and PEFT-based
 521 MTL methods without adding routing modules, experts, or
 522 adapters. The learned gates also provide a direct view of
 523 model behavior, yielding interpretable measures of capacity
 524 allocation, filter specialization, and task relationships.

525
 526 Overall, our results show that lightweight, normalization-driven
 527 designs can replace much heavier mechanisms while offering
 528 clearer interpretability. We hope this encourages a
 529 reevaluation of complexity in MTL and promotes simple,
 530 transparent alternatives.

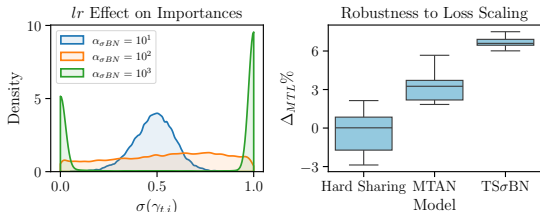


Figure 5: Effect of BN-specific learning rate multipliers on the $\sigma(\gamma_t)$ filter importances distribution (left) and relative performance of models under loss scale perturbations (right).

$\alpha_{\sigma BN}$	10^0	10^1	10^2	10^3
TSBN	+4.09%	+4.80%	+4.42%	-2.96%
$TS\sigma BN$	+4.02%	+5.67%	+6.93%	+4.33%

Figure 6: Impact of different BN specific learning rate multipliers on the performance of TSBN and $TS\sigma BN$ relative to STL on NYUv2.

540 ETHICS STATEMENT
541

542 This work does not involve human subjects, private data, or sensitive content. All datasets used
543 (NYUv2, Cityscapes, CelebA, PascalContext) are publicly available and widely adopted benchmarks.
544

545 REPRODUCIBILITY STATEMENT
546

547 We provide comprehensive experimental details in the main text in Section 6 and Appendix F,
548 including datasets, architectures, training protocols, and evaluation metrics.
549

550 REFERENCES
551

552 Ahmed Agiza, Marina Neseem, and Sherief Reda. Mtlora: Low-rank adaptation approach for efficient
553 multi-task learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern
554 recognition*, pp. 16196–16205, 2024.
555

556 Abhishek Aich, Samuel Schuler, Amit K Roy-Chowdhury, Manmohan Chandraker, and Yumin Suh.
557 Efficient controllable multi-task architectures. In *Proceedings of the IEEE/CVF International
558 Conference on Computer Vision*, pp. 5740–5751, 2023.
559

560 Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint
561 arXiv:1607.06450*, 2016.
562

563 Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-
564 decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine
565 intelligence*, 39(12):2481–2495, 2017.

566 Hao Ban and Kaiyi Ji. Fair resource allocation in multi-task learning. In *International Conference on
567 Machine Learning*, 2024.
568

569 Hakan Bilen and Andrea Vedaldi. Universal representations: The missing link between faces, text,
570 planktons, and cat breeds. *arXiv preprint arXiv:1701.07275*, 2017.

571 Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normaliza-
572 tion. *Neural Information Processing Systems*, 2018.
573

574 Felix JS Bragman, Ryutaro Tanno, Sebastien Ourselin, Daniel C Alexander, and Jorge Cardoso.
575 Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels.
576 In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1385–1394,
577 2019.

578 John Bronskill, Jonathan Gordon, James Requeima, Sebastian Nowozin, and Richard Turner. Tas-
579 knorm: Rethinking batch normalization for meta-learning. In *International Conference on Machine
580 Learning*, 2020.
581

582 David Bruggemann, Menelaos Kanakis, Stamatios Georgoulis, and Luc Van Gool. Automated search
583 for resource-efficient branched multi-task networks. *British Machine Vision Conference*, 2020.
584

585 Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific
586 batch normalization for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF
587 conference on Computer Vision and Pattern Recognition*, pp. 7354–7362, 2019.

588 Liang-Chieh Chen. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint
589 arXiv:1706.05587*, 2017.
590

591 Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille.
592 Detect what you can: Detecting and representing objects using holistic models and body parts. In
593 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1971–1978,
2014.

- 594 Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient
595 normalization for adaptive loss balancing in deep multitask networks. In *International Conference*
596 *on Machine Learning*, 2018.
- 597
598 Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and
599 Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign
600 dropout. *Advances in Neural Information Processing Systems*, 2020.
- 601
602 Zitian Chen, Yikang Shen, Mingyu Ding, Zhenfang Chen, Hengshuang Zhao, Erik G Learned-Miller,
603 and Chuang Gan. Mod-squad: Designing mixtures of experts as modular multi-task learners.
604 In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.
605 11828–11837, 2023.
- 606
607 Sumanth Chennupati, Ganesh Sistu, Senthil Kumar Yogamani, and Samir A. Rawashdeh. Multi-
608 net++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. *2019*
609 *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp.
610 1200–1210, 2019.
- 611
612 Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo
613 Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban
614 scene understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern*
615 *recognition*, pp. 3213–3223, 2016.
- 616
617 Hadi Daneshmand, Amir Joudaki, and Francis R. Bach. Batch normalization orthogonalizes repre-
618 sentations in deep random networks. *Neural Information Processing Systems*, 2021.
- 619
620 Weijian Deng, Yumin Suh, Xiang Yu, Masoud Faraki, Liang Zheng, and Manmohan Chandraker.
621 Split to learn: gradient split for multi-task human image analysis. In *Proceedings of the IEEE/CVF*
622 *Winter Conference on Applications of Computer Vision*, pp. 4351–4360, 2023.
- 623
624 Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas
625 Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit,
626 and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale.
627 *ICLR*, 2021.
- 628
629 Zhiwen Fan, Rishov Sarkar, Ziyu Jiang, Tianlong Chen, Kai Zou, Yu Cheng, Cong Hao, Zhangyang
630 Wang, et al. M³vit: Mixture-of-experts vision transformer for efficient multi-task learning with
631 model-accelerator co-design. *Advances in Neural Information Processing Systems*, 35:28441–
632 28457, 2022.
- 633
634 Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying
635 task groupings for multi-task learning. *Neural Information Processing Systems*, 2021.
- 636
637 Jonathan Frankle, David J Schwab, and Ari S Morcos. Training batchnorm and only batchnorm:
638 On the expressive power of random features in cnns. *International Conference on Learning*
639 *Representations*, 2021.
- 640
641 Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In
642 *International Conference on Machine Learning*, 2020.
- 643
644 Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen,
645 Rahul Mazumder, Lichan Hong, and Ed Chi. Dselect-k: Differentiable selection in the mixture
646 of experts with applications to multi-task learning. *Advances in Neural Information Processing*
647 *Systems*, 34:29335–29347, 2021.
- 648
649 Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a
650 unified view of parameter-efficient transfer learning. *ICLR*, 2021.
- 651
652 Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In
653 *Annual Meeting of the Association for Computational Linguistics*, 2018. URL [https://api-](https://api.semanticscholar.org/CorpusID:40100965)
654 [semanticscholar.org/CorpusID:40100965](https://api.semanticscholar.org/CorpusID:40100965).

- 648 Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization techniques in training
649 dnns: Methodology, analysis and application. *IEEE transactions on pattern analysis and machine*
650 *intelligence*, 2023.
- 651 Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate
652 shift. *International Conference on Machine Learning*, 2015.
- 653 Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and
654 Ser-Nam Lim. Visual prompt tuning. In *European conference on computer vision*, pp. 709–727.
655 Springer, 2022.
- 656 Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank
657 hypercomplex adapter layers. *Advances in neural information processing systems*, 34:1022–1035,
658 2021.
- 659 Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses
660 for scene geometry and semantics. *2018 IEEE/CVF Conference on Computer Vision and Pattern*
661 *Recognition*, 2017.
- 662 Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization
663 for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.
- 664 Baijiong Lin and Yu Zhang. LibMTL: A Python library for multi-task learning. *Journal of Machine*
665 *Learning Research*, 24(209):1–7, 2023.
- 666 Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor Wai-Hung Tsang. Reasonable effectiveness of random
667 weighting: A litmus test for multi-task learning. *Trans. Mach. Learn. Res.*, 2022, 2021.
- 668 Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for
669 multi-task learning. In *Neural Information Processing Systems*, 2021a.
- 670 Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. *Neural*
671 *Information Processing Systems*, 2024.
- 672 Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and
673 Wayne Zhang. Towards impartial multi-task learning. In *International Conference on Learning*
674 *Representations*, 2021b.
- 675 Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention.
676 In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.
677 1871–1880, 2019.
- 678 Shikun Liu, Stephen James, Andrew J Davison, and Edward Johns. Auto-lambda: Disentangling
679 dynamic task relationships. *Transactions on Machine Learning Research*, 2022a.
- 680 Yen-Cheng Liu, Chih-Yao Ma, Junjiao Tian, Zijian He, and Zsolt Kira. Polyhistor: Parameter-efficient
681 multi-task adaptation for dense vision tasks. *Advances in Neural Information Processing Systems*,
682 35:36889–36901, 2022b.
- 683 Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo.
684 Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the*
685 *IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021c.
- 686 Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learn-
687 ing efficient convolutional networks through network slimming. In *Proceedings of the IEEE*
688 *international conference on computer vision*, 2017.
- 689 Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In
690 *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.
- 691 Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in
692 batch normalization. *International Conference on Learning Representations*, 2019.

- 702 Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships
703 in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD
704 international conference on knowledge discovery & data mining*, pp. 1930–1939, 2018.
- 705
706 Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-
707 efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Annual Meeting of the
708 Association for Computational Linguistics*, 2021. URL [https://api.semanticscholar.
709 org/CorpusID:235309789](https://api.semanticscholar.org/CorpusID:235309789).
- 710 Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple
711 tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*,
712 pp. 1851–1860, 2019.
- 713 Krzysztof Maziarczyk, Efi Kokiopoulou, Andrea Gesmundo, Luciano Sbaiz, Gabor Bartok, and
714 Jesse Berent. Flexible multi-task networks by learning parameter allocation. *arXiv preprint
715 arXiv:1910.04915*, 2019.
- 716
717 Vincent Michalski, Vikram Voleti, Samira Ebrahimi Kahou, Anthony Ortiz, Pascal Vincent, Chris
718 Pal, and Doina Precup. An empirical study of batch normalization and group normalization in
719 conditional computation. *arXiv preprint arXiv:1908.00061*, 2019.
- 720
721 Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for
722 multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern
723 Recognition*, pp. 3994–4003, 2016.
- 724 Pramod Kaushik Mudrakarta, Mark Sandler, Andrey Zhmoginov, and Andrew G. Howard. K for
725 the price of 1: Parameter efficient multi-task and transfer learning. *International Conference on
726 Learning Representations*, 2019.
- 727
728 Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and
729 Ethan Fetaya. Multi-task learning as a bargaining game. In *International Conference on Machine
730 Learning*, 2022.
- 731 Alejandro Newell, Lu Jiang, Chong Wang, Li-Jia Li, and Jia Deng. Feature partitioning for efficient
732 multi-task architectures. *arXiv preprint arXiv:1908.04339*, 2019.
- 733
734 Dripta S Raychaudhuri, Yumin Suh, Samuel Schuster, Xiang Yu, Masoud Faraki, Amit K Roy-
735 Chowdhury, and Manmohan Chandraker. Controllable dynamic multi-task architectures. In
736 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.
737 10955–10964, 2022.
- 738
739 Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with
740 residual adapters. *Neural Information Processing Systems*, 2017.
- 741
742 Amir Rosenfeld and John K Tsotsos. Intriguing properties of randomly weighted networks: General-
743 izing while learning next to nothing. In *Conference on Computer and Robot Vision (CRV)*. IEEE,
744 2019.
- 745
746 Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task
747 architecture learning. In *Proceedings of the AAAI conference on artificial intelligence*, 2019.
- 748
749 Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normal-
750 ization help optimization? *Neural Information Processing Systems*, 2018.
- 751
752 Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Neural
753 Information Processing Systems*, 2018.
- 754
755 Guangyuan Shi, Qimai Li, Wenlong Zhang, Jiaxin Chen, and Xiao-Ming Wu. Recon: Reducing
756 conflicting gradients from the root for multi-task learning. *International Conference on Learning
757 Representations*, 2023.
- 758
759 Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support
760 inference from rgb-d images. In *European Conference on Computer Vision*, 2012.

- 756 Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese.
757 Which tasks should be learned together in multi-task learning? In *International Conference on*
758 *Machine Learning*, 2020.
- 759 Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Many task learning with task routing. In
760 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.
- 762 Cecilia Summers and Michael J Dinneen. Four things everyone should know to improve batch
763 normalization. *International Conference on Learning Representations*, 2020.
- 764 Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share
765 for efficient deep multi-task learning. *Neural Information Processing Systems*, 2020.
- 766 Mihai Suteu and Yike Guo. Regularizing deep multi-task networks using orthogonal gradients. *arXiv*
767 *preprint arXiv:1912.06844*, 2019.
- 769 Mihai Suteu and Yike Guo. Receding neuron importances for structured pruning. *arXiv preprint*
770 *arXiv:2204.06404*, 2022.
- 772 Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (ple): A
773 novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the*
774 *14th ACM conference on recommender systems*, pp. 269–278, 2020.
- 775 Simon Vandenhende, Stamatios Georgoulis, Bert De Brabandere, and Luc Van Gool. Branched
776 multi-task networks: deciding what layers to share. *British Machine Vision Conference*, 2020a.
- 777 Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction
778 networks for multi-task learning. In *European Conference on Computer Vision*, 2020b.
- 780 Tiffany J Vlaar and Benedict Leimkuhler. Multirate training of neural networks. In *International*
781 *Conference on Machine Learning*, 2022.
- 783 Matthew Wallingford, Hao Li, Alessandro Achille, Avinash Ravichandran, Charless Fowlkes, Rahul
784 Bhotika, and Stefano Soatto. Task adaptive parameter sharing for multi-task learning. In *Proceed-*
785 *ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7561–7570,
786 2022.
- 787 Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and
788 improving multi-task optimization in massively multilingual models. In *International Conference*
789 *on Learning Representations*, 2021.
- 790 Xuchen Xie, Junjie Xu, Ping Hu, Weizhuo Zhang, Yujun Huang, Weishi Zheng, and Ruixuan Wang.
791 Task-incremental medical image classification with task-specific batch normalization. In *Chinese*
792 *Conference on Pattern Recognition and Computer Vision (PRCV)*, pp. 309–320. Springer, 2023.
- 794 Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-
795 and-distillation network for simultaneous depth estimation and scene parsing. In *Proceedings of*
796 *the IEEE/CVF International Conference on Computer Vision*, pp. 675–684, 2018.
- 797 Yuqi Yang, Peng-Tao Jiang, Qibin Hou, Hao Zhang, Jinwei Chen, and Bo Li. Multi-task dense
798 prediction via mixture of low-rank experts. In *Proceedings of the IEEE/CVF conference on*
799 *computer vision and pattern recognition*, pp. 27927–27937, 2024.
- 800 Hanrong Ye and Dan Xu. Taskexpert: Dynamically assembling multi-task representations with
801 memorial mixture-of-experts. In *Proceedings of the IEEE/CVF international conference on*
802 *computer vision*, pp. 21828–21837, 2023.
- 804 Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural
805 networks through deep visualization. In *Deep Learning Workshop, International Conference on*
806 *Machine Learning (ICML)*, 2015.
- 807 Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn.
808 Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*,
809 2020.

810 Michal Zajac, Konrad Zolna, and Stanislaw Jastrzebski. Split batch normalization: Improving
811 semi-supervised learning under domain shift. *arXiv preprint arXiv:1904.03515*, 2019.
812

813 Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning
814 for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the*
815 *Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, 2022.

816 Wen Zhang, Lingfei Deng, Lei Zhang, and Dongrui Wu. A survey on negative transfer. *IEEE/CAA*
817 *Journal of Automatica Sinica*, 2022.
818

819 Bingchen Zhao, Haoqin Tu, Chen Wei, Jieru Mei, and Cihang Xie. Tuning layernorm in atten-
820 tion: Towards efficient multi-modal LLM finetuning. *International Conference on Learning*
821 *Representations*, 2024.

822 Xiangyu Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. A modulation module for
823 multi-task learning with applications in image retrieval. In *European Conference on Computer*
824 *Vision*, 2018.
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863

A INTERFERENCE

To further investigate task interference, we expand on the analysis presented in Section 4 and provide a more comprehensive view of gradient conflicts across all task pairs for NYUv2. Specifically, in figure 7 we plot the distribution of cosine similarities between gradients for every task pair across the shared parameters of the SegNet backbone.

In addition to the methods discussed in the main paper, we include Task-Specific Batch Normalization (TSBN) as a baseline. Interestingly, TSBN alone is sufficient to induce a mode around orthogonality, demonstrating that normalization can already reduce some degree of task interference. However, incorporating σ BN significantly amplifies this effect, further increasing the number of near-orthogonal gradients and reducing interference. This highlights the role of σ BN in not only mitigating conflicts but also improving gradient disentanglement across tasks.

It is important to note that the presented gradient distributions are measured after one epoch of training over the training set. As training progresses, we observe that the differences between methods become less pronounced. Regardless of the initial distribution, all approaches gradually converge toward a bell-shaped distribution centered around orthogonality. This suggests that while early-stage interference may impact optimization dynamics, multi-task models eventually adjust to reduce conflicts over time.

A notable exception is observed in MTAN, which produces more aligned gradients specifically for the semantic segmentation and surface normal estimation task pair. Despite this alignment, we do not observe a corresponding performance gain. This suggests that while reducing conflicts is beneficial, not all aligned gradients lead to improved task synergy, underscoring the notion that mitigating interference alone does not guarantee optimal performance.

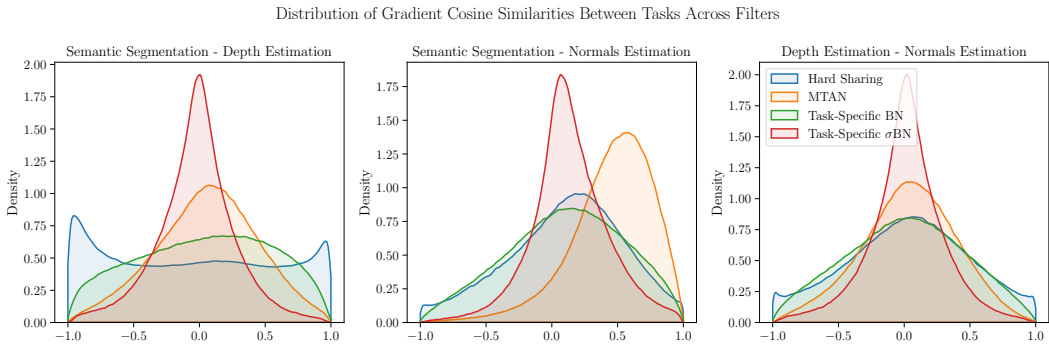


Figure 7: Distribution of gradient cosine similarities between all task pairs on the NYUv2 dataset using a SegNet backbone.

B DISENTANGLED TASK REPRESENTATIONS

We extend Figure 2 from Section 4 by visualizing encoder representations for all 40 tasks in the CelebA setting. As before, we use t-SNE to project the high-dimensional representations into a more interpretable space. Each data point is assigned representations for every task due to the nature of the soft parameter sharing paradigm, resulting in multiple embeddings per sample. In Figure 8, we observe that most tasks form well-separated clusters, though a few outliers exhibit some degree of overlap.

918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

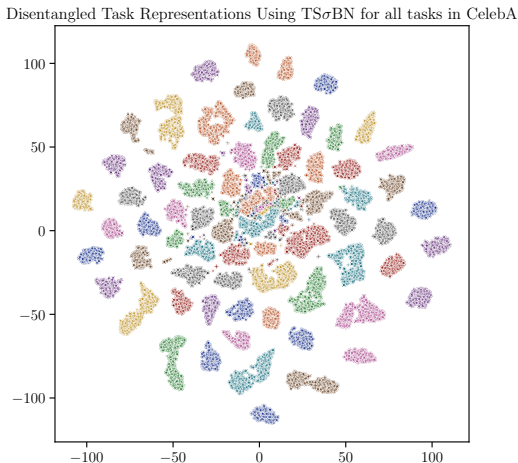


Figure 8: t-SNE visualization for all task representations for 1000 inputs from the CelebA dataset.

C ROBUST TASK RELATIONSHIPS

We utilize the CelebA dataset to identify relationships and hierarchies among the 40 binary classification tasks of facial attributes. We compute pairwise cosine similarities between task importance vectors, producing a task similarity matrix $S = \left[\frac{I_i \cdot I_j}{\|I_i\| \|I_j\|} \right]_{i,j \in \mathcal{T}}$, that serves as the foundation for identifying task clusters and hierarchies. Crucially, for these relationships to be useful, they must be robust - unstable hierarchies would offer little insight into model behavior or optimization dynamics. We find the relationships from TS σ BN to be highly stable, with an average Spearman rank correlation of 0.8 between similarity matrices from seven independent training runs.

For a qualitative assessment of task relationships, we compute representative clusters of tasks from the seven runs. To achieve this, we construct a co-occurrence matrix that captures the frequency with which each pair of tasks appears in the same cluster. This co-occurrence matrix effectively aggregates clustering information from all runs, highlighting task pairs that consistently exhibit strong relationships regardless of initialization. We then apply hierarchical clustering directly to this matrix to identify a representative cluster of tasks that frequently co-occur.

The identified clusters exhibit apparent semantic coherence, as shown in Table 5. Since these clusters are derived from filter-usage based relationships, tasks grouped tend to rely on similar specialized filters within the network. This suggests that the model internally organizes tasks based on shared feature representations. Notably, the clustering patterns appear to correlate with the spatial proximity of facial attributes. For instance, tasks related to hair characteristics (e.g., Bangs, Blond Hair) form a distinct cluster. In contrast, facial hair attributes (e.g. Goatee, Mustache) are grouped separately, indicating that the network leverages localized feature detectors. This spatial coherence reinforces the idea that task relationships emerge from shared activations of filters sensitive to specific facial regions, reflecting the model’s ability to capture both semantic and structural commonalities across tasks.

D DISCRIMINATIVE LEARNING RATES

We extend the ablation study from Section 7.1, investigating the impact of discriminative learning rates for σ BN layers. Specifically, we apply a higher learning rate to BN parameters, allowing them to adapt more rapidly to the shared convolutional layers before those layers undergo significant updates. This adjustment is controlled by a multiplier applied to the model’s base learning rate.

In this more detailed analysis, we examine the importance distributions of filters per task across different learning rate multipliers. Figure 9 presents the resulting distributions for four multiplier values: $10^0, 10^1, 10^2, 10^3$. As the multiplier increases, the variance of filter importance distributions grows, leading to progressively softer filter allocations. At a multiplier of 1, BN parameters remain close to their initialization, resulting in near-uniform filter sharing across tasks, similar to hard

#	Attributes
1	High Cheekbones, Mouth Slightly Open, Smiling
2	Bangs, Black Hair, Blond Hair, Brown Hair, Gray Hair, Straight Hair, Wavy Hair, Wearing Hat
3	Attractive, Bags Under Eyes, Big Nose, Young
4	Bald, Chubby, Double Chin, Receding Hairline, Wearing Necktie
5	Blurry, Heavy Makeup, Male, Pale Skin, Wearing Lipstick
6	5 o’Clock Shadow, Goatee, Mustache, No Beard, Sideburns
7	Arched Eyebrows, Bushy Eyebrows, Narrow Eyes
8	Eyeglasses, Rosy Cheeks
9	Big Lips, Oval Face, Pointy Nose
10	Wearing Earrings, Wearing Necklace

Table 5: Clusters of attributes extracted from a $TS\sigma BN$ model trained on the 40-task CelebA dataset. Task relationships correlate with the spatial proximity of facial features, suggesting that the model organizes tasks based on localized filter activations, capturing both semantic and structural similarities.

parameter sharing. On the opposite extreme, a multiplier of 10^3 effectively induces a binary filter mask, resembling a hard partitioning approach. Notably, σBN plays a crucial role in stabilizing this process, as its sigmoid activation mitigates potential gradient explosion. We use $\alpha_{\sigma BN} = 10^2$ in all our experiments.

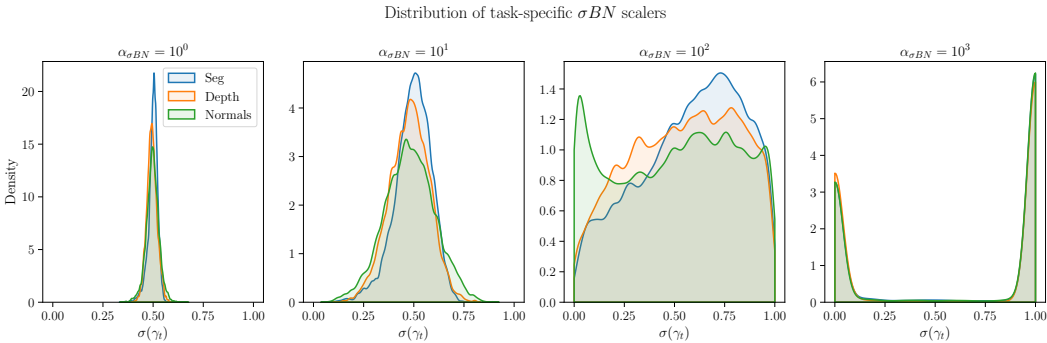


Figure 9: Detailed visualization of the effect of different learning rates on the distribution of task-specific σBN scaling parameters.

E EFFECTS OF TASK DIFFICULTY ON CAPACITY ALLOCATION

To further investigate MTL capacity allocation using the $TS\sigma BN$ framework, we conduct a synthetic experiment designed to control task difficulty and relationships systematically. Specifically, we modify the NYUv2 dataset by removing the surface normals estimation task and replacing it with a noisy variant of the depth estimation task. We generate a family of datasets where the additional depth task is corrupted by Gaussian noise of increasing variance. Formally, given the original depth labels D , we construct synthetic tasks:

$$\tilde{D}_\xi = D + \mathcal{N}(0, \xi * \sigma_D^2), \tag{6}$$

where ξ controls the level of corruption as a scaler of the original depth task’s variance. Using $TS\sigma BN$, we analyze how model capacity is allocated between shared and task-specific components, as well as how task relationships change, by computing cosine similarity over task importance vectors.

In figure 10 we plot the decomposed task capacities and pairwise similarities for datasets with ξ ranging between $[0, 3]$. As expected, when ξ is low, the original and noisy depth tasks exhibit strong

alignment, reinforcing high shared capacity. However, as ξ increases, the similarity between the tasks decreases, and their filter allocations become more distinct, with independent capacity increasing. This aligns with our hypothesis that related tasks co-adapt to share resources, whereas unrelated tasks require greater specialization. Overall, this experiment highlights how $\text{TS}\sigma\text{BN}$ automatically balances shared and independent capacity in response to increasing task difficulty and lower task similarity.

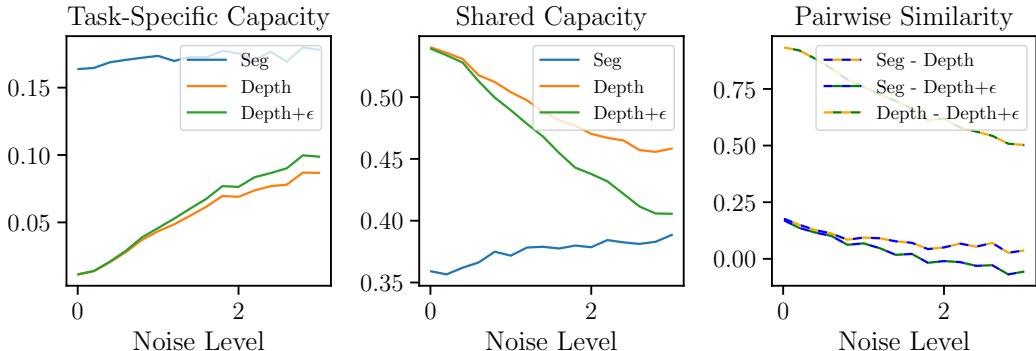


Figure 10: The effect of increasing task difficulty and decreasing similarity on capacity allocation in $\text{TS}\sigma\text{BN}$. For the noise scaling factor, a synthetic depth estimation task is generated with additive Gaussian noise. (Left) Independent task-specific capacities. (Middle) Shared capacity between tasks. (Right) Pairwise cosine similarity between task importance vectors.

F EXPERIMENTAL SETTINGS

Hardware. Experiments on NYUv2 and Cityscapes were run on an NVIDIA RTX 3090 GPU. Due to higher memory requirements, CelebA (40 tasks) and transformer-based models were trained on an NVIDIA A100 GPU.

F.1 CNNs WITH RANDOM INITIALIZATION

NYUv2. We follow the setup of Liu et al. (2019; 2024) for base architecture, training configuration, and evaluation metrics. A multi-task SegNet is used, with both encoder and decoder shared across tasks and lightweight task-specific heads composed of two convolutional layers. All methods are trained with Adam ($lr = 10^{-4}$), using a step schedule that halves the learning rate at epoch 100. Training runs for 200 epochs with a batch size of 4.

Cityscapes. Following Liu et al. (2022a), we use DeepLabV3 with a ResNet-50 backbone and task-specific ASPP decoders, which account for most of the parameters. Optimization is performed with SGD ($lr = 10^{-2}$, weight decay = 10^{-4} , momentum = 0.9) for 200 epochs using a CosineAnnealing scheduler and batch size of 4. For $\text{TS}\sigma\text{BN}$ layers, weight decay is disabled.

CelebA. We adopt the configuration from Liu et al. (2024); Ban & Ji (2024), using a shared CNN backbone with task-specific linear classifiers. Models are trained for 15 epochs with Adam ($lr = 3 \times 10^{-4}$) and batch size 256.

F.2 CNNs WITH PRETRAINED WEIGHTS

Implementation. Converting pretrained BN layers into σBN depends on their weights. A network trained from scratch may learn a purely linear transformation, but converting an affine layer to linear is not possible unless $\beta = 0$. To avoid conversion shock, we copy the pretrained biases but keep them frozen during training. In ResNet-50 pretrained on ImageNet, most BN scale parameters (γ) fall within (0,1), allowing them to be represented by the sigmoid function. We therefore apply the inverse sigmoid to initialize σBN scales, ensuring consistency with the pretrained distribution.

1080 **NYUv2**. We follow the default LibMTL configuration (Lin & Zhang, 2023), reporting results of
 1081 related methods as published. Models are trained with Adam ($lr = 10^{-4}$) for 200 epochs, using
 1082 StepLR with $\gamma = 0.5$ at epoch 100 and a batch size of 4.

1083 **Cityscapes**. Same as above, except the batch size is set to 16 due to memory constraints. All results,
 1084 including related methods, are averaged over three random seeds for fair comparison.
 1085

1086 F.3 VISION TRANSFORMERS

1088 Following prior work, we extract intermediate representations from the transformer backbone
 1089 and process them through a lightweight multi-scale fusion module. The module consists of four
 1090 Conv-TS σ BN-GELU blocks shared across tasks, implemented as two 1×1 convolutions for channel
 1091 adjustment squeezing two 3×3 convolutions with width 512; decoder inputs have width 196. In the
 1092 ViT patch embedding, we replace the normalization with a σ LN layer. All remaining LNs in the
 1093 backbone are converted to TSLN, since their pretrained scales often exceed the sigmoid co-domain.

1094 Following the MLoRE setup (Yang et al., 2024), we train a ViT-S backbone using Adam with base
 1095 learning rate 2×10^{-5} and polynomial decay. Learning-rate multipliers of 100 and 10 are applied
 1096 to TS σ BN and TSLN/TS σ LN layers respectively. Dropout and DropPath are disabled. Models are
 1097 trained for 60k iterations.
 1098

1099 F.4 SWIN TRANSFORMER

1101 We follow the MTLORA (Agiza et al., 2024) experimental protocol and use a pretrained Swin-
 1102 T backbone. Intermediate representations from each stage are extracted and passed through a
 1103 downsampling module before the task-specific HRNet heads. The original downsampler consists of a
 1104 single 1×1 convolution per scale; in our variant, we insert TS σ BN modules both before and after
 1105 this convolution. As in the ViT experiments, all remaining LayerNorm layers in the backbone are
 1106 converted to TSLN.

1107 In the PEFT setting most of the Swin-T backbone remains frozen, including the self-attention and
 1108 MLP blocks. The patch embedding and patch merging layers stay trainable. When combining
 1109 TS σ BN with LoRA, we add adapters only to the frozen linear layers and keep them shared across
 1110 tasks, so they do not contribute task-specific capacity. We integrate our model using the MTLORA
 1111 codebase and keep all training hyperparameters unchanged. The only modification is the optimizer
 1112 configuration required to support differential learning rates: as in all experiments we use a multiplier
 1113 of 100 for TS σ BN parameters and 10 for LN.

1114 G ADDITIONAL BENCHMARKS

1116 For completeness, we benchmark TS σ BN against multi-task optimization (MTO) methods. These
 1117 approaches are orthogonal to architectural soft-sharing: instead of allocating task-specific parameters,
 1118 they operate on a fully shared backbone and modify the optimization dynamics to mitigate conflict.
 1119

1120 MTO methods fall into two major categories: loss-balancing strategies (e.g., UW, DWA, RLW) and
 1121 gradient manipulation strategies (e.g., GradNorm, MGDA, PCGrad, CAGrad, Nash-MTL). Because
 1122 they maintain a fully shared backbone, these methods are often highly parameter-efficient, but this
 1123 comes with different trade-offs: they typically require additional computational overhead, involve
 1124 external solvers to compute descent directions, and can significantly increase training time.

1125 We evaluate TS σ BN within the LibMTL framework, using the NYUv2 setup with a pretrained
 1126 DeepLabV3-ResNet50 backbone. We compare against the suite of optimization-based methods
 1127 available in LibMTL, including UW (Kendall et al., 2017), GradNorm (Chen et al., 2018), MGDA
 1128 (Sener & Koltun, 2018), DWA (Liu et al., 2019), GLS (Chennupati et al., 2019), PCGrad (Yu et al.,
 1129 2020), GradDrop (Chen et al., 2020), IMTL (Liu et al., 2021b), GradVac (Wang et al., 2021), CAGrad
 1130 (Liu et al., 2021a), Nash-MTL (Navon et al., 2022), and RLW (Lin et al., 2021).

1131 Table 6 reports the results. TS σ BN achieves the highest overall multi-task gain among all optimization-
 1132 based methods on top of the same HPS architecture, while maintaining the simplicity of a purely
 1133 architectural soft-sharing mechanism.

1134
 1135
 1136
 1137
 1138
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146
 1147
 1148
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1170
 1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187

Method	Seg \uparrow	Depth \downarrow	Normal \downarrow	Δm (%) \uparrow
HPS	53.93	0.3825	23.57	0.00
+GradNorm	53.91	0.3842	23.17	0.41
+UW	54.29	0.3815	23.48	0.44
+MGDA	53.52	0.3852	22.74	0.69
+DWA	54.06	0.3820	23.70	-0.06
+GLS	54.59	0.3785	22.71	1.97
+PCGrad	53.94	0.3804	23.52	0.26
+GradDrop	53.73	0.3837	23.54	-0.19
+IMTL	53.63	0.3868	22.58	0.84
+GradVac	54.21	0.3859	23.58	-0.14
+CAGrad	53.97	0.3885	22.47	1.06
+Nash-MTL	53.41	0.3867	22.57	0.73
+RLW	54.04	0.3827	23.07	0.76
TSσBN	53.78	0.3735	22.30	2.48

Table 6: Comparison with optimization-based MTL methods on the LibMTL NYUv2 benchmark using a pretrained DeepLabV3-ResNet50 backbone. All optimization methods are applied on top of the same HPS architecture. TS σ BN achieves the highest overall multi-task improvement among all optimization-based approaches.