# DMTG: One-Shot Differentiable Multi-Task Grouping

**Yuan Gao** [1 2] **Shuguo Jiang** [1] **Moran Li** [3] **Jin-Gang Yu** [4] **Gui-Song Xia** [1]

## Abstract

We aim to address Multi-Task Learning (MTL) with a large number of tasks by Multi-Task Grouping (MTG). Given $N$ tasks, we propose to *simultaneously identify the best task groups from* $2^N$ *candidates and train the model weights simultaneously in one-shot*, with *the high-order task-affinity fully exploited*. This is distinct from the pioneering methods which sequentially identify the groups and train the model weights, where the group identification often relies on heuristics. As a result, our method not only improves the training efficiency, but also mitigates the objective bias introduced by the sequential procedures that potentially lead to a suboptimal solution. Specifically, *we formulate MTG as a fully differentiable pruning problem on an adaptive network architecture determined by an underlying* `Categorical` *distribution*. To categorize $N$ tasks into $K$ groups (represented by $K$ encoder branches), we initially set up $KN$ task heads, where each branch connects to all $N$ task heads to exploit the high-order task-affinity. Then, we gradually prune the $KN$ heads down to $N$ by learning a relaxed differentiable `Categorical` distribution, ensuring that each task is exclusively and uniquely categorized into only one branch. Extensive experiments on CelebA and Taskonomy datasets with detailed ablations show the promising performance and efficiency of our method. The codes are available at https://github.com/ethanygao/DMTG.

## 1. Introduction

Many real-world applications are essentially complex systems that involve the collaboration of a large number of tasks. For example, in autonomous driving (Caesar et al., 2020; Hu et al., 2023), the system needs to simultaneously perform lane detection (Tang et al., 2021), depth estimation (Godard et al., 2019), vehicle detection and instance segmentation (He et al., 2017; Cheng et al., 2021), pedestrians localization (Bertoni et al., 2019), etc. In order to tackle these real-world challenges, it is crucial to simultaneously learn a large number of diverse tasks within a Multi-Task Learning (MTL) framework (Kokkinos, 2017; Nekrasov et al., 2019; Dvornik et al., 2017; Bilen & Vedaldi, 2016; Zamir et al., 2016; Li & Gong, 2021; Wang & Tsvetkov, 2021; Liu et al., 2020; Yu et al., 2020), which reduces the inference time and facilitates an improved performance by leveraging the affinity among different tasks.

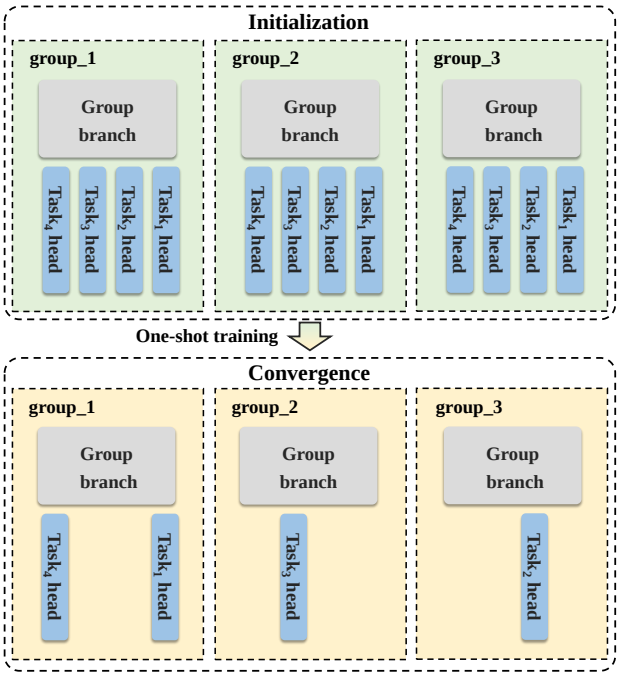



*Figure 1.* We formulate the Multi-Task Grouping (MTG) problem as network pruning. This figure illustrates the categorization of 4 tasks into 3 groups, where each branch represents a task group. As shown in the Upper Subfigure, at initialization, each group connects to all the task heads, ensuring full exploration of high-order task-affinity. Throughout MTG training, we simultaneously prune the task heads and train the weights of the group-specific branches. Our training process ensures that MTG converges to a categorization where each task exclusively and uniquely belongs to only one group, as illustrated in the Lower Subfigure.

[1]School of CS, Wuhan University [2]School of EI, Wuhan University [3]Tencent Youtu Lab [4]School of Automation Science and Engineering, South China University of Technology. Correspondence to: Gui-Song Xia <guisong.xia@whu.edu.cn>.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. 

It is thus critical to harness the affinity among those diverse tasks. Compared to learning them independently, simply combining them and feeding them into a fully shared network oftentimes deteriorates the performance of several or even most tasks. Such phenomenon is attributed to the presence of the inherent *negative transfer*, where the intuition is that the gradients from different tasks may interfere with each other when flowing into a shared encoder.

Pioneering works alleviate the negative transfer by designing novel Multi-Task Architectures (MTA) (Caruana, 1997; Argyriou et al., 2008; Kang et al., 2011; Ruder, 2017; Vandenhende et al., 2021) or applying the Multi-Task Optimization (MTO) methods (Sener & Koltun, 2018; Lin et al., 2019; Liu et al., 2021; Suteu & Guo, 2019; Yang & Hospedales, 2016), where state-of-the-art MTA assigns independent network parameters to different tasks, while MTO directly manipulates the gradients from different tasks before applying them to update the shared parameters. However, both MTA and MTO methods pose challenges when scaling to a large number of tasks, *i.e.*, the scalability is impeded in MTA for both training and evaluation due to the extra parameters, while the training of MTO cannot maintain scalability because it has to retain the backward graphs for each task. Recent researches also suggest that it is difficult to address the negative transfer solely by gradient manipulation in MTO (Xin et al., 2022; Kurin et al., 2022).

We instead propose to learn a large number of tasks by Multi-Task Grouping (MTG) (Standley et al., 2020). In MTG, input tasks are categorized into groups by their affinity, where a group of tasks, instead of a single task, is modeled by a unique encoder. When the group categorization is given, MTG for $K$ groups of $N$ tasks drastically reduces the training complexity from $O(N)$ (for MTA/MTO) to $O(K)$.

The primary challenge of MTG is to identify the group categorization, which involves investigating the exponential $2^N$ group candidates at maximum, given merely $N$ tasks. In order to migrate this issue, Standley et al. (2020) and Fifty et al. (2021) propose to average the pairwise affinities to approximate the high-order affinities[1]. Despite a reduced complexity from $2^N$ to $N^2$, the less precise assumption of linear tasks affinity in (Standley et al., 2020; Fifty et al., 2021) degrades the final performance. On the other hand, Song et al. (Song et al., 2022) advocate to train a meta-learner that directly predicts the final performance given a group categorization. However, the training of the meta-learner *per se* is extremely difficult and involves collecting numerous well-trained group samples. Moreover, existing methods perform group identification and grouped task learning in separated sequential procedures. As a result, the former potentially introduces objective bias *w.r.t.* the latter, especially when the groups are categorized based on heuristics. This also leads to potential performance degradation.

In view of those limitations, we propose to *formulate MTG as a pruning problem of an adaptive network architecture*, as shown in Figure 1, which enables to 1) *identify the best groups and train the grouped model weights simultaneously in one-shot*, as well as 2) *fully exploiting the high-order task affinities*. In our unified one-shot learning, we formulate the group identification as the model architecture learning/pruning, and the grouped task learning is established as the model weights training under a certain architecture. In this way, both procedures mutually facilitate each other to a better convergence. We jointly train both procedures simply by the task losses, where the high-order task affinities are directly exploited. Our approach excels in both efficiency and accuracy, which is distinct from pioneering two-shot methods that first approximately identify the grouping results, then train the grouped model from scratch subsequently.

Specifically, we formulate the categorization of $N$ tasks into $K$ groups as learning of a `Categorical` distribution, where the `Categorical` distribution is used to determine an adaptive network architecture. We then optimize the unified group identification and grouped task learning leveraging a pruning algorithm that is fully differentiable. To this end, our method starts with $K$ branches, each equipped with $N$ heads. It indicates that at the beginning, all the tasks are predictable by every group, ensuring full exploitation of the high-order task-affinity. After that, we optimize the model weights as well as the `Categorical` distribution such that the $KN$ heads are gradually pruned down to $N$, facilitating that each task is exclusively and uniquely predicted by only one branch. Our `Categorical` distribution is continuously relaxed and then optimized by Gumbel softmax (Maddison et al., 2016). Our pruning procedure *per se* is efficient, as we only expand the light-weighted *task heads* (*e.g.*, only the last network layers), instead of the heavy *encoders*, and the $K$ encoder branches (each for a group) in our method represent the minimal requirement of MTG.

Our method has been extensively validated on CelebA (Liu et al., 2015) and Taskonomy (Zamir et al., 2018) with detailed ablations. Our method exhibits two unique features:

- **Accuracy** with high-order task affinities exploited, which is ensured by 1) the grouping formulation of learning a continuous relaxed and differentiable `Categorical` distribution, and 2) the elimination of the objective bias by the one-shot training of unified group identification and grouped task learning.
- **Efficiency** with $O(K)$ training complexity given $K$ groups, which comes from 1) our pruning formulation instead of sampling group candidates to train from scratch, and 2) the one-shot training that unifies group identification and grouped task learning.

[1]For example, the performance of Task $A$ in Group $\{A, B, C\}$ is approximated by averaging that of $A$ in $\{A, B\}$ and $\{A, C\}$

## 2. Related work

### 2.1. Multi-Task Grouping

Multi-Task Grouping (MTG) aims to put collaborative tasks from a task pool into the same group, where a group of tasks can be learned efficiently by a shared network (Kang et al., 2011; Kumar & III, 2012; Li et al., 2021). Grouping tasks enables efficient learning of a vast array of tasks while also maintaining high interpretability. However, the primary challenge in MTG is that finding an optimal grouping solution in $2^N - 1$ grouping candidates can be difficult. Existing grouping methods (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022) have attempted to model an evaluation function to determine high-order task relationships based on low-order observations. Nonetheless, these methods perform group identification and grouped task learning separately, and potentially considering only low-order task affinity. In contrast, our grouping approach integrates group identification and grouped task learning within a one-shot training process, significantly improving running efficiency in large-scale task scenarios while thoroughly considering higher-order task relationships.

### 2.2. Multi-Task Architecture

Multi-Task Architecture (MTA) (Caruana, 1997; Argyriou et al., 2008; Kang et al., 2011; Ruder, 2017; Vandenhende et al., 2021) is a prevailing technology line in the Multi-Task Learning domain. It can be categorized as hard-parameter sharing (Kokkinos, 2017; Nekrasov et al., 2019; Dvornik et al., 2017; Bilen & Vedaldi, 2016; Zamir et al., 2016) and soft-parameter sharing (Gao et al., 2019; 2020; 2024; Long et al., 2015; 2017; Misra et al., 2016). The former shares a common feature extraction module among tasks, while the latter assigns a special feature extraction branch for each task, exchanging features through extra fusion modules. Although great success has been witnessed in designing novel MTL network architectures, they are less appropriate in addressing an extreme large number of tasks. Specifically, it is difficult to avoid the negative transfer due to a full-shared encoder module in hard-parameter sharing methods (Vandenhende et al., 2019; Guo et al., 2020; Brüggemann et al., 2020; Sun et al., 2020), while soft-parameter sharing methods (Ruder et al., 2019; Zhang et al., 2018; Xu et al., 2018; Zhang et al., 2019b) better address the negative transfer but introduce efficiency issues.

### 2.3. Multi-Task Optimization

Multi-Task Optimization (MTO) develops in parallel with Multi-Task Architecture, which aims to adjust task loss to balance the learning process of different tasks (Kendall et al., 2018; Chen et al., 2018c; Liu et al., 2019b; Lin et al., 2022; Chen et al., 2020; Guo et al., 2018). Advanced MTO methods directly manipulate gradients from different tasks to mitigate the training conflicts (Li & Gong, 2021; Wang & Tsvetkov, 2021; Liu et al., 2020; Yu et al., 2020), *e.g.*, projecting task gradients when their angle is greater than $90°$. In practice, revising gradients necessitates additional memory to store the gradient graph for each task, which can be potentially infeasible when dealing with an extremely large number of tasks. Most recently, Kurin et al. (2022) and Xin et al. (2022) reveal that the existing MTO methods may be sensitive to hyperparameters when dealing with different combinations of tasks. Our method aims to learn the categorization of tasks and is orthogonal to MTO methods.

### 2.4. Network Pruning

Network pruning (Cai et al., 2018; Chen et al., 2018b; Elsken et al., 2019; Ghiasi et al., 2019; He et al., 2020; Li et al., 2019) aims to detect and remove the redundancy of the networks without significant performance degradation. This pruning process can be implemented by Bayesian optimization (Bergstra et al., 2013), evolutionary algorithms (Real et al., 2019; Xie & Yuille, 2017), network transformation (Gordon et al., 2018), reinforcement learning (Guo et al., 2019; Tan et al., 2019; Zoph et al., 2018), and gradient descent (Akimoto et al., 2019; Liu et al., 2019a; Wu et al., 2019; Zhang et al., 2019a). We use differentiated pruning operations, which effectively enable integrating group identification with grouped task learning jointly in one-shot training. We are the first to implement network pruning into MTG to unify group identification and grouped task learning in an end-to-end architecture.

## 3. Method

Given $N$ tasks, we aim to efficiently chase the best categorization from the $2^N$ possibilities, with the high-order task affinities directly exploited. To this end, we *formulate MTG into a network pruning framework*, where we model the group identification as the architecture learning/pruning, and the grouped task learning as the model weights optimization under a certain architecture. As a result, the group identification and the grouped task learning are unified and can be jointly optimized in one-shot during the network pruning. Regarding the optimization, we design the group categorization as the learning of a `Categorical` distribution, which is then continuously relaxed into a differentiable `Concrete` distribution and subsequently optimized using the Gumbel softmax (Maddison et al., 2016).

In summary, our method is able to 1) exploit the high-order task affinities directly. 2) It avoids the potential objective bias when group identification and grouped task learning act as separated sequential procedures. 3) Given $K$ groups, our pruning algorithm preserves the efficiency of $O(K)$ training complexity for the encoder. 4) Our `Categorical` distri-

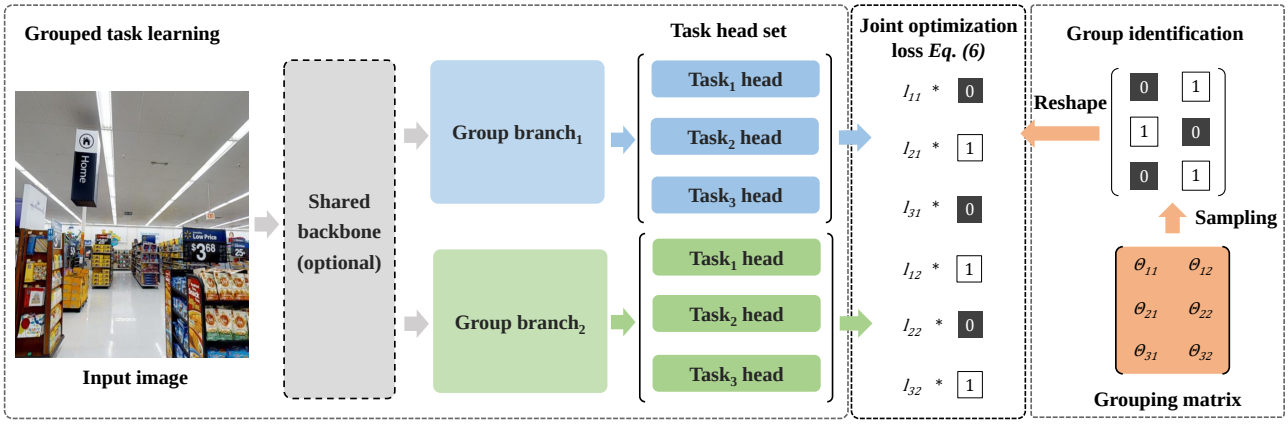



*Figure 2.* The overview of our method. We formulate the Multi-Task Grouping (MTG) problem as network pruning, where our method consists of a grouped task learning module and a group identification module. In order to categorize $N$ tasks into $K$ groups, our network is constructed with $K$ group-specific branches, optionally with shared lower layers. At initialization, we connect each branch to all the task heads (enabling them to predict all tasks), so that the high-order task-affinity can be exploited. We then formulate the grouped task learning as the model weights training for each group-specific branch, and the group identification as the network head pruning. The final grouped task losses are generated by the element-wise product of both modules, which in turn ensures both modules to be trained simultaneously in one-shot with the high-order task-affinity fully exploited. This figure illustrates categorizing 3 tasks into 2 groups.

bution formulation guarantees each task to be categorized into one group *exclusively and uniquely*. Thus, our learned groups and model weights are readily to use without retraining or validation (validation is needed when a certain task is categorized into multiple groups (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022), as discussed in Appendix F).

### 3.1. Problem Formulation

Formally, we consider categorizing a set of $N$ tasks $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_N\}$ into equal or less than $K$ groups $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_K\}$, such that each group contains 0 to $N$ tasks $\mathcal{G}_k = \{..., \mathcal{T}_i, ...\}$, and each task is exclusively and uniquely belongs to one group. Therefore, we have:

$$\begin{aligned} & \mathcal{T} = \cup_{k=1}^{K} \mathcal{G}_k, \\ s.t. \quad & \forall k, \quad |\mathcal{G}_k| \in \{0, ..., N\}, \\ & \forall (i,j), \quad \mathcal{G}_i \cap \mathcal{G}_j = \emptyset, \end{aligned} \tag{1}$$

where $|\cdot|$ is the cardinality. We optimize our problem exclusively to attain the highest average performance across these $N$ tasks, without relying on heuristic criteria. We also note that $K$ is the maximal-allowed number of groups, and we do not impose a strict requirement to yield precisely $K$ groups, *e.g.*, some groups may contain 0 task.

**Objective Bias in Two-Stage MTG Methods.** The objective bias in pioneering (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022) appears in two aspects: 1) *the group categorization is determined by heuristics but the retraining is based on the optimization of task losses*, and 2) *the difference in the inputs to the group identification and the grouped-model weights retraining stages lead to different objectives*. In other words, the groups are identified heuristically when all the $N$ tasks can synergy/regularize each other, but the retraining phase only sees a subset (group) of tasks thus exhibiting different gradients from the former.

As shown in Figure 2, the group identification and the grouped task (weights) learning in our method complement each other and are trained jointly in one-shot. On one hand, during the training of the task groups, the group identification module selects collaborative tasks to back-propagate gradients to the corresponding branch of the grouped task learning module. On the other hand, each branch of the grouped task learning module is responsible for one group, which in turn facilitates group identification.

### 3.2. Grouped Task Learning Module

We start with $K$ branches in the grouped task learning module, where each branch represents the encoder of each task group. We connect each branch to $N$ task heads to predict all the $N$ tasks, facilitating the exploration of high-order task affinity. Our method possesses an efficient training complexity of $O(K)$ for the network encoder.

Our method also enables to further reduce the training complexity, by implementing optional group-wise shared layers before splitting into the group-specific branches. This is illustrated in the dashed gray box in Fig. 2.

### 3.3. Group Identification Module

We model the categorization of $N$ tasks exclusively and uniquely into $K$ groups as the learning of an unknown `Categorical` distribution, where the `Categorical` distribution is used to determine an adaptive network architecture. As such, the underlying `Categorical` distribution can be optimized jointly with the model weights in one-shot, which we formulate as a pruning problem.

**`Categorical` Distribution Modeling.** Formally, let a random variable $z_{ik}$ indicate the assignment of task $i$ to group $k$, which is sampled from some discrete distribution. In order to assign $N$ tasks to $K$ groups, we have a set of random variables $Z = \{z_{ik}\} \in \mathbb{R}^{N\times K}$.

Recall Eq. (1) that each task is exclusively and uniquely categorized into one group, therefore, we have:

$$\sum_k z_{ik} = 1, \qquad \text{and} \qquad z_{ik} \in \{0,1\}, \quad \forall k, \tag{2}$$

which indicate that each row of $Z$ follows a `Categorical` distribution. Let the `Categorical` random variable $z_{ik}$ be parameterized by $s_{ik}$, we have:

$$z_{ik} \sim \texttt{Categorical}(s_{ik}), \tag{3}$$

where $s_{ik}$ is the probability of assigning task $i$ to group $k$.

**Network Architecture Formulation.** We establish an adaptive network architecture by the `Categorical` distribution, and formulate a pruning problem so that the Group Identification Module and the Grouped Task Learning Module can be optimized jointly in one-shot.

To this end, we formulate the sampled random variable $z_{ik}$ as a loss indicator or a task selector, which determines whether to back-propagate the loss of task $i$ to the $k$-th group. Specifically, let $L = \{L_{ik}\} \in \mathbb{R}^{N\times K}$ be the loss matrix of $KN$ task heads, the final loss can be obtained by:

$$L^{\text{task}}(\theta, S) = L(\theta) \odot Z(S), \tag{4}$$

where $\odot$ is the element-wise product, $\theta$ is the model weights, and $S = \{s_{ik}\} \in \mathbb{R}^{N\times K}$ is the set of parameters of the `Categorical` distributions.

As shown in Eq. 4, we formulate MTG as a pruning problem where $Z(S)$ is learned to prune the $KN$ losses $L(\theta)$. We note that the cost of training a $N \times K$ matrix $S$ and sampling $Z$ from $S$ is negligible *w.r.t.* the learning of $K$ group encoders, retaining the training complexity of our pruning formulation as $O(K)$ for the heavy encoder[2].

### 3.4. The Joint Optimization

Equation (3) involves a discrete sampling from $s_{ik}$ to $z_{ik}$, which results in a gradient blockage in Eq. (4) when back-propagating gradients from $Z$ to $S$. In this section, we continuously relax the discrete `Categorical` distribution, so that both the parameters for group identification $S$ and the weights for grouped task learning $\theta$ can be jointly optimized in one-shot by back-propagating the gradients from the task loss $L^{\text{task}}(\theta, S)$, through $Z(S)$ and $L(\theta)$, respectively.

**Continuous Relaxation.** By using the reparameterization trick from the `Concrete` distribution (Maddison et al., 2016), we are able to continuously sample $s_{ik}$ to produce $\tilde{z}_{ik}$ that approximate $z_{ik}$ of the `Categorical` distribution. This facilitates the gradient flow from $L^{\text{task}}(\theta, S)$ to $s_{ik}$ through $\tilde{z}_{ik}$. The reparameterized `Categorical` distribution is modeled by the differentiable Gumbel softmax:

$$\tilde{z}_{ik} = \frac{\exp((s_{ik} + g_{ik})/\tau)}{\sum_{m=1}^{K} \exp((s_{im} + g_{im})/\tau)} \tag{5}$$

where $g_{ik}$ is sampled from a `Gumbel` distribution, *i.e.*, $g_{ik} = -\log(-\log(\texttt{Uniform}(0,1)))$ (Maddison et al., 2016). $\tau$ is a small or annealing temperature, producing a discrete $\tilde{z}_{ik}$ after convergence as a good approximation of $z_{ik}$. Given $\widetilde{Z} = \{\tilde{z}_{ik}\} \in \mathbb{R}^{N\times K}$, the loss in Eq. (4) becomes:

$$L^{\text{relaxed task}}(\theta, S) = L(\theta) \odot \widetilde{Z}(S), \tag{6}$$

**Initialization.** We note that the parameter of the `Categorical` distribution, $s_{ik}$, can be initialized according to the prior knowledge of the task affinity. In our problem, we simply initialize each $s_{ik}$ to $1/K$, which implies that each task has an equal probability of being categorized into any group. In other words, we do not assume any task affinities and learn them in a fully data-driven manner. Based on that, we optimize our model by pruning the initial $KN$ task heads to $N$, where each task is exclusively and uniquely categorized into one group after convergence.

## 4. Experiments

In this section, we extensively validate our method on both **Taskonomy** (Zamir et al., 2018) and **CelebA** (Liu et al., 2015) datasets for various candidate groups. We detail the experimental setup in the following.

### 4.1. Experimental Setup

**Datasets.** We perform experiments on the Taskonomy dataset (Zamir et al., 2018) following (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022), and the CelebA dataset (Liu et al., 2015) following (Fifty et al., 2021). We use the official tiny train, validation, and test split of Taskonomy. The images from Taskonomy and CelebA are bilinearly downsampled to $256 \times 256$ and $64 \times 64$, respectively. Those datasets are introduced in detail in Appendix A.

**Benchmark Experiments.** We follow the experiment setups in (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022) to conduct 5 tasks on Taskonomy, *i.e.*, *semantic segmentation*, *depth estimation*, *surface normal*, *keypoint detection*, and *edge detection*, denoted as **Taskonomy-5**. We also conduct 9 tasks on CelebA dataset following (Fifty et al., 2021), *i.e.*, *5_o_Clock_Shadow*, *Black_Hair*, *Blond_Hair*, *Brown_Hair*, *Goatee*, *Mustache*, *No_Beard*, *Rosy_Cheeks*, and *Wearing_Hat*, referred to as **CelebA-9**. We perform

[2]Our training efficiency can be impeded in uncommon cases when the network heads are heavy, we discuss this in Appendix B.

the full 40 tasks of the CelebA dataset, *i.e.*, **CelebA-40**, in Appendix D, showcasing our scalability to numerous tasks.

**Network Backbone.** We use the same network backbone as (Standley et al., 2020; Fifty et al., 2021), *i.e.*, a variant of Xception network (Chollet, 2017) with 12 blocks, for the Taskonomy experiments. For CelebA, we use a variant of ResNet (He et al., 2016) following (Fifty et al., 2021).

**Optimization.** We use Adam optimizer for all of our experiments, where the initial learning rates are $0.0008$ and $0.0001$ for the CelebA and Taskonomy experiments, respectively. We use *plateau* learning rate decay which reduces by $0.5$ when the validation loss no longer improves. We train all the experiments for $100$ epochs, where our networks are initialized by the pre-trained naive MTL weights on the corresponding experiments. We copy the networks and the group-specific parameters for $K$ times to ensure that the same task is initialized identically across different groups. We initialize the Gumbel Softmax temperature $\tau$ of Eq. (5) as $2.5$ and $4$ for the CelebA and Taskonomy experiments, respectively. We follow (Fifty et al., 2021) to use the cross-entropy loss for the CelebA experiments, and follow (Standley et al., 2020; Fifty et al., 2021) to use the cross-entropy loss for semantic segmentation and $\ell_1$ loss for other tasks of the Taskonomy experiments.

**Evaluation Metrics.** Pioneering research in MTG commonly relied on the *total loss* as the evaluation metric (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022), which straightforwardly sums up the losses of all tasks. However, the magnitudes of losses from different tasks significantly vary due to 1) different loss types, such as cross-entropy losses for classification and $\ell_1$ losses for regression, and 2) diverse labels, such as image-level classification labels and pixel-level semantic segmentation labels. Consequently, simply calculating the *total loss* may lead to an overestimation of tasks with higher loss magnitudes while overshadowing those with lower loss magnitudes. This phenomenon contradicts the goal of MTG, *i.e.*, boosting *all the input tasks rather than a subset of them* (Standley et al., 2020).

To comprehensively assess improvements of an MTG method across all tasks, we follow (Maninis et al., 2019; Vandenhende et al., 2020; 2021) to eliminate the influence of loss magnitudes, which is termed **normalized gain**. Specifically, we initially calculate the normalized loss improvement (expressed as a percentage) *w.r.t.* the naive MTL architecture (*i.e.*, the shared-encoder architecture, oftentimes the worst baseline) for each task, then average them for all tasks:

$$\text{NormGain}_L = \frac{1}{N}\sum_{n=1}^{N}\frac{L^{\text{task } n}_{\text{Naive MTL}} - L^{\text{task } n}_{\text{method}}}{L^{\text{task } n}_{\text{Naive MTL}}}, \quad (7)$$

where $L$ denotes loss and $N$ is the total number of tasks. Similarly, in cases where a unified evaluation is applicable for all input tasks (*e.g.*, classification error when all input tasks are classifications), we can also present **normalized gain** *w.r.t.* such unified evaluation error:

$$\text{NormGain}_E = \frac{1}{N}\sum_{n=1}^{N}\frac{E^{\text{task } n}_{\text{Naive MTL}} - E^{\text{task } n}_{\text{method}}}{E^{\text{task } n}_{\text{Naive MTL}}}, \quad (8)$$

where $E$ denotes the unified evaluation error.

### 4.2. Experiments on Taskonomy with 5 Tasks

Following (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022), we compare our methods with the state-of-the-art MTG methods including **HOA** (Standley et al., 2020), **TAG** (Fifty et al., 2021), and **MTG-Net** (Song et al., 2022).

We also report the performance of Random Group (**RG**) which randomly divides the input tasks into a specific number of groups. We illustrate the baseline performance with **Naive MTL**, where all the tasks are trained simultaneously with a fully-shared encoder (*i.e.*, within 1 group). The performance where each task is trained separately without grouping is denoted as Single Task Learning (**STL**). We perform candidate numbers of groups as 3, 4, and 5.

The results in terms of losses are shown in Table 1. Our method outperforms SOTA methods by a large margin with a more efficient $O(K)$ training complexity for the encoder, we give detailed training time in Appendix B. Our method reduces the total loss by $22\%$ compared to naive MTL and by $13\%$ compared to STL when $K = 3$. As $K$ increases, our grouping performance further improves. Regarding the normalized metric NormGain *w.r.t.* loss, *i.e.*, Eq. (7), it also achieves a remarkable improvement of over $60\%$ *w.r.t.* naive MTL. Consistent observation can be obtained regarding the error statistics (*i.e.*, Eq. (8)), as shown in Appendix C.

It can be observed that the performances in terms of total loss and NormGain are not consistent for some MTG methods. For example, in Table 1 at $K = 3$, the **NormGain** of MTG-Net is over 10% higher than that of HOA, given that their **total losses** are comparative. This is because, as discussed in Evaluation Metrics of Sect. 4.1, the total loss is affected by loss magnitudes associated with different tasks, a slight improvement in a task with a large loss magnitude might overshadow a significant degradation in a task with a small loss magnitude. In contrast, the NormGain metrics address this issue by eliminating such undesirable influence through normalization, providing a more reasonable measurement *w.r.t.* the improvement of all the tasks. We further validate this by illustrating the loss and the relative gain *w.r.t.* Naive MTL *for each task* in Table 2, where our method achieves the best performance across almost all tasks.

We also show the training complexity of the heavy encoder relative to the Naive MTL method in Table 2. Given $K << N$, Our method achieves the best training efficiency except

| Groups | Methods | Depth Estimation | | Surface Normal | | Semantic Segmentation | | Keypoint Detection | | Edge Detection | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Loss ↓ | NormGain$_L$ (%) ↑ | Loss ↓ | NormGain$_L$ (%) ↑ | Loss ↓ | NormGain$_L$ (%) ↑ | Loss ↓ | NormGain$_L$ (%) ↑ | Loss ↓ | NormGain$_L$ (%) ↑ |
| - | Naive MTL | 8.67e-3 | - | 1.07e-1 | - | 8.28e-2 | - | 1.19e-2 | - | 1.31e-2 | - |
| - | STL | 1.60e-5 | +99.82 | 1.07e-1 | -0.18 | 9.16e-2 | -10.63 | 1.30e-4 | +98.91 | 1.56e-4 | +98.81 |
| $K=3$ | RG | 2.57e-2 | -195.88 | 1.08e-1 | -0.81 | 8.43e-2 | -1.88 | 6.87e-3 | +42.46 | 6.88e-3 | +47.45 |
| | HOA | 5.85e-3 | +32.47 | 1.11e-1 | -4.37 | 7.33e-2 | +11.49 | **2.00e-6** | **+99.98** | 8.60e-5 | +99.34 |
| | TAG | 5.15e-3 | +40.59 | 1.21e-1 | -12.93 | 8.43e-2 | -1.88 | **2.00e-6** | **+99.98** | 8.60e-5 | +99.34 |
| | MTG-Net | 2.04e-4 | +97.65 | **1.07e-1** | **+0.00** | 8.28e-2 | +0.00 | 6.39e-4 | +94.65 | 4.08e-4 | +96.88 |
| | Ours | **1.19e-7** | **+100.00** | 1.07e-1 | -0.05 | **6.65e-2** | **+19.64** | 4.30e-5 | +99.63 | **3.58e-7** | **+100.00** |
| $K=4$ | RG | 5.15e-3 | +40.59 | 1.07e-1 | +0.00 | 7.33e-2 | +11.49 | 1.19e-2 | +0.00 | 6.88e-3 | +47.45 |
| | HOA | 5.15e-3 | +40.59 | 1.06e-1 | +0.44 | 8.33e-2 | -0.61 | **2.00e-6** | **+99.98** | 8.60e-5 | +99.34 |
| | TAG | 5.15e-3 | +40.59 | 1.11e-1 | -4.37 | 7.33e-2 | +11.49 | **2.00e-6** | **+99.98** | 8.60e-5 | +99.34 |
| | MTG-Net | 2.04e-4 | +97.65 | 1.07e-1 | +0.00 | 8.28e-2 | +0.00 | 6.39e-4 | +94.65 | 4.08e-4 | +96.88 |
| | Ours | **1.19e-7** | **+100.00** | **1.05e-1** | **+1.43** | **6.46e-2** | **+21.96** | 4.70e-5 | +99.61 | **1.20e-5** | **+99.91** |
| $K=5$ | RG | 5.15e-3 | +40.59 | 1.07e-1 | +0.00 | 7.33e-2 | +11.49 | 6.87e-3 | +42.46 | 6.88e-3 | +47.45 |
| | HOA | 5.15e-3 | +40.59 | 1.06e-1 | +0.44 | 8.33e-2 | -0.61 | 2.00e-6 | +99.98 | 8.60e-5 | +99.34 |
| | TAG | 5.15e-3 | +40.59 | 1.11e-1 | -4.37 | 7.33e-2 | +11.49 | 2.00e-6 | +99.98 | 8.60e-5 | +99.34 |
| | MTG-Net | 2.04e-4 | +97.65 | 1.07e-1 | +0.00 | 8.28e-2 | +0.00 | 6.39e-4 | +94.65 | 4.08e-4 | +96.88 |
| | Ours | **1.19e-7** | **+100.00** | **1.05e-1** | **+1.62** | **6.34e-2** | **+23.43** | **1.00e-6** | **+99.99** | **4.17e-7** | **+100.00** |

*Table 2.* Performance (loss) on Taskonomy-5 *w.r.t.* each input task. Other parameters are the same as those in Table 1.

| Groups | Methods | Total Loss ↓ | NormGain$_L$ (%) ↑ | Relative Encoder Complex. |
|---|---|---|---|---|
| - | Naive MTL | 0.223 | - | 1 |
| - | STL | 0.199 | +57.35 | $O(N)$ |
| $K=3$ | RG | 0.231 | -21.73 | $O(K)$ |
| | HOA | 0.190 | +47.78 | $O(N^2)+O(K)$ |
| | TAG | 0.210 | +45.02 | $O(N)+O(K)$ |
| | MTG-Net | 0.191 | +57.83 | – |
| | Ours | **0.173** | **+63.85** | $O(K)$ |
| $K=4$ | RG | 0.204 | +19.90 | $O(K)$ |
| | HOA | 0.195 | +47.95 | $O(N^2)+O(K)$ |
| | TAG | 0.190 | +49.41 | $O(N)+O(K)$ |
| | MTG-Net | 0.191 | +57.83 | – |
| | Ours | **0.170** | **+64.58** | $O(K)$ |
| $K=5$ | RG | 0.198 | +28.40 | $O(K)$ |
| | HOA | 0.195 | +47.95 | $O(N^2)+O(K)$ |
| | TAG | 0.190 | +49.41 | $O(N)+O(K)$ |
| | MTG-Net | 0.191 | +57.83 | – |
| | Ours | **0.168** | **+65.01** | $O(K)$ |

*Table 1.* Experimental results on Taskonomy-5. We report the total loss and NormGain$_L$. NormGain$_L$ is the normalized gain according to the task loss calculated by Eq. (7). NormGain$_L$ and relative encoder complexity are measured *w.r.t.* Naive MTL.

for Naive MTL, but Naive MTL fails to deliver desirable accuracy through a fully shared encoder across all the tasks. Note that there are two terms for the training complexity of HOA and TAG, as they involve first identifying task groups according to $N$ tasks, then training $K$ networks, each for a task group, from scratch. The training complexity of MTG-Net is not included as it requires up to $O(2^N)+O(K)$ to sample up to $2^N$ task combinations and subsequent training them from scratch[3]. Note that as our method expands $N$ heads to $KN$, our efficiency can be impeded in uncommon cases where the network heads dominate the computations of the whole network, we discuss this in Appendix B.

### 4.3. Experiments on CelebA with 9 Tasks

We compare our method with the state-of-the-art methods **HOA** (Standley et al., 2020) and **TAG** (Fifty et al., 2021). **MTG-Net** (Song et al., 2022) is not included in the CelebA-9 experiments as MTG-Net does not scale well *w.r.t.* number of input tasks $N$, *i.e.*, MTG-Net requires to inefficiently sample up to $2^N$ task combinations and subsequent training them from scratch, which may take *thousands of GPU hours* as reported in (Song et al., 2022).

[3]For the Taskonomy-5 experiment, MTG-Net exhaustively samples 31 task combinations with 5 input tasks, as reported in (Song et al., 2022), resulting in a complexity of $O(2^N)+O(K)$.

| Groups | Methods | Total Error ↓ | NormGain$_E$ (%) ↑ | Relative Encoder Complex. |
|---|---|---|---|---|
| - | Naive MTL | 56.13 | - | 1 |
| - | STL | 59.93 | -8.70 | $O(N)$ |
| $K=2$ | RG | 54.87 | +1.06 | $O(K)$ |
| | HOA | 53.60 | +3.27 | $O(N^2)+O(K)$ |
| | TAG | 53.41 | +4.38 | $O(N)+O(K)$ |
| | Ours | **52.97** | **+5.75** | $O(K)$ |
| $K=3$ | RG | 54.57 | +1.54 | $O(K)$ |
| | HOA | 54.04 | +3.62 | $O(N^2)+O(K)$ |
| | TAG | 54.37 | +2.08 | $O(N)+O(K)$ |
| | Ours | **53.67** | **+4.64** | $O(K)$ |
| $K=4$ | RG | 54.57 | +1.54 | $O(K)$ |
| | HOA | 54.14 | +2.53 | $O(N^2)+O(K)$ |
| | TAG | 54.11 | +3.17 | $O(N)+O(K)$ |
| | Ours | **53.62** | **+4.62** | $O(K)$ |

*Table 3.* Experimental results on CelebA-9. As all the input tasks are classifications, we report the total classification error and NormGain$_E$, *i.e.*, the normalized gain according to the classification error calculated by Eq. (8). NormGain$_E$ and relative encoder complexity are measured in terms of Naive MTL.

Following (Fifty et al., 2021), we perform candidate numbers of groups as 2, 3, and 4 on CelebA-9. As all the input tasks in this experiment are classification tasks, therefore we report the total classification error and NormGain *w.r.t.* classification error, *i.e.*, Eq. (8), in Table 3. Table 3 illustrates the experiment results on CelebA-9, showing that our method consistently outperforms the state-of-the-art methods on the CelebA-9 experiments by a large margin with a more efficient training complexity of $O(K)$ for the encoder.

## 5. Ablation Analysis

We carefully investigate the following issues by ablation. 1) Whether our proposed one-shot MTG outperforms the common practice of two-shot methods (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022) given the same group categorization in Sect. 5.1. 2) Can our method generalize to the transformer backbones in Sect. 5.2. 3) The flexibility if we share more or less encoder layers in our method in Sect. 5.3. 4) Can our method scale to more input tasks in Appendix D. 5) The influence of different Gumbel Softmax temperatures in Appendix E. 6) The group categorization identified by different methods in Appendix F.

| Groups | Methods | Total Loss ↓ | NormGain$_L$ (%) ↑ |
|---|---|---|---|
| $K=3$ | Retrain from Scratch | 0.183 | +53.34 |
| | Retrain from Naive MTL Init. | 0.194 | +57.10 |
| | Ours (one-shot) | **0.173** | **+63.85** |
| $K=4$ | Retrain from Scratch | 0.183 | +53.34 |
| | Retrain from Naive MTL Init. | 0.194 | +57.10 |
| | Ours (one-shot) | **0.170** | **+64.58** |
| $K=5$ | Retrain from Scratch | 0.190 | +59.47 |
| | Retrain from Naive MTL Init. | 0.194 | +58.87 |
| | Ours (one-shot) | **0.168** | **+65.01** |

*Table 4.* Ablation on the merits of one-shot nature for MTG. We use the same group categorization on Taskonomy-5, as identified by our method, for all trials. We perform two double-shot MTG methods, which are retrained from *scratch* and *Naive MTL initialization*, respectively. Other parameters are the same as Table 1.

| Backbone | Methods | Total Loss ↓ | NormGain$_L$ (%) |
|---|---|---|---|
| ViT-Base | Naive MTL | 0.453 | - |
| | STL | 0.435 | +58.72 |
| | HOA | 0.379 | +62.22 |
| | TAG | 0.439 | +58.72 |
| | MTG-Net | 0.403 | +47.65 |
| | Ours | **0.326** | **+68.31** |

*Table 5.* Ablation on ViT-Base backbones of Taskonomy-5 with $K=3$. Other parameters are the same as Table 1.

### 5.1. Merits of One-shot Nature for MTG

The one-shot simultaneous group identification and grouped task learning is one of the key features of our method. Benefit from that, our method is able to 1) avoid the potential objective bias in the existing two-shot methods (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022), where group identification and grouped task learning act as separated sequential procedures, 2) further accelerate the training procedure as retraining from scratch is no longer needed.

In order to validate those benefits, we perform ablation on Taskonomy-5 using the group categorizations discovered by our method, and compare our method *w.r.t.* two double-shot methods, which are retrained from *scratch* and from *Naive MTL initialization*, respectively. As shown in Table 4, our one-shot method significantly outperforms both two-shot counterparts with the same group categorization. This suggests that the one-shot training strategy of our methods allows *the group branches to see more tasks at the early training stage than they have to predict in the end*, therefore the results of our one-shot method are better than those of training those grouped subsets of tasks from scratch.

### 5.2. Generalizing-ability to Transformer Backbone

We perform the Taskonomy-5 experiment by replacing the variant of Xception with ViT-Base backbone (Dosovitskiy et al., 2021). For HOA, TAG, and MTG-Net, we use the same group categorization as those in Table 1. The results are shown in Table 5, illustrating that our method can generalize to transformer backbones, and consistently outperforms our counterparts given the same network backbone.

| Shared Blocks | Total Loss ↓ | NormGain$_L$ (%) ↑ | FLOPs (G) |
|---|---|---|---|
| 0 | 0.174 | +63.72 | 56.89 |
| 3 | **0.166** | **+65.64** | 49.68 |
| 6 | 0.173 | +63.85 | 42.67 |
| 9 | 0.169 | +64.75 | **33.36** |

*Table 6.* Ablation on different amounts of shared backbone layers. The results are obtained on Taskonomy-5 with 12-block Xception backbone and $K=3$. Other parameters are the same as Table 1.

### 5.3. Flexibility with Amounts of Shared Layers

As shown in Fig. 2, our method enables to share the backbone encoder layers across different groups, which naturally introduces a flexible design regarding further training efficiency (*i.e.*, sharing more layers) and a better representation capability (*e.g.*, sharing specific layers).

We perform ablations with 0, 3, 6, 9 shared blocks on the 12-block Xception backbone of Taskonomy-5 with $K=3$. Table 6 shows that sharing different amounts of backbone layers happens to perform comparable with each other, where sharing 3 blocks out of 12 slightly outperforms other counterparts. We also note that sharing 9 blocks delivers good results with a significantly improved efficiency, *i.e.*, 58% FLOPs *w.r.t.* the model without any backbone sharing.

## 6. Discussion and Conclusion

**Limitations and Future Works.** Our method has the following two limitations. 1) The Gumbel Softmax is biased (Grathwohl et al., 2018; Tucker et al., 2017) and may be sensitive to different temperatures. We thus fix the temperature as a small value like (Chen et al., 2018a) to alleviate the bias issue, and we empirically show in Appendix E that different temperatures do not alter the performance significantly for our problem. We note that Gumbel Softmax is not our contribution and we will seek alternative optimizers in the future. 2) Our training efficiency can be impeded in uncommon cases where the network heads dominate the computations of the whole network (due to the expended $KN$ heads), as discussed in Appendix B. A more efficient training strategy for heavy heads is desirable in the future.

**Conclusion.** We tackle the challenges of Multi-Task Learning with numerous tasks using Multi-Task Grouping (MTG) techniques. Our approach efficiently identifies the best task groups from $2^N$ candidates given $N$ input tasks, with the high-order task affinity fully exploited. Moreover, our unified training approach of group identification and grouped task learning can be directly optimized using the task losses in one shot, which further improves the training efficiency and mitigates potential bias in separate training. We formulate MTG as a pruning problem, where the pruning process is also efficient as only the task heads are expended for pruning, instead of expanding the heavy encoders. We validate our methods on CelebA and Taskonomy datasets with extensive ablations. The results demonstrate the promising performance and the desirable efficiency of our method.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China (62306214, 62325111, 62076099), the Natural Science Foundation of Hubei Province of China (2023AFB196), and the Knowledge Innovation Program of Wuhan-Shuguang Project (2023010201020258).

## Impact Statement

This paper presents a novel one-shot differentiable method for general-purpose multi-task grouping. There is no potential societal consequence that must be specifically highlighted here.

## References

Akimoto, Y., Shirakawa, S., Yoshinari, N., Uchida, K., Saito, S., and Nishida, K. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *ICML*, pp. 171–180, 2019.

Argyriou, A., Evgeniou, T., and Pontil, M. Convex multi-task feature learning. *Machine learning*, 73:243–272, 2008.

Bergstra, J., Yamins, D., and Cox, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *ICML*, pp. 115–123, 2013.

Bertoni, L., Kreiss, S., and Alahi, A. Monoloco: Monocular 3d pedestrian localization and uncertainty estimation. In *ICCV*, pp. 6861–6871, 2019.

Bilen, H. and Vedaldi, A. Integrated perception with recurrent multi-task neural networks. In *NeurIPS*, volume 29, 2016.

Brüggemann, D., Kanakis, M., Georgoulis, S., and Gool, L. V. Automated search for resource-efficient branched multi-task networks. In *BMVC*, pp. 359, 2020.

Caesar, H., Bankiti, V., Lang, A., Vora, S., Liong, V., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, pp. 11621–11631, 2020.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2018.

Caruana, R. Multitask learning. *Machine learning*, 28: 41–75, 1997.

Chen, J., Song, L., Wainwright, M. J., and Jordan, M. I. Learning to explain: An information-theoretic perspective on model interpretation. In *ICML*, 2018a.

Chen, Y., Meng, G., Zhang, Q., Xiang, S., Huang, C., Mu, L., and Wang, X. Reinforced evolutionary neural architecture search. *arXiv preprint arXiv:1808.00193*, 2018b.

Chen, Z., Badrinarayanan, V., Lee, C., and Rabinovich, A. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, pp. 794–803, 2018c.

Chen, Z., Ngiam, J., Huang, Y., Luong, T., Kretzschmar, H., Chai, Y., and Anguelov, D. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *NeurIPS*, 33:2039–2050, 2020.

Cheng, B., Schwing, A., and Kirillov, A. Per-pixel classification is not all you need for semantic segmentation. *NeurIPS*, 34:17864–17875, 2021.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, pp. 1251–1258, 2017.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Dvornik, N., Shmelkov, K., Mairal, J., and Schmid, C. Blitznet: A real-time deep network for scene understanding. In *ICCV*, pp. 4154–4162, 2017.

Elsken, T., Metzen, J., and Hutter, F. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C. Efficiently identifying task groupings for multi-task learning. In *NeurIPS*, volume 34, pp. 27503–27516, 2021.

Gao, Y., Ma, J., Zhao, M., Liu, W., and Yuille, A. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *CVPR*, pp. 3205–3214, 2019.

Gao, Y., Bai, H., Jie, Z., Ma, J., Jia, K., and Liu, W. MTL-NAS: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *CVPR*, 2020.

Gao, Y., Zhang, W., Luo, W., Ma, L., Yu, J.-G., Xia, G.-S., and Ma, J. Aux-NAS: Exploiting auxiliary labels with negligibly extra inference cost. In *ICLR*, 2024.

Ghiasi, G., Lin, T., and Le, Q. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, pp. 7036–7045, 2019.

Godard, C., Aodha, A. M., Firman, M., and Brostow, G. Digging into self-supervised monocular depth estimation. In *CVPR*, pp. 3828–3838, 2019.

Gordon, A., Eban, E., Nachum, O., Chen, B., Wu, H., Yang, T., and Choi, E. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *CVPR*, pp. 1586–1595, 2018.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *ICLR*, 2018.

Guo, M., Haque, A., Huang, D., Yeung, S., and Li, F. Dynamic task prioritization for multitask learning. In *ECCV*, pp. 270–287, 2018.

Guo, M., Zhong, Z., Wu, W., Lin, D., and Yan, J. Irlas: Inverse reinforcement learning for architecture search. In *CVPR*, pp. 9021–9029, 2019.

Guo, P., Lee, C., and Ulbricht, D. Learning to branch for multi-task learning. In *ICML*, pp. 3854–3863. PMLR, 2020.

He, C., Ye, H., Shen, L., and Zhang, T. Milenas: Efficient neural architecture search via mixed-level reformulation. In *CVPR*, pp. 11993–12002, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., et al. Planning-oriented autonomous driving. In *CVPR*, pp. 17853–17862, 2023.

Kang, Z., Grauman, K., and Sha, F. Learning with whom to share in multi-task feature learning. In *ICML*, pp. 521–528, 2011.

Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, pp. 7482–7491, 2018.

Kokkinos, I. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, pp. 6129–6138, 2017.

Kumar, A. and III, H. D. Learning task grouping and overlap in multi-task learning. In *ICML*, pp. 1723–1730, 2012.

Kurin, V., Palma, A. D., Kostrikov, I., Whiteson, S., and Mudigonda, P. In defense of the unitary scalarization for deep multi-task learning. *NeurIPS*, 35:12169–12183, 2022.

Li, M., Gao, Y., and Sang, N. Exploiting learnable joint groups for hand pose estimation. In *AAAI*, volume 35, pp. 1921–1929, 2021.

Li, X. and Gong, H. Robust optimization for multilingual translation with imbalanced data. *NeurIPS*, 34:25086–25099, 2021.

Li, X., Zhou, Y., Pan, Z., and Feng, J. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *CVPR*, pp. 9145–9153, 2019.

Lin, B., Ye, F., Zhang, Y., and Tsang, I. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, 2022.

Lin, X., Zhen, H., Li, Z., Zhang, Q., and Kwong, S. Pareto multi-task learning. *NeurIPS*, 32, 2019.

Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q. Conflict-averse gradient descent for multi-task learning. *NeurIPS*, 34:18878–18890, 2021.

Liu, C., Chen, L., Schroff, F., Adam, H., Hua, W., Yuille, A., and Li, F. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, pp. 82–92, 2019a.

Liu, L., Li, Y., Kuang, Z., Xue, J., Chen, Y., Yang, W., Liao, Q., and Zhang, W. Towards impartial multi-task learning. In *ICLR*, 2020.

Liu, S., Johns, E., and Davison, A. End-to-end multi-task learning with attention. In *CVPR*, pp. 1871–1880, 2019b.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *ICCV*, pp. 3730–3738, 2015.

Long, D., Cohn, T., Bird, S., and Cook, P. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *ACL*, pp. 845–850, 2015.

Long, M., Cao, Z., Wang, J., and Yu, P. Learning multiple tasks with multilinear relationship networks. In *NeurIPS*, volume 30, 2017.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Maninis, K.-K., Radosavovic, I., and Kokkinos, I. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1851–1860, 2019.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Cross-stitch networks for multi-task learning. In *CVPR*, pp. 3994–4003, 2016.

Nekrasov, V., Dharmasiri, T., Spek, A., Drummond, T., Shen, C., and Reid, I. Real-time joint semantic segmentation and depth estimation using asymmetric annotations. In *ICRA*, pp. 7101–7107, 2019.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, pp. 4780–4789, 2019.

Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. Latent multi-task architecture learning. In *AAAI*, volume 33, pp. 4822–4829, 2019.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. *NeurIPS*, 31, 2018.

Song, X., Zheng, S., Cao, W., Yu, J., and Bian, J. Efficient and effective multi-task grouping via meta learning on task combinations. In *NeurIPS*, volume 35, pp. 37647–37659, 2022.

Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., and Savarese, S. Which tasks should be learned together in multi-task learning? In *ICML*, pp. 9120–9132, 2020.

Sun, X., Panda, R., Feris, R., and k. Saenko. Adashare: Learning what to share for efficient deep multi-task learning. In *NeurIPS*, volume 33, pp. 8728–8740, 2020.

Suteu, M. and Guo, Y. Regularizing deep multi-task networks using orthogonal gradients. *arXiv preprint arXiv:1912.06844*, 2019.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, pp. 2820–2828, 2019.

Tang, J., Li, S., and Liu, P. A review of lane detection methods based on deep learning. *Pattern Recognition*, 111:107623, 2021.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, D., and Sohl-Dickstein, J. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. In *NIPS*, 2017.

Vandenhende, S., Georgoulis, S., Brabandere, B. D., and Gool, L. V. Branched multi-task networks: Deciding what layers to share. *BMVC*, 2019.

Vandenhende, S., Georgoulis, S., and Van Gool, L. Mti-net: Multi-scale task interaction networks for multi-task learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pp. 527–543. Springer, 2020.

Vandenhende, S., Georgoulis, S., Gansbeke, W. V., Proesmans, M., Dai, D., and Gool, L. V. Multi-task learning for dense prediction tasks: A survey. *IEEE TPAMI*, 44 (7):3614–3633, 2021.

Wang, Z. and Tsvetkov, Y. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *ICLR*, 2021.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, pp. 10734–10742, 2019.

Xie, L. and Yuille, A. Genetic cnn. In *ICCV*, pp. 1379–1388, 2017.

Xin, D., Ghorbani, B., Gilmer, J., Garg, A., and Firat, O. Do current multi-task optimization methods in deep learning even help? *NeurIPS*, 35:13597–13609, 2022.

Xu, D., Ouyang, W., Wang, X., and Sebe, N. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *CVPR*, pp. 675–684, 2018.

Yang, Y. and Hospedales, T. Deep multi-task representation learning: A tensor factorisation approach. In *ICLR*, 2016.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning. *NeurIPS*, 33:5824–5836, 2020.

Zamir, A., Wekel, T., Agrawal, P., Wei, C., Malik, J., and Savarese, S. Generic 3d representation via pose estimation and matching. In *ECCV*, pp. 535–553. Springer, 2016.

Zamir, A., Sax, A., Shen, W., Guibas, L., Malik, J., and Savarese, S. Taskonomy: Disentangling task transfer learning. In *CVPR*, pp. 3712–3722, 2018.

Zhang, Y., Qiu, Z., Liu, J., Yao, T., Liu, D., and Mei, T. Customizable architecture search for semantic segmentation. In *CVPR*, pp. 11641–11650, 2019a.

Zhang, Z., Cui, Z., Xu, C., Jie, Z., Li, X., and J. Yang, J. Joint task-recursive learning for semantic segmentation and depth estimation. In *ECCV*, pp. 235–251, 2018.

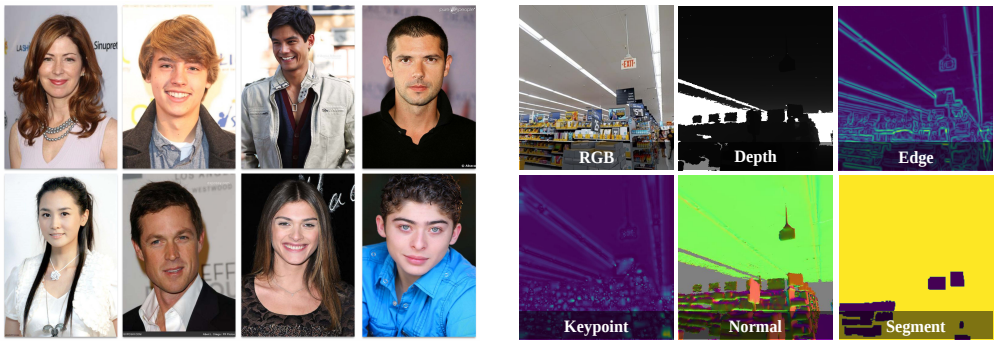
Zhang, Z., Cui, Z., Xu, C., Yan, Y., Sebe, N., and Yang, J. Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In *CVPR*, pp. 4106–4115, 2019b.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. Learning transferable architectures for scalable image recognition. In *CVPR*, pp. 8697–8710, 2018.

## A. Datasets

**CelebA** (Liu et al., 2015) is a large-scale face dataset that contains more than $200,000$ images from roughly $10,000$ identities, each of which has annotated $40$ face attributes representing the tasks to predict. Samples of CelebA datasets are illustrated in Fig. 1(a). For the CelebA experiments, the $9$ tasks implemented in CelebA-9 (Fifty et al., 2021) are *5_o_Clock_Shadow*, *Black_Hair*, *Blond_Hair*, *Brown_Hair*, *Goatee*, *Mustache*, *No_Beard*, *Rosy_Cheeks*, and *Wearing_Hat*. We also test our method with CelebA-40 using all the 40 labeled attributes of the CelebA dataset. All images in the CelebA dataset are resized to $64 \times 64$ resolution.

**Taskonomy** (Zamir et al., 2018) is one of the largest datasets with multi-task labels, covering 26 vision tasks from 2D to 3D. For the Taskonomy experiments, we use the official tiny train, validation, and test split with roughly $300,000$ images. The tasks used to learn the categorization are the same as those in (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022), which are *semantic segmentation*, *depth estimation*, *surface normal*, *keypoint detection*, and *canny edge detection*, as shown in Figure 1(b). We also follow (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022) to bilinearly downsample the Taskonomy images to $256 \times 256$ before training and testing.




(a) Illustration samples of the CelebA datasets (b) Illustration samples of the Taskonomy datasets

*Figure A1.* Illustration of the CelebA and Taskonomy datasets. Several image samples across different genders and races of CelebA dataset are shown in subfigure (a), while subfigure (b) exhibits an indoor RGB image with 5 annotated labels used in our experiments.

## B. The Training Time, FLOPs, and the Influence of Heavy Heads

We illustrate the training time of all the methods for different experiments and network backbones in Table A1. Table A1 shows that our method outperforms the previous arts in terms of both FLOPs and training time. We also note that for uncommon cases where the network heads dominate the computations of the whole network, *i.e.*, Taskonomy-9 experiments with the Xception Variant backbone, the efficiency of our method will be impeded by those heavy heads, as we expand $N$ heads to $KN$ (given $N$ tasks and $K$ groups). We leave the efficiency training strategy for heavy heads as our future work.

| Experiment | Network Backbone | Heads FLOPs Portion | Groups | HOA | | TAG | | Ours | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | FLOPs (G) | Time (h) | FLOPs (G) | Time (h) | FLOPs (G) | Time (h) |
| CelebA-9 | ResNet Variant | 0.32% | $K=2$ | 3.02 | 18.42 | 0.24 | 4.79 | **0.15** | **0.99** |
| | | | $K=3$ | 3.10 | 18.90 | 0.32 | 5.27 | **0.22** | **1.27** |
| | | | $K=4$ | 3.18 | 19.38 | 0.40 | 5.70 | **0.29** | **1.55** |
| Taskonomy-5 | Xception Variant | 59.73% | $K=3$ | 135.36 | 523.67 | 46.19 | 297.33 | **38.90** | **158.56** |
| | | | $K=4$ | 143.96 | 555.33 | 58.16 | 345.00 | **50.68** | **243.50** |
| | | | $K=5$ | 152.24 | 588.67 | 71.81 | 398.67 | **62.47** | **328.67** |
| Taskonomy-5 | ViT-Base | 28.44% | $K=3$ | 673.99 | 770.00 | 216.51 | 460.33 | **141.00** | **220.00** |
| | | | $K=4$ | 722.12 | 826.00 | 271.53 | 522.33 | **180.61** | **337.86** |
| | | | $K=5$ | 769.76 | 874.81 | 329.93 | 602.16 | **220.23** | **525.41** |

*Table A1.* Statistics of the training time and FLOPs for different experiments and network backbones. We illustrate the GFLOPs, and the training time (hour) is obtained on a single NVIDIA 4090 GPU. Xception Variant backbone is the same as that used in HOA (Standley et al., 2020) and TAG (Fifty et al., 2021), ResNet Variant backbone is the same as that used in TAG (Fifty et al., 2021). **Heads FLOPs Portion** is calculated by (the FLOPs of heads) / (the FLOPs of the whole network) of the **Naive MTL** architecture (*i.e.*, a fully-shared encoder with $N$ task heads).

## C. Results in Terms of Errors on the Taskonomy-5 Experiments

We used the loss metric in the main text (*i.e.*, Table 2) following the Taskonomy-5 experiments in the TAG paper (Fifty et al., 2021). In this section, we show the error statistics of the Taskonomy-5 experiments.

Specifically, we use mean Intersection over Union (**mIoU**) for semantic segmentation, Root Mean Square Error (**RMSE**) after aligning the transformation and scale for depth estimation, the **percent of vectors with an angle less than 30 degrees for surface normal**, **F1-score** for keypoint detection, and **F1-score** for edge detection. The results are shown in Table A2, demonstrating that our method remains the best for most tasks over the prior arts.

| Groups | Methods | Depth Estimation | | Surface Normal | | Semantic Segmentation | | Keypoint Detection | | Edge Detection | | Avg. $\text{NormGain}_E$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | RSME ↓ | $\text{NormGain}_E$ (%) ↑ | Acc. (< 30°) ↑ | $\text{NormGain}_E$ (%) ↑ | mIoU ↑ | $\text{NormGain}_E$ (%) ↑ | F1 ↑ | $\text{NormGain}_E$ (%) ↑ | F1 ↑ | $\text{NormGain}_E$ (%) ↑ | |
| - | Naive MTL | 8.67E-03 | - | 84.32 | - | 48.21 | - | 76.08 | - | 76.56 | - | - |
| - | STL | 1.59E-05 | 99.82 | 84.28 | -0.05 | 37.88 | -21.44 | 85.86 | 12.86 | 86.75 | 13.31 | 20.90 |
| K=3 | RG | 2.57E-02 | -195.87 | 84.35 | 0.03 | 42.84 | -11.13 | 78.71 | 3.47 | 79.71 | 4.11 | -39.88 |
| | HOA | 5.85E-03 | 32.48 | 83.56 | -0.90 | 47.25 | -1.99 | **86.86** | **14.17** | 87.07 | 13.73 | 11.50 |
| | TAG | 5.15E-03 | 40.59 | 82.00 | -2.76 | 42.84 | -11.13 | **86.86** | **14.17** | 87.07 | 13.73 | 10.92 |
| | MTG-Net | 2.04E-04 | 97.65 | 84.32 | 0.00 | 48.21 | 0.00 | 84.47 | 11.04 | 85.98 | 12.30 | 24.20 |
| | Ours | **1.19E-07** | **100.00** | **84.52** | **0.24** | **48.93** | **1.50** | 86.34 | 13.50 | **87.94** | **14.86** | **26.02** |
| K=4 | RG | 5.15E-03 | 40.59 | 84.35 | 0.03 | 48.21 | 0.00 | 76.08 | 0.00 | 79.71 | 4.11 | 8.95 |
| | HOA | 5.15E-03 | 40.59 | 84.39 | 0.08 | 48.46 | 0.51 | **86.86** | **14.17** | 87.07 | 13.73 | 13.82 |
| | TAG | 5.15E-03 | 40.59 | 83.56 | -0.90 | 47.25 | -1.99 | **86.86** | **14.17** | 87.07 | 13.73 | 13.12 |
| | MTG-Net | 2.04E-04 | 97.65 | 84.32 | 0.00 | 48.21 | 0.00 | 84.47 | 11.04 | 85.98 | 12.30 | 24.20 |
| | Ours | **1.19E-07** | **100.00** | **84.71** | **0.46** | **49.69** | **3.08** | 86.31 | 13.46 | **87.65** | **14.49** | **26.30** |
| K=5 | RG | 5.15E-03 | 40.59 | 84.35 | 0.03 | 48.21 | 0.00 | 78.71 | 3.47 | 79.71 | 4.11 | 9.64 |
| | HOA | 5.15E-03 | 40.59 | 84.39 | 0.08 | 48.46 | 0.51 | 86.86 | 14.17 | 87.07 | 13.73 | 13.82 |
| | TAG | 5.15E-03 | 40.59 | 83.56 | -0.90 | 47.25 | -1.99 | 86.86 | 14.17 | 87.07 | 13.73 | 13.12 |
| | MTG-Net | 2.04E-04 | 97.65 | 84.32 | 0.00 | 48.21 | 0.00 | 84.47 | 11.04 | 85.98 | 12.30 | 24.20 |
| | Ours | **1.19E-07** | **100.00** | **84.40** | **0.09** | **49.85** | **3.40** | **86.90** | **14.23** | **87.94** | **14.86** | **26.52** |

*Table A2.* Performance (error) on Taskonomy-5 *w.r.t.* each input task. Other parameters are the same as those in Table 1.

## D. Scalability to More Input Tasks

The training complexity for the encoder of our method is $O(K)$, indicating that our training complexity is only relevant to the group number $K$ rather than the task number $N$. This is distinct from the previous state-of-the-art methods (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022), and therefore naturally scale to an arbitrary number of input tasks[4].

We validate our method with a more challenging case of categorizing all the **40** tasks into 5 groups on the CelebA dataset, denoted as CelebA-40. For the same candidate groups $K = 5$, our method for CelebA-9 and CelebA-40 takes 1.7 and 2.9 GPU hours, respectively, on a single Nvidia 4090 GPU, demonstrating the scalability of our method *w.r.t.* more input tasks.

We illustrate the relative gain *w.r.t.* Naive MTL method for each task in Fig. A2. Compared with Table 3, the improvement of our method for CelebA-40 in Fig. A2 is not as significant as that for CelebA-9 in Table 3. This is attributed to the exponential $2^N$ difficulty in modeling the high-order task affinity, given a drastic increase of $N$ from 9 to 40. Despite that prior state-of-the-art methods suffer significant difficulties in training complexity when scaling to such an extreme case of CelebA-40, our method successfully improves most tasks for CelebA-40 by categorizing and optimizing collaborative tasks within the same group.

## E. Influence of Gumbel Softmax Temperatures

Gumbel Softmax (Maddison et al., 2016) is known to be biased (Grathwohl et al., 2018; Tucker et al., 2017) and can potentially be sensitive to different temperatures. In this section, we investigate different temperatures in Table A3, which demonstrates that, empirically, different temperatures do not alter the performance significantly for our problem.

## F. Discussion on Group Categorization

We illustrate the group categorization identified by different MTG methods in Table A4 and Table A5, for the Taskonomy-5 experiments of Table 1 and CelebA-9 experiments of Table 3, respectively.

We note that in the prior state-of-the-art HOA (Standley et al., 2020), TAG (Fifty et al., 2021), and MTG-Net (Song et al.,

[4]Our complexity would be relevant to $N$ only for uncommon cases where the network heads dominate the computation of the whole network, due to the expanded $KN$ heads as discussed in Appendix B.-* The complexities of the previous state-of-the-art methods are all determined by $N$, as shown in Table 1 and discussed in Footnote 3.

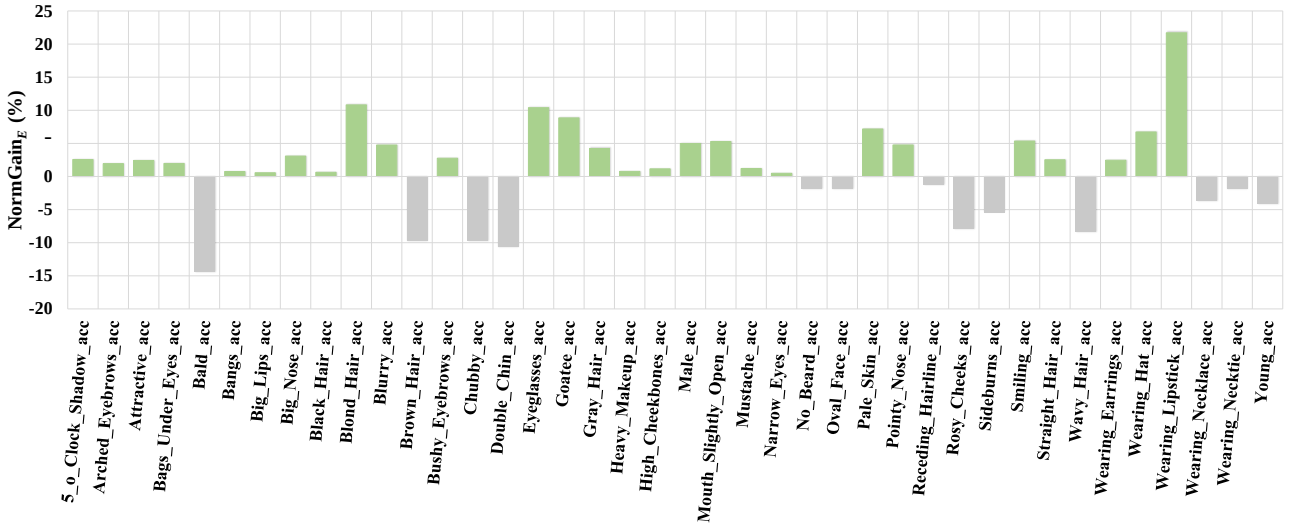



*Figure A2.* Normalized gain *w.r.t.* classification errors, NormGain$_E$, of our method in terms of Naive MTL for each task on the CelebA dataset with $N = 40$ tasks and $K = 5$ groups. Other parameters are identical to those in Table 3.

| Temperature Strategies of Gumbel Softmax | Total Loss |
|---|---|
| Annealing from 100 to 4 with a decay factor of 1/2 | 0.1656 |
| Annealing from 100 to 4 with a decay factor of 1/4 | 0.1665 |
| Annealing from 50 to 4 with a decay factor of 1/2 | 0.1730 |
| Annealing from 10 to 0.01 with a decay factor of 1/2 | 0.1710 |
| Annealing from 10 to 0.01 with a decay factor of 1/2 | 0.1710 |
| A fixed temperature of 4 | 0.1684 |

*Table A3.* Ablation on different Gumbel Softmax temperatures. The experiments are performed on Taskonomy-5 with the Xception variant backbone and $K = 5$. Other parameters are identical to those in Table 1.

2022), a given task could be assigned to multiple groups. In that case, only the best performance of the specific group categorization, instead of the averaged performance across all categorizations, is reported as the final result. This imposes an additional requirement of a carefully collected validation set to determine the best categorization for tasks categorized into multiple groups. In contrast, our method formulates group categorization as the learning of a `Categorical` distribution, which ensures that each task is categorized *exclusively and uniquely* into a single group. Consequently, our group categorization can be directly applied for testing, eliminating the need for a validation set to select the best categorization. In addition, the number of discovered groups may be less than the candidate input group number $K$ in our method, as our flexible `Categorical` distribution formulation allows each group to contain 0 to $N$ tasks, *i.e.*, Eq. (1).

We also note that the group categorizations are not consistent *w.r.t.* different targeting groups $K$, and this phenomenon appears for all the methods. For example, tasks A and B may be categorized into the same group given a certain $K$ but that categorization no longer holds when $K$ changes. One intuition behind this lies in the fact that the best task affinities, no matter the pairwise or high-order scenarios, are affected by different targeting groups $K$, especially when $K$ is significantly smaller than the number of input tasks $N$.

**Intuitions About the Grouped Tasks.** As shown in Table A4, our method has very similar grouping results as MTG-Net (Song et al., 2022) when $K = 5$ for the Taskonomy-5 experiment, where the surface normal and the semantic segmentation are categorized into the same groups. Intuitively, this might imply that as a large-scale indoor scene dataset, *most objects have planar surfaces (e.g., table desktop, wall, etc) in Taskonomy, therefore the surface normal (i.e., different planar surfaces) can be a good cue to identify different semantics (i.e., different objects)*. We further note that *the categorization of different tasks can be dataset-dependent* (e.g., the surface normal may no longer hold a good cue for the semantic segmentation when most objects do have planar surfaces in another dataset), and *our method is expected to well capture that dataset-dependent information through a fully data-driven manner*.

| Groups | Method | Categorizations | Depth estimation | Surface normal | Semantic segmentation | Keypoint detection | Canny edge detection |
|---|---|---|---|---|---|---|---|
| $K=3$ | HOA | group 1 | ■ | ■ | ■ | | |
| | | group 2 | | □ | | □ | |
| | | group 3 | | | | ■ | ■ |
| | TAG | group 1 | ■ | | ■ | | |
| | | group 2 | □ | ■ | | | |
| | | group 3 | | | | ■ | ■ |
| | MTG-Net | group 1 | □ | ■ | ■ | □ | □ |
| | | group 2 | □ | □ | | ■ | |
| | | group 3 | ■ | □ | | | ■ |
| | Ours | group 1 | | | | | |
| | | group 2 | ■ | | | | ■ |
| | | group 3 | | ■ | ■ | ■ | |
| $K=4$ | HOA | group 1 | | □ | | □ | |
| | | group 2 | | | | ■ | ■ |
| | | group 3 | | ■ | ■ | | |
| | | group 4 | ■ | □ | | | |
| | TAG | group 1 | □ | | □ | | |
| | | group 2 | ■ | □ | | | |
| | | group 3 | | | | ■ | ■ |
| | | group 4 | □ | ■ | ■ | | |
| | MTG-Net | group 1 | □ | ■ | ■ | □ | □ |
| | | group 2 | □ | □ | | ■ | |
| | | group 3 | ■ | □ | | | ■ |
| | | group 4 | □ | □ | | | |
| | Ours | group 1 | ■ | | | | |
| | | group 2 | | | | | |
| | | group 3 | | ■ | ■ | ■ | ■ |
| | | group 4 | | | | | |
| $K=5$ | HOA | group 1 | | □ | | □ | |
| | | group 2 | | | | ■ | ■ |
| | | group 3 | | ■ | ■ | | |
| | | group 4 | ■ | □ | | | |
| | | group 5 | | □ | | | |
| | TAG | group 1 | □ | | □ | | |
| | | group 2 | ■ | □ | | | |
| | | group 3 | | | | ■ | ■ |
| | | group 4 | □ | ■ | ■ | | |
| | | group 5 | □ | □ | □ | □ | |
| | MTG-Net | group 1 | □ | ■ | ■ | □ | □ |
| | | group 2 | □ | □ | | ■ | |
| | | group 3 | ■ | □ | | | ■ |
| | | group 4 | □ | □ | | | |
| | | group 5 | | | | | |
| | Ours | group 1 | | | | ■ | |
| | | group 2 | | | | | ■ |
| | | group 3 | ■ | | | | |
| | | group 4 | | | | | |
| | | group 5 | | ■ | ■ | | |

*Table A4.* Categorization results on Taskonomy-5 experiments with $K = 3, 4, 5$. The parameters are identical to those in Table 1. Note that the prior state-of-the-art methods, *i.e.*, HOA, TAG, and MTG-Net, may categorize a certain task into multiple groups. In that case, we count on the group with the best performance and report that in Table 1, which is denoted by the solid square ■. Otherwise, We denote by the hollow square □. While each task is categorized *exclusively and uniquely* into a single group in our method. We also note that by formulating MTG as the learning of a `Categorical` distribution, the flexibility of our method enables to finalize equal or less than $K$ groups, as we allow each group to contain 0 to $N$ tasks in Eq. (1).

| Groups | Methods | Categorizations | 5_o_Clock_Shadow | Black_Hair | Blond_Hair | Brown_Hair | Goatee | Mustache | No_Beard | Rosy_Cheeks | Wearing_Hat |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $K=2$ | HOA | group 1 | ■ | ■ | ■ | ■ | | ■ | | ■ | ■ |
| | | group 2 | | | | | ■ | | ■ | | |
| | TAG | group 1 | ■ | | | | | | ■ | | |
| | | group 2 | | ■ | ■ | ■ | ■ | ■ | | ■ | ■ |
| | Ours | group 1 | ■ | ■ | | ■ | | | | | ■ |
| | | group 2 | | | ■ | | ■ | ■ | ■ | ■ | |
| $K=3$ | HOA | group 1 | ■ | | ■ | | ■ | ■ | □ | ■ | ■ |
| | | group 2 | | ■ | | ■ | | | | | |
| | | group 3 | | | | | □ | | ■ | | |
| | TAG | group 1 | ■ | | | | | | □ | | |
| | | group 2 | | □ | ■ | | ■ | ■ | ■ | ■ | ■ |
| | | group 3 | | ■ | | ■ | | | | | |
| | Ours | group 1 | ■ | | | ■ | ■ | | ■ | | ■ |
| | | group 2 | | ■ | | | | ■ | | | |
| | | group 3 | | | ■ | | | | | ■ | |
| $K=4$ | HOA | group 1 | ■ | | | | □ | | | | |
| | | group 2 | | □ | ■ | | ■ | ■ | □ | ■ | ■ |
| | | group 3 | | ■ | | ■ | | | | | |
| | | group 4 | | | | | □ | | ■ | | |
| | TAG | group 1 | ■ | | | | | | ■ | | |
| | | group 2 | | □ | ■ | □ | | | □ | ■ | ■ |
| | | group 3 | | ■ | | ■ | | | | | |
| | | group 4 | | | | | ■ | ■ | | | |
| | Ours | group 1 | | | | | | | | | ■ |
| | | group 2 | ■ | | ■ | | | ■ | | | |
| | | group 3 | | | | ■ | | | | | |
| | | group 4 | | ■ | | | ■ | | ■ | ■ | |

*Table A5.* Categorization results on CelebA-9 experiments with $K = 2, 3, 4$. The parameters are identical to those in Table 3. The solid square ■ represents the group with the best performance, and the hollow square □ denotes the cases otherwise, when a certain task is categorized into multiple groups in HOA and TAG. Other conventions and explanations are the same as Table A4.