Efficient and Effective Multi-task Grouping via Meta Learning on Task Combinations

Xiaozhuang Song*
Southern University of Science and Technology
shawnsxz97@gmail.com

Shun Zheng
Microsoft Research
shun.zheng@microsoft.com

Wei Cao
Microsoft Research
wei.cao@microsoft.com

James J.Q. Yu
Southern University of Science and Technology
yujq3@sustech.edu.cn

Jiang Bian
Microsoft Research
jiang.bian@microsoft.com

Abstract

As a longstanding learning paradigm, multi-task learning has been widely applied into a variety of machine learning applications. Nonetheless, identifying which tasks should be learned together is still a challenging fundamental problem because the possible task combinations grow exponentially with the number of tasks, and existing solutions heavily relying on heuristics may probably lead to ineffective groupings with severe performance degradation. To bridge this gap, we develop a systematic multi-task grouping framework with a new meta-learning problem on task combinations, which is to predict the per-task performance gains of multi-task learning over single-task learning for any combination. Our underlying assumption is that no matter how large the space of task combinations is, the relationships between task combinations and performance gains lie in some low-dimensional manifolds and thus can be learnable. Accordingly, we develop a neural meta learner, MTG-Net, to capture these relationships, and design an active learning strategy to progressively select meta-training samples. In this way, even with limited meta samples, MTG-Net holds the potential to produce reasonable gain estimations on arbitrary task combinations. Extensive experiments on diversified multi-task scenarios demonstrate the efficiency and effectiveness of our method. Specifically, in a large-scale evaluation with 27 tasks, which produce over one hundred million task combinations, our method almost doubles the performance obtained by the existing best solution given roughly the same computational cost. Data and code are available at https://github.com/ShawnKS/MTG-Net.

1 Introduction

Multi-task learning (MTL), as a longstanding learning paradigm [37, 50], has been widely applied into a variety of machine learning applications, ranging from language understanding [11], visual recognition [33, 41], and robotic control [19] to drug discovery [36] and clinical therapeutics [17, 30]. The major motivation behind is to boost the performance of single-task learning (STL) by leveraging the additional supervision signals from other relevant tasks.

*Work is done during the internship at Microsoft Research Asia.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Nevertheless, naively grouping multiple tasks together for MTL easily results in severe performance degradation in practice. Many recent studies acknowledged this *negative-transfer* phenomenon and speculated that the major reason lied in the competition and interference among incompatible tasks [39, 47, 15, 41, 14, 30, 2, 6]. To mitigate such negative-transfer effect, recent years have witnessed a few explorations on developing new optimization approaches [39, 25, 29, 48] or model architectures [27, 33, 28, 15]. But explicitly identifying a group of tasks that would benefit from training together still remains a challenging fundamental problem without adequate investigations [41, 14]. Thoroughly studying this *multi-task grouping* (MTG) problem not only advances core research over MTL aside from optimization or architecture developments but also facilitates various MTL scenarios that can benefit from learning auxiliary objectives [2, 6].

However, identifying the best MTG solution is extremely challenging as it involves an exhaustive search over the whole space of task combinations, each of which corresponds to a whole procedure of model training and evaluation on its specific MTL objective. The cost of such an exhaustive search grows exponentially with the number of tasks and thus can be prohibitively expensive [6, 41, 14].

To reduce this prohibitive computational cost, most recent efforts [41, 14] relied on the assumption of high-order approximations (HOA), meaning *high-order performance gains can be estimated from corresponding pairwise gains by averaging*, where the performance gain denotes the improvement of MTL over STL for a specific task. For instance, considering the group composed of tasks {A, B, C}, they assumed that when learning with the joint objective of this group, the performance gain for A can be estimated by the average of associated pairwise gains for A, which were independently obtained by learning with {A, B} and {A, C}. Under the HOA assumption, they only need to collect the transferring gains for all pairwise combinations, of which the number is only quadratically dependent on the number of tasks. While substantially reducing computational requirements, the HOA assumption also results in wildly inaccurate estimations when there exist non-linear relationships between high-order gains and corresponding pairwise gains. As a result, such untrustworthy estimations can significantly hinder the final performance of searching for the optimal grouping option.

To pursue more accurate estimations of various high-order performance gains under affordable computational cost, we develop a systematic MTG framework with a new meta-learning problem on task combinations. To be specific, we divide all task combinations into two parts: one with a small number of combinations for meta training and the other with all the rest combinations for meta testing. Here the meta input is a specific task combination, and the meta objective is to predict the per-task performance gains (the meta label) for this combination. The underlying assumption of this meta-learning formulation is that *although the number of task combinations grow exponentially with the number of tasks, the relationships between task combinations and performance gains lie in some low-dimensional manifolds and thus can be learnable*. Following this formulation, we build a dedicated neural network, MTG-Net, as the meta learner, and further develop an active learning [35] strategy to progressively construct the meta-training set. With this strategy, we can largely improve the effectiveness of MTG-Net with only a small number of meta-training samples. After the meta-training stage, we leverage the gain predictions of MTG-Net on all task combinations to guide the final grouping selection. Extensive experiments on diversified multi-task scenarios, including vision, energy, and healthcare, demonstrate the efficiency and the effectiveness of our method. Moreover, we also visualize various latent structures discovered by MTG-Net, which empirically validate the practicability of our assumption.

In summary, our contributions include:

- To our best knowledge, this paper is the first effort to formulate a meta-learning problem for MTG and thus enable an efficient exploration of the exponentially growing space of task combinations, which was believed to be prohibitively challenging [41, 30, 2, 6].
- We build a neural meta learner, MTG-Net, that can capture the relationships between task combinations and performance gains effectively.
- We develop an active learning strategy to progressively select the most useful meta-training samples for efficient training of MTG-Net.
- We conduct extensive experiments across diversified multi-task scenarios. Specifically, in a large-scale evaluation (27 tasks) with over one hundred million task combinations, our method almost doubles the performance obtained by the existing solution.

2 Related Work

MTG The classic assumption of MTL in statistical learning was that putting related tasks into joint learning benefited from the inductive bias encouraging cross-task information sharing [8, 9, 7, 3, 5]. However, when this assumption of relatedness did not hold, the resulting *negative-transfer* effects can cause significant performance degradation. Thus the further research investigated the problem of simultaneously determining with which tasks each task should share while carrying over classical cross-task sharing [20, 22]. Nevertheless, these early MTL studies can hardly be adapted to modern deep neural networks [24] due to certain prohibitive assumptions [14]. As mentioned by [41, 14], identifying proper task groupings for deep neural networks conventionally required either computationally intensive cross-validation procedures or the human knowledge that is not always applicable to machine learning. The early attempt [41] on more general and systematic MTG for deep neural networks formalized a standard workflow: 1) collecting the transferring gains (or gain predictions) for all $2^N - 1$ task combinations, and then 2) conducting a brute-force search for the best grouping option given a specific budget, such as the maximal number of groups. The latter step is a deterministic searching procedure, but the former step is rather challenging because the optimal solution requires $2^N - 1$ times MTL training and evaluation, which can be prohibitively expensive when N increases. Thus [41] further proposed the HOA approximation to reduce the computational complexity from exponential into quadratic: $\binom{N}{2}$ times MTL on pairwise task combinations. Subsequently, [14] improved the efficiency of [41] by substituting $\binom{N}{2}$ times pairwise MTL procedures with a single run, in which the computation complexity is only linearly dependent on N due to maintaining and updating a task affinity matrix on the fly per each gradient updating. This further approximation, named as task affinity grouping (TAG) in [14], traded the accuracy in estimating pairwise MTL gains for efficiency, which may result in more error propagation from pairwise to higher-order. Different from all these HOA-based methods, this paper formulates a systematic meta-learning problem to estimate the transferring gains and builds an effective meta model to enable more accurate and robust generalization to massive high-order combinations.

Other Research on MTL & Meta Learning & Task Embedding Recent years witnessed plenty of explorations on MTL from other aspects, such as multi-objective optimization [39, 25, 29, 48], neural architecture search [28, 26, 16, 15, 42], and soft-sharing mechanisms [13, 33, 27]. While these studies have made remarkable progresses to improve MTL under a predefined task combination, the focus of this paper lies in *explicitly determining with which tasks to share*. Besides, as pointed by [37, 14], these two paradigms are complementary to each other and can contribute to performance improvements jointly. Moreover, our work falls into the paradigm of meta learning [43, 21, 4, 34, 40]. But different from traditional meta-learning studies, which primarily focused on the fast adaptation across tasks in few-shot scenarios, our meta objective is to estimate the performance gains for different task combinations. To fulfill this objective, we develop a meta network that treats a task combination as a set of task tokens and transforms them into task embeddings to learn their interactions. Meanwhile, we also note some existing studies leveraged a similar idea of learning task embeddings but for different purposes, such as inferring task similarities [1], estimating treatment effects [38], facilitating few-shot learning [10] or reinforcement learning [23], etc.

3 Meta Learning for Multi-task Grouping

In this section, we elaborate on how our meta learning framework works for MTG. First, in Section 3.1, we establish the basic notations and briefly introduce some crucial steps of MTG for better understanding. Then we formulate our meta learning problem on task combinations in Section 3.2. Next, we introduce the details of MTG-Net and the accompanied active learning strategy in Sections 3.3 and 3.4, respectively. Figure 1 gives an overview of our framework.

3.1 Preliminaries

Notations We adopt the following notations throughout this paper. N is the number of learning tasks in total. $\mathcal{T} = \{T_1, T_2, \cdots, T_N\}$ is the universal set of all N tasks, where T_j denotes the j-th task, and we have $|\mathcal{T}| = N$. $\mathcal{C} = \{C_1, C_2, \cdots, C_{2^N-1}\}$ is the universal set of all $2^N - 1$ combinations, which contains $\binom{N}{1}$ combinations containing only one task, $\binom{N}{2}$ combinations containing exact two tasks, $\binom{N}{3}$ three-task combinations, etc. $C_i = [t_1^i, t_2^i, \cdots, t_{|C_i|}^i]$ denotes the i-th task combination, where

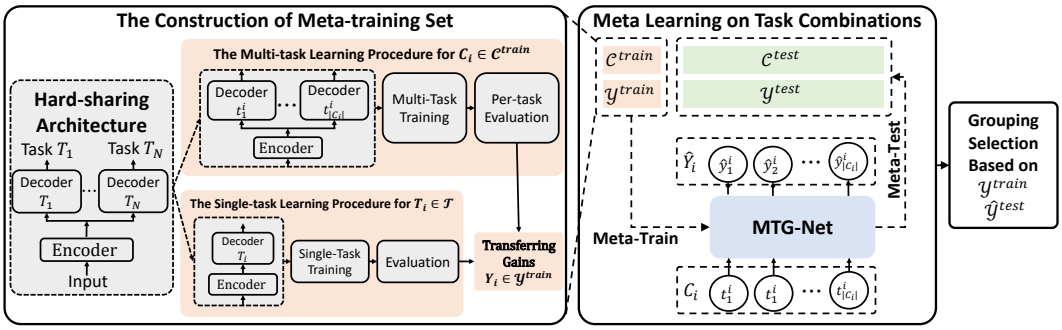


Figure 1: An overview of our meta learning framework for multi-task grouping

$t_j^i \in \mathcal{T}$ denotes the j-th task in this combination. $\mathcal{Y} = \{Y_1, Y_2, \cdots, Y_{2^N-1}\}$ is the universal set of all transferring gains corresponding to $\mathcal{C}$. $Y_i = [y_1^i, y_2^i, \cdots, y_{|C_i|}^i]$ includes all per-task transferring gains for task combination C_i, where y_j^i denotes the gain of conducting MTL on task combination C_i over performing STL for task t_j^i, and we have $|Y_i| = |C_i|$. $\mathcal{C}^{train}$ is the set of task combinations for meta training, and $\mathcal{Y}^{train}$ denotes the set of corresponding transferring gains. $\mathcal{C}^{test}$ is the set of task combinations for meta testing, and $\mathcal{Y}^{test}$ also includes corresponding transferring gains. B denotes the budget in grouping selection, which is the maximal number of selected task combinations.

The MTL Procedure Given task combination C_i, we define its MTL procedure as training with the joint objective of C_i and evaluating on the validation set for each task in C_i to collect per-task performance as an indication of generalization. In accordance with [41, 14], we adopt a widely used hard-sharing architecture for MTL, in which there is a single encoder to obtain shared representations followed by separate decoders for specific tasks. Moreover, we retain the same split of train, validation, and test sets for each MTL procedure and fix the optimization algorithm as well as other hyper-parameters. In this way, we can attribute the variations of generalization performance for a specific task largely to the different MTL objectives of various task combinations.

Transferring Gain Here we make more clarifications on collecting transferring gain y_j^i for task t_j^i by performing MTL on combination C_i. First, we conduct the aforementioned MTL procedure to collect the generalization performance on the validation set for task t_j^i in C_i. Then, we define the transferring gain of conducting MTL on C_i for t_j^i as the improvement of the performance of MTL over the base performance of STL for t_j^i. In practice, we can instantiate this improvement in different ways, such as the reduction of loss, the improvement in accuracy, etc.

Grouping Selection Once obtaining all transferring gains, we can select the optimal groupings that not only possess the best overall performance but also meet some practical considerations, such as inference latency. This selection procedure is in essence a constrained searching problem, which is to select a group of task combinations covering all N tasks to maximize the overall transferring gain (the average of per-task gain) while ensuring the number of groups does not exceed a given budget (B). As noted by [41, 14], this problem is NP-hard in general. A solution of recursive searching has the computational complexity of $O(2^{N \cdot B})$. In our experiments, when N is small (e.g., 5 vision tasks), we follow [41, 14] to use the branch-and-bound-like algorithm to search for the optimal groupings. While for large N and B (e.g., 27 medical tasks), the exact search becomes prohibitively expensive. In this case, we approximately select near-optimal groupings via a beam-search approach with polynomial complexity, which is attached in the appendix.

3.2 Meta Learning on Task Combinations

Different from all existing brute-force or heuristic approaches, we formulate a new meta-learning problem on task combinations as a more systematic framework for the estimations of transferring gains. As mentioned in preliminaries, a predefined configuration of the MTL procedure implies that the variations in transferring gains (Y_i) can be largely attributed to the MTL objectives of different task combinations (C_i). Therefore, we assume there is an underlying mapping $\mathcal{F}$ to determine the

transferring gains given a specific task combination: $Y_i = \mathcal{F}(C_i)$, for $C_i \in \mathcal{C}$. Furthermore, this view stimulates us to design a learning process to obtain a good approximation $\mathcal{F}^*$ to the oracle function $\mathcal{F}$ with limited meta samples from $\mathcal{C}$. The underlying assumption of the learnability with limited meta samples is that the relationships between task combinations and performance gains lie in some low-dimensional manifolds no matter how large the space of task combinations is. In our experiments, we empirically validate the practicability of this assumption.

To be specific, we divide the whole set $\mathcal{C}$ into two parts: one part $\mathcal{C}^{train}$ for meta training and the other $\mathcal{C}^{test}$ for meta testing. For every combination in $\mathcal{C}^{train}$, we conduct an MTL procedure and collect corresponding transferring gains ($\mathcal{Y}^{train}$) to learn a surrogate function $\mathcal{F}^*$. While for all combinations in $\mathcal{C}^{test}$, we directly apply $\mathcal{F}^*$ to produce estimated transferring gains, the set of which is denoted as $\hat{\mathcal{Y}}^{test}$. Accordingly, we summarize the whole meta-learning procedure as

$$\mathcal{F}^* = \texttt{Meta-Train}\big(\mathcal{C}^{train}, \mathcal{Y}^{train}\big), \quad \hat{\mathcal{Y}}^{test} = \texttt{Meta-Test}\big(\mathcal{C}^{test}, \mathcal{F}^*\big). \tag{1}$$

This meta-learning perspective provides an explicit trade-off between efficiency and effectiveness for MTG. On the one hand, as the size of $\mathcal{C}^{train}$ corresponds to the major computational cost, decreasing $|\mathcal{C}^{train}|$ helps improving efficiency. On the other hand, since the estimation performance on $\mathcal{C}^{test}$ largely determines the quality of the final grouping decisions, increasing $|\mathcal{C}^{train}|$ fosters effectiveness. Besides, given a fixed budget of $|\mathcal{C}^{train}|$, namely affordable computational cost, we can devote more efforts to develop advanced meta-learning strategies to improve the gain estimation. Moreover, this new view can incorporate the well-known HOA assumption as a special case. The HOA assumption groups all pairwise task combinations into $\mathcal{C}^{train}$ and takes the average of pairwise gains as the approximation of high-order gains for all other combinations in $\mathcal{C}^{test}$. Thus we can view it as an nonparametric method to represent the oracle mapping $\mathcal{F}$.

3.3 MTG-Net

In the following, we elaborate on the design of our meta learner. Specifically, we call this meta learner `MTG-Net`, namely a meta network to facilitate multi-task grouping. First, to provide rich representations for discrete task indicators, we introduce a classic technique from natural language research [32]: transforming discrete work tokens into vector representations while preserving semantic relationships. In our case, we also construct an embedding table $\mathbf{E} \in \mathbb{R}^{N \times D}$ to transform the input C_i into a vector of embeddings, denoted as $\mathbf{X}_i = \big[\boldsymbol{x}_1^i, \boldsymbol{x}_2^i, \cdots, \boldsymbol{x}_{|C_i|}^i\big] \in \mathbb{R}^{|C_i| x D}$, where D denotes the embedding dimension, and $\boldsymbol{x}_j^i \in \mathbb{R}^{1 \times D}$ corresponds to the embedding for task t_j^i.

Afterwards, to effectively capture the diversified interactions in different task combinations, we stack several self-attention encoding layers [45] over the dense representation $\boldsymbol{X}_i$. The reason is that the self-attention mechanism not only achieves remarkable successes in a wide variety of applications [12, 18] but also perfectly matches the unordered property of task combination and holds sufficient capacities to model those diversified inter-task interactions. Here we summarize this encoding procedure as $\boldsymbol{H}_i = \texttt{Self-Att-Enc}\big(\boldsymbol{X}_i\big)$, where $\boldsymbol{H}_i \in \mathbb{R}^{N \times D}$ is the vector of enriched representations that encode diversified interactions among tasks, and `Self-Att-Enc` is the self-attention-based encoder developed by [45]. Moreover, another benefit of `Self-Att-Enc` is that the encoded vector $\boldsymbol{H}_i$ shares the same length with the label vector Y_i. Naturally, we stack a linear mapping over $\boldsymbol{H}_i$ to obtain the final output as $\hat{Y}_i = \boldsymbol{H}_i \cdot \boldsymbol{w} + b$, where the vector $\hat{Y}_i = \big[\hat{y}_1^i, \hat{y}_2^i, \cdots, \hat{y}_{|C_i|}^i\big] \in \mathbb{R}^{|C_i|}$ corresponds to our estimation of Y_i, $\hat{y}_j^i$ corresponds to the estimated gain for t_j^i, $\boldsymbol{w} \in \mathbb{R}^D$ is a weight parameter, and $b \in \mathbb{R}^1$ is a scalar parameter being broadcasted to each entry of $\boldsymbol{H}_i \cdot \boldsymbol{w}$.

In summary, MTG-Net takes in a task combination (C_i) and emits a collection of gain estimations ($\hat{Y}_i$). Recalling Equation (1), we formulate the specific meta-learning procedure for MTG-Net as:

$$\begin{aligned} &\texttt{Meta-Train:} \quad \Theta^* = \arg\min_{\Theta} \sum_{C_i \in \mathcal{C}^{train}} \|\hat{Y}_i - Y_i\|_2^2, \\ &\texttt{Meta-Test:} \quad \hat{\mathcal{Y}}^{test} = \big\{\texttt{MTG-Net}_{\Theta^*}(C_i), \text{ for } C_i \in \mathcal{C}^{test}\big\}, \end{aligned} \tag{2}$$

where $\hat{Y}_i = \texttt{MTG-Net}_{\Theta}(C_i)$, Θ encapculates all parameters of MTG-Net, and Θ^* denotes a well-trained Θ. By replacing the computation-intensive MTL procedure with a network inference, we can collect the estimations of all transferring gains much more efficiently. Moreover, as long as the estimation performance is reasonable, we are still able to identify near-optimal groupings.

Algorithm 1: Active Learning for MTG-Net

Input: $\mathcal{C}^{train} = \{C_{2^N-1}\}$, $\mathcal{C}^{test} = \mathcal{C} \setminus \mathcal{C}^{train}$, K, α, η, Initialize Θ^*, $\hat{\mathcal{Y}}^{test}$ via (2)
for k ***in*** $[1, \cdots, K]$ **do**
  for j ***in*** $[1, \cdots, N]$ **do**
    $\mathcal{C}^{T_j} = \{C_i,$ **for** $C_i \in \mathcal{C}^{test}$ **if** $T_j \in C_i\}$
    $\mathcal{G}^{T_j} = \{\hat{Y}_i(T_j),$ **for** $C_i \in \mathcal{C}^{T_j}\}$
    $\mathcal{P}^{T_j} = \{\exp^{\alpha \cdot |G_i|},$ **for** $G_i \in \mathcal{G}^{T_j}\}$ // Build the sampling distribution
    $C_{next} = \texttt{Sample}_{\mathcal{P}^{T_i}}(\mathcal{C}^{T_i})$ // Select the next meta sample
    $\mathcal{C}^{train}$`.insert`(C_{next})
    $\mathcal{C}^{test}$`.remove`(C_{next}) // Update the meta-training set
    if $(k \cdot N - N + j) \,\%\, \eta == 0$ **then**
      Update Θ^*, $\hat{\mathcal{Y}}^{test}$ via (2) // Update MTG-Net and its predictions
    end
  end
end
Output: Θ^* and $\hat{\mathcal{Y}}^{test}$

3.4 Active Learning for MTG-Net

Given the meta-learning procedure (2) for MTG-Net, however, the construction of $\mathcal{C}^{train}$ is still a critical yet unsolved issue, which can have huge influences on the generalization capability of MTG-Net. To enable more efficient learning, we develop an active learning [35] strategy for MTG-Net. Here the key intuition is that paying more attention to the combinations with large gains can foster more efficient estimations of $\mathcal{Y}$ and also better facilitate the final grouping selection. Algorithm 1 includes the pseudo code of this strategy. First, we start with an MTG-Net trained on one meta sample that corresponds to the last task combination (C_{2^N-1}), which includes all N tasks. Next, we conduct K rounds of active selections. For each round, we repeat the same active learning process for all tasks. For each task (T_j), we first filter out a candidate set ($\mathcal{C}^{T_j}$) that incorporates all combinations including this task. Then for every combination (C_i) in $\mathcal{C}^{T_j}$, we collect the estimated transferring gain ($G_i = \hat{Y}_i(T_j)$) for this task, and we set the (unnormalized) sampling probability as $\exp^{\alpha \cdot |G_i|}$, where α is the hyper-parameter that measuring our preferences for remarkable gains (no matter positive or negative). Afterwards, we select a meta sample (C_{next}) following the distribution defined by $\mathcal{P}^{T_j}$, insert it into the meta-training set, and obtain updated Θ^*, $\hat{\mathcal{Y}}^{test}$ by re-running the meta procedure for MTG-Net if meeting a given updating interval (η).

Discussions on Efficiency Algorithm 1 includes $O(KN)$ times MTL procedures, which take up the major computational cost because the meta training in (2) takes negligible time compared with the corresponding MTL procedures to collect $\mathcal{Y}^{train}$. Besides, another issue that prevents from scaling to large N lies in the exponential dependence of $|\mathcal{C}^{test}|$ on N. In practice, we can tackle this problem by randomly sampling a group of combinations with a large yet affordable size to substitute $\mathcal{C}$. Moreover, since the encoder part of a hard-sharing architecture usually takes up the most computation, the computational cost of N-task MTL procedures does not have orders of magnitude differences for different N. Thus in this work we employ the number of MTL procedures as an approximate yet intuitive measure of computational cost. While as an anonymous reviewer mentioned, if each task has its specific data, the real computational time of MTL procedures should be considered carefully.

4 Experiments

We conduct extensive experiments to validate the effectiveness and the efficiency of our method across diversified multi-task scenarios, including vision, energy, and healthcare. In this section, we only include basic setups, main experimental results, and a part of visualization analyses. Due to the space limit, we leave other details (datasets, task specifications, networks for MTL, hyper-parameters, etc.) and more results (more visualizations, case studies, etc.) to the appendix.

4.1 Experimental Setups

MTL Datasets We conduct experiments on three MTL datasets. 1) Taskonomy [49] is a computer vision dataset including massive indoor scenes. Existing HOA-based methods [41, 14] leveraged 5 vision tasks of this dataset to evaluate MTG. We also follow the setups in [41] and denote the preprocessed dataset as `Taskonomy-5`. 2) ETTm1 [46] is an electric load dataset with 7 time series. We follow the multi-variate time-series forecasting problem in [46] and regard the forecasting for each series as a task. Accordingly, we denote this dataset as `ETTm1-7`. 3) MIMIC-III [17] is a healthcare database with rich electronic health records. Previous studies [30] constructed tens of clinical prediction tasks and found that simply performing MTL on these tasks did not always bring performance gains. We select 27 crucial clinical tasks to build the `MIMIC-III-27` dataset.

Transferring Gains & Grouping Selection Section 3.1 introduces how to collect ground-truth transferring gains and how to perform the final grouping selection given transferring gains (or predictions). Here we make some detailed specifications for each MTL dataset. For 5 vision tasks on `Taskonomy-5`, we follow [41, 14] to use the loss value as the metric and thus calculate the transferring gain as the relative reduction of the loss. For 7 forecasting tasks on `ETTm1-7`, we instantiate the transferring gain as the relative reduction of the mean absolute error. While for 27 clinical prediction tasks on `MIMIC-III-27`, all of which adopt AUROC as the metric, we define the transferring gain as the relative improvement of AUROC. Note that all these transferring gains are collected on the validation set, and we conduct the MTL procedure for each case with different random seeds. Only after conducting the grouping selection based on transferring gains, we report the performance on the test set for selected groupings. Moreover, we collect all ground-truth transferring gains for 31 task combinations on `Taskonomy-5` and 127 task combinations on `ETTm1-7`. However, 27 tasks on `MIMIC-III-27` produce over one hundred million task combinations, so we randomly sample $3,000$ combinations to represent the whole space. All these MTL procedures cost thousands of GPU hours in total, and we will release the collected meta datasets for future research.

Hyper-parameters of MTG-Net We use the same hyper-parameters for all MTL datasets. To be specific, we set the dimension of task embeddings as $D = 64$ and stack 2 self-attention encoding layers [45]. As for Algorithm 1, we set α as 25 to prioritize the selection of task combinations with large gains. Besides, we leverage a dynamic strategy to schedule η. At an early stage when $|\mathcal{C}^{train}| <= N+1$, we set η as 1 to frequently updating `MTG-Net` to pursue more effective selections. When $|\mathcal{C}^{train}| > N+1$, we set η as N to reduce the number of updating for `MTG-Net` to further improve efficiency. K is the hyper-parameter deciding the total number of meta-training samples, which is specified along with each figure. Moreover, we repeat the meta training and inference of `MTG-Net` with five random seeds and report the average results to eliminate the effects of randomness.

Baselines We compare `MTG-Net` with four baselines. 1) `Oracle` denotes the grouping solution based on all ground-truth transferring gains on the validation set, which forms an upper bound of all approximation methods. 2) `Random` denotes randomly selecting a group of task combinations satisfying a given budget. Since this solution has high variance, we conduct one million random trials and report the average performance, as did in [41]. 3) `HOA` [41] conducted $\binom{N}{2}$ MTL procedures on pairwise combinations to collect ground-truth pairwise gains and then estimated all rest high-order gains via the average of corresponding pairwise gains. 4) `TAG` [14] attempted to improve the efficiency of HOA by obtaining an approximation to the pairwise gains via a customized MTL procedure on all tasks, which involved N times extra forward and backward processes per each gradient updating to collect pairwise affinities on the fly.

4.2 Main Experiments

We present our main evaluation results for all three datasets in Figure 2. For each dataset, we include two configurations of K for `MTG-Net`: $K = 1$ and $K = \frac{N-1}{2}$. As illustrated in Section 3.4, our active learning strategy requires $O(KN)$ times MTL procedures. Therefore, these two types of K roughly correspond to the computational cost required by `TAG` and `HOA`, respectively.

In the left side of Figure 2, we show the final performance (on the test set) of selected groupings under different budgets. As the budget increases, meaning we have more combinations for a task to choose as the affiliated group, the final performance should also increase for a well-behaved

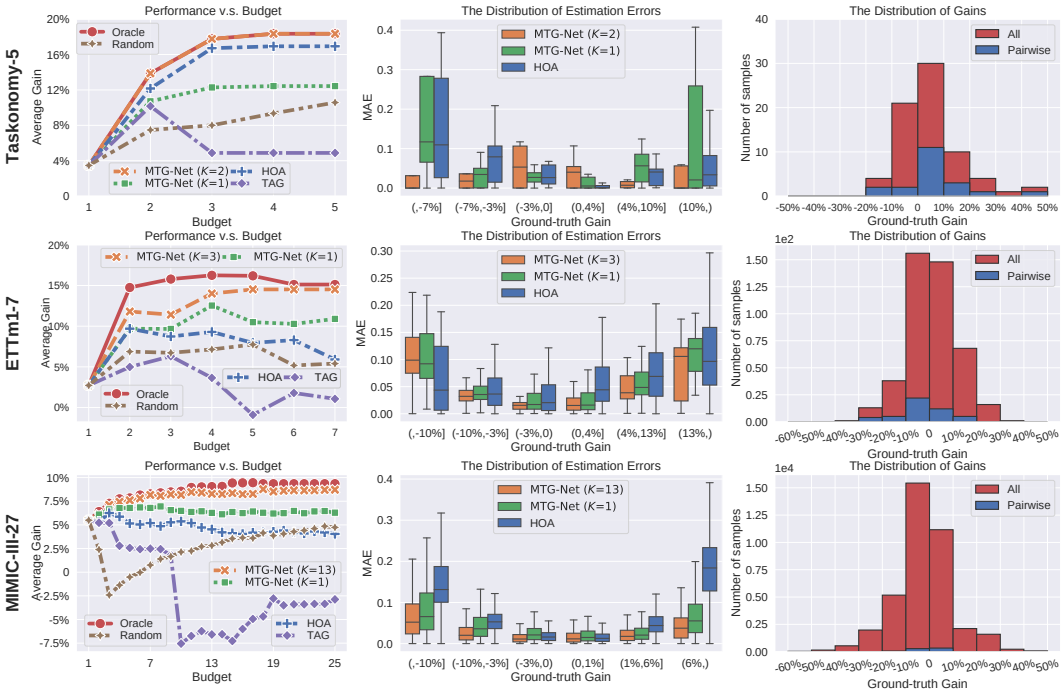
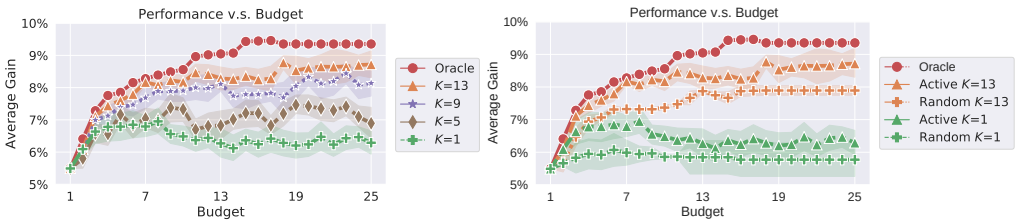


Figure 2: We show the final performance of selected groupings under different budgets (left column), the distribution of gain estimation errors (middle column), and the distribution of ground-truth gains (right column) on `Taskonomy-5`, `ETTm1-7`, and `MIMIC-III-27`.

MTG solution. However, we observe some fluctuation and even degradation effects. There are two reasons: 1) inaccurate estimations of transferring gains can largely misguide the grouping selection, and the grouping errors become much more evident as the budget increases; 2) we use the transferring gains collected on the validation set to guide the grouping selection, but these gains are not completely consistent with the actual performance gains on the test set. The latter reason also apply to `Oracle`, but its performance increases with the budget at most time. Thus, we can conclude that the degradation effect influenced by the latter reason is limited, and it is reliable to attribute the final grouping performance to the quality of the estimations for transferring gains.

Furthermore, we can find that `HOA` and `TAG` produce poor grouping decisions on `ETTm1-7` and `MIMIC-III-27`, while `HOA` can obtain pretty well grouping performance on `Taskonomy-5`, and `TAG` can produce reasonable performance when the budget is small. The key reason is that the number of task combinations increases exponentially as the number of tasks increases, as a result, `HOA` and `TAG` fail to provide accurate estimations for massive high-order task combinations solely based on the information of pairwise combinations. As for `Taskonomy-5`, there are large transferring gains covered by pairwise combinations, so `HOA` can obtain reasonable grouping performance. However, since `TAG` introduces another approximation step to predict pairwise affinities, its grouping performance suffers more degradation than `HOA` as the budget increases. In contrast, we can see that `MTG-Net` behaves much better on all cases. Specifically, when we set $K = \frac{N-1}{2}$, which means we only have the ground-truth gains on $\frac{N(N-1)}{2} + 1$ task combinations, `MTG-Net` is able to produce high-quality groupings that are very close to the ones derived by `Oracle`. In addition to the grouping performance, we also show the estimation errors of transferring gains (calculated on the validation set) on all task combinations in the middle column of Figure 2 and plot the distribution of ground-truth transferring gains in the right side. These auxiliary information aligns with the analyses mentioned above and further reveal the drawbacks of the HOA assumption that maps pairwise gains to high-order ones.

Given the superiority of `MTG-Net` over existing solutions, we further conduct experiments on the much challenging `MIMIC-III-27` benchmark to verify two critical factors that drive the success of `MTG-Net`. One is the number of meta-training samples. Figure 3a includes the performance comparisons of multiple `MTG-Net` counterparts with different K. Here we conduct multiple runs for each counterpart and plot the average performance as well as the standard deviation (shadow



(a) The impact of K.

(b) Active selection of $\mathcal{C}^{train}$ v.s. random selection

Figure 3: Different configurations of `MTG-Net` on `MIMIC-III-27`.

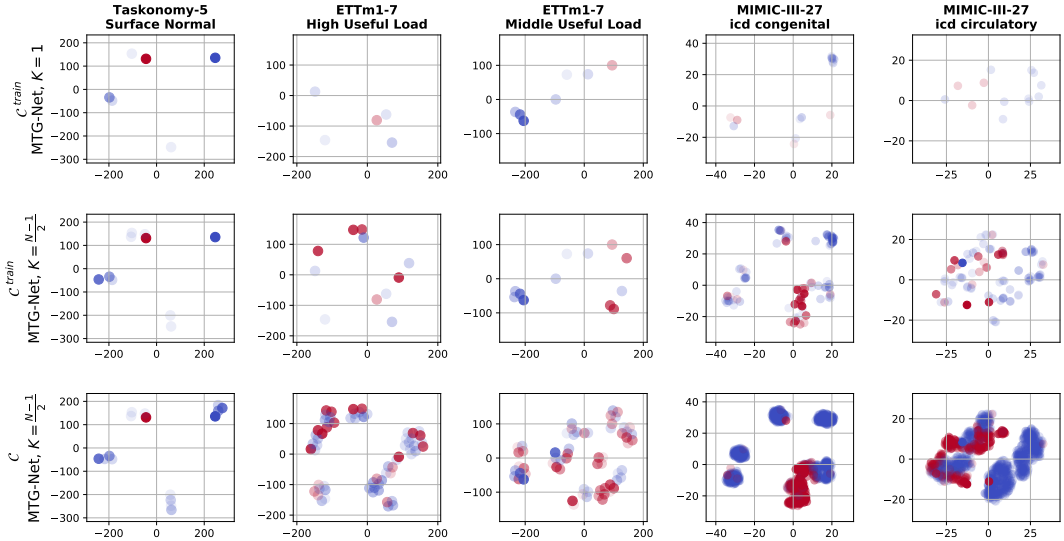


Figure 4: We select five tasks from different datasets and visualize the latent structures of transferring relationships discovered by `MTG-Net`. For each task, we aggregate its encoded embeddings (from $\boldsymbol{H}_i$) for all relevant task combinations and project them into a two-dimensional space via t-SNE [44]. The upper two rows only include task combinations on $\mathcal{C}^{train}$ for two Ks. The bottom row includes all combinations covering this task. We color each combination by its ground-truth gain for this task (red: positive; blue: negative; the deeper the color depth, the larger the gain (absolute value)).

area). We can clearly see that adding meta-training samples brings prominent improvements in the grouping performance for large budgets. While for small budgets, even $K = 1$ can produce reasonably good performance. This observation can help us to balance efficiency and effectiveness flexibly according to specific practical scenarios. Besides, the other crucial factor is the active learning strategy illustrated in Algorithm 1. Figure 3b includes the ablation study on the importance of active learning. We can observe that no matter K is large (13) or small (1), the active selection of meta samples produces remarkable improvements over the random selection.

4.3 Visualization Analyses

Figure 4 visualizes some latent structures of transferring relationships discovered by `MTG-Net`. By inspecting the upper two rows, we can clearly see that `MTG-Net` discovers a more comprehensive latent structure as the active learning strategy progressively increases $|\mathcal{C}^{train}|$ from $N+1$ to $\frac{N(N-1)}{2}+1$. From the bottom two rows, we can observe that the latent structure discovered on $\mathcal{C}^{train}$ is roughly consistent with the overall structure on $\mathcal{C}$. This observation intuitively reveals that the transferring relationships indeed lie in some low-dimensional manifolds, and our active learning strategy is effective in selecting those crucial combinations, which can function as the landmarks of the latent structure. Moreover, by comparing different latent structures across different tasks, we can observe that no matter how the MTL scenario and the number of task combinations vary, `MTG-Net` can still provide a reasonable approximation to the underlying latent structure.

5 Conclusion and Future Work

This paper introduces an efficient and effective approach for MTG, which is based on a new meta-learning formulation on task combinations. The key takeaway is that no matter how large the space of task combinations is, the relationships between task combinations and performance gains lie in some low-dimensional manifolds. Our experiments across diversified MTL scenarios demonstrate the practicability of this hypothesis. This is why we can make predictions for massive unseen task combinations with only ground-truth gains on a few actively selected combinations. The direct impact of this work is to benefit a wide range of real-world applications [6, 19, 36, 31], in which people have multiple learning tasks but do not know how to organize them into different groups effectively.

Moreover, we note that there are some valuable future research directions given the contribution of this work, demonstrating that the transferring relationships of a predefined MTL procedure across different task combinations could be meta learned. First, when the data or the model architectures changed, the transferring effects among tasks could also distinctly change. Both HOA [41] and us have observed these phenomena, which implies that multi-task transferring relationships may be a function of the data, the model, and some other factors (such as optimization algorithms). Thus how to enable effective meta learning on task combinations across different configurations of MTL procedures is a valuable research question to be answered. Besides, this work only considers the transferring relationships among a fixed set of tasks, while in practice we would always encounter new tasks. Accordingly, how to enable incremental meta learning or endow `MTG-Net` with the zero-shot capability is worthy of great research attention.

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *ICCV*, 2019.

[2] Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. Muppet: Massive multi-task representations with pre-finetuning. In *EMNLP*, 2021.

[3] Rie Kubota Ando, Tong Zhang, and Peter Bartlett. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 2005.

[4] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 2016.

[5] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine learning*, 2008.

[6] Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. Ext5: Towards extreme multi-task scaling for transfer learning. In *ICLR*, 2022.

[7] Jonathan Baxter. A model of inductive bias learning. *JAIR*, 2000.

[8] Richard Caruana. Multitask learning: A knowledge-based source of inductive bias. In *ICML*, 1993.

[9] Richard Caruana. Multitask learning. *Machine learning*, 1997.

[10] Xiaohan Chen, Siyu Tang, Krikamol Muandet, et al. Mate: Plugging in model awareness to task embedding for meta learning. In *NeurIPS*, 2020.

[11] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[13] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *ACL*, pages 845–850, 2015.

[14] Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. In *NeurIPS*, 2021.

[15] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In *ICML*, 2020.

[16] Siyu Huang, Xi Li, Zhi-Qi Cheng, Zhongfei Zhang, and Alexander Hauptmann. GNAS: A greedy neural architecture search method for multi-attribute learning. In *ACM-MM*, 2018.

[17] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 2016.

[18] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

[19] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[20] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *ICML*, 2011.

[21] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, 2015.

[22] Abhishek Kumar and Hal Daumé III. Learning task grouping and overlap in multi-task learning. In *ICML*, 2012.

[23] Lin Lan, Zhenguo Li, Xiaohong Guan, and Pinghui Wang. Meta reinforcement learning with task embedding and shared policy. In *IJCAI*, 2019.

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.

[25] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. In *NeurIPS*, 2019.

[26] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018.

[27] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Philip S Yu. Learning multiple tasks with multilinear relationship networks. In *NeurIPS*, 2017.

[28] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *CVPR*, 2017.

[29] Pingchuan Ma, Tao Du, and Wojciech Matusik. Efficient continuous pareto exploration in multi-task learning. In *ICML*, 2020.

[30] Matthew McDermott, Bret Nestor, Evan Kim, Wancong Zhang, Anna Goldenberg, Peter Szolovits, and Marzyeh Ghassemi. A comprehensive evaluation of multi-task learning and multi-task pre-training on ehr time-series data. *arXiv preprint arXiv:2007.10185*, 2020.

[31] Matthew McDermott, Bret Nestor, Evan Kim, Wancong Zhang, Anna Goldenberg, Peter Szolovits, and Marzyeh Ghassemi. A comprehensive ehr timeseries pre-training benchmark. In *CHIL*, 2021.

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013.

[33] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016.

[34] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *ICML*, 2017.

[35] Michael Prince. Does active learning work? a review of the research. *Journal of engineering education*, 2004.

[36] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

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

[38] Shiv Kumar Saini, Sunny Dhamnani, Akil Arif Ibrahim, and Prithviraj Chavan. Multiple treatment effect estimation using deep generative model with task embedding. In *WWW*, 2019.

[39] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *NeurIPS*, 2018.

[40] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[41] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *ICML*, 2020.

[42] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. In *NeurIPS*, 2020.

[43] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[44] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[46] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In *NeurIPS*, 2021.

[47] Sen Wu, Hongyang R Zhang, and Christopher Ré. Understanding and improving information transfer in multi-task learning. In *ICLR*, 2019.

[48] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *NeurIPS*, 2020.

[49] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.

[50] Yu Zhang and Qiang Yang. A survey on multi-task learning. *TKDE*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See the extra experiments provided in the appendix.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Datasets & Preprocessing

We perform our experiments using three datasets from three different domains: MIMIC-III, ETTm1 and Taskonomy. The description of these three datasets are as follows:

Taskonomy Taskonomy is currently the largest multi-task dataset in the computer vision domain. It's a diverse dataset with data collected from the 3D scans of near 600 buildings. There are more than 4 million instances in the dataset. we follow [41, 14] to conduct the splits of train, validation, and test sets within $3,923,543$ training instances, $45,715$ validation instances and $48,362$ test instances. Additionally. All labels used in the Taskonomy dataset were normalized to have zero mean and unit standard deviation. The five vision tasks on the Taskonomy dataset are: *Semantic Segmentation*, *Depth Estimation*, *Surface Normal*, *Keypoint Detection*, and *Canny Edge Detection*. Though the number of combinations is limited for the Taskonomy dataset, due to over ten million parameters in the network and about four million images for training, it is still time-consuming to finish all MTL procedures, which take $5,472$ GPU hours in total.

ETTm1 ETTm1 dataset contains the data collected from an electricity transformer from a region of China. It contains 6 power load series and an oil temperature series that are recorded every 15 minutes range from July 2016 to July 2018. The 6 power load series are: HF (High useFul load), HL (High useLess load), MF (Middle useFul load), ML (Middle useLess load), LF (Low useFul load), LL (Low useLess load). We use the observations from all series as input, and treat a specific series output as a single forecasting task. We follow [46] to split the dataset into training, validation and test set in chronological order by the ratio of $6:2:2$. In total, the construction processes of the meta-learning data ($\mathcal{C}$, $\mathcal{Y}$) on Taskonomy cost 346 GPU hours.

MIMIC-III The Medical Information Mart for Intensive Care (MIMIC-III) dataset collects the de-identified electronic heal records (EHRs) of patients admitted to critical care units at a Boston-area hospital from 2001–2012. Regarding the MIMIC-III dataset, as noted by [30, 31], the medical scenarios usually did not hold enough data compared with machine learning applications in natural language or computer vision. Accordingly, MTL can play a key role in those scenarios to boost the performance of machine learning models. Therefore, we follow the few-shot setup in [30, 31] to construct the training data. In total, we evaluated the experiments on EHR records collected from patients. We divided the dataset into 1755 training instances, 500 validation instances, and 500 test instances. We follow [30, 31] to extract the data from MIMIC-III and construct the train, validation, and test sets correspondingly. [31] defined 12 task categories, each of which includes one or multiple specific tasks. For instance, there are 19 major icd codes, and each one corresponds to a binary prediction task. We conduct MTL procedures for 27 one-task combinations, 351 two-task combinations (for HOA-based methods), and $3,000$ randomly selected high-order combinations (as an approximation to $\mathcal{C}$). The selected tasks are provided in Table 1. A task that has a name with format icd $[\text{diagosename}]$ means this is a task for predicting the icd code diagose's presence. Other tasks' definitions are all follow [30, 31]. In total, the construction processes of the meta-learning data ($\mathcal{C}$, $\mathcal{Y}$) on MIMIC-III cost $4,138$ GPU hours.

We denote the processed datasets as `Taskonomy-5`, `ETTm1-7`, and `MIMIC-III-27`, respectively. We conduct all corresponding MTL procedures to collect the transferring gains on the validation set and record the final performance on the test set. During the MTL procedure for each task combination, we restart the training with five random seeds and report the average performance to eliminate the randomness in neural network training.

B MTL Backbone Models & Hyperparameters

In this section, we provide our experimental details for reproducibility, including the backbone models used for different datasets and their hyperparameter settings. All experiments are implemented in PyTorch and conducted on a single NVIDIA V100 16GB GPU.

For `MIMIC-III-27`, we use the model proposed in [31]. It leverages a two-layer GRU encoder for all tasks while each task has a task-specific fully-connected layer as the decoder. The input sequence length is set to 48. The number of GRU layers is 2, and the hidden size of the GRU layer is 512. The learning rate is set to 0.001. The dropout rate of GRU layers is 0.1. The batch size is 512.

Datasets	Selected Tasks
Taskonomy-5	*Semantic Segmentation* (s) , *Depth Estimation* (d) , *Surface Normal* (n) , *Keypoint Detection* (k) , and *Edge Detection* (e)
ETTm1-7	High useFul load(HF) , High useLess load(HL) , Middle useFul load(MF) , Middle useLess load(ML) , High useFul load(LF) , High useLess load(LL) , Oil Temperature(OT)
MIMIC-III-27	*icd infection* (t_1) , *icd neoplasms* (t_2) , *icd endocrine* (t_3) , *icd blood* (t_4) , *icd mental* (t_5) , *icd nervous* (t_6) , *icd circulatory* (t_7) , *icd respiratory* (t_8) , *icd digestive* (t_9) , *icd genitourinary* (t_{10}) , *icd pregnancy* (t_{11}) , *icd skin* (t_{12}) , *icd musculoskeletal* (t_{13}) , *icd congenital* (t_{14}) , *icd ill defined* (t_{15}) , *icd injury* (t_{16}) , *dis 24h* (t_{17}) , *dis 48h* (t_{18}) , *mor 24h* (t_{19}) , *mor 48h* (t_{20}) , *los* (t_{21}) , *rea* (t_{22}) , *acu* (t_{23}) , *dnr 24h* (t_{24}) , *dnr 48h* (t_{25}) , *cmo 24h* (t_{26}) , *cmo 48h* (t_{27})

Table 1: Tasks for Taskonomy-5, MIMIC-III-5, MIMIC-III-27.

We leverage an early stopping strategy to train the model and the early stopping patience is 5. The final results of different task combinations on MIMIC-III-27 dataset are the average results of 5 random seeds. We utilize cross-entropy loss for all tasks on MIMIC-III-27.Its code is available at: https://github.com/mmcdermott/comprehensive_MTL_EHR.

For Taskonomy-5, we follow the model used in [41]. It leverages a modified Xception network as the encoder, which has 36 convolutional layers forming the feature extraction base. As for task-specific decoders, it uses four transposed convolutional layers and four convolution layers. We follow the hyperparameters of Xception encoder used in [41] for Taskonomy-5 experiments, which modifies the original proposed Xception network for a low inference time. It replaces the max-pooling layers with 2×2 convolution layers with a stride of 2, the input image size is 256×256. The learning rate is set to 0.1, and the batch size is 16. For model training, we follow [41], which trains the model for 100 epochs and picks the model that has the best performance on the validation set. The final results of different task combinations on Taskonomy-5 dataset are the average results of 3 random seeds. We utilize cross-entropy loss for *Semantic Segmentation* and $L1$ loss for all other tasks for model optimization. Its code is available at: https://github.com/tstandley/taskgrouping.

For ETTm1-7, we follow the model used in [46]. It leverages an encoder-decoder model with decomposition capabilities and an approximation to attention based on Fourier transform. The input sequence length is set to 96, and the prediction length is set to 24. The number of encoder layers is 2, and the number of the decoder layer is 1. The number of attention heads in Autoformer's auto-correlation layers is 8. The dropout rate is 0.05. The hidden dimension of embedding in encoder layers and decoder layers is 512, and the dimension of the fully-connected layer is 2048. The learning rate is set to 0.0001 and the batch size is 32. We leverage an early stopping strategy to train the model and the early stopping patience is 10. The final results of different task combinations on ETTm1-7 dataset are the average results of 5 random seeds. We utilize $L2$ loss for all tasks on ETTm1-7. Its code is available at: https://github.com/thuml/Autoformer.

C Grouping Selection Algorithm

For Taskonomy-5 and ETTm1-7, we follow the brute-force search solution used in [41, 14]. Considering the overwhelming computational cost of applying the brute-force search on MIMIC-III-27, we utilize a beam search algorithm as an alternative. The beam search algorithm keeps track of *multiple best* search sequences with a specific beam size rather than *one* search sequence compared to greedy search algorithm. It provides a tradeoff between accuracy *v.s.* computational cost via its flexible choice of the beam size. The pseudo-code of the beam search algorithm is provided at Algorithm 2.

D Additional Experiments & Analyses

We provide additional experiments and experimental analyses in this section. Generally, we provide the case studies on three aspects: (1) We provide case studies on the task level gains, which includes the per-task gains on each dataset with different methods; (2) We provide further visualization results

Algorithm 2: Beam Search for Grouping Selection

Input: A set of task combinations $\mathcal{Y}$, all task combinations $\mathcal{Y}_a$, budget c_0, number of search beginning points β, remaining budget c_r, beam size γ. Initially, $c_r = 1$.

```
Function beamSearch(Y_a, c_0, β):
    runningList ← best β candidates in Y_a;
    while c_r ≤ c_0 do
        for List in runningList do
            Y = Filter(Y_a, List);
            while Y is not empty do
                X ← Y.pop ;
                newList ← List ∪ X ;
                runningList ← runningList ∪ newList
            end
        end
        runningList ← Best γ candidates in runningList;
        c_r = c_r + 1;
    end
    S ← ∅;
    for List ∈ runningList do
        S ← Better(S, List);
    end
    return S
Function Filter(Y, List):
    Remove candidates that cannot improve List's performance on any task.
Function Better(S_1, S_2):
    Score_1 ← total gain of S_1;
    Score_2 ← total gain of S_2;
    if Score_1 > Score_2 then
        return S_1
    else
        return S_2
```

Output: Best group selection S

Table 2: Case studies on `Taskonomy-5`, we list the best task combinations identified by different methods.

Task	Methods	s	d	n	k	e	Estimated Gain	Valid Gain	Test Gain
s	HOA	✓	–	✓	–	–	+19.49%	+19.49%	+17.35%
	`MTG-Net`	✓	–	✓	–	–	+19.49%	+19.49%	+17.35%
	Oracle	✓	–	✓	–	–	–	+19.49%	+17.35%
d	HOA	–	✓	✓	–	–	+14.61%	+14.61%	+16.76%
	`MTG-Net`	–	✓	✓	–	–	+14.61%	+14.61%	+16.76%
	Oracle	–	✓	✓	–	–	–	+14.61%	+16.76%
n	HOA	–	–	✓	–	✓	+1.69%	+1.69%	+1.71%
	`MTG-Net`	–	✓	✓	–	✓	+1.80%	+1.80%	+1.82%
	Oracle	–	✓	✓	–	✓	–	+1.80%	+1.82%
k	HOA	–	–	✓	✓	–	+44.33%	+44.33%	+37.09%
	`MTG-Net`	–	–	✓	✓	–	+44.33%	+44.33%	+37.09%
	Oracle	–	–	✓	✓	–	–	+44.33%	+37.09%
e	HOA	–	–	✓	–	✓	+4.12%	+4.12%	+4.25%
	`MTG-Net`	–	✓	✓	–	✓	+11.04%	+11.04%	+12.14%
	Oracle	–	✓	✓	–	✓	–	+11.04%	+12.14%

Table 3: Case studies on ETTm1-7, we list the best task combinations identified by different methods.

Task	Methods	HF	HL	MF	ML	LF	LL	OT	Estimated Gain	Valid Gain	Test Gain
HF	HOA	✓	–	–	✓	–	–	–	+9.63%	+9.63%	+7.99%
	MTG-Net	✓	–	–	✓	✓	–	–	+19.20%	+19.20%	+16.43%
	Oracle	✓	–	–	✓	✓	–	–	-	+19.20%	+16.43%
HL	HOA	–	✓	–	✓	–	–	–	+4.84%	+4.84%	-2.41%
	MTG-Net	✓	✓	✓	✓	–	✓	✓	+7.57%	+7.57%	-1.15%
	Oracle	✓	✓	–	–	✓	–	–	-	+8.22%	-1.09%
MF	HOA	–	–	✓	–	✓	–	–	+12.75%	+12.75%	+20.68%
	MTG-Net	✓	✓	✓	–	✓	–	–	+16.08%	+16.08%	+16.53%
	Oracle	–	–	✓	✓	✓	–	✓	-	+24.60%	+23.70%
ML	HOA	–	✓	–	✓	–	–	–	+4.64%	+4.64%	+0.65%
	MTG-Net	✓	✓	✓	✓	–	✓	✓	+7.92%	+7.92%	+3.80%
	Oracle	–	–	✓	✓	✓	–	✓	-	+8.71%	+10.99%
LF	HOA	–	–	–	–	✓	–	–	0	0	0
	MTG-Net	–	–	–	✓	✓	✓	–	+6.61%	+6.61%	+5.57%
	Oracle	–	–	–	✓	✓	✓	–	-	+6.61%	+5.57%
LL	HOA	–	–	✓	–	–	✓	–	+0.99%	+0.99%	-10.36%
	MTG-Net	✓	✓	✓	✓	–	✓	-	+12.78%	+12.78%	+8.97%
	Oracle	✓	✓	✓	✓	–	✓	-	-	+12.78%	+8.97%
OT	HOA	✓	–	–	–	–	–	✓	+16.80%	+16.80%	+24.80%
	MTG-Net	✓	✓	✓	✓	–	✓	✓	+32.47%	+32.47%	+41.26%
	Oracle	✓	✓	✓	✓	–	✓	✓	-	+32.47%	+41.26%

on tasks' encoded embeddings; (3) We provide a comparison on estimation errors between active selection and random selection with different K.

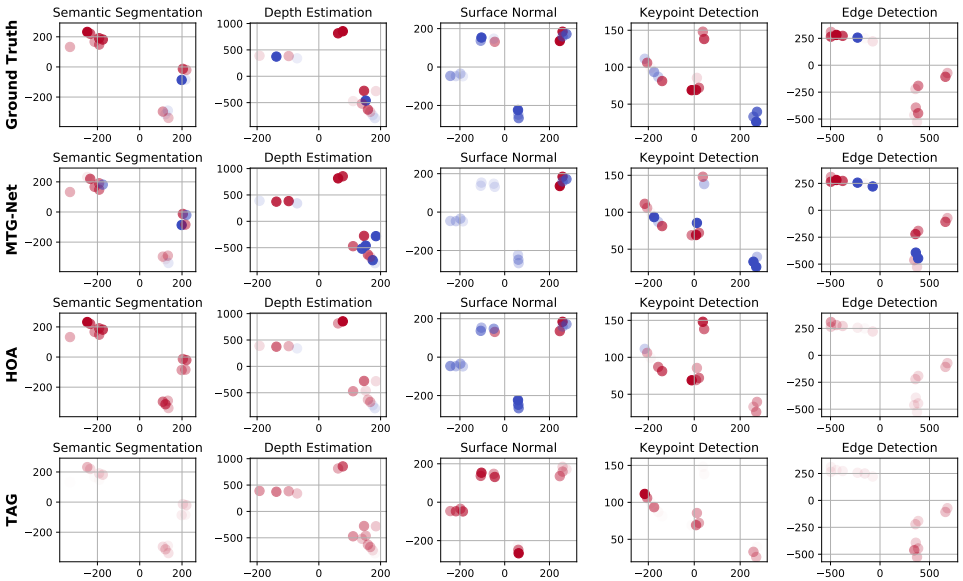
D.1 Case Study

For tasks on each dataset, we identify the task combination that provides the largest gain for it, and provide the best combination selection results for the all tasks on Taskonomy-5, ETTm1-7, and MIMIC-III-27, respectively.

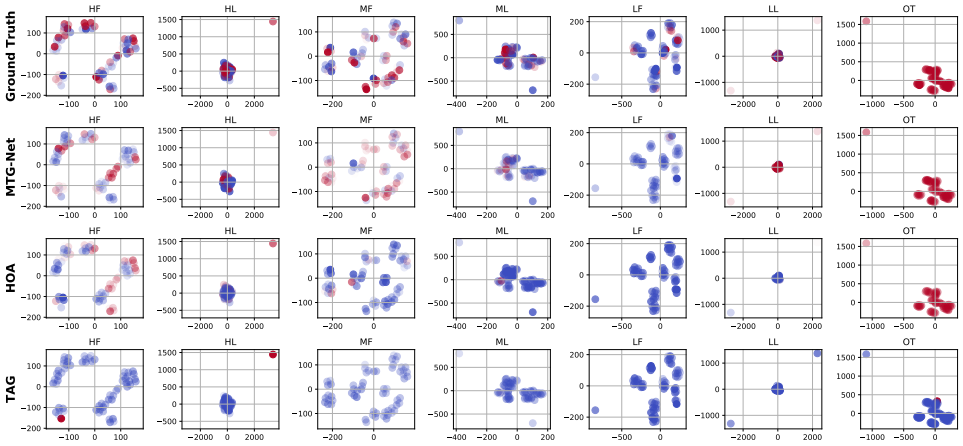
The experimental results can be seen through in Table 2, Table 3, and Table 4. For the combination selection results on Taskonmy-5, we can see the estimated gains of all tasks in this setting are the same as the ground truth. It depicts that such task combinations are sampled during training, and validates that MTG-Net has the ability to sample task combinations with large gains. Therefore, MTG-Net successfully identifies the optimal combination solutions for each task on Taskonmy-5. Furthermore, the results on MIMIC-III-27 show the generalization ability of MTG-Net, that is, it can predict and pick task combinations with large gains on unseen groups (t_3 and t_{24}). Thus, MTG-Net performs well on both settings for per-task best combination selection. Therefore, we think the final grouping performance advantages of MTG-Net are two-fold: (i) The active learning strategy encourages the model to sample task combinations with larger gains, which also makes it collect the ground truth gains of the sampled combinations. Thus, if the active learning strategy works, MTG-Net can collect the combinations with the largest gains through iterative sampling directly; (ii) Meta learning on task combinations allows MTG-Net to have the ability to generalize over the unseen task combinations. To sum up, the active learning strategy not only provides the 'fuel' for MTG-Net training, but also collects the combinations which possibly can be the optimal solutions for tasks. On the other hand, MTG-Net guides the sampling of active learning by predicting the performance of task combinations.

D.2 Further Visualization Results

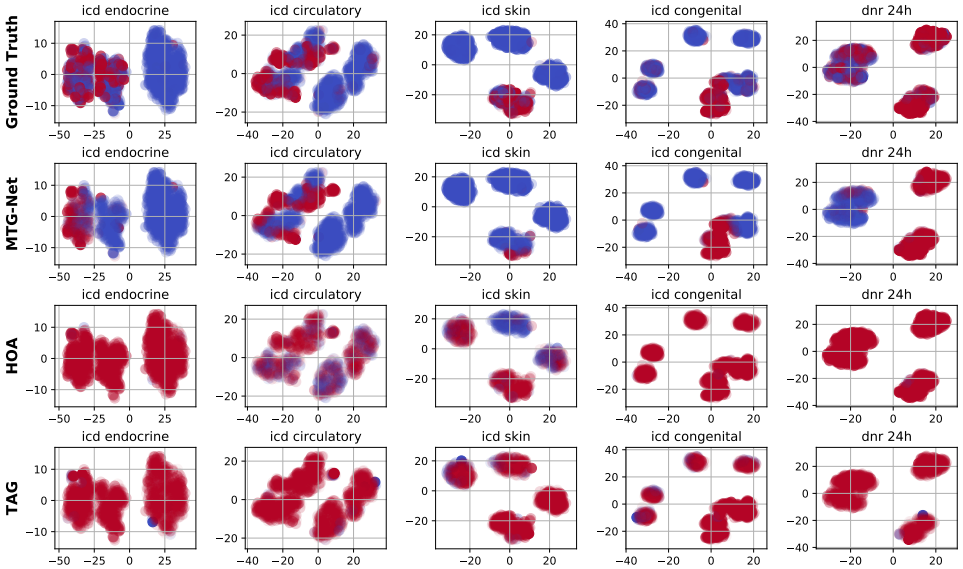
This section provides further visualization results and analyses on tasks' encoded embeddings, which are projected into a two dimensional space by t-SNE. The points' are colored by the ground truth value (red: positive; blue: negative; the deeper the color depth, the larger the gain (absolute


Ground Truth
MTG-Net
HOA
TAG
Semantic Segmentation
Depth Estimation
Surface Normal
Keypoint Detection
Edge Detection

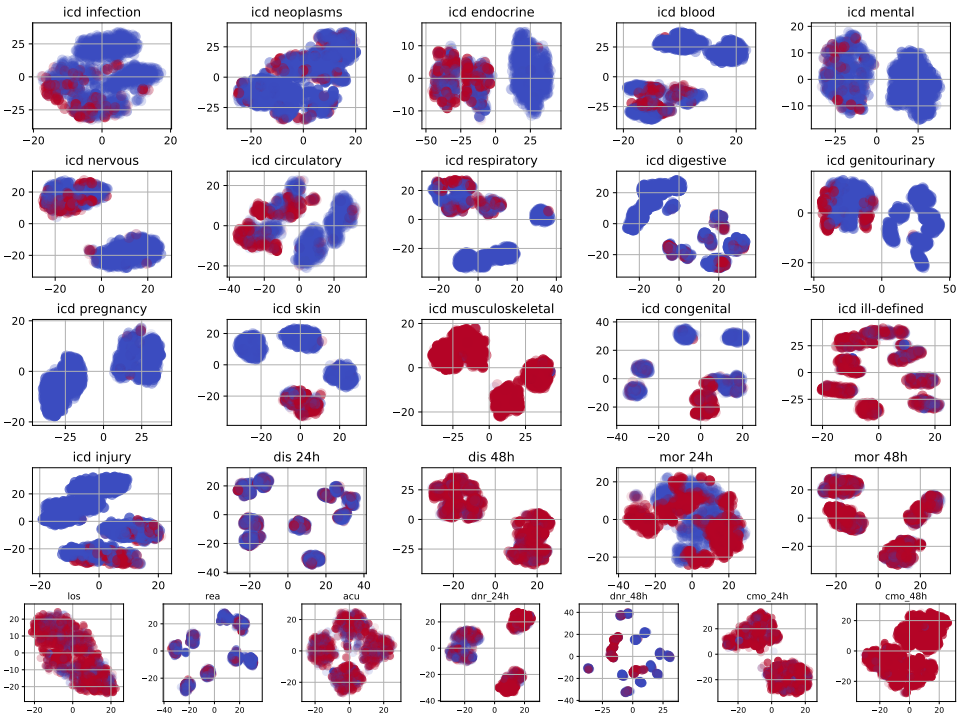
(a)


Ground Truth
MTG-Net
HOA
TAG
HF
HL
MF
ML
LF
LL
OT

(b)


Ground Truth
MTG-Net
HOA
TAG
icd endocrine
icd circulatory
icd skin
icd congenital
dnr 24h

(c)



(d)

Figure 5: We visualize the latent structures of transferring relationships discovered by `MTG-Net` in Figure 5a, Figure 5b, and Figure 5c. For each task, we aggregate its encoded embeddings for all relevant task combinations and project them into a two-dimensional space via t-SNE [44]. We color each combination by gain estimations that are produced by different approaches for this task. The combinations in the first row are colored by the ground-truth gains, while the combinations in the second to fourth row are colored by gain estimations produced by `MTG-Net`, `HOA`, and `TAG`, respectively. Figure 5a includes all 5 tasks in `Taskonomy-5`, Figure 5b includes all 7 tasks in `ETTm1-7` , and Figure 5c includes 5 selected tasks in `MIMIC-III-27`. Besides, Figure 5d shows the visualization of latent structures of transferring relationships that are colored by combinations' ground truth gains for all 27 tasks in `MIMIC-III-27`.

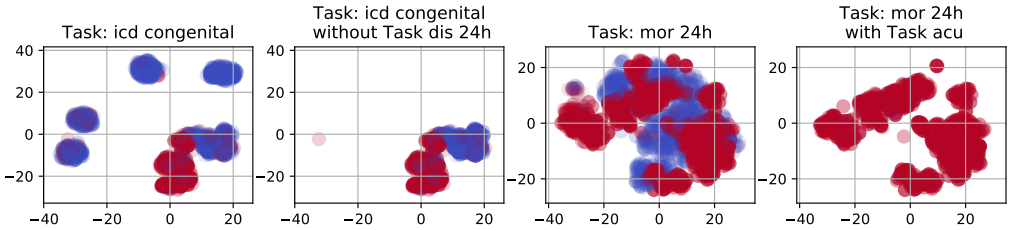


Figure 6: Two examples on pairwise task impact on the latent structures of transferring relationships discovered by `MTG-Net`. These diagrams illustrate the latent structures discovered by `MTG-Net` of two tasks: *icd congenital* and *mor 24h* in `MIMIC-III-27`. The first diagram illustrates *icd congenital*'s t-sne visualization results on all relevant task combinations, while the second diagram shows *icd congenital*'s t-sne visualization results on its relevant task combinations that exclude task *dis 24h*. The third diagram illustrates *mor 24h*'s t-sne visualization results on all relevant task combinations, while the second diagram shows *icd congenital*'s t-sne visualization results on its relevant task combinations that include task *dis 24h*.

Table 4: Case studies on MIMIC-III-27, we list the best task combinations identified by different methods.

Tasks	Methods	Selected Task Combinations	Estimated Gain	Valid Gain	Test Gain
t_3	HOA	t_3 t_{10}	+2.71%	+2.71%	-2.95%
	MTG-Net	t_1 t_2 t_3 t_4 t_7 t_8 t_{10} t_{11} t_{12} t_{13} t_{16} t_{20} t_{22} t_{24} t_{25} t_{26} t_{27}	+9.35%	+7.88%	+5.92%
	Oracle	t_1 t_2 t_3 t_7 t_8 t_{10} t_{11} t_{14} t_{15} t_{16} t_{22} t_{24} t_{26}	–	+8.35%	+6.65%
t_{11}	HOA	t_{11} t_{15}	+0.40%	+0.40%	+0.81%
	MTG-Net	t_{11}	0	0	0
	Oracle	t_{11} t_{15}	–	+0.40%	+0.81%
t_{16}	HOA	t_{16} t_{20}	+27.19%	+27.19%	+30.55%
	MTG-Net	t_1 t_2 t_3 t_4 t_7 t_8 t_{10} t_{11} t_{12} t_{13} t_{16} t_{20} t_{22} t_{24} t_{25} t_{26} t_{27}	+50.30%	+50.30%	+27.90%
	Oracle	t_5 t_6 t_9 t_{13} t_{14} t_{17} t_{18} t_{20} t_{21} t_{23} t_{25} t_{27}	–	+50.30%	+27.90%
t_{18}	HOA	t_{12} t_{18}	+2.22%	+2.22%	+0.70%
	MTG-Net	t_1 t_3 t_4 t_5 t_8 t_{10} t_{11} t_{12} t_{13} t_{15} t_{18} t_{21} t_{23} t_{24} t_{25} t_{26}	+10.10%	+5.73%	+2.58%
	Oracle	t_2 t_7 t_8 t_{10} t_{11} t_{15} t_{16} t_{18} t_{19} t_{20} t_{23} t_{26}	–	+6.12%	+8.37%
t_{19}	HOA	t_{19} t_{23}	+10.90%	+10.90%	+4.37%
	MTG-Net	t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16} t_{17} t_{18} t_{19} t_{20} t_{21} t_{22} t_{23} t_{24} t_{25} t_{26} t_{27}	+20.62%	+20.62%	+7.25%
	Oracle	t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16} t_{17} t_{18} t_{19} t_{20} t_{21} t_{22} t_{23} t_{24} t_{25} t_{26} t_{27}	–	+20.62%	+7.25%
t_{24}	HOA	t_{24} t_{27}	+9.00%	+9.00%	+8.49%
	MTG-Net	t_1 t_4 t_7 t_8 t_9 t_{12} t_{13} t_{14} t_{15} t_{16} t_{20} t_{21} t_{22} t_{23} t_{24}	+24.88%	+20.49%	+16.32%
	Oracle	t_1 t_2 t_3 t_4 t_6 t_8 t_{11} t_{12} t_{13} t_{15} t_{20} t_{21} t_{22} t_{23} t_{24} t_{27}	–	+26.10%	+21.74%

value)). In Figure 5a, Figure 5c and Figure 5b, we provide further visualization results on encoded embeddings by the predictions of different approaches. These figures show that MTG-Net learns the latent relationships between tasks more effectively compared with HOA and TAG. Besides, for a comprehensive view of the encoded representations of MIMIC-III-27's tasks, we provide the encoded task representations' t-SNE visualization results for all tasks in MIMIC-III-27 in Figure 5d.

Besides, we additionally provide experiments to show how pairwise impact on task encoded embedding's structures in Figure 6. The two diagrams on the left of Figure 6 illustrate the latent structures discovered by MTG-Net of task *icd congenital* in MIMIC-III-27. (Task *icd congenital* is a task to predict the presence of a congenital anomaly). The diagram on the left includes its t-sne visualization results on all relevant task combinations, while the diagram on the right shows its t-sne visualization results on its relevant task combinations exclude the *dis 24h*. However, as we can see, when we remove the task combinations that it's trained with *dis 24h* (*dis 24h* is used to predict whether the patients will be discharged within the next 24 hours), the *icd congenital*'s representation will no longer appear in some clusters that consist mainly of negative samples. It shows that there's a significant *negative transfer* from *dis 24h* to task *icd congenital*. Moreover, as it can be seen in the two diagrams on the right of Figure 6, when task *mor 24h* (whether the patient's recorded time of death is within the subsequent 24 hours) is trained with *acu* (if the patient will die, where will the patient stay at the end), its representation will no longer appear into a negative cluster, which suggests that there is a strong positive transfer between these two tasks. On the other hand, the task's

Dataset	Meta Model	Layers = 1	Layers = 2	Layers = 3	Layers = 4	Layers = 5
Taskonomy-5	MTG-Net (K=1)	+12.17%	+12.43%	+12.43%	+12.35%	+12.35%
Taskonomy-5	MTG-Net (K=2)	+17.90%	+18.34%	+18.34%	+17.90%	+18.34%
ETTm1-7	MTG-Net (K=1)	+10.52%	+10.61%	+10.61%	+10.57%	+10.52%
ETTm1-7	MTG-Net (K=3)	+14.44%	+14.52%	+14.50%	+14.44%	+14.50%
MIMIC-III-27	MTG-Net (K=1)	+6.25%	+6.28%	+6.29%	+6.29%	+6.31%
MIMIC-III-27	MTG-Net (K=13)	+8.63%	+8.73%	+8.74%	+8.76%	+8.78%

Table 5: Max-budget grouping performance of multiple MTG-Net models with different numbers of encoding layers.

Dataset	Meta Model	D = 8	D = 16	D = 32	D = 64	D = 128
Taskonomy-5	MTG-Net (K=1)	+12.35%	+12.43%	+12.43%	+12.43%	+12.35%
Taskonomy-5	MTG-Net (K=2)	+17.90%	+18.34%	+18.34%	+18.34%	+18.34%
ETTm1-7	MTG-Net (K=1)	+10.52%	+10.59%	+10.61%	+10.61%	+10.61%
ETTm1-7	MTG-Net (K=3)	+14.42%	+14.50%	+14.50%	+14.52%	+14.52%
MIMIC-III-27	MTG-Net (K=1)	+6.07%	+6.10%	+6.17%	+6.28%	+6.32%
MIMIC-III-27	MTG-Net (K=13)	+8.12%	+8.45%	+8.62%	+8.73%	+8.79%

Table 6: Max-budget grouping performance of multiple MTG-Net models with different numbers of hidden dimension.

encoded representation of MTG-Net successfully capture such task relationships and exhibits them in a structured latent space that can be seen from the visualization results above.

D.3 Hyper-parameter testing results for MTG-Net

This section supplements more hyper-parameter testing results for MTG-Net. We showed that MTG-Net is robust to different configurations of the number of encoding layers and the hidden dimension (D), which in this paper are set as 2 (layers) and 64 (D), respectively. Below we show the max-budget grouping performance of multiple MTG-Net models with different setups of these two hyper-parameters on MIMIC-III-27. The results can be seen in Table 5 and Table 6.

Moreover, from Table 7 we can see that MTG-Net can still obtain reasonable performance with other setups of α, and by significantly preferring task combinations with larger gains, MTG-Net is able to obtain slightly more improvements in terms of the final grouping performance. The rationale behind this observation is that the estimation errors on larger transferring gains can cause more direct error propagations to the grouping selection algorithm, thus paying more attention to these estimation errors helps most in the final grouping performance.

Besides, the hyper-parameter η controls the frequency of updating MTG-Net with actively selected meta samples, which allows a trade-off between efficiency and effectiveness for MTG-Net training. In this work, we adopt a dynamic strategy: 1) in the earliest round (K = 1), we set η as 1 to encourage more frequent MTG-Net updating per each active selection, 2) and in latter epochs (K > 1), we set η as N to further improve the efficiency by updating MTG-Net per N active selections. In the following table, we compare this dynamic strategy with two fixed strategies (η = 1 or η = N) to reveal the impact of η. From Table 8 we can see that our dynamic strategy for η can help MTG-Net with different Ks achieve competitive performance as setting η as 1 while ensuring that the number of MTG-Net updating steps is not an order of magnitude larger than that of setting η as N.

D.4 Detailed task combinations selected by MTG-Net at each step of active learning

Here we provide a few snapshots of the active learning procedure as a quick response in Table 9. From the table, we can observe diversified combinations being selected for different tasks at different

Dataset	Meta Model	α=0.01	α=0.1	α=1	α=10	α=25	α=100
Taskonomy-5	MTG-Net (K=1)	+11.35%	+11.50%	+11.95%	+12.43%	+12.43%	+12.43%
Taskonomy-5	MTG-Net (K=2)	+17. 75%	+17.90%	+18.14%	+18.34%	+18.34%	+18.34%
ETTm1-7	MTG-Net (K=1)	+10.11%	+10.15%	+10.52%	+10.61%	+10.61%	+10.61%
ETTm1-7	MTG-Net (K=3)	+14.30%	+14.37%	+14.46%	+14.52%	+14.52%	+14.52%
MIMIC-III-27	MTG-Net (K=1)	+5.86%	+6.07%	+6.15%	6.26%	+6.28%	+6.28%
MIMIC-III-27	MTG-Net (K=13)	+7.91%	+8.13%	+8.61%	+8.70%	+8.73%	+8.74%

Table 7: Max-budget grouping performance of MTG-Net with different α.

Meta Model	η	Updating Steps	Taskonomy-5	ETTm1-7	MIMIC-III-27
MTG-Net (K=1)	η = 1	$K * N \rightarrow N$	+12.47%	+10.65%	+6.29%
MTG-Net (K=1)	η = N	$K \rightarrow 1$	+11.27%	+9.97%	+5.97%
MTG-Net (K=1)	η = 1 if K =1 else N	$N + K - 1 \rightarrow N$	+12.43%	+10.61%	+6.28%
MTG-Net (K=$(N-1)/2$)	η = 1	$K * N \rightarrow N * (N-1)/2$	+18.34%	+14.55%	+8.75%
MTG-Net (K=$(N-1)/2$)	η = N	$K \rightarrow (N-1)/2$	+17.75%	+14.26%	+8.69%
MTG-Net (K=$(N-1)/2$)	η = 1 if K =1 else N	$N + K - 1 \rightarrow (N-1) * 3/2$	+18.34%	+14.52%	+8.73%

Table 8: Max-budget grouping performance with different setups of η and update steps.

rounds. More importantly, these results intuitively reveal that MTG-Net can calibrate its predictions after updating with the ground-truth gains.

D.5 How can MTG-Net benefits from the active learning strategy?

Figure 7 shows how the errors of gain estimations on unseen task combinations vary with K, which determines the scale of the meta-training set. We can observe that given the same size of $\mathcal{C}^{train}$, the active selection of meta-samples produces fewer estimation errors compared with the random selection. Combining Figure 7 with Figure 3 that illustrates random selection *v.s.* active selection in the main paper, we can observe that the performance of gain estimations highly correlates with the final grouping performance. Moreover, the active selection has a lower estimation error compared with random selection, and it performs better on the final grouping performance.

Dataset	Round K	Task j	Selected Combinations	Ground-truth Gain	Predicted Gain Before Selection	Predicted Gain After Selection
Taskonomy-5	1	s	$[s,k]$	6.3%	1.6%	3.2%
Taskonomy-5	2	s	$[s,d,n,e]$	7.2%	15.9%	9.1%
Taskonomy-5	1	n	$[s,d,n]$	-16.7%	-3.4%	-5.1%
Taskonomy-5	2	n	$[n,e]$	1.6%	-4.5%	0.2%
ETTm1-7	1	HF	[HF,MF,LL]	-5.5%	-2.3%	-6.5%
ETTm1-7	3	HF	[HF,ML,LF]	19.2%	5.4%	7.3%
ETTm1-7	1	LL	[LL,OT]	-3.5%	-5.5%	-2.8%
ETTm1-7	3	LL	[HF,HL,MF,LF,LL]	-2.0%	4.1%	-3.5%
MIMIC-III-27	1	t_1	$[t_1,t_2,t_3,t_5,t_8,t_9,t_{10},t_{16},$ $t_{17},t_{19},t_{21},t_{23},t_{24},t_{25}]$	-8.0%	-2.0%	-7.3%
MIMIC-III-27	7	t_1	$[t_1,t_3,t_6,t_7,t_9,t_{10},t_{11},$ $t_{14},t_{16},t_{26},t_{27}]$	4.4%	6.8%	5.7%
MIMIC-III-27	13	t_1	$[[t_1,t_{12},t_{13},t_{14},t_{15},t_{17},$ $t_{18},t_{19},t_{24},t_{26}]$	-7.7%	-9.3%	-8.4%
MIMIC-III-27	1	t_24	$[t_4,t_8,t_9,t_{12},t_{13},t_{14},t_{17},$ $t_{20},t_{24}]$	-3.2%	-36.1%	-5.1%
MIMIC-III-27	7	t_{24}	$[t_8,t_{10},t_{11},t_{13},t_{15},t_{19},t_{21},$ $t_{22},t_{23},t_{24},t_{25}]$	20.1%	20.7%	20.5%
MIMIC-III-27	13	t_24	$[\ t_1,t_2,t_3,t_4,t_7,t_8,t_9,t_11,$ $t_12,t_16,t_20,t_22,t_23,t_24\ ]$	21.1%	23.5%	22.3%

Table 9: Snapshots of the active learning procedure.

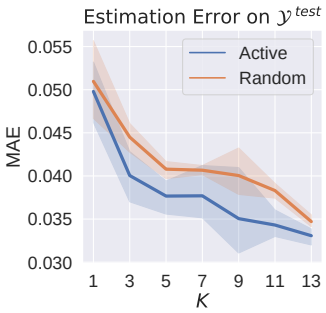


Figure 7: The estimation errors on $\mathcal{C}^{test}$ as K increases.