
Automated Multi-Task Learning for Joint Disease Prediction on Electronic Health Records

Suhan Cui

The Pennsylvania State University
University Park, PA, USA
suhan@psu.edu

Prasenjit Mitra

The Pennsylvania State University
University Park, PA, USA
pmitra@psu.edu

Abstract

Electronic Health Records (EHR) have become a rich source of information with the potential to improve patient care and medical research. In recent years, machine learning models have proliferated for analyzing EHR data to predict patients' future health conditions. Among them, some studies advocate for multi-task learning (MTL) to jointly predict multiple target diseases for improving the prediction performance over single task learning. Nevertheless, current MTL frameworks for EHR data have significant limitations due to their heavy reliance on human experts to identify task groups for joint training and design model architectures. To reduce human intervention and improve the framework design, we propose an automated approach named **AutoDP**, which can search for the optimal configuration of task grouping and architectures simultaneously. To tackle the vast joint search space encompassing task combinations and architectures, we employ surrogate model-based optimization, enabling us to efficiently discover the optimal solution. Experimental results on real-world EHR data demonstrate the efficacy of the proposed **AutoDP** framework. It achieves significant performance improvements over both hand-crafted and automated state-of-the-art methods, also maintains a feasible search cost at the same time. Source code can be found via the link: <https://github.com/SH-Source/AutoDP>.

1 Introduction

In the era of big data and digital healthcare, the voluminous Electronic Health Records (EHR) can revolutionize patient care, medical research, and clinical decision-making. Using these, the machine learning (ML) community has been designing models to predict patients' future health conditions, e.g., models for mortality prediction [1], diagnosis prediction [2, 3] and hospital readmission [4]. Although most existing machine learning based prediction models are designed to be single-task, i.e. predicting the risk of a single target disease, some works [5, 6, 7, 8, 9] designed multi-task learning (MTL) models to jointly predict multiple targets. The motivation lies in the fact that two or more diseases might be related to each other in terms of sharing common comorbidities, symptoms, risk factors, etc. Consequently, training on related diseases simultaneously can offer additional insights and potentially enhance prediction performance. While multi-task learning offers potential advantages, the existing MTL frameworks for EHR data still suffer from the following limitations.

Limitations of the existing MTL frameworks for EHR data. To design an effective MTL framework, two fundamental challenges need to be addressed:

(1) *How can we determine which tasks should be trained together?* The task grouping problem [10] involves finding groups of tasks to train jointly. Multi-task learning only provides advantages when the tasks are synergistic, i.e., training on the tasks together makes the model learn general knowledge that helps in performing the tasks better in the test set and prevents overfitting. Thus, given a large set

of related tasks in a domain, we may need to group the tasks (allowing tasks to belong to multiple groups) together to create groups of tasks on each of which we will train a model. However, existing works usually rely on human expert discretion to select multiple tasks upfront and create a shared model for those tasks [5, 6, 7, 8, 9]. Hence, none of them has addressed the general problem of task grouping for EHR data. Moreover, due to the complexity of disease correlations, grouping synergistic tasks together is extremely challenging for human experts. It not only demands substantial effort (trying out every possible task combination) but also introduces the risk of task interference (putting disparate diseases together), potentially leading to performance degradation. Therefore, how to design the appropriate task grouping for MTL on EHR data presents a critical challenge.

(2) *How can we design model architectures for MTL?* Existing works [5, 6, 7, 8, 9] typically rely on hand-crafted architectures for multi-task learning, which consist of a shared EHR encoder followed by several task-specific classifiers. However, due to the large number of possible operations as well as network topologies, manually tuning an optimal architecture for MTL is impossible. Furthermore, the optimal architectures for different task groups might also be distinct. Thus, things can even get worse when the number of tasks grows and different task combinations are involved for joint tuning. Therefore, we need a more efficient and effective approach to design the optimal MTL architectures for EHR data.

Automating the MTL framework design for EHR data. To address the aforementioned challenges, we look to Automated Machine Learning (AutoML) [11]. Since AutoML relies on data-driven approaches to automate the design of machine learning algorithms, it has the potential to improve the design of an MTL framework for EHR data and reduce human interventions. Several attempts have been explored in other domains, e.g., computer vision, to improve the design of task grouping [12, 10, 13] and MTL architectures [14, 15, 16, 17, 18]. However, the exploration of AutoML in healthcare domain remains relatively limited [19]. To the best of our knowledge, there are no existing work that automates the finding of groups of tasks for MTL towards designing an optimal framework for classification tasks using EHR data, which is a notable gap in the field.

Joint optimization over task grouping and architecture search. Moreover, currently there exists no end-to-end optimization framework for automating MTL, even in other domains. Current approaches independently address the problems of task grouping and architecture design. First, a line of work [12, 10, 13] solves the task grouping problems by learning the task correlations. They operate under the underlying assumption that MTL architectures are the same across different task groups, which might not be practical nor optimal. Second, researchers also apply Neural Architecture Search (NAS) [20] to automatically design MTL architectures for a predefined set of tasks [14, 15, 16, 17, 18, 21]. No existing work has integrated these two approaches to address both problems simultaneously. However, combining them naively could lead to sub-optimal results, as sequential optimization might result in inaccurate estimations for both aspects. Therefore, we need a more generalized AutoML framework for the joint optimization of both task grouping and architecture search.

Overview of the proposed approach. Therefore, in this paper, we show that an integrated approach for multi-task grouping and neural architecture search provides significant improvements. First, we extend existing single-task models like Retain [22], Adacare [16] to MTL in an EHR setting. Second, we apply DARTS [23], an NAS method used in MTL settings in different domains to the EHR domain. We use one shared model for predicting multiple tasks. These adaptations improve over the single-task setting. Second, we explore the impact of automated task grouping in the EHR setting by grouping tasks and finding an optimal NAS model for each task group. This further improves the performance. Finally, we propose an integrated framework an **Automated** multi-task learning framework, **AutoDP**, for joint **Disease Prediction** on electronic health records, which aims at jointly searching for the optimal task grouping and the corresponding neural architectures that maximize the multi-task performance gain. We show that this third method provides the maximal performance gain.

Specifically, in **AutoDP**, we employ a surrogate model-based optimization approach [24] for efficient search. First, we define the joint search space of task combinations and architectures that includes all possible configurations for MTL. We want to find optimal solutions from this search space. To achieve that, the first question is how we can evaluate the performance of each configuration. Performing the ground truth evaluation for every configuration is infeasible, since it requires an entire multi-task learning procedure for each pair of architecture and task combination. Therefore, instead of exhaustively evaluating all the configurations, we build a surrogate model to predict the multi-task

gains for any given configurations from the search space. In this way, we only need to evaluate the ground truth gains for a subset of samples from the search space, and use them to train the surrogate model for estimating the rest ones. The intuition is that there exists an underlying mapping from each configuration to the expected multi-task gains; thus it can be learned by a neural network. The remaining question is how we can effectively train the surrogate model using as few samples as possible. To this end, we further propose a progressive sampling strategy to guide the surrogate model training for improving sample efficiency. That is we train the surrogate model through multiple iterations. At every iteration, we select some points from the search space and update the surrogate model accordingly. The selection is conditioned on the current surrogate model and involves both exploitation and exploration. That is, we iteratively select the points that bring higher performance gains and also come from unexplored areas, which makes the training samples represent the whole search space. Eventually, after we obtain the trained surrogate model, we further use it to derive the final optimal task grouping and architectures. Because of the huge search space, it is not practical to use brute-force search. Hence, we develop a greedy search method to find the near-optimal solution.

In summary, our contributions are as follows:

- We are the first to propose an automated approach for multi-task learning on electronic health records **AutoDP**, which largely improves the design of task grouping and model architectures by reducing human interventions. Specifically, this work is the first to automate the design for the optimal task grouping and model architectures for MTL on EHR data.
- We are the first to propose a surrogate model based optimization framework that jointly searches for the optimal task grouping and corresponding model architectures with high efficiency in any domain.
- We propose a progressive sampling strategy to construct the training set for the surrogate model, which improves sample efficiency by reducing the required number of ground truth evaluations during searching. Importantly, we balance exploitation and exploration so that the sampled configurations can represent the whole search space and are highly accurate.
- We propose a greedy search algorithm to derive the final MTL configuration using the trained surrogate model and find a near-optimal solution from the huge search space efficiently.
- Experimental results on real world EHR data - MIMIC IV [25] demonstrate that **AutoDP** improves classification performance significantly over existing hand-crafted and automated methods under feasible computational costs.

2 Related Work

Multi-Task Learning with EHR. To enhance prediction performance while forecasting patients' health conditions based on their historical data [26], existing studies employ multi-task learning to simultaneously predict multiple related target diseases or conditions, resulting in improved performance compared to single-task training. For example, Wang, et al. [7] investigated the advantages of joint disease prediction using traditional machine learning models. More recently, researchers have applied recurrent neural network (RNN) based models to conduct multi-task learning on EHR data [27, 6, 5], which is able to predict tasks like mortality, length of stay, ICD-9¹ diagnoses and etc. Additionally, Zhao, et al. [8] also utilized a transformer based method for multi-task clinical risk prediction on multi-modal EHR data. However, all these studies manually select the set of tasks for joint training without task grouping and utilize a hand-crafted MTL model architecture, which largely limits their performance.

Multi-Task Grouping. Due to the limitation of manually selected task groups, some of the work focus on obtaining the optimal task grouping through searching. Specifically, Standley, et al. [10] is the first work that systematically analyze the task correlations. For improving the efficiency, they use pair-wise MTL gains to estimate the high-order MTL gains, and obtain the pair-wise gains by training one model for each task pair. Based on the estimated gains, they derive the optimal task grouping using brute-force search. Fifty, et al. [12] further improves the efficiency by training one model to derive all the pair-wise gains. They derive the task affinity based on the gradient information during training. Furthermore, Song, et al. [13] propose a more general method that employs a meta model to

¹<https://www.cdc.gov/nchs/icd/icd9.htm>

learn the task correlations and estimates the high-order MTL gains more effectively. These works normally assume that the model architecture is the same across different task groups. But in practice, we can maximize performance gains by applying different model architectures with respect to each task group. Thus, we need a more general framework that considers the model architectures during task grouping.

Multi-Task NAS. Neural Architecture Search (NAS) [20] stands as a prominent research area in AutoML, focusing on the exploration of optimal deep network architectures through a data-driven approach. Although the main stream of NAS focuses on the setting of single task learning, some researchers also try to employ NAS in multi-task learning applications, predominantly for searching computer vision MTL architectures. Notably, studies done by Ahn, et al. [14] and Bragman, et al. [15] employ reinforcement learning and variational inference, respectively, to determine whether each filter in convolutions should be shared across tasks. Furthermore, other recent works [16, 17, 18, 21] leverage differentiable search algorithms [23], to determine the optimal sharing patterns across multiple network layers for diverse tasks. Despite the demonstrated advancements, a common limitation is their reliance on human experts to pre-define a set of tasks for joint training. This constraint poses challenges in practical scenarios where task grouping is not readily available, thereby limiting their broader applicability. What is more, their frameworks often search for better MTL architectures on top of one or several backbone architectures such as ResNet [28]. However, such backbone architectures might not be available for EHR applications in medical domain. Therefore, a new multi-task NAS framework is needed for EHR data.

3 Methodology

3.1 Preliminaries

Problem definition. Assume we have the input EHR data for multiple patients where each patient is represented as $\mathbf{X} \in \mathbb{R}^{L \times d_e}$, where L is the time sequence length and d_e is the hidden dimension of the input features. We have N prediction tasks using the EHR data, denoted as $\mathcal{T} = \{T_1, T_2, \dots, T_N\}$. We seek to maximize the overall MTL performance gain for all these prediction tasks compared to single task training. First, we define MTL gain. Conduct a single task training on each task independently using a specific backbone model (such as RNN), and obtain the single-task performance for all tasks in terms of a predefined metric (such as average precision), denoted as $\{s_1, s_2, \dots, s_N\}$. Then, the MTL gain is defined as:

$$g_i = \frac{(m_i - s_i)}{s_i}, i = 1, \dots, N, \quad (1)$$

where m_i is the multi-task performance for T_i . Therefore, our objective is to maximize the overall gain for all tasks: $G = \frac{1}{N} \sum_{i=1}^N g_i$.

To achieve that, our proposed method solves two searching problems at the same time using AutoML. First, we search for a list of task combinations that defines which tasks should be trained together. Second, we determine the optimal model architecture for each task combination. We aim at finding the optimal configuration for both, such that the highest overall gain G is attained.

Task grouping search space. For N tasks, there are $2^N - 1$ task combinations, $\mathcal{C} = \{C_1, C_2, \dots, C_{2^N-1}\}$, where every C is a subset of $\mathcal{T}$. Given a budget B , we aim at searching for maximally B task combinations from $\mathcal{C}$ to determine which tasks should be trained together. The task combinations should cover all N tasks so that we are able to obtain $\{m_1, m_2, \dots, m_N\}$. If one task T_n appears in multiple task combinations, we simply choose the highest performance for it as m_n .

Architecture search space. For every task combination, we also need to search for an MTL architecture to model the EHR data. We adopt the hard sharing mechanism as in most existing works [27, 5], which consists of a shared encoder for extracting the latent representation of the input EHR and multiple task specific classifiers to generate the output for every task.

Specifically, we enable the search for the optimal shared encoder. For the search space of the encoder, we adopt the setting of directed acyclic graph (DAG) [23]. The architecture is represented as a DAG that consists of P ordered computation nodes, and each node is a latent feature that has connections to all previous nodes. For each connection (also called edge), we can choose one operation from a

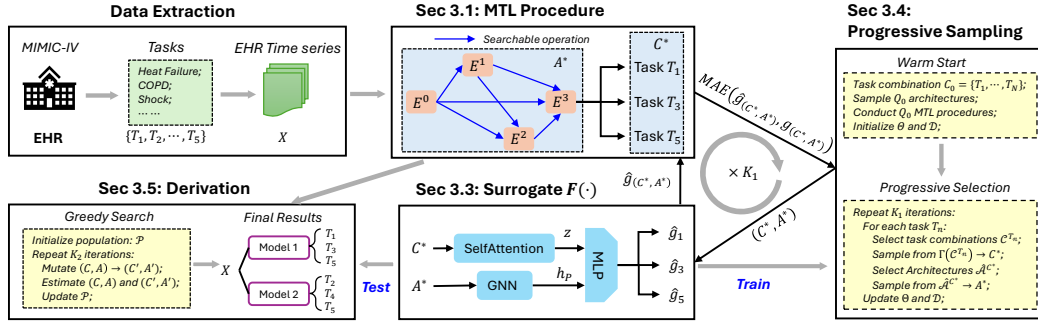


Figure 1: Overview of the proposed **AutoDP**

predefined set of candidate operations $\mathcal{O}$ for feature transformation. Let $\mathbf{E}^0 = \mathbf{X}$, the formulation of node p is defined as follows:

$$\mathbf{E}^p = \sum_{i=0}^{p-1} o_{(i,p)}(\mathbf{E}^i), o_{(i,p)} \in \mathcal{O}, \quad (2)$$

where node features $\mathbf{E}^i \in \mathbb{R}^{L \times d_e}$'s all have the same dimension as $\mathbf{X}$, and $o_{(i,p)}$ is the operation that transform $\mathbf{E}^i$ to $\mathbf{E}^p$. Essentially, sampling one architecture from the search space is equivalent to sampling one operation for every edge in the DAG. In this way, we can get the set of all possible architectures denoted as $\mathcal{A}$.

Finally, to predict, we take the last node representation $\mathbf{E}^P$ as the encoded feature for the input EHR, and use task-specific classifiers to output final predictions, which are all fixed fully connected network layers.

MTL procedure. To evaluate a specific sample from the joint search space $\mathcal{C} \times \mathcal{A}$, we need to conduct an MTL experiment to obtain the multi-task performances. Specifically, given an architecture $A \in \mathcal{A}$ and a task combination $C \in \mathcal{C}$, we train the model A to predict for all tasks in C and get the multi-task performances for those tasks. Then, we can compute their gains by Eq.(1). In this way, we are able to evaluate how much gains that this sample (C, A) could achieve.

3.2 Overview

We propose a surrogate model based AutoML framework to search for the optimal task grouping and corresponding architectures simultaneously. To achieve that, we need to first evaluate the MTL gains for all the samples in the joint search space $\mathcal{C} \times \mathcal{A}$, and then select the best B samples (pairs of task combinations and architectures) that maximize G . However, it is not practical to obtain the ground-truth gains for every sample, since the whole search space is normally very huge and every MTL procedure is also considerably expensive. Therefore, we build a neural network (called surrogate model) to learn the mapping from a pair of task combination and architecture to the multi-task gains:

$$\mathbf{g}_{(C,A)} = F(C, A), C \in \mathcal{C}, A \in \mathcal{A}, \quad (3)$$

where $\mathbf{g}_{(C,A)} \in \mathbb{R}^{|C|}$ is the per-task gains for task combination $C \in \mathcal{C}$ if using A as the model, and $F(\cdot)$ is the surrogate model. In this way, we only need to evaluate the ground truth gains for a small subset of samples from the search space, and use them to train the surrogate model for estimating all other unseen samples. The assumption is that the multi-task gains are essentially determined by the configuration of the task combination and the architecture, so there exists an underlying mapping that could be learned by a neural network. We set universal hyperparameters and optimization settings for all MTL procedures, hence the influence of other factors can be ignored.

Specifically, we introduce the model architecture of the surrogate model in Section 3.3. Then, we outline the training procedure of the surrogate model in Section 3.4, where we propose an active learning strategy to collect training samples. Eventually, we use greedy search to derive the final configuration of task grouping and architectures by utilizing the trained surrogate model, as discussed in Section 3.5. The framework overview is shown in Figure 1.

3.3 Surrogate Model

For learning the mapping from an input configuration to the multi-task gains, the surrogate model is required to encode both architectures and task combinations. Also, the model needs to output multi-task gains. Therefore, we design a new surrogate model that consists of two encoders that respectively transform the input architecture and task combination into latent representations. Then, two representations are fused together to predict the multi-task gains.

Architecture Encoding. For encoding a given architecture A , we apply a graph encoder [29] that is specifically designed for modeling DAGs, which is suitable for encoding the architectures in our search space. It can sequentially update the hidden states for the P computation nodes in preceding order by aggregating information from all predecessors. For node p , we have:

$$\mathbf{h}_p = \text{Aggregate}(\mathbf{W}_0 \cdot \mathbf{h}_0, \mathbf{W}_1 \cdot \mathbf{h}_1, \dots, \mathbf{W}_{p-1} \cdot \mathbf{h}_{p-1}), \quad (4)$$

where $\mathbf{h}_0 \in \mathbb{R}^{d_s}$ is the input node representation which contains trainable parameters, and $\mathbf{W} \in \mathbb{R}^{d_s \times d_s}$'s are learnable transition matrices constructed for each operation in $\mathcal{O}$. For every operation in the architecture, we also apply the corresponding $\mathbf{W}$ in our graph encoder. For aggregating all incoming representations, we apply average pooling to obtain the node representation $\mathbf{h}_p$. Finally, we use the node representation for the last node $\mathbf{h}_P$ as the overall encoding for the input architecture.

Task Combination Encoding. For encoding a given task combination C , we use the self attention mechanism [30] to model the high order interactions among the selected tasks in C . Specifically, we randomly initialize the embedding for all N tasks, and for task combination C , we have:

$$\mathbf{z} = \text{Pool}(\text{SelfAttention}(\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{|C|})), \quad (5)$$

where $\mathbf{u} \in \mathbb{R}^{d_s}$'s are corresponding embeddings for the selected tasks and $\mathbf{z} \in \mathbb{R}^{d_s}$ is the final representation for task combination C . Additionally, we also use average pooling on top of the self attention layers to obtain $\mathbf{z}$.

Prediction. Eventually, we apply a two layer MLP to fuse both architecture encoding $\mathbf{h}_P$ and task combination encoding $\mathbf{z}$, and output the predicted gains for all selected tasks $\hat{\mathbf{g}}_{(C,A)} \in \mathbb{R}^{|C|}$. We use the mean absolute error to supervise the surrogate model as follows:

$$\mathcal{L}(\hat{\mathbf{g}}_{(C,A)}, \mathbf{g}_{(C,A)}) = \|\hat{\mathbf{g}}_{(C,A)} - \mathbf{g}_{(C,A)}\|_1, \quad (6)$$

where $\mathbf{g}_{(C,A)} \in \mathbb{R}^{|C|}$ is the ground truth gains generated by conducting an MTL procedure for (C, A) .

3.4 Progressive Sampling

In order to efficiently train the surrogate model defined in previous section, we develop a progressive sampling method to collect training samples. Start with an empty training set and a random initialized surrogate model, we progressively sample more points from the search space $\mathcal{C} \times \mathcal{A}$, and then use them to train the surrogate model. Specifically, we include two stages for the surrogate model training:

Warm start. Firstly, we warmup the surrogate model by selecting a small number of samples from the search space. Specifically, we use the task combination that contains all N tasks $C_0 = \{T_1, \dots, T_n\}$ and randomly sample Q_0 architectures from $\mathcal{A}$. Then we conduct Q_0 MTL procedures to evaluate their gains by training on C_0 . In this way, we collect Q_0 training samples as the initial training set denoted as $\mathcal{D}$. We further train the surrogate model on $\mathcal{D}$, and denote the model parameters as Θ .

Progressive selection. Then, we progressively select more points and train the surrogate model as introduced in Algorithm 1. Totally, we conduct K_1 rounds of sampling. For each round, we iterate through all N tasks. With respect to one task T_n , we build the acquisition function Γ over the set of task combinations that contains T_n based on the predicted gains for T_n . Then, we select one task combination C^* that have highest value. We apply Upper Confidence Bound [31] as the acquisition function that considers both exploration and exploitation by explicitly estimating the mean and variance of predicted gains (line 11 marked by blue). Besides that, we would also like to see the effect of exploration vs exploitation. so we try out different settings of Γ . Specifically, we propose three variants of **AutoDP**, namely **AutoDP** $^{\mu+\sigma}$, **AutoDP** $^{\mu}$ and **AutoDP** $^{\sigma}$, which corresponds to the original setting, including only μ or only σ in Γ . In this way, we can compare the results with pure exploration and pure exploitation during sampling, and find out the optimal strategy for **AutoDP**.

Algorithm 1: Progressive Selection

Input: Training set $\mathcal{D}$, surrogate model parameter Θ , Q_1 , Q_2 , K_1 ; $Q_1 > Q_2$.**Output:** Updated $\mathcal{D}$ and Θ

```
1 for  $k = 1, 2, \dots, K_1$  do
2   for  $n = 1, 2, \dots, N$  do
3     Collect all task combinations that contains  $T_n$ :  $\mathcal{C}^{T_n} = \{C_j | \forall C_j \in \mathcal{C}, T_n \in C_j\}$ ;
4     for  $\forall C_j \in \mathcal{C}^{T_n}$  do
5       Randomly sample  $Q_1$  architectures from  $\mathcal{A}$ , denote the set as  $\mathcal{A}^{C_j}$ ;
6       Forward the surrogate model to collect gains for  $T_n$  with every architecture in  $\mathcal{A}^{C_j}$ :
7        $\mathcal{G} = \{g[T_n] | \forall A \in \mathcal{A}^{C_j}, g = F(C_j, A)\}$ ;
8       Select the top  $Q_2$  architectures from  $\mathcal{A}^{C_j}$  with highest gains in  $\mathcal{G}$ , denoted as  $\hat{\mathcal{A}}^{C_j}$ ;
9       Calculate the mean over top  $Q_2$  gains from  $\mathcal{G}$ , denoted as  $\mu^{C_j}$ ;
10      Calculate the variance over top  $Q_2$  gains from  $\mathcal{G}$ , denoted as  $\sigma^{C_j}$ ;
11    end
12    Compute the acquisition values over  $\mathcal{C}^{T_n}$  as:  $\Gamma(\mathcal{C}^{T_n}) = \{\mu^{C_j} + \lambda \cdot \sigma^{C_j}, \forall C_j \in \mathcal{C}^{T_n}\}$ ;
13    Sample a task combination  $C^*$  from  $\mathcal{C}^{T_n}$  that has highest value in  $\Gamma(\mathcal{C}^{T_n})$ , and randomly
14    sample an architecture  $A^*$  from  $\hat{\mathcal{A}}^{C^*}$ ;
15    Conduct an MTL procedure on  $(C^*, A^*)$ , and collect the ground truth labels  $g_{(C^*, A^*)}$ ;
16    Add  $(C^*, A^*, g_{(C^*, A^*)})$  to  $\mathcal{D}$ ;
17  end
18 end
19 Update  $\Theta$  by training the surrogate model on  $\mathcal{D}$ ;
```

Moreover, we also select one architecture A^* with high predicted gain for T_n when combined with C^* . The selection of C^* and A^* is interdependent, and the details are introduced in Algorithm 1. In this way, we collect one sample (C^*, A^*) to update the training set $\mathcal{D}$ with respect to each T_n . At the end of each round, we also update the surrogate model parameters Θ with the updated $\mathcal{D}$. After K_1 rounds, we are able to obtain a well trained surrogate model for estimating the whole search space.

3.5 Derivation

We derive the final results using the trained surrogate model. Due to the huge search space, it is still not practical to use brute force search to get the global optimum. Therefore, we propose to apply a greedy method to search for near-optimal solutions. We introduce the detailed procedure in Algorithm 2. The high level idea is that we first randomly initialize the configuration, and then gradually improve its multi-task gain by random mutation and greedy selection.

Specifically, given the budget B , we aim at searching for B samples from the search space $\mathcal{C} \times \mathcal{A}$ such that the overall gain G is maximized. We first randomly initialize the population $\mathcal{P}$ that contains B pairs of task combinations and architectures. Then, at every iteration, we randomly mutate one pair (C, A) from the population and see whether the overall multi-task gain will increase. If so, we

Algorithm 2: Greedy Search

Input: Trained surrogate model $F(\cdot)$, B , K_2 .**Output:** Searched population $\mathcal{P}$.

```
1 Randomly sample  $B$  pairs from  $\mathcal{C} \times \mathcal{A}$  to initialize  $\mathcal{P}$ ;
2 for  $v = 1, 2, \dots, K_2$  do
3   Randomly select one pair  $(C, A)$  from  $\mathcal{P}$ ;
4   Mutate  $(C, A)$  to  $(C', A')$  by changing one task in  $C$  or one operation in  $A$ , and obtain a new
   population  $\mathcal{P}'$ ;
5   Estimate  $\mathcal{P}'$  and  $\mathcal{P}$  using  $F(\cdot)$ ;
6   Choose the better one:  $\mathcal{P} \leftarrow \text{Select}(\mathcal{P}', \mathcal{P})$ ;
7 end
```

update $\mathcal{P}$ accordingly. After K_2 iterations, we can obtain a near-optimal solution. In practice, we also apply multiple initial populations to avoid getting stuck on local optima. Although we only get an approximate solution, our method can already achieve significant improvements over baselines as shown in Section 4.2.

4 Experiments

4.1 Set Up

Dataset & Tasks. We adopt MIMIC - IV dataset [25] for our experiments, which is a publicly available database sourced from the electronic health record of the Beth Israel Deaconess Medical Center. Specifically, we extract the clinical time series data for the 56,908 ICU stays from the database as our input EHR data, with an average sequence length of 72.9. With respect to each ICU stay, we also extract **25 prediction tasks** (listed in Table 3), including chronic, mixed, and acute care conditions. Each condition is associated with a binary label indicating whether the patient has the corresponding condition during the ICU stay.

Baselines. To compare the proposed method with existing work, we choose several state-of-art-baselines, including both *hand-crafted* and *automated* methods. Specifically, as described below, we include several human-designed EHR encoders to compare with the searched architecture we defined in Eq. (2). Also, we include one NAS method and one multi-task grouping method as the automated baselines. More importantly, we combine the multi-task grouping method with the NAS method and hand-crafted encoders to show the superiority of our joint optimization method.

- EHR encoders: We choose four models that are widely utilized for analyzing EHR time series, including LSTM [32], Transformer [30], Retain [22] and Adacare [1].
- NAS: We choose DARTS [23] as the NAS baseline, which is a differentiable search method for efficient architecture search. We apply it to our search space $\mathcal{A}$ to find better EHR encoders. Several state-of-the-art works in other domains have also used it to find MTL architectures [16, 17, 18, 21].
- Multi-task grouping: MTG-Net [13] is the current state-of-the-art multi-task grouping algorithm, which uses a meta learning approach to learn the high-order relationships among different tasks. We refer to this method as MTG in latter sections.

Evaluation Metric. We use two widely used metrics for binary classification to evaluate our method and baselines: **ROC** (Area Under the Receiver Operating Characteristic curve) and **AVP** (Averaged Precision). During surrogate model training, we use **AVP** as the metric to compute multi-task gains as in Eq. (1), since it is a more suitable choice for considering the class imbalance.

Please also refer to Appendix A for the implementation details.

4.2 Performance Evaluation

We show our results in Table 1. Each experiment is run five times and the average of the runs are reported. We run three settings: **Task @ 5**, **Task @ 10** and **Task @ 25**, which refers to using the first 5 tasks, 10 tasks and 25 tasks respectively. Since grouping all 25 tasks takes a long time to run, we include two small settings that only have the first 5 or 10 tasks in Table 3 for grouping. Our results demonstrate our hypotheses: (a) applying Retain, Adacare, and DARTS improves over the single-task setting, (b) applying different NAS models for each group further improves the performance, and finally, (c) **AutoDP** provides the best results in terms of averaged per-task gain for **ROC** and **AVP**, a significant improvement over existing MTL frameworks for EHR data.

First, without considering task grouping, we train one shared model to predict for all tasks in three settings and compute the multi-task gains for them. Results show that this setting only provides minimal improvement over single task training. Note that the automated method (DARTS) performs better than other hand-crafted methods. We also see that sequential optimization over task grouping and architecture search (MTG+DARTS) performs better than MTG + other hand-crafted encoders.

Moreover, we see that the three variants of **AutoDP** performed better than the other methods. Among them, **AutoDP** ^{$\mu+\sigma$} performs the best, which means the balance of exploration and exploitation is the most effective strategy for training the surrogate model. For the last method, we also report the standard deviations and p-values of statistical tests (compared to MTG+DARTS), which justifies that

Table 1: Performance comparison in terms of averaged per-task gain over single task backbone (All results are in the form of percentage values %).

Settings	Included Tasks	Tasks @ 5		Tasks @ 10		Tasks @ 25	
	Metric	ROC	AVP	ROC	AVP	ROC	AVP
One model for all tasks	LSTM	+0.09	+0.18	+1.06	+3.22	+1.83	+7.46
	Transformer	+0.97	+4.82	+1.41	+4.14	+1.75	+7.45
	Retain	+0.46	+1.80	+0.66	+0.75	+1.41	+5.88
	Adacare	+1.03	+5.21	+1.32	+4.05	+1.68	+6.94
	DARTS	+1.28	+5.01	+2.01	+6.87	+1.87	+7.71
Task Grouping + One model for each group	MTG+LSTM	+0.51	+2.10	+0.65	+1.87	+1.74	+7.40
	MTG+Transformer	+0.91	+3.64	+1.20	+3.95	+1.79	+9.15
	MTG+Retain	+0.55	+3.11	+1.51	+5.20	+1.54	+8.87
	MTG+Adacare	+1.25	+5.78	+1.44	+4.63	+1.75	+7.84
	MTG+DARTS	+1.47	+6.41	+2.02	+6.65	+2.41	+11.76
Variants of AutoDP	AutoDP ^{μ}	+1.49	+7.12	+2.08	+7.53	+2.68	+12.70
	AutoDP ^{σ}	+1.95	+7.68	+2.49	+8.45	+2.62	+13.37
	AutoDP ^{$\mu+\sigma$}	+1.69	+7.74	+2.55	+8.81	+2.80	+13.43
	(std)	± 0.08	± 0.25	± 0.13	± 0.29	± 0.12	± 0.33
	(p-value)	0.045	0.029	0.036	0.045	0.027	0.032

the improvement is significant. *The runtime is approximately as the same for MTG+DARTS and **AutoDP** and thus this is a fair comparison.*

Beside the overall performance gain, we also look at the distribution of performance gains for each individual task as shown in Figure 2. We can observe that the proposed method does not have the issue of negative transfer, since all tasks have a positive gain. Also, for some of the tasks, it can achieve over 20% improvement, which further shows the effectiveness of **AutoDP**.

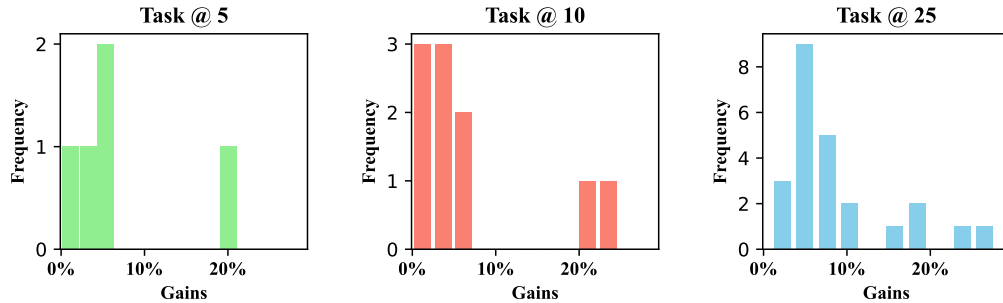


Figure 2: Histogram of task gains for **AutoDP** in terms of Averaged Precision.

4.3 Hyperparameter & Complexity Analysis

We analyze the effect of two vital hyperparameters of our method: K_1 and B , since they are the crucial parameters that largely define the complexity of our method during searching and inference respectively. We choose the setting of **Task @ 25** for a comprehensive analysis of all tasks. We try out different values and report the corresponding performance gain (**AVP**) in Figure 3.

First, K_1 determines the number of training samples collected during searching. Given that each sample invokes an MTL procedure, it constitutes the major portion of the search cost. Therefore, our goal is to find an optimal value for K_1 , striking a balance between cost-effectiveness and achieving commendable performance. We notice that the performance change plateaus after K_1 reaches 25.

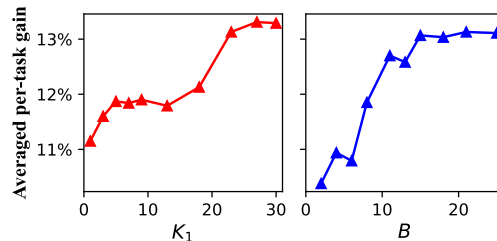


Figure 3: Analysis for the number of progressive sampling rounds K_1 and the budget of task groups B under the setting of Task @ 25.

That is, the surrogate model effectively learns the distribution of the search space after consuming 25×25 training samples during active selection (25 samples per round). Consequently, we can empirically decide to halt the iteration at this point.

Second, B determines the number of task groups for the final configuration, which indicates the number of MTL models required for achieving the expected performance gain after searching. We also observe similar phenomenon that the performance becomes stable after B reaches 12. We could also choose the optimal value for B accordingly.

4.4 Ablation Study

We further analyze the effect of several components within **AutoDP**, including progressive sampling (Section 3.4), greedy search (Section 3.5), and task grouping as a whole. We replace these components with naive or human intuition-inspired baselines and report the performances in Table 2. Removing any of the components from the original framework leads to noticeable performance decreases, demonstrating the effectiveness of the designed components.

Table 2: Ablation results in terms of **AVP**.

Settings	Task @ 5	Task @ 10	Task @ 25
AutoDP	+7.74	+8.81	+13.43
Random sampling	+6.75	+7.04	+11.30
Random search	+6.89	+7.15	+12.04
Disease grouping	+6.29	+6.99	+8.61

Specifically, we replace progressive sampling and greedy search with purely random methods, referred to as Random Sampling and Random Search. In all three settings, performance generally decreases, highlighting the contributions of these components of **AutoDP**.

Additionally, we use disease-based grouping (Appendix B) to first assign tasks into different groups based on their medical relevance and then employ DARTS to search for the model architecture for each group. This allows us to analyze the effectiveness of automated task grouping. By comparing disease-based grouping with the searched configurations (Appendix C), we observe that **AutoDP** does not strictly follow medical classifications for task grouping but achieves significant performance improvements over disease-based grouping. This indicates the necessity of using an automated search algorithm to find the optimal task grouping, which surpasses human intuition.

5 Conclusions and Future Work

In this paper, we propose **AutoDP**, an automated multi-task learning framework for joint disease prediction on EHR data. Compared to existing work, our method largely improves the design of task grouping and model architectures by reducing human interventions. Experimental results on real-world EHR data demonstrate that the proposed framework achieves significant improvement over existing state-of-the-art methods, while maintaining a feasible search cost. There are also some valuable future directions based on the current version of **AutoDP**.

First, from the application perspective, if we aim at deploying **AutoDP** to real-world healthcare systems, it would be advantageous to apply it to more complex problem settings. For example, the incorporation of diverse clinical data sources beyond EHR such as claims, drugs, medical images and texts will significantly enhance the practical utility of **AutoDP**.

Additionally, considering the dynamic nature of healthcare environments with continuously updated input data and evolving tasks, adapting the surrogate model to accommodate new data and tasks would be imperative.

Moreover, addressing privacy concerns within healthcare systems is a promising direction. Therefore, extending **AutoDP** with data processing pipelines for automatic feature engineering could offer enhanced privacy safeguards and further improve its applicability in sensitive healthcare contexts.

Finally, we assume all the tasks have the same input EHR data in our problem setting, which might not always be the case in practical scenarios. Chances are that, for some diseases, there are large and well-annotated data, while for the others, there are limited data available. How we should extend the current framework to handle more heterogeneous diseases/tasks remains a challenge.

References

- [1] Liantao Ma, Junyi Gao, Yasha Wang, Chaohe Zhang, Jiangtao Wang, Wenjie Ruan, Wen Tang, Xin Gao, and Xinyu Ma. Adacare: Explainable clinical health status representation learning via scale-adaptive feature extraction and recalibration. In *AAAI*, 2020.
- [2] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine learning for healthcare conference*, pages 301–318. PMLR, 2016.
- [3] Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1903–1911, 2017.
- [4] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*, 2019.
- [5] Qiuling Suo, Fenglong Ma, Giovanni Canino, Jing Gao, Aidong Zhang, Pierangelo Veltri, and Gnasso Agostino. A multi-task framework for monitoring health conditions via attention-based recurrent neural networks. In *AMIA annual symposium proceedings*, volume 2017, page 1665. American Medical Informatics Association, 2017.
- [6] Narges Razavian, Jake Marcus, and David Sontag. Multi-task prediction of disease onsets from longitudinal laboratory tests. In *Machine learning for healthcare conference*, pages 73–100. PMLR, 2016.
- [7] Xiang Wang, Fei Wang, Jianying Hu, and Robert Sorrentino. Exploring joint disease risk prediction. In *AMIA annual symposium proceedings*, volume 2014, page 1180. American Medical Informatics Association, 2014.
- [8] Xiongjun Zhao, Xiang Wang, Fenglei Yu, Jiandong Shang, and Shaoliang Peng. Unimed: Multimodal multitask learning for medical predictions. In *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1399–1404. IEEE, 2022.
- [9] Enliang Xu, Shiwan Zhao, Jing Mei, Eryu Xia, Yiqin Yu, and Songfang Huang. Multiple mace risk prediction using multi-task recurrent neural network with attention. In *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–2. IEEE, 2019.
- [10] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020.
- [11] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.
- [12] Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516, 2021.
- [13] Xiaozhuang Song, Shun Zheng, Wei Cao, James Yu, and Jiang Bian. Efficient and effective multi-task grouping via meta learning on task combinations. *Advances in Neural Information Processing Systems*, 35:37647–37659, 2022.
- [14] Chanhoh Ahn, Eunwoo Kim, and Songhwai Oh. Deep elastic networks with model selection for multi-task learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6529–6538, 2019.
- [15] Felix JS Bragman, Ryutaro Tanno, Sebastien Ourselin, Daniel C Alexander, and Jorge Cardoso. Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1385–1394, 2019.
- [16] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33:8728–8740, 2020.
- [17] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In *International conference on machine learning*, pages 3854–3863. PMLR, 2020.

- [18] Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia, and Wei Liu. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 11543–11552, 2020.
- [19] Jonathan Waring, Charlotta Lindvall, and Renato Umeton. Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial intelligence in medicine*, 104:101822, 2020.
- [20] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.
- [21] Lijun Zhang, Xiao Liu, and Hui Guan. Automtl: A programming framework for automating efficient multi-task learning. *Advances in Neural Information Processing Systems*, 35:34216–34228, 2022.
- [22] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512, 2016.
- [23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.
- [24] Shiqing Liu, Haoyu Zhang, and Yaochu Jin. A survey on surrogate-assisted efficient neural architecture search. *arXiv preprint arXiv:2206.01520*, 2022.
- [25] Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. Mimic-iv. *PhysioNet*. Available online at: [https://physionet.org/content/mimiciv/1.0/\(accessed August 23, 2021\)](https://physionet.org/content/mimiciv/1.0/(accessed August 23, 2021)), 2020.
- [26] Cao Xiao, Edward Choi, and Jimeng Sun. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association*, 25(10):1419–1428, 2018.
- [27] Hrayr Harutyunyan, Hrant Khachatryan, David C Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific data*, 6(1):96, 2019.
- [28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [29] Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. D-vae: A variational autoencoder for directed acyclic graphs. *Advances in Neural Information Processing Systems*, 32, 2019.
- [30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [31] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.
- [32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [33] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, 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*, 3(1):1–9, 2016.
- [34] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Appendix

We include the following sections in Appendix, providing the additional details of the proposed framework, additional experimental results, and a discussion about limitation and future work.

A Implementation

To prepare our dataset, we adopt the data pre-processing pipeline outlined in Harutyunyan, et al. [27]. Given that the original implementation² is designed for MIMIC-III [33], we make specific modifications to tailor it for MIMIC-IV. The 25 labels are defined using the Clinical Classifications Software (CCS) for ICD-9 code³. Consequently, we first map the ICD-10 codes⁴ in the MIMIC-IV database to ICD-9 codes before generating the labels. After processing, we have the feature dimension d_e as 76. We partition the dataset as train, validation and test sets with a ratio of 0.7 : 0.15 : 0.15.

We implement the framework using the PyTorch framework and run it on an NVIDIA A100 GPU. Given the dataset we have, we first train a vanilla LSTM for every task independently, and report the backbone performance in Table 3, which can be further used to compute multi-task gains. For the proposed method, we run three settings of experiments: **Task @ 5**, **Task @ 10** and **Task @ 25**, which refers to using the first 5 tasks, 10 tasks and 25 tasks respectively. For different settings, we use specific hyperparameters as shown in Table 4. Besides that, we define the candidate operation set $\mathcal{O}$ as {Identity, Zero, FFN, RNN, Attention}, which includes widely used operations for processing EHR time series. Among them, Identity means maintaining the output identical to the input. Zero means setting all the values of the input feature to 0. Attention and FFN represents one self-attention layer and one feed-forward layer respectively, which are the same as in Transformer [30]. RNN is one recurrent layer, and we adopt LSTM [32] in our framework. For all the MTL procedures and baseline training, we apply the batch size of 64 and learning rate of $3e - 4$. For training the surrogate model, we use the batch size of 5 and learning rate of $5e - 5$. During searching, we compute all multi-task gains on the validation set for guiding the surrogate model training. After we obtain the optimal configuration, we train the searched models and report their multi-task gains on the test set.

B Disease Based Grouping

To show the effectiveness of automated task grouping, we conduct experiments using a predefined task grouping based on disease categories. We asked GPT-4 [34] to classify the 25 prediction tasks into different groups based on their medical meaning. The result is shown in Table 5. Using this grouping, we further apply DARTS to each group and report the multi-task gains as shown in Table 2. Compared to **AutoDP**, there is a notable performance drop for the disease based grouping. This means human intuition dose not provide the optimal task grouping, which underscores the necessity of employing search algorithm to automatically discover better task grouping for MTL.

C Visualization of the Searched Configurations

Here, we show two example of the final configuration for setting **Task @ 10** in Figure 4 and for **Task @ 25** in Figure 5. The proposed **AutoDP** identifies 5 and 10 different task groups respectively and also searches for the corresponding architectures. We can observe that some of the tasks tend to be trained independently, while others are grouped together for joint training. This supports our claim that fine-grained task grouping is necessary to bring the optimal performance gain. Also, the optimal architecture is also different for each task group, which further justifies the necessity of joint optimization over task grouping and architecture search.

²<https://github.com/YerevaNN/mimic3-benchmarks>

³<https://www.cdc.gov/nchs/icd/icd9.htm>

⁴<https://www.cms.gov/medicare/coding-billing/icd-10-codes/2018-icd-10-cm-gem>

Table 3: Performance of the single task backbone.

Task	ROC	AVP
Acute and unspecified renal failure	0.7827	0.5647
Acute cerebrovascular disease	0.9079	0.4578
Acute myocardial infarction	0.7226	0.1761
Cardiac dysrhythmias	0.6948	0.5168
Chronic kidney disease	0.7296	0.4383
Chronic obstructive pulmonary disease and bronchiectasis	0.6791	0.2689
Complications of surgical procedures or medical care	0.7229	0.4045
Conduction disorders	0.6712	0.1880
Congestive heart failure; nonhypertensive	0.7601	0.5129
Coronary atherosclerosis and other heart disease	0.7351	0.5589
Diabetes mellitus with complications	0.8844	0.5559
Diabetes mellitus without complication	0.7484	0.3355
Disorders of lipid metabolism	0.6730	0.5816
Essential hypertension	0.6298	0.5258
Fluid and electrolyte disorders	0.7396	0.6129
Gastrointestinal hemorrhage	0.7076	0.1281
Hypertension with complications and secondary hypertension	0.7141	0.4243
Other liver diseases	0.6849	0.2303
Other lower respiratory disease	0.6371	0.1417
Other upper respiratory disease	0.7602	0.2228
Pleurisy; pneumothorax; pulmonary collapse	0.7051	0.1417
Pneumonia	0.8171	0.3786
Respiratory failure; insufficiency; arrest (adult)	0.8651	0.5497
Septicemia (except in labor)	0.8291	0.4866
Shock	0.8792	0.5574

Table 4: Hyperparameter setting.

Parameters		Task @ 5	Task @ 10	Task @ 25
# of tasks	N	5	10	25
Dimension of $F(\cdot)$	d_s	64	64	64
# of nodes	P	2	2	3
Progressive sampling	Q_0	10	10	20
	Q_1	50	100	100
	Q_2	10	20	20
	λ	0.5	0.5	0.5
	K_1	20	30	25
Greedy search	K_2	1000	1000	1000
	B	3	5	10
Runtime	GPU Hours	~ 20	~ 75	~ 200

Table 5: Disease Based Grouping.

Groups	Diseases
Cardiovascular Diseases	Acute cerebrovascular disease Acute myocardial infarction Cardiac dysrhythmias Congestive heart failure; nonhypertensive Coronary atherosclerosis and other heart disease Essential hypertension
Respiratory Diseases	Chronic obstructive pulmonary disease and bronchiectasis Other lower respiratory disease Other upper respiratory disease Pleurisy; pneumothorax; pulmonary collapse Pneumonia (except that caused by tuberculosis or sexually transmitted disease) Respiratory failure; insufficiency; arrest (adult)
Kidney Diseases	Acute and unspecified renal failure Chronic kidney disease
Metabolic Diseases	Diabetes mellitus with complications Diabetes mellitus without complication Disorders of lipid metabolism Fluid and electrolyte disorders
Gastrointestinal Diseases	Gastrointestinal hemorrhage
Infections	Septicemia (except in labor)
Surgical/Medical Complications	Complications of surgical procedures or medical care
Neurological/Cardiac Conditions	Conduction disorders Shock
Liver Diseases	Other liver diseases

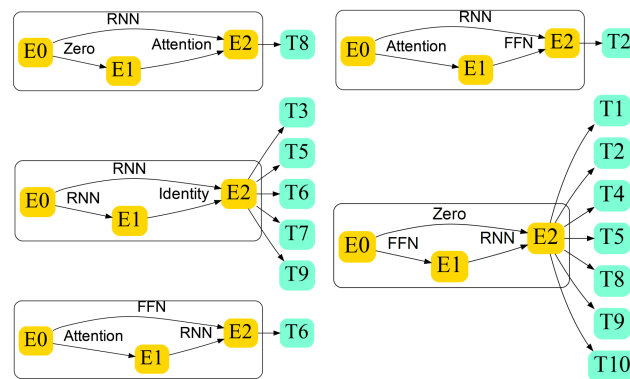


Figure 4: Illustration of the searched configuration under the setting of Task @ 10.

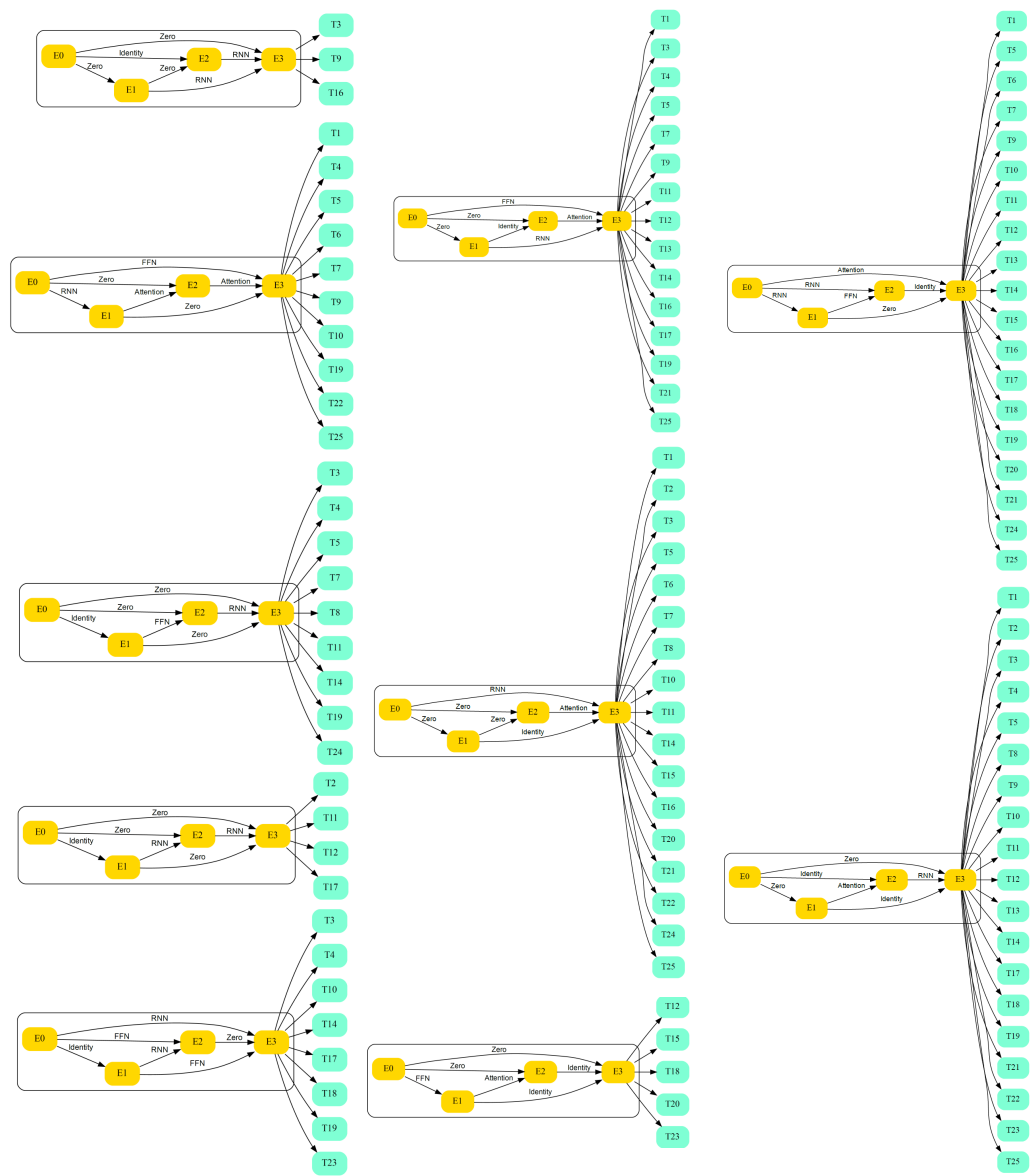


Figure 5: Illustration of the searched configuration under the setting of Task @ 25.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The introduction section presents all contributions and scope.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We have discuss the limitation and future work in Section 5.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: We do not have theory assumptions in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We have introduced all implementation details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: We have uploaded the code in supplementary materials, and we use public data.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: We have introduced all implementation details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: We include error bars and statistical test for the proposed method.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: We provide full information for the computing resources.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines?>

Answer: [\[Yes\]](#)

Justification: We have read the code of ethics and make sure the paper conform with it.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[NA\]](#)

Justification: There are negligible societal impacts of the work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: There are no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have followed the standard procedure for accessing MIMIC-IV dataset.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We do not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We do not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: We do not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.