Connectivity Patterns are Task Embeddings

Zhiheng Xi†, Rui Zheng†, Yuansen Zhang, Xuanjing Huang, Zhongyu Wei, Minlong Peng, Mingming Sun, Qi Zhang††, Tao Gui††

1 School of Computer Science, Fudan University, Shanghai, China
2 School of Data Science, Fudan University, Shanghai, China
3 Cognitive Computing Lab Baidu Research, Beijing, China
4 Institute of Modern Languages and Linguistics, Fudan University, Shanghai, China

{zhxi22,zhangys22}@m.fudan.edu.cn, {pengminlong, sunmingming01}@baidu.com,
{rzheng20,xjhuang,zywei,qz,taogui}@fudan.edu.cn

Abstract

Task embeddings are task-specific vectors designed to construct a semantic space of tasks, which can be used to predict the most transferable source task for a given target task via the similarity between task embeddings. However, existing methods use optimized parameters and representations as task embeddings, resulting in substantial computational complexity and storage requirements. In this work, we draw inspiration from the operating mechanism of deep neural networks (DNNs) and biological brains, where neuronal activations are sparse and task-specific, and we use the connectivity patterns of neurons as a unique identifier associated with the task. The proposed method learns to assign importance masks for sub-structures of DNNs, and accordingly indicate the task-specific connectivity patterns. In addition to the storage advantages brought by the binary masking mechanism and structured sparsity, the early-bird nature of the sparse optimization process can deliver an efficient computation advantage. Experiments show that our method consistently outperforms other baselines in predicting inter-task transferability across data regimes and transfer settings, while keeping high efficiency in computation and storage.

1 Introduction

With the rapid development and excellent performance of large pre-trained language models (PLMs), the most prevalent paradigm in natural language processing (NLP) has become pre-training then fine-tuning (Peters et al., 2018; Devlin et al., 2019a; Brown et al., 2020; Lewis et al., 2020; Raffel et al., 2020). Extending upon the two-step training procedure, previous works show that intermediate-task transfer, i.e., fine-tuning the model on an intermediate source task before the target task, can yield further gains (Phang et al., 2018; Wang et al., 2019a). Nevertheless, the improvement by intermediate-task transfer heavily relies on the selection of a proper intermediate task because some source tasks lead to performance degradation (Yogatama et al., 2019; Prukachatkan et al., 2020). One straightforward approach is to enumerate every possible (source, target) task combination, but it is extremely expensive. Therefore, recent works explore methods to predict inter-task transferability accurately with high efficiency.

The current state-of-the-art (SOTA) works are established on task embeddings, (i.e., leveraging a single vector to represent a task). They predict inter-task transferability by computing the similarity between task embeddings. Task2Vec (Achille et al., 2019; Vu et al., 2020) develops task embeddings based on the Fisher information matrix while requiring fine-tuning the full model and consuming a large amount of storage (Zhou et al., 2022). Recently, researchers propose that the efficiently tuned parameters like prompts (Li and Liang, 2021; Liu et al., 2021) and LoRA (Hu et al., 2022) encode rich information for a task and thus can serve as task embeddings (Poth et al., 2021; Vu et al., 2022;
Zhou et al., 2022). However, these tuned parameters are sensitive to model initialization and stochasticity (Li and Liang, 2021; Lester et al., 2021), and optimizing these parameters consumes significantly more computational resources than traditional fine-tuning (Ding et al., 2022).

Different from them, we draw inspiration from the shared working mechanisms of DNNs and biological brains to develop high-quality task embeddings. We start by considering which parts of knowledge within the model are being utilized for a given task. Typically, recent works in sparse optimization and model pruning have shown that sub-structures (e.g., neurons, attention heads, channels, and layers) from different parts of the model exhibit specialization in distinct knowledge and possess varying degrees of importance for a particular task (Dalvi et al., 2020; Liu et al., 2017; Voita et al., 2019a; Glorot et al., 2011; Georgiadis, 2019; Li et al., 2022). These are consistent with the findings in neuroscience that activities of neurons and connectivities in biological brains are sparse (Kerr et al., 2005; Poo and Isaacson, 2009; Barth and Poulet, 2012) and task-specific (Duncan, 2010; Fox et al., 2005; Crinion et al., 2003; Newton et al., 2007). The aforementioned remarkable findings motivate us to use task-specific connectivity patterns in DNNs to represent tasks.

In this work, we propose a novel task embedding, namely Connectivity Patterns as Task Embedding (CoPATE), and apply it to predict the inter-task transferability, as illustrated in Figure 1. Our key insight is that in over-parameterized DNNs, there exist connectivity patterns (i.e., the structures of subnetworks) that are functional for one certain task, and can capture high-density task-specific information. Concretely, we assign importance masks to attention heads and intermediate neurons of PLMs, jointly train the masks and the model, and extract task embeddings according to the learned masks. Our method has two strengths in efficiency: 1) it is computation-friendly as we extract connectivity patterns early in the training; 2) it is storage-friendly because our embedding granularity is coarse-grained, and CoPATE can be represented by a binary mask. Experiments show that compared to other approaches, CoPATE has superior inter-task prediction capability across data regimes and transfer settings. Our codes are available at Github\(^1\).

Our contributions can be summarized as follows:

- Inspired by the working mechanisms of DNNs and biological brains, we propose CoPATE, a novel task embedding that represents tasks with sparse connectivity patterns.
- We propose a method to obtain CoPATE with sparse optimizing techniques, and show the significant positive correlation between embedding similarity and task transferability.
- We conduct thorough experiments on 342 transfer combinations with different settings to show the effectiveness of our method. We further explore an intermediate-curriculum transfer setting to investigate whether there is a beneficial curriculum for a target task.

2 Identifying Sparse, Task-specific Connectivity Patterns

In this section, we demonstrate the framework to identify task-specific connectivity patterns. We represent the task-specific connectivity patterns via the structure of essential subnetworks found by sparse optimizing and pruning techniques (Liu et al., 2017; Chen et al., 2021a; Zheng et al., 2022), including the searching stage (Sec 2.1) and the extracting stage (Sec 2.2).

2.1 Finding Connectivity Patterns

Typically, BERT is constructed by multiple transformer encoder layers that have uniform structure (Vaswani et al., 2017). Each layer has a multi-head self-attention (MHA) block, a feed-forward network (FFN), and residual connections around each block. The MHA is formulated as:

\[
\text{MHA}(x) = \sum_{i=1}^{N_h} \text{Att}(W^K_i, W_Q^i, W_V^i, W_O^i)(x),
\]

where \(x\) is input, \(N_h\) is the number of heads, and the projections \(W^K_i, W_Q^i, W_V^i, W_O^i \in \mathbb{R}^{d_h \times d_f}\) denote the key, query, value and output matrices in the \(i\)-th attention head. Here \(d\) is the hidden size (e.g., 768), and \(d_h = d/N_h\) denotes the output dimension of each head (e.g., 64).

An FFN parameterized by \(W_U \in \mathbb{R}^{d \times d_f}\) and \(W_D \in \mathbb{R}^{d_f \times d}\) comes next:

\[
\text{FFN}(x) = \text{gelu}(XW_U) \cdot W_D,
\]

where \(d_f = 4d\).

\(^1\)https://github.com/WooooDyy/CoPaTE
Learnable Importance Masks We adopt a coarse-grained structured pruning strategy to shape connectivity patterns. Specifically, we use the modified network pruning approach (Liu et al., 2017; Chen et al., 2021a) to find which heads and intermediate neurons are essential for a given task. We first assign learnable importance masks to each head and intermediate neuron:

\[
MHA(x) = \sum_{i=1}^{N_h} m_H^i \cdot \text{Att}W_{K_i}W_{Q_i}W_{V_i}(x),
\]

\[
\text{FFN}(x) = m_V \cdot \text{gelu}(XW_U) \cdot W_D,
\]

where \(m_H^i\) denotes the masks for heads, \(i\) is the index of head, and \(m_V\) denotes the masks for FFN. Then, we can jointly train BERT with importance masks but with a sparsity-inducing regularizer:

\[
\mathcal{R}(m) = \lambda_H \|m_H\|_1 + \lambda_V \|m_V\|_1,
\]

where \(m = \{m_H, m_V\}\), \(\lambda_H\) and \(\lambda_V\) denote regularization strength for the two kinds of masks respectively. Hence, the final optimizing objective is:

\[
\min_{\theta, m} \mathcal{L}(\theta, m) + \mathcal{R}(m),
\]

where \(\mathcal{L}\) is the original loss function of fine-tuning.

2.2 Extracting Connectivity Patterns

Early-stopping Strategy Note that the joint training is still as expensive as traditional fine-tuning. Fortunately, (You et al., 2020) and (Chen et al., 2021b) point out that the importance masks converge early in the searching stage. This inspires us to stop the joint training early and dig out early-bird connectivity patterns to generate task embeddings. Nevertheless, it is difficult to determine the exact search termination time as the termination moments of different tasks are different. Moreover, masks of MHA and FFN typically have different convergence rates. Hence, we adopt a termination metric following (Xi et al., 2022) which terminates the searching process when the normalized mask distances between several consecutive miniepochs are all smaller than a threshold \(\gamma^2\).

Pruning Strategy After the joint training, we can perform pruning to the original models to extract important connectivity patterns that encode task-specific information. Specifically, the self-attention heads and intermediate neurons with the smallest importance masks are believed to contribute the least to the task and the corresponding masks are set to 0, while the masks of the surviving elements

\[\text{Appendix C}\] for more details of the termination metric.
**Algorithm 1: COPATE Generation**

**Input:** model parameters $\theta$, learnable importance masks $m$, learning rate $\eta$, sparsity for self-attention heads $p_h$, and sparsity for intermediate neurons $p_f$.

1 **Procedure** TASK-SPECIFIC CONNECTIVITY PATTERNS SEARCHING

   2 Initialize $\theta$ to pre-trained weights;
   3 Initialize $m = \{m_h, m_f\}$ to $1$;
   4 repeat
   5 \[ \theta = \theta - \eta \nabla_{\theta} (\mathcal{L}(\theta, c) + \mathcal{R}(c)); \]
   6 \[ m = m - \eta \nabla_{m} (\mathcal{L}(\theta, m) + \mathcal{R}(m)); \]
   7 until the convergence condition in Sec.2.2 is satisfied, or the fine-tuning is done;

8 **Procedure** GENERATING COPATE WITH LEARNED MASKS

9 Reset $m_h$ and $m_f$ to binary form with $p_h$ and $p_f$ according to mask magnitudes, respectively;

10 $\text{Emb} = [m_h; m_f]$.

are set to $1$. Therefore, we can generate storage-efficient task embeddings with the resulting model structure.

3 COPATE: Connectivity Patterns as Task Embedding

In this section, we first show how we generate task embeddings with task-specific connectivity patterns at hand (Sec 3.1). Next we provide empirical evidence for the appropriateness of using the obtained task embeddings to predict inter-task transferability in Sec. 3.2.

3.1 Task Embedding Generating

Typically, the structure of a neural network can be represented as a mask vector:

\[ m = [m^1, m^2, ..., m^N], \quad m^i \in \{0, 1\}, \]

where $N$ denotes the number of elements (i.e., substructures) that construct the network and the value of mask $m^i$ indicates whether the $i$-th element is pruned or not. In our framework, the elements are self-attention heads and intermediate neurons, so the structured subnetworks are represented by:

\[ m_h = [m^0_h, m^1_h, ..., m^N_h], \]

\[ m_f = [m^0_f, m^1_f, ..., m^N_f], \]

where $N_L$ denotes the number of transformer layers, $N_h$ denotes the number of heads in each layer and $N_f$ is the number of intermediate neurons in each layer. Hence, the resulting task embedding is:

\[ \text{Emb} = [m_h; m_f]. \]

We summarize the procedure of generating COPATE in Algorithm 1. COPATE is quite storage-efficient owing to its binary form. For example, BERT\textsubscript{BASE} consumes only $4626$ bytes to store\textsuperscript{3}.

3.2 Positive Correlation between COPATE Similarity and Task Transferability

We first calculate the similarity between COPATEs of different tasks with Hamming Similarity, which is defined as the number of positions at which the corresponding symbols are the same:

\[ \text{Sim}(V_1, V_2) = \frac{\sum_{i=1}^{n} \sigma(V_1[i], V_2[i])}{n}, \]

where $\sigma(v_1, v_2) = 1$ if $v_1 = v_2$ else $0$. Since the numbers of self-attention heads and intermediate neurons differ significantly, we calculate the similarity of the two types of elements separately, and each contributes equally to the final similarity.

We then explore whether the similarity between COPATEs is correlated with task transferability. We calculate related transfer gain to measure the impact of transfer learning. Specifically, given a source task $s$ and a target task $t$, if a baseline PLM that is directly fine-tuned on the target dataset (without any intermediate transferring) achieves a performance of $T(t)$, while a transferred model achieves a performance of $T(s, t)$, the relative transfer gain can be expressed as: $G(s, t) = \frac{T(s, t) - T(t)}{T(t)}$.

Figure 2 shows how the relative transfer gain changes as a function of the similarity between the source and target task embeddings. Overall, there is a significant positive correlation between the similarity of task embeddings and task transferability on the majority of the target tasks (16 out of 19). It is possible for the correlation coefficient to attain a high magnitude in many cases, such as on the DROP task, where the correlation coefficient is $0.78 \ (p = 0.00013)$.

The exciting results suggest that COPATE is promising in accurately predicting inter-task transferability. Concretely, for a novel target task, we

\textsuperscript{3}BERT\textsubscript{BASE} has $(12 \times 12)$ heads and $(3072 \times 12)$ intermediate neurons, and requires $37008$ bits $= 4626$ bytes to store.
Table 1: Evaluation results of intermediate task selection methods. In-class means that the candidate source tasks have the same type as the target task, while all-class means the candidate source tasks come from all types of tasks. EP means epochs to search for connectivity patterns. R1 denotes Regret@1 and R3 denotes Regret@3. For NDCG, higher is better; for Regret, lower is better. The best performance in each group is highlighted in **bold**.

### 4 Predicting Task Transferability

In this section, we perform thorough experiments to empirically demonstrate the capability of CoPATE in predicting inter-task transferability.

#### 4.1 Experimental Setup

**Datasets** We conduct experiments with 8 tasks of text classification or regression (CR) and 11 tasks of question answering (QA) following previous works (Vu et al., 2020; Zhou et al., 2022). We list the datasets in Appendix A.

**Data Regimes** For every (source, target) dataset pair, we perform transfer experiments in three data regimes to simulate real-world situations: FULL → FULL, FULL → LIMITED, and LIMITED → LIMITED. The FULL regime includes all training data, while in LIMITED settings, we limit the amount of training data by randomly selecting 1K training examples.

#### Baselines

We compare our method with following strong baselines: (1) **TEXT EMB** (Vu et al., 2020) averages sentence representations by BERT over the whole dataset. (2) **TASK EMB** (Achille et al., 2019; Vu et al., 2020) embeds tasks based on the Fisher information matrix which captures the curvature of the loss surface. (3) **PTUNING** (Vu et al., 2022) interprets the fine-tuned soft prompts in each transformer layer as task embeddings. (4) **LORA** (Zhou et al., 2022) injects trainable rank decomposition matrices into layers of the model and takes the fine-tuned matrices as task embeddings.

**Evaluation Metrics** We use the following metrics to evaluate the performance of methods:

1. **Normalized Discounted Cumulative Gain (NDCG)** (Järvelin and Kekäläinen, 2002) is a broadly used information retrieval metric aiming to evaluate the quality of a ranking with attached relevances, and it penalizes top-ranked and bottom-ranked mismatches with different weight\(^4\).

2. **Regret@k** (Renggli et al., 2022) measures the relative performance difference between the top \(k\) selected

\(^4\)See Appendix D for more details about NDCG.
source tasks and the optimal source task in our experiments, we include \( k = 1 \) and \( k = 3 \).

**Implementation Details** We perform transfer experiments with all (source, target) combinations and use BERT\(_{BASE}\) (Devlin et al., 2019b) as the backbone. All the intermediate tuning and target tuning take 3 epochs. For \( \text{FULL} \to \text{FULL} \) regime, we use the results from (Vu et al., 2020). We implement all baseline methods according to their open-source codes and the Transformers library (Wolf et al., 2020). When searching for connectivity patterns in our method, we jointly train the masks and the BERT model for 5 epochs. When extracting early-bird embeddings (i.e., EARLY-EMB), we set the max searching epoch number to 1. We perform 5 restarts for stable results in \( \text{LIMITED} \) regimes. See Appendix F for more details.

## 4.2 Experimental Results

Table 1 demonstrates the detailed evaluating results. Overall, the proposed COPATE achieves superior performance across task types, transfer scenarios and data regimes, revealing that it is a robust and accurate predictor of beneficial transfer.

**FULL \( \to \) FULL** In this regime, our method attains impressive performance compared to other baselines. For example, in the setting of \( \text{in-class} \) transfer of Classification tasks, COPATE exceeds the most competitive baseline by 1.0 in NDCG, and the Regret@3 score achieves 1.2. It is also observed that excessive training steps for identifying task-specific connectivity patterns do not necessarily result in large performance improvement in this regime. The efficient EARLY-EMB performs slightly worse than \( \text{LTH}_{EP=5} \), but still performs comparably.

**FULL \( \to \) LIMITED** In this few-shot regime, our method achieves comparable performance to SOTA baselines. However, we find that in QA tasks, the performance of COPATE degrades sharply as the number of training steps utilized during the search stage decreases. Compared to \( \text{LTH}_{EP=5} \), EARLY-EMB’s NDCG on in-class and all-class decreased by 10.1 and 8.9, respectively. This trend is also observable in \( \text{LIMITED} \to \text{LIMITED} \) regime. It is not surprising as QA tasks are typically more complex and the connectivity patterns require more training steps to converge better. This suggests a trade-off between performance and efficiency when facing limited examples, and additional training resources should be allocated to the search stage to extract high-quality task embeddings.

**LIMITED \( \to \) LIMITED** In this regime, COPATE demonstrates exceptional performance and surpasses other existing baselines by a significant margin. For instance, our method outperforms the strongest baseline by 9.5 in terms of NDCG on in-class transfer of QA tasks, and 4.6 on all-class transfer of QA tasks.

## 5 Discussion

### 5.1 Ablation Study

In this section, we perform ablation studies to show the contribution of each component of our method.

**Head vs. FFN** Previous experiments utilize both masks of attention heads and intermediate neurons to compute similarity. Here, the contribution of each component is evaluated individually by separately using them to calculate similarity and subsequently assessing the NDCG. Table 2 shows that both components play essential roles in ranking source tasks. We observe that on CR tasks, heads outperform FFN by a large margin, revealing that heads are more important in such tasks.

**Impact of Sparsity** Figure 3 illustrates the relationship between the level of sparsity and the performance of the obtained embeddings. The performance of the model is significantly impacted by variations in the pruning ratio of heads or FFN when the target tasks are CR, while such variations have a limited effect when the target tasks are QA, revealing that CR tasks are more sensitive to embedding sparsity. After comprehensive consideration, we believe that 1/3 and 0.4 are reasonable sparsity for heads and FFN, respectively.
We average the results on all datasets. The #Time is quantified as a multiple of the duration of the traditional fine-tuning for a single epoch. We get results from one NVIDIA 3090 GPU for a fair comparison. The #Storage is in bytes and each float number requires 4 bytes.

We include more ablation studies of pruning strategies, early-stopping thresholds, and the sparsity-inducing regularizer in Appendix I.

### 5.2 Computation and Storage Consumption

Table 3 lists the computational and storage cost of each method. CoPATE demonstrates efficiency in both aspects thanks to proper designs (i.e., early-stopping, structured pruning and binary form of embeddings), particularly EARLY-EMB, which exhibits the fastest generation speed and only requires 4.6K bytes to store. TASKEMB is also computation-efficient, but it requires much more storage than CoPATE. While TEXTEMB is the only method that is comparable to our approach in terms of efficiency, it falls behind EARLY-EMB with an average difference of 1.6 in NDCG.

#### Further Storage-efficiency with Task-specific Layers

Previous studies have established that in BERT, layers are redundant (Dalvi et al., 2020), and that shallower transformer layers contain more general information while deeper layers contain more task-specific information (Voita et al., 2019a; Kim et al., 2020; Sajjad et al., 2020). These insights shed light on further reducing the storage of CoPATE by representing tasks using a select number of layers, or even a single layer. Figure 4 illustrates the evaluated performance. We observe that: (1) Using a select number of layers does not result in a significant decrease in performance, and sometimes delivers better performance. (2) Top-down strategy outperforms bottom-up strategy, and consistently exceeds the full model in few-shot settings, showing that deep layers can effectively encode task-specific information, which is in line with previous studies. As a result, if we adopt the last six layers for embedding generation, 50% of the storage can be saved, while little decrease in performance is incurred. We also explore the potential of generating embeddings using a single layer, while sacrificing little performance in Appendix J.

#### 5.3 CoPATE Captures Task Relationships

The heatmap in Figure 5 illustrates the hierarchical clustering of the similarities between CoPATEs. The results indicate that the obtained embeddings effectively capture various intuitive task relationships. We observe that tasks with similar characteristics congregate in clusters, such as QA tasks (WiKiHop, SQuAD-1, SQuAD-2, DuoRC-s, DuoRC-p, NewsQA, and HotpotQA), similarity and paraphrasing tasks (STS-B and MRPC), NLI tasks (QNLI and MNLI), and single sentence classification tasks (SST-2 and CoLA). In particular, a closer examination of the clustering reveals that SQuAD1 and SQuAD2 are closely grouped together, with the latter being an extension of the former (Rajpurkar et al., 2016, 2018). Furthermore, the tight clustering of DuoRC-p and DuoRC-s is also noteworthy, as they are variations of the same movie plots with different lengths (Saha et al., 2018).

#### 5.4 Intermediate-curriculum Transfer

Here, we extend the boundary of intermediate-task transfer and examine the potential benefits of a specific intermediate task curriculum (i.e., a particular order to arrange several tasks) to a target task using CoPATE. Three distinct curriculum strategies are considered: (1) **Similar-first strategy** which selects the three tasks that are most similar to the

<table>
<thead>
<tr>
<th>Curriculum Type</th>
<th>Similar-first</th>
<th>Different-first</th>
<th>Recursive-similar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Gain</td>
<td>+2.35</td>
<td>+2.43</td>
<td>+2.56</td>
</tr>
</tbody>
</table>

Table 4: Performance gain yielded by each curriculum. The results are an average on all 19 tasks.
target task and arranges the intermediate tasks in a sequential order of similarity. (2) **Different-first strategy** which also selects the three tasks that are most similar to the target task, but arranges the intermediate tasks in an order of dissimilarity. (3) **Recursive-similar strategy** which starts from the target task, recursively finds the task that is most similar to the current task three times, stacks them, and then sequentially pops these found tasks for intermediate fine-tuning. The results in Table 4 show that: (1) Each curriculum can boost the target task, validating the value of intermediate-task transfer. (2) The recursive-similar strategy yields the most performance gain, suggesting that making each intermediate task learned better can deliver more benefits to target tasks. (3) The different-first strategy performs better than the similar-first, implying that intermediate tasks that are similar to the target task should be assigned later.

### 6 Related Work

**Predicting Beneficial Intermediate Tasks** It has been shown that intermediate-task transfer can deliver performance gains for many target tasks (Phang et al., 2018; Wang et al., 2019a; Talmor and Berant, 2019; Liu et al., 2019), but improper intermediate tasks can result in negative transfer results (Yogatama et al., 2019; Pruksachatkun et al., 2020). Hence, researchers try to accurately identify the most beneficial source task based on metadata or extracted representations of tasks (Alonso and Plank, 2017; Vu et al., 2020; Poth et al., 2021). Recent works represent tasks with embeddings that are generated from data representations (Vu et al., 2020), model weight information (Achille et al., 2019; Vu et al., 2020), and efficiently tuned parameters (Poth et al., 2021; Vu et al., 2022; Zhou et al., 2022). Different from them, we start from a model architecture perspective and use connectivity patterns to represent tasks.

**Techniques to Obtain Sparse Subnetworks** Researchers have explored a variety of techniques to obtain sparse networks by removing sub-structures like weights (Louizos et al., 2018; Frankle and Carbin, 2019; Sanh et al., 2020; Xu et al., 2021), channels (He et al., 2017; Luo et al., 2017; Liu et al., 2017; Molchanov et al., 2019), attention heads (Voita et al., 2019b; Michel et al., 2019; Li et al., 2021) and layers (Fan et al., 2020; Sajjad et al., 2020). These approaches first identify unimportant sub-structures and subsequently remove them. With the increasing size of PLMs, sparse subnetworks have become increasingly important for efficient deployment and inference in NLP, lead-
ing to a proliferation of related research (Prasanna et al., 2020; Hou et al., 2020; Lagunas et al., 2021; Xia et al., 2022). Our proposed method, which uses connectivity patterns as task embeddings, is orthogonal to these existing techniques.

7 Conclusion

In this work, we propose CoPATE, a novel task embedding that represents tasks with sparse connectivity patterns, and develop a method to get such embeddings. Comprehensive experiments show that the proposed method outperforms other competitive approaches in predicting inter-task transferability while achieving efficiency in both computation and storage. We hope that our work may motivate future work in introducing connectivity patterns as task embeddings to fields like meta learning, multi-task learning, and model interpretability.

Limitations

While the proposed method has demonstrated superior performance and high efficiency, there are several limitations that warrant further investigation: (1) In few-shot settings where the number of training examples is limited, the performance of our method and other baselines drops significantly. Future work should focus on uncovering essential features of the task in few-shot scenarios and generating embeddings of higher quality. (2) The storage consumption has been reduced to a small amount, however, the number of neurons is still relatively large compared to that of heads and therefore becomes a bottleneck for further decreasing storage requirements. As discussed in Sec 5.2, one possible solution is reducing the number of layers used to generate the embedding. Future work could also include assigning intermediate neurons into groups to make the embedding coarser in granularity, thus reducing storage requirements.

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No.62076069, 62206057, 61976056), Shanghai Rising-Star Program (23QA1400200), and Natural Science Foundation of Shanghai (23ZR1403500).

References


Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxia Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. CoRR, abs/2110.07602.


Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer


diverse, explainable multi-hop question answering.


Appendices

A List of Datasets

See Table 5 for details of datasets.

<table>
<thead>
<tr>
<th>Task</th>
<th>Train</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Text classification / Regression (CR)</strong></td>
<td></td>
</tr>
<tr>
<td>MNLI (Williams et al., 2018)</td>
<td>393K</td>
</tr>
<tr>
<td>QQP (Iyer et al., 2017)</td>
<td>364K</td>
</tr>
<tr>
<td>QNLI (Wang et al., 2019b)</td>
<td>105K</td>
</tr>
<tr>
<td>SST-2 (Socher et al., 2013)</td>
<td>67K</td>
</tr>
<tr>
<td>CoLA (Warstadt et al., 2019)</td>
<td>8.5K</td>
</tr>
<tr>
<td>STS-B (Cer et al., 2017)</td>
<td>7K</td>
</tr>
<tr>
<td>MRPC (Dolan and Brockett, 2005)</td>
<td>3.7K</td>
</tr>
<tr>
<td>RTE (Dagan et al., 2005)</td>
<td>2.5K</td>
</tr>
<tr>
<td><strong>Question Answering (QA)</strong></td>
<td></td>
</tr>
<tr>
<td>SQuAD-2 (Rajpurkar et al., 2018)</td>
<td>162K</td>
</tr>
<tr>
<td>NewsQA (Trischler et al., 2017)</td>
<td>120K</td>
</tr>
<tr>
<td>HotpotQA (Yang et al., 2018)</td>
<td>113K</td>
</tr>
<tr>
<td>SQuAD-1 (Rajpurkar et al., 2016)</td>
<td>108K</td>
</tr>
<tr>
<td>DuoRC-s (Saha et al., 2018)</td>
<td>100K</td>
</tr>
<tr>
<td>DuoRC-p (Saha et al., 2018)</td>
<td>86K</td>
</tr>
<tr>
<td>DROP (Dua et al., 2019)</td>
<td>77K</td>
</tr>
<tr>
<td>WikiHop (Welbl et al., 2018)</td>
<td>51K</td>
</tr>
<tr>
<td>BoolQ (Clark et al., 2019)</td>
<td>16K</td>
</tr>
<tr>
<td>ComQA (Abujabal et al., 2019)</td>
<td>11K</td>
</tr>
<tr>
<td>CQ (Bao et al., 2016)</td>
<td>2K</td>
</tr>
</tbody>
</table>

Table 5: The datasets used in our experiments, grouped by task class and sorted by training dataset size.

B More Results of Correlation between CoPATE Similarity and Inter-task Transferability

See Figure 6 for more results of correlation between CoPATE similarity and inter-task transferability. There is a significant positive correlation between the similarity of task embeddings and task transferability on most target tasks.

C More Details of Early-stopping Strategy

We use Hamming distance to calculate the normalized normalized mask distance. We stop the searching stage when the normalized mask distances between consecutive 5 miniepochs are all smaller than $\gamma$. Each miniepoch consists of 0.05 epochs. We set $\gamma$ to 0.05 in all settings. This is not the best choice for all transfer scenarios, but we unify the value of hyper-parameters for the sake of generality.

D More Details of NDCG

The NDCG is defined using the Discounted Cumulative Gain (DCG), which is a measure of the relevance score for a list of items, each discounted by its position in the ranking. The DCG of a ranking $R$ at a particular rank position $p$ can be calculated as:

$$DCG_p(R) = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i + 1)}$$

In our experiments, $R$ refers to a ranking of source tasks where the relevance $rel_i$ of the source task with rank $i$ is set to the averaged target performance, i.e. $rel_i \in [0, 100]$. We set $p = |S|$, which is the number of intermediate tasks.

The NDCG finally normalizes the DCG of the ranking predicted by the task selection approach ($R_{pred}$) by the golden ranking produced by the empirical transfer results ($R_{true}$). An NDCG of 100% indicates the best ranking.

$$NDCG_p(R) = \frac{DCG_p(R_{pred})}{DCG_p(R_{true})}$$

E More Details of Regret@k

Regret@k is defined as:

$$\text{Regret}_k = \frac{O(S, t) - M_k(S, t)}{\max_{s \in S} E[T(s, t)] - \max_{s \in S_k} E[T(s, t)]} \times 100\%$$

where $T(s, t)$ means the performance on target task $t$ when transferring from source task $s$. $O(S, t)$ is the expected target task performance of the optimal selection. $M_k(S, t)$ denotes the highest performance on $t$ among the $k$ top-ranked source tasks of the evaluated selection method. In our experiments, we include $k = 1$ and $k = 3$.

F More Implementation Details

For classification/regression tasks, we set the max sequence length to 128. For question answering tasks, we set the max sequence length to 384. The batch size for all experiments is set to 32. Our experiments are performed on twelve NVIDIA GeForce RTX 3090 GPUs. We perform 3 restarts for our experiments and report the mean. For
Figure 6: More results of correlation between CoPATE similarity and inter-task transferability. Each point represents a source task to a target task.
Method | CR |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EARLY-EMB</td>
<td>66.7</td>
</tr>
<tr>
<td>w/o Head</td>
<td>62.2</td>
</tr>
<tr>
<td>w/o FFN</td>
<td>66.7</td>
</tr>
</tbody>
</table>

Table 6: Ablation results when heads or intermediate neurons are removed from similarity computing in **FULL → LIMITED** regime.

<table>
<thead>
<tr>
<th>Method</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>EARLY-EMB</td>
<td>63.4</td>
</tr>
<tr>
<td>w/o Head</td>
<td>57.8</td>
</tr>
<tr>
<td>w/o FFN</td>
<td>63.3</td>
</tr>
</tbody>
</table>

Table 7: Ablation results when heads or intermediate neurons are removed from similarity computing in **LIMITED → LIMITED** regime.

PTUNING, we adopt P-Tuning v2 in (Liu et al., 2021), which implements a prompt tuning method by introducing additional attention prefix matrices to each transformer layer. We set the prefix length to 20. For LORA, we set the $r$ to 8 and $\alpha$ to 8. For the searching stage of winning tickets, we set the regularization strength $\lambda_H$ and $\lambda_F$ to $1e^{-4}$.

### G More Results of Head v.s. FFN

Table 6 and Table 7 show the results of Head v.s. FFN in **FULL → LIMITED** and **LIMITED → LIMITED**, respectively. We can still find that both of them are important for high-quality task embeddings.

### H More Results of Impact of Sparsity

Figure 7 and Figure 8 show the results of impact of sparsity in **FULL → LIMITED** and **LIMITED → LIMITED**, respectively. We can still find that 1/3 and 0.4 are reasonable sparsity for heads and FFN, respectively.

### I More Ablation Studies

#### I.1 Impact of Pruning Strategies

In this section, we investigate the impact of different pruning strategies to the embedding performance. Results in Table 8, Table 9 and Table 10 show that layerwise pruning and global pruning are proper strategies for self-attention heads and FFN, respectively.

#### I.2 Impact of Different Early-Stopping Thresholds

In this section, we investigate the impact of different early-stopping thresholds $\gamma$. Each line in the figure represents a data regime, and we report the mean results of different transfer settings. We can observe that the performance of embeddings converges when $\gamma$ reduces to near 0.05.

#### I.3 Importance of Sparsity-inducing Regularizer

In this section, we investigate the importance of the sparsity-inducing regularizer during the connectivity pattern searching stage. Results in Table 11 show that the regularizer is indispensable for
Table 8: Impact of pruning strategies in $\text{FULL} \rightarrow \text{FULL}$ regime. The results are NDCG scores averaged on different transfer settings.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>FFN-Global</th>
<th>FFN-Layerwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head-Global</td>
<td>76.2</td>
<td>78.7</td>
</tr>
<tr>
<td>Head-Layerwise</td>
<td>$\mathbf{82.8}$</td>
<td>81.8</td>
</tr>
</tbody>
</table>

Table 9: Impact of pruning strategies in $\text{FULL} \rightarrow \text{LIMITED}$ regime. The results are NDCG scores averaged on different transfer settings.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>FFN-Global</th>
<th>FFN-Layerwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head-Global</td>
<td>55.4</td>
<td>58.7</td>
</tr>
<tr>
<td>Head-Layerwise</td>
<td>$\mathbf{64.7}$</td>
<td>63.8</td>
</tr>
</tbody>
</table>

Table 10: Impact of pruning strategies in $\text{LIMITED} \rightarrow \text{LIMITED}$ regime. The results are NDCG scores averaged on different transfer settings.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>FFN-Global</th>
<th>FFN-Layerwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head-Global</td>
<td>61.1</td>
<td>61.3</td>
</tr>
<tr>
<td>Head-Layerwise</td>
<td>$\mathbf{62.4}$</td>
<td>61.9</td>
</tr>
</tbody>
</table>

Table 11: Ablation results if we remove the sparsity-inducing regularizer during connectivity pattern searching. We report the average results of different settings.

<table>
<thead>
<tr>
<th>Method</th>
<th>FULL → FULL</th>
<th>FULL → LIMITED</th>
<th>LIMITED → LIMITED</th>
</tr>
</thead>
<tbody>
<tr>
<td>EARLY-EMB w/o Regularizer</td>
<td>82.8</td>
<td>64.7</td>
<td>62.4</td>
</tr>
<tr>
<td></td>
<td>72.3</td>
<td>58.2</td>
<td>52.5</td>
</tr>
</tbody>
</table>

generating high-quality task embeddings.

**J Further Storage-efficiency with Single Layer**

In this study, we examine the performance of CO-PATE when utilizing a single layer to generate task embeddings. The results, as illustrated in Figure 10, demonstrate the performance of each layer. The findings indicate that a single layer can yield performance comparable to that of the full model. Specifically, when the fifth layer is used to generate the embedding, there is a significant reduction of 91.7% in the storage space required for the embedding, while the final NDCG score is only slightly lower, at 0.67 on average, as compared to the full model.
Figure 10: The impact of using one single transformer layer for embedding generation. The NDCG is an average of different transfer settings.
ACL 2023 Responsible NLP Checklist

A For every submission:

✔ A1. Did you describe the limitations of your work?
   The Limitations section is after the conclusion part.

✘ A2. Did you discuss any potential risks of your work?
   We think there is no ethical statements need to be included.

✔ A3. Do the abstract and introduction summarize the paper’s main claims?
   The abstract is at the beginning of the article and the introduction is Section 1.

✘ A4. Have you used AI writing assistants when working on this paper?
   Left blank.

B ✔ Did you use or create scientific artifacts?
   Section 5.1 (Experimental Setups), Appendix A

✔ B1. Did you cite the creators of artifacts you used?
   Section 4.1 (Experimental Setup), Appendix A

✘ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
   They are all open-source artifacts that are publicly available.

✘ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
   They are all open-source artifacts that are publicly available.

✘ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
   They are all open-source artifacts that are publicly available, and do not contain this kind of private information.

✔ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
   Section 4.1 (Experimental Setup), Appendix A

✔ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
   Section 4.1 (Experimental Setup), Appendix A

C ✔ Did you run computational experiments?
   Section 4 (Predicting Task Transferability), Section 5 (Discussion)

✔ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
   Section 4.1 (Experimental Setup), Section 5.2 (Computation and Storage Consumption) and Appendix F

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Yes, Section 4.1 (Experimental Setup) and Appendix F

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4 (Predicting Task Transferability), Section 5 (Discussion) and Appendix F

C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Yes, Section 4.1 (Experimental Setup) and Appendix F

D. Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?
No response.

D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.