Task Addition in Multi-Task Learning by Geometrical Alignment

Soorin Yim [1] **Dae-Woong Jeong** [1] **Sung Moon Ko** [1] **Sumin Lee** [1] **Hyunseung Kim** [1] **Chanhui Lee** [1] **Sehui Han** [1]

Abstract

Training deep learning models on limited data while maintaining generalization is one of the fundamental challenges in molecular property prediction. One effective solution is transferring knowledge extracted from abundant datasets to those with scarce data. Recently, a novel algorithm called Geometrically Aligned Transfer Encoder (*GATE*) has been introduced, which uses soft parameter sharing by aligning the geometrical shapes of task-specific latent spaces. However, *GATE* faces limitations in scaling to multiple tasks due to computational costs. In this study, we propose a task addition approach for *GATE* to improve performance on target tasks with limited data while minimizing computational complexity. It is achieved through supervised multi-task pre-training on a large dataset, followed by the addition and training of task-specific modules for each target task. Our experiments demonstrate the superior performance of the task addition strategy for *GATE* over conventional multi-task methods, with comparable computational costs.

1. Introduction

Molecular property prediction is a key area of computational chemistry, aimed at developing models that map molecular structures to their properties (Deng et al., 2023). These properties can vary from fundamental characteristics such as electron affinity and critical compressibility factor to complex and specific properties like chromophore lifetime. These predictions are crucial for accelerating the development of next-generation materials, optimizing chemical synthesis, and understanding molecular interactions.

However, accurately predicting molecular properties poses a significant challenge due to the complex relationships between molecular structures and their properties. Transfer learning has emerged as an essential technique to address the scarcity of labeled data and the high dimensionality of feature spaces in this field (Shen & Nicolaou, 2019). By leveraging knowledge gained from related tasks or domains, transfer learning enables models to generalize better to target tasks with limited training data, enhancing prediction accuracy and robustness. This is particularly beneficial in molecular property prediction, where experimental data collection is often expensive and time-consuming.

In the context of molecular property prediction, Geometrically Aligned Transfer Encoder (*GATE*) has been introduced as a promising foundation approach for knowledge transfer (Ko et al., 2024b;a). *GATE* aligns the geometrical shapes of latent spaces across tasks to transfer mutual information. However, *GATE*'s pairwise transfer between tasks results in $O(N^2)$ computational complexity as the number of tasks increases, making it computationally expensive.

To address this issue, we propose a task addition approach for *GATE*, a two-stage framework that reduces the computational cost while maintaining effective knowledge transfer. In the first stage, supervised pre-training is performed on a large dataset. In the second stage, modules for each target task are added and trained on smaller datasets of the target task while the parameters pre-trained on source tasks remain fixed. This approach allows pre-trained models to be reused for diverse target tasks, significantly reducing training time.

In this paper, we introduce task addition for *GATE* to minimize computational cost while maintaining generalizability. The key contributions of this work are as follows:

- We extend *GATE* with a task addition approach.
- Task-added *GATE* outperforms single-task learning (SINGLE) and task-added multi-task learning (MTL) across various molecular property prediction tasks.
- The training time for task-added *GATE* is significantly faster than training MTL models from scratch and comparable to SINGLE and task-added MTL.
- Our results show that task-added *GATE* is less dependent on the choice of source tasks, unlike task-added MTL, which heavily depends on source tasks.

[1]LG AI Research, Seoul, Republic of Korea. Correspondence to: Sehui Han <hansse.han@lgresearch.ai>.

Accepted at AI for Science Workshop at 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.
Copyright 2024 by the author(s).

2. Task Addition by Geometrical Alignment

Geometrical alignment is an effective method in a multi-task prediction setups. However, as shown in (Ko et al., 2024a), the algorithm requires significant computational power as the number of tasks increases. To address this, we hereby introduce a specific mathematical description of the task addition method for *GATE* to accelerate the algorithm with minimal loss of prediction power.

In the *GATE* algorithm, the core assumption is that the essence of prediction performance lies in the geometrical characteristics of the corresponding latent space of a given input. In multi-task learning, different downstream tasks induce various latent spaces, yet the input remains equivalent for molecular property prediction tasks. Therefore, the primary strategy is to align the geometrical shapes of latent spaces from different prediction tasks to maximize the utilization of mutual information.

The fundamental architecture of *GATE* is precisely described in (Ko et al., 2024b), including the algorithm diagram, thorough mathematical structure, experiments, and an introduction to Riemannian differential geometry. The architecture of the task-added *GATE* is depicted in Figure 1. As shown in the figure, a molecule is initially represented using the Simplified Molecular-Input Line-Entry system (SMILES), which serves as the input. It is then embedded into a real-number embedding vector. This embedding process is mandatory since the *GATE* algorithm requires infinitesimal perturbations around the given input vector. Consequently, the universal embedding vector must be capable of acquiring perturbation points independent of the choice of tasks.

The embedding vectors are then fed into the task-specific encoders to generate task-specific latent vectors, z_n. For this process, we utilize Directed Message Passing Neural Network (DMPNN) (Yang et al., 2019) and conventional Multi-Layer Perceptron (MLP) layers. These latent vectors serve two main purposes: one is to compute prediction values for tasks, and the other is to align the latent spaces.

To ensure model's accuracy, we first introduce a simple regression loss. After the latent vectors pass through the task-specific head network, the final predicted value should closely match to the given label. We use a simple Mean Squared Error (MSE) loss for this purpose.

$$l_{\text{reg}} = \frac{1}{K}\sum_{i}^{K}(y_i - \hat{y}_i)^2 \tag{1}$$

Here, K, y_i, and $\hat{y}_i$ represent the number of target tasks to add, the target label, and the predicted value, respectively. In the multi-task extended *GATE* architecture, the regression loss is summed over the entire number of tasks, resulting in the computation complexity of the algorithm being $\mathcal{O}(N^2)$. However, as the number of tasks is now restricted to the number of target tasks, the entire process becomes significantly faster.

The other, and the most important part of the algorithm, is the alignment of latent spaces. To align the geometrical shapes of these latent spaces, one should know the specific mapping relation between one another. In the mathematical description, the mapping should be formulated by coordinate transformation induced by a Jacobian at an arbitrary point.

$$z'^i \equiv \sum_j \frac{\partial z'^i}{\partial z^j} z^j \tag{2}$$

Deriving an analytic form of the Jacobian from data-driven methods without assuming the underlying geometry is typically impossible. We bypass this issue by predicting the transformed vector directly using a neural network. Neural networks are generally smooth and differentiable due to the backpropagation learning scheme, allowing us to assume the latent space is also smooth and differentiable. Hence, it is plausible to assume the latent space as Riemannian. The diffeomorphism invariance of Riemannian geometry ensures that a locally flat frame can be found anywhere on a manifold, which we utilize to align latent geometries.

We set up autoencoder models to map a vector in latent space to a locally flat frame on a universal manifold and vice versa, as depicted in Figure 2. Specifically, each encoder, ϕ maps a latent vector from a task-specific latent space to a vector in a locally flat frame, while each decoder, ϕ^{-1}, maps a vector from the locally flat frame to the task-specific latent space. Unlike the original *GATE*, since this situation involves task addition, the models for source tasks should be pre-trained. Therefore, the mapping models can be categorized into two types: those with fixed parameters and those with learnable parameters. Mapping from source tasks should be fixed.

$$z'_\alpha = \text{Transfer}_{\alpha\to LF}(z_\alpha) \tag{3}$$

$$\hat{z}_\alpha = \text{Transfer}^{-1}_{LF\to\alpha}(z'_\alpha) \tag{4}$$

And mapping from the target task should be learnable.

$$z'_t = \text{Transfer}_{t\to LF}(z_t) \tag{5}$$

$$\hat{z}_t = \text{Transfer}^{-1}_{LF\to t}(z'_t) \tag{6}$$

Here, α denotes the source task, and t indicates the target task. The index α ranges from 1 to the number of source tasks. For example, if there are 1 to 10 different source tasks and one new target task is added, then α ranges from 1 to 10.

Considering $\text{Transfer}_{\alpha\to LF}(z_\alpha)$ as an example, where α is set to 4, this indicates that the latent vector of the 4th task is mapped to a locally flat frame on a universal manifold.

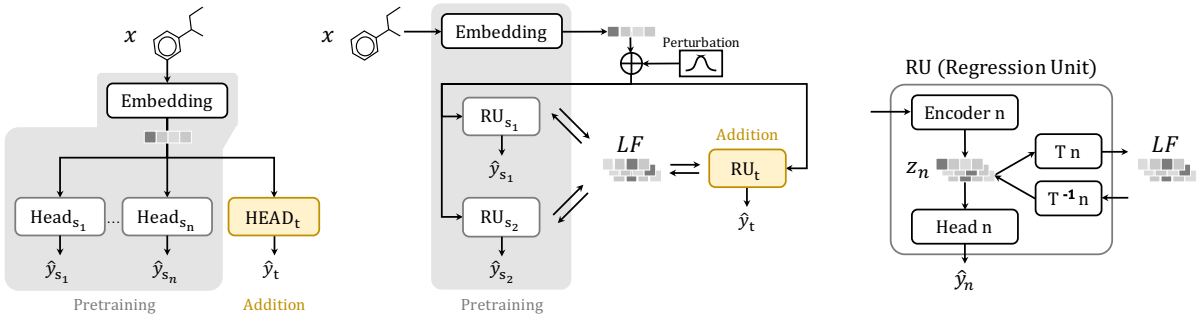

Figure 1. Schematic diagram of task addition in multi-task learning, where a model is pre-trained on two source tasks and one target task is added. (left) In conventional multi-task learning, each task shares a common latent vector and uses a task-specific head for making predictions. During pre-training, embedding and heads for each source task are trained. Subsequently, heads for target tasks are added and trained with target data while modules trained in the pre-training stage are kept frozen. (middle) Task addition for *GATE* algorithm, which comprises embedding and task-specific modules called regression units (RU). During pre-training, embedding and RUs for source tasks are trained. Then, RUs for target tasks are added and trained with target data, while modules trained in the pre-training stage are kept frozen. (right) In *GATE*, the regression unit for task n consists of four modules: encoder, transfer (T n), inverse transfer (T^{-1}n), and head. In *GATE*, latent vectors for heads are not directly shared; instead, they are transferred to a universal locally flat (LF) space, enabling knowledge transfer through the alignment of geometrical shapes of source and target latent spaces.

One can introduce a simple autoencoder loss that forces the input latent vector and reconstructed vector to be equal for the mapping networks.

$$l_{\text{auto}} = \sum_{\alpha} \text{MSE}(z_\alpha, \hat{z}_\alpha) \tag{7}$$

Additionally, a different set of losses can be formulated to align the geometrical shapes of latent spaces. So far, we have not imposed any constraints on a mapping network that should map a latent vector to a locally flat frame. As introduced in (Ko et al., 2024b), we will introduce three different kinds of constraints afterward.

One loss involves matching latent vectors on the locally flat frame that are mapped from different task-specific latent spaces. This loss aims to align the geometry point-wise by requiring latent vectors on a locally flat frame to be equal to one another. This loss is simply induced by the fact that the input of the model always starts from the same molecule regardless of the task choice. Hence, while the task-specific latent vectors may differ, if the mapping encoder correctly maps the vectors to the locally flat frame on the universal manifold, the mapped vectors from different tasks should be equal on the locally flat frame. We call this loss the consistency loss.

$$l_{cons} = \sum_{\alpha} \text{MSE}(z'_\alpha, z'_t) \tag{8}$$

As mentioned earlier, z'_α and z'_t represent vectors on the locally flat frame from the source and target latent spaces, respectively. One can introduce another loss function, namely the mapping loss, to enhance the bonding between latent spaces. This loss requires the predicted value for a given task using a standard downstream route to be equal to the predicted value for the same task using a detoured route.

$$z'_\alpha = \text{Transfer}_{\alpha \to LF}(z_\alpha) \tag{9}$$

$$\hat{z}_{\alpha \to t} = \text{Transfer}^{-1}{}_{LF \to t}(z'_\alpha) \tag{10}$$

Here, $\hat{z}_{\alpha \to t}$ indicates a latent vector of the target task mapped from the vector in the source task latent space by the source mapping and target inverse-mapping networks. Although $\hat{z}_{\alpha \to t}$ originates from the source task latent space, the source mapping network should map the vector to a locally flat frame, and the target inverse-mapping network should theoretically map the vector to the same latent vector in the target latent space. Therefore, if z_t and $\hat{z}_{\alpha \to t}$ go through the same head network, the predicted value should be the same. This loss is called the mapping loss.

$$l_{map} = \sum_{\alpha} \text{MSE}(y_t, \hat{y}_{\alpha \to t}) \tag{11}$$

y_t represents a predicted value from a standard downstream route, and $\hat{y}_{\alpha \to t}$ indicates a predicted value from a detoured route.

Despite these losses, geometrical alignment remains insufficient as they only align geometry at specific points. In general, neural networks often enjoy a vast number of degrees of freedom, providing enough flexibility to significantly distort the geometrical shape. Hence, even though the geometry is matched at given points, the surrounding shape around those points may still not be aligned.

To impose stronger constraints, one should consider not only a specific point but also the geometrical shape around a given input point. Typically, this is done by assuming

an analytic form of the metric and computing Riemannian properties such as curvatures. However, finding the analytic form of the metric is nearly impossible. Therefore, we will impose geometrical constraints without knowing the analytic form of the metric.

The key notion is the distance between points on a curved space. While a distance can be easily defined in Euclidean space, it is not as straightforward in curved space. One should solve the geodesic (freely falling motion) equation to find the geodesic path between points and integrate the infinitesimal distances along the path with metric weighting.

$$S^2 = \int_l \sum_\mu \sum_\nu g_{\mu\nu} dx^\mu dx^\nu \tag{12}$$

However, we now know that one can always find a locally flat frame on a Riemannian manifold. By utilizing this fact, if one is trying to compute the distance between a point on a locally flat frame and a its infinitesimal perturbations, the distance is simplified into a mere Euclidean distance.

$$\begin{aligned} S^2 &= \textstyle\int_l \sum_\mu \sum_\nu g_{\mu\nu} dx^\mu dx^\nu \\ &= \textstyle\int_l \sum_\mu \sum_\nu \eta_{\mu\nu} dx^\mu dx^\nu \\ &= \textstyle\int_a^b dx^2 \end{aligned} \tag{13}$$

Here, a represents a given latent vector, while b denotes a perturbed vector around the latent vector a. As we assume the perturbation is infinitesimal, the distance between a given point and its perturbations can be reduced to the following form:

$$S = |b - a| \tag{14}$$

This distance should also be the same for both source and target tasks after mapping into a locally flat frame. The loss that ensures this equality is called the distance loss.

$$l_{dis} = \frac{1}{M} \sum_\alpha C_\alpha \sum_i^M \mathrm{MSE}(s_\alpha^i, s_t^i) \tag{15}$$

Here, M is the number of perturbation points around the input vector, C_α is the weight of distance losses according to the source tasks, and s_s^i is the displacement between given data and their perturbations. Specific descriptions of loss terms are as follows.

$$s_\alpha^i \equiv |(z'_\alpha) - (z'^i_\alpha)| \qquad s_t^i \equiv |(z'_t) - (z'^i_t)| \tag{16}$$

$$z'^i_\alpha = \mathrm{Transfer}_{\alpha \to LF}(\mathrm{Encoder}_\alpha(x^i)) \tag{17}$$

$$z'^i_t = \mathrm{Transfer}_{t \to LF}(\mathrm{Encoder}_t(x^i)) \tag{18}$$

The index i denotes ith perturbation point.

The distance loss is the key term in this architecture. This loss restricts the local geometrical shape around input vector points, significantly reducing the vast number of degrees of freedom of the model. As mentioned earlier, the latent space should be smooth and differentiable, limiting the freedom to retain the shape of local geometry around input vectors and preventing drastic deformations[1].

Gathering all the introduced loss terms gives rise to a total loss function for the architecture.

$$l_{tot} = l_{reg} + \alpha l_{auto} + \beta l_{cons} + \gamma l_{map} + \delta l_{dis} \tag{19}$$

The total loss contains numerous hyperparameters that need to be tuned. These parameters affect the prediction performance of the model. Hence, some of the hyperparameters should be tuned with care to achieve superior performance. For instance, parameters γ, δ, and C_α are sensitive to performance, while others are not. Therefore, we often leave the other parameters as 1 and focus on finding the best set of parameters for γ, δ, and C_α. If the relationship between tasks is well known by domain knowledge, it is a very good strategy to begin with.

3. Experiments

3.1. Experimental Setup

3.1.1. Datasets

To evaluate our algorithm, we used 20 datasets, each corresponding to a different molecular property, sourced from three open databases: PubChem (Kim et al., 2022), Ochem (Sushko et al., 2011), and CCCB (III, 2022). Each property was standardized by subtracting the mean and dividing by the standard deviation. Detailed information about each dataset is provided in Appendix A. We split each dataset into training and test sets using an 80:20 ratio. For evaluating model performance for extrapolation, we split the dataset based on the scaffold (Bemis & Murcko, 1996). The training set was further uniformly split into four folds for cross-validation.

3.1.2. Models

We employed ten tasks as source tasks for the supervised pre-training of *GATE* and MTL. The remaining ten tasks were used as target tasks for task addition, individually. For each target task, we added the regression unit for the target task to the pre-trained model, resulting in ten different models for *GATE*, each for a target task.

To benchmark the performance of *GATE*, we also trained MTL models in task addition setup. In the pre-training stage, a pre-trained MTL model with a shared encoder and task-specific heads for source tasks were trained. Then, for each target task, task-specific head was added to the model architecture and trained on the target task while the pre-trained parameters were frozen. We also trained ten

[1]For further information, check experiments and ablation section in (Ko et al., 2024b)

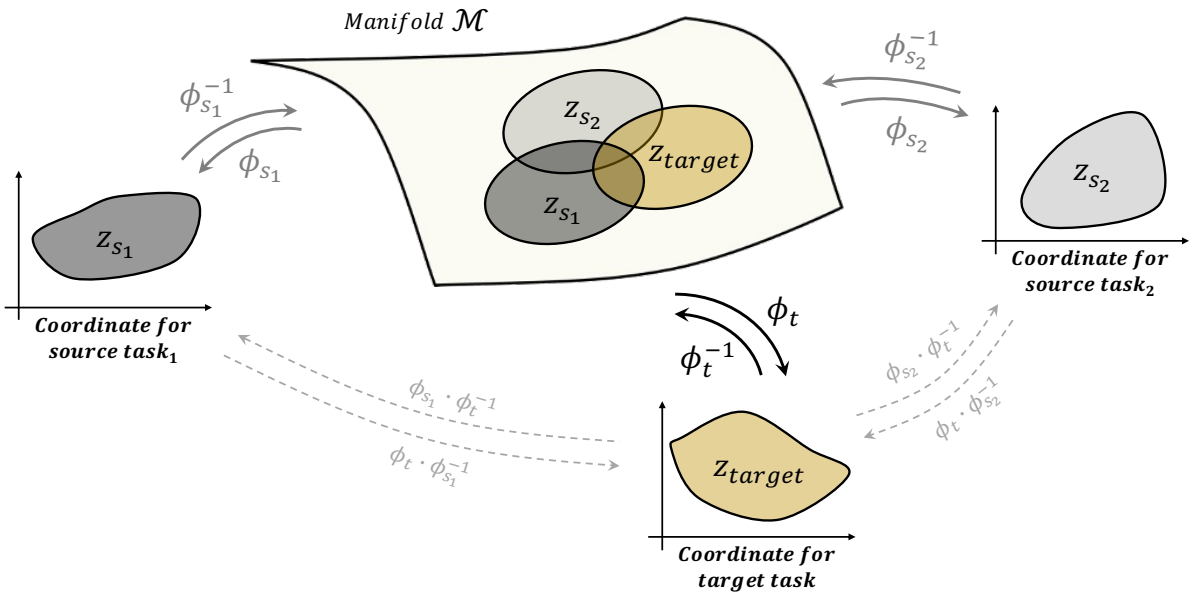

Figure 2. Conceptual depiction of the *GATE* algorithm for task addition where the model is pre-trained on two source tasks and one target task is added. Knowledge from source tasks is transferred to a target task by aligning the geometry of the target task to the geometries of source tasks. This alignment is achieved by finding a transfer function, ϕ, which maps an arbitrary point from a task-specific coordinate to a universal manifold, M. One can transform an arbitrary point in the overlapping region from one task coordinate to another by composing transfer functions. By matching the overlapping points on a manifold, one can align the inherent geometry of target data to the geometry of source data. This allows the information to flow from the source to target tasks. Grey-scaled regions are trained in the pre-training stage and frozen in the addition stage.

task-added MTL models, each for a target task.

For performance comparison, we trained *GATE* and MTL on all 20 tasks from scratch (referred to as 'Vanilla20' models), serving as reference performance for task-added *GATE* and MTL. To compare computational costs, we trained 10 MTL and *GATE* models from scratch (referred to as 'Vanilla11'), trained on 11 tasks (10 source tasks + one target task).

In summary, we trained the following three different types of models for both *GATE* and MTL:

- Task-added model: Pre-trained on 10 source tasks followed by addition and training of one target task.
- Vanilla20: Trained on all 20 tasks from scratch.
- Vanilla11: Trained on 10 source tasks and one target task from scratch.

Lastly, we also trained single-task models ('SINGLE') on each target task without multi-task learning. The same architecture for encoders and heads was used across all models for fair comparison. A detailed descriptions of the model architecture and hyperparameters are provided in Appendix Table 3 and 4, respectively.

3.2. Results

3.2.1. Task Addition Reduces Computational Costs

The primary motivation for task addition is to reduce the computational complexity of *GATE*. We analyzed training times for SINGLE, MTL, and *GATE*, as shown in Table 1. Task-added MTL was the fastest to train, even taking less time than SINGLE. Task-added *GATE* was 13 % slower than SINGLE, but 9.29 times faster than vanilla11 MTL. Task-added MTL was 10.9 times faster than vanilla11 MTL, while task-added *GATE* was 39.13 times faster than vanilla11 *GATE* . The significant improvement in the training speed of *GATE* with task addition is due to the time complexity of vanilla *GATE* being $\mathcal{O}(N^2)$, compared to MTL's $\mathcal{O}(N)$. In summary, task-added *GATE* is significantly faster than training vanilla MTL or *GATE* from scratch, and its training time is comparable to that of SINGLE.

3.2.2. Task-Added *GATE* Enables Knowledge Transfer

The other goal of task addition is to maintain model performance compared to training the entire model from scratch. The performance of SINGLE, task-added MTL, and task-added *GATE* is illustrated in Figure 3. The average Root

Table 1. Training times for SINGLE, MTL, and *GATE* algorithm.

MODEL	TIME PER EPOCH[S]
SINGLE	5.40
TASK-ADDED MTL	5.12
TASK-ADDED *GATE*	6.09
VANILLA11 MTL	56.00
VANILLA11 *GATE*	238.32

Mean Squared Error (RMSE) of task-added MTL, SINGLE, and task-added *GATE* across 10 target tasks are 0.742, 0.670, and 0.647, respectively. Similarly, the average Pearson correlation of task-added MTL, SINGLE, and task-added *GATE* across 10 target tasks are 0.527, 0.662, and 0.688, respectively. Detailed numerical results can be found in Appendix Table 5. These metrics demonstrate that *GATE* outperforms SINGLE, indicating that knowledge transfer through task addition enhances performance.

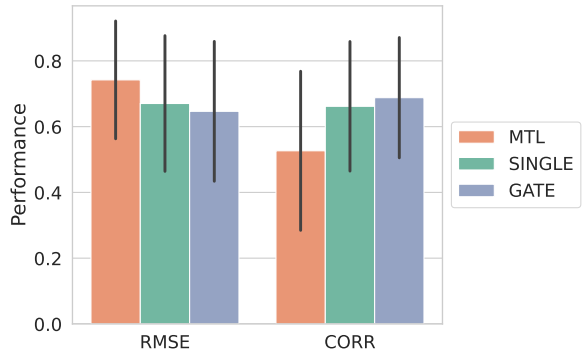

Figure 3. The performance of task-added MTL, SINGLE, and task-added *GATE* algorithm. Average RMSE and Pearson correlation values are displayed across 10 target tasks, with error bars indicating standard deviation. Detailed performance values can be found in Appendix Table 5.

In contrast, MTL underperforms compared to SINGLE, suggesting that not all MTL architectures are suitable for task addition. This observation aligns with findings reported in previous studies (van de Ven et al., 2022; Deng et al., 2023). Overall, these results indicate that *GATE* generalizes well in an extrapolation setting by using task-specific latent vector for each task, minimizing interference that might be caused by strongly shared latent vector in MTL.

This effect is further highlighted by comparing the correlation recovery rates of *GATE* and MTL in Figure 4. On average, task-added *GATE* recovered 98.3 % of the Pearson correlation of vanilla20 *GATE*, while task-added MTL achieved 70.4 % of the Pearson correlation of vanilla20 MTL. *GATE* shows higher correlation recovery rates than MTL for all ten tasks. In the worst-case scenario, task-added *GATE* achieves a recovery rate of 79.3 % for CMQ, whereas task-added MTL only recovers 15.1 % for CML. This highlights that task-added *GATE* shows robust performance regardless of the target task, whereas the performance of task-added MTL heavily depends on the nature of the target task. In summary, these results demonstrate that *GATE* effectively transfers knowledge from source tasks, while MTL shows limited performance in task addition setup.

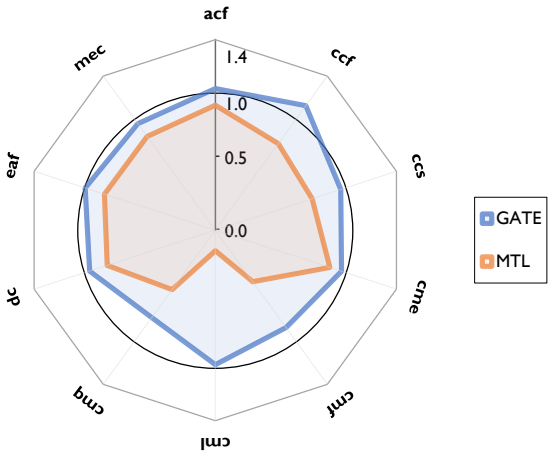

Figure 4. Correlation recovery rate of MTL and *GATE* algorithm. The correlation of task-added *GATE* and MTL is divided by the correlation of vanilla20 *GATE* and MTL, respectively. The complete names of the abbreviated tasks are listed in Appendix Table 2.

3.2.3. DEPENDENCE ON SOURCE TASKS

To analyze the factors that affect the performance of task addition, we examined the correlation between source tasks and target tasks. Specifically, for each source-target task pair, we selected molecules that have labels for both source and target tasks. If a source and target task share at least ten molecules in common, we calculated the absolute Pearson correlation between their labels to measure the degree of relatedness between source and target tasks. Figure 5 reports the maximum absolute correlation of source tasks for each target task.

For MTL, the correlation recovery rate tends to decrease as the maximum correlation between the target task and the source tasks decreases. This indicates that if no source task is closely related to the target task, the performance of multi-task learning (MTL) can decrease. On the other hand, the correlation recovery rate of task-added *GATE* is less dependent on the correlation with source tasks. This explains why task-added MTL performs worst for CML,

which has a maximum related source task correlation of 0.387. Conversely, the correlation recovery rate of *GATE* for CML is 0.988, achieving a 7.3 % increase compared to SINGLE. These results demonstrate the ability of task-added *GATE* to effectively extract mutual information even in challenging situations.

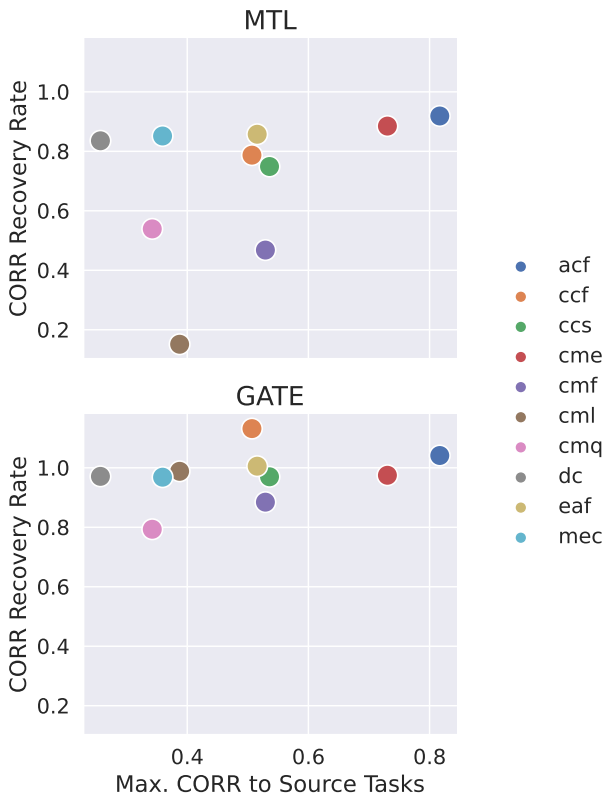

Figure 5. The relationship between correlation to source tasks and task addition performance. The performance of task-added MTL decreases as the maximum absolute correlation to the source tasks decreases, whereas the performance of task-added *GATE* is less dependent on the correlation to the source tasks. The complete names of the abbreviated tasks are provided in Appendix Table 2.

Our experimental results demonstrate that task-added *GATE* maintains the performance of vanilla *GATE* trained from scratch and shows robust performance across various target tasks. which is not achievable in conventional MTL architecture. Additionally, task-added *GATE* achieves training times comparable to SINGLE and significantly faster than training vanilla *GATE* and MTL from scratch. Collectively, these findings suggest that task addition is an effective approach for reducing computational costs without sacrificing performance for *GATE*.

4. Discussion

In this study, we proposed and evaluated a two-stage, multi-task learning approach called task addition for the *GATE* algorithm. Task addition aims to enhance performance on target tasks with limited data while minimizing computational complexity. This is achieved through leveraging supervised multi-task pre-training on a large dataset, followed by the addition of task-specific modules. Task-added *GATE* demonstrated significant performance improvements over SINGLE and task-added MTL, with more efficient training times compared to vanilla MTL and *GATE* trained from scratch.

The *GATE*'s unique approach to knowledge transfer via geometrical alignment enhances the model's performance. Unlike conventional multi-task learning (MTL), which shares latent vectors directly, *GATE* transfers knowledge through a universal locally flat (LF) space, aligning the geometrical shapes of latent spaces across tasks. This method mitigates negative transfer effects and ensures efficient information flow from source to target tasks.

In the context of molecular property prediction, where data scarcity and high dimensionality are significant challenges, the *GATE* algorithm offers a robust solution. Our findings indicate that task-added *GATE* can maintain the performance of its vanilla counterpart while achieving training times comparable to SINGLE and task-added MTL models. This suggests that *GATE* is particularly well-suited for the efficient screening of novel compounds with desired properties by adding related tasks to the model.

While our results are promising, there are areas for further investigation. We used supervised multi-task learning as a pre-training strategy. Recently, self-supervised learning has been widely adopted as a pre-training strategy, and recent studies have showed that self-supervised pre-training followed by multi-task fine-tuning can improve performance (Xu et al., 2024). Adopting such strategies to *GATE* may further enhance the model's performance.

In summary, the proposed task addition approach for *GATE* offers a powerful tool for improving performance on target tasks with limited data in molecular property prediction. By effectively balancing computational efficiency and task performance, the *GATE* algorithm stands out as a promising candidate for addressing the challenges of data scarcity in this domain.

References

Bemis, G. W. and Murcko, M. A. The properties of known drugs. 1. molecular frameworks. *Journal of medicinal chemistry*, 39(15):2887–2893, 1996.

Deng, J., Yang, Z., Wang, H., Ojima, I., Samaras, D.,

and Wang, F. A systematic study of key elements underlying molecular property prediction. *Nature Communications*, 14:6395, 2023. ISSN 2041-1723. doi: 10.1038/s41467-023-41948-6. URL https://doi.org/10.1038/s41467-023-41948-6.

III, R. D. J. Nist computational chemistry comparison and benchmark database. *NIST Standard Reference Database*, 101, 2022. URL http://cccbdb.nist.gov/.

Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B. A., Thiessen, P. A., Yu, B., Zaslavsky, L., Zhang, J., and Bolton, E. E. PubChem 2023 update. *Nucleic Acids Research*, 51(D1):D1373–D1380, 10 2022. ISSN 0305-1048. doi: 10.1093/nar/gkac956. URL https://doi.org/10.1093/nar/gkac956.

Ko, S. M., Lee, S., Jeong, D.-W., Kim, H., Lee, C., Yim, S., and Han, S. Multitask extension of geometrically aligned transfer encoder, 2024a.

Ko, S. M., Lee, S., Jeong, D.-W., Lim, W., and Han, S. Geometrically aligned transfer encoder for inductive transfer in regression tasks. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=3z60EWfh1p.

Shen, J. and Nicolaou, C. A. Molecular property prediction: recent trends in the era of artificial intelligence. *Drug Discovery Today: Technologies*, 32-33:29–36, 2019. ISSN 1740-6749. doi: https://doi.org/10.1016/j.ddtec.2020.05.001. URL https://www.sciencedirect.com/science/article/pii/S1740674920300032. Artificial Intelligence.

Sushko, I., Novotarskyi, S., Körner, R., Pandey, A. K., Rupp, M., Teetz, W., Brandmaier, S., Abdelaziz, A., Prokopenko, V. V., Tanchuk, V. Y., et al. Online chemical modeling environment (ochem): web platform for data storage, model development and publishing of chemical information. *Journal of computer-aided molecular design*, 25:533–554, 2011.

van de Ven, G. M., Tuytelaars, T., and Tolias, A. S. Three types of incremental learning. *Nature Machine Intelligence*, 4:1185–1197, 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00568-3. URL https://doi.org/10.1038/s42256-022-00568-3.

Xu, Z., Shi, Z., Wei, J., Mu, F., Li, Y., and Liang, Y. Towards few-shot adaptation of foundation models via multitask finetuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=1jbh2e0b2K.

Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., Palmer, A., Settels, V., Jaakkola, T., Jensen, K., and Barzilay, R. Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 59, 07 2019. doi: 10.1021/acs.jcim.9b00237.

A. Detailed Explanation of Datasets

We evaluated task addition using 20 molecular property datasets. During preprocessing, we excluded data with incorrect units, typographical errors, and measurements taken under extreme conditions. Before training, we standardized all datasets by their mean and standard deviation. Below, we provide the physical meaning of each molecular property.

- **Acentric Factor (ACF)** : A measure of the non-sphericity of molecules, which quantifies how much the behavior of a fluid deviates from that of a spherical molecule
- **Absorbance Maximum Wavelength (AW)** : The wavelength at which a substance absorbs the maximum amount of light
- **Boiling Point (BP)** : The temperature at which a compound changes state from liquid to gas at a given atmospheric pressure.
- **Critical Compressibility Factor (CCF)** : A dimensionless number that describes the behavior of a substance at its critical point.
- **Collision Cross Section (CCS)** : A measure of the probability of interaction between particles, representing the effective area that one particle presents to another for a collision to occur.
- **Chromophore Emission Max (CME)** : The wavelength at which a chromophore emits the maximum amount of light upon excitation.
- **Chromophore Emission FWHM (CMF)** : The width of the emission spectrum at half of its maximum intensity, indicating the range of wavelengths emitted by the chromophore.
- **Chromophore Life Time (CML)** : The average time a chromophore remains in an excited state before returning to its ground state.
- **Chromophore Quantum Yield (CMQ)** : The efficiency of photon emission by a chromophore, defined as the ratio of the number of photons emitted to the number of photons absorbed.
- **Decomposition (DC)** : The process by which a chemical compound breaks down into simpler substances or elements.
- **Dipole Moment (DM)** : A measure of the separation of positive and negative charges in a molecule, indicating the polarity of the molecule.
- **Electron Affinity (EAF)** : The amount of energy released when an electron is added to a neutral atom or molecule in the gas phase to form a negative ion.
- **Flash Point (FP)** : The lowest temperature at which a liquid can form an ignitable mixture in air near its surface, indicating its flammability.
- **HOMO (HM)** : The highest energy molecular orbital that is occupied by electrons in a molecule under normal conditions.
- **LUMO (LM)** : The lowest energy molecular orbital that is unoccupied by electrons, which can accept electrons during a chemical reaction.
- **Log P (LP)** : The logarithm of the partition coefficient between octanol and water, indicating the hydrophobicity or lipophilicity of a compound.
- **Molar Extinction Coefficient (MEC)** : A measure of how strongly a chemical species absorbs light at a given wavelength per molar concentration.
- **Melting Point (MP)** : The temperature at which a solid turns into a liquid at atmospheric pressure.
- **pKa (PKA)** : The negative logarithm of the acid dissociation constant, indicating the strength of an acid in solution.
- **Refractive Index (RI)** : A measure of how much light is bent, or refracted, when entering a material from another medium.

B. Architecture and Hyperparameters

Training process for *GATE* is described in (Ko et al., 2024b). Our model is composed of five different types of modules, whose parameter sizes are listed in Table 3.

Table 2. Dataset statistics

NAME	ACRONYM	TYPE	COUNT	MEAN	STD
ACENTRIC FACTOR	ACF	ADDITION	1850	-0.33	0.23
ABSORBANCE MAXIMUM WAVELENGTH	AW	PRETRAINING	11896	440.39	67.80
BOILING POINT	BP	PRETRAINING	8044	183.65	96.24
CRITICAL COMPRESSIBILITY FACTOR	CCF	ADDITION	1357	0.25	0.03
COLLISION CROSS SECTION	CCS	ADDITION	4006	205.06	57.84
CHROMOPHORE EMISSION MAX	CME	ADDITION	6407	504.95	100.38
CHROMOPHORE EMISSION FWHM	CMF	ADDITION	2862	72.14	26.04
CHROMOPHORE LIFE TIME	CML	ADDITION	2738	0.49	0.64
CHROMOPHORE QUANTUM YEILD	CMQ	ADDITION	5609	0.35	0.29
DECOMPOSITION	DC	ADDITION	12998	210.86	58.31
DIPOLE MOMENT	DM	PRETRAINING	11224	0.30	0.53
ELECTRON AFFINITY	EAF	ADDITION	198	15.24	28.36
FLASH POINT	FP	PRETRAINING	9409	114.21	82.87
HOMO	HM	PRETRAINING	97262	-5.66	0.64
LUMO	LM	PRETRAINING	97262	-1.63	0.89
LOG P	LP	PRETRAINING	31264	10.33	9.79
MOLAR EXTINCTION COEFFICIENT	MEC	ADDITION	16324	7.67	0.64
MELTING POINT	MP	PRETRAINING	22901	376.53	92.31
PKA	PKA	PRETRAINING	9514	6.62	3.11
REFRACTIVE INDEX	RI	PRETRAINING	11143	1.49	0.10

Table 3. Network parameters

NETWORK	LAYER	INPUT, OUTPUT SIZE	HIDDEN SIZE	DROPOUT
BACKBONE	DMPNN	[134,149], 100	200	0
BOTTLENECK	MLP LAYER	100, 50	50	0
TRANSFER	MLP LAYER	50, 50	50	0.2
INVERSE TRANSFER	MLP LAYER	50, 50	50	0.2
HEAD	MLP LAYER	50, 1	-	0.2

The hyperparameters for training are described in Table 4.

Table 4. Hyperparameters

HYPERPARAMETER	VALUE
LEARNING RATE	0.0005
OPTIMIZER	ADAMW
BATCH SIZE	512
EPOCH	1000
# OF PERTURBATION	5
$\alpha, \beta, \gamma, \delta$	1, 1, 1, 1

C. Experimental Results

Here, we describe the explicit test results in this section. The numerical results for Figure 3 is reported in Table 5.

The correlation recovery rate depicted in Figure 4 are listed in Table 6.

Table 5. The performance of SINGLE, task-added MTL, and task-added *GATE*algorithm

	RMSE		CORRELATION	
	MEAN	STD	MEAN	STD
TASK-ADDED MTL	0.742	0.179	0.527	0.242
SINGLE	0.670	0.206	0.662	0.197
TASK-ADDED *GATE*	0.647	0.213	0.688	0.183

Table 6. The correlation recovery rate of MTL, and *GATE* algorithm for each target task.

TASK	MTL	GATE
ACF	0.919	1.041
CCF	0.787	1.132
CCS	0.749	0.970
CME	0.885	0.975
CMF	0.468	0.884
CML	0.151	0.988
CMQ	0.539	0.793
DC	0.835	0.971
EAF	0.857	1.005
MEC	0.852	0.968