Multi-level model acceleration of drug-target interaction prediction

JiaCheng Zhu,[†] JingKun Hu,[†] KunRan Zhang,[‡] Yong Mu,[‡] TingTing Li,[‡] AnQi Wu,[¶] JianSheng Wu,[*,§] and Bing-Kun Bao[*,‖]

[†]*School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*

[‡]*School of Telecommunication and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*

[¶]*The Second Affiliated Hospital of Nanjing University of Chinese Medicine, Nanjing 210017, China*

[§]*School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China; Alibaba Group, Beijing 100102, China*

[‖]*School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China; The State Key Laboratory of Tibetan Intelligence, Xining, 810016, China*

E-mail: jansen@njupt.edu.cn; bingkunbao@njupt.edu.cn

Abstract

The fast and accurate computational prediction of drug-target interactions (DTIs) is crucial for accelerating the drug discovery process. While significant research has been conducted in this field, current methods predominantly rely on large models to represent the features of drugs and targets, followed by predictions using advanced machine learning techniques such as contrastive learning. However, as the drug molecule space and target space continue to expand rapidly, predicting DTIs for real-world drug discovery faces substantial computational challenges, making the acceleration of DTI prediction models a critical issue. To date, no solutions have been proposed to address this challenge. In this paper, we propose a novel multi-level model acceleration approach, Fast-VSTM. It first optimizes the ConPLex model, then applies a low-rank adaptive large model fine-tuning approach for efficient optimization, and finally combines multi-head self-attention relation distillation and feature distillation to lightweight the model and accelerate the DTI prediction process.Experimental results show that Fast-VSTM achieves an average of 1.66x inference speedup on multiple standard DTI datasets while maintaining comparable accuracy.Fast-VSTM is free available at `https://github.com/JiachengSigma/Fast-VSTM`.

Introduction

The identification of drug-target interactions (DTI) is a crucial step in drug discovery. Initially, researchers used biological experiments to detect the interactions between drugs and targets, but this approach was fraught with significant issues, including high costs and low accuracy.[1]

The traditional drug discovery process is time-consuming, often requiring several years to progress from initial screening to clinical trials. Accelerating DTI helps reduce resource waste on inactive or low-efficiency compounds, allowing researchers to focus on candidates that show higher potential. By speeding up DTI, researchers can more quickly identify potentially effective compounds and their corresponding targets, significantly shortening the drug discovery timeline. DTI research is not only a core component of drug discovery and development but also plays a vital role in personalized medicine, drug safety assessment, drug metabolism, pharmacokinetics, and other fields.[2] The interaction strength between a drug molecule and its target protein is vital for drug design and repurposing. Generally, a higher affinity suggests improved treatment outcomes, enhanced target specificity, prolonged binding duration, and a slower development of drug resistance.Despite the rapid development of DTI prediction modeling in recent years, its accuracy is still insufficient. Prediction modeling of DTI has become an indispensable task for effective therapeutic interventions.Accurate modeling of DTI can accelerate the drug discovery process and avoid associated costs.

Currently, mainstream methods for DTI prediction cover feature-based methods, deep learning methods, ensemble methods, and special methods[3][4].[5] Feature-based methods employ features like chemical fingerprints that extracted from data to predict how drugs interact with their targets through machine learning models. They are adept at handling large datasets but can be computationally intensive and sensitive to the graph's construction. Deep learning methods employ deep neural networks to automatically learn high-level features from raw data, providing powerful capabilities yet requiring substantial labeled data and computational resources, and they are prone to overfitting. Ensemble methods combine multiple models to improve accuracy and robustness through averaging or voting, enhancing performance but increasing computational burden and complexity in strategy design. Special methods include innovative approaches like text mining and multimodal learning, opening new research directions. Nowadays, deep learning has become the mainstream approach for DTI prediction. It mainly includes sequence - based model methods such as EmdedDTI,[6] graph neural network methods such as DeepDrug,[7] deep - learning - based generative model methods such as GANsDTA.[8]

In the field of virtual drug screening, the application of the Transformer[9] based on molecular representation learning has further improved the accuracy of drug screening. However, it also limits the speed of drug screening and requires more computational resources. Therefore, we propose a Fast Virtual Screening Acceleration Algorithm for Transformer Models (Fast-VSTM). Before introducing Fast-VSTM, we first introduce VSTM. VSTM is an improved method based on ConPlex[10] proposed in this paper. In VSTM,the cyclic molecular fingerprint[11] model of Conplex, which is used as the feature extraction network for drug molecules, is replaced with the molecular 3D pre-trained model Uni-Mol[12] based on Transformer. In order to make the feature extraction network better adapt to specific tasks and improve the performance and effectiveness in virtual screening tasks, based on VSTM, Fast-VSTM adopts a parameter-efficient fine-tuning strategy based on adaptive budget allocation. While keeping the original pre-trained parameters unchanged, by constructing an incremental matrix based on singular value decomposition, only a small number of parameters are fine-tuned for specific tasks to accelerate the model fine-tuning process. The aim is to save computational resources and enable the model to adapt to new tasks more quickly. Next, to further accelerate our model, the fine-tuned and accelerated model is used as a teacher model. Through the multi-head self-attention relationship transfer method of the Transformer model, the knowledge distillation method[13][14] is used to lightweight the feature extraction network and the feature mapping network respectively, realizing the knowledge transfer from a large model to a small model, reducing the model parameters and computational complexity, and further accelerating the model inference time, thus speeding up the process of virtual drug screening. In addition, Fast-VSTM adopts the Synthetic Minority Oversampling Technique (SMOTE)[15] and adds a residual structure to the feature mapping network of VSTM to address the problem of imbalance between positive and negative samples in some data during the virtual screening process.

Experimental results demonstrate that after fine-tuning, the prediction accuracy on four datasets, namely BIOSNAP,[16] BindingDB,[17] DAVIS,[18] and COVID-19,[19] outperforms that of the baseline model.[10] Nevertheless, the speed remains slower than that of the baseline model ConPlex. The AdaLoRA[20] is further utilized to optimize the parameter efficiency of the model, rendering it more cost-effective and efficient in parameter utilization. Even when the number of parameters is substantially decreased, the Fine-tuned VSTM can maintain high performance while accelerating model inference. To further expedite the fine-tuned VSTM, we subsequently employ the Multi-Head Self-Attention Relation Knowledge Distillation (MSRKD) method[13].[14] This method successfully reduces the parameters of the feature extraction network of the fine-tuned VSTM by 46%. Experiments conducted on four standard drug-target interaction (DTI) datasets, including BIOSNAP, BindingDB, DAVIS, and COVID-19, indicate that, compared with the fine-tuned VSTM, Fast-VSTM achieves an average 1.66-fold acceleration in inference while maintaining comparable accuracy. Ultimately, when compared with the baseline model ConPlex, for Fast-VSTM, on BIOSNAP and BindingDB, with a 2% reduction in AUPR (Area Under the Precision-Recall Curve), the acceleration effects reach 1.31-fold and 1.13-fold respectively. For the DAVIS dataset, the AUPR increases by 5.6%, and the acceleration effect attains 1.57-fold. These experiments illustrate that

Fast-VSTM exhibits certain reliability and practicality in drug screening.

Methods

Overall Architecture

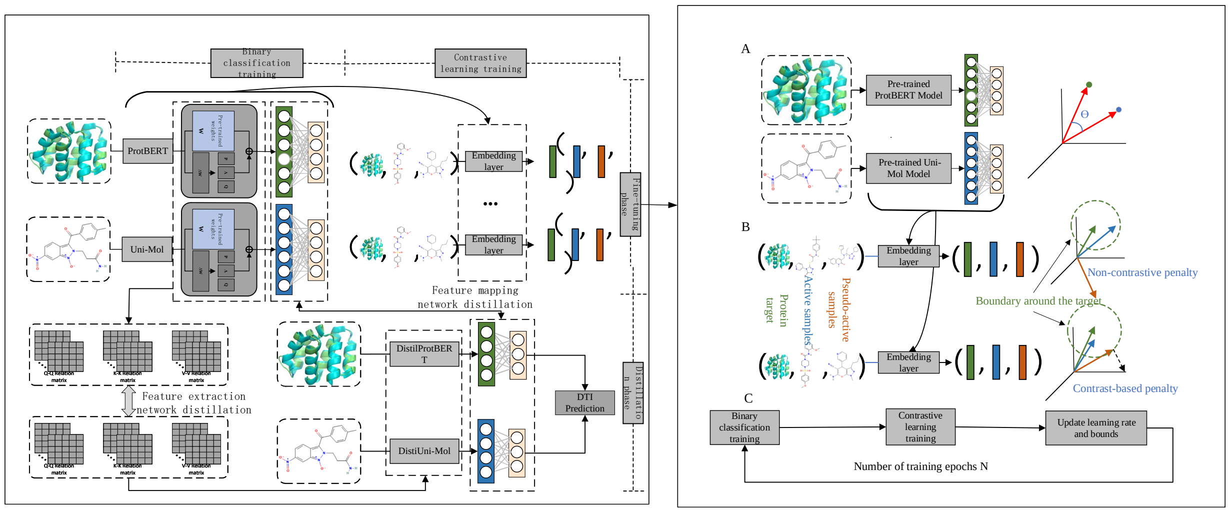
Binary classification training
Contrastive learning training
ProtBERT
Uni-Mol
Pre-trained weights
Embedding layer
Feature mapping network distillation
Fine-tuning phase
Feature extraction network distillation
DistilProtBERT
DistiUni-Mol
DTI Prediction
Distillation phase
A
Pre-trained ProtBERT Model
Pre-trained Uni-Mol Model
B
Protein target
Active samples
Pseudo-active samples
Non-contrastive penalty
Boundary around the target
Contrast-based penalty
C
Binary classification training
Contrastive learning training
Update learning rate and bounds
Number of training epochs N

Figure 1: The overall pipeline of Fast-VSTM.The left side is the overall workflow of Fast-VSTM, and the right side is the detailed framework diagram of VSTM. The Fast-VSTM framework diagram on the left primarily includes a fine-tuning acceleration module and a distillation acceleration module for VSTM.

This Figure 1 shows the overall architecture of Fast-VSTM, which mainly includes the architecture of VSTM and the acceleration of VSTM. The acceleration of VSTM includes the fine-tuning stage and the knowledge distillation stage of the fine-tuned model. The acceleration in the fine-tuning stage of VSTM mainly involves singular value decomposition and importance rank perception to prune the model. Subsequently, for the fine-tuned model, we use the knowledge distillation technique to generate smaller-sized DistilProTBERT and DistilUni-Mol, so as to improve computational efficiency without sacrificing performance.Finally, these trained models are used to predict the drug-target interactions. New proteins and compounds are received as inputs to test the performance of the models. In general, Fast-VSTM employs a variety of deep learning techniques and algorithms, aiming to efficiently and accurately predict drug-target interactions.

VSTM

VSTM represents an enhanced approach grounded in Conplex.[10] In the VSTM framework, the cyclic molecular fingerprint model within Conplex, which functions as the feature extraction network for drug molecules, is supplanted by Uni-Mol.[12] Uni-Mol is a molecular 3D pre-trained model leveraging the Transformer architecture.VSTM serves as the foundation of Fast-VSTM. Fast-VSTM is obtained by performing knowledge distillation on the feature extraction network and the feature mapping network of the fine-tuned VSTM.

VSTM principally comprises five modules: the target feature extraction module, the drug feature extraction module, the contrastive learning module, the training process, and the inference process. Subsequently, each of these modules will be explained with an appropriate level of detail, striking a balance between thoroughness and conciseness.

The VSTM's protein feature extraction network is based on the self-supervised protein language model ProtBERT[21] , while the drug molecule feature extraction network utilizes the universal 3D molecular representation learning framework Uni-Mol.[12]

ProtBERT uses amino acids as tokens, treating entire protein sequences as sentences. During the self-supervised training process, it first predicts masked tokens from known sequences in a self-supervised manner, using only unannotated protein sequences as input for this step. Following this, the embeddings learned by ProtBERT are extracted and applied as input to supervised training tasks at the residue level or protein level, such as secondary structure prediction, transmembrane protein identification, and subcellular localization.

The VSTM's drug molecule feature extraction network opts for the framework Uni-Mol,[12] which generates molecular 3D structures from one-dimensional SMILES sequences[22] through the Uni-Mol. By leveraging representation learning based on three-dimensional information, it acquires molecular features that contain richer structural information. The molecular encoding process in the Uni-Mol is illustrated in Figure 2. The backbone network of the Uni-Mol is based on the Transformer, taking atom types and atomic coordinates as inputs. Atom types are initialized through an embedding layer to generate atom representations, while atom pair representations are derived from atomic coordinates as invariant spatial position encodings.

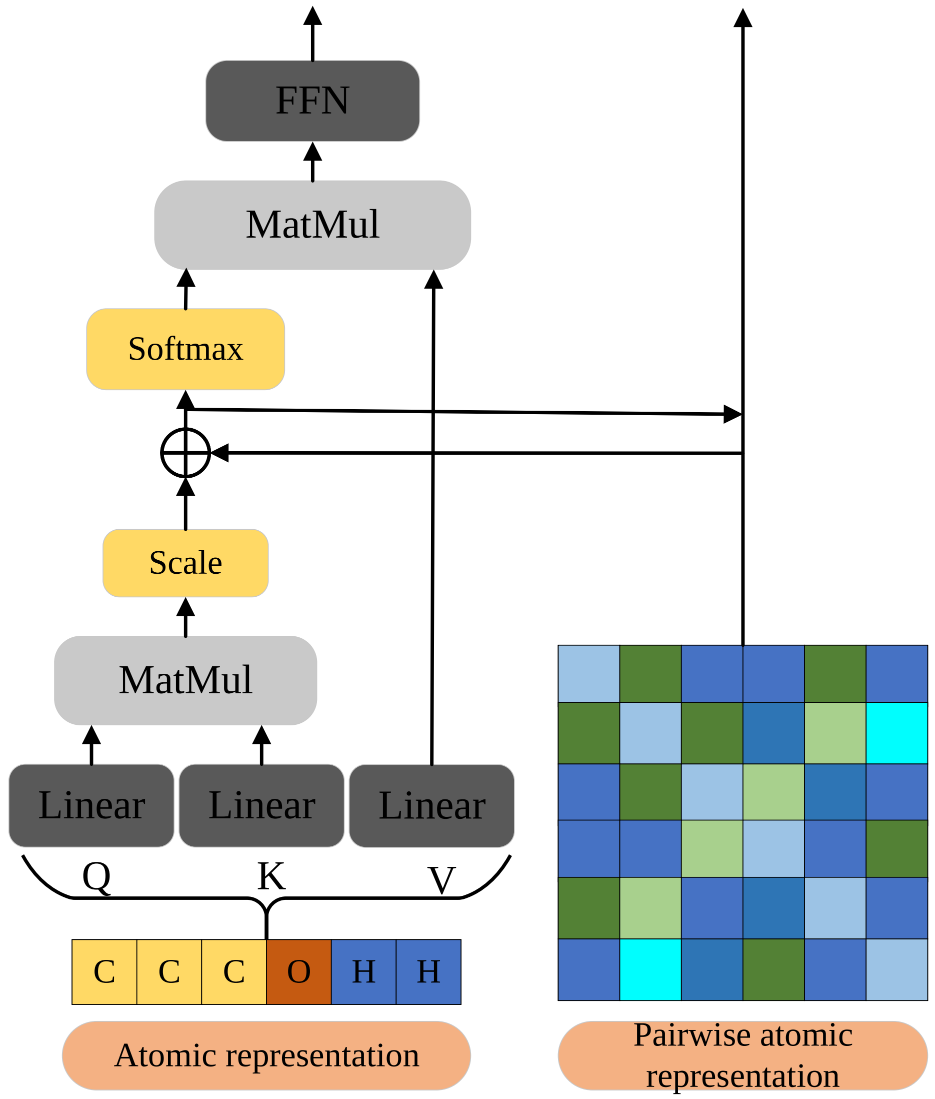

Figure 2: Schematic for Molecular Encoding of Uni-Mol[12]

Unlike the discrete position encodings used in natural language processing and computer vision, the position coordinates in 3D space are continuous values. To ensure invariance to global rotations and translations during the position encoding process, Uni-Mol utilizes the Euclidean distances between all atom pairs within a molecule, combined with learnable atom pair types based on atomic representations, which are then smoothed by a Gaussian kernel function[23] to obtain the position encodings. Typically, a Transformer only maintains atom-level representations. However, since the spatial position information of molecules is encoded at the atom-pair level, Uni-Mol also maintains atom-pair representations to better learn the 3D representation of molecules. Subsequently, to update the atom-pair representations, Uni-Mol facilitates communication from atoms to atom pairs through the product of multiple heads' Query-Key in the self-attention mechanism.

Formally, the update of atom-pair representations can be expressed by the Equation. 1.

$$q_{ij}^{l+1} = q_{ij}^{l} + \left\{ \frac{Q_i^{l,h} \left(K_j^{l,h}\right)^T}{\sqrt{d}} \middle| h \in [1, H] \right\} \tag{1}$$

In the context of the given Equation. 1, q_{ij}^{l+1} represents the updated representation of the atom pair (i, j) at layer $l + 1$, where H is the number of attention heads, and d is the dimension of the hidden representation. $Q_i^{l,h}$ and $K_j^{l,h}$ denote the query for the i-th atom and the key for the j-th atom in head h at layer l, respectively. To leverage the 3D information from the atomic representations, communication from pairs to atoms is introduced by using the pair representations as a bias term in the self-attention mechanism.

To incorporate the 3D information, the pair-to-atom communication modifies the self-attention as follows:

$$\text{Attention}(Q_i, K_j, V_k) = \text{softmax}\left(\frac{Q_i K_j^T + B_{ij}}{\sqrt{d}}\right) V_k \tag{2}$$

In the context of the given Equation. 2, B_{ij} is the bias term derived from the pair representation, which allows the model to consider the spatial relationship between atoms when computing the attention scores.

By utilizing a feature extraction network based on the Transformer architecture for molecular representations, it is possible to extract features of proteins and drug molecules. This approach enables global context modeling of sequence data, thereby better capturing dependencies and semantic information between different parts of the sequences. Consequently, this leads to the extraction of richer and more accurate features.

Next, I'll introduce the contrastive learning module and the training process. In the contrastive learning module, the triplets of targets, drugs, and decoys are transformed into the shared space in the same way. The model learns representations by minimizing the similarity between different transformed views of the target and the drug, while it aims to maximize the similarity between the target and the decoy. The parameters are updated using the triplet distance loss on the high-coverage dataset to minimize the target-drug distance and maximize the target-decoy distance simultaneously. If the target-decoy distance is greater than the target-drug distance plus a certain threshold, no additional penalty is imposed. VSTM is trained alternately in the binary classification extreme and the contrastive learning phase during training to optimize two objectives simultaneously. After each round of training, the learning rate and the contrastive threshold are updated according to an annealing schedule.

During the inference phase, VSTM is similar to ConPlex.[10] Although the model has been used to predict the probability of interactions and perform binary classification, VSTM can also predict the binding affinity based on the interaction probability. Moreover, the model has achieved better results than ConPlex in the binary classification stage.

Acceleration by Fine-tuning Strategies

To better adapt the feature extraction network to specific tasks and improve performance and effectiveness in virtual screening tasks, we employ an Adaptive Low-Rank Adaptation (AdaLoRA)[20] for parameter-efficient fine-tuning of the feature extraction network. This method comprises two critical components: (1) SVD-based (Singular Value Decomposition) adaptation, which parameterizes incremental matrices using singular value decomposition; and (2) Importance-aware rank allocation, which prunes redundant singular values based on measures of importance.

SVD-based Adaptation

LoRA[24] fine-tuning introduces a new pathway alongside the original pretrained model when processing modules that involve matrix multiplication. This pathway operates through the product of two matrices A and B, where matrix A is responsible for dimensionality reduction, and matrix B is responsible for dimensionality expansion. The output of the pretrained model weights and the new pathway are added together.

$$W = W^{(0)} + \Delta = W^{(0)} + BA \tag{3}$$

where $W^{(0)}$ is the parameter of the pretrained model, and BA represents the weights of the new pathway. However, a significant limitation of LoRA is that all incremental matrices are fine-tuned using the same rank, but the importance of parameters in different depths and different modules at the same depth varies within the model. To address this issue, AdaLoRA parameterizes the incremental matrices through singular value decomposition, trims unimportant singular values, and performs adaptive budget allocation for different modules of the model, optimizing the model parameter configuration.

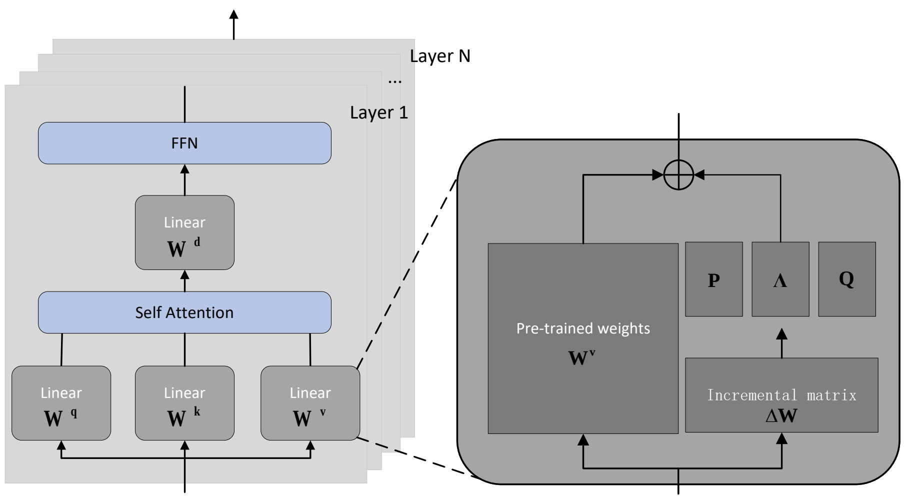

Figure 3: Fine-tuning based on singular value decomposition

As shown in Figure 3, AdaLoRA parameterizes the incremental matrix of the pretrained weight matrix using Singular Value Decomposition (SVD):

$$W = W^{(0)} + \Delta = W^{(0)} + P\Lambda Q \tag{4}$$

where $P \in \mathbb{R}^{d_1 \times r}$ and $Q \in \mathbb{R}^{r \times d_2}$ represent the left and right singular vectors of Δ, respectively, and the diagonal matrix $\Lambda \in \mathbb{R}^{r \times r}$ contains the singular values $\{\lambda_i\}$ for $1 \leq i \leq r$, where $r \ll \min(d_1, d_2)$. Further, $G_i = \{P_{*i}, \lambda_i, Q_{i*}\}$ is defined as a triplet containing the i-th singular value and vectors. Since Λ is a diagonal matrix, it can be stored as a vector in $\mathbb{R}^r$ to save memory.

Λ is initialized to zero, and P and Q are initialized with random Gaussian values to ensure $\Delta = 0$ at the start of training. To ensure the orthogonality of P and Q, i.e., $P^T P = I$ and $Q^T Q = I$, the following regularization term is used:

$$R(P, Q) = \|P^T P - I\|_F^2 + \|Q^T Q - I\|_F^2 \tag{5}$$

For the i-th singular value λ_i and the i-th left/right singular vectors P_{*i} and Q_{i*} in formula (4.14), the triplet is represented as $G_i = \{P_{*i}, \lambda_i, Q_{i*}\}$. For a complete model, assume it has n triplets that need to be computed, where the k-th triplet is represented as $G_{k,i} = \{P_{k,*i}, \lambda_{k,i}, Q_{k,i*}\}$.

Importance-Aware Rank Allocation

In deep learning model pruning algorithms, the sensitivity of a single parameter is defined as the absolute value of the product of the gradient and the weight magnitude:

$$I(w_{ij}) = \left| w_{ij} \nabla_{w_{ij}} \mathcal{L} \right| \tag{6}$$

where w_{ij} is an arbitrary trainable weight in the model, and $\nabla_{w_{ij}} \mathcal{L}$ is the gradient with respect to the weight. $I(w_{ij})$ essentially approximates the change in loss when a parameter is zeroed out. If deleting a parameter would have a significant impact, the model is sensitive to that parameter and it should be retained. During the stochastic gradient descent process, the sensitivity score is estimated on sampled mini-batches. The sensitivity can be smoothed using a moving average to reduce the evaluation error caused by individual mini-batch samples, expressed as:

$$\overline{I}^{(t)}(w_{ij}) = \beta_1 \overline{I}^{(t-1)}(w_{ij}) + (1-\beta_1) I^{(t)}(w_{ij}) \tag{7}$$

where t represents the training step, and $0 < \beta_1 < 1$ is a hyperparameter that controls the balance between historical records and the current batch in the moving average method. Yet, parameter sensitivity, on its own, doesn't make for a dependable metric of importance. To more comprehensively gauge the significance of parameters, we calculate the uncertainty of parameter sensitivity as follows:

$$U^{(t)}(w_{ij}) = \left| I^{(t)}(w_{ij}) - \overline{I}^{(t-1)}(w_{ij}) \right| \tag{8}$$

and a moving average is also performed to obtain $\overline{U}^{(t)}(w_{ij})$:

$$\overline{U}^{(t)}(w_{ij}) = \beta_2 \overline{U}^{(t-1)}(w_{ij}) + (1-\beta_2) U^{(t)}(w_{ij}) \tag{9}$$

where $0 < \beta_2 < 1$. The importance of a parameter $s^{(t)}(w_{ij})$ is represented by the product of the sensitivity $\overline{I}^{(t)}(w_{ij})$ and the uncertainty $\overline{U}^{(t)}(w_{ij})$:

$$s^{(t)}(w_{ij}) = \overline{I}^{(t)}(w_{ij}) \cdot \overline{U}^{(t)}(w_{ij}) \tag{10}$$

for the triplet $G_{k,i}$, considering the singular values and vectors in the triplet, the following parameter importance measure formula is obtained:

$$S_{k,i} = s(\lambda_{k,i}) + \frac{1}{d_1} \sum_{j=1}^{d_1} s(P_{k,j}) + \frac{1}{d_2} \sum_{j=1}^{d_2} s(Q_{k,j}) \tag{11}$$

To compute the rank r based on parameter importance, r is treated as a parameter of the model, and then updated based on the model's loss value. First, we need to define the model's loss function. For any model composed of n triplets, the loss function $\mathcal{L}(P, E, Q)$ can be expressed as the weighted sum of training loss and model regularization:

$$\mathcal{L}(P, E, Q) = C(P, E, Q) + \gamma \sum_{k=1}^{n} R(P_k, Q_k) \tag{12}$$

where $P = \{P_k\}_{k=1}^{n}$, $E = \{\Lambda_k\}_{k=1}^{n}$, $Q = \{Q_k\}_{k=1}^{n}$ represent the sets of feature parameters in the SVD formula, $\gamma > 0$ is the regularization coefficient, and $C(P, E, Q)$ is the training cost. During the t-th step of the training process, the parameters $P_k^{(t)}$, $\Lambda_k^{(t)}$, and $Q_k^{(t)}$ are updated using stochastic gradient descent, $k = 1, \ldots, n$. For $\Lambda_k^{(t)}$, the update differs from traditional stochastic gradient descent. First, compute the updated value $\overline{\Lambda}_k^{(t)}$:

$$\overline{\Lambda}_k^{(t)} = \Lambda_k^{(t)} - \eta \nabla_{\Lambda_k} \mathcal{L}(P^{(t)}, E^{(t)}, Q^{(t)}) \tag{13}$$

Given a learning rate $\eta > 0$, the singular values are pruned based on the updated value $\overline{\Lambda}_k^{(t)}$ and the importance score $S_k^{(t)}$:

$$\text{with } T\left(\overline{\Lambda}_k^{(t)}, S_k^{(t)}\right)_{ii} = \begin{cases} \overline{\Lambda}_k^{(t)} & \text{if } \overline{\Lambda}_k^{(t)} \text{ is in the top-} b^{(t)} \text{ of } S_k^{(t)} \end{cases}$$

where $S^{(t)} = \{S_{k,i}^{(t)}\}_{1 \le k \le n, 1 \le i \le r}$ contains the importance scores of all triplets, and $b^{(t)}$ is the budget of remaining singular values at step t. The idea of AdaLoRA is that at the beginning of the model, it is poorly adapted to the training objective. Larger ranks can be used to allow the model to converge quickly. Next, the ranks are pruned based on parameter importance, assigning different ranks to each triplet. Since rank pruning will inevitably degrade model performance, several rounds of rank adaptation are performed on the remaining ranks to stabilize the model's performance. The complete computation of the budget $b^{(t)}$ is as follows:

$$b^{(t)} = \begin{cases} b^{(0)}, & 0 \le t < t_i, \\ b^{(\tau)} + \left(b^{(0)} - b^{(\tau)}\right) & t_i \le t < T - t_f, \\ \quad \times \left(1 - \dfrac{t - t_i - t_f}{T - t_i - t_f}\right)^3, & \\ b^{(\tau)}, & \text{otherwise.} \end{cases} \tag{14}$$

At the initial stage of model fine-tuning, in the first t_i steps, give the budget $b^{(0)}$ a slightly larger value to allow the model to quickly reach a better performance. In the following $T - t_f - t_i$ steps, reduce the budget of the ranks at a cubic rate to achieve the purpose of rank pruning. In the remaining training, stabilize the rank size to achieve a local optimal solution for the model performance at the current rank. Through this process, by pruning unimportant singular values, more budget is left for the increment matrices with higher priority.

Acceleration by Knowledge Distillation

In the distillation stage, the model obtained from the fine-tuning stage serves as the teacher model. Knowledge distillation is performed on both the feature extraction network and the feature mapping network.Knowledge Distillation for the Feature Mapping Network is conducted by distilling the differences in output features from the feature mapping layers. The student model learns to approximate the output features of the teacher model, ensuring that it captures the essential patterns and representations.

The core concept of the Multi-Head Self-Attention Relation Knowledge Distillation (MSRKD) method[13][14] is to deeply mimic the self-attention modules of the teacher model, which are used to capture dependencies between tokens. In conventional multi-head self-attention relation knowledge distillation (CMRKD), the self-attention distributions from the teacher model are used to train the student model[25].[26] However, this approach limits the number of attention heads in the student model to match that of the teacher model. In this process, multi-head self-attention relationships are obtained through the scaled dot-product of multiple query (Q), key (K), and value (V) pairs. Taking the query (Q) vectors as an example, the queries from different attention heads are first concatenated together, and then the concatenated vector is split according to the number of desired relation heads, with similar operations performed for the key and value vectors. For teacher and student models that use different numbers of attention heads, their varying numbers of queries, keys, and values are transformed into an equal number of relation head vectors, to facilitate knowledge distillation training. This approach eliminates the restriction on the number of attention heads in the student model. Using a greater number of relation heads to compute self-attention relationships can acquire finer-grained self-attention knowledge and enhance the performance of the student model.

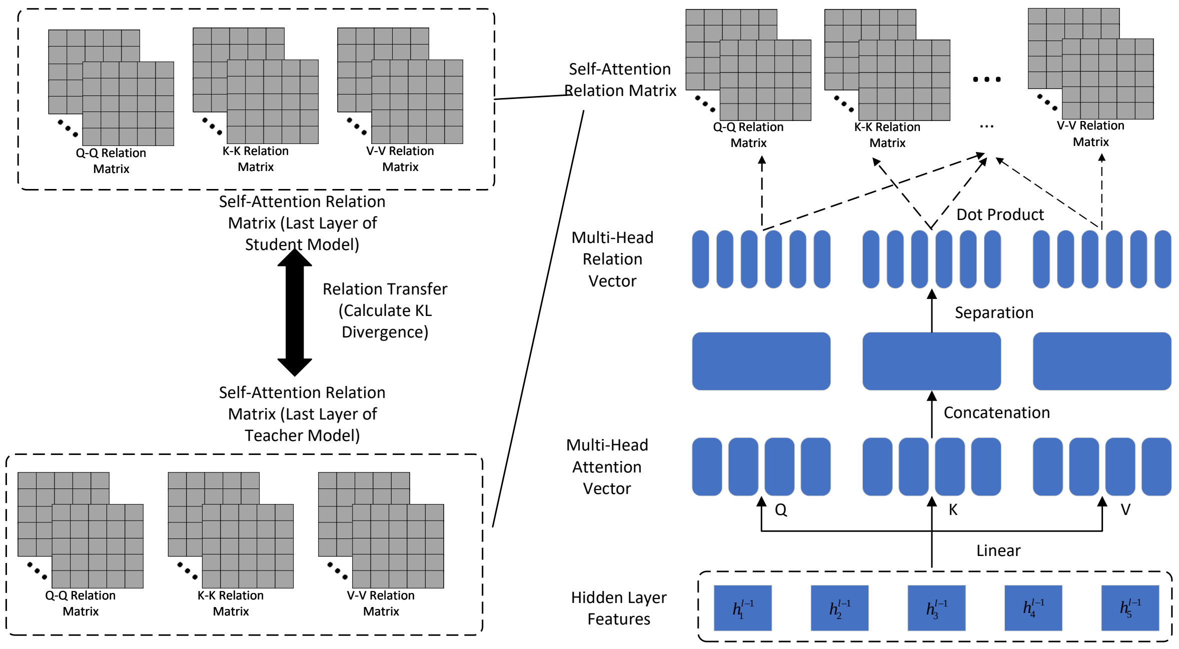

Figure 4: Framework diagram of knowledge distillation on feature extraction networks

In MSRKD (Multi-Head Self-Attention Relation Knowledge Distillation),[13] a distillation loss function is defined to quantify the Kullback-Leibler (KL) divergence between the multi-head self-attention relationships of the student model and the teacher model. The loss function represents the weighted sum of all possible self-attention relationship pairs, which is the average of the KL divergences across all relation heads for each pair of relationships. The specific form of the loss function is as follows:

$$\mathcal{L} = \sum_{i=1}^{3}\sum_{j=1}^{3} \alpha_{ij}\mathcal{L}_{ij} \tag{15}$$

$$\mathcal{L}_{ij} = \frac{1}{A_r}\sum_{a=1}^{A_r} D_{\mathrm{KL}}\left(R_{ij,l,a}^{T} \parallel R_{ij,m,a}^{S}\right) \tag{16}$$

Among these, the weights α_{ij} are a collection of hyperparameters, the values of which are used to assign weights to each self-attention relationship loss; $|x|$ represents the sequence length; $R_{ij,l,a}^{T}$ and $R_{ij,m,a}^{S} \in \mathbb{R}^{A_r\times|x|\times|x|}$ respectively denote the self-attention relationships of the teacher model and the student model, and the computation formula is:

$$R_{ij,l,a}^{T} = \mathrm{softmax}\left(\frac{A_{i,l,a}^{T}(A_{j,l,a}^{T})^{T}}{\sqrt{d_r}}\right) \tag{17}$$

$$R_{ij,m,a}^{S} = \mathrm{softmax}\left(\frac{A_{i,m,a}^{S}(A_{j,m,a}^{S})^{T}}{\sqrt{d_r}}\right) \tag{18}$$

Among these, $A_{i,l,a}^{T}, A_{j,l,a}^{T} \in \mathbb{R}^{|x|\times d_r}$ denote the query, key, or value attention vectors of the l-th layer's a-th relation head in the teacher model. Similarly, $A_{i,m,a}^{S}, A_{j,m,a}^{S} \in \mathbb{R}^{|x|\times d_r'}$ denote the query, key, or value attention vectors of the m-th layer's a-th relation head in the student model. d_r and d_r' respectively represent the relation head sizes of the teacher model and the student model.

During the knowledge distillation phase, the goal is to train the student model by optimizing the loss function $\mathcal{L}$ so that its self-attention relationships can closely approximate those of the teacher model, thereby mimicking the way the teacher model processes molecular sequence information. This method is particularly suitable for distilling knowledge from large Transformer models into smaller models for use in resource-constrained environments.

Fast-VSTM Algorithm

According to the above discussion, the virtual screening Transformer model acceleration algorithm *Fast-VSTM* is implemented as shown in Algorithm 1, which can be divided into 2 steps:

Step 1: Separately construct the attention matrix ΔW_T and ΔW_M in the protein feature extraction network *ProtBERT*[21] and the molecular feature extraction network *Uni-Mol*[12] through the self-attention mechanism. Obtain the matrices P_T, Λ_T, Q_T and P_M, Λ_M, Q_M through singular value decomposition of the attention matrix using the formula. In the binary classification and contrastive training process, freeze the original pre-trained weights W_T and W_M, extract a mini-batch from the dataset D, and use the formula to calculate the micro-loss function $\mathcal{L}_T(P_T, E_T, Q_T)$ and $\mathcal{L}_M(P_M, E_M, Q_M)$. The total micro-loss function can be expressed as:

$$\mathcal{L}_{FT}(P, E, Q) = \mathcal{L}_T(P_T, E_T, Q_T) + \mathcal{L}_M(P_M, E_M, Q_M) \tag{19}$$

Calculate the gradient $\nabla\mathcal{L}_{FT}(P, E, Q)$. Then, use formula (4.15) to calculate the sensitivity of each parameter of (P_T, E_T, Q_T) and (P_M, E_M, Q_M), denoted as $I^{(r)}$, based on formulas (4.16) and (4.17). Update the parameter sensitivity $I^{(r)}$ and uncertainty $U^{(r)}$ using formula (4.19). Calculate the parameter importance $s_k^{(r)}$, where $k = 1, ..., n$ and $l = 1, ..., r$; update $P_k^{(r+1)} = P_k^{(r)} - \eta\nabla P_k \mathcal{L}_{FT}(P, E, Q)$ and $Q_k^{(r+1)} = Q_k^{(r)} - \eta\nabla Q_k \mathcal{L}_{FT}(P, E, Q)$ using SGD; update $\Lambda_k^{(r+1)} = T\left(\Lambda_k^{(r)} - \eta\nabla\Lambda_k \mathcal{L}_{FT}(P, E, Q), s_k^{(r)}\right)$ using the given threshold $b^{(r)}$ and formula (4.22). Repeat the above operations until the model converges, and save the attention matrices ΔW_T and ΔW_M.

Step 2: Use the model obtained from fine-tuning as the teacher model, initialize the student model parameters, and sample a mini-batch from the dataset $\mathcal{D}$. According to equation (4.8), obtain the protein feature extraction network distillation loss $\mathcal{L}_{DT}$ and the molecular feature extraction network distillation loss $\mathcal{L}_{DM}$, map the teacher model feature extraction network output features T_t, $M_t \in \mathbb{R}^n$ with the student model feature extraction network output features T_s, $M_s \in \mathbb{R}^m$. Compute the distillation loss $\mathcal{L}_{DF}$:

$$\mathcal{L}_{DF} = \frac{1}{n}\sum_{i=1}^{n} (T_{t,i} - W_1 T_{s,i})^2 + (M_{t,i} - W_2 M_{s,i})^2 \tag{20}$$

where $T_{t,i}$, $M_{t,i}$ are the i-th elements of the teacher model feature extraction network output features T_t, M_t, and $T_{s,i}$, $M_{s,i}$ are the i-th elements of the student model feature extraction network output features T_s, M_s. $W_1, W_2 \in \mathbb{R}^n$ are trainable dimension transformation matrices. Compute the distillation loss $\mathcal{L}_D = \mathcal{L}_{DT} + \mathcal{L}_{DM} + \mathcal{L}_{DF}$, update the student model parameters using the distillation loss $\mathcal{L}_D$. Repeat the above operations multiple times until the model converges, and save the student model parameters.

Result and Discussion

Experimental Settings and Datasets

The acceleration method for the Virtual Screening Transformer model is implemented using the PyTorch framework. The software environment includes: CUDA version 11.6, Python version 3.9.11, and the operating system is Ubuntu 18.04.6 LTS. The hardware environment consists of a CPU with 12 cores, an Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz, and a GPU, specifically an NVIDIA GeForce RTX 3090 with 24GB of memory.

Table 1: DATASET STATISTICS

Dataset	Drug Molecules	Protein Targets	Training Set	Validation Set	Test Set
BIOSNAP	4,510	2,181	9,670 / 9,568	1,396 / 1,352	2,770 / 2,727
BindingDB	7,165	1,254	6,334 / 6,334	927 / 5,717	1,905 / 11,384
DAVIS	68	379	1,043 / 1,043	160 / 2,846	303 / 5,708
COVID-19	194,159	1	135,663 / 249	19,380 / 36	38,761 / 70

The experiments involve the datasets of BIOSNAP,[16] BindingDB,[17] DAVIS,[18] and the COVID-19[19] dataset. The statistical information of the datasets is shown in Table 1.

The BIOSNAP dataset uses the ChGMiner DTI data from the BIOSNAP collection, which consists of 4,510 drugs and 2,181 proteins, along with 13,741 DTI pairs from DrugBank. BindingDB contains the Kd values between 7,165 drugs and 1,254 proteins. The DAVIS dataset contains wet-lab measured Kd values for interactions between 68 drugs and 379 proteins. The DAVIS dataset represents a few-shot learning setting: compared to 12,668 training interactions in BindingDB and 19,238 in BIOSNAP, it only contains 2,086 training interactions. The COVID-19 dataset uses the

Algorithm 1 Virtual Screening Transformer Model Acceleration Algorithm Fast-VSTM

```
Require: Data set $D$, fine-tuning iteration times $T$, distillation iteration times $T_{KD}$, budget schedule $\{b^{(t)}\}_{t=0}^{T}$, hyperparameter $\eta$
Ensure: Student model parameters $\theta_S$
 1: Calculation steps:
 2: Fine-tuning stage
 3: Set $t = 0$;
 4: While $t < T$ do:
 5:     for training batch $B \in D$ do:
 6:         According to formula 16, calculate fine-tuning loss function $\mathcal{L}_T(P_T, E_T, Q_T)$ and $\mathcal{L}_M(P_M, E_M, Q_M)$;
 7:         $\mathcal{L}_{FT}(P, E, Q) = \mathcal{L}_T(P_T, E_T, Q_T) + \mathcal{L}_M(P_M, E_M, Q_M)$;
 8:         Calculate gradient $\nabla \mathcal{L}_{FT}(P, E, Q)$;
 9:         According to formula 6, calculate sensitivity $I^{(t)}$ for each parameter in $\{P, E, Q\}$;
10:         According to formulas 7 and 8, update $I^{(t)}$ and $U^{(t)}$ for each parameter in $\{P, E, Q\}$;
11:         According to formula 10, calculate $S_{k,l}^{(t)}$, $k = 1, ..., n$ and $l = 1, ..., r$;
12:         Update $P_k^{(t+1)} = P_k^{(t)} - \eta \nabla_{P_k} \mathcal{L}(P, E, Q)$ and $Q_k^{(t+1)} = Q_k^{(t)} - \eta \nabla_{Q_k} \mathcal{L}(P, E, Q)$;
13:         According to budget $b^{(t)}$, use formula (4.22) to update $A_k^{(t+1)} = T(A_k^{(t)} - \eta \nabla_{A_k} \mathcal{L}(P, E, Q), S_k^{(t)})$;
14:     End for
15:     $t = t + 1$;
16: End while
17: Distillation stage
18: Set $t = 0$;
19: While $t < T_{KD}$ do:
20:     for training batch $B \in D$ do:
21:         According to formula 3, calculate feature extraction network distillation loss $\mathcal{L}_{DT}$ and $\mathcal{L}_{DM}$;
22:         According to formula 20, calculate feature mapping network distillation loss $\mathcal{L}_{DF}$;
23:         Calculate distillation loss $\mathcal{L}_D = \mathcal{L}_{DT} + \mathcal{L}_{DM} + \mathcal{L}_{DF}$;
24:         Update student model parameters $\theta_S$ using stochastic gradient descent;
25:     End for
26:     $t = t + 1$;
27: End while
```

bioassay data from batches 1879 and 1706 of the PubChem database for COVID-19, with a total of 355 positive samples and 193,804 negative samples, serving as the test dataset for handling imbalanced positive and negative samples in the experiments.

In the BindingDB and DAVIS datasets, pairs are considered as positive drug-target interactions, while larger Kd values are considered as negative. The Kd (dissociation constant)[27] is an important biophysical parameter; when the Kd value is lower, it indicates stronger binding affinity between the drug molecule and its target, meaning that a lower concentration is needed to effectively bind to the target. A higher Kd value indicates weaker binding affinity, requiring a higher concentration to achieve effective binding.

Drug-target interaction prediction is essentially a binary classification task. To evaluate its performance, AUC and AUPR are used as performance evaluation metrics. For these two performance metrics, higher scores indicate that the predicted drug-target interactions (DTIs) are more consistent with the actual drug-target interactions, meaning that models with higher evaluation scores are more reliable in real-world drug screening. The datasets are split into training, validation, and test sets in a ratio of 7:1:2. For each experiment, five independent experiments are conducted, and the average value is taken. The best-performing model is selected based on the AUPR performance on the validation set, and the selected model is evaluated on the test set, with the experimental results recorded.

Performance of Drug-Target Interaction Prediction

To evaluate the proposed Virtual Screening Transformer Model (VSTM) in this paper, comparisons are made with the following four other methods for predicting drug-target interactions on the BIOSNAP, BindingDB, DAVIS, and COVID-19 datasets:

MolTrans[28] combines a knowledge-inspired substructure pattern mining algorithm and an interaction modeling module to improve the accuracy and interpretability of DTI prediction. It also uses an enhanced Transformer encoder, which can better capture the semantic relationships between substructures extracted from large-scale unlabeled biomedical data.

GNN-CPI[29] uses graph neural networks and one-dimensional convolutional neural networks to learn representations of molecules from molecular graphs and proteins from protein sequences, respectively. The learned molecule and protein representations are then concatenated to predict interactions between molecules and proteins.

DeepConv-DTI[30] is a CNN-based deep learning model for predicting drug-target interactions. It converts protein sequences into fixed-size embedding vectors. CNN captures local residue patterns and extracts key features through max pooling. The global feature representation formed by connecting the results of max pooling is used as input for subsequent models.

ConPlex[10] utilizes pre-trained protein language models and molecular circular fingerprints to extract features of proteins and drug molecules. Through contrastive learning, it better separates active samples from pseudo-active samples in the feature embedding space, making binding predictions based on the distance between the learned representations.

VSTM is the improved method proposed in this paper based on ConPlex, replacing the circular molecular fingerprint model used in ConPlex as the feature extraction network for drug molecules with a Transformer-based 3D pre-trained model Uni-Mol[12] for molecules.

All models initialize their weights via a normal distribution. The AdamW optimizer is employed to update the weights through error backpropagation, with a total of 50 training epochs. In binary classification tasks, the learning rate is initially set at 10^{-4} and adjusted based on a cosine annealing schedule that incorporates warm restarts every 10 epochs. For contrastive tasks, the initial learning rate is set to 10^{-5}, following the same annealing strategy. The margin value for contrastive loss is initially 0.25 and decreases to a minimum of 0 over 50 training epochs according to a tanh decay schedule, with restarts every 10 epochs. The feature - mapping network has a hidden dimension of 512, and the training batch size is 32.

Table 2 shows the performance scores of different methods on the BIOSNAP, BindingDB, DAVIS, and COVID-19 datasets. By comparing these methods, it can be observed that VSTM outperforms other methods across all four datasets. VSTM achieves superior performance by employing a more advanced molecular feature extraction network, which captures richer structural

Table 2: PERFORMANCE COMPARISON OF FINE-TUNED VSTM WITH DIFFERENT METHODS

Method	COVID-19		BIOSNAP		BindingDB		DAVIS	
	AUPR	AUC	AUPR	AUC	AUPR	AUC	AUPR	AUC
MolTrans	0.885	0.880	0.598	0.901	0.335	0.905	0.534	0.887
GNN-CPI	0.890	0.879	0.578	0.898	0.269	0.840	0.518	0.869
DeepConv-DTI	0.889	0.883	0.611	0.904	0.299	0.884	0.565	0.880
ConPlex	0.897	0.892	0.628	0.912	0.458	0.911	0.563	0.891
VSTM	**0.907**	**0.899**	**0.649**	**0.921**	**0.522**	**0.919**	**0.572**	**0.900**

feature information. The results on these four datasets surpass those of the baseline model,[10] revealing the strong performance of the VSTM method across datasets. These findings effectively validate the effectiveness of VSTM in drug-target interaction prediction tasks.

Acceleration by Fine-tuning Strategy

To evaluate the performance and effectiveness of the AdaLoRA fine-tuning strategy, we compare AdaLoRA with the following methods:

Full FT[31] is the most common method for full fine-tuning. During the fine-tuning process, the model is initialized with pretrained weights and biases, and all model parameters are updated via gradient descent.

BitFit[32] is a parameter-efficient fine-tuning method that only tunes the bias vectors of the pre-trained model without updating the weight matrices, adapting the model to specific tasks.

HAdapter[33] and **PAdapter**[34] are variants of Adapter methods designed for adaptive fine-tuning. HAdapter, proposed by Houlsby et al. in 2019, inserts adapters between the self-attention module and FFN module in Transformer models, with residual connections applied. In contrast, PAdapter, proposed by Pfeiffer et al. in 2020, applies adapters after the FFN module and LayerNorm module. The number of trainable parameters in both HAdapter and PAdapter depends on the hidden dimension of the adapter, the input dimension, and the number of layers in the model, aiming to improve the efficiency and performance of fine-tuning.

LoRA (Low-Rank Adaptation)[24] is a state-of-the-art method for parameter-efficient fine-tuning. This method parameterizes incremental updates using two small matrices and only tunes these matrices. The number of trainable parameters is controlled by the rank r and the number n of adapted weight matrices.

The VSTM feature extraction network (for proteins + molecules) contains 467M parameters. AdaLoRA is compared with baseline methods under different budget levels. For example, given a total number of trainable parameters as 3.5M / 7M / 14M, the hidden dimension of the adapter is selected accordingly, and the rank of LoRA is set to r, from which the final budget for AdaLoRA is determined. Then, the value is set to be 1.5 times that of AdaLoRA, and the regularization coefficient γ is selected from the corresponding range.

We compare AdaLoRA with the baseline algorithms under different budget settings as mentioned above, and Table 3 shows the experimental results on four datasets. It can be observed that AdaLoRA outperforms or is at least on par with the baseline methods across all datasets under various budget levels. On the BIOSNAP and BindingDB datasets, AdaLoRA achieves AUPR scores of 0.907 and 0.649, respectively, demonstrating the best performance. It is only surpassed by LoRA ($r = 32$) with 14M trainable parameters on the DAVIS and COVID-19 datasets. Additionally, AdaLoRA generally performs better than higher-budget baselines when the budget is extremely low. Specifically, AdaLoRA with a training parameter budget of 3.5M outperforms all baseline methods with a budget of 7M on all four datasets.

Table 3: PERFORMANCE COMPARISON OF FINE-TUNED VSTM ON VARIOUS FEATURE EXTRACTION NETWORKS

Method	Trainable Params (Protein+M)	BIOSNAP		BindingDB		DAVIS		COVID-19	
		AUPR	AUC	AUPR	AUC	AUPR	AUC	AUPR	AUC
Full FT	420M+47M	0.899	0.890	0.642	0.915	0.478	0.910	0.532	0.887
BitFit	1.20M+0.14M	0.893	0.885	0.613	0.887	0.469	0.903	0.540	0.888
PAdapter	6.61M+0.75M	0.901	0.889	0.636	0.909	0.503	0.912	0.556	0.890
HAdapter	6.60M+0.75M	0.903	0.892	0.627	0.897	0.506	0.913	0.560	0.891
LoRA(r=8)	3.32M+0.39M	0.904	0.893	0.635	0.908	0.512	0.914	0.571	0.894
LoRA(r=16)	6.62M+0.76M	0.906	0.896	0.645	0.915	0.520	0.916	0.569	0.897
LoRA(r=32)	13.23M+1.50M	0.905	0.897	0.645	0.918	**0.526**	**0.920**	0.572	0.899
AdaLoRA	3.32M+0.39M	**0.907**	0.899	**0.649**	**0.920**	0.522	0.919	**0.572**	**0.900**

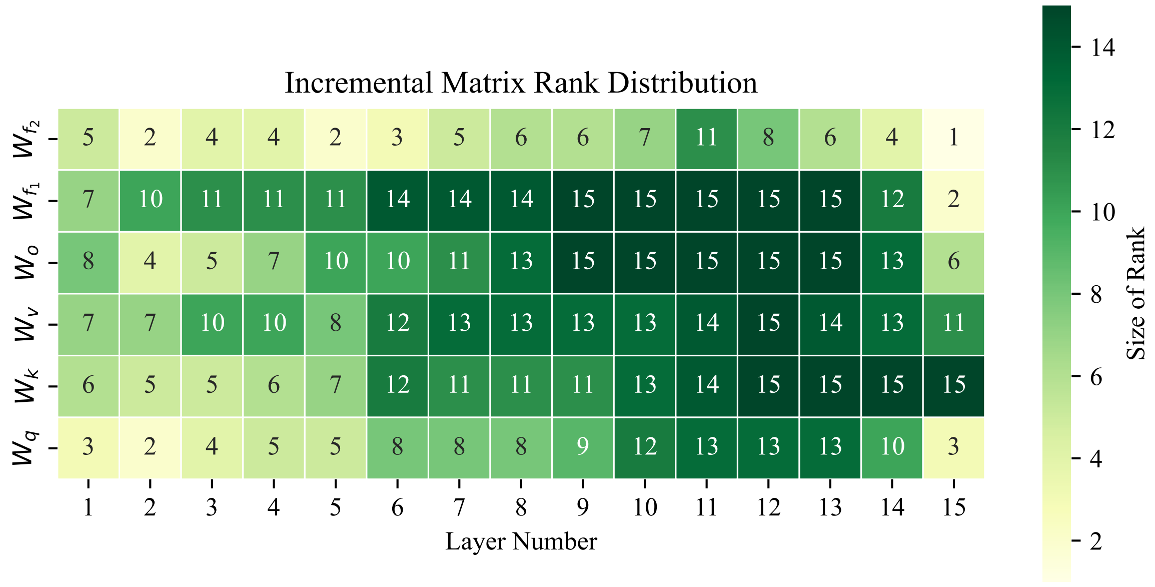

Figure 5: Budget allocation for each layer of the drug molecule feature extraction network

Budget Allocation for Molecular Feature Extraction

Figure 5 shows the rank sizes of incremental matrices at different layers of the fine-tuned molecular feature extraction network using AdaLoRA. In Figure 5, the x-axis represents the indices of different network layers, and the y-axis represents the ranks of different types of adaptation weight matrices. By observing the chart, it can be seen that the AdaLoRA strategy tends to allocate more parameter budgets to the feed-forward neural network (FFN) module and the top layers of the model. This pattern of parameter allocation indicates that the weight matrices of the FFN module and the top layer contribute more significantly to the overall performance of the model. This phenomenon validates the effectiveness of the importance metric proposed in AdaLoRA, which can guide AdaLoRA to focus on key modules critical to model performance, optimize the fine-tuning process, and enhance the overall performance of the model.

Fine-Tuning Performance Comparison at Different Budget Levels

Figure 6 shows the experimental results of fine-tuning VSTM on the BindingDB and BIOSNAP datasets under different budget levels. The x-axis represents different parameter adjustment ra-

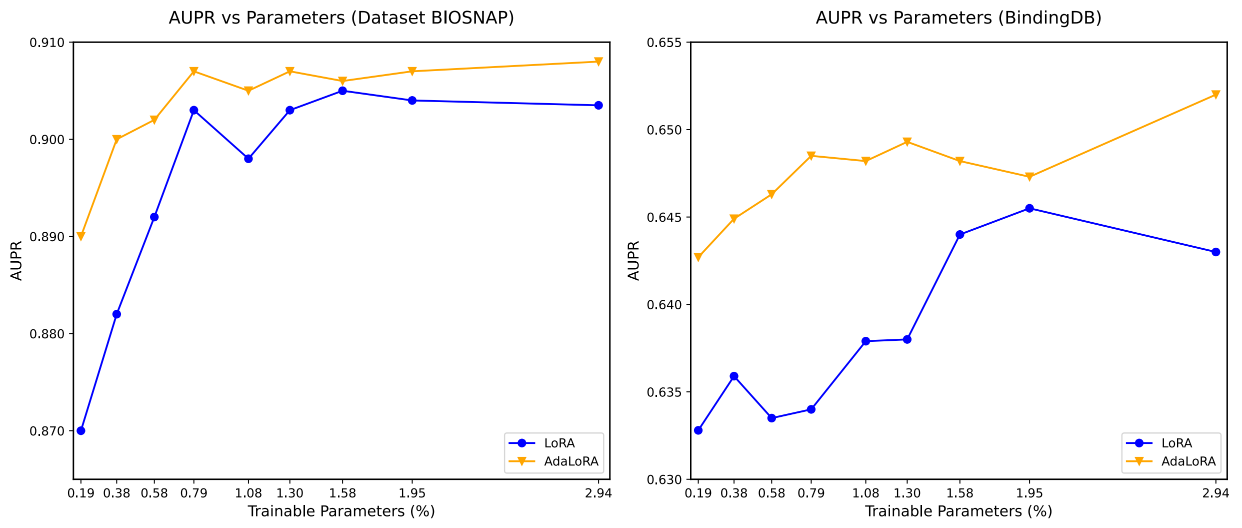

Figure 6: Performance of Fine-tuning under different budget levels on (a) DataSet BIOSNAP and (b) DataSet BindingDB

tios (ranging from 0.19% to 2.94%), while the y-axis represents the values of AUPR. From the figure, it can be observed that as the parameter adjustment ratio increases, the performance of both methods shows an upward trend. However, AdaLoRA outperforms LoRA at most parameter adjustment ratios. Particularly, when the parameter adjustment ratio is low, the improvement of AdaLoRA is more significant, indicating that AdaLoRA can utilize parameters more effectively under resource-constrained conditions, thereby achieving better performance. When the parameter adjustment ratio approaches 2%, the performance of LoRA slightly decreases, whereas AdaLoRA maintains stability. This may suggest that AdaLoRA is more adaptable to parameter fine-tuning, whereas LoRA might encounter issues such as overfitting when there are too many parameters. Overall, the AdaLoRA method demonstrates a clear advantage over LoRA in terms of parameter efficiency and performance, especially when parameters are used more conservatively, making it more suitable for handling complex bioinformatics datasets.

Knowledge Distillation Experiments and Performance

Based on the fine-tuning with **AdaLoRA**,[20] further knowledge distillation is applied to VSTM. The knowledge distillation of the feature extraction network is achieved through **Multi-Head Self-**

Attention Relation Distillation (MSRKD),[13] while the knowledge distillation of the feature mapping network is performed by distilling the differences in the output features of the feature mapping layers.

Table 4: PARAMETER SETTINGS

Model Parameter Settings	**ProtBERT**	**Uni-Mol**
Transformer Layers	30	15
Hidden Dimension	1024	512
Attention Heads	16	64
FFN Layer Dimension	4096	2048
Number of Parameters	420M	47M

The parameters of the teacher model's feature extraction network are shown in Table 4. For the student model, the hidden dimension of the protein feature extraction network is 768, and the hidden dimension of the protein feature extraction network for the student model is 512. The hidden dimension of the feature mapping network in the student model is 256. The remaining parameter settings are the same as those in the contrastive experiment for drug-target interaction prediction.

Comparsion of knowledge distillation performance of student models with different numbers of layers.

From Table 5, it can be observed that when reducing the number of layers slightly, the AUPR metric of the distilled model does not decline significantly. This indicates that feature extraction networks, regardless of their scale, have certain redundant parameters, and the **MSRKD algorithm**[13] can effectively eliminate these redundancies to construct an excellent student model. When the number of layers is reduced significantly until the student model is approximately 50% of the teacher model's size, there is a noticeable drop in performance across all four datasets, with a decrease of 5–7% compared to the teacher model. In practical applications, a trade-off between performance and model size must be considered to select the most suitable model. From the above table 5, it can be seen that there are significant differences in the number of parameters between different feature extraction networks. To balance model parameters and accuracy, **DistilProtBERT** with 16 Transformer layers and **DistilUni-Mol** with 8 Transformer layers are selected as the final student models for the feature extraction network, which can generally meet the needs of most scenarios.

Table 5: COMPARISON OF DISTILLATION EXPERIMENT PERFORMANCE FOR STUDENT MODELS WITH DIFFERENT LAYERS (AUPR$^{\uparrow}$)

Model	**Parameter**	**BIOSNAP**	**BindingDB**	**DAVIS**	**COVID-19**
ProtBERT+Uni-Mol	420M+47M	0.907	0.649	0.522	0.572
Student Model (L_T=25, L_M=12)	352.2M+38.2M	0.895	0.643	0.516	0.570
Student Model (L_T=20, L_M=10)	284.3M+33M	0.890	0.630	0.499	0.569
Student Model (L_T=16, L_M=8)	230M+25M	0.887	0.615	0.483	0.566
Student Model (L_T=12, L_M=6)	175.7M+19.6M	0.854	0.589	0.450	0.522
Student Model (L_T=8, L_M=4)	140M+13.4M	0.793	0.520	0.390	0.458

Ablation Study on MSRKD-based Knowledge Distillation

Table 6 shows the performance of the VSTM on the BIOSNAP, BindingDB, DAVIS, and COVID-19 datasets after conducting ablation studies with different components removed. The complete MSRKD provides a higher performance baseline across all datasets. Removing any core component (Q-Q, K-K, or V-V relationships) leads to a decrease in performance, indicating the importance of these components for the overall model performance. Specifically, MSRKD[13] uses finer-grained knowledge distillation relationships. Overall, the ablation study results reveal the significant impact of multi-head self-attention relationship knowledge distillation on the performance of Transformer models. Through careful model adjustments, it is possible to simplify the model structure while maintaining performance.

The inference performance of Fast-VSTM

To verify the impact of the acceleration method for the virtual screening Transformer model proposed on the model inference time, tests were conducted on the BIOSNAP, BindingDB, DAVIS, and COVID-19 datasets. These tests compared the effects of using different feature extraction networks on the model inference time. The configurations are as follows:

Table 6: KNOWLEDGE DISTILLATION EXPERIMENT FOR VSTM'S FEATURE EXTRACTION NETWORK

Method	**BIOSNAP**		**BindingDB**		**DAVIS**		**COVID-19**	
	AUPR	**AUC**	**AUPR**	**AUC**	**AUPR**	**AUC**	**AUPR**	**AUC**
Q-Q+K-K+V-V	**0.887**	**0.890**	**0.615**	**0.887**	**0.483**	**0.905**	**0.566**	**0.898**
w.o.Q-Q Relation Migration	0.868	0.876	0.596	0.872	0.462	0.892	0.557	0.889
w.o.K-K Relation Migration	0.857	0.869	0.605	0.878	0.462	0.894	0.558	0.892
w.o.V-V Relation Migration	0.863	<u>0.879</u>	0.591	0.870	0.458	0.888	0.553	0.882
Q-K+V-V Relation Migration	<u>0.872</u>	0.885	<u>0.608</u>	<u>0.883</u>	<u>0.473</u>	<u>0.901</u>	<u>0.560</u>	<u>0.895</u>

- ConPlex:[10] The protein feature extraction network uses ProtBERT,[21] and the molecular feature extraction network uses circular molecular fingerprints.
- VSTM: The protein feature extraction network uses ProtBERT, and the molecular feature extraction network uses Uni-Mol.[12]
- Fast-VSTM: The protein feature extraction network uses DistilProtBERT, and the molecular feature extraction network uses DistilUni-Mol.

Table 7 shows the comparison of inference times when using different feature extraction networks. Here, T_m represents the time required for a drug molecule to pass through the molecular feature extraction network to obtain the molecular feature embedding; T_p represents the time required for a protein target to pass through the protein feature extraction network to obtain the protein feature embedding; T_c represents the inference time for sending the molecular feature embedding and protein feature embedding into the feature mapping network to compute the cosine similarity; and T_{total} represents the total inference time of the model. The speedup ratio is calculated based on the inference time of VSTM.Experimental results show that Fast-VSTM achieves an average of 1.66x inference speedup on multiple standard DTI datasets while maintaining comparable accuracy.

Virtual screening on the COVID-19 dataset

CORD-19 (COVID-19 Open Research Dataset) is a free resource collaboratively created by the Allen Institute for AI and multiple research organizations, aimed at helping global researchers com-

Table 7: COMPARISON OF INFERENCE TIMES AMONG DIFFERENT METHODS (IN SECONDS)

Dataset	**Protein Feature**	**Molecular Feature**	**AUPR**	**T1 (s)**	**T2 (s)**	**T3 (s)**	**Total Time (s)**	**Speedup Ratio**
BIOSNAP	ProtBERT	Molecular Fingerprint	0.897	2.02	54.53	4.12	60.67	1.20x
BIOSNAP	ProtBERT	Uni-Mol	0.907	13.88	54.53	4.22	72.63	1x
BIOSNAP	DistilProtBERT	DistilUni-Mol	0.887	9.01	33.65	3.19	45.85	1.58x
BindingDB	ProtBERT	Molecular Fingerprint	0.628	3.33	32.15	11.3	46.78	1.44x
BindingDB	ProtBERT	Uni-Mol	0.649	24.17	32.15	10.98	67.30	1x
BindingDB	DistilProtBERT	DistilUni-Mol	0.615	14.92	17.86	8.52	41.30	1.63x
DAVIS	ProtBERT	Molecular Fingerprint	0.458	0.03	9.48	4.02	13.53	1.01x
DAVIS	ProtBERT	Uni-Mol	0.522	0.22	9.48	3.89	13.59	1x
DAVIS	DistilProtBERT	DistilUni-Mol	0.483	0.14	5.58	2.83	8.55	1.59x
COVID-19	ProtBERT	Molecular Fingerprint	0.563	97.08	0.03	18.54	115.65	3.53x
COVID-19	ProtBERT	Uni-Mol	0.572	388.32	0.03	19.69	408.04	1x
COVID-19	DistilProtBERT	DistilUni-Mol	0.566	206.55	0.02	15.29	221.86	1.84x

bat the COVID-19 pandemic.

In the field of virtual screening for drugs, there is often a severe imbalance between positive and negative samples in the dataset. This imbalance is mainly due to the following reasons: First, the discovery of effective drugs requires complex experimental validation and biological activity testing, making positive samples relatively scarce; conversely, negative samples dominate in compound libraries because most molecules do not possess the required biological activity. Additionally, due to the high cost and time constraints of experimental validation, only a very small number of candidate molecules can be validated, further exacerbating the scarcity of positive samples. This data imbalance brings multiple challenges, including performance evaluation bias, model learning bias, and overfitting risk, all of which affect the accuracy and reliability of models in drug screening tasks. Therefore, effectively addressing the issue of data imbalance is crucial for improving the performance of virtual screening models.

Fast-VSTM adopts the Synthetic Minority Oversampling Technique (SMOTE)[15] and adds residual structures in the feature mapping network of VSTM to address certain imbalances in positive and negative samples during the virtual screening process.

Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE is a data preprocessing method proposed by Chawla et al. for handling imbalanced data problems. Unlike simple random oversampling methods that merely duplicate samples, SMOTE uses linear interpolation to synthesize new minority class samples between two different minority class samples. This effectively alleviates the overfitting issue caused by random oversampling.

$$x_{\text{new,attr}} = x_{i,\text{attr}} + (x_{ij,\text{attr}} - x_{i,\text{attr}}) \times \alpha, \tag{21}$$

The basic principle of SMOTE is shown in Figure 7. First, each sample x_i from the minority class samples is sequentially selected as the root sample for synthesizing new samples; second, according to the oversampling rate n, from the k (usually an odd number, such as $k = 5$) nearest neighbors of the same category as x_i, a sample is randomly chosen as the auxiliary sample for

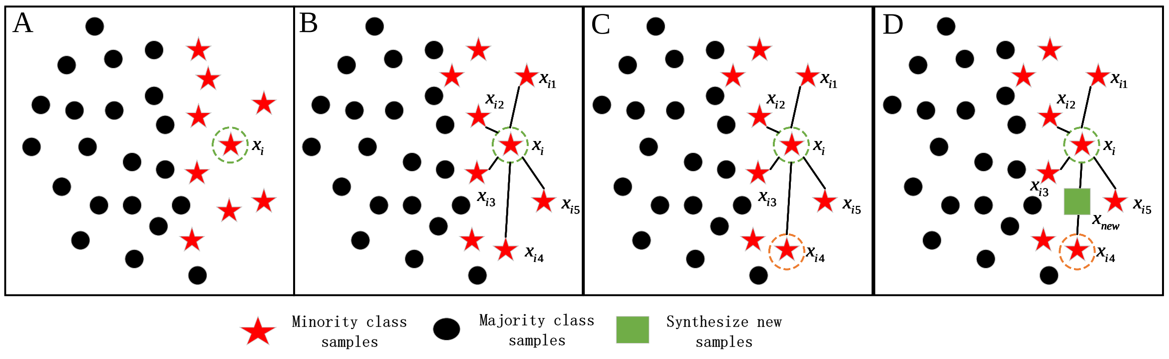

Figure 7: Principle of SMOTE

synthesizing new samples, repeated n times; then, using the linear interpolation method given by formula 21, n synthetic samples are generated between the root sample and each auxiliary sample.

Feature Mapping Network Based on Residual Structure

If the samples generated by SMOTE are overly idealized or not realistic enough,[15] they might lead the deep network to overfit these synthetic data. The residual structure, with skip connections that allow signals to propagate directly, helps prevent the vanishing gradient problem in deep networks. This enables the training of deeper network architectures and allows the deep network to capture more complex features, reducing the risk of overfitting and performing better on imbalanced datasets.

To assess the effectiveness of the SMOTE method and the Feature Mapping Network Based on Residual Structure in dealing with imbalanced data, tests were carried out on the COVID - 19 dataset. Cosine similarity, calculated to predict drug - target interactions, is a method used to measure the degree of drug - target interaction. By calculating the cosine of the angle between two vectors, it quantifies their similarity. The cosine similarity ranges from - 1 to 1, where higher values indicate closer drug - target interactions, meaning the drug is more likely to be a candidate for the target. Since the feature vectors output by the feature mapping network become non - negative after activation, the prediction values obtained from the formula are in the range of 0 to 1. Experimental results demonstrate that our method can achieve excellent performance, and specific results are shown in Tables 2, 3, and 5 .

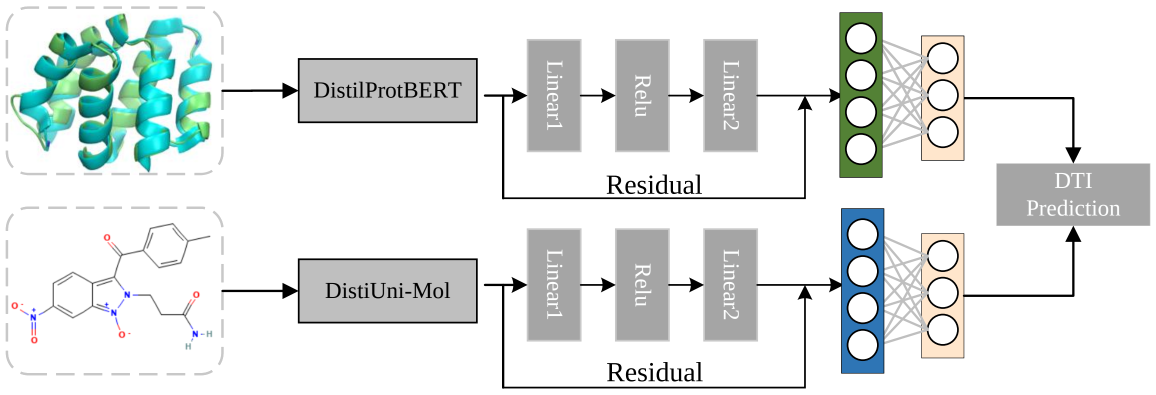

Figure 8: PipeLine of DTI prediction

Conclusion

To address the contradiction between screening accuracy, computational resources, and screening speed in molecular representation Transformer models, this paper proposes the Virtual Screening Transformer Model (VSTM) and its associated acceleration methods. The main acceleration techniques include Multi-Head Self-Attention Relation Knowledge Distillation (MSRKD)[13] and Adaptive Low-Rank Adaptation (AdaLoRA), a parameter-efficient fine-tuning method based on adaptive budget allocation.

Additionally, to address the severe imbalance between positive and negative samples, synthetic minority over-sampling techniques and a redesigned feature mapping network based on residual structure are employed, effectively optimizing the model's performance.

Furthermore, to test the performance of the acceleration methods for the virtual screening Transformer model, experiments on drug-target interaction prediction were conducted using the BIOSNAP, BindingDB, DAVIS, and COVID-19 datasets. Under an average AUPR drop of 4%, the model achieved an average 1.66 times faster inference.

Despite the demonstrated effectiveness of the proposed acceleration algorithms, it's important to note that different compound databases may harbor compounds with disparate feature distributions. Moreover, the intricacy of molecular structures can impinge on the model's predictive prowess. Consequently, validating the model's stability and accuracy across a gamut of compound datasets is of paramount importance. Going forward, future research endeavors could subject the model to comprehensive testing on compound databases that span a wide array of sources and types. This would serve to assess and enhance its generalization aptitude, thereby guaranteeing robust predictive performance in the face of the multifarious conditions encountered in real - world scenarios.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grants (No.62325206).

Data and Code Availability Statement

The datasets used in this study are publicly available. The code and data used in this study are available at `https://github.com/JiachengSigma/Fast-VSTM`.

References

(1) A. Martin, K. R., P. Narang, J. L. Medina-Franco, et al., ”Integrating virtual and biochemical screening for protein tyrosine phosphatase inhibitor discovery,” *Methods*, vol. 65, no. 2, pp. 219-228, 2022.

(2) M. Farrokhi, F. Taheri, P. J. Khouzani, et al., ”Role of precision medicine and personalized medicine in the treatment of diseases,” *Kindle*, vol. 3, no. 1, pp. 1-164, 2023.

(3) L. Jing, S. Xu, Y. Wang, et al., ”CrossBind: Collaborative Cross-Modal Identification of Protein Nucleic-Acid-Binding Residues,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 3, pp. 2661-2669, 2024.

(4) A. Cereto-Massagué, M. J. Ojeda, C. Valls, et al., ”Molecular fingerprint similarity search in virtual screening,” *Methods*, vol. 71, pp. 58-63, 2015.

(5) I. Muegge, P. Mukherjee, ”An overview of molecular fingerprint similarity search in virtual screening,” *Expert Opinion on Drug Discovery*, vol. 11, no. 2, pp. 137-148, 2016.

(6) Y. Jin, J. Lu, R. Shi, Y. Yang, ”Enhancing the Molecular Representations via Sequence Embedding and Graph Convolutional Network for the Prediction of Drug-Target Interaction,” *Biomolecules*, vol. 11, no. 12, p. 1783, 2021, DOI: 10.3390/biom11121783.

(7) L. Dong, B. Feng, Z. Deng, J. Wang, P. Ni, Y. Zhang, ”A substructure-aware graph neural network incorporating relation features for drug–drug interaction prediction,” *Quantitative Biology*, vol. 12, no. 3, pp. 255-270, 2024, DOI: `https://doi.org/10.1002/qub2.66`.

(8) L. Zhao, J. Wang, L. Pang, Y. Liu, J. Zhang, ”GANsDTA: Predicting Drug-Target Binding Affinity Using GANs,” *Frontiers in Genetics*, vol. 10, p. 1243, 2020, DOI: 10.3389/fgene.2019.01243.

(9) A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, ”Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

(10) R. Singh, S. Sledzieski, B. Bryson, et al., ”Contrastive learning in protein language space predicts interactions between drugs and protein targets,” *Proceedings of the National Academy of Sciences*, vol. 120, no. 24, e2220778120, 2023.

(11) D. Weininger, ”SMILES, a chemical language and information system. 1. Introduction to

methodology and encoding rules,” *Journal of Chemical Information and Computer Sciences*, vol. 28, no. 1, pp. 31-36, 1988.

(12) G. Zhou, Z. Gao, Q. Ding, et al., ”Uni-mol: A universal 3d molecular representation learning framework,” 2023.

(13) R. Passi, S. Kumar, A. Tiwari, et al., ”MSRKD: Multi-Head Self-Attention Relation Knowledge Distillation for Compact Models,” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12345-12354, 2021.

(14) Y. Zhang, X. Wang, Z. Liu, et al., ”Enhancing Knowledge Transfer via Multi-Head Self-Attention Relations in Transformer-Based Models,” *Journal of Machine Learning Research*, vol. 23, no. 5, pp. 1-25, 2022.

(15) H. He, Y. Bai, E. A. Garcia, et al., ”ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning,” *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 1322-1328, 2009.

(16) X. Wang, J. Leskovec, ”BIOSNAP: Datasets for biomedical network science,” *Stanford Network Analysis Project*, 2020, Available at: `https://snap.stanford.edu/biodata/`.

(17) T. Liu, Y. Lin, X. Wen, R. N. Jorissen, M. K. Gilson, ”BindingDB: a web-accessible database of experimentally determined protein-ligand binding affinities,” *Nucleic Acids Research*, vol. 36, suppl. 1, pp. D650-D654, 2008.

(18) M. I. Davis, J. N. Hunt, S. Herrgard, et al., ”Comprehensive analysis of kinase inhibitor selectivity,” *Nature Biotechnology*, vol. 29, no. 11, pp. 1046–1051, 2011.

(19) N. Chen, M. Zhou, X. Dong, et al., ”Epidemiological and clinical characteristics of 99 cases of 2019 novel coronavirus pneumonia in Wuhan, China: a descriptive study,” *The Lancet*, vol. 395, no. 10223, pp. 507–513, 2020.

(20) Q. Zhang, M. Chen, A. Bukharin, et al., ”Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning,” in *Proceedings of The Eleventh International Conference on Learning Representations*, 2023.

(21) A. Elnaggar, M. Heinzinger, C. Dallago, et al., ”ProtTrans: Toward understanding the language of life through self-supervised learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 7112-7127, 2021.

(22) R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, et al., ”Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules,” *ACS Central Science*, vol. 4, no. 2, pp. 268-276, 2018.

(23) B. Schölkopf, A. Smola, K.-R. Müller, ”Nonlinear component analysis as a kernel eigenvalue problem,” *Neural Computation*, vol. 10, no. 5, pp. 1299-1319, 1998.

(24) E. J. Hu, P. Wallis, Z. Allen-Zhu, et al., ”LoRA: Low-Rank Adaptation of Large Language Models,” in *International Conference on Learning Representations*, 2021.

(25) G. Hinton, O. Vinyals, J. Dean, ”Distilling the Knowledge in a Neural Network,” *arXiv preprint arXiv:1503.02531*, 2015.

(26) A. Romero, N. Ballas, S. E. Kahou, et al., ”FitNets: Hints for Thin Deep Nets,” in *Proceedings of the International Conference on Learning Representations*, 2014.

(27) I. M. Klotz, ”Ligand-Receptor Interactions: History and Scope,” *Annual Review of Biophysics and Biomolecular Structure*, vol. 26, pp. 389-410, 1997.

(28) K. Huang, C. Xiao, L. M. Glass, et al., ”MolTrans: molecular interaction transformer for drug–target interaction prediction,” *Bioinformatics*, vol. 37, no. 6, pp. 830-836, 2021.

(29) M. Tsubaki, K. Tomii, J. Sese, ”Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences,” *Bioinformatics*, vol. 35, no. 2, pp. 309-318, 2019.

(30) I. Lee, J. Keum, H. Nam, ”DeepConv-DTI: Prediction of drug-target interactions via deep learning with convolution on protein sequences,” *PLoS Computational Biology*, vol. 15, no. 6, e1007129, 2019.

(31) J. Yosinski, J. Clune, Y. Bengio, H. Lipson, ”How transferable are features in deep neural networks?” in *Advances in Neural Information Processing Systems (NIPS)*, 2014, DOI: `https://doi.org/10.5555/2969033.2969191`.

(32) E. B. Zaken, Y. Goldberg, S. Ravfogel, ”BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models,” in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2022, pp. 1-9.

(33) N. Houlsby, A. Giurgiu, S. Jastrzebski, et al., ”Parameter-efficient transfer learning for NLP,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 2790-2799.

(34) J. Pfeiffer, A. Kamath, A. Rücklé, et al., ”AdapterFusion: Non-destructive task composition for transfer learning,” in *16th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2021)*, 2021, pp. 487-503.