GAT-PROTAC-Net: Graph Attention Neural Network for activity prediction of PROTACs

Evianne Rovers
evianne.rovers@mail.utoronto.ca

Kevin Umaiyalan
kevin.umaiyalan@mail.utoronto.ca

Runshi Yang
runshi.yang@mail.utoronto.ca

Abstract

For certain diseases, like cancer and autoimmune disorders, accumulation of proteins in human cells are at the source for the development of the disease. Proteolysis Targeting Chimeras (PROTACs) are drugs that target proteins for degradation in the cells. Therefore, a PROTAC is a therapeutic option to treat diseases with overproduction of proteins. The design of PROTACs is a trial-and-error process with limited known computational methods to aid in the design process. DeepPROTACs is the first machine learning based method to screen PROTACs for activity. The model has a reported accuracy of 78% on the PROTAC-database and 68% on a test case study of unseen data. The accuracy leaves room for improvement in prediction accuracy on both the PROTAC-database and test case study. Additionally, the neural network does not consider (graph) attention layers in the network which could improve the accuracy. Therefore, the project aimed to recreate the DeepPROTACs model and develop a new deep learning model called (GAT-PROTAC-Net) that includes attention layers to improve the accuracy of prediction. [1]

1 Introduction

The production and degradation of proteins in cells - cellular homeostasis – is a critical process often deregulated in disease. It is controlled by post-translational modifications that include the addition and removal of markers attached to proteins that serve as flags for downstream processes. For example, ubiquitination - the addition of a ubiquitin tag to a protein - can flag a protein for subsequent degradation. Certain diseases, including cancer, autoimmune diseases, and neurodegenerative diseases are associated with increased protein levels in the cells [3, 9]. Adding the ubiquitin tag to these proteins to target the proteins for degradation could have a therapeutic benefit. E3 ligases which are part of the degradation system can be brought in close proximity to target proteins by compounds that simultaneously engage both proteins (Figure 1). These compounds are called Proteolysis Targeting Chimeras (PROTACs) and consist of two small molecules - each binding to one protein - and a linker that connects these small molecules.

2 Related works

Many PROTACs have been designed to prove this concept with 10 PROTACs reaching clinical trials (e.g. clinicaltrials.gov identifiers NCT03888612, NCT04772885, NCT04830137). However, the design of the PROTACs have been mostly a trial-and-error process with limited computational rational design. The limiting factor in PROTAC design is determining linker length and composition.

[1]`https://github.com/Runshi-Yang/CSC413-Final-Project`

Preprint. Under review.

A structural study showed that efficacy of PROTACs is determined by their ability to induce at least one stable ternary complex which is determined by the size, composition and flexibility of the linker [7]. Current methods for new PROTACs mostly consist of experimentally screen for all linker lengths and compositions to find an active PROTAC[14, 5, 6].

Machine learning can detect patterns in data that cannot be seen by the human eye. Therefore, machine learning models can potentially detect the patterns in the 3D structures of the three components (E2-PROTAC-Target) that could elucidate the mechanism of active PROTACs versus non-active PROTACs. DeepPROTAC is the first deep-learning neural network that uses deep learning to classify PROTACs for activity based on the information of the individual proteins and the PROTAC [8]. This model has an accuracy of 78% data from the PROTAC-db[12, 13] and 68% accuracy on a test case study with unseen data. For this network, the ternary complex (E3-PROTAC-target) gets split up in 5 components (E3 protein, molecule binding E3, linker, molecule binding target, target protein). Each component is processed with a neural network before combining the information. Also, the linker is presented as a SMILES string. Therefore, by using the PROTAC as a whole in a graph representation, more information is encoded in the data and could result in higher accuracy. Furthermore, DeepPROTACs does not utilize (graph) attention layers, an early-stopping mechanism or weight decay. Therefore, we propose to create a neural network that is trained on the proteins and the PROTAC using graph convolutional and attention layers and using regularization techniques, such as early-stopping and weight decay. We will test the ability to reproduce the results obtained in the DeepPROTAC paper first and then compare the performance of the GAT-PROTAC-Net to the DeepPROTAC paper on both the validation accuracy and the area under the receiver operating characteristic (AUROC) on the PROTAC-database [12, 13] and a test case study presented in the DeepPROTAC paper [8].

3 Methods/Experiment

3.1 Dataset creation

The DeepPROTAC paper github does not provide the database used for training. Therefore, a training set was created by extracting PROTACs from the PROTAC-db [12, 13], a database listing all the PROTACs currently developed and their activity profile. The preparation of the dataset is described in the supplementary methods. The database ratio of active/total is 676/949. The test case study was prepared by creating a database of the PROTACs describe in the DeepPROTAC paper [8].

3.2 Reproduction of DeepPROTAC model

The DeepPROTAC model consists of 5 neural networks each processing a part of the data, namely E3 protein, E3 molecule, linker, target molecule and target protein. Only the linker is represented as a SMILES string and processed using an embedding layer + LSTM layer followed by a fully connected layer. The other neural networks have graph input and consist of a node embedding layer, two graph convolutional layers (GCN) and a max-pooling layer. All the output of the 5 neural networks are concatenated and fed into two fully connected layers which predicts a binary classifier (Figure 2). First, the original model published was trained with the following (published) hyperparameters: 30 epochs, batch size 1, 2 GCN layers, max pooling layer, learning rate 0.0001. Then, the model was optimized for these hyperparameters: batch size [1-256], number of GCN layers [1-3] and type of pooling layer [max,mean,sum]. The model was trained for 200 epochs and the highest validation accuracy was recorded. Then, the model was optimized using an early-stopping mechanism where if the validation loss was not changing for 20 epochs the training was stopped [epochs 200] and weight decay [0.1-0.0001].

3.3 Creation of GAT-PROTAC-Net

Since DeepPROTAC separates all the components, we tested the hypothesis of minimising the separation of the components. We explored a deep neural network starting with 3 neural networks processing either the E3 protein, PROTAC or target protein. Rather than representing the PROTAC in the SMILES representation, we use the graph representation as this will provide information about atom bonds and types. Each neural network consists of an node embedding layer, a graph convolutional layer, a graph attention layer [11] and max-pooling layer. For the graph attention layer operation, the attention coefficients are calculated by:

$$\alpha(h_i, h_j) = \text{softmax}(\text{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}h_i||\mathbf{W}h_j])) \tag{1}$$

where h_i and h_j are the hidden representations of two different nodes i and j in a set of nodes with hidden representation $h = h_1, h_2, ..., h_n$, $\mathbf{W}$ is a weight matrix, $\mathbf{a}$ is a learnable parameter vector, $||$ denotes concatenation, and LeakyReLU is the Leaky Rectified Linear Unit activation function. The output features are calculated by:

$$h_i = \sum_{j \in N_i} \alpha_{i,j} W h_j \tag{2}$$

The reason to use a graph attention layer is for two reasons: One, the dot products between the central node's features and its neighbors' features allows the layer to focus on the most relevant parts of the input by assigning higher weights to nodes that are more similar to the central node. In other words, the attention mechanism allows the layer to "pay attention" to the most informative parts of the graph. For example, the graph representation includes nodes and edges representing the backbone atoms and bonds in residues which are present in all proteins. The atoms and bonds of the side-chains in residues differ from protein to protein and therefore the atoms belonging to that group could contain more relevant information for a classification tasks. Second, by assigning attention weights based on node similarity, the graph attention layer can learn to recognize patterns even if they appear in different locations or with different orientations in different graphs. This allows for better generalizability on unseen graphs, for example, proteins or pockets that are not in the training set.

Afterwards, the output of the neural networks are concatenated and run through an attention layer. The self-attention mechanism can be defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

where Q, K, and V are matrices representing the queries, keys, and values, respectively, and d_k is the dimensionality of the key vectors. This attention layer is self-attention layer which helps in determining which neural network and information is most important for PROTAC activity prediction.The attention layer is followed by two fully connected layers. See visualisation of model architecture in Figure 3. The algorithm for our GAT-PROTAC-Net is included in the appendix listing 1.

The hyperparameters for the GAT-PROTAC-Net model was optimized using Optuna [2], the following parameters were optimized: batchsize [1-512], epoch [1-1000], learning rate [0.00001-0.01] and weight decay [0.000001-0.01]. The early-stopping mechanism was implemented where the training was stopped if the validation loss was not changing for 20 epochs [epochs 600].

3.4 DeepPROTAC vs GAT-PROTAC-Net

The original model, the best model out of the hyperparameter for DeepPROTAC and GAT-PROTAC-Net, were run on the training dataset and the final validation accuracy and AUROC was recorded. Then, each model was tested on a test case study which contains 11 actives among the 16 compounds and the prediction accuracy and AUROC are recorded.

4 Results

4.1 Reproduction of DeepPROTAC model

The original model (DeepPROTAC-original) had a validation accuracy of 75% with an AUROC of 0.81 on the training set (Table 4.3). For the hyperparameter tuning, we observe, similarly to the DeepPROTACs paper, that the model with batch size 1, 2 GCN layers and a max pooling layer performs the best. We ran the model with 200 epochs and recorded the highest validation accuracy observed, which in this case was 81.05% with a AUROC of 0.8830 (Table A.2). We observed that this accuracy was reached after the 30 epochs that was mentioned in the paper. Therefore, we implemented a early stopping mechanism to run for more epochs, but would ensure stopping when the validation loss would level. We also observed that the model used an Adam optimizer without weight decay. Therefore, the model was further optimized by implementing weight decay. The DeepPROTAC-Optimized with a weight decay of 0.0001, learning rate of 0.0005 and the early stopping mechanism reached a validation accuracy of 77% and 0.8570 AUROC (Table 4.3).

4.2 Creation of GAT-PROTAC-Net

The best hyperparameters for the GAT-PROTAC-Net obtained using Optuna [2] are batch size 149, epochs 149, learning rate 0.0005, weight decay 0.0001, which resulted in validation loss of 0.31.

4.3 DeepPROTAC vs GAT-PROTAC-Net

When comparing the DeepPROTAC-Original, DeepPROTAC-Optimized, and our model GAT-PROTAC-Net during the training stage, we see that the GAT-PROTAC-Net outperforms the DeepPROTAC-Original and DeepPROTAC-optimized with a validation accuracy of 86% and AUROC of 0.90. The optimized models also show better convergence (Figure 4). However, testing all the models on the test set, we observe that the prediction accuracy is the same for all models (69%) with varying AUROC's. To overcome, the class imbalance, the models were trained and tested using an over-sampled dataset, which did not result in improvement in validation or test accuracy (Table A.2).

Table 1: Performance of model on training dataset & test case set

Model	**Small dataset**		**Test case study**	
	Validation Accuracy	**AUROC**	**Prediction Accuracy**	**AUROC**
DeepPROTAC-Original	75%	0.81	69%	0.64
DeepPROTAC-Optimized	77%	0.85	69%	0.56
GAT-PROTAC-Net	86%	0.90	69%	0.45

5 Discussion

The increase in validation accuracy for both the DeepPROTAC-optimized and GAT-PROTAC-Net can be attributed to increasing number of epochs that is regularized by the early-stopping mechanism. Similarly, with weight decay, the weights in the neural networks are regularized to prevent overfitting and to reduce the complexity of the model. This would result in improved generalizability of the model. The additional improvement in the accuracy for the GAT-PROTAC-Net can be attributed to the attention layers that help filter and focus on the important information in the node features and contenated information. Although, improvement in validation accuracy is observed, this is not observed for the test case study, which demonstrates that these models cannot generalize well to new unseen data. More importantly, when looking at the predictions, all models predict all PROTACs to be active even when over-sampling is employed to even out the ratio of classes. Limitation of this study is the dataset size of 949 datapoints and even with oversampling the dataset is considered very small in the machine learning field. The size is limited by the availability of the 3D structures of the proteins and negative PROTACs. Therefore, future research should be done in increasing the dataset size, for example, using predicted structures for the 3D structures (AlphaFold) or other data augmentation methods.

6 Conclusion

Overall, it is observed that the validation accuracy of model can increase by regularization techniques and using attention layers in the neural network both on graph data and vector data. However, due to low training dataset size, the generalizability of all models was low.

A Appendix

A.1 Supplementary Methods

A.1.1 Creation of database

The number of PROTACs available in the PROTAC-database[12, 13] is 5273. The PROTACs are matched up to corresponding 3D crystal structures from the Protein Data Bank [4]. After curation of the database for the presence of 3D crystal structures, the database for training consists of 949 PROTACS. The graph representation of the proteins and PROTAC were generated using PyMol [10] and OpenBabel [1] respectively. Each PROTAC is labelled 0 for 'non-active' if degradation efficacy is <80% or >100nM, otherwise labelled 1 for 'active' if degradation efficacy is >80% or <100nM. One data augmentation method was used to enlarge the dataset, namely over-sampling. With over-sampling the set of inactives are repeatedly sampled till the ratio is even.

A.2 Supplementary Tables

Table 2: Reproduction of DeepPROTAC model on small dataset

Hyperparameters	Values	Highest validation Accuracy	AUROC
Batch size	**1**	81.05%	0.8830
	8	78.95%	0.8541
	16	79.47%	0.8608
	32	78.95%	0.8607
	64	78.42%	0.8438
	128	76.32%	0.8166
	256	78.95%	0.8747
Num of GCN layers	one	80.00%	0.8651
	two	81.05%	0.8830
	three	79.47%	0.8681
Pooling layer	**Max Pooling**	81.05%	0.8830
	Mean Pooling	79.47%	0.8421
	Sum Pooling	76.32%	0.7986

Table 3: Performance of model on training dataset & test case set after over-sampling

Model	Small dataset Validation Accuracy	AUROC	Test case study Prediction Accuracy	AUROC
DeepPROTAC-Original	79%	0.87	69%	0.62
DeepPROTAC-Optimized	80%	0.58	31%	0.58
GAT-PROTAC-Net	78%	0.87	69%	0.63

A.3 Supplementary Figures

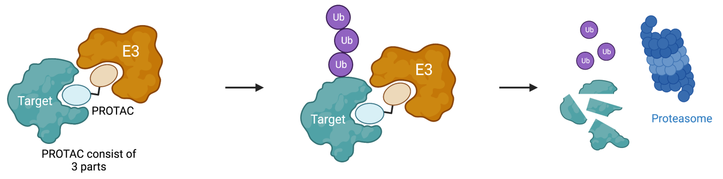

Figure 1: PROTAC bind to two protein simulatenously to induce ubiquitination and degradation of target protein. The complex consists of three parts the target protein (green), PROTAC and E3 protein (orange). The PROTAC consists of three parts, the molecule binding the target (green oval), the molecule binding the E3 (orange oval) and the linker (black).

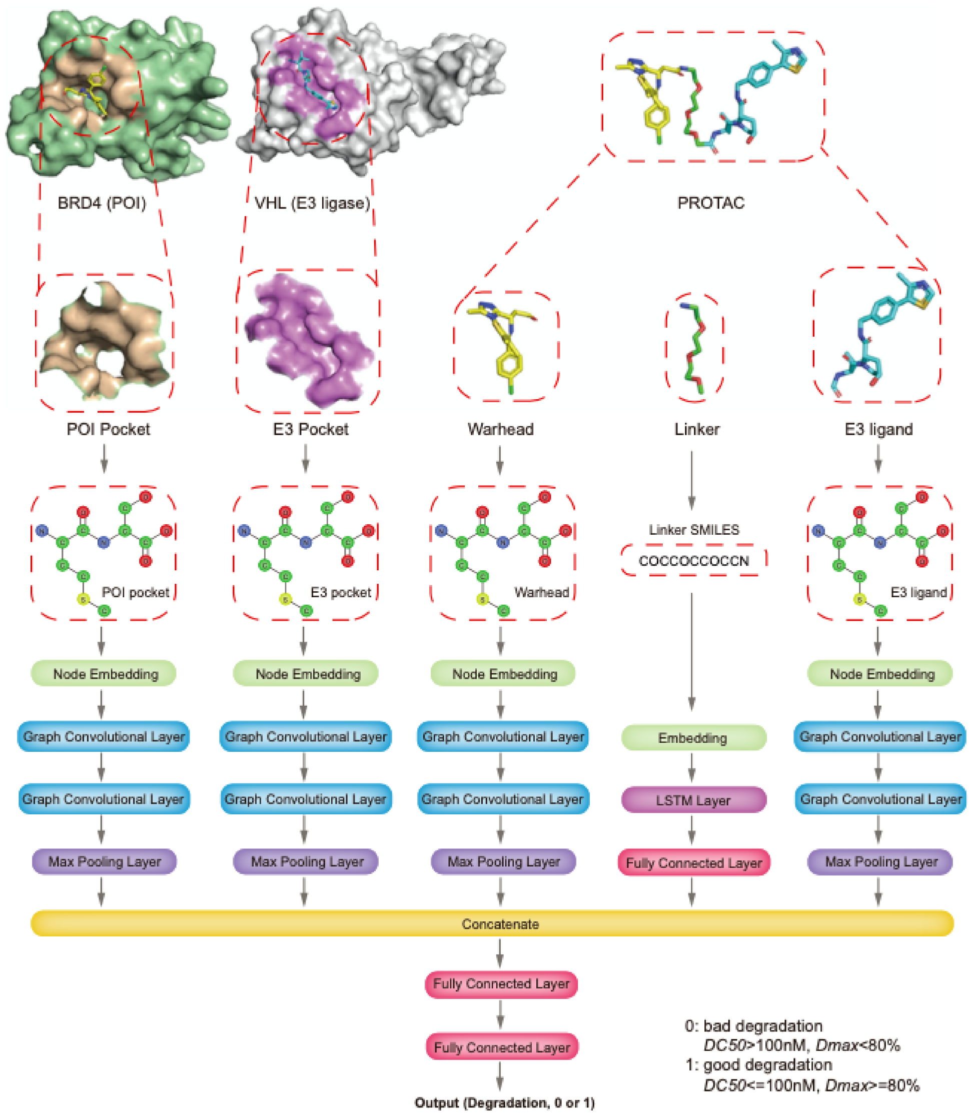

Figure 2: DeepPROTAC

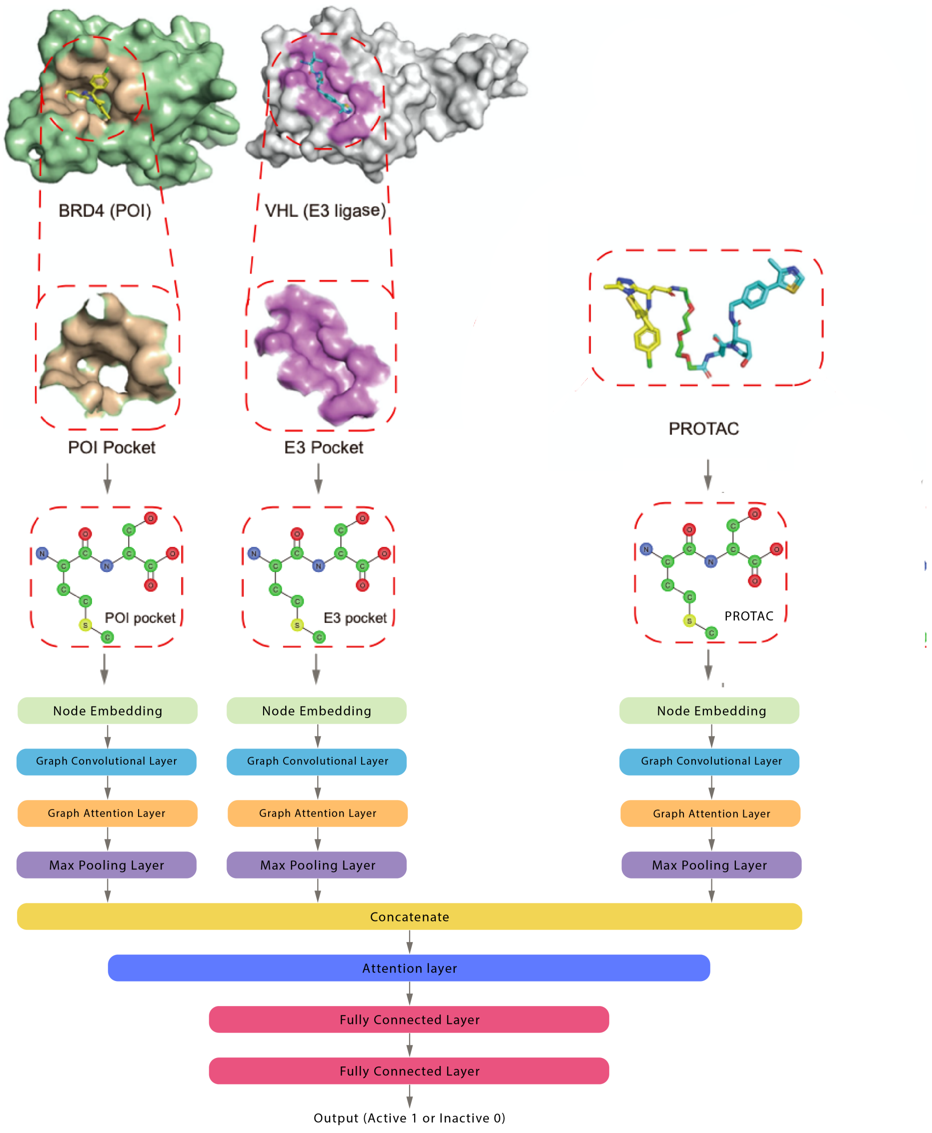

Figure 3: GAT-PROTAC-Net

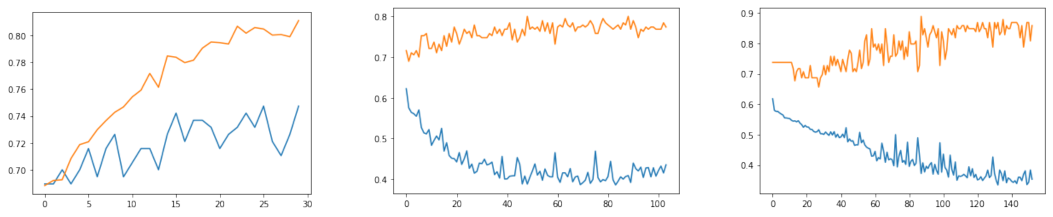

Figure 4: Validation loss (blue) and validation accuracy (orange) for DeepPROTAC-original, DeepPROTAC-Optimized and GAT-PROTAC-Net

A.4 Supplementary Code

Listing 1: Graph Convolutional Network

```
class GraphConv(nn.Module):
    def __init__(self, num_embeddings):
        super().__init__()
        self.embed = nn.Embedding(num_embeddings, embedding_dim = 64)
        self.gcn1 = GCNConv(64, 128)
        self.gat = GATConv(128,64,1)
        self.gcn2 = GCNConv(128, 64)

    def forward(self, data):
        x, edge_index, batch = data.x, data.edge_index, data.batch
        edge_attr = data.edge_attr.to(torch.float)
        x = self.embed(x)
        x = self.gcn1(x, edge_index, edge_attr)
        x = F.relu(x)
        x = self.gat(x, edge_index, edge_attr)
        x = global_max_pool(x, batch)
        return x

class SmilesNet(nn.Module):
    def __init__(self, batch_size = 1):
        super().__init__()
        self.batch_size = batch_size
        self.embed = nn.Embedding(41, 64, padding_idx=0)
        self.lstm = nn.LSTM(64, 64, batch_first=True, bidirectional=True)
        self.fc = nn.Linear(128, 64)

    def forward(self, x, s):
        x = self.embed(x)
        x = pack_padded_sequence(x, s, batch_first=True, enforce_sorted=False)
        out, (h, c) = self.lstm(x, None)
        out, _ = pad_packed_sequence(out, batch_first=True)
        y = self.fc(out[:,-1,:])
        return y

class ProtacModel(nn.Module):
    def __init__(self, ligase_pocket_model, target_pocket_model, PROTAC_model):

        super().__init__()
        self.ligase_pocket_model = ligase_pocket_model
        self.target_pocket_model = target_pocket_model
        self.PROTAC_model = PROTAC_model
        self.fc1 = nn.Linear(64*3,64)
        self.relu = nn.LeakyReLU(negative_slope=0.01)
        self.fc2 = nn.Linear(64,2)
        self.attention = nn.MultiheadAttention(64, 1, batch_first=True)
        self.Q = nn.Linear(64, 64)
        self.K = nn.Linear(64, 64)
        self.V = nn.Linear(64, 64)

    def forward(self, ligase_pocket, target_pocket, PROTAC):
        v_0 = self.ligase_pocket_model(ligase_pocket)
        v_0 = torch.unsqueeze(v_0, 1)
        v_1 = self.target_pocket_model(target_pocket)
        v_1 = torch.unsqueeze(v_1, 1)
        v_2 = self.PROTAC_model(PROTAC)
        v_2 = torch.unsqueeze(v_2, 1)
```

```
v_c = torch.cat((v_0, v_1, v_2), 1)
v_a,a_w = self.attention(self.Q(v_c),self.K(v_c),self.V(v_c))
a,b,c = v_a.shape
v_a = torch.reshape(v_a,(a,b*c))
v_f = self.relu(self.fc1(v_a))
v_f = self.fc2(v_f)
return v_f
```

References

[1] Open Babel: An open chemical toolbox. *Journal of cheminformatics*, 3(10), oct 2011. ISSN 1758-2946. URL https://pubmed.ncbi.nlm.nih.gov/21982300/.

[2] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[3] M. Békés, D. R. Langley, and C. M. Crews. PROTAC targeted protein degraders: the past is prologue. *Nature reviews. Drug discovery*, 21:181–200, 2022. ISSN 1474-1784. doi: 10.1038/s41573-021-00371-6. URL http://www.ncbi.nlm.nih.gov/pubmed/35042991.

[4] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, jan 2000. ISSN 0305-1048. doi: 10.1093/NAR/28.1.235. URL https://academic.oup.com/nar/article/28/1/235/2384399.

[5] M. L. Drummond and C. I. Williams. In Silico Modeling of PROTAC-Mediated Ternary Complexes: Validation and Application. *Journal of Chemical Information and Modeling*, 59 (4):1634–1644, apr 2019. ISSN 15205142. doi: 10.1021/ACS.JCIM.8B00872/SUPPL_FILE/CI8B00872_SI_001.PDF. URL https://pubs.acs.org/doi/full/10.1021/acs.jcim.8b00872.

[6] M. L. Drummond, A. Henry, H. Li, and C. I. Williams. Improved accuracy for modeling ProTAC-mediated ternary complex formation and targeted protein degradation via new in silico methodologies. *Journal of Chemical Information and Modeling*, 60(10):5234–5254, oct 2020. ISSN 15205142. doi: 10.1021/ACS.JCIM.0C00897/SUPPL_FILE/CI0C00897_SI_001.PDF. URL https://pubs.acs.org/doi/full/10.1021/acs.jcim.0c00897.

[7] M. S. Gadd, A. Testa, X. Lucas, K. H. Chan, W. Chen, D. J. Lamont, M. Zengerle, and A. Ciulli. Structural basis of PROTAC cooperative recognition for selective protein degradation. *Nature Chemical Biology 2017 13:5*, 13(5):514–521, mar 2017. ISSN 1552-4469. doi: 10.1038/nchembio.2329. URL https://www.nature.com/articles/nchembio.2329.

[8] F. Li, Q. Hu, X. Zhang, R. Sun, Z. Liu, S. Wu, S. Tian, X. Ma, Z. Dai, X. Yang, S. Gao, and F. Bai. DeepPROTACs is a deep learning-based targeted degradation predictor for PROTACs. *Nature Communications 2022 13:1*, 13(1):1–14, nov 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-34807-3. URL https://www.nature.com/articles/s41467-022-34807-3https://github.com/fenglei104/.

[9] K. Ma, X. X. Han, X. M. Yang, and S. L. Zhou. Proteolysis targeting chimera technology: A novel strategy for treating diseases of the central nervous system. *Neural Regeneration Research*, 16(10):1944–1949, oct 2021. ISSN 18767958. doi: 10.4103/1673-5374.308075. URL https://journals.lww.com/nrronline/Fulltext/2021/16100/Proteolysis_targeting_chimera_technology__a_novel.7.aspx.

[10] Schrödinger, LLC. The PyMOL Molecular Graphics System, Version∼1.8, nov 2015.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2018.

[12] G. Weng, C. Shen, D. Cao, J. Gao, X. Dong, Q. He, B. Yang, D. Li, J. Wu, and T. Hou. PROTAC-DB: an online database of PROTACs. *Nucleic Acids Research*, 49(D1):D1381–D1387, jan 2021. ISSN 0305-1048. doi: 10.1093/NAR/GKAA807. URL https://academic.oup.com/nar/article/49/D1/D1381/5917660.

[13] G. Weng, X. Cai, D. Cao, H. Du, C. Shen, Y. Deng, Q. He, B. Yang, D. Li, and T. Hou. PROTAC-DB 2.0: an updated database of PROTACs. *Nucleic Acids Research*, 51(D1):D1367–D1372, jan 2023. ISSN 0305-1048. doi: 10.1093/NAR/GKAC946. URL `https://academic.oup.com/nar/article/51/D1/D1367/6775390`.

[14] D. Zaidman, J. Prilusky, and N. London. PRosettaC: Rosetta based modeling of PROTAC mediated ternary complexes. *Journal of Chemical Information and Modeling*, 60(10):4894–4903, oct 2020. ISSN 15205142. doi: 10.1021/ACS.JCIM.0C00589/SUPPL_FILE/CI0C00589_SI_001.PDF. URL `https://pubs.acs.org/doi/full/10.1021/acs.jcim.0c00589`.