Condensing Molecular Docking CNNs via Knowledge Distillation

Andrew T. McNutt,[†,¶] Yanjing Li,[‡,¶] Paul Francoeur,[†] and David Ryan Koes[*,†]

[†]*Department of Computational and Systems Biology, University of Pittsburgh, Pittsburgh, PA*

[‡]*Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA*

[¶]*Contributed equally to this work*

E-mail: dkoes@pitt.edu

Abstract

Knowledge of the bound protein-ligand structure is critical to many drug discovery tasks. One tool for *in silico* bound structure elucidation is molecular docking, which samples and scores ligand binding conformations. Recent work has demonstrated that convolutional neural networks (CNNs) for protein-ligand pose scoring outperform conventional scoring functions. Scoring performance can be further increased by taking the average of multiple CNN models, termed ensembles. However, ensembles of large parameter models require significant computational resources and therefore are difficult to apply to high-throughput molecular docking for virtual screening. We investigate knowledge distillation as a framework to condense the knowledge of large, powerful CNN model ensembles into a single reduced CNN model for a significant reduction in computational cost. Ensemble KD produces single models that outperform non-KD trained single models.

Introduction

Understanding the bound, three-dimensional (3D) conformation of small molecules to their targets is critical for drug discovery. Molecular docking serves as an *in silico* tool for the prediction of protein-ligand binding conformations. Molecular docking is generally composed of two steps: sampling, in which the conformational space of the complex is explored, and scoring, where a score denoting the quality of the protein-ligand complex is given to rank the conformations. Traditional scoring functions include force field, empirical, and knowledge-based approaches.[1] Force field-based scoring functions employ molecular mechanics-based energy functions to analyze atomic interactions and calculate scores.[2] Empirical scoring functions assign weights to the energetic interaction terms to better fit experimentally determined binding affinities.[1] Knowledge-based scoring functions extract interaction frequencies or observed patterns from 3D structures of protein-ligand complexes to determine the likelihood of a new sample being from the data distribution.

While traditional scoring functions have been successful in scoring protein-ligand interactions using hand-crafted 3D features, recent advances in deep learning provide the ability to predict scores directly from the 3D structure of the protein-ligand conformation. Deep learning scoring functions utilize multiple layers of learned weights to learn complex functions directly from protein and ligand poses to evaluate the quality of binding. Convolutional neural networks (CNNs) are widely used in computer vision, particularly for grid-structured data, such as images.[3] They can capture both low-level and high-level spatial patterns or features by performing convolutions across the input. Recent studies have demonstrated the considerable potential of CNNs in molecular docking scoring, exemplified by AtomNet[4] and PotentialNet.[5] Notably, GNINA[6,7] has proposed several effective CNN architectures for scoring binding poses and predicting affinity. More recent work has proposed graph neural networks (GNNs) for both pose scoring and pose sampling.[8–11] Both CNNs and GNNs show marked improvement in docking over traditional scoring methods; however, due to heavy computation, they require the use of a GPU for reduced docking time.

McNutt et al.[6] observed that an ensemble of CNN models achieves the best binding pose ranking performance but requires longer running times compared to a single model, especially when used without a GPU (458 s and 72 s for the best ensemble and single model, respectively). When molecular docking is used for high-throughput screening, docking upwards of tens of millions of molecules, reducing computational cost is of foremost importance.[12] Knowledge distillation (KD) was developed to condense the knowledge of a large neural network into another, smaller neural network for faster inference with similar model performance.[13] The small model, the "student," is trained to match the representation of the large neural network, the "teacher," on the training dataset of the teacher. Ensemble KD transfers the knowledge learned by multiple teacher models to a single student model by minimizing the discrepancy between the average representation of the teachers and the student.[14,15] Ensemble KD could therefore significantly reduce the computational overhead of workflows that use an ensemble of large models. We evaluate Ensemble KD to condense the pose ranking power of CNN model ensembles into a single model, thereby decreasing the molecular docking time. Our work shows ensemble KD produces more powerful single models than non-KD model training. We find the distilled models performance is correlated with pose score variation within the teacher ensemble.

Methods

We briefly describe the data and model architectures used for our study and then explore the setup for the distillation of the CNN model ensembles.

Data

Training data

The same datasets used to train GNINA's CNN models are also used in this work. Three datasets are used to impart the CNN models with different expertise.[7] The PDBbind General dataset v2016[16,17] consists of 11,324 protein-ligand complexes with experimentally determined binding affinity data and 201,839 generated binding poses. The CrossDock2020 v1.3 dataset[7] contains cognate protein-ligand complexes and non-cognate (crossdocked) protein-ligand complexes. This dataset includes a total of 22,566,449 binding poses, with protein-ligand poses being at least 0.25 Å RMSD from any other pose in the same pocket to provide a diverse set of poses for training. The updated version 1.3 addresses ligand and receptor misalignment problems and incorrect bond typing problems present in earlier versions. Finally, we used the Redock v1.3 dataset, a subset of the CrossDock2020 v1.3 dataset which removes all non-cognate ligand docked poses and consists of 550,562 generated binding poses.

Every protein-ligand pose in each dataset has two labels. One label is binary, indicating whether the pose is less than or equal to 2 Å RMSD from the ground truth ligand conformation. During redocking experiments, the crystallographic pose of the ligand within the target protein serves as the ground truth. During crossdocking evaluations, the ground truth is designated as the ligand pose after alignment of the cognate protein structure with the receptor conformation used for docking. The second label provides the experimentally determined binding affinity of the protein-ligand complex. For the CrossDock2020 dataset, ligands are assumed to have the same affinity for all receptors of a given pocket. Rather than using the raw K_d of the complex, we take the $-\log_{10}(K_d)$ to teach the scoring functions the magnitude of the complex's binding affinity.

More information on the model training datasets and their labels are provided in Francoeur et al.[7].

Evaluation data

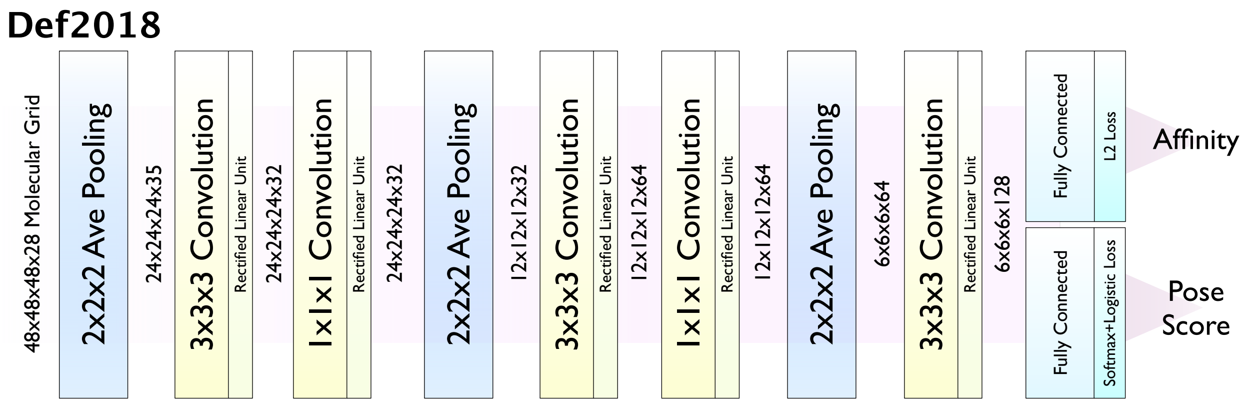
As in McNutt et al.[6], trained models are evaluated on two tasks within the GNINA molecular docking pipeline: redocking and crossdocking. Redocking is evaluated using the PDBbind Refined v.2019 dataset;[16,17] ligands smaller than 150 Da or larger than 1000 Da are removed, resulting in 4,260 protein-ligand complexes. Redocking evaluates the models performance when docking ligands to their cognate receptor structures. However, redocking performance is an unrealistic benchmark since most real-world docking is between new ligands and a non-cognate receptor structure. To this end, we perform a crossdocking evaluation using the dataset from Wierbowski et al.[18]; this dataset provides a benchmark to evaluate protein-ligand docking without a cognate receptor structure. The dataset was filtered via the same size and parsing criterion as the redocking dataset. We used a downsampled subset of the whole dataset for evaluation, which contains 7,970 protein-ligand pairs. More details on the evaluation datasets and the filtering process is provided in McNutt et al.[6].

GNINA's molecular docking time was benchmarked on a random subset of 100 complexes from the PDBbind Core set v2016.[19] A list of the PDB IDs of this subset and the specifications of the benchmark are detailed in the supplement.

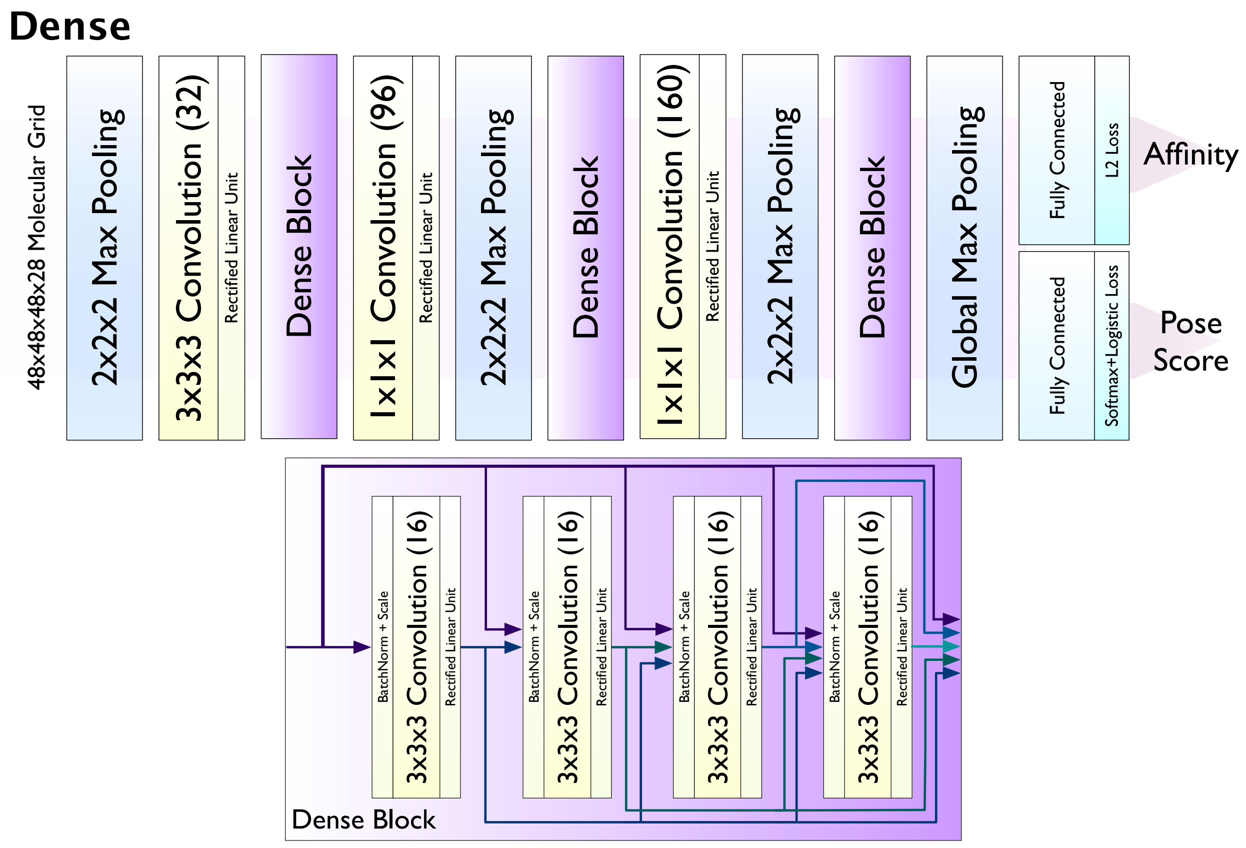
Molecular docking models

Knowledge distillation is performed on two different CNN model architectures. Both architectures take a 3D grid of Gaussian-like atom type densities as input and predict a CNNscore, indicating the quality of the protein-ligand pose, and a binding affinity, as a pK. The smallest architecture, Default2018 (Figure 1a) is composed of a small number of convolutional and average pooling layers. The larger Dense architecture (Figure 1b) contains several max pooling layers, convolutions, and distint units called "dense blocks".[20] Dense blocks are comprised of series of convolutions in which every subsequent layer is provided the outputs of all preceding layers. The Dense and Default2018 architectures are composed of 685,123 and 389,059 parameters, respectively.

The Default2018 architecture was trained on each of the three datasets (PDBBind General, CrossDock2020 v1.3, and Redock v1.3) to generate three models while the Dense architecture was only trained on the CrossDock2020 dataset due to resource constraints. Therefore, there are four variants of CNN models: `general_default2018`, `redock_default2018`, `crossdock_default2018`, and `dense`.



(a) Default2018



(b) Dense

Figure 1: The model architectures take in a voxelized protein-ligand complex and predict both a pose score, termed 'CNNscore', and affinity value.

Ensemble knowledge distillation

Ensemble KD trains a student model on the representation of several teacher models, illustrated in Figure 2. GNINA provides four variants of CNN models for molecular docking; each variant is trained five times using different random seeds. These five models form an ensemble for each variant. Our ensemble KD utilizes the five teacher models to train a single student model with the same architecture as the teacher models. Additionally, we consider one more ensemble of the CNN models: "All Default2018 Ensemble", consisting of all scoring functions with the Default2018 architecture. During KD, the student model is trained on the same dataset as the teachers were trained on. For the All Default2018 Ensemble, we train the student on the CrossDock2020 v1.3 dataset since this is largely a superset of the other training datasets. Overall, 6 single models are distilled from 5 ensembles (Table S2).

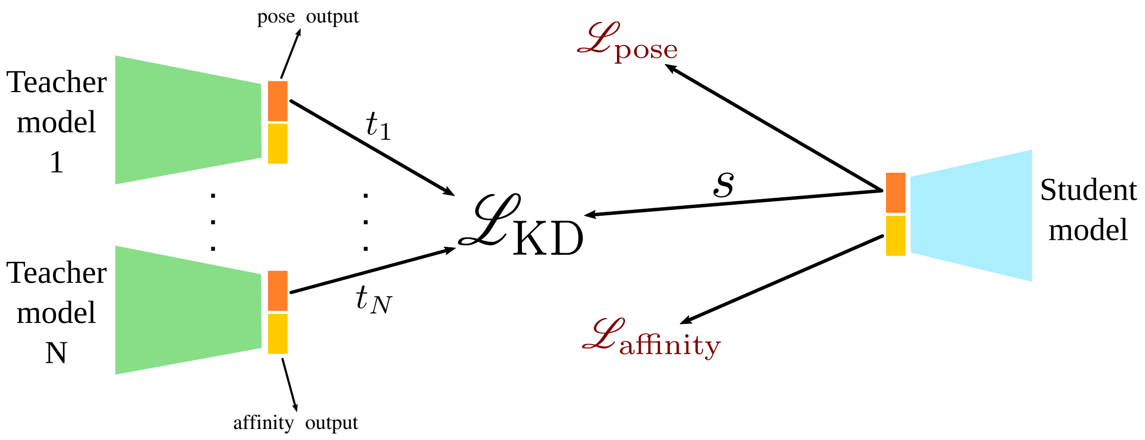

pose output
Teacher model 1
Teacher model N
t_1
t_N
$\mathscr{L}_{\text{KD}}$
s
$\mathscr{L}_{\text{pose}}$
$\mathscr{L}_{\text{affinity}}$
Student model
affinity output

Figure 2: Ensemble knowledge distillation. The ground-truth-based losses are in red, while the KD loss uses the teacher pose classifier outputs, pre-softmax, as the label. The KD loss only utilizes the pose classifier output. The affinity prediction of the students is trained with the same loss as the teachers.

The teacher models predict both a probability that the pose is high quality, $\hat{y}^T_{\text{pose}}$, and an affinity, $\hat{y}^T_{\text{affinity}}$, which are trained on the ground truth pose, y_{pose}, and affinity, y_{affinity}, using cross entropy (1) and hinged mean squared error (2), respectively.

$$\mathscr{L}_{\text{pose}} = -y_{\text{pose}} \log \hat{y}^S_{\text{pose}} - (1 - y_{\text{pose}}) \log (1 - \hat{y}^S_{\text{pose}}) \tag{1}$$

$$\mathscr{L}_{\text{affinity}} = 0.1 \begin{cases} (y_{\text{affinity}} - \hat{y}^S_{\text{affinity}})^2 & (y_{\text{pose}} = 1) \,|\, (y_{\text{pose}} = 0 \;\&\; y_{\text{affinity}} < \hat{y}^S_{\text{affinity}}) \\ 0 & y_{\text{pose}} = 0 \;\&\; y_{\text{affinity}} \geq \hat{y}^S_{\text{affinity}} \end{cases} \tag{2}$$

The student is trained using the same ground-truth-based losses on its affinity, $\hat{y}^S_{affinity}$, and pose classifier, $\hat{y}^S_{pose}$. Additionally, the student is trained to model the output of the teacher's pose classifier with the KD loss (3), the sum of KL divergence between the pre-softmax values of the pose classifier of the student, s, and each teacher, t_i.[13]

$$\mathscr{L}_{\text{KD}} = T^2 \sum_{i=1}^{N} KL(\sigma(s/T), \sigma(t_i/T)) \tag{3}$$

where T is the temperature and σ is the softmax function. The total loss (4) on the student is a weighted sum of the KD loss and the ground truth loss.

$$\mathscr{L}_{\text{total}} = \alpha * (\mathscr{L}_{\text{pose}} + \mathscr{L}_{\text{affinity}}) + \beta * \mathscr{L}_{\text{KD}} \tag{4}$$

where α and β weight the contributions of the ground-truth-based losses and the KD loss, respectively. We set $\alpha = 1$, $\beta = 1$, and $T = 1$ for our KD trainings.

Training details and hyperparameters are provided in the supplement.

Evaluation Metrics

We use the same evaluation metrics as in McNutt et al.[6]. After training the CNN models, we employ them in the GNINA molecular docking pipeline with default settings. We set the random seed to ensure consistency between runs of a given complex. We utilize **TopN** to evaluate the pose ranking performance of the molecular docking pipeline. TopN indicates the percentage of complexes with at least one pose, ranked N or higher, less than 2Å RMSD to ground truth. That is, Top2 is the percentage of docking runs where at least one of the top two ranked poses is less than 2 Å RMSD from the ground truth. We also investigate the computational cost of using these CNN models by evaluating the time to dock ligands. Since our focus is on increasing the efficiency of CNNs for high-throughput molecular docking, we examine the computational cost of the CNNs during CPU-only docking. GPUs can significantly speed up CNN usage; however, docking millions of molecules generally requires parallelizing the process across hundreds to thousands of machines which is prohibitively expensive if GPUs are used. Therefore, we report the **Avg Time Per System**, the average CPU-only docking time across the 100 complex subset of the PDBbind Core set v2016.[19] More information on the calculation of these metrics is provided in the supplement.

Results

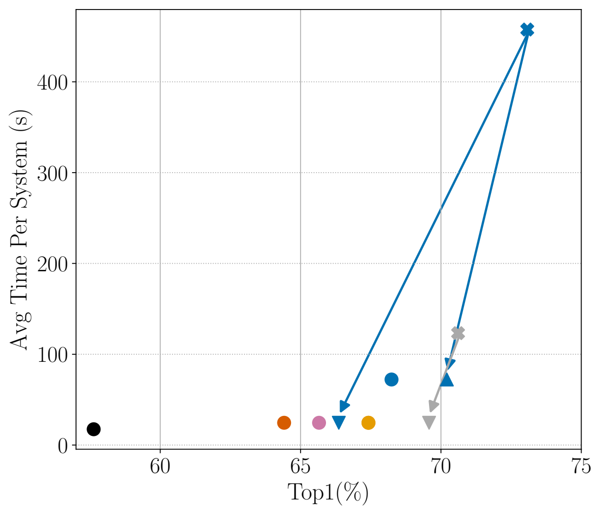
Train on CrossDock2020 v1.3

We retrained all of the built-in GNINA scoring functions using the updated CrossDock2020 v1.3 dataset. This resulted in five new versions of `crossdock_default2018`, `redock_default2018`, and `dense` that show improved pose ranking performance on the crossdocking task (Figure S3, S2, and S4). Following Francoeur et al.[7], we utilized Caffe[21] for the `crossdock_default2018` and `redock_default2018` models. Due to complications with later KD training, the new versions of the `dense` models were trained with PyTorch. Performance of these models is comparable to the Caffe trained models (Figures S1). The PyTorch-trained `dense` models are used for all molecular docking evaluations and KD training. All of the training details and hyperparameters are outlined in the supplement.

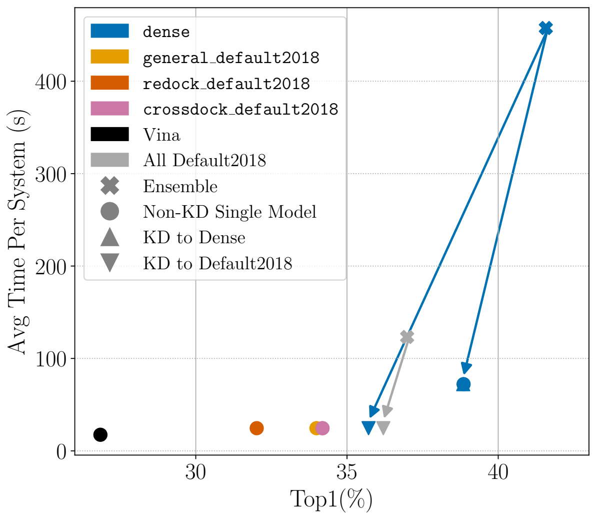
The only decrease in performance is seen in the redocking performance of the `redock_default2018` models (Figure S2a). Both `crossdock_default2018` and `dense` have slightly increased performance on the redocking task after training on the new version of the dataset. We see a general increase in performance for all models during crossdocking (Figure S2b, S3b, and S4b), with `dense` showing the largest increase in performance over the 1.0 version model.

Top1-Time Benchmarking

Our primary goal is to retain docking performance while minimizing running time. We focus our attention on the Avg Time per System when we do not use a GPU since this is the most likely use case during high-throughput docking of large libraries. We present the best performing distilled and non-distilled single models and all the ensembles in Figure 3. Comparisons of all the single models, ensembles, and knowledge distilled models on redocking and crossdocking are provided in Table 1 and 2, respectively.



(a) Redocking



(b) Crossdocking

Figure 3: Evaluation of Top1 vs average time to dock one protein-ligand sample using CPU only. We show the best performing single models, across both tasks, for the non-KD and KD trained models visualized as circles and triangles, respectively. Ensemble KD, shown by the arrows, reduces the computational overhead of the highest performing ensembles (denoted by the "X"s), while having better or comparable rescoring performance on both tasks than any non-KD single model.

First, we compare the performance of the distilled Default2018 ensembles to the non-KD single models. Overall, the distilled Default2018 ensemble models do not have large differences from their non-KD trained counterparts. The distilled `general_default2018` ensemble shows slightly reduced performance on the redocking task, while the other Default2018 models show a slight increase in performance on the redocking task (Figure S6a, S7a, and S8a). The `crossdock_default2018` model shows a slight reduction in performance on the cross-

Table 1: Average redocking performance of GNINA scoring functions. ± indicates the standard deviation of five models trained from different random seeds, except in the case of "All Default2018 Ensemble" distillation which only has three student models.

Model name	Single Model		Ensemble		KD → Default2018	KD → Dense
	Top1 (%)	Avg CPU Time (s)	Top1 (%)	Avg CPU Time (s)	Top1(%)	Top1(%)
`crossdock_default2018`	65.01(±0.62)	24.6	66.48	53.1	66.33(±0.45)	N/A
`redock_default2018`	63.23(±0.69)	24.6	66.41	53.1	65.04(±0.43)	N/A
`general_default2018`	66.82(±0.71)	24.6	66.41	53.1	65.92(±0.48)	N/A
`dense`	67.97(±0.25)	72.2	73.08	457.6	65.69(±0.44)	69.71(±0.48)
All Default2018	N/A	N/A	70.61	123.2	69.46(±0.18)	N/A

Table 2: Average crossdocking performance of GNINA scoring functions. ± indicates the standard deviation of five models trained from different random seeds, except in the case of "All Default2018 Ensemble" distillation which only has three student models.

Model name	Single Model		Ensemble		KD → Default2018	KD → Dense
	Top1 (%)	Avg CPU Time (s)	Top1 (%)	Avg CPU Time (s)	Top1(%)	Top1(%)
`crossdock_default2018`	33.43(±0.59)	24.6	34.30	53.1	33.07(±0.22)	N/A
`redock_default2018`	31.08(±0.78)	24.6	32.87	53.1	32.22(±0.33)	N/A
`general_default2018`	33.23(±0.69)	24.6	35.43	53.1	34.17(±0.48)	N/A
`dense`	38.27(±0.51)	72.2	41.57	457.6	35.93(±0.20)	38.38(±0.39)
All Default2018	N/A	N/A	36.99	123.2	35.93(±0.40)	N/A

docking task, while the `general_default2018` and `redock_default2018` show a slight increase in performance for crossdocking (Figure S6b, S7b, and S8b). Finally, we distill the ensemble composed of all models with a Default2018 architecture, referred to as "All Default2018 Ensemble," into a single Default2018 model. This distilled Default2018 model shows the highest performance of any Default2018 distilled model (Figure S9). The distilled model has higher performance than any single non-distilled Default2018 model and higher performance than any ensemble of Default2018 model on both redocking and crossdocking tasks. This distillation shows the greatest decrease in docking time while nearly maintaining the performance of the full ensemble on both docking tasks.

The `dense` ensemble is the highest performing CNN model for both redocking and crossdocking, but it is also the most computationally demanding (Figure 3) so distilling the `dense` ensemble's knowledge would be the most useful for accurate high-throughput docking. Figure S10 shows the `dense` ensemble distilled into a Dense architecture has about the same crossdocking performance as the non-KD trained `dense` single models while demonstrating improved performance on redocking over the non-KD trained `dense` single models. Additionally, since the Dense architecture has a higher number of parameters than the Default2018 architecture, we distill the `dense` ensemble into a Default2018 architecture. We compare the `dense` ensemble to Default2018 architecture distillation to the non-KD trained `crossdock_default2018` single models and ensemble since it has the same architecture and training dataset. The distillation into the Default2018 architecture shows about the same redocking performance as `crossdock_default2018` single models and a higher crossdocking performance relative to both the `crossdock_default2018` single models and ensemble. However, the distillation of the `dense` ensemble into a single Default2018 model shows about the same crossdocking performance and reduced redocking performance when compared to distilling the "All Default2018 Ensemble" into a Default2018 model(Figure 3, Table 2 and 1). There is significant pose ranking performance lost during the distillation of the `dense` ensemble to both the Dense and Default2018 architecture.

Discussion

In this work, we established the effectiveness of KD in reducing the cost of CNN model ensembles used by GNINA for scoring protein-ligand binding conformations while maintaining their pose ranking superiority over non-KD-trained single models. Upon applying ensemble KD, the Top1 performance for the `redock_default2018`, All Default2018, and `dense` distilled models in redocking and crossdocking tasks landed between the best-performing single models and their corresponding ensembles (Figures S7, S9, and S10). The distilled `general_default2018` model did not outperform the best single model for redocking; however, the distilled models show better performance than the single models for crossdocking (Figure S6). The `crossdock_default2018` distilled models showed the opposite trend, with increased performance on redocking tasks and decreased performance on crossdocking tasks over the single models (Figure S8). We hypothesize the failing of these distillations is due to a lack of diversity among the single model pose outputs in these ensembles (Figure S18). This aligns with the low improvement seen of the `crossdock_default2018` ensemble over the single models on crossdocking (Figure S8b) and the decrease in performance of the `general_default2018` ensemble over the highest performing single model for redocking (Figure S6a). We also note that the CNNscore learned through KD training is indicative of lower RMSD to ground truth poses (Figures S12, S13, S14, S16 and S17). Therefore, the KD process is teaching the students a useful pose predictor that generalizes well for molecular docking, rather than prompting the students to mimic the teachers.

Ensemble KD proved markedly effective at reducing the cost of large ensembles while still maintaining much of the performance. Distilling the All Default2018 Ensemble, composed of 15 different models, into a single model proved effective for both redocking and crossdocking (Figure S9). This distillation reduces the computational cost of docking from about 123 seconds to about 25 seconds (CPU-only docking), a 5x speedup, with only a small decrease in Top1 percentage on both tasks (Tables 1 and 2). This distillation performance is notably different from the distillation of the basic Default2018 ensembles (`general_default2018`, `redock_default2018`, and `crossdock_default2018`), which is likely due to the diversity of pose ranking performance of the models composing the All Default2018 ensemble (Figure S18). Since "All Default2018" is composed of models trained on different datasets, each of the teachers provided a different expertise during the distillation.

Distillation of the `dense` ensemble into a Default2018 architecture was also effective. The Dense architecture has about twice as many parameters compared to the Default2018 architecture, which, combined with the more compute-demanding dense convolutions, results in a CPU-only docking time of about 458s. Distilling this ensemble into a Default2018 model slightly reduces Top1 pose ranking performance from 73.1% to 65.69% and 41.6% to 35.9% on redocking and crossdocking, respectively. The Default2018 architecture distilled from the `dense` Ensemble has the best performance of any single Default2018 architecture model. This paves the way for future research focusing on the development of efficient molecular docking scoring functions derived from larger architectures. Due to their increased capacity, large models can capture a more comprehensive representation of the protein-ligand complex, thereby achieving higher generalization capabilities. Moreover, deriving a single model from an ensemble of larger models significantly trims docking time, particularly for large-scale docking endeavors.

While ensemble KD exhibits considerable potential for enhancing the efficiency of existing molecular docking CNNs, there is much room for further exploration. One of the key challenges is the current lack of understanding regarding which protein and molecule features contribute most significantly to the distillation process. Additionally, our work focused only on distilling the pose scoring of the CNN models as the KD field largely focuses on the distillation of classification tasks. Future work can explore simultaneously distilling the affinity prediction regression value using newly proposed regression distillation methods.[22] Additionally, KD could be used for the distillation of increasingly large state-of-the-art molecular docking graph neural networks (GNNs)[11] to reduce model computation with minimal performance impact.

Conclusion

While traditional scoring methods are interpretable and efficient CNNs provide a more performative yet computationally expensive alternative for evaluating binding pose quality. Previous work has found that ensembles of CNN models perform better at ranking binding poses, but require longer running times. We applied ensemble KD to condense the knowledge of these ensembles into a single CNN, resulting in shorter running times with higher pose ranking performance than any non-KD trained single model. This efficiency gain is more apparent when the teacher model has a large number of parameters or the ensemble is composed of a diverse set of models. We were able to distill an ensemble of 15 models into a single model, gaining about a 5x docking runtime speedup, with pose ranking performance higher than any other single CNN model with comparable computational cost on both redocking and crossdocking tasks.

Acknowledgement

The authors thank Jonathan King and Emma Flynn for their comments during the preparation of the manuscript.

This work is supported by R35GM140753 from the National Institute of General Medical Sciences.

Data and Software Availability: Training datasets and evaluation datasets are available at `https://github.com/gnina/models/tree/master/data/CrossDocked2020` and `https://github.com/gnina/models/tree/master/data/Gnina1.0`, respectively.

All scripts for KD training and evaluation of the trained models as well as the trained PyTorch model weights are available on our github: `https://github.com/YanjingLiLi/GNINA_Knowledge_Distillation`.

References

(1) Li, J.; Fu, A.; Zhang, L. An overview of scoring functions used for protein–ligand interactions in molecular docking. *Interdisciplinary Sciences: Computational Life Sciences* **2019**, *11*, 320–328.

(2) Huang, N.; Kalyanaraman, C.; Bernacki, K.; Jacobson, M. P. Molecular mechanics methods for predicting protein–ligand binding. *Physical Chemistry Chemical Physics* **2006**, *8*, 5166–5177.

(3) Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems* **2021**,

(4) Wallach, I.; Dzamba, M.; Heifets, A. AtomNet: a deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *arXiv preprint arXiv:1510.02855* **2015**,

(5) Feinberg, E. N.; Sur, D.; Wu, Z.; Husic, B. E.; Mai, H.; Li, Y.; Sun, S.; Yang, J.; Ramsundar, B.; Pande, V. S. PotentialNet for molecular property prediction. *ACS central science* **2018**, *4*, 1520–1530.

(6) McNutt, A. T.; Francoeur, P.; Aggarwal, R.; Masuda, T.; Meli, R.; Ragoza, M.; Sunseri, J.; Koes, D. R. GNINA 1.0: molecular docking with deep learning. *Journal of cheminformatics* **2021**, *13*, 1–20.

(7) Francoeur, P. G.; Masuda, T.; Sunseri, J.; Jia, A.; Iovanisci, R. B.; Snyder, I.; Koes, D. R. Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of chemical information and modeling* **2020**, *60*, 4200–4215.

(8) Stärk, H.; Ganea, O.; Pattanaik, L.; Barzilay, R.; Jaakkola, T. Equibind: Geometric deep learning for drug binding structure prediction. International Conference on Machine Learning. 2022; pp 20503–20521.

(9) Lu, W.; Wu, Q.; Zhang, J.; Rao, J.; Li, C.; Zheng, S. Tankbind: Trigonometry-aware neural networks for drug-protein binding structure prediction. *Advances in Neural Information Processing Systems* **2022**,

(10) Corso, G.; Stärk, H.; Jing, B.; Barzilay, R.; Jaakkola, T. DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking. International Conference on Learning Representations (ICLR). 2023.

(11) Corso, G.; Deng, A.; Polizzi, N.; Barzilay, R.; Jaakkola, T. Deep Confident Steps to New Pockets: Strategies for Docking Generalization. International Conference on Learning Representations (ICLR). 2024.

(12) Bender, B. J.; Gahbauer, S.; Luttens, A.; Lyu, J.; Webb, C. M.; Stein, R. M.; Fink, E. A.; Balius, T. E.; Carlsson, J.; Irwin, J. J., et al. A practical guide to large-scale docking. *Nature protocols* **2021**, *16*, 4799–4832.

(13) Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* **2015**,

(14) Allen-Zhu, Z.; Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816* **2020**,

(15) Tian, Y.; Krishnan, D.; Isola, P. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699* **2019**,

(16) Wang, R.; Fang, X.; Lu, Y.; Wang, S. The PDBbind database: Collection of binding affinities for protein- ligand complexes with known three-dimensional structures. *Journal of medicinal chemistry* **2004**, *47*, 2977–2980.

(17) Liu, Z.; Su, M.; Han, L.; Liu, J.; Yang, Q.; Li, Y.; Wang, R. Forging the basis for developing protein–ligand interaction scoring functions. *Accounts of chemical research* **2017**, *50*, 302–309.

(18) Wierbowski, S. D.; Wingert, B. M.; Zheng, J.; Camacho, C. J. Cross-docking benchmark for automated pose and ranking prediction of ligand binding. *Protein Science* **2020**, *29*, 298–305.

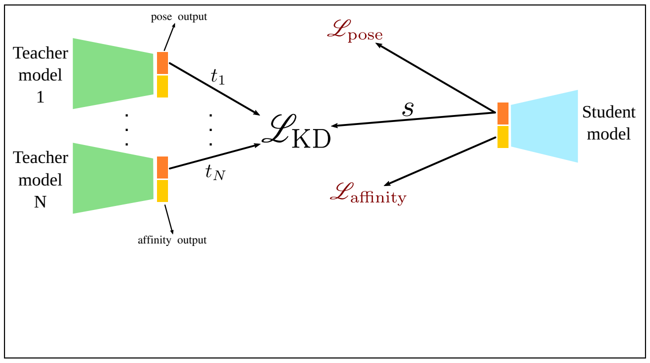
(19) Su, M.; Yang, Q.; Du, Y.; Feng, G.; Liu, Z.; Li, Y.; Wang, R. Comparative assessment of scoring functions: the CASF-2016 update. *Journal of chemical information and modeling* **2018**, *59*, 895–913.

(20) Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K. Q. Densely connected convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017; pp 4700–4708.

(21) Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. Caffe: Convolutional architecture for fast feature embedding. Proceedings of the 22nd ACM international conference on Multimedia. 2014; pp 675–678.

(22) Saputra, M. R. U.; De Gusmao, P. P.; Almalioglu, Y.; Markham, A.; Trigoni, N. Distilling knowledge from a deep pose regressor network. Proceedings of the IEEE/CVF international conference on computer vision. 2019; pp 263–272.

TOC Graphic


pose output
Teacher model 1
Teacher model N
affinity output
t_1
t_N
$\mathscr{L}_{\text{KD}}$
s
$\mathscr{L}_{\text{pose}}$
$\mathscr{L}_{\text{affinity}}$
Student model