Deep Docking, Part 2: An Amplified DDU Platform for Ultra-Large Virtual Screening

Supplementary Information

Mohit Pandey[1,2], Tanvir Sajed[1,2], Ivan Semenov[1,2], Renfei Zhang[3], Fuqiang Ban[1,2], Ekaterina Manskaia[1,2], Martin Ester[3], Artem Cherkasov[1,2]*

[1] *Vancouver Prostate Centre, University of British Columbia, Vancouver, British Columbia, Canada*
[2]*Faculty of Medicine, University of British Columbia, Vancouver, BC, Canada*
[3]*School of Computing Science, Simon Fraser University, Burnaby, BC, Canada*

*Corresponding author: acherkasov@prostatecentre.com

S1. Architecture Design

Feed Forward Neural Networks

FFNNs, also known as multi-layer perceptron (MLPs), are a type of neural network that models complex, non-linear relationships in data, using one or more hidden layers. Each hidden layer receives its inputs from the previous layer and applies linear transformations characterized by learnable weight matrices. These weight matrices are randomly initialized at first and then iteratively adjusted through gradient descent to improve the model's ability to map inputs to outputs. After the transformation, activation functions are applied element-wise to the data, enabling the network to learn non-linear patterns. Mathematically, the transformation at layer l can be expressed as:

$$Z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)} \quad (1)$$

$$a^{(l)} = f(Z^{(l)}) \quad (2)$$

where $W^{(l)}$ is the weight matrix at layer l, $b^{(l)}$ is the bias vector, $a^{(l-1)}$ represents the activations from the previous layer, $Z^{(l)}$ is the weighted sum of inputs at layer l, adjusted by biases, and f is a non-linear activation function such as rectified linear unit (ReLU) or sigmoid.

We implemented three FFNNs in PyTorch, each with increasing architectural complexity, designed to enhance the performance of the docking score classifier.

MLP3k: The MLP3K consists of an input projection layer that maps the input to 2000 nodes, three fully connected hidden layers with 1000, 512, and 32 nodes, followed by an output layer of size 2 for binary classification. Each non-output layer is activated using the ReLU function, and dropout (p = 0.5) is applied after the first and last hidden layers to prevent overfitting.

MLP6k: The MLP6K consists of an input projection layer that maps the input to 6000 nodes, followed by six fully connected hidden layers with 6000, 4000, 3000, 1500, 512, and 128 nodes, and a final output layer of size 2 for binary classification. Each non-output layer undergoes batch normalization before activation using the ReLU function. Dropout (p = 0.5) is applied after the input layer, and after the first and last hidden layers. A residual connection is introduced to connect the input layer and the first hidden layer.

MLPTx: The MLPTx consists of an input projection layer that maps the input to a hidden layer with 2048 nodes, followed by batch normalization and ReLU activation. The learned projection is encoded with two encoder blocks, each consisting of multi-head self-attention with 8 heads and dropout (p=0.1), a feedforward layer, and layer normalization. The FFNN network expands the feature dimension to 8192 nodes using a fully connected layer with Gaussian Error Linear Unit (GELU) activation and dropout (p = 0.1), before being compressed back to 2048 nodes through fully connected and dropout layers. The input to the encoder block is added to the output of the feedforward layer through skip connection, and this sum is passed through layer normalization. The output is then refined by a highway network, which applies a gated mechanism to selectively combine the output from the previous block with its transformed representation. The transformed input is passed through a ReLU activation, while the gate is computed using a sigmoid function. Mathematically, this is given by:

$$y = T \odot H(x) + (1 - T) \odot x \tag{3}$$

where x is the input feature representation, $H(x)$ is the transformed representation given by $H(x) = ReLU(W_H x + b_H)$, T is the transformation gate computed using sigmoid activation function $\sigma(W_T x + b_T)$, and $\odot$ is the element-wise multiplication.

This weighted combination of features is passed through three hidden layers with 1024, 512, and 128 neurons, each followed by batch normalization and ReLU activation. Dropout (p = 0.5) is applied before the final output layer, which consists of two nodes for binary classification.

To estimate classification uncertainties for non-greedy acquisition methods, we utilized Monte Carlo dropout with 10 forward passes during inference across all three architectures.

Graph Neural Networks

Message Passing Neural Networks (MPNNs): MPNNs are well-suited for constructing feature representations of graph-structured data, such as molecules. To capture structural and topological relationships, MPNNs operate in two phases: message passing and readout. During the message passing phase, each atom exchanges information with its neighboring atoms and/or bonds, to iteratively update its hidden representation. In the readout phase, the updated atom features are aggregated to generate a graph-level representation of the molecule. Mathematically, this can be summarized as:

$$h_i^{(l)} = \gamma^{(l-1)}(h_i^{(l-1)}, \oplus_{j \in N_i} M^{(l-1)}(h_i^{(l-1)}, h_j^{(l-1)}, e_{ij})) \tag{4}$$

where $h_i^{(l-1)}$ and $h_i^{(l)}$ are the feature representation of node i from the layer $l-1$ and l, e_{ij} is the edge feature between node i and node j, $M^{(l-1)}$ is a message passing function that computes the message from node j to node i, $\oplus$ is a permutation-invariant operation that aggregations messages from neighborhood of node i, and $\gamma^{(l-1)}$ is an update function that integrates the node's current feature $h_i^{(l-1)}$ with its aggregated messages.

In our model, each node starts with an initial feature vector of size 58, while each edge is represented by a six-dimensional feature vector. The features are projected into a hidden space of dimension 64 before passing through the message-passing layers. Then we updated the node features, and in some cases the edge features using different message passing architectures, including Graph Convolutional Network (GCN), Graph Isomorphism Network (GIN), Graph Isomorphism Network with Edge Features (GINE), and Graph Attention Network (GAT)[1–4] . For models using GAT, four attention heads are used to compute attention-weighted node embeddings. Each message-passing layer also includes dropout (p = 0.2), and an update function ReLU. Additionally, we aggregated node features using either summation, averaging, max pooling, or attention pooling. For the multi-headed attention pooling mechanism, we assigned atom-wise attention weight to each atom. The graph-level representation Z_G is computed as:

$$Z_G = \sum_i \alpha_i h_i^{(L)} \tag{5}$$

where α_i is the attention weight of the i^{th} node obtained through softmax normalization:

$$\alpha_i = softmax(\frac{1}{\sqrt{d}} 1 (W h_i^{(L)})^T) \tag{6}$$

where 1 is a d-dimensional seed vector with all entries set to 1, $W \in R^{dxd}$ is a learnable weight matrix. This aggregated information is then passed through a fully connected layer, which maps it to an output node. When specified, a sigmoid activation function is applied to constrain the outputs to the range (0,1).

To enhance docking score prediction in AL-based virtual screening pipelines, we systematically benchmark ten state-of-the-art DL architectures across 12 protein targets from four major drug-target families that were used in the original DD. This panel includes nuclear receptors, represented by androgen receptor (AR), estrogen receptor-alpha (ERα), and peroxisome proliferator-activated receptor gamma (PPARγ); kinases, including calcium/calmodulin-dependent protein kinase kinase 2 (CAMKK2), cyclin-dependent kinase 6 (CDK6), and vascular endothelial growth factor receptor 2 (VEGFR2); G protein-coupled receptors (GPCRs), such as adenosine A2A receptor (ADORA2A), thromboxane A2 receptor (TBXA2R), angiotensin II receptor type 1 (AT1R); and ion channels, represented by Nav1.7 sodium channel (Nav1.7), Gloeobacter ligand-gated ion channel (GLIC), and gamma-aminobutyric acid type A receptor (GABAA). The models span three

major representation paradigms: (i) multi-layer perceptrons (MLPs) trained on molecular fingerprints (mlp3K, mlp6K, mlpTx), (ii) MLLMs based on SMILES inputs including MoLFormer, and its architectural variants with Attention Highway (MoLFormer-AH) and Dense Residual connections (MoLFormer-DR), and (iii) graph neural networks (GNNs) that capture structural connectivity via message passing including Graph Convolutional Networks (GCN), Graph Isomorphism Networks (GIN, GINE), and Graph Attention Networks (GAT). To ensure comparability, we follow standard docking protocols from recent DL studies, using QuickVina2 with receptor grids and box sizes matched to prior settings.

Throughout this study, we refer to compounds with favorable docking scores as actives, while those with poor docking scores are labeled as inactives. Given the inherent class imbalance in docking-based virtual screening, where compounds meeting a favorable score threshold (virtual hits) represent a small minority relative to the broader compound pool (virtual non-hits), we emphasize the F1-score as the primary evaluation metric, complementing ROC-AUC, precision, and recall (see Supplementary Figure 1).

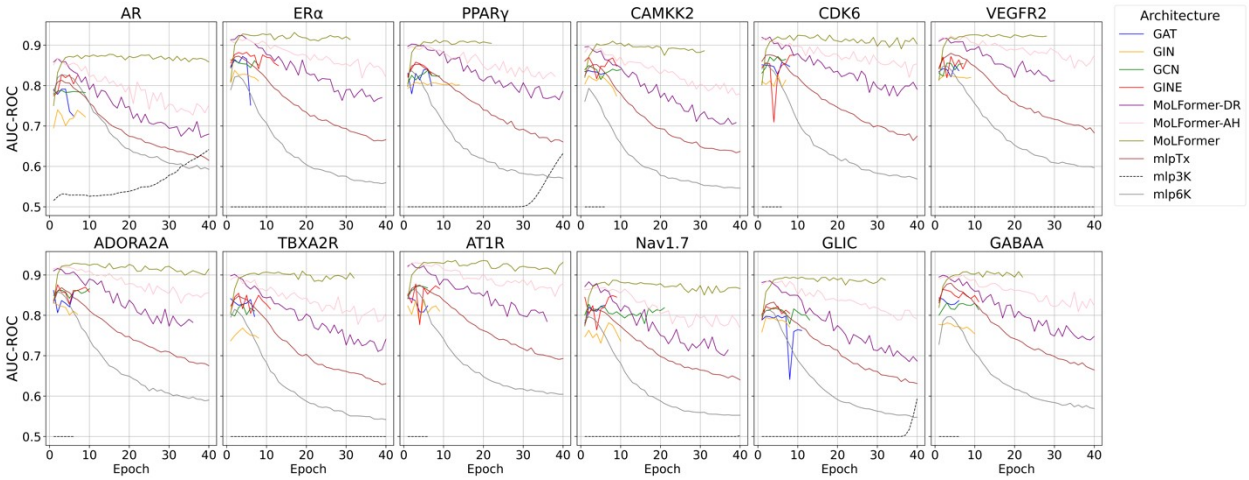
Our experiments show that the mlpTx, mlp6K, MoLFormer-DR and MoLFormer-AH consistently rank among the top-performing architectures. In particular, MoLFormer-DR achieve the highest F1-score in eight out of twelve targets, followed by MoLFormer-AH, mlp6k, and mlpTx. Although mlpTx ranks among the top four architectures, it never achieves the highest F1-score and is generally inferior in quality to the other three architectures. . Further analysis of F1-score trends over training epochs reveals that mlp6K exhibits a sharp increase in early iterations before experiencing a decline, suggesting potential overfitting. In contrast, mlpTx, MoLFormer-DR and MoLFormer-AH demonstrate a more stable training trajectory, with F1-scores improving

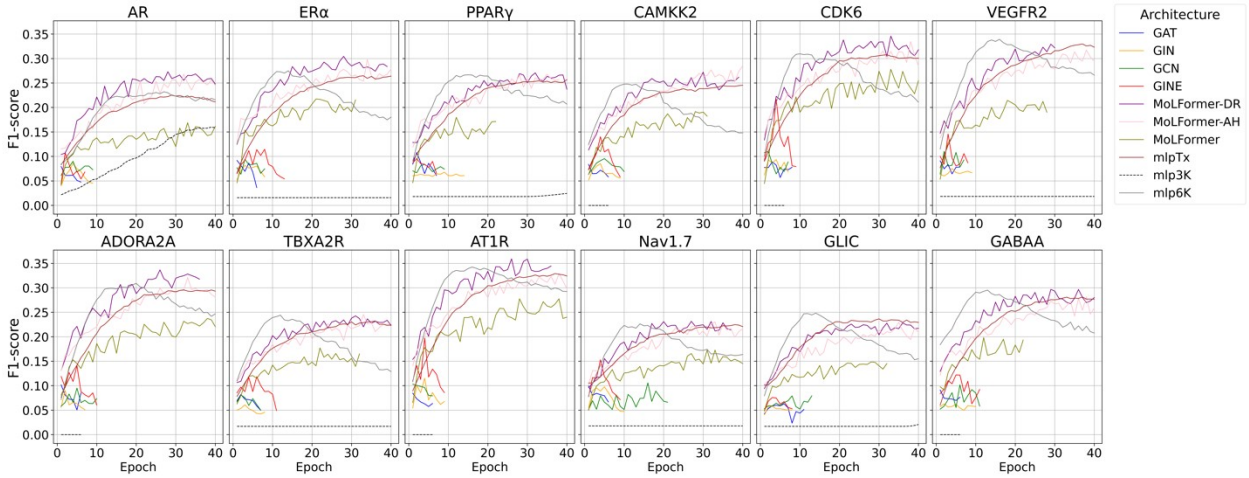
gradually without abrupt declines. This stability suggests that these architectures generalize better to unseen molecular data.

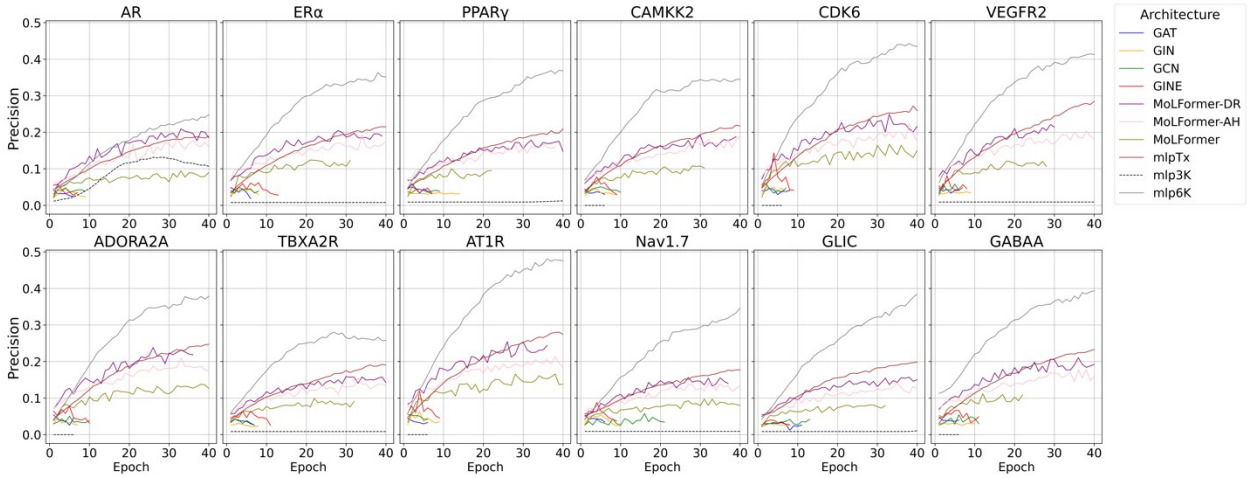
A closer look at precision and recall trade-offs shows that MoLFormer DR and MoLFormer AH deliver comparable overall F1 scores. MoLFormer DR provides slightly higher precision, reflecting improved selectivity and reliability of predicted actives, while MoLFormer AH achieves marginally higher recall. Given that virtual screening pipelines often prioritize precision to reduce false positives and computational overhead in downstream docking, MoLFormer DR offers a more practical balance between accuracy and efficiency.

Beyond predictive accuracy, computational efficiency is a crucial factor in large-scale molecular docking applications. It is important to note, that MoLFormer-DR and MoLFormer-AH, as LLM-based architectures, can directly process SMILES representations, bypassing the need for precomputed molecular fingerprints. This capability eliminates the significant storage overhead associated with traditional fingerprint-based methods like mlp6K and mlpTx, where computing and storing Morgan fingerprints for billions of molecules can require terabytes of disk space. Consequently, MoLFormer-DR demonstrates the highest classification performance and offers superior efficiency for large scale virtual screening campaigns, effectively handling vast chemical libraries where storage and computational resources are critical constraints.

It is noteworthy that the metric curves shown in Figure S1-S3 are generated by evaluating the model on the frozen test set after each training epoch in inference mode (no parameter updates). Early stopping and model selection are based solely on validation set performance; test set metrics are recorded for monitoring purposes only and do not influence any training decisions.


AR
ERα
PPARγ
CAMKK2
CDK6
VEGFR2
ADORA2A
TBXA2R
AT1R
Nav1.7
GLIC
GABAA
AUC-ROC
Epoch
Architecture
GAT
GIN
GCN
GINE
MoLFormer-DR
MoLFormer-AH
MoLFormer
mlpTx
mlp3K
mlp6K


AR
ERα
PPARγ
CAMKK2
CDK6
VEGFR2
ADORA2A
TBXA2R
AT1R
Nav1.7
GLIC
GABAA
F1-score
Epoch
Architecture
GAT
GIN
GCN
GINE
MoLFormer-DR
MoLFormer-AH
MoLFormer
mlpTx
mlp3K
mlp6K


AR
ERα
PPARγ
CAMKK2
CDK6
VEGFR2
ADORA2A
TBXA2R
AT1R
Nav1.7
GLIC
GABAA
Precision
Epoch
Architecture
GAT
GIN
GCN
GINE
MoLFormer-DR
MoLFormer-AH
MoLFormer
mlpTx
mlp3K
mlp6K

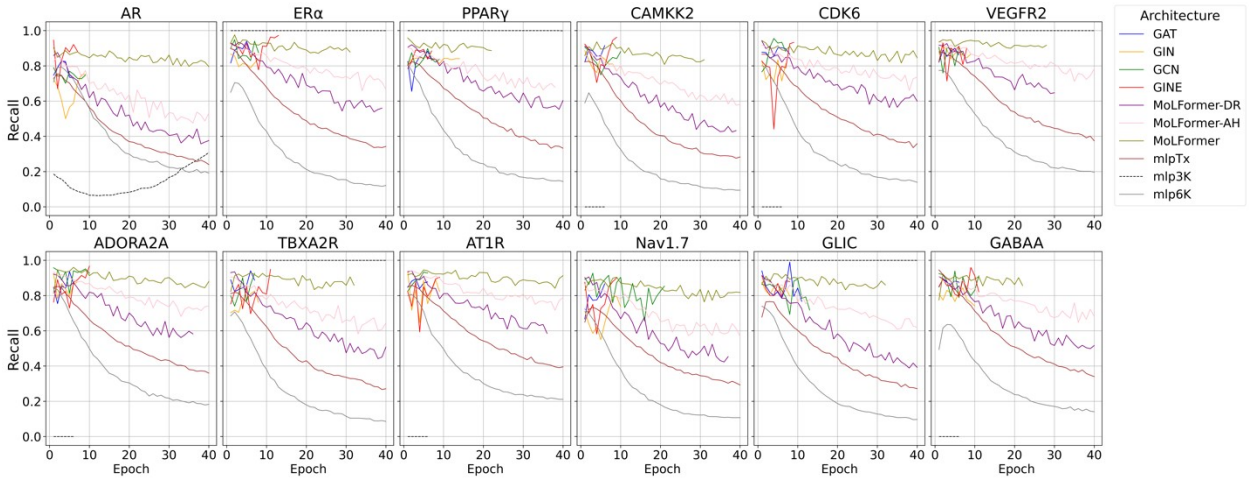


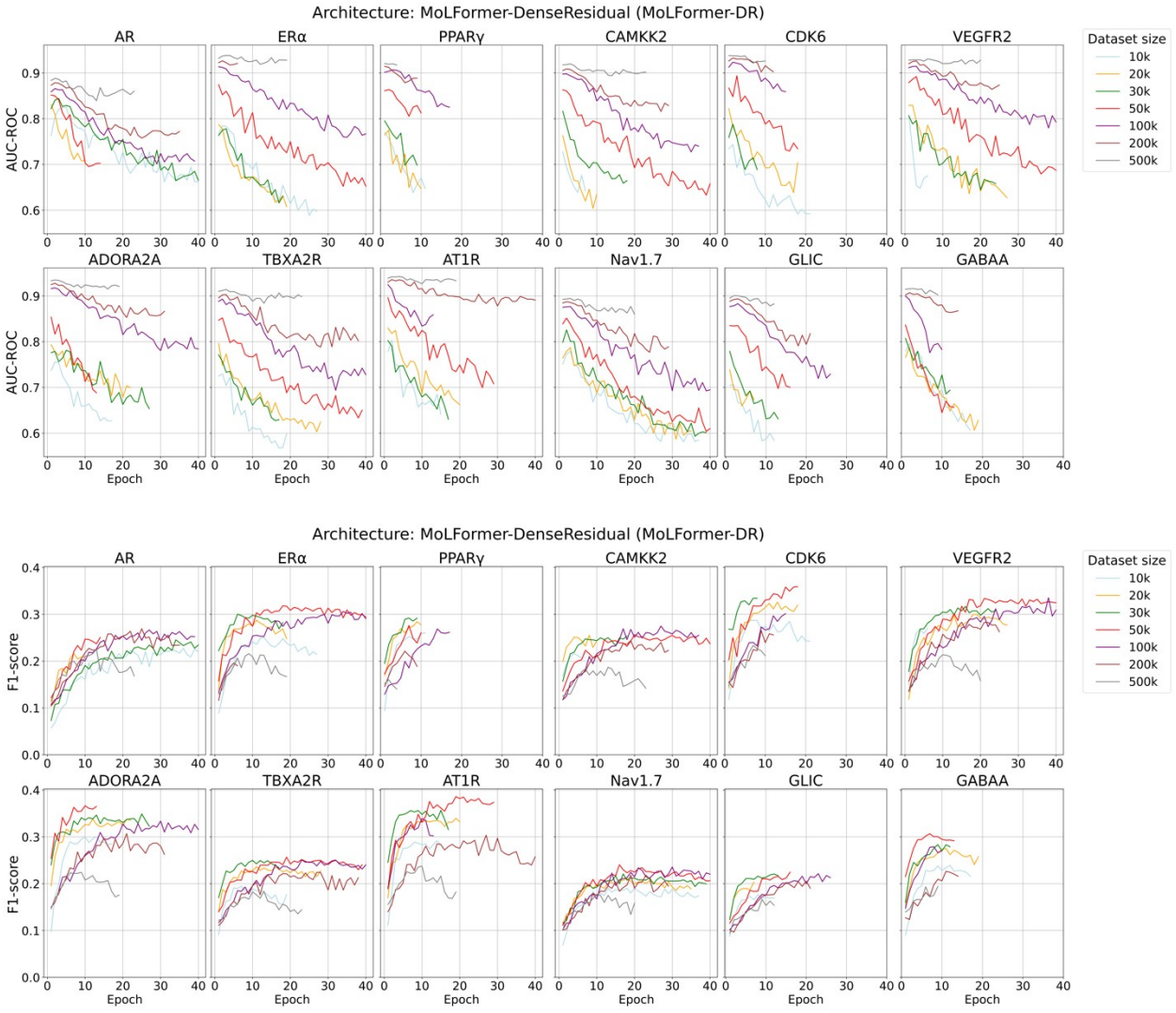
Figure S1: Architectures Comparison. All evaluation metrics reported in this section (F1-score, ROC-AUC, precision, and recall) are computed on the frozen held-out test set of 100,000 randomly sampled compounds from the ZINC22 one-million subset. This test set is fixed prior to any model training and is never used for model selection, early stopping, or hyperparameter tuning.

S2. Effect of dataset size

The size of the training dataset plays a crucial role in determining both the computational feasibility and predictive accuracy of DL models in molecular docking. To systematically evaluate its impact, we conducted 70 controlled experiments across two top-performing architectures—MolFormer-AH, and MoLFormer-DR. Each model has been trained on incrementally increasing subsets of docked molecules comprising 10K, 20K, 30K, 50K, 100K, 200K, and 500K compounds. Performance has then been measured using F1-score, precision, recall, and ROC-AUC metrics (Supplementary Figure 2, 3).

As the result,both MoLFormer-AH and MoLFormer-DR models exhibit a non-monotonic trend, with peak F1-scores observed for intermediate training dataset sizes (~50K molecules), with a surprising decline at larger training dataset sizes. This performance drop may be attributed to overfitting or difficulties in extracting meaningful representations from excessively large and diverse training sets. Since MoLFormer-AH and MoLFormer-DR process raw SMILES strings,

they are therefore more sensitive to representation learning challenges when trained on large, chemically diverse datasets. The decline in performance at larger dataset sizes suggests that beyond a certain threshold, the models struggle to extract useful patterns from an increasingly complex chemical landscape.



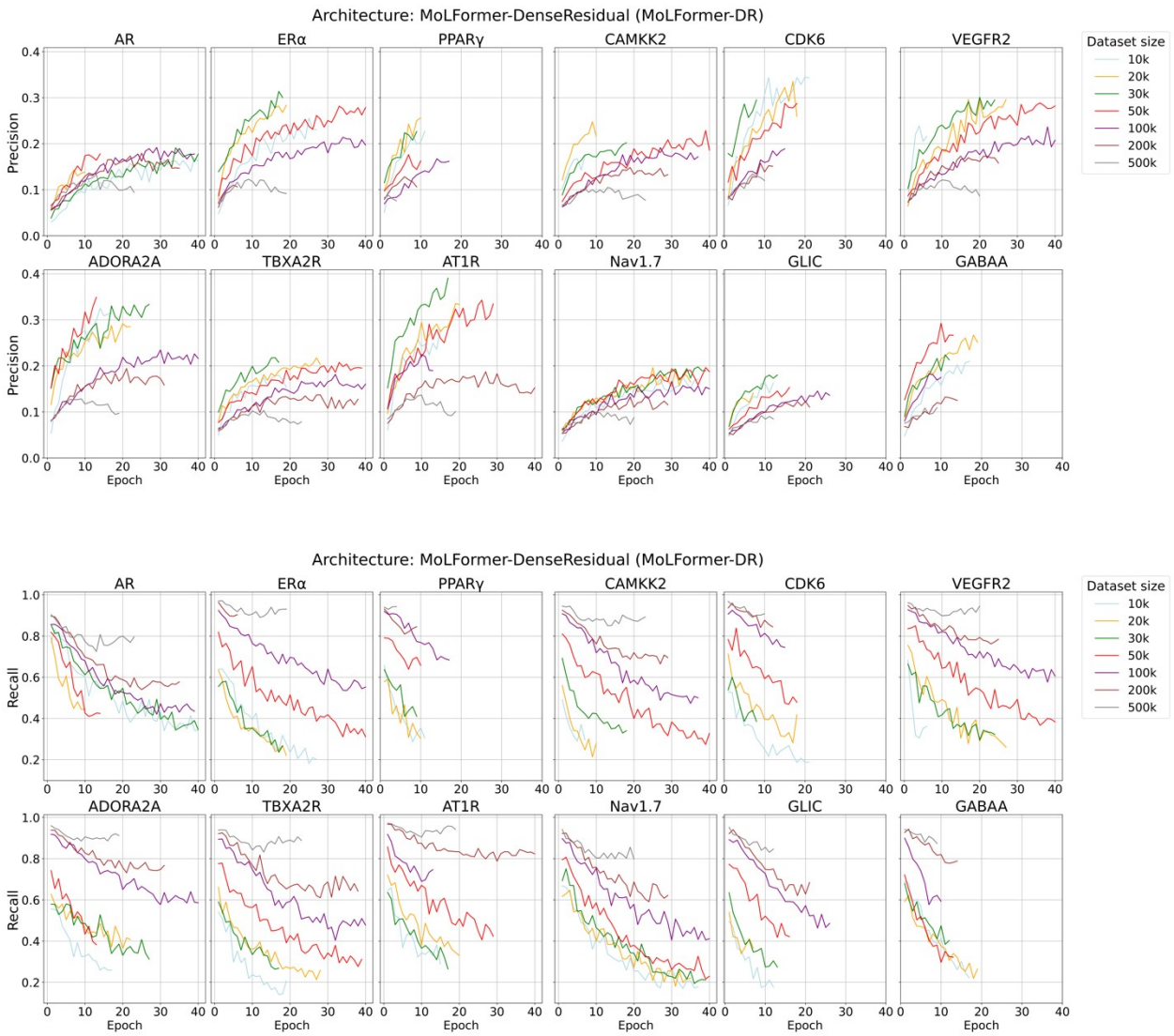
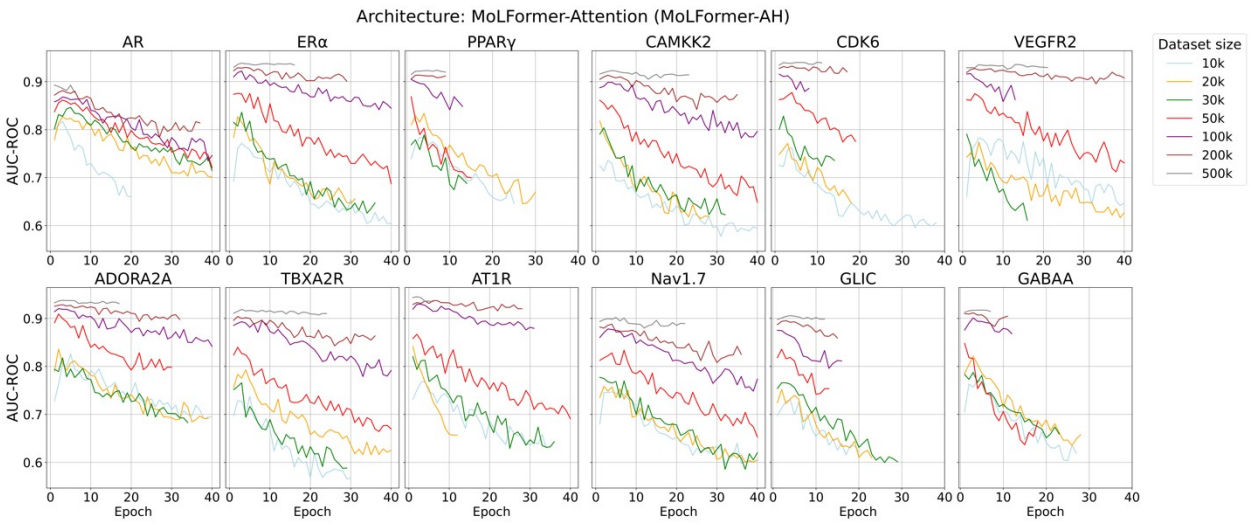
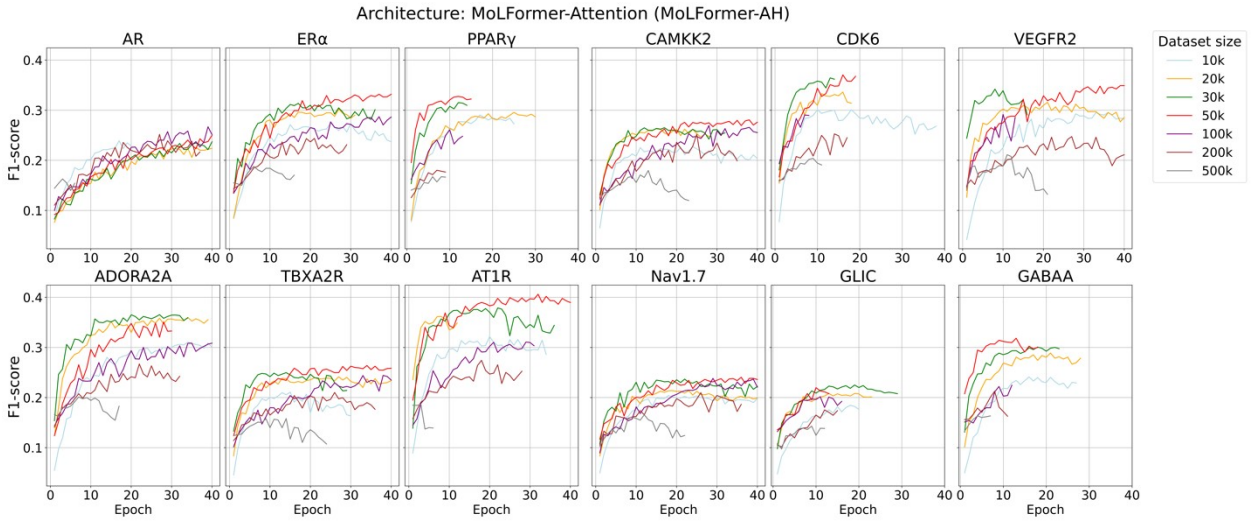
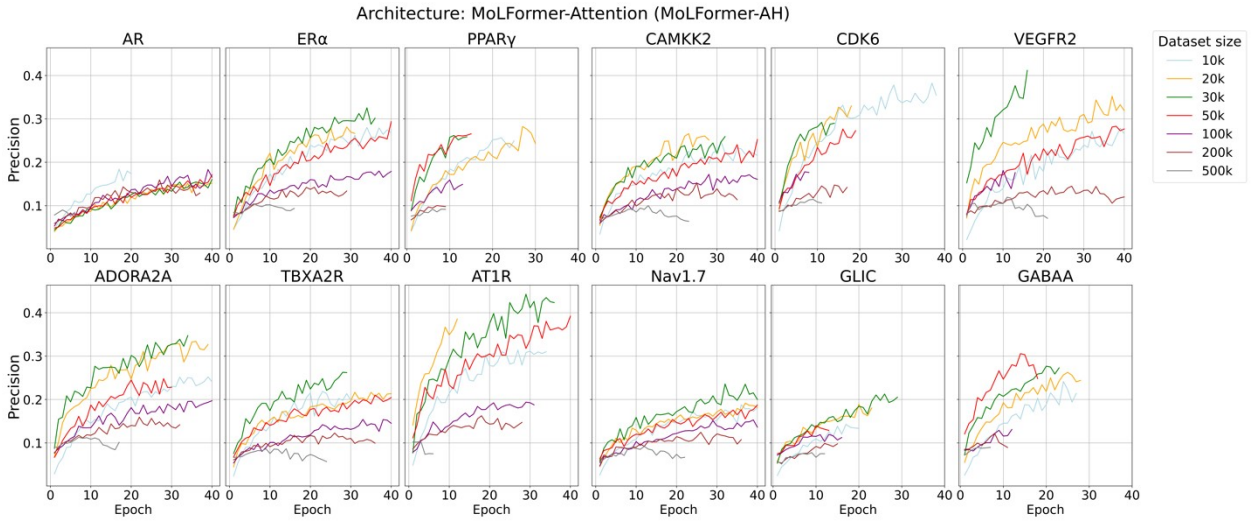


Figure S2. Dataset size comparison for MoLFormer-DR architecture. All evaluation metrics reported in this section (F1-score, ROC-AUC, precision, and recall) are computed on the frozen held-out test set of 100,000 randomly sampled compounds from the ZINC22 one-million subset. This test set is fixed prior to any model training and is never used for model selection, early stopping, or hyperparameter tuning.


Architecture: MoLFormer-Attention (MoLFormer-AH)
AR
ERα
PPARγ
CAMKK2
CDK6
VEGFR2
ADORA2A
TBXA2R
AT1R
Nav1.7
GLIC
GABAA
AUC-ROC
Epoch
Dataset size
10k
20k
30k
50k
100k
200k
500k


Architecture: MoLFormer-Attention (MoLFormer-AH)
AR
ERα
PPARγ
CAMKK2
CDK6
VEGFR2
ADORA2A
TBXA2R
AT1R
Nav1.7
GLIC
GABAA
F1-score
Epoch
Dataset size
10k
20k
30k
50k
100k
200k
500k


Architecture: MoLFormer-Attention (MoLFormer-AH)
AR
ERα
PPARγ
CAMKK2
CDK6
VEGFR2
ADORA2A
TBXA2R
AT1R
Nav1.7
GLIC
GABAA
Precision
Epoch
Dataset size
10k
20k
30k
50k
100k
200k
500k

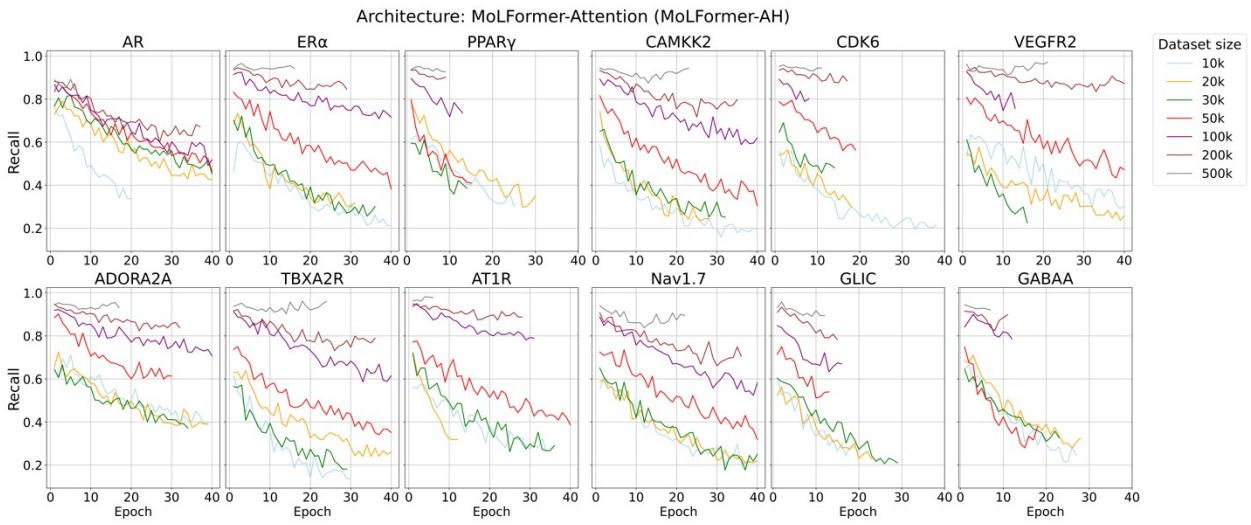


Figure S3. Dataset size comparison for MoLFormer-AH architecture. All evaluation metrics reported in this section (F1-score, ROC-AUC, precision, and recall) are computed on the frozen held-out test set of 100,000 randomly sampled compounds from the ZINC22 one-million subset. This test set is fixed prior to any model training and is never used for model selection, early stopping, or hyperparameter tuning.

S3. Active Learning

Active Learning (AL) improves surrogate model performance while minimizing docking costs by iteratively selecting informative molecules using an acquisition function. We investigated the impact of different acquisition functions, binary thresholding scheme to convert docking scores to binary labels for training ML classification models, and the role of systematically reducing the database of available molecules for acquisition at each AL iteration on the performance of the overall AL pipeline. Using twelve drug targets and one million molecules from the Zinc22 Set, we run 30 AL iterations. MoLFormer-DR is initially trained on 20K (train), 50K (val), and 100K (test) samples. A one million–compound REAL Diversity subset balances broad chemical diversity (drawn from Enamine's multibillion REAL Space) with runtime feasibility for tens of ablation studies, each with 30 AL iterations. In each iteration, 100 top-scoring molecules are docked and augmented to the

training set. Due to class imbalance from strict docking thresholds, we focus on test F1-score as the primary evaluation metric.

Binary Thresholding

We evaluated the impact of two docking score thresholding strategies on the performance of AL across twelve drug targets. While using the fixed cutoff approach, the docking score threshold is set at the 1% quantile of the initial training dataset and remains unchanged throughout the process. In contrast, the dynamic cutoff strategy recalculates this threshold at each iteration based on the 1% quantile of the current training set, ensuring that the cutoff can only become stricter (more negative) or remain unchanged (Figure S4).

Our results indicate that the fixed cutoff strategy consistently outperforms the dynamic cutoff approach in terms of F1 metric (Figure 5a). Under the fixed cutoff, F1 metrics generally improve over successive AL iterations, aligning with the expected behavior of AL: as more informative molecules are selected, the model progressively refines its decision boundary. In contrast, with a dynamic cutoff, F1 metrics peak at the first iteration and subsequently decline as the cutoff becomes increasingly stringent. This deterioration continues until the cutoff stabilizes, at which point the model begins to recover but never reaches the performance of the fixed cutoff strategy.

The poorer performance of the dynamic cutoff strategy is primarily driven by concept drift rather than by a change in class balance. The ratio of active to inactive compounds in the training set remains approximately constant at 1 percent to 99 percent across iterations, since the threshold is always defined as the 1 percent quantile of the current training set. However, the absolute docking score corresponding to this cutoff becomes progressively more negative as high scoring molecules accumulate in the training set. As a result, molecules labeled as active in earlier iterations may fall below the updated cutoff in later iterations, which effectively changes the learning objective over

time. In contrast, the held out test set is consistently evaluated using the original threshold defined at iteration 1. This creates a progressive misalignment between the evolving training labels and the fixed test labels. This misalignment is the primary cause of the observed decline in F1 under the dynamic threshold strategy.

Furthermore, previously selected actives may no longer align with the newly defined actives in subsequent iterations, disrupting the model's ability to generalize. This progressive misalignment between training data and evolving selection criteria leads to the observed decline in classification performance.

A dynamic cutoff might, in principle, enable a more refined search for high-scoring compounds; however, in our experiments a fixed cutoff yields stronger predictive performance and a broader diversity of high-scoring candidates

Database Management

We evaluate the impact of two database management strategies on AL performance across the twelve drug targets. In the inactive retention strategy, only docked molecules are removed from the unlabeled molecular pool at each iteration, while in the inactive removal strategy, both docked and predicted inactive molecules are removed. To isolate the effects of these strategies, we use a dynamic docking score cutoff and restrict our analysis to greedy acquisition, aligning with the original DD framework.

Across all targets, both strategies exhibit similar early-stage F1 performance, but under a dynamic cutoff F1 typically peaks in the first iteration and declines thereafter as the threshold tightens (Figure 5b). *The dynamic cutoff introduces concept drift over time: as the score cutoff shifts toward more negative values, the training label assignments diverge progressively from the fixed test set threshold, making consistent model refinement increasingly difficult regardless of database*

management strategy. However, the inactive retention strategy generally maintains a higher F1 score (and a slower decline) over multiple targets than inactive removal, likely because retaining predicted virtual inactives preserves chemical diversity and sustained exploration. That said, improvements in F1 score are not consistent on every target, and overall performance is broadly comparable. Despite a slight reduction in peak performance, the inactive removal strategy significantly accelerates the AL process by rapidly reducing the size of the unlabeled pool. Across all twelve targets, the pool contracts by over 99% at the second iteration, as most predicted inactives are filtered out early in the process (Figure S5). This rapid depletion slows in later iterations but eventually leads to pool sizes falling below the acquisition batch size (fewer than 100 molecules), resulting in early termination of AL runs.

Our findings align with previous DD studies, which demonstrated that greedy acquisition with inactive removal offers computational efficiency while maintaining comparable predictive performance to inactive retention. However, our results highlight a critical trade-off: while inactive removal reduces computational cost and is well-suited for ultra-large molecular libraries (e.g., trillion-scale virtual screens), it also constrains the chemical space available for model refinement. In contrast, inactive retention maintains a more diverse molecular pool, which can improve predictive performance but demands significantly greater computational resources. The choice between these strategies should be guided by dataset size, available computational power, and the need for chemical diversity in AL-driven VS.

Acquisition Function

To systematically evaluate the impact of acquisition functions in AL for molecular docking, we analyze their performance across the twelve drug targets under different database management strategies (retention vs. removal) and docking score cutoff paradigms (fixed vs. dynamic). The

acquisition functions tested include greedy, random, Bayesian Active Learning by Disagreement (BALD), Upper Confidence Bound (UCB), uncertainty-based (UNC), margin, least confidence, and entropy-based sampling. Since users typically select the AL iteration that yields the highest predictive performance, we rank the acquisition functions based on their peak F1 metric at any iteration rather than restricting the assessment to the final iteration.

Across both fixed-cutoff regimes (database retention and removal), Greedy consistently attains the highest or near-highest F1 on most targets (Figure S6 and Figure S7). Greedy preferentially samples the model's top-scoring molecules, yielding a larger fraction of true positives and boosting precision (and thus F1) early and persistently. Uncertainty-driven criteria (BALD, Least Confidence, Entropy) are competitive on a subset of targets and iteration while BALD, in particular, appears in the top three for some targets under fixed cutoff and retention strategy (Figure S7). Under the fixed cutoff experiments. The relative performance of most acquisition functions varies considerably across targets and iterations, making it difficult to recommend a single non-greedy strategy as the best.

The performance of all the considered acquisition functions changes dramatically when dynamic docking score cutoffs are introduced, introducing progressive concept drift over successive iterations: as the score cutoff tightens, training labels shift away from the fixed test set threshold, making the classification task progressively more challenging regardless of the acquisition strategy used (Figure 6c and Figure S8). In this setting, the effectiveness of all investigated uncertainty-based acquisition functions declines gradually, with BALD maintaining a top-three ranking for most drug targets under the database removal strategy. Since BALD relies on disagreement across dropout-based ensemble predictions, its performance deteriorates as concept drift across AL iterations causes the training labels to increasingly diverge from the fixed test set criterion, leading to a model that struggles to distinguish informative samples from noise.

Under dynamic cutoff with database removal, random acquisition ranks second worst overall and terminates earlier than most acquisition functions. This outcome underscores the limitations of model-agnostic sampling under progressive concept drift induced by the dynamic threshold. As concept drift accumulates across iterations, predictive signals do degrade, but methods that exploit even imperfect uncertainty still prioritize informative molecules better than uninformed random draws. Consequently, random acquisition, being independent of model predictions, performs poorly and offers limited robustness, serving mainly as a simple baseline under extreme class skew. Interestingly, despite both being non-uncertainty-based strategies, greedy acquisition performs worse than random acquisition under dynamic cutoffs and inactive removal strategy. It ranks worst across all drug targets and terminates earlier than any other acquisition functions. This suggests that as concept drift intensifies, methods that depend on model-driven selection become increasingly unreliable, even when they do not explicitly leverage uncertainty metrics. Greedy acquisition, by strictly selecting the highest-scoring molecules, fails to explore diverse regions of the molecular space, exacerbating the misalignment between shifting training labels and the fixed test threshold and leading to premature model saturation.

When using dynamic cutoffs in conjunction with database reduction, margin emerges as the most stable acquisition function, ranking in the top three acquisition functions for most drug targets (Figure 6c).. Unlike purely exploitative strategies such as greedy selection, margin targets points with the smallest gap between the top two class probabilities, i.e., the most decision-boundary–ambiguous molecules. This boundary focused querying remains effective even as dynamic cutoffs induce concept drift because margin relies on the relative ordering of the two most likely labels rather than the absolute confidence of the full predictive distribution. Most acquisition strategies exhibit highly variable performance across different targets, reinforcing the difficulty of generalizing a

single optimal strategy when concept drift and iterative molecular filtering alter the learning landscape dynamically.

Our findings demonstrate that performance of acquisition functions depends strongly on both the database management strategy and cutoff selection strategy. Identifying a single acquisition function that consistently improves predictive performance is difficult, because the training set class distribution keeps shifting relative to the a priori drawn held-out test set. Ultimately, the choice should be guided by the experimental goal and scale: when screening very large libraries and prioritizing a small set of top candidates, exploitation-heavy policies are preferable, whereas when the aim is to enrich a broader pool of actives for downstream assays, exploration-oriented strategies are more suitable. In practice, when employing a dynamic cutoff with database removal, margin and BALD tend to provide the most stable performance across targets. Conversely, under a fixed cutoff with database retention, greedy is a strong default owing to its consistently high F1.

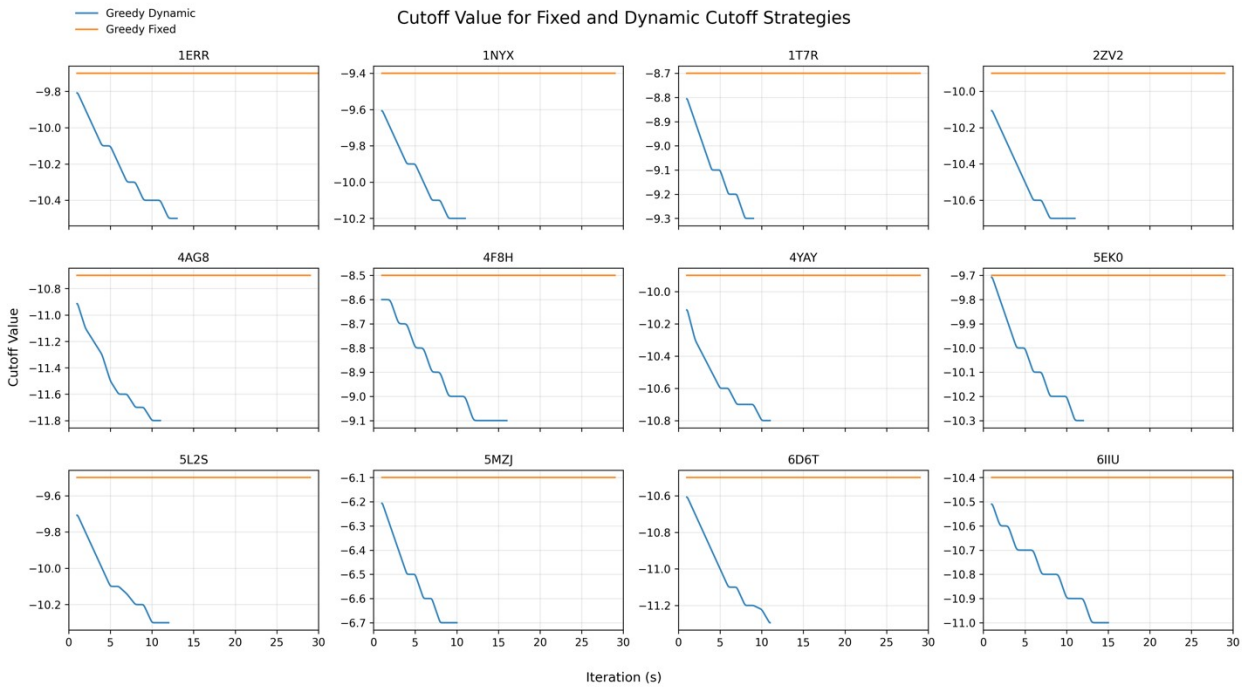


Figure S4 | Fixed cutoff threshold stays constant while Dynamic cutoff progressively tightens each AL round, lowering the threshold over iterations.

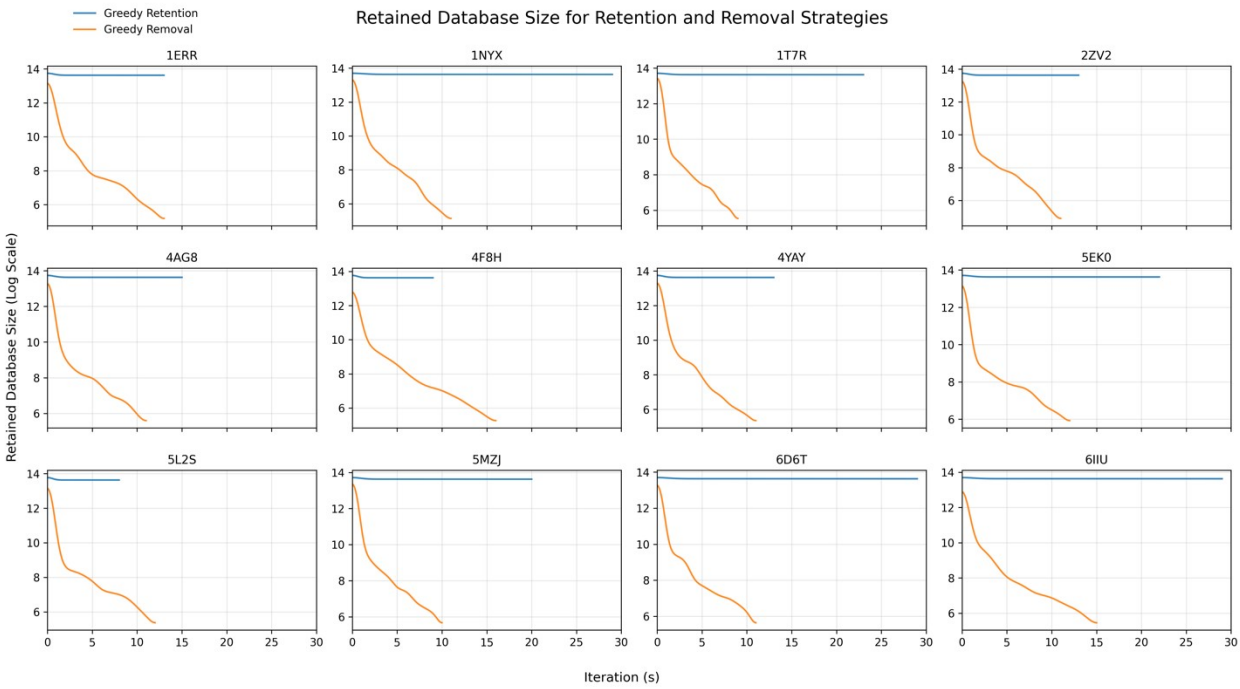


Figure S5. | Retention strategy keeps the unlabeled database pool constant across iterations, while Removal strategy shrinks it by orders of magnitude (note the logarithmic scale).

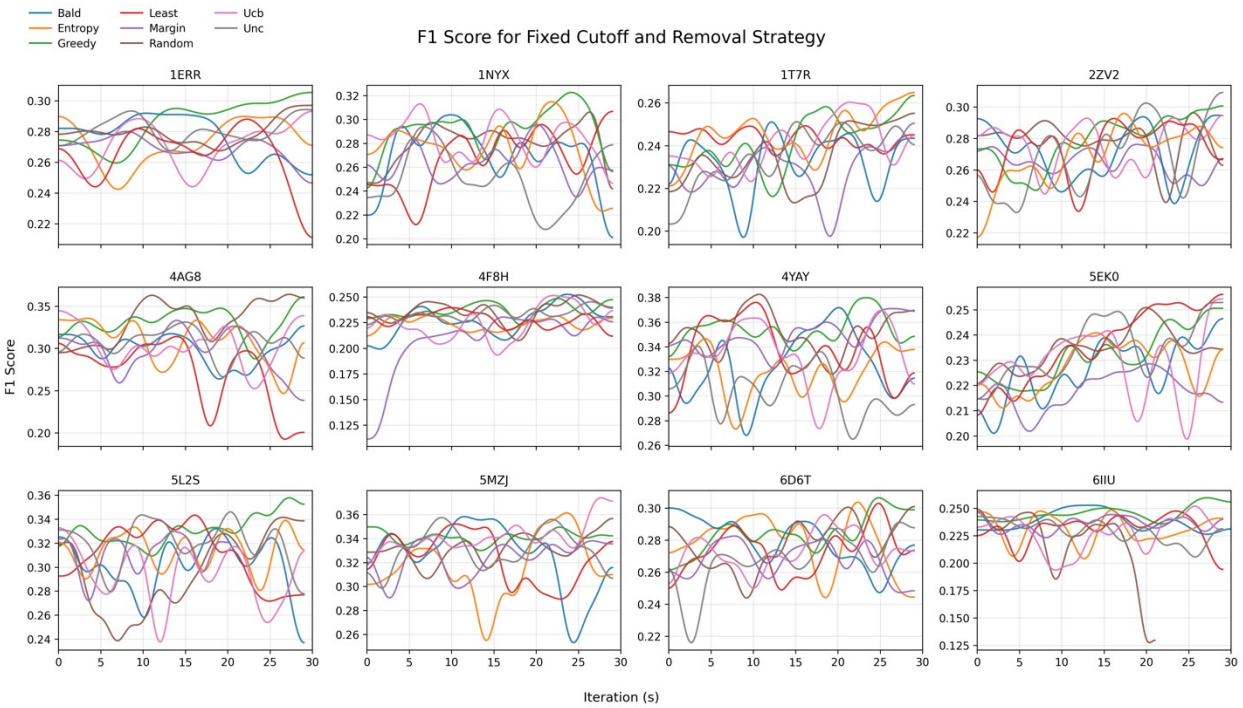


Figure S6. Progression of F1-metric with AL iterations for different acquisition functions for fixed cutoff and removal database management strategy.

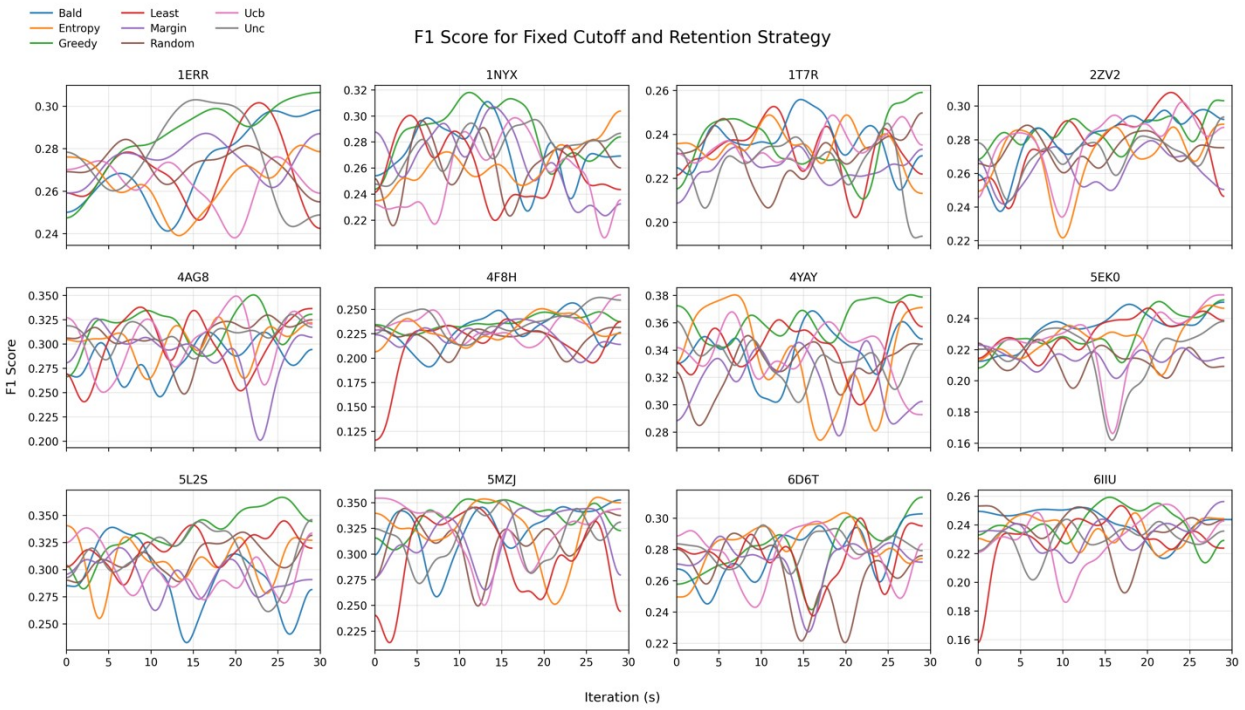


Figure S7. Progression of F1-metric with AL iterations for different acquisition functions for Fixed cutoff and retention database management strategy.

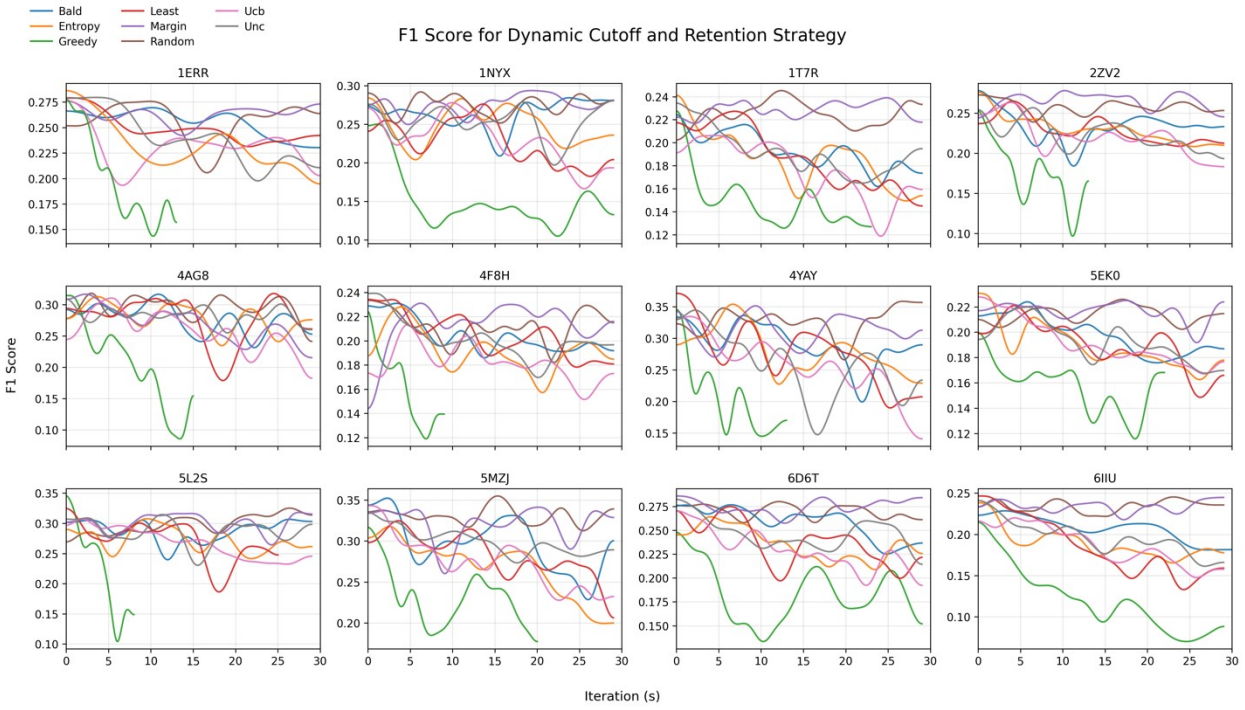


Figure S8**.** Progression of F1-metric with AL iterations for different acquisition functions for Dynamic cutoff and retention database management strategy.

S4. DDU Practical Recommendation

The recommendations above reflect the most consistent findings across the 12-target benchmark panel and the PGK2 prospective campaign. Because no single acquisition function dominates universally across all targets (Fig. 5C), users with prior experimental activity data for their target may use these to validate the acquisition function choice before committing to a full campaign. All parameters listed are configurable in *config/params.yml* without modification of source code.

Parameter	params.yml key	Standard campaign (≤1B compounds)	Ultra-large campaign (>1B compounds)	Notes
Model architecture	model_architecture	molformer_dr	molformer_dr	Best F1 in 8/12 targets; no fingerprint precomputation required; optimal at ~20K training examples
Score threshold mode	fixed_cutoff	true	true	Fixed threshold yields consistently higher F1 than dynamic across all 12 targets (Fig. 5A)
Score threshold value	given_cutoff	1% quantile of initial training set scores	1% quantile of initial training set scores	Set automatically from initial docking; typical range −6.5 to −9.0 kcal/mol depending on target
Acquisition function	acquisition_function	greedy	bald or margin	Greedy is the strongest default under fixed threshold + retention (Figs. S6–S7); BALD/margin are most stable under removal (Fig. S8)
Database management	drop_db	false (retention)	true (removal)	Retention preserves chemical diversity and avoids premature pool depletion; removal required at >1B scale for practical runtime
Initial seed set size	initial_budget	20,000	20,000	Performance gains are marginal beyond 20K–50K samples (Fig. 4B); larger seeds increase docking cost without proportional benefit
AL budget per iteration	al_budget	100	100	Consistent with original DD protocol; sufficient for progressive model refinement
Number of iterations	n_iterations	30	5	Under fixed threshold/retention, F1 improves monotonically — 10 iterations is safe and well-supported; at ultra-large scale, 5 iterations balances enrichment quality against compute budget (demonstrated in PGK2 campaign)
Test set monitoring	test_rand_samples	true (benchmarking) / false (prospective)	false	Enable for transparent performance reporting; disable in production runs to return all sampled molecules to the AL pool
Final docking of top hits	final_docking	true	true	Recommended to obtain confirmed docking poses for top-K predicted actives at campaign end
Final docking top-K	final_docking_top_k	200–1000 (user-defined)	200–1000 (user-defined)	Adjust based on downstream assay capacity
Compute mode	mode	local (1–8 GPUs)	slurm (≥40 GPUs)	Single GPU sufficient for ≤1B libraries (~20 min/iteration); multi-GPU SLURM required at 10B scale
GPU VRAM	—	≥24 GB	≥24 GB per GPU	Measured peak: 22.3 GB on V100
System RAM	—	≥64 GB	≥64 GB per node	Measured peak: 43.6 GB

CPU cores	—	≥4	≥4 per node	Measured peak: ~3.7 core-equivalents
Scratch storage	—	≥50 GB	≥50 GB per node	Measured for 30-iteration run on 1M compounds; enable clear_prev_iterations: true to reduce footprint
Clear intermediate files	clear_prev_iterations	false (keep for inspection)	true	Recommended at ultra-large scale to manage disk usage

Table S1 | DDU recommended configurations for prospective virtual screening campaigns. Parameter names in *monospace* correspond to settings in *config/params.yml*. Recommendations are derived from benchmarking across 12 protein targets and the prospective PGK2 ultra-large screening campaign. "Standard" refers to libraries of manageable size (≤1B compounds) where full database retention is computationally feasible; "Ultra-large" refers to campaigns at the billion-scale or beyond where database removal is required for practical turnaround.

S5. Computational Resource Benchmarks

To provide a concrete computational reference for community adoption, we measured GPU memory utilisation, system RAM, CPU load, inference throughput, and storage requirements for each phase of the DDU pipeline on a 1-million-molecule library (target 6IIU; hardware: 50 Tesla V100-SXM2-32GB GPUs for docking and inference sub-jobs, 1 V100-32GB for training, Intel Xeon Silver 4116 orchestrator, 810 GB total system RAM).

Phase	**Molecules processed**	**GPU subjobs**	**Wall time (50 GPUs)**	**Throughput per GPU**		**Peak VRAM per GPU**	**Peak orchestrator RSS**	**Batch size**
				mean	**min-max**			

Docking (QuickVina2-GPU)	170,000	50	22.8 min	6.6 mol/s	2.8–6.9 mol/s	14.8 GB	2.5 GB	—
Model training (MoLFormer-DR, 40 epochs)	20,000 (labelled)	1	10.0 min	1,331 samples/s	—	11.8 GB allocated / 12.1 GB reserved	4.4 GB	2,048
Pool inference (MoLFormer-DR)	830,000	50	1.3 min	1,759 mol/s	1,581–1,884 mol/s	1.2 GB	4.3 GB	512
Total per AL iteration	**1,020,000**	**50**	**~34 min**	—	—	—	—	—

Table S2. *DDU computational resource requirements per AL iteration (1M-molecule library, 50 Tesla V100-SXM2-32GB GPUs, target 6IIU).*

Docking VRAM (14.8 GB) was identical across all 50 sub-jobs (std = 0), reflecting deterministic QuickVina2-GPU parameter loading. Orchestrator RSS peaks during training/inference due to in-memory retention of the 1M-molecule SMILES DataFrame (170 MB) plus model tensors. CPU load on the orchestrator peaked at 3.1 core-equivalents during training (from 24 physical cores available).

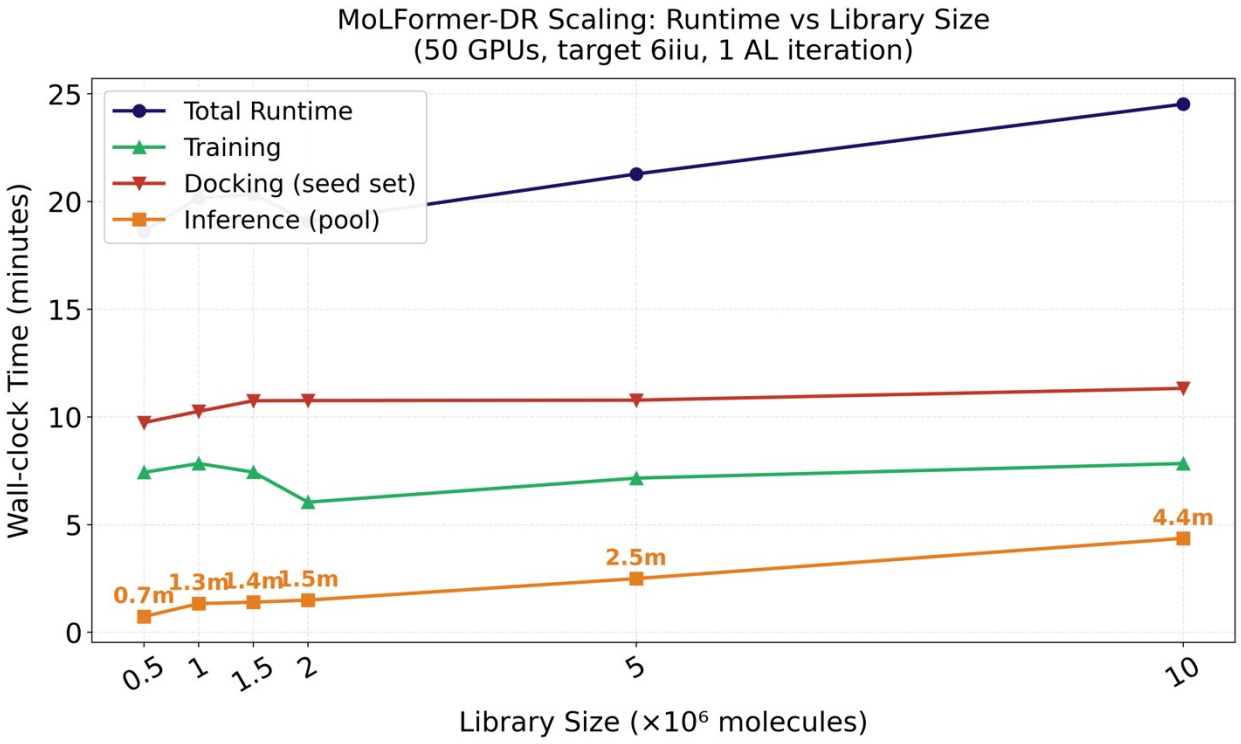


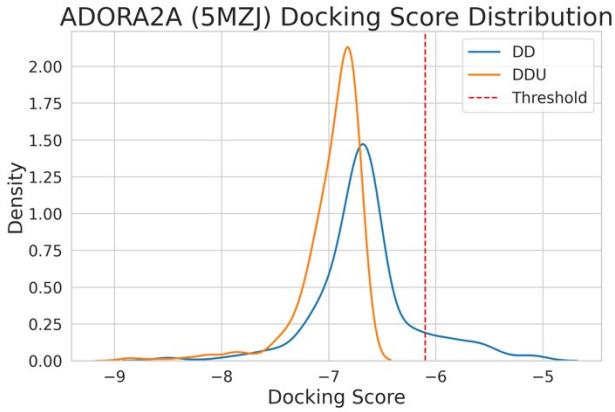
Figure S9. Bottleneck step is a function of library size. Docking is the wall-clock bottleneck because each molecule must be geometrically prepared and scored by QuickVina2-GPU; this step does not benefit from batching in the way neural forward passes do. Training time is independent of library size (it depends only on the labelled set, fixed at 20,000 molecules). Inference scales linearly with library size. As library size increases beyond 100-million-mark, inference step becomes the rate determining step for the DDU workflow.

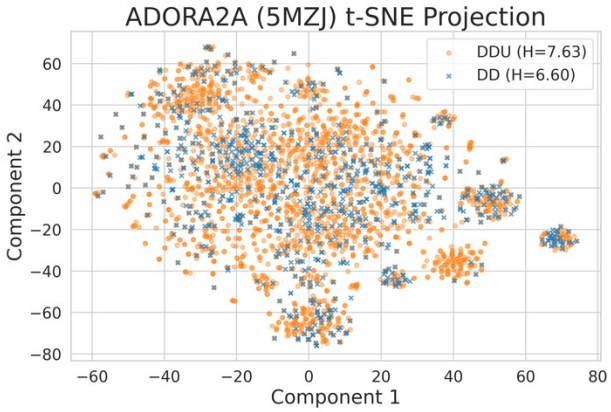
Storage. The Enamine REAL 10.1B-molecule library occupies 940 GB on disk (~90 MB per million SMILES rows). Each DDU iteration produces approximately 1 GB of output: model checkpoint (268 MB), docking score files, and inference score arrays (~730 MB combined). The DDU orchestrator requires no pre-computed molecular fingerprints; the 1M-molecule SMILES library occupies only 170 MB in RAM as a pandas DataFrame, compared to >200 GB required for pre-computed Morgan fingerprints at the same scale.

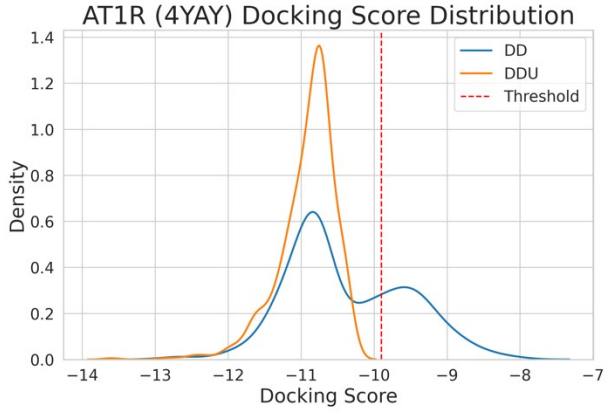
Minimum hardware requirements. The most VRAM-intensive phase is docking (14.8 GB peak), setting the minimum GPU specification at ≥16 GB VRAM. This is met by widely available

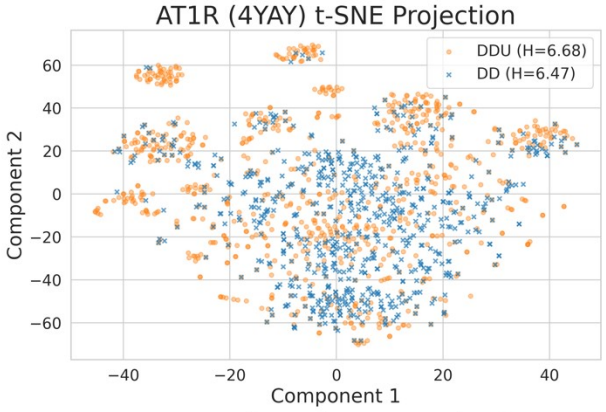
GPUs including the NVIDIA V100-16GB, A100-40GB, RTX 3090 (24 GB), and RTX 4090 (24 GB). The orchestrator node requires ≥16 GB of system RAM and ≥4 CPU cores. DDU does not require a supercomputer: reducing from 50 to 1 GPU increases only the wall time of the docking and inference phases linearly, with no effect on model accuracy or AL convergence. On a single V100-32GB GPU, one AL iteration on a 1-million-molecule library is estimated to take approximately 8 hours (dominated by sequential docking).

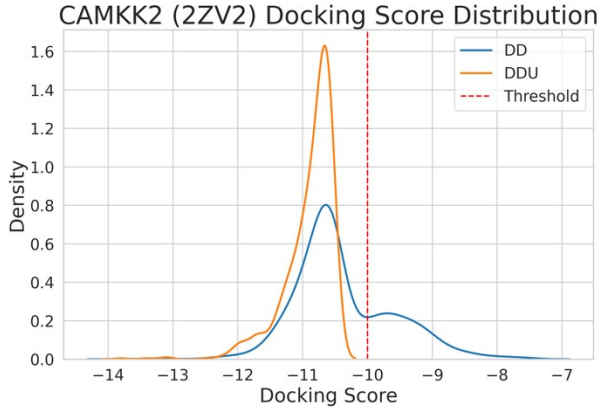
S6. **Comparative performance of Deep Docking and DDU**

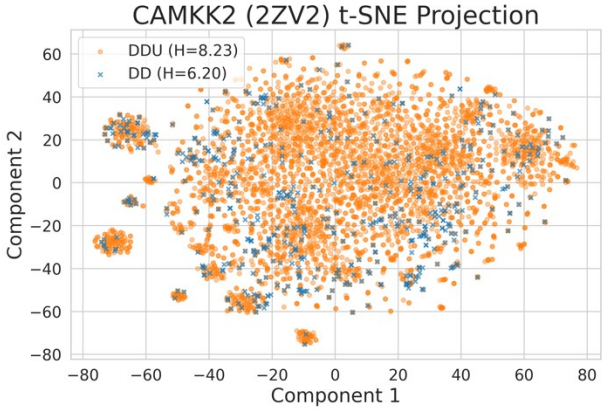

ADORA2A (5MZJ) Docking Score Distribution
DD
DDU
Threshold
Density
Docking Score

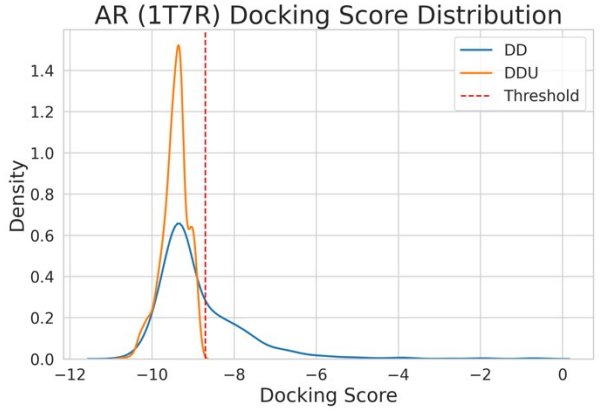

ADORA2A (5MZJ) t-SNE Projection
DDU (H=7.63)
DD (H=6.60)
Component 2
Component 1

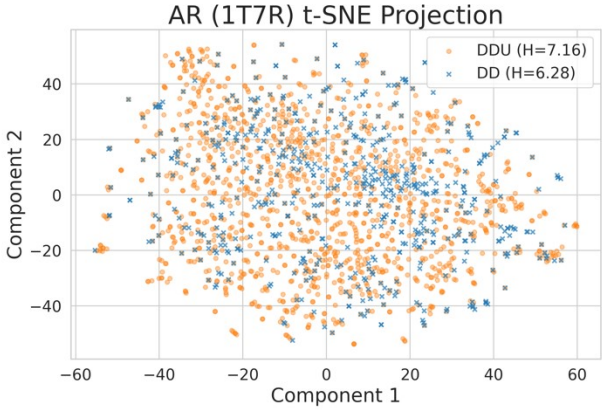

AT1R (4YAY) Docking Score Distribution
DD
DDU
Threshold
Density
Docking Score

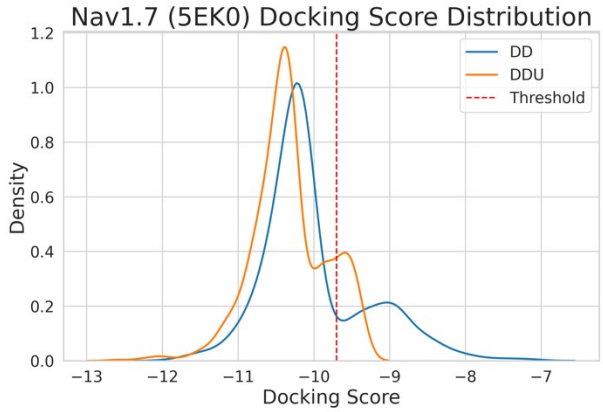

AT1R (4YAY) t-SNE Projection
DDU (H=6.68)
DD (H=6.47)
Component 2
Component 1

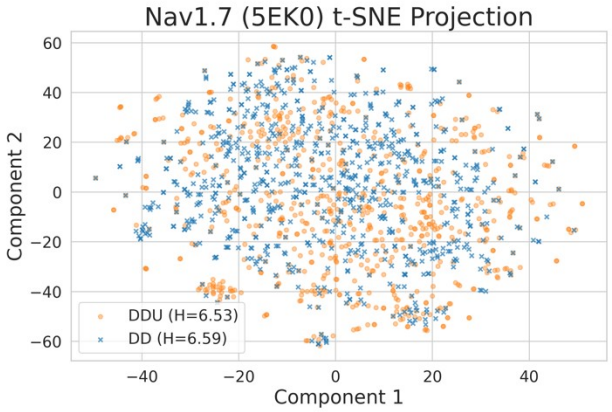

CAMKK2 (2ZV2) Docking Score Distribution
DD
DDU
Threshold
Density
Docking Score

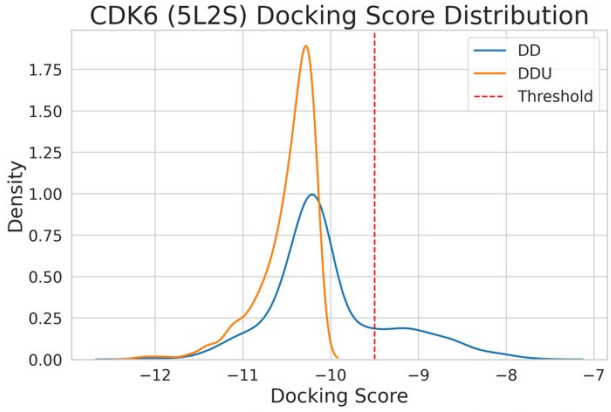

CAMKK2 (2ZV2) t-SNE Projection
DDU (H=8.23)
DD (H=6.20)
Component 2
Component 1

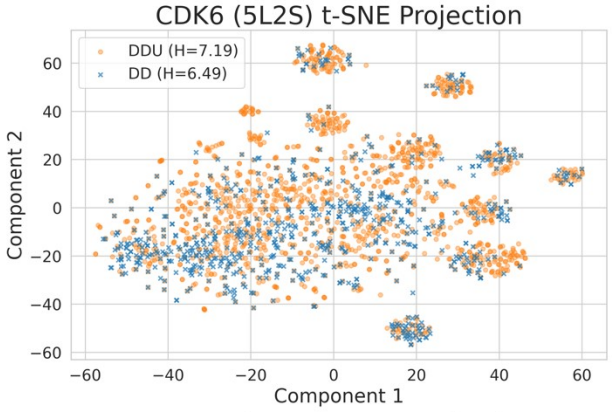

AR (1T7R) Docking Score Distribution
DD
DDU
Threshold
Density
Docking Score

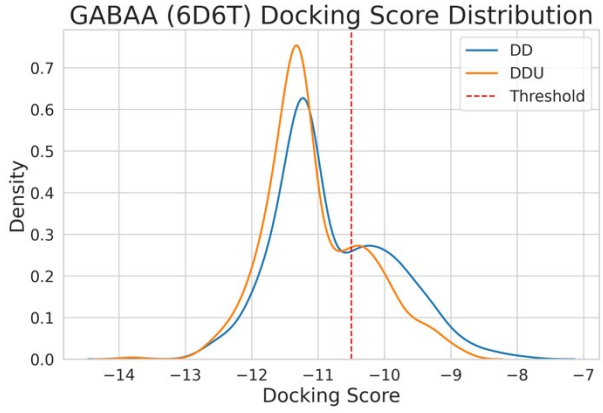

AR (1T7R) t-SNE Projection
DDU (H=7.16)
DD (H=6.28)
Component 2
Component 1

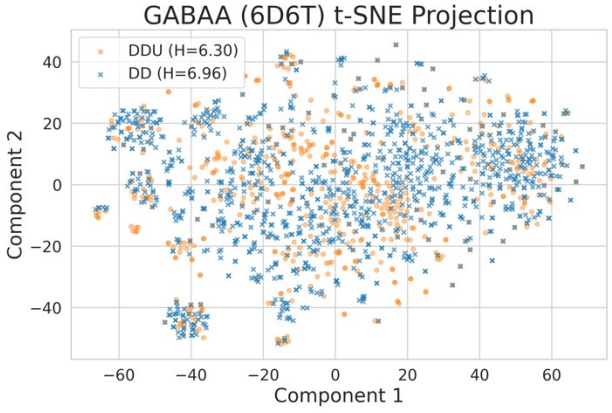

Nav1.7 (5EK0) Docking Score Distribution
DD
DDU
Threshold
Density
Docking Score

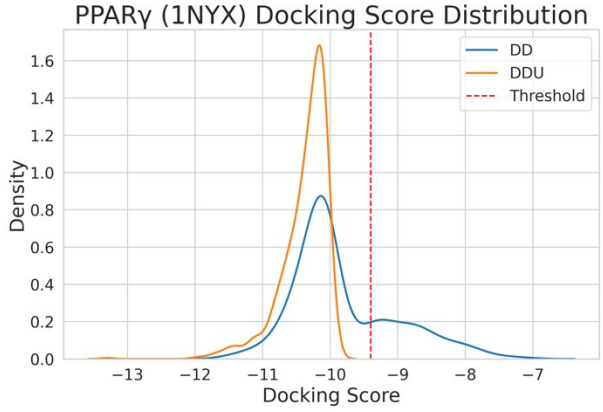

Nav1.7 (5EK0) t-SNE Projection
DDU (H=6.53)
DD (H=6.59)
Component 1
Component 2


CDK6 (5L2S) Docking Score Distribution
DD
DDU
Threshold
Density
Docking Score


CDK6 (5L2S) t-SNE Projection
DDU (H=7.19)
DD (H=6.49)
Component 1
Component 2


GABAA (6D6T) Docking Score Distribution
DD
DDU
Threshold
Density
Docking Score


GABAA (6D6T) t-SNE Projection
DDU (H=6.30)
DD (H=6.96)
Component 1
Component 2


PPARγ (1NYX) Docking Score Distribution
DD
DDU
Threshold
Density
Docking Score

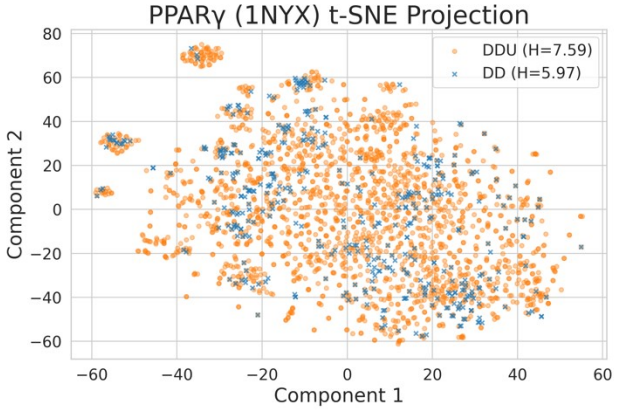

PPARγ (1NYX) t-SNE Projection
DDU (H=7.59)
DD (H=5.97)
Component 1
Component 2

Figure S10. Supplementary figure to figure 3 from the main text: Docking score distributions (left panels) and t-SNE projections (right panels) are shown for the top 2000 virtual hits identified by Deep Docking (blue) and DDU (orange) for eight targets from the benchmark target set that were not included in main text Figure 3.

References

1. Kipf, T. N. & Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* https://arxiv.org/pdf/1609.02907 (2016).
2. Veličković, P. *et al.* Graph Attention Networks. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings* https://doi.org/10.1007/978-3-031-01587-8_7 (2017) doi:10.1007/978-3-031-01587-8_7.
3. Hu, W. *et al.* Strategies for Pre-training Graph Neural Networks. *8th International Conference on Learning Representations, ICLR 2020* https://arxiv.org/pdf/1905.12265 (2019).
4. Xu, K., Jegelka, S., Hu, W. & Leskovec, J. How Powerful are Graph Neural Networks? *7th International Conference on Learning Representations, ICLR 2019* https://arxiv.org/pdf/1810.00826 (2018).