Title

HDBind: encoding of molecular structure with hyperdimensional binary representations.

Permalink

https://escholarship.org/uc/item/5wq587rn

Journal

Scientific Reports, 14(1)

Authors

Jones, Derek
Zhang, Xiaohua
Bennion, Brian
et al.

Publication Date

2024-11-23

DOI

10.1038/s41598-024-80009-w

Peer reviewed

scientific reports

OPEN

HDBind: encoding of molecular structure with hyperdimensional binary representations

Derek Jones[1,2]✉, Xiaohua Zhang[3], Brian J. Bennion[3], Sumukh Pinge[1], Weihong Xu[1], Jaeyoung Kang[1], Behnam Khaleghi[1], Niema Moshiri[1], Jonathan E. Allen[2] & Tajana S. Rosing[1]

Traditional methods for identifying "hit" molecules from a large collection of potential drug-like candidates rely on biophysical theory to compute approximations to the Gibbs free energy of the binding interaction between the drug and its protein target. These approaches have a significant limitation in that they require exceptional computing capabilities for even relatively small collections of molecules. Increasingly large and complex state-of-the-art deep learning approaches have gained popularity with the promise to improve the productivity of drug design, notorious for its numerous failures. However, as deep learning models increase in their size and complexity, their acceleration at the hardware level becomes more challenging. Hyperdimensional Computing (HDC) has recently gained attention in the computer hardware community due to its algorithmic simplicity relative to deep learning approaches. The HDC learning paradigm, which represents data with high-dimension binary vectors, allows the use of low-precision binary vector arithmetic to create models of the data that can be learned without the need for the gradient-based optimization required in many conventional machine learning and deep learning methods. This algorithmic simplicity allows for acceleration in hardware that has been previously demonstrated in a range of application areas (computer vision, bioinformatics, mass spectrometery, remote sensing, edge devices, etc.). To the best of our knowledge, our work is the first to consider HDC for the task of fast and efficient screening of modern drug-like compound libraries. We also propose the first HDC graph-based encoding methods for molecular data, demonstrating consistent and substantial improvement over previous work. We compare our approaches to alternative approaches on the well-studied MoleculeNet dataset and the recently proposed LIT-PCBA dataset derived from high quality PubChem assays. We demonstrate our methods on multiple target hardware platforms, including Graphics Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs), showing at least an order of magnitude improvement in energy efficiency versus even our smallest neural network baseline model with a single hidden layer. Our work thus motivates further investigation into molecular representation learning to develop ultra-efficient pre-screening tools. We make our code publicly available at https://github.com/LLNL/hdbind.

Keywords Hyperdimensional computing, Machine learning, Representation learning, Computational chemistry, Drug discovery

The modern drug discovery process consists of multiple sequential steps that progress from an initial large collection of candidates, sampled from the estimated $10^{60} - 10^{100}$ possible drug-like small molecule structures, to a smaller targeted set of *hit* or lead compounds with potential activity with protein targets of interest[1]. These candidates are filtered according to their likelihood of success based on a scoring function that uses either physics-based modeling[2] or, increasingly, properties inferred directly from data using machine learning[3–5]. The results of the *virtual screen* are then used to identify molecular *leads* for more rigorous—and expensive—experimental validation[1]. Public catalogs of drug-like molecules have grown to comprise tens of billions of possibilities[6], while the number of available protein structures has simultaneously grown with the introduction of AI-enabled 3D structure prediction tools, resulting in over 200 million publicly available predicted structures[7,8]. Even the exhaustive interrogation of the approximately 20,000 human proteins poses a considerable computational

[1]Department of Computer Science and Engineering, University of California-San Diego, La Jolla, CA, USA. [2]Global Security Computing Applications Division, Lawrence Livermore National Laboratory, Livermore, CA, USA. [3]Biosciences and Biotechnology Division, Lawrence Livermore National Laboratory, Livermore, CA, USA. ✉email: wdjones@ucsd.edu

challenge. While increasingly complex deep learning architectures are demonstrating state-of-the-art (SOA) results on a wide range of molecular property prediction tasks[9–11], it is becoming increasingly clear that energy efficiency will become a greater priority over time as these models are, repeatedly, trained and deployed on increasingly vast human protein-drug interactome[12].

Hyper-dimensional computing (HDC) is an emerging paradigm of lightweight machine learning that leverages the orthogonality of vectors in high dimensional space coupled with simple arithmetic operations for learning that are, comparatively to SOA deep learning architectures, simple to implement in hardware, and thus primed to take advantage of emerging hardware acceleration breakthroughs[13–27]. HDC has been demonstrated as a versatile and efficient approach for a growing variety of application domains including proteomics[28], molecular property prediction[29], medical image classification, visual scene understanding[30,31], and biosignal classification[18,19]. HDC requires the specification of an *encoding* method to transform the original input data representation into a high-dimensional vector space as *hypervectors*[14,15]. Then, given a similarity metric defined on the high-dimensional space, commonly chosen as the cosine similarity, which is sensitive only to the relative orientation, similar hypervectors can then be aggregated in order to build higher-level class prototype representations that form the *associative memory* of the model[14,15]. Inference then simply requires computing the similarity between a query hypervector and the elements of the associative memory[14,15]. Despite the potential of HDC to provide a lightweight and energy efficient method for classification in the context of screening protein–ligand interactions, to the best of our knowledge, there has only been a single previously reported study of HDC on a molecular machine learning task in general[29], but this work does not consider the problem of protein-drug interactions. Our work is the first to use HDC to accelerate protein–drug interactions, in combination with a range of molecular representations including the well-studied Extended Connectivity FingerPrint (ECFP)[32] as well as representations extracted from a state of the art Large Language Model (LLM) and self-supervised graph pretraining algorithms[9,10]. Our work considers a panel of 6 molecular property prediction tasks derived from MoleculeNet[11]. We show improved performance compared to the SOA HDC-based approach MoleHD[33] as well as baseline traditional ML methods. Our results additionally show improvement in some cases over the finetuned LLM, MolFormer-XL[9]. We consider the LIT-PCBA binding interaction dataset, which collects experimental data across 15 protein targets selected from high-confidence PubChem Bioassay data[34]. Our work is thus the first HDC-based study of a real-world collection of molecular activity data beyond the benchmark datasets that have been considered until now, demonstrating a compelling use case for HDC in a challenging real-world application with a fair comparison to traditional physics-based molecular docking and baseline Multi-layer Perceptron (MLP) that is typically trained on top of a given molecular vector representation for a downstream task.

Materials and methods

Hyperdimensional computing (HDC)

Hyperdimensional computing (HDC) is an emerging paradigm for building lightweight and error-robust models for classification and clustering[14,15]. HDC leverages the properties of high-dimensional vector spaces. With increasingly large dimension size $D : D \in \mathbb{Z}^+$, the distance between any pair of randomly selected vectors converges towards the expected distance between all vectors[15,35]. Thus, nearly all vectors are unrelated and can be considered as *quasi-orthogonal*; it is then possible to attribute unique vectors to semantically meaningful properties of the dataset $X : x \in \mathbb{R}^n$ (i.e. element type, number of bonds, etc.)[15,36]. An *encoding* function $\phi(x)$ is specified to produce the representations in the high-dimension space $H : h \in \mathbb{R}^D$ from the samples of the dataset. The encoding function ϕ may incorporate prior knowledge about the mapping between the ambient data dimension and the high dimensional space or may be a parameterized function such as a neural network that is learned from the data[15,33,37]. Simple arithmetic operations can be used to reason with the the high-dimensional vectors h. The *binding* operator $\otimes : H \times H \to H$ is used to create ordered tuples of points in H. We define $\otimes$ as the hadamard or element-wise product, which is associative and commutative:

$$\otimes(a, b) = \sum_i a_i b_i. \tag{1}$$

The bundling operator $\oplus : H \times H \to H$ allows for the composition of information from disparate sources into a single representation[15]. We define $\oplus$ as the element-wise sum, which is associative and commutative:

$$\oplus(a, b) = \sum_i a_i + b_i. \tag{2}$$

Lastly, permutation Π is used to, efficiently, incorporate positional information into the representation h[15,29,38].

Learning in HDC

```
Require:
  Encoded training hypervectors, $H$
  Training labels, $Y$
  Batch size, $B$
  procedure BUILDAM($H, Y, B$)
    for $(h_b, y_b) \in$ GenerateBatches($H, Y; B$) do
      for $k \leftarrow 0$ to $K$ do
        $h_k = h_b[y_b == k]$                    ▷ collect all hypervectors in class $k$ for batch $b$
        $\mathbb{A}[k] += \sum_{i=1}^{B} h_{k,i}$          ▷ sum the hypervectors $h_k$ and add to the associative memory
      end for
    end for
    return $\mathbb{A}$                                        ▷ return initialized associative memory
  end procedure
```

Algorithm 1. HDC AM: build associative memory module $\mathbb{A}$

```
Require:
  Associative memory module, $\mathbb{A}$
  Encoded training hypervectors, $H : h \in H$
  Training labels, $Y : y \in Y$
  Batch Size $B$
  procedure UPDATEAM($\mathbb{A}$, $H$, $Y$, $B$)
    for $(h_b, y_b) \in$ GenerateBatches($H, Y; B$) do
      $s_b = \rho(\mathbb{A}, h_b)$      ▷ Compute pairwise cosine similarity in parallel for the batch and the associative mem.
      $\hat{y}_b = \operatorname{argmax}_k s_b[:,k]$        ▷ Get the indices of the most similar AM element as the predicted class
      $e = \hat{y}_b \neq y_b$                          ▷ compute a binary mask using the model's errors
      $M = (h_b[e,:], y_b[e,:])$
      for $(h, y) \in M$ do                                     ▷ for each mistake
        $\mathbb{A}[y] += h$                        ▷ Add the mistake to the correct a.m. entry
        $\mathbb{A}[1-y] -= h$                    ▷ Subract the mistake from the incorrect a.m. entry
      end for
    end for
    return $\mathbb{A}$                                        ▷ return updated associative memory
  end procedure
```

Algorithm 2. HDC retrain: update associative memory module $\mathbb{A}$

```
Require:
  Associative memory module, $\mathbb{A}$
  Encoded testing hypervectors, $H : h \in H$
  Batch Size $B$
  procedure PREDICTAM($\mathbb{A}$,$H$, $B$)
    $\hat{y} = \{\}$
    for $h_b \in$ GenerateBatches($H; B$) do
      $s_b = \rho(\mathbb{A}, h_b)$      ▷ Compute pairwise cosine similarity in parallel for the batch and the associative mem.
      $\hat{y}_b = \operatorname{argmax}_k s_b[:,k]$        ▷ Get the indices of the most similar AM element as the predicted class
      $\hat{y} = \hat{y} \cup \hat{y}_b$                                ▷ append the batch predictions
    end for
    return $\hat{y}$                                                     ▷ return predictions
  end procedure
```

Algorithm 3. HDC test

HDC supports the development of lightweight classification models without the need of numerical optimization approaches such as stochastic gradient descent (SGD) or more sophisticated alternatives typically used to train deep neural networks[39–41]. Learning in HDC for a set of K classes proceeds by the construction of prototypes $h_k : k \in K$ for each class k:

$$h_k = \bigoplus_{i|y_i=k} \phi(x_i) \tag{3}$$

where x_i is the ith sample from the dataset X and y_i is the respective class label. The initial epoch of training consists of building the *associative memory* $\mathbb{A}$ of the model by applying $\phi(x)$ to the input dataset X producing a representative prototype vector h_k for each class with a single pass over the training set (Algorithm 1). To perform inference on a query hypervector h_q, we simply compute:

$$\hat{y} = \underset{k \in K}{\operatorname{argmax}}\ \rho(h_k, h_q) = \underset{k \in K}{\operatorname{argmax}}\ \rho(h_k, \phi(x_q)) \tag{4}$$

where ρ denotes a user-specified similarity metric and x_q is the query data point.[15]. In our work we implement ρ as the cosine similarity:

$$\rho(h_k, h_q) = \rho(h_k, \phi(x_q)) = \cos(\theta) = \frac{h_k \cdot \phi(x_q)}{||h_k||\ ||\phi(x_q)||} \tag{5}$$

After constructing $\mathbb{A}$ with single-pass learning, it can be further refined with a *re-training* phase (Algorithm 2). This phase tests the model predictions on the training set then updates $\mathbb{A}$ accordingly. This operation functions to increase the distance from the incorrect class prototype(s) while decreasing the distance to the correct class prototype. For testing the learned $\mathbb{A}$, we simply compare the hypervectors of the test set with $\mathbb{A}$ using the user-defined similarity metric ρ and select the index of the most similar prototype to represent the predicted class (Algorithm 3).

Encoding molecular data for HDC

Small drug-like molecules are often described using the "simplified molecular-input line-entry ststem" (i.e. SMILES) which encodes the structure as an ASCII string[42]. The SMILES string itself describes a depth-first traversal of the 2D molecular graph structure. The ECFP representation considers the graph representation of the molecule and is widely used in computational chemistry for tasks such as similarity search in chemical libraries as well as a feature for ML models. ECFP is based on the Morgan algorithm[43], which was originally proposed to solve the molecular isomorphism problem and is widely used for chemical similarity analysis as well as general purpose representations for machine learning. The ECFP algorithm makes changes to MorganFP that improve efficiency, such as a user-defined iteration limit, a cache to store intermediate atom identifiers between iterations, and a hashing scheme to record the resulting representations[32]. Thus, ECFP effectively uses a bottom-up approach to collect progressively larger molecular substructures that are guaranteed to coherently preserve the graph structure as any entry in the ECFP corresponds to a valid subgraph of the input molecular graph whereas a randomly selected substring of a SMILES may not correspond to a valid subgraph or even a valid SMILES string[32]. ECFP allows for a user to specify the number of bits (i.e. vector length) $n \in \mathbb{Z}^+$ in a representation, commonly chosen as 1024 or 2048[5,11]. Further, a maximum radius size $r \in \mathbb{Z}^+$ (i.e., number of edges (bonds) from a root node (atom)) for collecting substructure-graphs is specified to constrain the search for substructure information. Thus each binary value in the ECFP representations indicates the presence or lack thereof for a chemical substructure.

Random projection fingerprint encoding (RPFP)

Random Projection (RP) provides a simple method for dimensionality reduction[44,45]. RP can also be considered as the basis of an encoding method to produce high-dimensional embeddings h that preserve the relative distances of the input data[15,38]:

$$z = xW^\top \tag{6}$$

$$h = \sigma(z) \tag{7}$$

where $W \in \mathbb{R}^{D \times n}$ is a matrix whose rows are randomly sampled from the surface of the unit sphere[15]. The quantization operator $\sigma(z)$ is defined as:

$$\sigma(z) = \begin{cases} -1 & \text{where } z \leq 0 \\ 1 & \text{where } z > 0 \end{cases} \tag{8}$$

Direct ECFP encoding (DECFP)

The direct ECFP encoding (DECFP) approach simply uses the rdkit[46] function GetMorganFingerprintAsBitVect to compute fingerprints for each molecule, which given their sparse binary properties satisfy our definition of hypervectors. The nBits parameter is adjusted to equal D corresponding to the hypervector dimension. This can be described as:

$$z = \mathrm{E}_{n,r}(s) \tag{9}$$

$$h = \sigma(z) \tag{10}$$

where s denotes the SMILES string corresponding to a particular sample. As no matrix multiplications are required, the entire encoding process is carried out on the CPU.

Large-scale self-supervised representations

Data-driven molecular representation learning has caught much attention in recent years in tandem with the rise of deep learning[9,10,47,48]. We investigate the SOA approach, MoLFormer[9], as the basis of the molecular representation we consider. MoLFormer uses the masked language model framework[49,50] and thus employs self-supervision to learn to predict missing tokens from within a SMILES sequence[9]. An alternative pretraining paradigm instead uses the molecular graph representation along with graph-centric augmentations (atom masking, bond deletion, subgraph removal) and self-supervised contrastive learning objectives[10,51]. The SOA approach MolCLR[10] is considered in our work. Previous work has considered the use of neural networks for the basis of an HDC embedding[33], however we are the first to our knowledge to consider a model obtained from an extensive training run on large collections of publicly available molecular data[9]. Similarly to the ECFP encoding, we use the random projection approach described previously to realize the HDC embeddings as HDB-MoLFormer (Fig. 2) and HDB-MolCLR. This is a similar strategy to previous work which uses a deep convolutional neural network as a feature extractor to generate input representations for the random projection layer[52].

Effectively, the HDB-MoLFormer and HDB-MolCLR strategies may be considered as a neural network of L layers where the initial $0 \leq l \leq L-1 : l \leq L$ layers are trained using a gradient-based optimization scheme with a self-supervised (pre-)training objective. The Lth layer in this network then uses a randomly sampled linear projection layer (bias omitted) with a sign activation function (Eq. 8) to truncate the input values to be in the binary space $\{-1, 1\}$. The outputs and their labels are collected to form the associative memory of the model which are subsequently used for HDC training and inference (Algorithm 1, 2, and 3).

Ranking compounds with HDC

To rank compounds for the HDC methods, we use the confidence estimation equation as described in MoleHD[29]. For a binary classifier, the range of similarity differences between the positive and negative classes are transformed linearly to the interval [0,1].:

$$\eta = \frac{1}{2} + \frac{\rho(h_q, h_0) - \rho(h_q, h_1)}{4} \tag{11}$$

where h_0 and h_1 are respectively the negative and positive class prototype hypervectors contained in the model associative memory $\mathbb{A}$ and h_q is the query hypervector. Intuitively, if h_q is equally similar to both h_0 and h_1, $\eta = \frac{1}{2}$. If h_q is more similar to h_1, $\eta > 0.5$, otherwise if h_q is more similar to h_0, then $\eta < 0.5$.

Metrics

To facilitate comparison with previous work on MoleculeNet[11], we use the receiver operating characteristic—area under the curve (ROC-AUC) to measure performance of different models. The ROC-AUC metric compares the true positive rate (TPR) and false positive rate (FPR) of a classifier at various thresholds of a models score to identify a positive class. The area under the curve produced by the various thresholds is measured with respect to a perfect classifier (TPR=1, FPR=0 for all thresholds).

It is common in the high-throughput screening literature to measure performance in terms of a scoring function in terms of the to encounter the enrichment factor (EF) metric[53,54], which attempts to measure how well a screening method may be able to improve the density of actives in a large database of molecular candidates. The EF metric is typically defined in terms of the hit rate for a sample compared to the background hit rate of the full database. As modern databases may reach billions, a tractable sample is chosen for ruther validation, such as the top 1% of compounds as ranked by the outputs of some scoring function. Let a_s, a_b represent the number of actives and n_s, n_b the size of the sample and database respectively. Then let $p_s = a_s/n_s$ be the probability of selecting an active from a sample of ranked compounds (i.e. sample hit rate) and $p_b = a_b/n_b$ be the probability of selecting an active compound from the database (i.e. background hit rate) of the database. The enrichment factor (EF) is then calculated as the ratio between the two quantities:

$$\text{EF-}x\% = \frac{p_s}{p_b} = \left(\frac{a_s}{n_s}\right) / \left(\frac{a_b}{n_b}\right) = \frac{a_s}{a_b} \cdot \frac{n_b}{n_s} \tag{12}$$

where $x = n_s/n_b$ is the fraction of top ranked molecules sampled from the database (e.g. $x = 1\%$). This measurement of enrichment however is subject to the limitation of its sensitivity to the proportion of the active to inactive compounds in the test set, which is typically highly skewed in binding activity datasets[34,55]. Several works have proposed an alternative metric which instead uses a fixed false positive rate to measure the enrichment factor[56–58]. This approach removes the limitation of being dependent on the active to inactive ratio. To facilitate direct comparison to previously published methods[58], we report this definition of *roc-enrichment* using a false positive rate of $x\%$ as ER-$x\%$:

$$\text{ER-}x\% = \text{ROC-Curve}(\text{FPR-}x\%) \times 100 = \text{TPR}_{\text{FPR-}}x\% \times 100 \tag{13}$$

where TPR is the true positive rate given by the ROC-Curve at the false positive rate of $x\%$ (FPR-$x\%$). We use $x = 1\%$ to compare with previous work[58], however when considering large databases it may be more tractable to consider smaller sample sizes (i.e. $x = 0.1\%, 0.2\%,$ and 0.5%).

Training details

All methods presented are trained on the Lassen high-performance computing cluster at Lawrence Livermore National Laboratory. Coarse-grained parallelism was achieved for each dataset by randomly sampling a task and running independently on each node of a given allocation. Each node is equipped with an IBM Power 9 CPU, 256GB of main memory, and 4x Nvidia V100 GPUs. Our experiments only consider a single GPU for all methods. All HDC methods share the same training and testing algorithms (Algorithm 1, 2, and 3), with the only difference being the encoding algorithms used to produce the high-dimensional vector representations. A batch size of 128 was used for training all HDC models considered to enable fair comparison between different hypervector dimension sizes D and GPU memory usage. All MLP models are optimized using Ray.Tune hyperparameter optimization library[59]. We use the AsynchronousHyperBand scheduler with default parameters to sample 50 configurations. The best model, according to the minimum validation loss, is selected to train on the full dataset and evaluated on the test set for performance metrics.

Energy analysis

To estimate energy usage, we use the following equation:

$$E = \bar{P} \times t \tag{14}$$

where E is the energy usage (Joules), $\bar{P}$ is the average power output (Watts) of the processor (CPU, GPU, or FPGA) over the course of the program execution, and t is the execution time or latency of the program. To collect power measurements for CPU and GPU we use the the variorum power and performance measurement tool[60]. We collect all performance measurements, not including the FPGA, on the Lassen HPC cluster using a single Nvidia V100 GPU.

Results

Molecular property classification on MoleculeNet

Previous work on supervised learning approaches

The MoleculeNet benchmark is a common performance benchmark for machine learning methods across a variety of regression and classification tasks. We consider a series of 6 classification tasks to compare with recently published SOA methods[9,10]. N-gram[61], GeomGCL[62], MolCLR[10], and MolFormer-XL[9] represent self-supervised methods with SOA results as reported previously[9]. MolCLR[10] is a molecular graph pretraining method composed of atom masking, bond deletion, and subgraph removal graph augmentations whose encoded representations are used as input to the normalized temperature-scaled cross-entropy (NT-Xent) contrastive loss[63]. MolCLR is trained on approximately 10 million SMILES strings collected from the PubChem database[64]. MoLFormer-XL[9] is another recently proposed self-supervised pretraining method that is instead built using the masked language model framework[49,50] and further expands the training set considered by MolCLR[10] by two orders of magnitude, training on over 1 billion SMILES from PubChem[64]. The pre-trained MoLFormer and MolCLR models are then fine-tuned on the target MoleculeNet classification tasks by training an MLP on top of the output layers of the pre-trained networks using a supervised loss (e.g. cross-entropy or negative log-likelihood). Representative baseline supervised machine learning methods are collected from previously published methods[9,65,66] except for our own implementation of the MLP.

HDC methods on MoleculeNet

To our knowledge, MoleHD[33] is the only known previously published HDC approach for molecular property prediction in general. MoleHD uses an encoding of the SMILES string that is built upon the byte-pair encoding algorithm that accounts for atoms as cohesive structures and is trained using ChEMBL[67,68]. MoleHD collects the unique tokens collected by the SmilesPair Encoding algorithm[67] and maps these tokens to unique, quasi-orthogonal vectors of high dimension (e.g. 10,000). MoleHD also considers n-gram encoding methods, however the SPE method appears to produce the best overall method which we base our implementation on and our comparison. Results for all of the discussed models are compared to our proposed HDBind (HDB) approaches that consider two state-of-the-art self-supervised pretraining frameworks, MoLFormer[10] and MolCLR[9] and the well studied Extended Connectivity Fingerprint (ECFP)[32] which incorporates substructure information derived directly from the molecular graph and its atom types and connectivity. Our hypothesis is that the explicit graph representation considered by the ECFP algorithm[32] provides coherent substructure information (i.e. each ECFP bit corresponds to a valid molecular subgraph) that is crucial to identify in molecular property classification tasks[49]. Further, our hypothesis for large scale pretraining methods is that the random projection will preserve the structure of the original data in a randomly selected high dimensional space, with low required precision, allowing for extremely large vectors to be stored. Previous work has demonstrated the utility of these pre-trained representations in a variety of molecular property classification tasks, which we expect will benefit our proposed encoding approaches.

MoleculeNet classification results

Our results are given for 6 binary classification tasks in Table 1. We give results for HDC models with hypervector dimensionality $D = 10,000$, as increasing the dimensionality to larger sizes (e.g. $1e^5, 1e^6$) tends to yield marginal improvement at best on most tasks considered. Our results suggest that the best overall HDC model is HDB-MoLFormer, which is based upon the representation extracted from MoLFormer[9] that is then randomly projected to the HDC representation. HDB-MoLFormer and is best in three of the 6 tasks among the HDC methods that we consider. The HDB-DECFP, which simply uses the representation generated directly

Method	BBBP	Tox21	ClinTox	HIV	BACE	SIDER
Molecules	2039	7831	1478	41,127	1513	1427
Tasks	1	12	2	1	1	27
RF[9]	71.4	76.9	71.3	78.1	86.7	68.4
SVM[9]	72.9	81.8	66.9	79.2	86.2	68.2
MLP	79.0	67.2	82.2	73.1	70.3	58.6
MGCN[65]	85.0	70.7	63.4	73.8	73.4	55.2
D-MPNN[66]	71.2	68.9	90.5	75.0	85.3	63.2
N-gram[61]	91.2	76.9	85.5	**83.0**	87.6	63.2
GeomGCL[62]	–	**85.0**	91.9	–	–	64.8
MolCLR$_{GIN}$[10]	73.6	79.8	93.2	80.6	**89.0**	68.0
MoLFormer-XL[9]	**93.7**	84.7	**94.8**	82.2	88.21	**69.0**
MoleHD[33]	84.4	–	98.7	–	–	56.6
HDB-RPFP	94.8 (0.3)	**70.8** (0.9)	86.3 (4.0)	71.8 (1.3)	71.3 (0.7)	55.2 (2.0)
HDB-MolCLR	66.8 (0.4)	68.0 (0.8)	71.2 (4.0)	70.6 (0.7)	**82.4** (0.5)	61.2 (1.9)
HDB-MoLFormer	**99.2** (0.1)	67.3 (1.0)	**98.8** (0.0)	**79.2** (0.6)	66.8 (0.4)	55.4 (1.9)
HDB-DECFP	93.8 (0.2)	69.6 (0.8)	90.6 (4.0)	77.8 (0.3)	74.7 (1.1)	**61.4** (1.6)
HDB-Combo	97.4 (0.3)	70.1 (1.2)	90.7 (3.4)	77.4 (0.8)	67.0 (2.7)	58.8 (2.8)

Table 1. Comparison of supervised and self-supervised baselines on representative MoleculeNet benchmarks considered in previous work using the area under the curve of the receiver operating characteristic. All values are scaled by a factor of 100 for reader convenience. All methods are evaluated using scaffold splits to minimize the molecular similarity between the training and testing sets. All reported HDC models (HDBind and MoleHD[33]) use dimension $D = 10$k. *Denotes our implementation. '-' denotes no value reported in the original work. Values in parentheses denote standard deviation of the average of 10 trials per task in each dataset. Results above the horizontal line correspond to SOA supervised and self-supervised baselines, below correspond to HDC methods.

Method	Encode (J/mol)
HDB-DECFP	0.06
HDB-MoLFormer	0.39
HDB-Combo	0.50
MLP-small	0.07
MLP-large	0.07

Table 2. Encode energy estimates for each processor choice for the HDBind. We calculate power using the sum of the average CPU and GPU power output for all models. We include the energy for all feature extraction steps for all models (including MLP baselines) in addition to the encoding costs for HDBind models. Models that only require ECFP computation possess the lowest energy penalties (MLP baslines and HDB-DECFP).

from the ECFP algorithm[32], achieves competitive performance with HDB-MoLFormer on nearly each of the six tasks, while exceeding HDB-MoLFormer on three of six tasks though it is best only on the SIDER dataset. HDB-DECFP does not require the GPU for encoding the data into hypervectors, as opposed to our random projection-based approaches, allowing for significant energy savings (Table 2). Additionally, HDB-MoLFormer achieves SOA on two of the six tasks (BBBP, ClinTox) even when compared with the fine-tuned MoLFormer-XL[9], demonstrating the ability of the approach to preserve learned substructure information provided by the more expensive pretraining. The HDB-Combo model, which combines the MoLFormer and DECFP representations (Fig. 1) achieves generally high performance five of the six tasks (BBBP, Tox21, ClinTox, HIV, and SIDER) though it fails to achieve the best overall performance on any task. Our results further show that increasing the hypervector dimension fails to significantly increase the performance of the HDB-Combo model further on the MoleculeNet classification benchmarks (SI Table S3, SI Figures S1–S6).

LIT-PCBA

Virtual molecular lead identification

The problem of virtual screening requires the specification of a scoring function that is applied to each of the candidate molecules, then these molecules are ranked accordingly then a filtered set above some threshold of the scoring function is selected for further processing with progressively more accurate but expensive algorithms. Scoring functions that approximate the experimental binding activity can be roughly divided into those that rely upon physics theory, machine learning, or some combination of the two[54,71–75]. A general workflow then

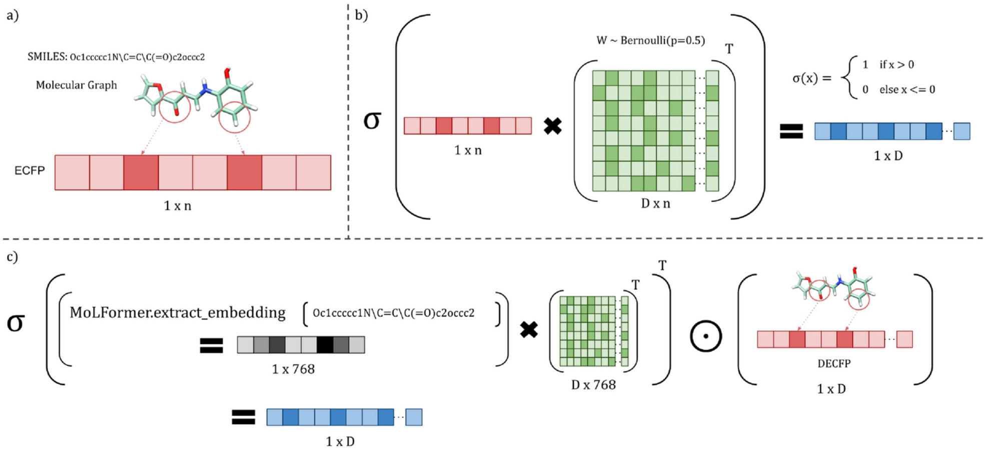

Fig. 1. Description of (**a**) the ECFP representation, (**b**) the Random Projection FingerPrint (RPFP), and (**c**) the HDB-Combo encoding. The HDB-Combo encoding uses the Hadamard (i.e., element-wise) product of the ECFP hypervector (DECFP) with the random projection of the MoLFormer representation, embedding the features learned from the large-scale self-supervised pretraining along with the coherent graph substructure information provided by the ECFP algorithm.

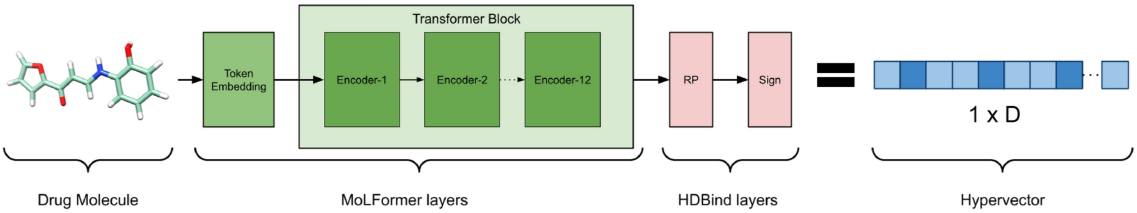

Fig.2. HDB-MoLFormer architecture description.

first applies faster but less accurate docking methods, followed by more expensive and accurate calculations based on MM/GBSA or MD simulations[74]. Physics-based methods such as *molecular docking*[76,77] are generally believed to be on the "fast" end of the spectrum of accuracy versus latency. More accurate methods including molecular mechanics/generalized Born surface area (MM/GBSA),[78,79] which provides a more accurate binding energy calculation for a given docking pose, or binding free-energy calculations based upon intensive atomistic molecular dynamics (MD) simulations, are infeasible to run for even a relatively small number of candidate possibilities[77,80]. Benchmark datasets have long been used to validate a scoring function's ability to distinguish active versus inactive molecules for a given protein target[55,81]. Recent research has identified limitations that have made these datasets trivial to learn thus overestimating the expected generalization performance when applied to real-world datasets[34,58,82–85]. The recently proposed LIT-PCBA[34] benchmark dataset is derived from high-confidence PubChem assay data (7761 actives and 382,674 unique inactives, 1:50 class ratio) and provides a rigorous test set constructed using the Atomwise-developed AVE (asymmetric validation embedding) bias-minimizing algorithm[58,86]. We additionally use a random stratified split of each protein-target specific dataset as a control with a 75%/25% train/test split ratio. To our knowledge, this represents the first demonstration of an HDC approach on a dataset of experimentally determined binding measurements of this scale of 100s of thousands[33].

Enrichment results on LIT-PCBA

In Fig. 3, we choose to report the roc-enrichment factor (ER-1%) metric (Eq. 13)[57,58,87]. We consider two representative alternative approaches for molecular screening using either machine learning or physics-based knowledge, Pafnucy[70] and GRIM[69]. Pafnucy is a 3D Convolutional Neural Network (3D-CNN) trained on the PDBBind[88] dataset to predict the binding affinity of a protein–ligand complex[70]. GRIM[69] is a fingerprint method that transforms the 3D atomic information, described using physics-based knowledge, in to a vector of 210 integers describing the molecular interaction which are then used as the basis of the GRscore. Each of these methods requires a molecular docking simulation to generate plausible 3D structures of the binding complex[87]. Our proposed HDBind models considerably outperform our implementation of the MoleHD (using PyTorch) baseline with Smiles Pair Encoding (SPE)[89] (SI Table S4). In Fig. 3 we give results compared to each of the representative methods we described. For dimension size $D < 10$k, our HDBind methods generally perform competitively with the GRIM and Pafnucy approaches across each molecular encoding approach. For $D > 10$ k however, a noticeable improvement is observed for the HDB-Combo, which combines the graph structural

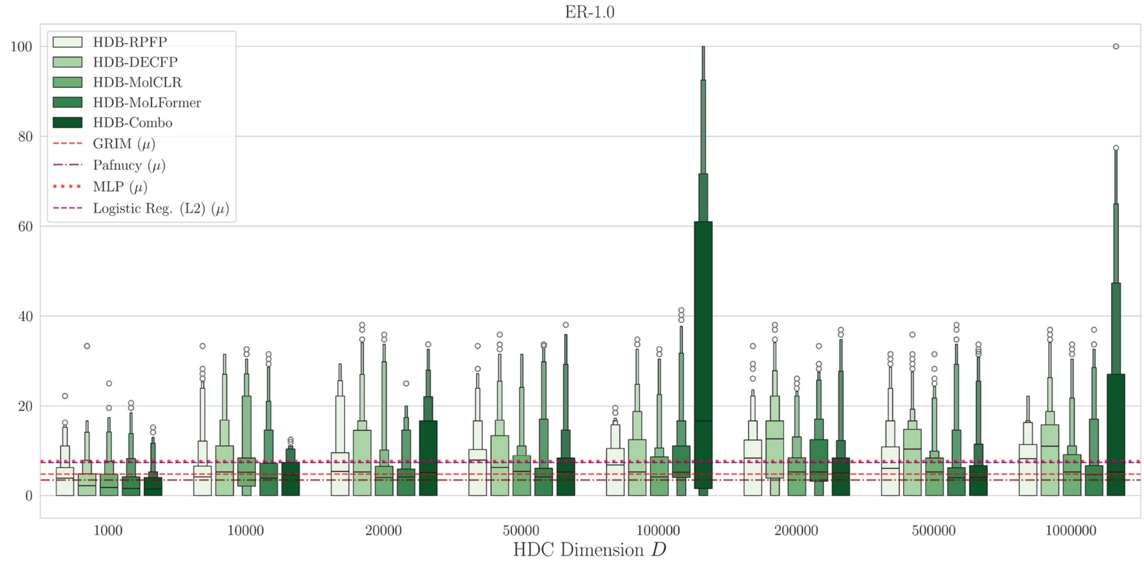

Fig. 3. Boxenplots of the ER-1% roc-enrichment metric for HDBind models we present on the AVE split of the LIT-PCBA dataset. The red dashed line refers to the mean previously reported best overall (re-scoring) method on LIT-PCBA, GRIM[58,69]. The dark red dotted dash line represents the previously reported Pafnucy 3D-CNN result on LIT-PCBA[58,70]. The dotted red line denotes the mean ER-1% metric for our MLP baseline. The purple dashed line denotes our logistic regression baseline. For both Pafnucy and GRIM, we report the mean ER-1% over all 15 protein targets. For our MLP and logistic regression baselines, we report the mean over all 15 datasets and 10 random seeds. Additional sample sizes are included in SI Figures S11–S15.

information provided by the ECFP encoded into hypervectors (DECFP) with the pretrained representation extracted from the MoLFormer SMILES LLM[9]. Moreover, the performance of all HDBind molecular encoding methods tend to improve beyond the performance of GRIM and Pafnucy with increasing dimension size. To the best of our knowledge, our results, including HDB-Combo, represent the largest improvement in performance on this task that has been published to date[58].

Classification results on LIT-PCBA

We choose the MLP as our baseline in order to compare against a standard approach for downstream prediction tasks that is relatively efficient compared to more complex approaches[4,9,10,54] for which demonstrating energy efficiency would be trivial (Table 2). The MLP is trained directly on the ECFP representation to predict the binding activity of a drug molecule on each dataset, with no protein or 3D-structure information provided. The ECFP is generated using length 1024 and radius of 1. In this evaluation, we consider two splits of the dataset, a random stratified split, and the AVE split, to respectively assess model performance when making predictions on molecules *similar* to the training set and when making predictions on molecules that are maximally *dissimilar* to the training set. In Figs. 4 and 5 we characterise the effect of hypervector dimension choice D on classification performance of the active versus inactive molecules for both splits using the ROC-AUC metric. In the case of the random split, scaling D beyond 10k however does not yield increasing returns as the models appear to saturate or even degrade performance with $D = 1,000,000$. Despite this, for values of $D \geq$ 10k, nearly all models outperform our MLP baseline. However in the case of the AVE split, the benefit of increasing dimensionality is more pronounced as nearly every model considered benefits from larger vector representations, most noticeably the HDB-Combo method. Again, for values of $D \geq$ 10k, each our models outperform our MLP baseline model. We provide additional statistical significance of our results versus the baseline MLP model in SI Figures S7–S10.

Discussion

Energy efficiency as a performance metric

Energy efficiency is often over-looked in machine learning based systems in general and specifically for virtual drug screening[12]. A current trend in development often results in the use of a select set of foundation models that are expensive to train, but provide highly informative feature representations that can be leveraged for specific tasks. Increasingly these models are learned using self-supervision and result in state of the art performance in molecular property prediction[9,10]. Our setting assumes that for a given collection of drug-like molecules (purchasable libraries[90]), a feature extraction step (compute ECFP, MoLFormer or MolCLR embedding, etc.) will be performed once, resulting in a fixed cost that can be amortized over time with multiple subsequent screens performed as novel proteins are encountered and potentially complex queries over the interaction space are performed (novel viruses, mutations of known proteins, comparison of activity for new chemistry, etc.).

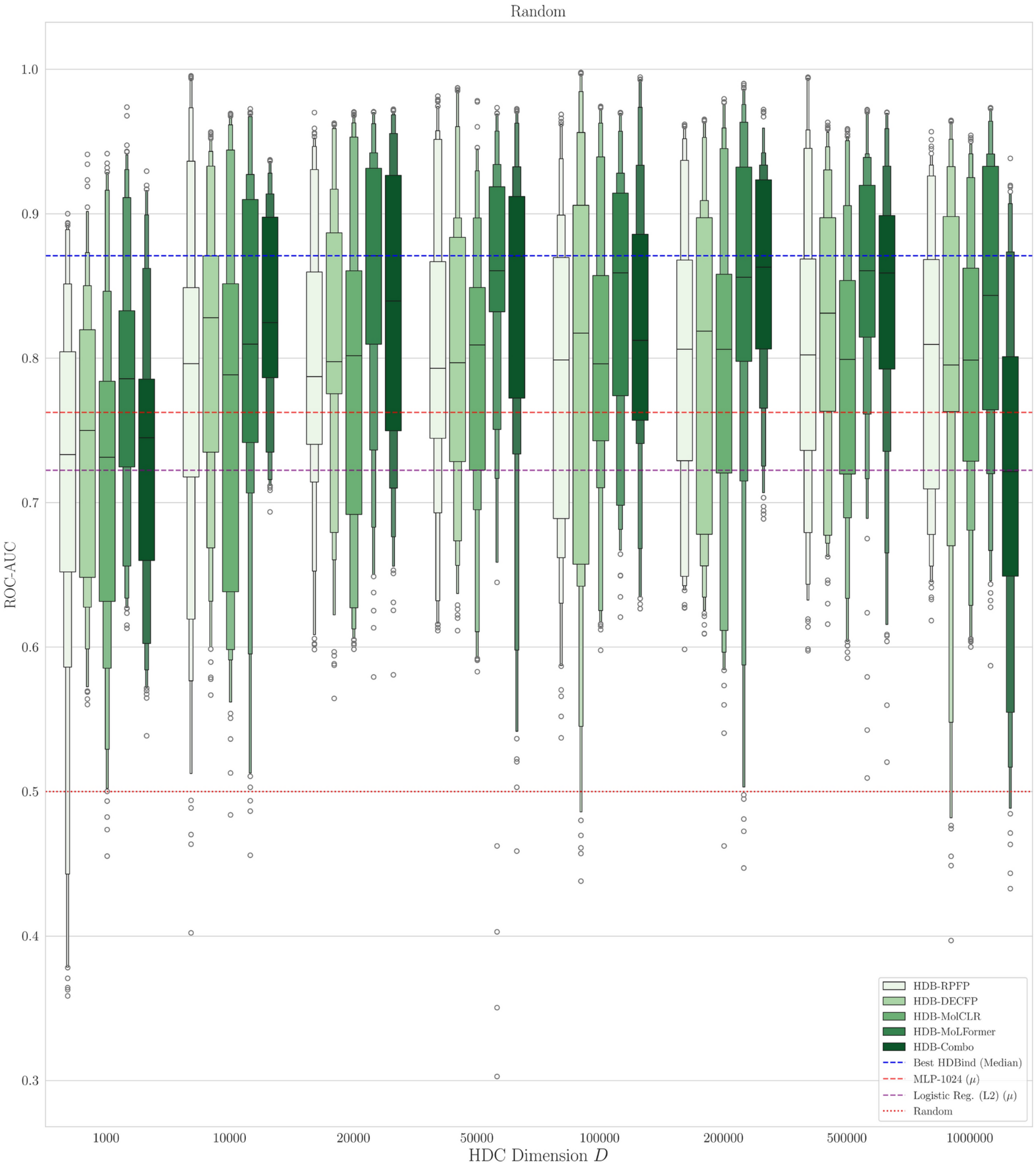

Fig. 4. Boxenplot Comparison of ROC-AUC metric across different HDB model input representations and dimension size *D* on our random split of the LIT-PCBA dataset. The red and purple dashed lines represents the mean roc-auc over all 15 datasets for our MLP and logistic regression baseline models respectively. The dotted red line denotes random performance. The blue dashed line corresponds to the best HDBind ROC-AUC distribution.

Feature extraction and encoding

We provide data describing the efficiency of the various feature extraction techniques we consider (SI Tables S5 and S6). The ECFP was found to be the most efficient representation to compute. Our results confirm that while the MoLFormer[9] embedding helps to achieve our best models on the LIT-PCBA dataset with respect to the ROC-AUC metric, the energy penalty per molecule paid to achieve these models limits their utility versus a baseline MLP model using an ECFP embedding (Table 2). We additionally consider the combination of the ECFP and MoLFormer methods by element-wise multiplication (i.e. binding) of the constituent hypervectors.

HDBind inference on FPGA hardware

The popularity of GPUs lies within their superior performance in exploiting parallelism relative to CPUs, which is further enhanced by their relative ease of programming compared to other hardware platforms[91]. FPGAs however have found utility as target platforms for energy efficient algorithm implementations due to their

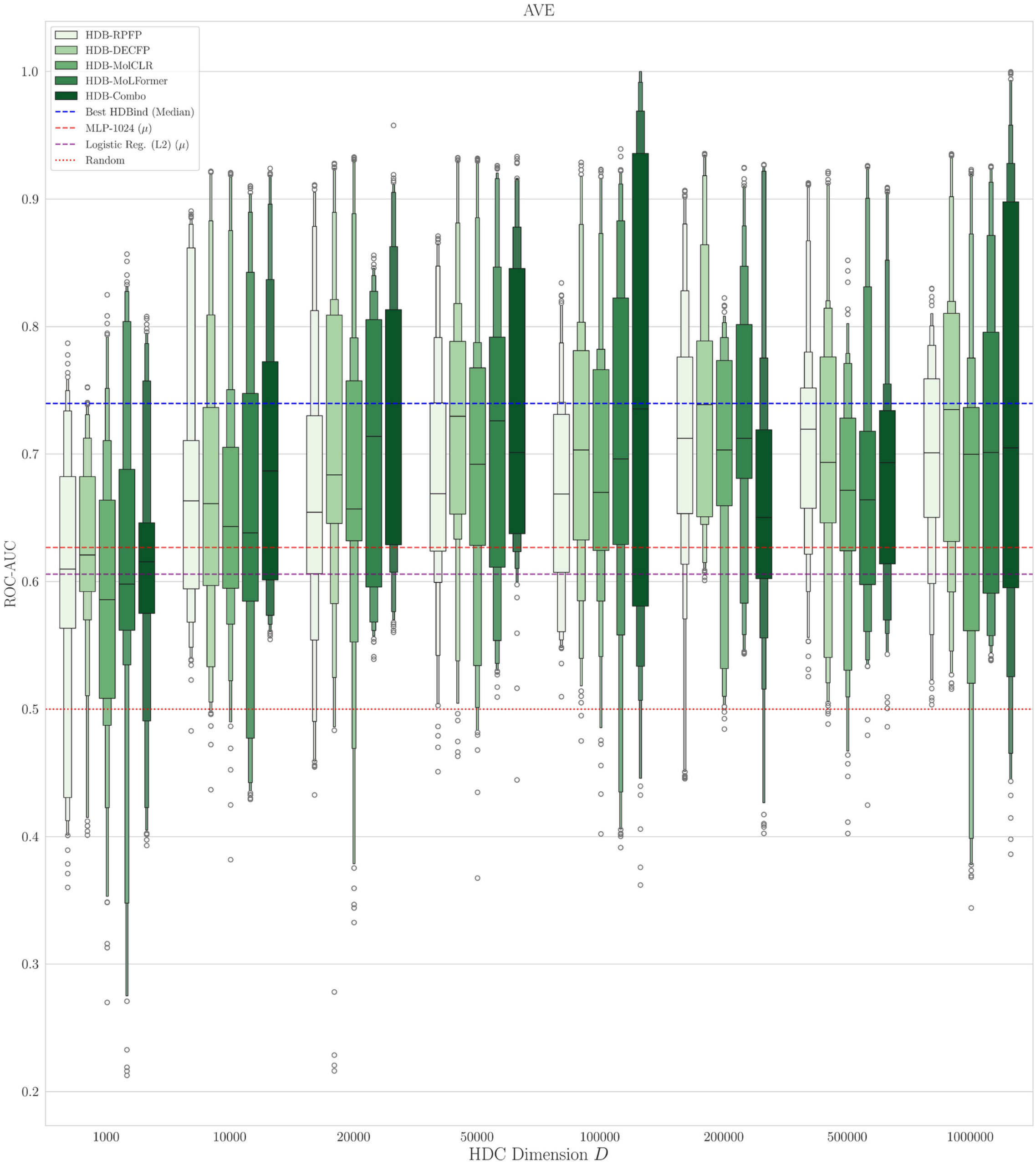

Fig. 5. Boxenplot Comparison of ROC-AUC metric across different HDB model input representations and dimension size D on the bias-minimizing AVE split of the LIT-PCBA dataset. The red and purple dashed lines represents the mean roc-auc over all 15 datasets for our MLP and logistic regression baseline models respectively. The dotted red line denotes random performance. The blue dashed line corresponds to the best HDBind ROC-AUC distribution.

relative high degree of flexibility available to developers to use on-chip resources. The price of the resource can be amortized over time by savings in energy costs versus GPU hardware implementations. In this context, we explored key components of HDBind individually, synthesizing and implementing them based on insights from HD2FPGA[92] using Vitis HLS 2021.2[93]. This approach facilitated the evaluation of these components on the Xilinx Alveo U280 FPGA, enabling the assessment of potential energy efficiency improvements over traditional computing models. The power measurements, obtained from Vitis Analyzer, provided data on the energy consumption of our implementations. While the algorithm remains consistent with that detailed in the “Materials and methods” section, its implementation on FPGA, as guided by the architectural methodologies outlined in the HD2FPGA[92], has afforded us a more granular approach to adjusting parallelism factors. This adaptability not only enhances the efficiency of our current implementations but also ensures that our approach can be scaled up for more capable future devices.

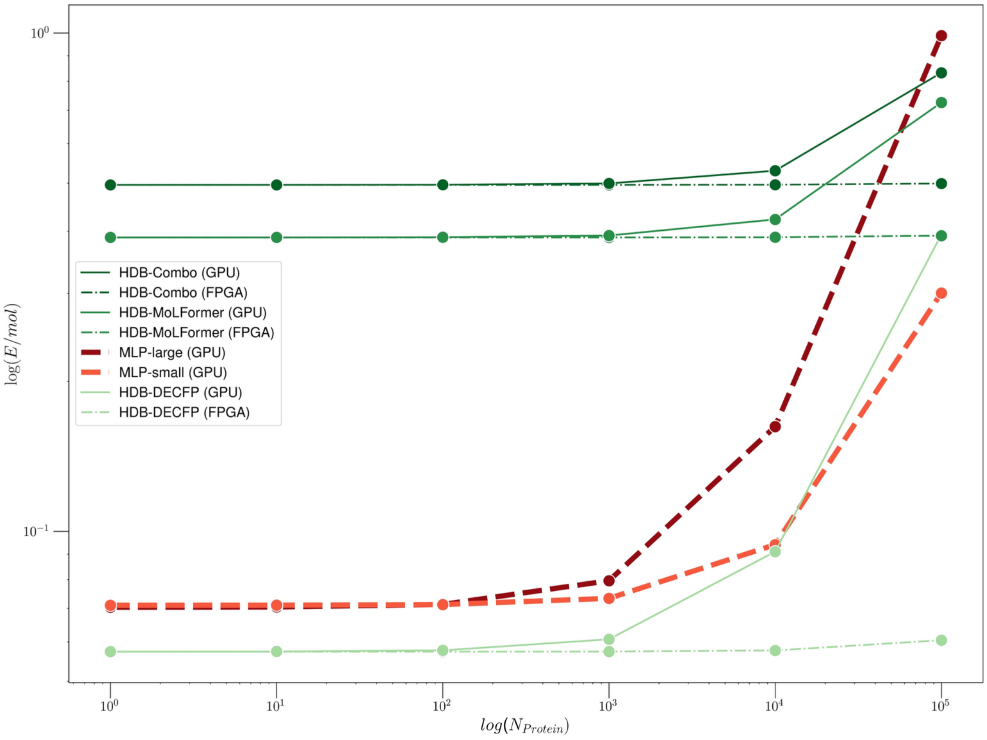

Fig. 6. Energy usage of HDBind versus our largest and smallest MLP baselines, MLP-large and MLP-small respectively, versus number of protein targets screened with a fixed library of molecules. Energy is reported in terms of expenditure per molecule. We report the mean values per molecule using the HIV dataset from MoleculeNet[11]. For each method we include the feature extraction and encoding costs for each molecule. The HDB-DECFP model immediately outperforms all methods for screens involving a single protein on both the GPU and FPGA hardware. In particular, the HDB-DECFP with FPGA inference maintains the advantage over all screen sizes we consider (SI Table S8). The HDB-MoLFormer model pays a relatively high initial encoding cost that is amortized sufficiently to outperform the MLP-large baseline at the scale of 10s of thousands of protein targets on GPU and FPGA hardware. HDB-Combo pays the highest encoding cost overall, however at the scale of 10s of thousands of proteins, becomes more efficient than MLP-large when running inference on the FPGA.

Method	Device	Test (J/mol)	Improvement factor
HDBind	FPGA[95]	0.75	12.2
HDBind	GPU[94]	3.37	2.7
MLP-small	GPU	2.30	4.0
MLP-large	GPU	9.18	1.0

Table 3. Test energy estimated for each processor choice for HDBind inference[94,95] versus the MLP baseline models. Values are scaled by 10^{-6} for reader convenience.

Energy analysis on GPU and FPGA

Figure 6 illustrates the advantage of using HDBind when presented with increasingly large collections of protein targets. For all models, we consider the energy for encoding and testing steps. We measure the power required for encoding as the sum of the average power output of the CPU and GPU. For testing, HDBind uses either a custom kernel for similarity search on the GPU[94] or FPGA[95]. We measure the power required for testing using the average power output of the GPU or FPGA in isolation. HDB-DECFP is the most efficient encoding method overall and is expectantly similar to the energy required for the MLP encodings as each are simply computing the ECFP representation (Table 2). HDB-MoLFormer and HDB-Combo each require extraction of the LLM embedding on the GPU which imposes a relatively large overall encoding penalty (~ 0.39 J/mol) that requires larger numbers of proteins (10 s of thousands) to be screened before the energy efficiency improvement of the testing step is realized versus the MLP baseline models (Table 2). Compared to inference on the GPU, the inference energy efficiency improves by a factor of $4.5\times$ when considering the FPGA for inference. The improvement on FPGA is also considerable compared to both the largest and smallest MLP architectures (approx. $12.2\times, 3.1\times$ respectively) (Table 3). As a virtual screen campaign scales beyond the consideration of a

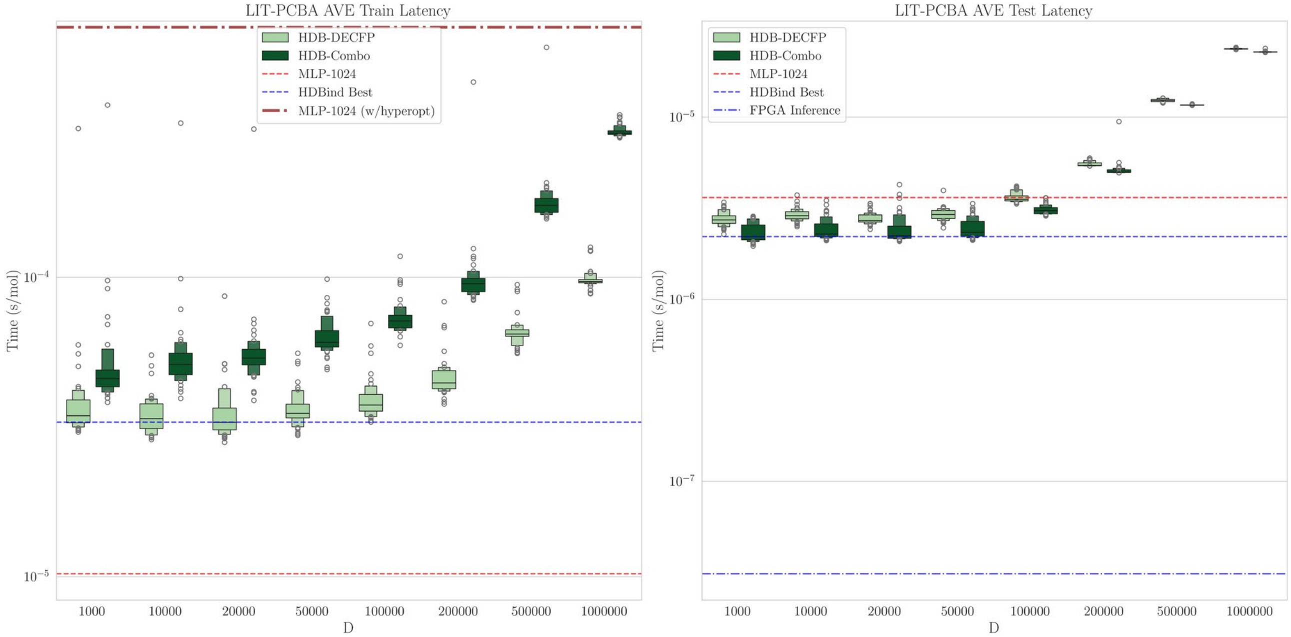

Fig. 7. Processing latency measurements on the training (left) and testing (right) sets of LIT-PCBA. Times correspond to the GPU execution time per molecule measured using the PyTorch CUDA backend. (Left) Horizontal lines for MLP and HDBind correspond to the median time for training. The HDBind methods maintain similar training times for values of D up to approximately 100k. When considering the overhead for hyperparameter optimization for the MLP, HDBind demonstrates improved latency for all values of D considered in this study, including $D = 1,000,000$. (Right) Horizontal lines for MLP and HDBind correspond to the median time for training. The horizontal line for FPGA inference represents the mean time per molecule. The HDBind methods maintain similar testing times for values of D up to 100k. Inference on the FPGA for HDBind is over an order of magnitude faster than the MLP baseline.

single protein, for a fixed library of molecules, the improvement in energy usage that is attained by HDBind on the FPGA grows by several orders of magnitude compared to the GPU-based HDBind implementation and the MLP baselines considered in our work. In Fig. 7 we give the latency of each method for the training and testing steps, normalized per molecule. When considering the overhead incurred from hyperparameter optimization of the MLP baseline, training an HDBind model becomes approximately an order of magnitude more efficient for most values of D that we considered. For testing, the HDBind models demonstrate a slight improvement over the MLP for values of D approaching 100k. When considering the FPGA for testing, the advantage grows approximately by an order of magnitude over the GPU-accelerated MLP. Given the inherent hardware-friendly characteristics of our implementation and the encouraging outcomes, advancing towards Application Specific Integrated Circuit (ASIC) development is promising, offering substantial benefits in efficiency and performance.

Conclusion

We have demonstrated the first comprehensive study of structure-based molecular encoding methods for HDC that additionally demonstrate consistent improvement over the competing SOA SMILES-based approaches. We additionally are the first to consider the use of molecular foundation models with HDC that leverage self-supervised learning on large unlabelled collections of SMILES strings and molecular graphs as input, demonstrating an improvement over SOA and ECFP-based approaches. Additionally we compare the performance of all methods on a broad collection of quantitative and qualitative molecular property prediction tasks in the well-studied MoleculeNet. We are the first to demonstrate the viability of HDC on a challenging benchmark protein-drug activity dataset, LIT-PCBA, which outperform physics-based molecular docking while being competitive with our MLP baseline method. The analysis of the computational burden and energy usage show clear advantages to using HDC approaches. We leave it to future work to improve the decision boundary learned by the HDC model, one approach employs metric learning to pretrain the projection layer, as considered in previous HDC works[37]. Improving the latency of the feature extraction and encoding steps can help improve the efficiency of the overall system[96]. In the case of the self-supervised representations we considered, MolCLR and MoLFormer, improving the energy efficiency of these steps will lower the threshold at which these methods will become more attractive compared to our MLP baseline. Our code is available for use with the publicly available datasets, to enable reproduction of our study at https://github.com/LLNL/hdbind.

Data availability

All code is publicly available on github https://github.com/LLNL/hdbind. All MoleculeNet data is available at https://moleculenet.org/ and all LIT-PCBA data is available at https://drugdesign.unistra.fr/LIT-PCBA/. Additi onal reasonable requests can be made by contacting the corresponding authors.

Received: 27 July 2024; Accepted: 14 November 2024
Published online: 23 November 2024

References

1. Schneider, P. et al. Rethinking drug design in the artificial intelligence era. *Nat. Rev. Drug Discov.* **19**, 353–364 (2020).
2. Yu, Y. et al. Uni-Dock: GPU-Accelerated docking enables ultralarge virtual screening. *J. Chem. Theory Comput.* **19**, 3336–3345 (2023).
3. Volkov, M. et al. On the frustration to predict binding affinities from Protein–Ligand structures with deep neural networks. *J. Med. Chem.* (2022).
4. Jones, D. et al. Improved Protein–Ligand binding affinity prediction with structure-based deep fusion inference. *J. Chem. Inf. Model.* **61**, 1583–1592 (2021).
5. Minnich, A. J. et al. AMPL: A data-driven modeling pipeline for drug discovery. *J. Chem. Inf. Model.* **60**, 1955–1968 (2020).
6. Grygorenko, O. O. et al. Generating multibillion chemical space of readily accessible screening compounds. *iScience* **23**, 101681 (2020).
7. Database, A. P. S. AlphaFold database. https://alphafold.ebi.ac.uk/ (accessed 13 Nov 2023).
8. Baek, M. et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science* **373**, 871–876 (2021).
9. Ross, J. et al. Large-scale chemical language representations capture molecular structure and properties. *Nat. Mach. Intell.* **4**, 1256–1264 (2022).
10. Wang, Y., Wang, J., Cao, Z. & Barati Farimani, A. Molecular contrastive learning of representations via graph neural networks. *Nat. Mach. Intell.* **4**, 279–287 (2022).
11. Wu, Z. et al. MoleculeNet: A benchmark for molecular machine learning. *Chem. Sci.* **9**, 513–530 (2018).
12. Schwartz, R., Dodge, J., Smith, N. A. & Etzioni, O. Green AI. *Commun. ACM* **63**, 54–63 (2020).
13. Plate, T. A. Holographic reduced representations. *IEEE Trans. Neural Netw.* **6**, 623–641 (1995).
14. Kanerva, P. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognit. Comput.* **1**, 139–159 (2009).
15. Thomas, A., Dasgupta, S. & Rosing, T. A theoretical perspective on hyperdimensional computing. *J. Artif. Intell. Res.* **72**, 215–249 (2021).
16. Karunaratne, G. et al. In-memory hyperdimensional computing. arXiv. arXiv 1906, 01548 (2019).
17. Ge, L. & Parhi, K. K. Classification using hyperdimensional computing: A review. *IEEE Circuits Syst. Mag.* **20**, 30–47 (2020).
18. Rahimi, A., Kanerva, P., Benini, L. & Rabaey, J. M. Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of ExG signals. *Proc. IEEE* **107**, 123–143 (2019).
19. Burrello, A., Cavigelli, L., Schindler, K., Benini, L. & Rahimi, A. Laelaps: An Energy-Efficient seizure detection algorithm from long-term human iEEG recordings without false alarms. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 752–757 (2019).
20. Rasanen, O. J. & Saarinen, J. P. Sequence prediction with sparse distributed hyperdimensional coding applied to the analysis of mobile phone use patterns. *IEEE Trans. Neural Netw. Learn. Syst.* **27**, 1878–1889 (2016).
21. Mitrokhin, A., Sutor, P., Fermüller, C. & Aloimonos, Y. Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception. *Sci. Robot.* **4** (2019).
22. Salamat, S., Imani, M. & Rosing, T. Accelerating hyperdimensional computing on FPGAs by exploiting computational reuse. *IEEE Trans. Comput.* **69**, 1159–1171 (2020).
23. Kanerva, P., Kristoferson, J. & Holst, A. Random indexing of text samples for latent semantic analysis. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 22 (2000).
24. Kang, J., Khaleghi, B., Kim, Y. & Rosing, T. Xcelhd: An efficient gpu-powered hyperdimensional computing with parallelized training. In *The 27th Asia and South Pacific Design Automation Conference* (2022).
25. Kang, J. *et al.* RelHD: A graph-based learning on FeFET with hyperdimensional computing. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*, 553–560 (2022).
26. Pinge, S. *et al.* SpecHD: Hyperdimensional computing framework for FPGA-based mass spectrometry clustering. *arXiv* (2023). arXiv:2311.12874.
27. Kazemi, A. et al. Achieving software-equivalent accuracy for hyperdimensional computing with ferroelectric-based in-memory computing. *Sci. Rep.* **12**, 19201 (2022).
28. Xu, W., Kang, J., Bittremieux, W., Moshiri, N. & Rosing, T. HyperSpec: Ultrafast mass spectra clustering in hyperdimensional space. *J. Proteome Res.* **22**, 1639–1648 (2023).
29. Ma, D., Thapa, R. & Jiao, X. MoleHD: Efficient drug discovery using brain inspired hyperdimensional computing. In *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 390–393 (2022).
30. Abhijith, M. & Nair, D. R. Neuromorphic high dimensional computing architecture for classification applications. In *2021 IEEE International Conference on Nanoelectronics, Nanophotonics, Nanomaterials, Nanobioscience & Nanotechnology (5NANO)*, 1–10 (IEEE, 2021).
31. Rahimi, A. et al. High-dimensional computing as a nanoscalable paradigm. *IEEE Trans. Circuits Syst. I Regul. Pap.* **64**, 2508–2521 (2017).
32. Rogers, D. & Hahn, M. Extended-connectivity fingerprints. *J. Chem. Inf. Model.* **50**, 742–754 (2010).
33. Ma, D. & Jiao, X. Hyperdimensional computing vs. neural networks: Comparing architecture and learning process. *arXiv*[SPACE]arXiv:2207.12932 (2022).
34. Tran-Nguyen, V.-K., Jacquemard, C. & Rognan, D. LIT-PCBA: An unbiased data set for machine learning and virtual screening. *J. Chem. Inf. Model.* (2020).
35. Kainen, P. C. Utilizing geometric anomalies of high dimension: When complexity makes computation easier. In *Computer Intensive Methods in Control and Signal Processing: The Curse of Dimensionality*(eds. Kárný, M. & Warwick, K.) 283–294 (Birkhäuser Boston, 1997).
36. Yu, T., Zhang, Y., Zhang, Z. & De Sa, C. Understanding hyperdimensional computing for parallel Single-Pass learning. *arXiv* (2022). arXiv:2202.04805.
37. Xu, S., Pinge, F. & Rosing. HyperMetric: Robust hyperdimensional computing on error-prone memories using metric learning. In *2023 IEEE 41st International Conference on Computer Design (ICCD)*, vol. 0, 243–246 (2023).
38. Gupta, S. *et al.* THRIFTY: training with hyperdimensional computing across flash hierarchy. In *Proceedings of the 39th International Conference on Computer-Aided Design*, no. Article 27 in ICCAD '20, 1–9 (Association for Computing Machinery, 2020).

39. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
40. Kingma, D. P. & Ba, J. A method for stochastic optimization. arXiv, Adam (2014) arXiv:1412.6980.
41. You, Y. *et al.* Large batch optimization for deep learning: Training BERT in 76 minutes. *arXiv*. arXiv:1904.00962 (2019).
42. Weininger, D. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **28**, 31–36 (1988).
43. Morgan, H. L. The generation of a unique machine description for chemical Structures-A technique developed at chemical abstracts service. *J. Chem. Doc.* **5**, 107–113 (1965).
44. Dasgupta, S. Experiments with random projection. In *Proceedings of the sixteenth conference on uncertainty in artificial intelligence*, UAI'00, 143–151 (Morgan Kaufmann Publishers Inc., 2000).
45. Bingham, E. & Mannila, H. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, 245–250 (Association for Computing Machinery, 2001).
46. Landrum, G. *et al.* rdkit/rdkit: 2021_09_2 (q3 2021) release (2021).
47. Duvenaud, D. K. *et al.* Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems 28* (eds. Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M. & Garnett, R.) 2224–2232 (Curran Associates, Inc., 2015).
48. Chithrananda, S., Grand, G. & Ramsundar, B. ChemBERTa: Large-Scale Self-Supervised pretraining for molecular property prediction. *arXiv* (2020). arXiv:2010.09885.
49. Liu, Y. *et al.* RoBERTa: A robustly optimized BERT pretraining approach. *arXiv* (2019). arXiv:1907.11692.
50. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (eds. Burstein, J., Doran, C. & Solorio, T.) 4171–4186 (Association for Computational Linguistics, 2019).
51. Sun, F. -Y., Hoffmann, J., Verma, V. & Tang, J. InfoGraph: Unsupervised and semi-supervised Graph-Level representation learning via mutual information maximization. *arXiv*. arXiv:1908.01000 (2019).
52. Dutta, A. *et al.* HDnn-PIM: Efficient in memory design of hyperdimensional computing with feature extraction. In *Proceedings of the Great Lakes Symposium on VLSI 2022, GLSVLSI '22*, 281–286 (Association for Computing Machinery, 2022).
53. Bender, A. & Glen, R. C. A discussion of measures of enrichment in virtual screening: comparing the information content of descriptors with increasing levels of sophistication. *J. Chem. Inf. Model.* **45**, 1369–1375 (2005).
54. Gentile, F. et al. Deep docking: A deep learning platform for augmentation of structure based drug discovery. *ACS Cent Sci* **6**, 939–949 (2020).
55. Mysinger, M. M., Carchia, M., Irwin, J. J. & Shoichet, B. K. Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking. *J. Med. Chem.* **55**, 6582–6594 (2012).
56. Jain, A. N. & Nicholls, A. Recommendations for evaluation of computational methods. *J. Comput. Aided Mol. Des.* **22**, 133–139 (2008).
57. Cleves, A. E. & Jain, A. N. Structure- and ligand-based virtual screening on DUD-E+: Performance dependence on approximations to the binding pocket. *J. Chem. Inf. Model.* **60**, 4296–4310 (2020).
58. Tran-Nguyen, V.-K., Bret, G. & Rognan, D. True accuracy of fast scoring functions to predict High-Throughput screening data from docking poses: The simpler the better. *J. Chem. Inf. Model.* **61**, 2788–2797 (2021).
59. Liaw, R. *et al.* Tune: A research platform for distributed model selection and training. *arXiv preprint*[SPACE]arXiv:1807.05118 (2018).
60. Variorum: Vendor-agnostic computing power management.
61. Liu, S., Demirel, M. F. & Liang, Y. N-gram graph: Simple unsupervised representation for graphs, with applications to molecules. *Adv. Neural Inf. Process. Syst.* 8464–8476 (2018).
62. Li, S., Zhou, J., Xu, T., Dou, D. & Xiong, H. GeomGCL: Geometric graph contrastive learning for molecular property prediction. *arXiv*. arXiv:2109.11730 (2021).
63. Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. A simple framework for contrastive learning of visual representations. *arXiv*. arXiv:2002.05709 (2020).
64. Kim, S. et al. PubChem 2023 update. *Nucleic Acids Res.* **51**, D1373–D1380 (2023).
65. Lu, C. et al. Molecular property prediction: A multilevel quantum interactions modeling perspective. *AAAI* **33**, 1052–1060 (2019).
66. Yang, K. et al. Analyzing learned molecular representations for property prediction. *J. Chem. Inf. Model.* **59**, 3370–3388 (2019).
67. Li, X. & Fourches, D. SMILES pair encoding: A Data-Driven substructure tokenization algorithm for deep learning. *J. Chem. Inf. Model.* **61**, 1560–1569 (2021).
68. Zdrazil, B. et al. The ChEMBL database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods. *Nucleic Acids Res.* **52**, D1180–D1192 (2024).
69. Desaphy, J., Raimbaud, E., Ducrot, P. & Rognan, D. Encoding protein-ligand interaction patterns in fingerprints and graphs. *J. Chem. Inf. Model.* **53**, 623–637 (2013).
70. Stepniewska-Dziubinska, M. M., Zielenkiewicz, P. & Siedlecki, P. Development and evaluation of a deep learning model for protein-ligand binding affinity prediction. *Bioinformatics* **34**, 3666–3674 (2018).
71. Stafford, K. A., Anderson, B. M., Sorenson, J. & van den Bedem, H. AtomNet PoseRanker: Enriching ligand pose quality for dynamic proteins in virtual High-Throughput screens. *J. Chem. Inf. Model.* **62**, 1178–1189 (2022).
72. Clyde, A. et al. AI-accelerated protein-ligand docking for SARS-CoV-2 is 100-fold faster with no significant change in detection. *Sci. Rep.* **13**, 2105 (2023).
73. Clyde, A. et al. High-Throughput virtual screening and validation of a SARS-CoV-2 main protease noncovalent inhibitor. *J. Chem. Inf. Model.* **62**, 116–128 (2022).
74. Lau, E. Y. et al. Discovery of Small-Molecule inhibitors of SARS-CoV-2 proteins using a computational and experimental pipeline. *Front. Mol. Biosci.* **8**, 678701 (2021).
75. Stevenson, G. A. *et al.* High-throughput virtual screening of small molecule inhibitors for SARS-CoV-2 protein targets with deep fusion models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, no. Article 74 in SC '21, 1–13 (Association for Computing Machinery, 2021).
76. Trott, O. & Olson, A. J. AutoDock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J. Comput. Chem.* **31**, 455–461 (2010).
77. Eberhardt, J., Santos-Martins, D., Tillack, A. F. & Forli, S. AutoDock vina 1.2.0: New docking methods, expanded force field, and python bindings. *J. Chem. Inf. Model.* **61**, 3891–3898 (2021).
78. Massova, I. & Kollman, P. A. Combined molecular mechanical and continuum solvent approach (MM-PBSA/GBSA) to predict ligand binding. *Perspect. Drug Discov. Des.* **18**, 113–135 (2000).
79. Greenidge, P. A., Kramer, C., Mozziconacci, J.-C. & Wolf, R. M. MM/GBSA binding energy prediction on the PDBbind data set: successes, failures, and directions for further improvement. *J. Chem. Inf. Model.* **53**, 201–209 (2013).
80. Wright, D. W., Hall, B. A., Kenway, O. A., Jha, S. & Coveney, P. V. Computing clinically relevant binding free energies of HIV-1 protease inhibitors. *J. Chem. Theory Comput.* **10**, 1228–1241 (2014).
81. Huang, N., Shoichet, B. K. & Irwin, J. J. Benchmarking sets for molecular docking. *J. Med. Chem.* **49**, 6789–6801 (2006).

82. Chaput, L., Martinez-Sanz, J., Saettel, N. & Mouawad, L. Benchmark of four popular virtual screening programs: construction of the active/decoy dataset remains a major determinant of measured performance. *J. Cheminform.* **8**, 56 (2016).
83. Wallach, I. & Heifets, A. Most Ligand-Based classification benchmarks reward memorization rather than generalization. *J. Chem. Inf. Model.* **58**, 916–932 (2018).
84. Chen, L. et al. Hidden bias in the DUD-E dataset leads to misleading performance of deep learning in structure-based virtual screening. *PLoS One* **14**, e0220113 (2019).
85. Sieg, J., Flachsenberg, F. & Rarey, M. In need of bias control: Evaluating chemical data for machine learning in Structure-Based virtual screening. *J. Chem. Inf. Model.* **59**, 947–961 (2019).
86. Jiang, D. et al. InteractionGraphNet: A novel and efficient deep graph representation learning framework for accurate Protein–Ligand interaction predictions. *J. Med. Chem.* **64**, 18209–18232 (2021).
87. Jain, A. N. Surflex-Dock 2.1: robust performance from ligand energetic modeling, ring flexibility, and knowledge-based search. *J. Comput. Aided Mol. Des.* **21**, 281–306 (2007).
88. Su, M. et al. Comparative assessment of scoring functions: The CASF-2016 update. *J. Chem. Inf. Model.* **59**, 895–913 (2019).
89. Li, X. & Fourches, D. SMILES pair encoding: A Data-Driven substructure tokenization algorithm for deep learning. *J. Chem. Inf. Model.* **61**, 1560–1569 (2021).
90. REAL compounds—enamine. https://enamine.net/compound-collections/real-compounds. (accessed 05 Oct 2021).
91. Jones, D. et al. Accelerators for classical molecular dynamics simulations of biomolecules. *J. Chem. Theory Comput.* **18**, 4047–4069 (2022).
92. Zhang, T. *et al.* Hd2fpga: Automated framework for accelerating hyperdimensional computing on fpgas. In *2023 24th International Symposium on Quality Electronic Design (ISQED)*, 1–9. https://doi.org/10.1109/ISQED57927.2023.10129332 (2023).
93. AMD technical information portal. https://docs.amd.com/r/en-US/ug1399-vitis-hls. (accessed 30 May 2024).
94. Kang, J., Khaleghi, B., Rosing, T. & Kim, Y. OpenHD: A GPU-Powered framework for hyperdimensional computing. *IEEE Trans. Comput.* **71**, 2753–2765 (2022).
95. Zhang, T. *et al.* HD2FPGA: Automated framework for accelerating hyperdimensional computing on FPGAs. In *2023 24th International Symposium on Quality Electronic Design (ISQED)*, 1–9 (IEEE, 2023).
96. Gu, Y., Dong, L., Wei, F. & Huang, M. MiniLLM: Knowledge distillation of large language models. *arXiv*. arXiv:2306.08543 (2023).

Acknowledgements

This research was supported by funds from the UC National Laboratory Fees Research Program of the University of California, Grant Number L23GF6259. This work was supported in part by CRISP and PRISM, centers in JUMP 1.0 and 2.0, SRC programs sponsored by DARPA, SRC #236160. Computing support for this work came from the Lawrence Livermore National Laboratory (LLNL) Institutional Computing Grand Challenge program. Part of this research was also supported by the American Heart Association under CRADA TC02274-4. Funding in part by DTRA project HDTRA1036045. All work performed at Lawrence Livermore National Laboratory is performed under the auspices of the U.S. Department of Energy under Contract DE-AC52-07NA27344, LLNL-JRNL-847376. We thank Michael K. Gilson (UCSD), Rose Yu (UCSD), Stewart He (LLNL), Dan Kirshner (LLNL), Kevin McLoughlin (LLNL), and Amanda Paulson (UCSF) for their helpful feedback in the development of this work.

Author contributions

D.J., J.A., N.M., and T.R. conceived the experiments. D.J. conducted the experiments and wrote the draft manuscript. D.J., J.A., N.M., and T.R. edited the manuscript. X.Z. and B.B. contributed molecular docking simulations and their data. J.K., B.K., and W.X. contributed code. S.P. contributed hardware energy efficiency measurements. All authors read and approved the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-80009-w.

Correspondence and requests for materials should be addressed to D.J.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.