Context-enriched molecule representations improve few-shot drug discovery

Anonymous Author(s)
Affiliation
Address
email

Abstract

A central task in computational drug discovery is to construct models from known active molecules to find further promising molecules for subsequent screening. However, typically only very few active molecules are known. Therefore, few-shot learning methods have the potential to improve the effectiveness of this critical phase of the drug discovery process. We introduce a new method for few-shot drug discovery. Its main idea is to enrich a molecule representation by knowledge about known context or reference molecules. Our novel concept for molecule representation enrichment is to associate molecules from both the support set and the query set with a large set of reference (context) molecules through a modern Hopfield network. Intuitively, this enrichment step is analogous to a human expert who would associate a given molecule with familiar molecules whose properties are known. The enrichment step reinforces and amplifies the covariance structure of the data, while simultaneously removing spurious correlations arising from the decoration of molecules. Our approach is compared with other few-shot methods for drug discovery on the FS-Mol benchmark dataset. On FS-Mol, our approach outperforms all compared methods and therefore sets a new state-of-the art for few-shot learning in drug discovery. An ablation study shows that the enrichment step of our method is the key to improve the predictive quality. In a domain shift experiment, we further demonstrate the robustness of our method.

1 Introduction

To improve human health, combat diseases, and tackle pandemics there is a steady need of discovering new drugs in a fast and efficient way. However, the drug discovery process is time-consuming and cost-intensive (Arrowsmith, 2011). Deep learning methods have recently been shown to reduce time and costs of this process (Chen et al., 2018; Walters and Barzilay, 2021). They diminish the required number of both wet-lab measurements and molecules that must be synthesized (Merk et al., 2018; Schneider et al., 2020). However, as of now, deep learning approaches use only the molecular information about the ligands after being trained on a large training set. At inference time, they yield highly accurate property and activity prediction (Mayr et al., 2018; Yang et al., 2019), generative (Segler et al., 2018a; Gómez-Bombarelli et al., 2018), or synthesis models (Segler et al., 2018b; Seidl et al., 2022).

Deep learning methods in drug discovery usually require large amounts of biological measurements. To train deep learning-based activity and property prediction models with high predictive performance, hundreds or thousands of data points per task are required. For example, well-performing predictive models for activity prediction tasks of ChEMBL have been trained with an average of 3,621 activity points per task, i.e., drug target, by (Mayr et al., 2018). The ExCAPE-DB dataset provides on average 42,501 measurements per task (Sun et al., 2017; Sturm et al., 2020). (Wu et al., 2018) pub-

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

lished a large scale benchmark for molecular machine learning, including prediction models for the SIDER dataset (Kuhn et al., 2016) with an average of 5,187 data points, Tox21 (Huang et al., 2016b; Mayr et al., 2016) with on average 9,031, and ClinTox (Wu et al., 2018) with 1,491 measurements per task. However, for typical drug design projects, the amount of available measurements is very limited (Stanley et al., 2021; Waring et al., 2015; Hochreiter et al., 2018), since in-vitro experiments are expensive and time-consuming. Therefore, methods that need only few measurements to build precise prediction models are desirable. This problem — i.e., the challenge of learning from few data points — is the focus of machine learning areas like meta-learning (Schmidhuber, 1987; Bengio et al., 1991; Hochreiter et al., 2001) and few-shot learning (Miller et al., 2000; Bendre et al., 2020; Wang et al., 2020).

Few-shot learning tackles the low-data problem that is ubiquitous in drug discovery. Few-shot learning methods have been predominantly developed and tested on image datasets (Bendre et al., 2020; Wang et al., 2020), and have recently been adapted to drug discovery problems (Chen et al., 2022; Wang et al., 2021; Stanley et al., 2021; Altae-Tran et al., 2017). They are usually categorized into three groups according to their main approach (Bendre et al., 2020; Wang et al., 2020; Adler et al., 2020). a) Data-augmentation-based approaches augment the available samples and generate new, more diverse data points (Chen et al., 2020; Zhao et al., 2019; Antoniou and Storkey, 2019). b) Embedding-based and nearest neighbour approaches learn embedding space representations. Predictive models can then be constructed from only few net data points by comparing these embeddings. For example in Matching Networks (Vinyals et al., 2016) an attention mechanism that relies on embeddings is the basis for the predictions. Prototypical Networks (Snell et al., 2017) create prototype representations for each class using the above mentioned representations in the embedding space. c) Optimization-based or fine-tuning methods utilize a meta-optimizer that focuses on efficiently navigating the parameter space. For example, with MAML the meta-optimizer learns initial weights that can be adapted to a novel task by few optimization steps (Finn et al., 2017).

Most of these approaches have already been applied to few-shot drug discovery (see Sec. 4). Surprisingly, almost all these few-shot learning methods in drug discovery are worse than a naive baseline, which does not even use the support set (see Section 5). We hypothesize that the under-performance of these methods stems from disregarding the context — both in terms of similar molecules and similar activities. Therefore, we propose a method that informs the representations of the query and support set with a large number of context molecules covering the chemical space.

Enriching molecule representations with context using associative memories. In data-scarce situations, humans extract co-occurrences and covariances by associating current perceptions with memories (Bonner and Epstein, 2021; Potter, 2012). When we show a small set of active molecules to a human expert in drug discovery, the expert associates them with known molecules to suggest further active molecules (Gomez, 2018; He et al., 2021). In an analogous manner, our novel concept for few-shot learning uses associative memories to extract co-occurrences and the covariance structure of the original data and to amplify them in the representations (Fürst et al., 2021). We use Modern Hopfield Networks (MHNs) as an associative memory, since they can store a large set of context molecule representations (Ramsauer et al., 2021, Theorem 3). The representations that are retrieved from the MHNs replace the original representations of the query and support set molecules. Those retrieved representations have amplified co-occurrences and covariance structures, while peculiarities and spurious co-occurrences of the query and support set molecules are averaged out.

In this work, our contributions are the following:

- We propose a new architecture **MHNfs** for few-shot learning in drug discovery.
- We achieve a new state-of-the-art on the benchmarking dataset FS-Mol.
- We introduce a novel concept to enrich the molecule representations with context by associating them with a large set of context molecules.
- We add a naive baseline to the FS-Mol benchmark that yields better results than almost all other published few-shot learning methods.
- We provide results of an ablation study and a domain shift experiment to further demonstrate the effectiveness of our new method.

2 Problem setting

Drug discovery projects revolve around models $g(\boldsymbol{m})$ that can predict a molecular property or activity $\hat{y}$, given a representation $\boldsymbol{m}$ of an input molecule from a chemical space $\mathcal{M}$. We consider machine learning models $\hat{y} = g_{\boldsymbol{w}}(\boldsymbol{m})$ with parameters $\boldsymbol{w}$ that have been selected using a training set. Typically, deep learning based property prediction uses a molecule encoder $f^{\mathrm{ME}} : \mathcal{M} \to \mathbb{R}^d$. The molecule encoder can process different symbolic or low-level representations of molecules, such as molecular descriptors (Bender et al., 2004; Unterthiner et al., 2014; Mayr et al., 2016), SMILES (Weininger, 1988; Mayr et al., 2018; Winter et al., 2019; Segler et al., 2018a), or molecular graphs (Merkwirth and Lengauer, 2005; Kearnes et al., 2016; Yang et al., 2019; Jiang et al., 2021) and can be pre-trained on related property prediction tasks.

For few-shot learning, the goal is to select a high-quality predictive model based on a small set of molecules $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ with associated measurements $\boldsymbol{y} = \{y_1, \ldots, y_N\}$. The measurements are usually assumed to be binary $y_n \in \{-1, 1\}$, corresponding to the molecule being inactive or active. The set $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ is called the *support set* that contains samples from a prediction task and N is the *support set size*. The goal is to construct a model that correctly predicts y for an $\boldsymbol{x}$ that is not in the support set — in other words, a model that generalizes well.

Standard supervised machine learning approaches typically just show limited predictive power at this task (Stanley et al., 2021) since they tend to overfit on the support set due to a small number of training samples. These approaches learn the parameters $\boldsymbol{w}$ of the model $g_{\boldsymbol{w}}$ from the support set in a supervised manner. However, they heavily overfit to the support set when N is small. Therefore, few-shot learning methods are necessary to construct models from the support set that generalize well to new data.

3 MHNfs: Hopfield-based molecular context enrichment for few-shot drug discovery

We aim at increasing the generalization capabilities of few-shot learning methods in drug discovery by enriching the molecule representations with molecular context. In comparison to the support set, which encodes information about the task, the context set – i.e. a large set of molecules – includes information about a large chemical space. The query and the support set molecules perform a retrieval from the context set and thereby enrich their representations. We detail this in the following.

3.1 Model architecture

We propose an architecture which consists of three consecutive modules. The first module, a) the *context module* f^{CM}, enriches molecule representations by retrieving from a large set of molecules. The second module, b) the *cross-attention module* f^{CAM} (Hou et al., 2019; Chen et al., 2021), enables the effective exchange of information between the query molecule and the support set molecules. Finally the prediction for the query molecule is computed by using the usual c) *similarity module* f^{SM} (Koch et al., 2015; Altae-Tran et al., 2017):

$$\begin{aligned}
\text{context module:} \qquad \boldsymbol{m}' &= f^{\mathrm{CM}}(\boldsymbol{m}, \boldsymbol{C}) \\
\boldsymbol{X}' &= f^{\mathrm{CM}}(\boldsymbol{X}, \boldsymbol{C}), && (1) \\
\text{cross-attention module:} \qquad [\boldsymbol{m}'', \boldsymbol{X}''] &= f^{\mathrm{CAM}}([\boldsymbol{m}', \boldsymbol{X}']), && (2) \\
\text{similarity module:} \qquad \hat{y} &= f^{\mathrm{SM}}(\boldsymbol{m}'', \boldsymbol{X}'', \boldsymbol{y}), && (3)
\end{aligned}$$

where $\boldsymbol{m} \in \mathbb{R}^d$ is a molecule embedding from a trainable or fixed molecule encoder, and $\boldsymbol{m}'$ and $\boldsymbol{m}''$ are enriched versions of it. Similarly, $\boldsymbol{X} \in \mathbb{R}^{d \times N}$ contains the stacked embeddings of the support set molecules and $\boldsymbol{X}'$ and $\boldsymbol{X}''$ are their enriched versions. $\boldsymbol{C} \in \mathbb{R}^{d \times M}$ is a large set of stacked molecule embeddings, $\boldsymbol{y}$ are the support set labels, and $\hat{y}$ is the prediction for the query molecule. Square brackets indicate concatenation, for example $[\boldsymbol{m}', \boldsymbol{X}']$ is a matrix with $N + 1$ columns. The modules f^{CM}, f^{CAM}, and f^{SM} are detailed in the paragraphs below. An overview of our architecture is given in Figure 1. The architecture also includes skip connections bypassing $f^{\mathrm{CM}}(.,.)$ and $f^{\mathrm{CAM}}(.)$ and layer normalization (Ba et al., 2016), which are not shown in Figure1.

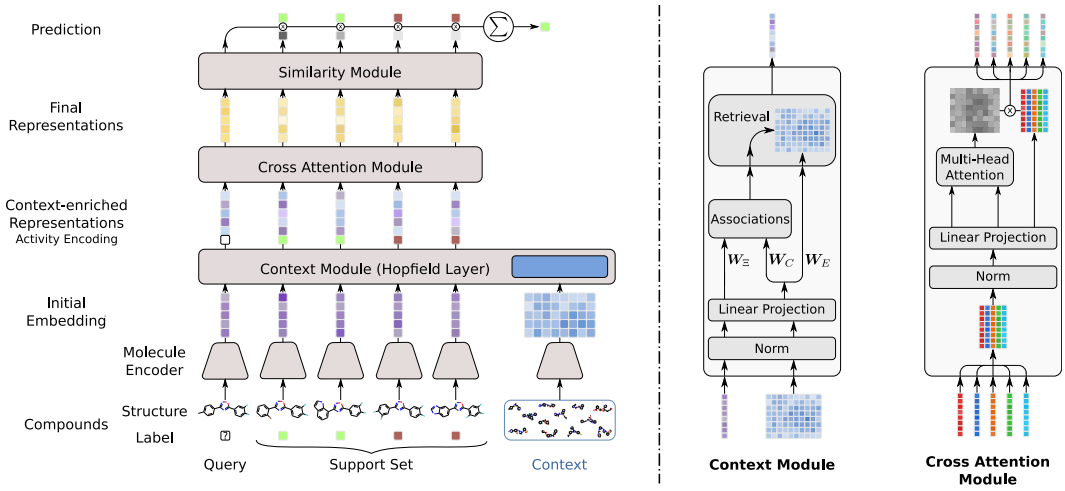

Figure 1: Schematic overview of our architecture. **Left:** All molecules are fed through a shared molecule encoder to obtain embeddings. Then, the context module (CM) enriches the representations by associating them with context molecules. The cross-attention module (CAM) enriches representations by mutually associating the query and support set molecules. Finally, the similarity module computes the prediction for the query molecule. **Right:** Detailed depiction of the operations in the CM and the CAM.

A shared molecule encoder f^{ME} creates embeddings for the query molecule $\boldsymbol{m} = f^{\mathrm{ME}}(m)$, the support set molecules $\boldsymbol{x}_n = f^{\mathrm{ME}}(x_n)$, and the context molecules $\boldsymbol{c}_m = f^{\mathrm{ME}}(c_m)$. There are many possible choices for fixed or adaptive molecule encoders (see Section 2), of which we use descriptor-based fully-connected networks because of their computational efficiency and good accuracy (Dahl et al., 2014; Mayr et al., 2016, 2018). For notational clarity we denote the course of the representations through the architecture:

$$\underset{\substack{\text{symbolic or}\\ \text{low-level repr.}}}{m} \xrightarrow{f^{\mathrm{ME}}} \underset{\substack{\text{molecule}\\ \text{embedding}}}{\boldsymbol{m}} \xrightarrow{f^{\mathrm{CM}}} \underset{\substack{\text{context}\\ \text{repr.}}}{\boldsymbol{m}'} \xrightarrow{f^{\mathrm{CAM}}} \underset{\substack{\text{similarity}\\ \text{repr.}}}{\boldsymbol{m}''}, \tag{4}$$

$$\underset{\substack{\text{symbolic or}\\ \text{low-level repr.}}}{x_n} \xrightarrow{f^{\mathrm{ME}}} \underset{\substack{\text{molecule}\\ \text{embedding}}}{\boldsymbol{x}_n} \xrightarrow{f^{\mathrm{CM}}} \underset{\substack{\text{context}\\ \text{repr.}}}{\boldsymbol{x}'_n} \xrightarrow{f^{\mathrm{CAM}}} \underset{\substack{\text{similarity}\\ \text{repr.}}}{\boldsymbol{x}''_n}. \tag{5}$$

3.2 Context module (CM)

The context module associates the query and support set molecules with a large set of context molecules, and represents them as weighted average of context molecule embeddings. The context module is realised by a continuous Modern Hopfield Network (MHN) (Ramsauer et al., 2021). An MHN is a content-addressable associative memory which can be built into deep learning architectures. There exists an analogy between the energy update of MHNs and the attention mechanism of Transformers (Vaswani et al., 2017; Ramsauer et al., 2021). MHNs are capable of storing and retrieving patterns from a memory $\boldsymbol{M} \in \mathbb{R}^{e \times M}$ given a state pattern $\boldsymbol{\xi} \in \mathbb{R}^e$ that represents the query. The retrieved pattern $\boldsymbol{\xi}^{\mathrm{new}} \in \mathbb{R}^e$ is obtained by

$$\boldsymbol{\xi}^{\mathrm{new}} = \boldsymbol{M}\,\boldsymbol{p} = \boldsymbol{M}\,\mathrm{softmax}\left(\beta \boldsymbol{M}^T \boldsymbol{\xi}\right), \tag{6}$$

where $\boldsymbol{p}$ is called the vector of associations and β is a scaling factor or inverse temperature. Modern Hopfield Networks have been successfully applied to chemistry and computational immunology (Seidl et al., 2022; Widrich et al., 2020).

We use this mechanism in the form of a *Hopfield layer*, which first maps raw patterns to an associative space using linear transformations, and uses multiple simultaneous queries $\boldsymbol{\Xi} \in \mathbb{R}^{d \times N}$:

$$\mathrm{Hopfield}(\boldsymbol{\Xi}, \boldsymbol{C}) := (\boldsymbol{W}_E \boldsymbol{C})\,\mathrm{softmax}\left(\beta\,(\boldsymbol{W}_C \boldsymbol{C})^T\,(\boldsymbol{W}_\Xi \boldsymbol{\Xi})\right), \tag{7}$$

where $\boldsymbol{W}_E \in \mathbb{R}^{d \times d}$ and $\boldsymbol{W}_C, \boldsymbol{W}_\Xi \in \mathbb{R}^{e \times d}$ are trainable parameters of the Hopfield layer, $\mathrm{softmax}$ is applied column-wise, and β is a hyperparameter. Note that in principle the $\boldsymbol{\Xi}$ and $\boldsymbol{C}$ could have a different second dimension as long as the linear transformations map to the same dimension e. Note that all embeddings that enter this module are first layer normalized (Ba et al., 2016). Several of these Hopfield layers can run in parallel and we refer to them as "heads" in analogy to Transformers (Vaswani et al., 2017).

The context module of our new architecture uses a Hopfield layer, where the query patterns are the embeddings of the query molecule $\boldsymbol{m}$ and the support set molecules $\boldsymbol{X}$. The memory is composed of embeddings of a large set of M molecules from a chemical space, for example reference molecules, here called context molecules $\boldsymbol{C}$. Then the original embeddings $\boldsymbol{m}$ and $\boldsymbol{X}$ are replaced by the retrieved embeddings, which are weighted averages of context molecule embeddings:

$$\boldsymbol{m}' = \mathrm{Hopfield}(\boldsymbol{m}, \boldsymbol{C}) \quad \text{and} \quad \boldsymbol{X}' = \mathrm{Hopfield}(\boldsymbol{X}, \boldsymbol{C}). \tag{8}$$

This retrieval step reinforces the covariance structure of the retrieved representations (see Appendix A.7). Note that the embeddings of the query and the support set molecules have not yet influenced each other. These updated representations $\boldsymbol{m}', \boldsymbol{X}'$ are passed to the cross-attention module.

3.3 Cross-attention module (CAM)

For embedding-based few-shot learning methods in the field of drug discovery, Altae-Tran et al. (2017) showed that the representations of the molecules can be enriched, if the architecture allows information exchange between query and support set molecules. Altae-Tran et al. (2017) uses an attention-enhanced LSTM variant which updates the query and the support set molecule representations in an iterative fashion, being aware of each other. We further develop this idea and combine it with the idea of using a transformer encoder layer (Vaswani et al., 2017) as a cross-attention module (Hou et al., 2019; Chen et al., 2021).

The cross-attention module updates the query molecule representation $\boldsymbol{m}'$ and the support set molecule representations $\boldsymbol{X}'$ by mutually exchanging information, using the usual Transformer mechanism:

$$[\boldsymbol{m}'', \boldsymbol{X}''] = \mathrm{Hopfield}([\boldsymbol{m}', \boldsymbol{X}'], [\boldsymbol{m}', \boldsymbol{X}']), \tag{9}$$

where $[\boldsymbol{m}', \boldsymbol{X}'] \in \mathbb{R}^{d \times (N+1)}$ is the concatenation of the representations of the query molecule $\boldsymbol{m}'$ with the support set molecules $\boldsymbol{X}'$ and we exploited that the Transformer is a special case of the Hopfield layer. Again, normalization is applied (Ba et al., 2016) and multiple Hopfield layers, i.e., heads, can run in parallel, be stacked, and equipped with skip-connections. The representations $\boldsymbol{m}''$ and $\boldsymbol{X}''$ are passed to the similarity module.

3.4 Similarity module (SM)

In this module, pairwise similarity values $k(\boldsymbol{m}'', \boldsymbol{x}_n'')$ are computed between the representation of a query molecule $\boldsymbol{m}''$ and each molecule $\boldsymbol{x}_n''$ in the support set as done recently (Koch et al., 2015; Altae-Tran et al., 2017). Based on these similarity values, the activity for the query molecule is predicted, building a weighted mean over the support set labels:

$$\hat{y} = \sigma\left(\tau^{-1} \frac{1}{N} \sum_{n=1}^{N} y_n' \, k(\boldsymbol{m}'', \boldsymbol{x}_n'')\right), \tag{10}$$

where our architecture employs dot product similarity of normalized representations $k(\boldsymbol{m}'', \boldsymbol{x}_n'') = \boldsymbol{m}''^T \boldsymbol{x}_n''$. $\sigma(.)$ is the sigmoid function and τ is a hyperparameter. Note that we use a balancing strategy for the labels $y_n' = \begin{cases} N/(2\sqrt{N_A}) & \text{if } y_n = 1 \\ -N/(2\sqrt{N_I}) & \text{else} \end{cases}$, where N_A is the number of actives and N_I is the number of inactives of the support set.

3.5 Architecture, hyperparameter selection, and training details

Hyperparameters. The main hyperparameters of our architecture are the number of heads, the embedding dimension, the dimension of the association space of the CAM and CM, the learning

rate schedule, the scaling parameter β, and the molecule encoder. The following hyperparameters were selected by manual hyperparameter selection on the validation tasks. The molecule encoder consists of a single layer with output size $d = 1024$ and SELU activation (Klambauer et al., 2017). The CM consists of one Hopfield layer with 8 heads. The dimension e of the association space is set to 512 and $\beta = 1/\sqrt{e}$. Since we use skip connections between all modules the output dimension of the CM and CAM matches the input dimension. The CAM comprises one layer with 8 heads and an association-space dimension of 1088. For the input to the CAM, an activity encoding was added to the support set molecule representations to provide label information. The SM uses $\tau = 22.6$. For the context set, we randomly sample 5% from a large set of molecules – i.e., the molecules in the FS-Mol training split – for each batch. For inference, we used a fixed set of 5% of training set molecules as the context set for each seed. We hypothesize that these choices about the context could be further improved (Section 6). We provide considered and selected hyperparameters in Appendix A.1.6.

Loss function, regularization and optimization. We use the Adam optimizer (Kingma and Ba, 2014) to minimize the cross-entropy loss between the predicted and known activity labels. We use a learning rate scheduler which includes a warm up phase, followed by a section with a constant learning rate, which is 0.0001, and a third phase in which the learning rate steadily decreases. As a regularization strategy, for the CM and the CAM a dropout rate of 0.5 is used. The molecule encoder has a dropout with rate 0.1 for the input and 0.5 elsewhere (see also Appendix A.1.6).

Compute time and resources. Training a single **MHNfs** model on the benchmarking dataset FS-Mol takes roughly 90 hours of wall-clock time on an A100 GPU. In total, roughly 15,000 GPU hours were consumed for this work.

4 Related work

Several approaches to few-shot learning in drug discovery have been suggested (Altae-Tran et al., 2017; Nguyen et al., 2020; Guo et al., 2021; Wang et al., 2021). (Nguyen et al., 2020) evaluated the applicability of MAML and its variants to graph neural networks (GNNs) and (Guo et al., 2021) also combine GNNs and meta-learning. (Altae-Tran et al., 2017) suggested an approach called Iterative Refinement Long Short-Term Memory, in which query and support set embeddings can share information and update their embeddings. Property-aware relation networks (PAR) (Wang et al., 2021) use an attention mechanism to enrich representations from cluster centers and then learn a relation graph between molecules. (Chen et al., 2022) propose to adaptively learn kernels and apply their method to few-shot drug discovery with predictive performance for larger support set sizes. Recently, (Stanley et al., 2021) generated a benchmark dataset for few-shot learning methods in drug discovery and provided some baseline results.

Many successful deep neural network architectures use external memories, such as the neural Turing machine (Graves et al., 2014), memory networks (Weston et al., 2014), end-to-end memory networks (Sukhbaatar et al., 2015). Recently, the connection between continuous modern Hopfield networks (Ramsauer et al., 2021), which are content-addressable associative memories, and Transformer architectures (Vaswani et al., 2017) has been established. We refer to (Le, 2021) for an extensive overview of memory-based architectures. Architectures with external memories have also been used for meta-learning (Vinyals et al., 2016; Santoro et al., 2016) and few-shot learning (Munkhdalai and Yu, 2017; Ramalho and Garnelo, 2018; Ma et al., 2021).

5 Experiments

5.1 Benchmarking on FS-Mol

Experimental setup. Recently, the dataset FS-Mol (Stanley et al., 2021) was proposed to benchmark few-shot learning methods in drug discovery. It was extracted from ChEMBL27 and comprises in total 489,133 measurements, 233,786 compounds and 5,120 tasks. Per task, the mean number of data points is 94. The dataset is well balanced as the mean ratio of active and inactive molecules is close to 1. The FS-Mol benchmark dataset defines 4,938 training, 40 validation and 157 test tasks, guaranteeing disjoint task sets. (Stanley et al., 2021) precomputed extended connectivity fingerprints (ECFP) (Rogers and Hahn, 2010) and key molecular physical descriptors, which were defined by RDKit (Landrum et al., 2006). While methods would be allowed to use other representations of the input molecules, such as the molecular graph, we used a concatenation of these ECFPs and

Table 1: Results on FS-MOL [ΔAUC-PR]. The best method is marked bold. Error bars represent standard errors across tasks according to Stanley et al. (2021). The metrics are also averaged across five training re-runs and ten draws of support sets. In brackets the number of tasks per category is reported.

Method	All [157]	Kin. [125]	Hydrol. [20]	Oxid.[7]
GNN-ST[a] (Stanley et al., 2021)	.029 ± .004	.027 ± .004	.040 ± .018	.020 ± .016
MAT[a] (Maziarka et al., 2020)	.052 ± .005	.043 ± .005	.095 ± .019	.062 ± .024
Random Forest[a] (Breiman, 2001)	.092 ± .007	.081 ± .009	.158 ± .028	.080 ± .029
GNN-MT[a] (Stanley et al., 2021)	.093 ± .006	.093 ± .006	.108 ± .025	.053 ± .018
Similarity Search	.118 ± .008	.109 ± .008	.166 ± .029	.097 ± .033
GNN-MAML[a] (Guo et al., 2021)	.159 ± .009	.177 ± .009	.105 ± .024	.054 ± .028
PAR(Wang et al., 2021)	.164 ± .008	.182 ± .009	.109 ± .020	.039 ± .008
Frequent hitters	.182 ± .010	.207 ± .009	.098 ± .009	.041 ± .005
ProtoNet[a] (Snell et al., 2017)	.207 ± .008	.215 ± .009	.209 ± .030	.095 ± .029
Siamese Networks (Koch et al., 2015)	.223 ± .010	.241 ± .010	.178 ± .026	.082 ± .025
IterRefLSTM (Altae-Tran et al., 2017)	.234 ± .010	.251 ± .010	.199 ± .026	.098 ± .027
ADKF-IFT[b] (Chen et al., 2022)	.234 ± .009	.248 ± .020	**.217** ± .017	**.106** ± .008
MHNfs (ours)	**.241** ± .009	**.259** ± .010	.199 ± .027	.096 ± .019

[a] metrics from Stanley et al. (2021). [b] results from Chen et al. (2022).

RDKit-based descriptors. For the main benchmark, the support set size was fixed to 16, using a stratified random split. We use all these settings of FS-Mol and therefore ensure a fair method comparison.

Methods compared. Baselines for few-shot learning and our proposed method **MHNfs** were compared against each other. The **Frequent Hitters** model is a naive baseline that ignores the provided support set and therefore has to learn to predict the average activity of a molecule. This method can potentially discriminate so-called frequent-hitter molecules (Stork et al., 2019) against molecules that are inactive across many tasks. We also added **Similarity Search** (Cereto-Massagué et al., 2015) as a baseline. Similarity search is a standard chemoinformatics technique, used in situations with single or few known actives. In the simplest case, the search finds similar molecules by computing a fingerprint or descriptor-representation of the molecules and using a similarity measure $k(.,.)$ — such as Tanimoto Similarity (Tanimoto, 1960). Thus, Similarity Search, as used in chemoinformatics, can be formally written as $\hat{y} = 1/N \sum_{n=1}^{N} y_n \; k(\boldsymbol{m}, \boldsymbol{x}_n)$; where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mathrm{n}$ come from a fixed molecule encoder, such as chemical fingerprint or descriptor calculation. A natural extension of Similarity Search with fixed chemical descriptors is **Neural Similarity Search or Siamese networks** (Koch et al., 2015), which extend the classic similarity search by learning a molecule encoder: $\hat{y} = \sigma\left(\tau^{-1} \frac{1}{N} \sum_{n=1}^{N} y'_n \; f_{\boldsymbol{w}}^{\mathrm{ME}}(\boldsymbol{m})^T \; f_{\boldsymbol{w}}^{\mathrm{ME}}(\boldsymbol{x}_n)\right)$. Furthermore, we re-implemented the **IterRefLSTM** (Altae-Tran et al., 2017) in Pytorch. The **IterRefLSTM** model consists of three modules. First, a molecule encoder maps the query and support set molecules to its representations $\boldsymbol{m}$ and $\boldsymbol{X}$. Second, an attention-enhanced LSTM variant, the actual **IterRefLSTM**, iteratively updates the query and support set molecules, enabling information sharing between the molecules: $[\boldsymbol{m}', \boldsymbol{X}'] = \mathrm{IterRefLSTM}_L([\boldsymbol{m}, \boldsymbol{X}])$, where the hyperparameter L controls the number of iteration steps of the **IterRefLSTM**. Third, a similarity module computes attention weights based on the representations: $\boldsymbol{a} = \mathrm{softmax}\left(k\left(\boldsymbol{m}', \boldsymbol{X}'\right)\right)$. These representations are then used for the final prediction: $\hat{y} = \sum_{i=1}^{N} a_i y_i$. For further details, see Appendix A.1.5. The **Random Forest** baseline uses the chemical descriptors and is trained in standard supervised manner on the support set molecules for each task. The method **GNN-ST** is a graph neural network (Stanley et al., 2021; Gilmer et al., 2017) that is trained from scratch for each task. The **GNN-MT** uses a two step strategy: First, the model is pretrained on a large dataset on related tasks; second, an output layer is constructed to the few-shot task via linear probing (Stanley et al., 2021; Alain and Bengio, 2016). The **Molecule Attention Transformer (MAT)** is pre-trained in a self-supervised fashion and fine-tuning is performed for the few-shot task (Maziarka et al., 2020). **GNN-MAML** is based on MAML (Finn et al., 2017), and uses a model-agnostic meta-learning strategy to find a general core model from which one can easily adapt to single tasks. **ProtoNet** (Snell et al., 2017) includes a molecule encoder, which maps query and support set molecules to representations in an embedding space. In this embedding space, prototypical

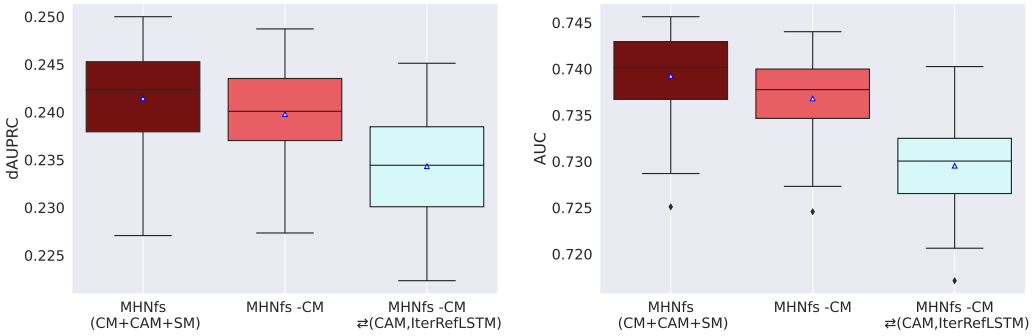

Figure 2: Results of the ablation study. The boxes show the median, mean and the variability of the average predictive performance of the methods across training re-runs and draws of support sets. The performance significantly drops when the context module is removed (light red bars), and when additionally the cross-attention module is replaced with the **IterRefLSTM** module (light blue bars). This indicates that our two newly introduced modules, CM and CAM, play a crucial role in MHNfs.

representations of each class are built by taking the mean across all related support set molecules for each class (details in Appendix A.1.4). For all methods the most important hyperparameters were adjusted on the validation tasks of FS-Mol. The **PAR** model (Wang et al., 2021) includes a GNN which creates initial molecule embeddings. These molecule embeddings are then enriched by an attention mechanism. Finally, another GNN learns relations between support and query set molecules. The **PAR** model has shown good results for datasets which just include very few tasks such as Tox21 (Wang et al., 2021). Chen et al. (2022) suggest a framework for learning deep kernels by interpolating between meta-learning and conventional deep kernels, which results in the **ADKF-IFT** model. The model has exhibited especially high performance for large support set sizes.

Training and evaluation. For the model implementations, we used PyTorch (Paszke et al., 2019, BSD license). We used PyTorch Lightning (Falcon et al., 2019, Apache 2.0 license) as a framework for training and test logic, hydra for config file handling (Yadan, 2019, Apache 2.0 license) and Weights & Biases (Biewald, 2020, MIT license) as an experiment tracking tool. We performed five training reruns with different seeds for all methods, except Classic Similarity Search as there is no variability across seeds. Each model was evaluated ten times by drawing support sets with ten different seeds.

Results. The results in terms of area under precision-recall curve (AUC-PR) are presented in Table 1, where the difference to a random classifier is reported (ΔAUC-PR). The standard error is reported across tasks. Surprisingly, the naive baseline **Frequent Hitters**, that neglects the support set, has outperformed most of the few-shot learning methods, except for the embedding-based methods **Siamese Networks**, **ProtoNet**, **IterRefLSTM**, and **MHNfs**. **IterRefLSTM**, which has not been included in the FS-Mol benchmark study, reaches the second best performance. **MHNfs** has outperformed all other methods with respect to ΔAUC-PR across all tasks, including the **IterRefLSTM** model (p-value 1.72e-7, paired Wilcoxon test), the **ADKF-IFT** model (p-value <1.0e-8, Wilcoxon test), and the **PAR** model (p-value <1.0e-8, paired Wilcoxon test).

5.2 Ablation study

MHNfs has two new main components compared to the previous state-of-the-art method **IterRefLSTM**: i) the context module, and ii) the cross-attention module which replaces the LSTM-like module. To assess the effects of these components, we performed an ablation study. Therefore, we compared **MHNfs** to a method that does not have the context module ("MHNfs -CM") and to a method that does not have the context module and uses an LSTM-like module instead of the CAM ("MHNfs -CM $\rightleftharpoons$(CAM,IterRefLSTM)"). For the ablation study, we used all 5 training reruns and evaluated each model 10 times on the test set with different support sets. The results of this ablation steps are presented in Figure 2. Both removing the CM and exchanging the CAM with the **IterRefLSTM** module were detrimental for the performance of the method (p-value 0.002 and $1.72e{-}7$, respectively; paired Wilcoxon test). The difference was even more pronounced under domain shift (see Appendix A.3.3). Appendix A.3.2 contains a second ablation study that examines

the overall effects of the context, the cross-attention, the similarity module, and the molecule encoder of **MHNfs**.

5.3 Domain shift experiment

We performed an experiment in which we evaluate models, that were pretrained on FS-Mol, on the Tox21 (Mayr et al., 2016) dataset. There is a strong domain shift from the drug-like molecules of FS-Mol to the environmental chemicals, pesticides, and food additives of Tox21, such this dataset poses a challenging setting for few-shot learning methods. The experiment is described in detail in Appendix A.2. Our **MHNfs** approach has reached an AUC of $.679 \pm .018$ and has significantly outperformed the **IterRefLSTM**-based model ($p_{\Delta\mathrm{AUC-PR}}$-value $3.4\mathrm{e}{-5}$, paired Wilcoxon test) and the Classic Similarity Search ($p_{\Delta\mathrm{AUC-PR}}$-value 2.4e-9 paired Wilcoxon test) and therefore showed robust performance on the toxicity domain, see Table A6.

6 Conclusion and discussion

We have introduced a new architecture for few-shot learning in drug discovery that is based on the novel concept to enrich molecule representations with context. In a benchmarking experiment, the architecture was assessed for its ability to learn accurate predictive models from small sets of labelled molecules and in this setting it outperformed all other methods. In a domain shift study, the robustness and transferability of the learned models has been assessed and again **MHNfs** exhibited the best performance. The resulting predictive models often reach an AUC larger than .70, which means that enrichment of active molecules is expected (Simm et al., 2018) when the models are used for virtual screening. It has not escaped our notice that the specific context module we have proposed could immediately be used for few-shot learning tasks in computer vision, but might be hampered by computational constraints. **Limitations.** While the implementation of our method is currently limited to small, organic drug-like molecules as inputs, our conceptual approach can also be used for macro-molecules such as RNA, DNA or proteins. The output domain of our method comprises biological effects, such that the prediction must be understood in that domain. Our method demands higher computational costs and memory footprint as other embedding-based methods because of the calculations necessary for the context module. While we hypothesize that our approach could also be successful for similar data in the materials science domain, this has not been assessed. Our study is also constrained by a limited amount of hyperparameter search for all methods. Deep learning methods usually have a large number of hyperparameters, such as hidden dimensions, number of layers, learning rates, of which we were only able to explore the most important ones. The composition and choice of the context set is also under-explored and might be improved by selecting reference molecules with an appropriate strategy. **Broader impact.** *Impact on machine learning and related scientific fields.* We envision that with (a) the increasing availability of drug discovery and material science datasets, (b) further improved biotechnologies, and (c) accounting for characteristics of individuals, the drug and materials discovery process will be made more efficient. For machine learning and artificial intelligence, the novel way in which representations are enriched with context might strengthen the general research stream to include more context into deep learning systems. Our approach also shows that such a system is more robust against domain shifts, which could be a step towards Broad AI (Chollet, 2019; Hochreiter, 2022). *Impact on society.* If the approach proves useful, it could lead to a faster and more cost-efficient drug discovery process. Especially the COVID-19 pandemic has shown that it is crucial for humanity to speed up the drug discovery process to few years or even months. We hope that this work contributes to this effort and eventually leads to safer drugs developed faster. *Consequences of failures of the method.* As common with methods in machine learning, potential danger lies in the possibility that users rely too much on our new approach and use it without reflecting on the outcomes. Failures of the proposed method would lead to unsuccessful wet lab validation and negative wet lab tests. Since the proposed algorithm does not directly suggest treatment or therapy, human beings are not directly at risk of being treated with a harmful therapy. Wet lab and in-vitro testing would indicate wrong decisions by the system. *Leveraging of biases in the data and potential discrimination.* As for almost all machine learning methods, confounding factors, lab or batch effects, could be used for classification. This might lead to biases in predictions or uneven predictive performance across different drug targets or bioassays.

References

Adler, T., Brandstetter, J., Widrich, M., Mayr, A., Kreil, D., Kopp, M., Klambauer, G., and Hochreiter, S. (2020). Cross-domain few-shot learning by representation fusion. *arXiv preprint arXiv:2010.06498*.

Alain, G. and Bengio, Y. (2016). Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*.

Alperstein, Z., Cherkasov, A., and Rolfe, J. T. (2019). All smiles variational autoencoder. *arXiv preprint arXiv:1905.13343*.

Altae-Tran, H., Ramsundar, B., Pappu, A. S., and Pande, V. (2017). Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293.

Antoniou, A. and Storkey, A. (2019). Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation. *arXiv preprint arXiv:1902.09884*.

Arrowsmith, J. (2011). Phase ii failures: 2008-2010. *Nature reviews drug discovery*, 10(5).

Axelrod, S. and Gomez-Bombarelli, R. (2022). Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):1–14.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bender, A., Mussa, H. Y., Glen, R. C., and Reiling, S. (2004). Similarity searching of chemical databases using atom environment descriptors (molprint 2d): evaluation of performance. *Journal of chemical information and computer sciences*, 44(5):1708–1718.

Bendre, N., Marín, H. T., and Najafirad, P. (2020). Learning from few samples: A survey. *arXiv preprint arXiv:2007.15484*.

Bengio, Y., Bengio, S., and Cloutier, J. (1991). Learning a synaptic learning rule. In *Seattle international joint conference on neural networks*.

Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com.

Bonner, M. F. and Epstein, R. A. (2021). Object representations in the human brain reflect the co-occurrence statistics of vision and language. *Nature Communications*, 12(4081).

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Cereto-Massagué, A., Ojeda, M. J., Valls, C., Mulero, M., Garcia-Vallvé, S., and Pujadas, G. (2015). Molecular fingerprint similarity search in virtual screening. *Methods*, 71:58–63.

Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., and Blaschke, T. (2018). The rise of deep learning in drug discovery. *Drug discovery today*, 23(6):1241–1250.

Chen, H., Li, H., Li, Y., and Chen, C. (2021). Sparse spatial transformers for few-shot learning. *arXiv preprint arXiv:2109.12932*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Chen, W., Tripp, A., and Hernández-Lobato, J. M. (2022). Meta-learning feature representations for adaptive gaussian processes via implicit differentiation. *arXiv preprint arXiv:2205.02708*.

Chollet, F. (2019). On the measure of intelligence. *arXiv preprint arXiv:1911.01547*.

Dahl, G. E., Jaitly, N., and Salakhutdinov, R. (2014). Multi-task neural networks for qsar predictions. *arXiv preprint arXiv:1406.1231*.

Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*.

Eckert, H. and Bajorath, J. (2007). Molecular similarity analysis in virtual screening: foundations, limitations and novel approaches. *Drug discovery today*, 12(5-6):225–233.

Falcon, W. et al. (2019). Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3:6.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

Fürst, A., Rumetshofer, E., Tran, V., Ramsauer, H., Tang, F., Lehner, J., Kreil, D., Kopp, M., Klambauer, G., Bitto-Nemling, A., et al. (2021). Cloob: Modern hopfield networks with infoloob outperform clip. *arXiv preprint arXiv:2110.11316*.

Geppert, H., Horváth, T., Gärtner, T., Wrobel, S., and Bajorath, J. (2008). Support-vector-machine-based ranking significantly improves the effectiveness of similarity searching using 2d fingerprints and multiple reference compounds. *Journal of chemical information and modeling*, 48(4):742–746.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.

Gomez, L. (2018). Decision making in medicinal chemistry: The power of our intuition. *ACS Medicinal Chemistry Letters*, 9(10):956–958.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Guo, Z., Zhang, C., Yu, W., Herr, J., Wiest, O., Jiang, M., and Chawla, N. V. (2021). Few-shot graph learning for molecular property prediction. In *Proceedings of the web conference 2021*, pages 2559–2567.

He, J., You, H., Sandström, E., Nittinger, E., Bjerrum, E. J., Tyrchan, C., Czechtizky, W., and Engkvist, O. (2021). Molecular optimization by capturing chemist's intuition using deep neural networks. *Journal of cheminformatics*, 13(1):1–17.

Hertz, T., Hillel, A. B., and Weinshall, D. (2006). Learning a kernel function for classification with small training samples. In *Proceedings of the 23rd international conference on machine learning*, pages 401–408.

Hochreiter, S. (2022). Toward a broad ai. *Communications of the ACM*, 65(4):56–57.

Hochreiter, S., Klambauer, G., and Rarey, M. (2018). Machine learning in drug discovery. *Journal of Chemical Information and Modeling*, 58(9):1723–1724.

Hochreiter, S., Younger, A. S., and Conwell, P. R. (2001). Learning to learn using gradient descent. In *International conference on artificial neural networks*, pages 87–94. Springer.

Hou, R., Chang, H., Ma, B., Shan, S., and Chen, X. (2019). Cross attention network for few-shot classification. *Advances in neural information processing systems 32*.

Huang, R., Xia, M., Nguyen, D.-T., Zhao, T., Sakamuru, S., Zhao, J., Shahane, S. A., Rossoshek, A., and Simeonov, A. (2016a). Tox21challenge to build predictive models of nuclear receptor and stress response pathways as mediated by exposure to environmental chemicals and drugs. *Frontiers in Environmental Science*, 3:85.

Huang, R., Xia, M., Sakamuru, S., Zhao, J., Shahane, S. A., Attene-Ramos, M., Zhao, T., Austin, C. P., and Simeonov, A. (2016b). Modelling the tox21 10 k chemical profiles for in vivo toxicity prediction and mechanism characterization. *Nature communications*, 7(1):1–10.

Jiang, D., Wu, Z., Hsieh, C.-Y., Chen, G., Liao, B., Wang, Z., Shen, C., Cao, D., Wu, J., and Hou, T. (2021). Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of cheminformatics*, 13(1):1–23.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. (2016). Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in neural information processing systems 30*, pages 972–981.

Koch, G., Zemel, R., Salakhutdinov, R., et al. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.

Kuhn, M., Letunic, I., Jensen, L. J., and Bork, P. (2016). The sider database of drugs and side effects. *Nucleic acids research*, 44(D1):D1075–D1079.

Landrum, G. et al. (2006). Rdkit: Open-source cheminformatics.

Le, H. (2021). Memory and attention in deep learning. *arXiv preprint arXiv:2107.01390*.

Li, J., Cai, D., and He, X. (2017). Learning graph-level representation for drug discovery. *arXiv preprint arXiv:1709.03741*.

Li, P., Li, Y., Hsieh, C.-Y., Zhang, S., Liu, X., Liu, H., Song, S., and Yao, X. (2021). Trimnet: learning molecular representation from triplet messages for biomedicine. *Briefings in Bioinformatics*, 22(4):bbaa266.

Ma, Y., Liu, W., Bai, S., Zhang, Q., Liu, A., Chen, W., and Liu, X. (2021). Few-shot visual learning with contextual memory and fine-grained calibration. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 811–817.

Mayr, A., Klambauer, G., Unterthiner, T., and Hochreiter, S. (2016). Deeptox: toxicity prediction using deep learning. *Frontiers in environmental science*, 3:80.

Mayr, A., Klambauer, G., Unterthiner, T., Steijaert, M., Wegner, J. K., Ceulemans, H., Clevert, D.-A., and Hochreiter, S. (2018). Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical science*, 9(24):5441–5451.

Maziarka, Ł., Danel, T., Mucha, S., Rataj, K., Tabor, J., and Jastrzębski, S. (2020). Molecule attention transformer. *arXiv preprint arXiv:2002.08264*.

Merk, D., Friedrich, L., Grisoni, F., and Schneider, G. (2018). De novo design of bioactive small molecules by artificial intelligence. *Molecular informatics*, 37(1-2):1700153.

Merkwirth, C. and Lengauer, T. (2005). Automatic generation of complementary descriptors with molecular graph networks. *Journal of chemical information and modeling*, 45(5):1159–1168.

Miller, E. G., Matsakis, N. E., and Viola, P. A. (2000). Learning from one example through shared densities on transforms. In *Proceedings ieee conference on computer vision and pattern recognition. cvpr 2000 (cat. no. PR00662)*, volume 1, pages 464–471.

Munkhdalai, T. and Yu, H. (2017). Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR.

Nguyen, C. Q., Kreatsoulas, C., and Branson, K. M. (2020). Meta-learning gnn initializations for low-resource molecular property prediction. *arXiv preprint arXiv:2003.05996*.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2019). Automatic differentiation in pytorch. In *Conference on neural information processing systems*.

Potter, M. (2012). Conceptual short term memory in perception and thought. *Frontiers in Psychology*, 3:113.

Ramalho, T. and Garnelo, M. (2018). Adaptive posterior learning: few-shot learning with a surprise-based memory module. In *International Conference on Learning Representations*.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Gruber, L., Holzleitner, M., Adler, T., Kreil, D., Kopp, M. K., Klambauer, G., Brandstetter, J., and Hochreiter, S. (2021). Hopfield networks is all you need. In *International conference on learning representations*.

Riniker, S. and Landrum, G. A. (2013). Open-source platform to benchmark fingerprints for ligand-based virtual screening. *Journal of cheminformatics*, 5(1):1–17.

Rogers, D. and Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR.

Schmidhuber, J. (1987). Evolutionary principles in self-referential learning.

Schneider, P., Walters, W. P., Plowright, A. T., Sieroka, N., Listgarten, J., Goodnow, R. A., Fisher, J., Jansen, J. M., Duca, J. S., Rush, T. S., et al. (2020). Rethinking drug design in the artificial intelligence era. *Nature reviews drug discovery*, 19(5):353–364.

Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. (2018a). Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131.

Segler, M. H., Preuss, M., and Waller, M. P. (2018b). Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698):604–610.

Seidl, P., Renz, P., Dyubankova, N., Neves, P., Verhoeven, J., Wegner, J. K., Segler, M., Hochreiter, S., and Klambauer, G. (2022). Improving few-and zero-shot reaction template prediction using modern hopfield networks. *Journal of chemical information and modeling*, 62(9):2111–2120.

Sheridan, R. P. and Kearsley, S. K. (2002). Why do we need so many chemical similarity search methods? *Drug discovery today*, 7(17):903–911.

Simm, J., Klambauer, G., Arany, A., Steijaert, M., Wegner, J. K., Gustin, E., Chupakhin, V., Chong, Y. T., Vialard, J., Buijnsters, P., et al. (2018). Repurposed high-throughput image assays enables biological activity prediction for drug discovery. *Cell Chemical Biology*, page 108399.

Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*.

Stanley, M., Bronskill, J. F., Maziarz, K., Misztela, H., Lanini, J., Segler, M., Schneider, N., and Brockschmidt, M. (2021). Fs-mol: A few-shot learning dataset of molecules. In *Conference on neural information processing systems workshop*.

Stork, C., Chen, Y., Sicho, M., and Kirchmair, J. (2019). Hit dexter 2.0: machine-learning models for the prediction of frequent hitters. *Journal of chemical information and modeling*, 59(3):1030–1043.

Sturm, N., Mayr, A., Le Van, T., Chupakhin, V., Ceulemans, H., Wegner, J., Golib-Dzib, J.-F., Jeliazkova, N., Vandriessche, Y., Böhm, S., et al. (2020). Industry-scale application and evaluation of deep learning for drug target prediction. *Journal of Cheminformatics*, 12(1):1–13.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. *Advances in neural information processing systems*, 28.

Sun, J., Jeliazkova, N., Chupakhin, V., Golib-Dzib, J.-F., Engkvist, O., Carlsson, L., Wegner, J., Ceulemans, H., Georgiev, I., Jeliazkov, V., et al. (2017). Excape-db: an integrated large scale dataset facilitating big data analysis in chemogenomics. *Journal of cheminformatics*, 9(1):1–9.

Tanimoto, T. (1960). Ibm type 704 medical diagnosis program. *IRE transactions on medical electronics*, (4):280–283.

Torres, L., Monteiro, N., Oliveira, J., Arrais, J., and Ribeiro, B. (2020). Exploring a siamese neural network architecture for one-shot drug discovery. In *2020 ieee 20th international conference on bioinformatics and bioengineering (bibe)*, pages 168–175.

Unterthiner, T., Mayr, A., Klambauer, G., Steijaert, M., Wegner, J. K., Ceulemans, H., and Hochreiter, S. (2014). Deep learning as an opportunity in virtual screening. In *Advances in neural information processing systems workshop*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638.

Walters, W. P. and Barzilay, R. (2021). Critical assessment of ai in drug discovery. *Expert opinion on drug discovery*, pages 1–11.

Wang, X., Huan, J., Smalter, A., and Lushington, G. H. (2010). Application of kernel functions for accurate similarity search in large chemical databases. In *BMC bioinformatics*, volume 11, pages 1–14. BioMed Central.

Wang, Y., Abuduweili, A., Yao, Q., and Dou, D. (2021). Property-aware relation networks for few-shot molecular property prediction. *Advances in Neural Information Processing Systems*, 34:17441–17454.

Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34.

Waring, M. J., Arrowsmith, J., Leach, A. R., Leeson, P. D., Mandrell, S., Owen, R. M., Pairaudeau, G., Pennie, W. D., Pickett, S. D., Wang, J., et al. (2015). An analysis of the attrition of drug candidates from four major pharmaceutical companies. *Nature reviews drug discovery*, 14(7):475–486.

Weininger, D. (1988). Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36.

Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. *arXiv preprint arXiv:1410.3916*.

Widrich, M., Schäfl, B., Pavlović, M., Ramsauer, H., Gruber, L., Holzleitner, M., Brandstetter, J., Sandve, G. K., Greiff, V., Hochreiter, S., et al. (2020). Modern hopfield networks and attention for immune repertoire classification. In *Advances in neural information processing systems 33*.

Willett, P. (2014). The calculation of molecular structural similarity: principles and practice. *Molecular informatics*, 33(6-7):403–413.

Winter, R., Montanari, F., Noé, F., and Clevert, D.-A. (2019). Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. *Chemical science*, 10(6):1692–1701.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. (2018). Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530.

Yadan, O. (2019). Hydra - a framework for elegantly configuring complex applications. Github. Visited 2022-04-25.

Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., et al. (2019). Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388.

Ye, M. and Guo, Y. (2018). Deep triplet ranking networks for one-shot recognition. *arXiv preprint arXiv:1804.07275*.

Zaslavskiy, M., Jégou, S., Tramel, E. W., and Wainrib, G. (2019). Toxicblend: Virtual screening of toxic compounds with ensemble predictors. *Computational Toxicology*, 10:81–88.

Zhao, A., Balakrishnan, G., Durand, F., Guttag, J. V., and Dalca, A. V. (2019). Data augmentation using learned transformations for one-shot medical image segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 8543–8553.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We refer e.g. to Section 5 in which we provide this information.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report error bars for all performance metrics. For all experiments the variability across re-runs, different support sets and prediction tasks was assessed.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 3.5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Section 6.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Contents of the appendix

A Appendix

A.1 Details on methods

Few-shot learning methods in drug discovery can be described as models with adaptive parameters $\boldsymbol{w}$ that use a support set $\boldsymbol{Z} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ [1] as additional input to predict a label $\hat{y}$ for a molecule $\boldsymbol{m}$

$$\hat{y} = g_{\boldsymbol{w}}(\boldsymbol{m}, \boldsymbol{Z}). \tag{A1}$$

Optimization-based methods, such as MAML (Finn et al., 2017), use the support set to update the parameters $\boldsymbol{w}$

$$\hat{y} = g_{a(\boldsymbol{w};\boldsymbol{Z})}(\boldsymbol{m}), \tag{A2}$$

where $a(.)$ is a function that adapts $\boldsymbol{w}$ of g based on $\boldsymbol{Z}$ for example via gradient-descent.

Embedding-based methods use a different approach and learn representations of the support set molecules $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, sometimes written as stacked embeddings $\boldsymbol{X} \in \mathbb{R}^{d \times N}$, and the query molecule $\boldsymbol{m}$, and some function that associates these two types of information with each other. We describe the embedding-based methods Similarity Search in Section A.1.2, Neural Similarity Search in Section A.1.3, ProtoNet in Section A.1.4, IterRefLSTM in Section A.1.5, PAR in Section A.1.7, and MHNfs in the main paper and details in Section A.1.6. The "frequent hitters" baseline is described in Section A.1.1.

A.1.1 Frequent hitters: details and hyperparameters

The "frequent hitters" model g^{FH} is a baseline that we implemented and included in the method comparison. This method uses the usual training scheme of sampling a query molecule $\boldsymbol{m}$ with a label y, having access to a support set $\boldsymbol{Z}$. In contrast to the usual models of the type $g_{\boldsymbol{w}}(\boldsymbol{m}, \boldsymbol{Z})$, the frequent hitters model g^{FH} neglects the support set:

$$\hat{y} = g^{\mathrm{FH}}_{\boldsymbol{w}}(\boldsymbol{m}). \tag{A3}$$

Thus, during training for the same molecule $\boldsymbol{m}$, the model might have to predict both $y = 1$ and $y = -1$, since the molecule can be active in one task and inactive in another task. Therefore, the

[1] We use $\boldsymbol{Z}$ to denote the support set of already embedded molecules to keep the notation uncluttered. More correctly, the methods have access to the raw support set $Z = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, where x_n is a symbolic, such as the molecular graph, or low-level representation of the molecule.

Table A1: Hyperparameter space considered for the Frequent hitters model. The hyperparameters of the best configuration are marked bold.

Hyperparameter	Explored values
Number of hidden layers	1, **2**, 4
Number of units per hidden layer	1024, **2048**, 4096
Output dimension	**512**, 1024
Activation function	**ReLU**
Learning rate	**0.0001**, 0.001
Optimizer	Adam,**AdamW**
Weight decay	0, **0.01**
Batch size	32, 128, 512, 2048, **4096**
Input Dropout	0, **0.1**
Dropout	0.1, 0.2, 0.3, **0.4**, 0.5
Layer-normalization	False, **True**
• Affine	**False**, True
Similarity function	**dot product**

model tends to predict average activity of a molecule to minimize the cross-entropy loss. We chose an additive combination of the Morgan fingerprints, RDKit fingerprints, and MACCS keys for the input representation to the MLP.

Hyperparameter search. We performed manual hyperparameter search on the validation set and report the explored hyperparameter space (Table A1). We use early-stopping based on validation average-precision, a patience of 3 epochs and train for a maximum of 20 epochs with a linear warm-up learning-rate schedule for the first 3 epochs.

A.1.2 Classic similarity search: details and hyperparameters

Similarity Search (Cereto-Massagué et al., 2015) is a classic chemoinformatics technique used in situations in which a single or few actives are known. In the simplest case, molecules that are similar to a given active molecule are searched by computing a fingerprint or descriptor-representation $f^{\text{desc}}(\boldsymbol{m})$ of the molecules and using a similarity measure $k(.,.)$, such as Tanimoto Similarity(Tanimoto, 1960). Thus, the Similarity Search as used in chemoinformatics can be formally written as:

$$\hat{y} = 1/N \sum_{n=1}^{N} y_n \, k(f^{\text{desc}}(m), f^{\text{desc}}(x_n)), \tag{A4}$$

where the function f^{desc} maps the molecule to its chemical descriptors or fingerprints and takes the role of both the molecule encoder and the support set encoder. The association function f^{assoc} consists of a) the similarity measure $k(.,.)$ and then b) mean pooling across molecules weighted by their similarity and activity.

Notably, there are many variants of Similarity Search (Cereto-Massagué et al., 2015; Wang et al., 2010; Eckert and Bajorath, 2007; Geppert et al., 2008; Willett, 2014; Sheridan and Kearsley, 2002; Riniker and Landrum, 2013) of which some correspond to recent few-shot learning methods with a fixed molecule encoder. For example, (Geppert et al., 2008) suggest to use centroid molecules, i.e., prototypes or averages of active molecules. This is equivalent to the idea of Prototypical Networks (Snell et al., 2017). Riniker and Landrum (2013) are aware of different fusion strategies for sets of active or inactive molecules, which corresponds to different pooling strategies of the support set. Overall, the variants of the classic Similarity Search are highly similar to embedding-based few-shot learning methods except that they have a fixed instead of a learned molecule encoder.

Hyperparameter search. For the Similarity Search, there were two decisions to make which was firstly the similarity metric and secondly the question whether we should use a balancing strategy like shown in Section 3.4. We decided for the dot-product as a similarity metric and using the balancing strategy. These decisions were made by evaluating the models on the validation set.

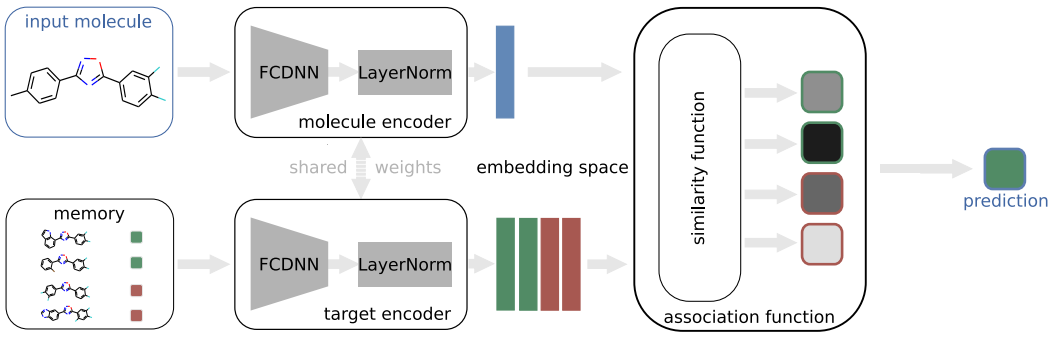

Figure A1: Schematic overview of the implemented Neural Similarity Search variant

A.1.3 Neural Similarity Search or Siamese networks: details and hyperparameters

A lot of related work already was done (Koch et al., 2015; Hertz et al., 2006; Ye and Guo, 2018; Torres et al., 2020). We adapted these ideas, such that a fully-connected deep neural network followed by a Layer Normalization (Ba et al., 2016) operation, $f_{\boldsymbol{w}}^{\mathrm{ME}}$ with adaptive parameters $\boldsymbol{w}$, is used in a Siamese fashion to compute the embeddings for the input molecule and the support set molecules. Within the association function block, pairwise similarity values for the input molecule and each support set molecule are computed, associating both embeddings via the dot product. Based on these similarity values, the activity for the input molecule is predicted, building the weighted mean over the support set molecule labels:

$$\hat{y} = \sigma\left(\tau^{-1}\frac{1}{N}\sum_{n=1}^{N} y'_n \; f^{\mathrm{ME}}(m)^T \; f^{\mathrm{ME}}(x_n)\right), \tag{A5}$$

where $\sigma(.)$ is the sigmoid function and τ is a hyperparameter in the range of $\sqrt{d}$. Note that this method uses a balancing strategy for the labels $y'_n = \begin{cases} N/(2\sqrt{N_A}) & \text{if } y_n = 1 \\ -N/(2\sqrt{N_I}) & \text{else} \end{cases}$, where N_A is the number of actives and N_I is the number of inactives of the support set. Figure A1 provides an schematic overview of the Neural Similarity Search variant.

We trained the networks using the Adam optimizer (Kingma and Ba, 2014) to minimize binary cross-entropy loss.

Hyperparameter search. We performed manual hyperparameter search on the validation set. We report the explored hyperparameter space (Table A2). Bold values indicate the selected hyperparameters for the final model.

A.1.4 ProtoNet: details and hyperparameters

Prototypical Networks (ProtoNet) (Snell et al., 2017), learn a prototype $\boldsymbol{r}$ for each class. Concretely, the support set Z is class-wise separated into $Z^+ := \{(x, y) \in Z \mid y = 1\}$ and $Z^- := \{(x, y) \in Z \mid y = -1\}$. For the subsets Z^+ and Z^- prototypical representations $\boldsymbol{r}^+$ and $\boldsymbol{r}^-$ can be computed by

$$\boldsymbol{r}^+ = \frac{1}{|Z^+|} \cdot \sum_{(x,y)\in Z^+} f^{\mathrm{ME}}(x) \tag{A6}$$

and

$$\boldsymbol{r}^- = \frac{1}{|Z^-|} \cdot \sum_{(x,y)\in Z^-} f^{\mathrm{ME}}(x). \tag{A7}$$

The prototypical representations $\boldsymbol{r}^+, \boldsymbol{r}^- \in \mathbb{R}^d$ and the query molecule embedding $\boldsymbol{m} \in \mathbb{R}^d$ are then used to make the final prediction:

$$\hat{y} = \frac{\exp(-\boldsymbol{d}(\boldsymbol{m}, \boldsymbol{r}^+))}{\exp(-\boldsymbol{d}(\boldsymbol{m}, \boldsymbol{r}^+)) + \exp(-\boldsymbol{d}(\boldsymbol{m}, \boldsymbol{r}^-))}, \tag{A8}$$

Table A2: Hyperparameter space considered for the Neural Search model selection. The hyperparameters of the best configuration are marked bold.

Hyperparameter	Explored values
Number of hidden layers	1, **2**, 4
Number of units per hidden layer	**1024**, 4096
Output dimension	**512**, 1024
Activation function	ReLU, **SELU**
Learning rate	0.0001, **0.001**, 0.01
Optimizer	**Adam**
Weight decay	**0**, $1 \cdot 10^{-4}$
Batch size	**4096**
Input Dropout	**0.1**
Dropout	**0.5**
Layer-normalization	False, **True**
• Affine	**False**
Similarity function	cosine similarity, **dot product**, MinMax similarity

where $\boldsymbol{d}$ is a distance metric.

Hyperparameter search. Hyperparameter search has been done in Stanley et al. (2021), to which we refer here. ECFP fingerprints and descriptors created by a GNN, which operates on the molecular graph, are fed into a fully connected neural network, which maps the input into an embedding space with the dimension of 512. (Stanley et al., 2021) use the Mahalanobis distance to measure the similarity between a query molecule and the prototypical representations in the embedding space. The learning rate is 0.001 and the batch size is 256. The implementation can be found here `https://github.com/microsoft/FS-Mol/blob/main/fs_mol/protonet_train.py` and important hyperparameters are chosen here `https://github.com/microsoft/FS-Mol/blob/main/fs_mol/utils/protonet_utils.py`.

Connection to Siamese networks and contrastive learning with InfoNCE. If instead of the negative distance $-\boldsymbol{d}(.,.)$ the dot product similarity measure with appropriate scaling is used, ProtoNet for two classes becomes equivalent to Siamese Networks. Note that in our study, another difference is that ProtoNet uses a GNN as encoder, whereas Siamese Networks use a descriptor-based fully-connected network as encoder. In case of dot product as similarity measure, the objective also becomes equivalent to contrastive learning with the InfoNCE objective (Oord et al., 2018).

A.1.5 IterRefLSTM: details and hyperparameters

(Altae-Tran et al., 2017) modified the idea of Matching Networks (Vinyals et al., 2016) by replacing the LSTM with their Iterative Refinement Long Short-Term Memory (IterRefLSTM). The use of the IterRefLSTM empowers the architecutre to update not only the embeddings for the input molecule but also adjust the representations of the support set molecules.

For IterRefLSTM, $\boldsymbol{m} = f^{\mathrm{ME}}_{\boldsymbol{\theta}_1}(m)$ and $\boldsymbol{x}_n = f^{\mathrm{ME}}_{\boldsymbol{\theta}_2}(x_n)$ are two potentially different molecule encoders for input molecule m and the support set molecules $x_1, \ldots, x_N$. The next step in IterRefLSTM is:

$$[\boldsymbol{m}', \boldsymbol{X}'] = \mathrm{IterRefLSTM}_L([\boldsymbol{m}, \boldsymbol{X}]).$$

Here, $\boldsymbol{m}'$ and $\boldsymbol{X}'$ contain the updated representations for the query molecule and the support set molecules. The IterRefLSTM denotes the function which updates these representations. The main property of the IterRefLSTM module is that it is permutation-equivariant, thus a permutation $\pi(.)$ of the input elements results in the permutation of output elements: $\pi([\boldsymbol{m}', \boldsymbol{X}']) = \mathrm{IterRefLSTM}_L(\pi([\boldsymbol{m}, \boldsymbol{X}]))$. The full architecture in invariant to permutations of the support set elements. For details, we refer to (Altae-Tran et al., 2017). The hyperparameter $L \in \mathbb{N}$ controls the number of iteration steps of the IterRefLSTM.

Table A3: Hyperparameter space considered for the IterRef model selection. The hyperparameters of the best configuration are marked bold.

Hyperparameter	Explored values
Molecule encoder	
• Number of hidden layers	**0**, 1, 2, 4
• Number of units per hidden layer	**1024**, 4096
• Output dimension	**512**, 1024
• Activation function	ReLU, **SELU**
• Input dropout	**0.1**
• Dropout	**0.5**
IterRef embedding layer	
• L	1, **3**
Similarity module:	
• Metric	cosine similarity, **dot product**, MinMax similarity
• Similarity space dimension	512, **1024**
Layer-normalization	False, **True**
• Affine	**False**, True
Training	
• Learning rate	0.0001, **0.001**, 0.01
• Optimizer	**Adam**, AdamW
• Weight decay	**0**, 0.0001
• Batch size	**2048**, 4096

As similarity module, the IterRefLSTM uses the following:

$$\boldsymbol{a} = \mathrm{softmax}\left(\boldsymbol{k}\left(\boldsymbol{m}', \boldsymbol{X}'\right)\right)$$
$$\hat{y} = \sum_{n=1}^{N} a_n \, y_n,$$

where $\hat{y}$ is the prediction for the query molecule. For the computation of the attention values $\boldsymbol{a}$, the softmax function is used. $\boldsymbol{k}$ is a similarity metric, such as the cosine similarity.

Hyperparameter search. All hyperparameters were selected based on manual tuning on the validation set. We report the explored hyperparameter space in Table A3. Bold values indicate the selected hyperparameters for the final model.

A.1.6 MHNfs: details and hyperparameters

The MHNfs consists of a molecule encoder, the context module, the cross-attention-module, and the similarity module. The molecule encoder is a fully-connected Neural Network, consisting of one layer with 1024 units. For the context module, a Hopfield layer with 8 heads is used and also the cross-attention module include 8 heads. We chose a concatenation of ECFPs and RDKit-based descriptors as the inputs for the MHNfs model. Notably, the RDKit-based descriptors were pre-processed in a way that instead of raw values quantils, which were computed by comparing a raw value with the distributation of all FS-Mol training molecules, were used. All descriptors were normalized based on the FS-Mol training data.

Hyperparameter search. All hyperparameters were selected based on manual tuning on the validation set. We report the explored hyperparameter space in Table A4. Bold values indicate the selected hyperparameters for the final model. Early stopping points for the different re-runs are chosen based on the ΔAUC-PR metric on the validation set. For the five re-runs the early-stopping points, that were automatically chosen by their validation metrics, were the checkpoints at epoch 94, 192, 253, 253 and 309.

Model training. Figure A2 shows the learning curve of an exemplary training run of a MHNfs model on FS-Mol. The left plot shows the loss on the training set and the right plot shows the

Table A4: Hyperparameter space considered for the MHNfs model selection. The hyperparameters of the best configuration are marked bold.

Hyperparameter	Explored values
Molecule encoder	
• Number of hidden layers	**0**, 1, 2, 4
• Number of units per hidden layer	**1024**, 4096
• Output dimension	**512**, 1024
• Activation function	ReLU, **SELU**
• Input dropout	**0.1**
• Dropout	**0.5**
Context module (hopfield layer)	
• Heads	**8**, 16
• Association space dimension	512 [512;2048]
• τ	22.6 [15;40]
• Dropout	0.1, **0.5**
Cross-attention module (transformer mechanism)	
• Heads	1, **8**, 10, 16, 32, 64
• Number units in the hidden feedforward layer	**567** [512; 4096]
• Association space dimension	1088 [512;2048]
• Dropout	0.1, **0.5**, 0.6, 0.7
• Number of layers:	**1**, 2, 3
Similarity module:	
• Metric	cosine similarity, **dot product**, MinMax similarity
• Similarity space dimension	512, **1024**
Layer-normalization	False, **True**
• Affine	**False**, True
Training	
• Learning rate	**0.0001**, 0.001, 0.01
• Optimizer	**Adam**, AdamW
• Weight decay	**0**, 0.0001
• Batch size	**4096**
• Warm-up phase (epochs)	**5**
• Constant learning rate phase (epochs)	25, **35**
• Decay rate	0.994
• Max. number of epochs	**350**

validation loss. The dashed line indicates the checkpoint of the model which was saved and then used for inference on the test set, whereas the stopping point was evaluated maximizing the ΔAUC-PR metric on the validation set.

Performance improvements in comparison to a naive baseline. Figure A3 shows a task-wise performance comparison between MHNfs and the frequent hitter model. Each point indicates a task in the test set and is colored according to their super-class membership. In 132 cases the MHNfs outperforms the frequent hitter model. In 25 cases the frequent hitter model yields better performance.

A.1.7 PAR: details and hyperparameters

The PAR model (Wang et al., 2021) includes a pre-trained GNN encoder, which creates initial embeddings for the query and support set molecules. These embeddings are fed into an attention mechanism module which also uses activity information of the support set molecules to create enriched representations. Another GNN learns relations between query and support set molecules.

Hyperparameter search. For details we refer to (Wang et al., 2021) and `https://github.com/tata1661/PAR-NeurIPS21/blob/main/parser.py`. All hyperparameters were selected based on manual tuning on the validation set. The hyperparameter choice for Tox21 (Wang et al., 2021) was used as a starting point. We report the explored hyperparameter space in Table A5. Bold values

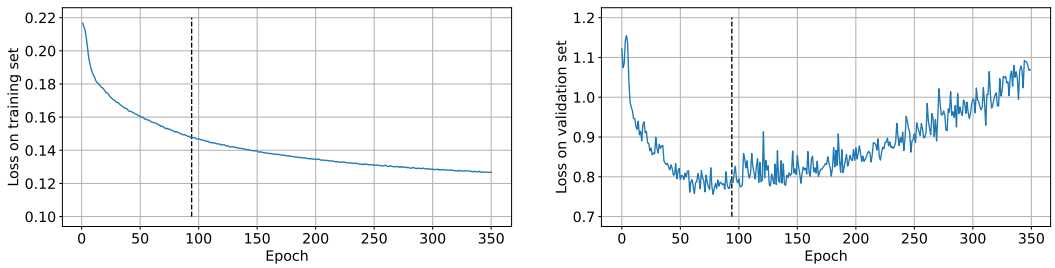

Figure A2: Exemplary MHNfs learning curve on FS-Mol. On the x-axis the number of epochs is displayed and on the y-axis thee training loss (left) and the validation loss (right) The dashed line indicates the determined early-stopping point which is determined based on ΔAUC-PR on the validation set.

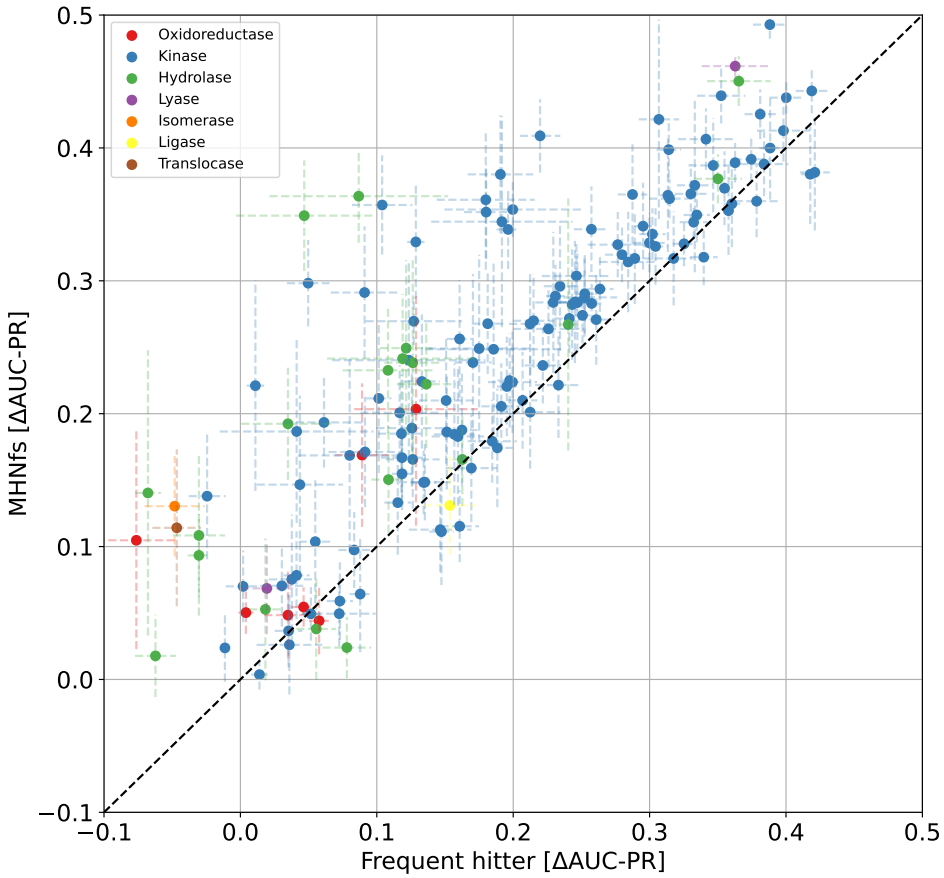

Figure A3: Performance comparison of MHNfs with the frequent hitter model. Each point refers to a task in the test set. Dashed lines indicate variablility across training re-runs and different test support sets. The most points are located above the dashed line, which indicates that MHNfs performs better than den FH baseline at this task.

Table A5: Hyperparameter space considered for the PAR model selection. The hyperparameters of the best configuration are marked bold.

Hyperparameter	Explored values
Training	
• Meta learning rate	$1.0 \cdot 10^{-05}$, $\mathbf{1.0 \cdot 10^{-04}}$, $1.0 \cdot 10^{-03}$, $1.0 \cdot 10^{-02}$
• Inner learning rate	0.01, **0.1**
• Update step	**1**, 2
• Update step test	**1**, 2
• Weight decay	$\mathbf{5.0 \cdot 10^{-05}}$, $1.0 \cdot 10^{-03}$
• Epochs	200000
• Eval. steps	2000
Encoder	
• Use pre-trained GNN	**yes**, no
Attention-based module	
• Map dimension	128, **512**
• Map layer	**2**, 3
• Pre fc layer	**0**, 2
• Map dropout	**0.1**, 0.5
• Context layer	**2**, 3, 4
Relation graph	
• Hidden dimension	8, 128, **512**
• Number of layers	2, **4**
• Number of layers for relation edge update	2, **3**
• Batch norm	yes, **no**
• Relation dropout 1	**0**, 0.25, 0.5
• Relation dropout 2	**0.2**, 0.25, 0.5

indicate the selected hyperparameters for the final model. Notably, we just report hyperparameter choices which were different from standard choices. We used a training script provided by (Wang et al., 2021), which can be found here `https://github.com/tata1661/PAR-NeurIPS21`.

A.2 Domain shift experiment

Experimental setup. For the domain shift experiment, we used the Tox21 dataset. This dataset consists of 12,707 chemical compounds, for which measurements for up to 12 different toxic effects are reported (Mayr et al., 2016; Huang et al., 2016a). It was published with a fixed training, validation and test split. State-of-the-art supervised learning methods that have access to the full training set reach AUC performance values between 0.845 and 0.871 (Klambauer et al., 2017; Duvenaud et al., 2015; Li et al., 2017, 2021; Zaslavskiy et al., 2019; Alperstein et al., 2019). For our evaluation, we re-cast Tox21 as a few-shot learning setting and draw small support sets from the 12 tasks. The compared methods were pre-trained on FS-Mol and obtain small support sets from Tox21. Based on the support sets, the methods had to predict the activities of the Tox21 test set. Note that there is a strong domain shift from drug-like molecules of FS-Mol to environmental chemicals, pesticides, food additives of Tox21. The domain shift also concerns the outputs where a shift from kinases, hydrolases, and oxidoreductases of FS-Mol to nuclear receptors and stress responses of Tox21 is present.

Methods compared. We compared the new method **MHNfs**, the runner-up method **IterRefLSTM**, and **Similarity Search** — since it has been widely used for such purposes for decades (Cereto-Massagué et al., 2015).

Training and evaluation. We followed the procedure of Stanley et al. (2021) for data-cleaning, preprocessing and extraction of the fingerprints and descriptors used in FS-Mol. After running the cleanup step, 8,423 molecules remained for the domain shift experiments. From the training set, 8 active and 8 inactive molecules per task were randomly selected to build the support set. The test set molecules were used as query molecules. The validation set molecules were not used at all. During test-time, a support set was drawn ten times for each task. Then, the performance of the models were

Table A6: Results of the domain shift experiment on Tox21 [AUC, ΔAUC-PR]. The best method is marked bold. Error bars represent standard deviation across training re-runs and draws of support sets

Method	AUC	ΔAUC-PR
Similarity Search (baseline)	.629 ± .015	.061 ± .008
IterRefLSTM (Altae-Tran et al., 2017)	.664 ± .018	.067 ± .008
MHNfs (ours)	**.679** ± .018	**.073** ± .008

Table A7: Results of the ablation study on FS-Mol [AUC, ΔAUC-PR]. The error bars represent standard deviation across training re-runs and draws of support sets. The p-values indicate whether the difference between two models in consecutive rows is significant.

Method	AUC	ΔAUC-PR	p_{AUC}[a]	$p_{\Delta\mathrm{AUC-PR}}$[a]
MHNfs (CM+CAM+SM)	.739 ± .005	.241 ± .006		
MHNfs -CM	.737 ± .004	.240 ± .005	0.030	0.002
MHNfs -CM -CAM	.719 ± .006	.223 ± .006	< 1.0e-8	<1.0e-8
Similarity Search	.604 ± .003	.113 ± .004	<1.0e-8	< 1.0e-8
IterRefLSTM (Altae-Tran et al., 2017)[b]	.730 ± .005	.234 ± .005	<1.0e-8	8.73e-7

[a] paired Wilcoxon rank sum test [b] IterRefLSTM is compared to MHNfs -CM

evaluated for these support sets, using the area under precision-recall curve (AUC-PR), analogously to the FS-Mol benchmarking experiment reported as the difference to a random classifier (ΔAUC-PR), and the area under receiver operating characteristic curve (AUC) metrics. The performance values report the mean over all combinations regarding the training reruns and the support set sampling iterations. Error bars indicate the standard deviation.

Results. The Hopfield-based context retrieval method has significantly outperformed the IterRefLSTM-based model ($p_{\Delta\mathrm{AUC-PR}}$-value $3.4\mathrm{e}{-5}$, p_{AUC}-value 2.5e-6, paired Wilcoxon test) and the Classic Similarity Search ($p_{\Delta\mathrm{AUC-PR}}$-value 2.4e-9, p_{AUC}-value 7.6e-10, paired Wilcoxon test) and therefore showed robust performance on the toxicity domain, see Table A6. Notably, all models were trained on the FS-Mol dataset and then applied to the Tox21 dataset without adjusting any weight parameter.

A.3 Details on the ablation study

The MHNfs has two new main elements compared to the previous state-of-the art method IterRefLSTM, which are the context module and the cross-attention-module. In this ablation study we aim to investigate i) the importance of all design elements, which are the context module, the cross-attention module, and the similarity module, and ii) the superiority of the cross-attention module compared to the IterRefLSTM module.

A.3.1 Ablation study A: comparison against IterRefLSTM

For a fair comparison between the cross-attention module and the IterRefLSTM we used a pruned MHN version ("MHNfs -CM") which has no context module and compared it with the IterRefLSTM model. The evaluation includes five training re-runs each and ten different support set samplings. The results, reported as the mean across training re-runs and support sets, can be seen in Table A7. We performed a paired Wilcoxon rank sum test for both the AUC and the ΔAUC-PR metric. Both p-values indicate high significance.

A.3.2 Ablation study B: all design elements

We evaluate the performance of all main elements within the MHNfs, which are the context module, the cross-attention module, the similarity module and the molecule encoder. For this analysis, we start with the complete MHNfs which includes all modules and report AUC and ΔAUC-PR performance values. Then, we iteratively omit the individual modules, measuring whether there is a

Table A8: Results of the ablation study on Tox21 [AUC, ΔAUC-PR]. The error bars represent standard deviation across training re-runs and draws of support sets. The p-values indicate whether a model is significantly different to the MHNfs in terms of the AUC and ΔAUC-PR metric.

Method	AUC	ΔAUC-PR	p_{AUC}[a]	$p_{\Delta\mathrm{AUC-PR}}$[a]
MHNfs (CM+CAM+SM)	.679 ± .018	.073 ± .008		
MHNfs -CM	.662 ± .028	.069 ± .012	6.28e-8	0.002
MHNfs -CM -CAM	.640 ± .018	.057 ± .009	<1.0e-8	<1.0e-8
Similarity Search	.629 ± .015	.061 ± .008	<1.0e-8	<1.0e-8
IterRefLSTM	.664 ± .018	.067 ± .008	2.53e-6	3.38e-5

[a] paired Wilcoxon rank sum test

significant performance difference with and without the module. Table A7 shows the results, where performance values for the full MHNfs, a MHNfs model without the context module ("MHNfs -CM") and a MHNfs module without the context and the cross-attenion module ("MHNfs -CM -CAM") is included. Notably, the model without the context module and without the cross-attention module just consists of a learned molecule encoder and the similarity module. We evaluted the impact of the learned molecule encoder by replacing it with a fixed encoder, which maps a molecule to its descriptors. The model with the fixed encoder is a classic chemoinformatics method which is called Similarity Search (Cereto-Massagué et al., 2015).

For the evaluation, we performed five training re-runs for every model and sampled ten different support sets for every task. Table A7 shows the results in terms of AUC and ΔAUC-PR. We performed paired Wilcoxon rank sum tests on both metrics, comparing two methods in consecutive rows in the table. The table shows that every module has a significant impact as omitting a module results in a significant performance drop. The comparison between the MHNfs version without the context module and without the cross-attention module with the Similarity Search showed a significant superiority of the learned molecule encoder in comparison to the fixed encoder.

A.3.3 Ablation study C: Under domain shift on Tox21

Referring to Section A.3.2, the context module and the cross-attention module showed their importance for the global architecture. This importance gets even more pronounced for the domain shift experiment on Tox21 as one can see in Table A8.

Again, five training re-runs and ten support set draws are used for evaluation. Including the context module makes a clear and significant difference for both metrics AUC and ΔAUC-PR.

A.4 Generalization to different support set sizes

In this section, we test the ability of MHNfs to generalize to different support set sizes. During training in the FS-Mol benchmarking setting, the MHNfs model has access to support sets of size 16. However, at inference, the support set size might be different. Figure A4 provides performance estimates of the support-set-size-16 MHNfs models on other support set sizes. Note that the estimates could be seen as approximate lower bounds of the predictive performance on settings with different support set sizes (y-axis labels). For a model used in production or in a real-world drug discovery setting, MHNfs should be trained with varying support set sizes that resemble the distribution of real drug discovery projects.

A.5 Generalization to different context sets

In this section, we test the ability of MHNfs to generalize to different context sets. While the FS-Mol training split is used as a context during training, we assessed whether our model is robust to different context sets for inference. To this end we preprocessed the GEOM dataset (Axelrod and Gomez-Bombarelli, 2022) from which we used 100,000 molecules that passed all pre-processing checks. From this set, we sample 10,000 molecules as context set for MHNfs. Because GEOM contains drug-like molecules, similar to FS-Mol the predictive performance remains stable (see Table A9).

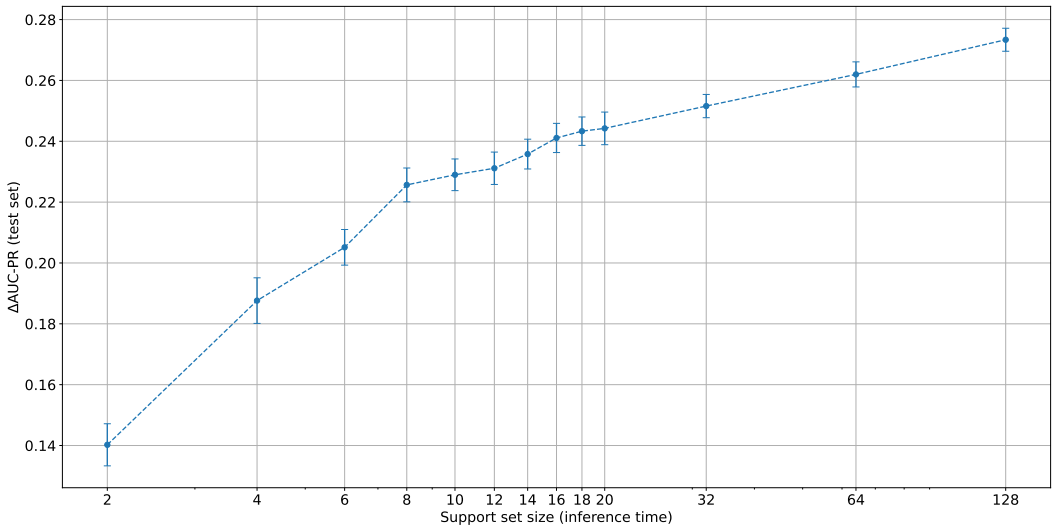

Figure A4: Performance of MHNfs for different support set sizes during inference time. The MHNfs models are trained with support sets of the size 16.

Table A9: MHNfs performance for different context sets [ΔAUC-PR]. The error bars represent standard deviation across training re-runs and draws of support sets.

Dataset used as a context	ΔAUC-PR
FS-Mol (Stanley et al., 2021)	.2414 ± .006
GEOM (Axelrod and Gomez-Bombarelli, 2022)	.2415 ± .005

A.6 Details and insights on the context module

The context module replaces the initial representations of query and support set molecules by a retrieval from the context set. The context set is a large set of molecules and covers a large chemical space. The context module learns how to replace the initial molecule embeddings such that the context-enriched representations are put in relation to this large chemical space and still contains all necessary information for the similarity-based prediction part. Figure A5 shows the effect of the context module for the MHNfs model. Extreme initial embeddings, such as the purple embedding on the right, are pulled more into the known chemical space, represented by the context molecules. Notably, the replacement described above is a soft replacement, because also the initial embeddings contribute to the context-enriched representations due to skip-connections.

A.7 Reinforcing the covariance structure in the data using modern Hopfield networks

We follow the argumentation of (Fürst et al., 2021, Theorem A3) that retrieval from an associative memory of a MHN reinforces the covariance structure.

Let us assume that we have one molecule embedding from the query set $\boldsymbol{m} \in \mathbb{R}^d$ and one molecule embedding from the support set $\boldsymbol{x} \in \mathbb{R}^d$ and both have been enriched with the context module with memory $\boldsymbol{C} \in \mathbb{R}^{d \times M}$(ignoring linear mappings):

$$\boldsymbol{m}' = \boldsymbol{C}\,\mathrm{softmax}(\beta \boldsymbol{C}^T \boldsymbol{m}) \tag{A9}$$

$$\boldsymbol{x}' = \boldsymbol{C}\,\mathrm{softmax}(\beta \boldsymbol{C}^T \boldsymbol{x}) \tag{A10}$$

Then the similarity of the retrieved representations as measured by the dot product can be expressed in terms of covariances:

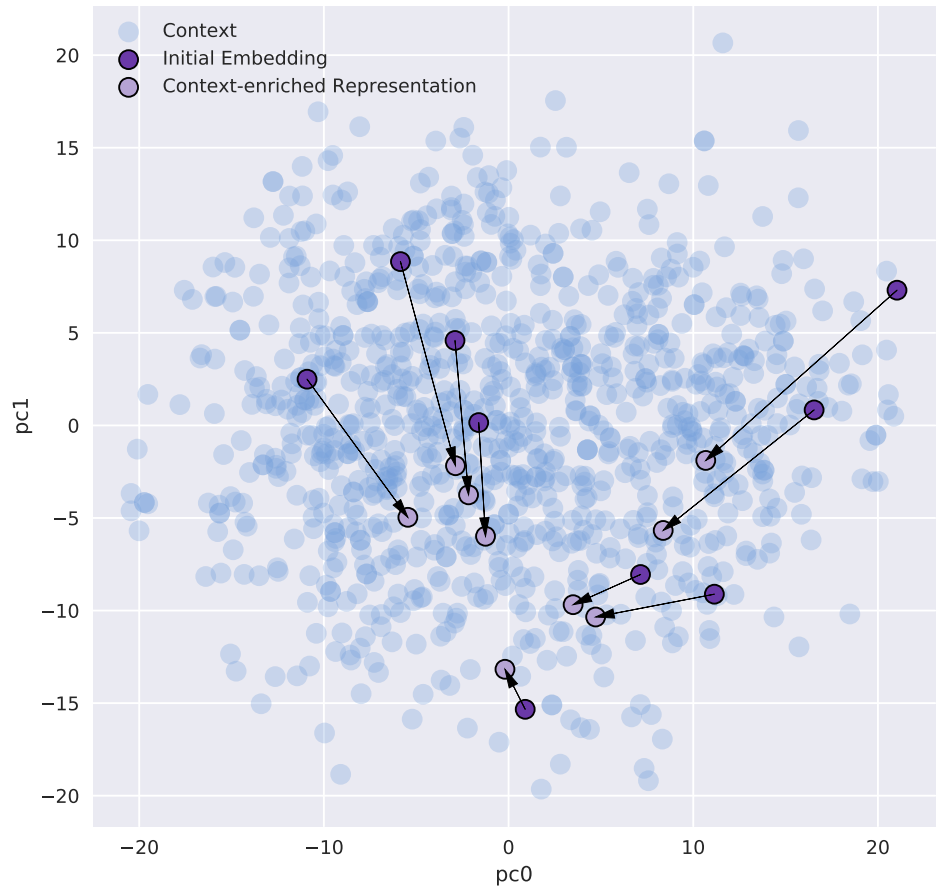

Figure A5: PCA plot of molecule embeddings. Each dot in the plot represents a molecule embedding, of which the first two principal components are displayed on the x- and y-axis. Blue dots represent context molecules. Dark purple dots represent initial embeddings for some exemplary molecules, of which some exhibit extreme characteristics and are thus located away from the center. Arrows and light purple dots represent the enriched molecule embeddings after the retrieval step. Especially molecules from extreme positions are moved stronger to the center and thus are more similar to known molecules after retrieval.

$$\boldsymbol{m}'^T \boldsymbol{x}' = \text{softmax}(\beta \boldsymbol{C}^T \boldsymbol{m})^T \boldsymbol{C}^T \boldsymbol{C} \text{softmax}(\beta \boldsymbol{C}^T \boldsymbol{x}) = \tag{A11}$$

$$= (\overline{\boldsymbol{c}} + \text{Cov}(\boldsymbol{C}, \boldsymbol{m})^T \boldsymbol{m})^T \ (\overline{\boldsymbol{c}} + \text{Cov}(\boldsymbol{C}, \boldsymbol{x}) \boldsymbol{x}), \tag{A12}$$

where $\overline{\boldsymbol{c}}$ is the row mean of $\boldsymbol{C}$ and following the *weighted covariances* are used:

$$\text{Cov}(\boldsymbol{C}, \boldsymbol{m}) = \boldsymbol{C} \text{J}^{\text{m}}(\beta \boldsymbol{C} \boldsymbol{m}) \boldsymbol{C}^T \qquad \text{Cov}(\boldsymbol{C}, \boldsymbol{x}) = \boldsymbol{C} \text{J}^{\text{m}}(\beta \boldsymbol{C} \boldsymbol{x}) \boldsymbol{C}^T. \tag{A13}$$

$\text{J}^{\text{m}} : \mathbb{R}^M \mapsto \mathbb{R}^{M \times M}$ is a mean Jacobian function of the softmax (Fürst et al., 2021, Eq.(A172)).

The Jacobian J of $\boldsymbol{p} = \text{softmax}(\beta \boldsymbol{a})$ is $\text{J}(\beta \boldsymbol{a}) = \beta \ \left(\text{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^T\right)$.

$$\boldsymbol{b}^T \text{J}(\beta \boldsymbol{a}) \ \boldsymbol{b} \ = \ \beta \ \boldsymbol{b}^T \left(\text{diag}(\boldsymbol{p}) \ - \ \boldsymbol{p} \, \boldsymbol{p}^T\right) \ \boldsymbol{b} \ = \ \beta \ \left(\sum_i p_i \, b_i^2 \ - \ \left(\sum_i p_i \, b_i \right)^2 \right) , \tag{A14}$$

this is the second moment minus the mean squared, which is the variance. Therefore, $\boldsymbol{b}^T \text{J}(\beta \boldsymbol{a}) \boldsymbol{b}$ is β times the covariance of $\boldsymbol{b}$ if component i is drawn with probability p_i of the multinomial distribution $\boldsymbol{p}$. In our case the component i is context sample $\boldsymbol{c}_i$. J^{m} is the average of $\text{J}(\lambda \boldsymbol{a})$ over $\lambda = 0$ to $\lambda = \beta$.

Note that we can express the enriched representations using these covariance functions:

$$\boldsymbol{m}' = (\overline{\boldsymbol{c}} + \mathrm{Cov}(\boldsymbol{C}, \boldsymbol{m})^T \boldsymbol{m}) \tag{A15}$$

$$\boldsymbol{x}' = (\overline{\boldsymbol{c}} + \mathrm{Cov}(\boldsymbol{C}, \boldsymbol{x})^T \boldsymbol{x}), \tag{A16}$$

which connects retrieval from MHNs with reinforcing the covariance structure of the data.