Drug Repurposing using Link Prediction on Knowledge Graphs with Applications to Non-Volatile Memory

Sarel Cohen[1], Moshik Hershcovitch[2], Martin Taraz[1], Otto Kißig[1], Andrew Wood[3], Daniel Waddington[2], Peter Chin[3], and Tobias Friedrich[1]

[1] Hasso Plattner Institute, Potsdam
{sarel.cohen, tobias.friedrich}@hpi.de
{martin.taraz, otto.kissig}@student.hpi.de,
[2] IBM Research
moshikh@il.ibm.com
daniel.waddington@ibm.com,
[3] Boston University
{aewood, spchin}@bu.edu

Abstract. The active global SARS-CoV-2 pandemic caused more than 167 million cases and 3.4 million deaths worldwide. The development of completely new drugs for such a novel disease is a challenging, time intensive process and despite researchers around the world working on this task, no effective treatments have been developed yet. This emphasizes the importance of *drug repurposing*, where treatments are found among existing drugs that are meant for different diseases. A common approach to this is based on *knowledge graphs*, that condense relationships between entities like drugs, diseases and genes. Graph neural networks (GNNs) can then be used for the task at hand by predicting links in such knowledge graphs. Expanding on state-of-the-art GNN research, Doshi *et al.* recently developed the DR-COVID model. We further extend their work using additional output interpretation strategies. The best aggregation strategy derives a top-100 ranking of candidate drugs, 32 of which currently being in COVID-19-related clinical trials. Moreover, we present an alternative application for the model, the generation of additional candidates based on a given pre-selection of drug candidates using collaborative filtering. In addition, we improved the implementation of the DR-COVID model by significantly shortening the inference and pre-processing time by exploiting data-parallelism. As drug repurposing is a task that requires high computation and memory resources, we further accelerate the post-processing phase using a new emerging hardware — we propose a new approach to leverage the use of high-capacity Non-Volatile Memory for aggregate drug ranking.

Keywords: Drug Repurposing, Knowledge Graphs, Link Prediction, Collaborative Filtering, Non-Votile Memory, NVM, MCAS, Python, PyMM

1 Introduction

With the novel coronavirus, a global pandemic with serious socio-economic implications for most parts of our daily lives is active [13]. The limited ability to take precautions for an unsuspected event like this and the rapid spread make finding an effective treatment as necessary as difficult, since the disease-specific knowledge is limited at the beginning and human lives are lost every day. Known and approved drugs happen to be well-studied, thus, they pose a good starting point for swift development of treatments, and an emerging tactic in fighting the pandemic [19]. DrugBank, an extensive database compiling information about drugs approved by the US Food and Drug Administration as well as experimental drugs, contained more than 2 300 approved drugs and over 4 500 experimental drugs as of 2018; both with a strong upward trend [22]. This emphasizes the need for computer aided development of treatments.

Drug repurposing with knowledge graphs, as first described by [1], is the current state-of-the-art approach for finding possible treatments for novel diseases among known drugs using machine learning. Applying drug repurposing allows for a better way to maneuver through the pandemic. It can lead to better treatments for patients infected with one of the COVID-19 strains and a better understanding of the characteristics of the individual strains. Today, we approach the problem of drug repurposing using machine learning, focusing on deep learning methods. The idea of predicting unknown links between entities in a knowledge graph is traditionally known as *Collaborative Filtering*, as described by [17]. In this work we expand on the concept of *graph embeddings*, which map a fixed-size feature vectors to graph nodes and relations. A state-of-the-art technique for the creation of such embeddings based on deep neural networks (DNNs) is TransE [2].

Knowledge graph embeddings are already utilized to solve different tasks related to drug discovery, e.g., they are used to predict potential drug targets for diseases to reduce cost and increase speed in the drug development process in general [26]. Regarding the specific application of drug repurposing relying on edge prediction in a knowledge graph of biomedical data (see Section 2), [6] present a novel classification approach to this problem by implementing and merging various different ideas and techniques into one ensemble classifier. At its core, they deploy a DNN with an encoder-decoder structure. The encoder mechanism of it, which is based on the *Decagon* graph neural network by [28], was initially proposed for the prediction of side effects of concurrent drug use.

Our Contribution. In this paper we extend the work done by Doshi and Chepuri [4]. Specifically we continue our work in Drug Repurposing [11, 12]. We offer the following contributions to the complex networks community analyzing medicine networks:

1. We improve the post prediction step of [4] by using a clustering of similar diseases and increasing by more than 50% the number of predicted drugs in the top-100 that were or are in clinical trials.

2. We explore the additional application of finding drug candidates similar to a manually pre-selected candidate using collaborative filtering on the same model output. We show that many drugs that are in clinical trial can be found by detecting the drugs that are the most similar (*e.g.* using cosine-distance on the embedding of the drugs) to a given known drug (or a subset of drugs) which is or was in clinical trials.
3. We re-implement[4] the model described by [4] and improve it by allowing flexible neighborhood capture sizes. We also improve the implementation by [12] by improving training speed, inference time, readability and by reducing pre-processing time from 30 minutes to 2 minutes by leveraging matrix operations. We further extend the implementation to support Self-Label-Enhancement.

We also contribute to the *way* drug repurposing is computed. Drug repurposing is a task that requires high computation and memory resources. The emerging hardware of Intel Optane Persistent Memory Modules (Optane-PM) communicates via the memory bus, mitigating bottlenecks such as PCI-express lane availability, using the same interface as DRAM. While there are other Persistent Memory technologies, Optane-PM being the most mature product on the market is based on 3D-XPoint (3DXP) technology and operates at a cache-line granularity with a latency of around 300ns [8], which is more than an order of magnitude faster than the current state of the art NVMe SSDs, but approximately three times slower than DRAM. Additionally, it has high capacity which is 8x larger than the available DRAM — a single DIMM of Optane-PM can reach 512GB. We note that it is practically necessary to use Optane-PM as the scale of the problem increases [24, 23].

To the best of our knowledge, in this paper, we show for the first time an application of the emerging Optane-PM for the task of Drug Repurposing. We generate a large dataset for the Drug Repurposing problem by extending (both vertically and horizontally) the dataset we have and evaluate two simple aggregation strategies which are implemented and processed on the Optane-PM . We obtain fast and promising results for the use of Optane-PM to process large datasets in the context of Drug Repurposing.

2 Dataset

Our work relies on the Drug Repurposing Knowledge Graph (DRKG) by [7], which compiles data from different biomedical databases. It contains $97,238$ entities belonging to 13 entity types and $5,874,261$ triplets belonging to 107 edge types. We restrict ourselves to 98 edge types between 4 entity types, namely gene, compound, anatomy and disease, which leaves us with a knowledge graph with $69,036$ entities and $4,885,854$ edges. In particular, it contains drugs and

[4] Our implementation of the experiments and the model can be found here: https://drive.google.com/file/d/1hYxMe3AFwcJ4UKsn8SPsZVPW3buXe0u4/view?usp=sharing.

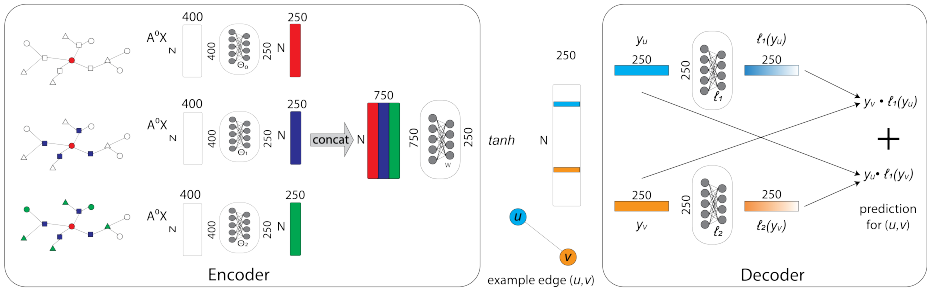


Fig. 1. The architecture of our model as described in Section 3.

substances as *compound* entities, as well as different COVID-19 variants as *disease* entities. The edge types include e.g. *compound-treats-disease* edges, which is the kind of edge our model predicts.

One part of DRKG are the precomputed TransE embeddings trained using `dgl-ke` by [27]. To train our model to predict whether a given edge in some *compound-treats-disease* relation exists, we have to create suitable training data. To provide our model with both positive and negative samples for training, for each positive edge we sample 30 non-edges in the dataset, which results in a ratio similar to Dr-COVID. This process tries to account for the imbalance of edges and non-edges in the ground truth. The set of edges included in the dataset is not complete, however, it is quite certain to be correct. Consequently, the positive edges are given a higher weight in the loss calculation, and the higher number of negative edges (which are not certain to be truly negative) are given a lower weight. To prevent too much imbalance in the individual minibatches, we use a weighted random batch sampler that over-samples the positive samples yielding an expected ratio of $1 : 1.5$ of positive to negative samples in each batch.

3 Model Architecture

A Graph Neural Network (GNN) is a message passing framework where vertex embeddings are passed along edges of a graph. A single GNN layer traditionally performs a single round of message passing where messages are transformed via an *edge function*, are collected together into a single message via an *aggregator function*, and finally are used to produce new messages using a *vertex function*. We refer the reader to [28, 18, 10] for a more in-depth description.

In our experiments, we used a traditional encoder-decoder architecture using a two-layer GNN encoder and a custom decoder. The architecture of our model is illustrated in Figure 1. It consists of a SIGN [5] architecture encoder, which provides an embedding $y \in R^{250}$ for each node. We apply *tanh* to the encoder output and forward it into our decoder. Given two nodes u, v, the decoder takes their encodings y_u, y_v and assigns a score $s_{u,v} \in [0, 1]$, which measures the probability for an edge between nodes u and v to exist. The decoder consists of two linear layers $\ell_1(u)$ and $\ell_2(v)$ that process the encodings y_u and y_v via

a sigmoid function, that is, $\sigma(y_v \cdot \ell_1(y_u) + y_u \cdot \ell_2(y_v))$. The loss of the model is computed using a binary cross entropy loss with logits with weights set as described in Section 2.

Implementation. The dataset presents itself as a list of triples, each posing source, relation-type and sink of an edge. This is accompanied by precomputed knowledge graph embeddings. For the preprocessing we first filter out the edges belonging to the part of the knowledge graph we restrict ourselves to. We then construct a graph with the help of DGL [21]. To compute the neighborhood embeddings we feed into the model, we first derive an adjacency matrix $A \in \{0,1\}^{n \times n}$ from the reduced graph, from which the edges we try to predict, i.e., *compound-treats-disease* edges, have been removed. We then derive the normalized graph Laplacian $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ where $D_{i,i}$ is the degree of node i. Suppose $X \in R^{n \times 400}$ is the matrix of graph embeddings for the n nodes, then the kth neighborhood is defined as $\tilde{A}^k X$.

4 Output Interpretation

In this section we present different strategies for interpreting the scores that the model outputs for the application of predicting the top-k most promising compound nodes for a given set of disease nodes D. Note that this is important as there are multiple COVID-19 diseases. Let n be the total amount of compound nodes. Predicting all $n \cdot |D|$ edge combinations, our model yields a matrix of scores $S \in R^{|D| \times n}$. For each of the following strategies we first perform a standardization of the scores per disease using $\hat{s}_{dc} = \frac{s_{dc} - \mu(s_{d*})}{\sigma(s_{d*})}$, where d is the index of a disease in D, c being the index of the compound, $\mu(s_{d*})$ and $\sigma(s_{d*})$ denote the mean and standard deviation over all diseases.

Certain "mild" diseases may be affected by plenty of compounds resulting in those being linked more likely. The standardization helps to achieve a better comparability across different diseases, allowing us to identify the suited compounds for every disease individually and compare those. However, this could also give good scores to some compounds in the case of diseases with no "good" scores in the first place, potentially yielding some less useful proposals.

An aggregation strategy takes our matrix of standardized scores $(\hat{s}_{dc})$ and derives a list of compounds from it, the top-k of which are our result. We propose the following aggregation strategies. For **global score mean**, we calculate the means of $(\hat{s}_{dc})$ along axis 0, that is, over all diseases per compound; then we sort the compounds by their respective scores and select the top-k. For **global score maximum**, we find the maxima of $(\hat{s}_{dc})$ along axis 0; then again we sort the compounds and select the top-k. For **union over disease rankings**, we calculate top-x compounds per disease with x as small as possible such that we get at least k unique compounds in the union. We then concatenate all those top-x lists together to get a top-k compound list.

We also propose **greedy max-min fairness**. Inspired by a game-theoretic approach from auction theory, where we think of the COVID-strains as players

and the compounds as items from which we can only pick a small set, we try to heuristically find a set of compounds that will maximize the COVID strain whose total score is the minimum. Note that Global Score Mean can be considered as allocating the drugs to the COVID strains in a way that obtains the maximum social welfare. In contrast, in the Greedy Max Min Fairness we allocate the candidate drugs among the COVID-strains in a way favoring fairness over social welfare. More precisely, we rank the drugs by iteratively selecting the drug that benefits the disease with the lowest sum of scores over all already selected drugs. From this ordering we then pick the top-k drugs. Because our standardized model outputs $\hat{s}_{dc}$ can be negative, we normalize these by additively shifting them into the positive numbers. This bias however does not interfere with the resulting order because it increases uniformly on all parts of the sum.

Furthermore, in **cluster score maximum**, grouping similar disease types can be used to enhance the accuracy of our top-k predictions. We perform such a grouping using the k-means clustering algorithm. For each cluster, which now represents a group of similar diseases, we use a mean reduction to calculate the score of a compound and then reduce to the maximum across these clusters. A sensible number of clusters to create can be chosen by performing a principal component analysis (PCA) [15] on the standardized scores. Lastly, for **union over cluster rankings**, we perform the top-x selection on clusters calculated with the clustering method described above. This not only allows us to use a greater x because we have fewer lists to pick from, but also to get more consistent top picks because of the internal averages that we apply inside each cluster.

5 Collaborative Filtering

Suppose we already have pre-selected some candidates for clinical trials. Now we would like to identify similar candidates that could be interesting. This new application can be approached using collaborative filtering on our model output. We measure the similarity[5] along the model's edge predictions per compound.

We test this application by ranking the remaining compounds of our dataset by the cosine similarity to pre-selected candidates. Our pre-selections are sampled randomly from the clinical trial dataset. In the case of one single pre-selected candidate, for selecting the top-100 drugs ranked by similarity to the pre-selected candidate we get a mean of 18 (min. 0, max. 32) hits. Conducting the experiment with 15 pre-selected candidates and selecting drugs corresponding to the top-100 of a global ranking of all similarities yields on average 18 (min. 0, max. 37) hits.

6 Rank Aggregation Using Non-Volatile Memory

In this section we demonstrate the use of Non-Volatile Memory for aggregate drug prediction. In general, Optane-PM can perform arbitrary matrix calculations

[5] To precisely define the cosine similarity between two given drugs i, j, let $\hat{s}_{*i}, \hat{s}_{*j}$ be their prediction scores along the disease dimension. Then their similarity is defined as $\hat{s}_{*i} \cdot \hat{s}_{*j}$.

Aggregation strategy	# hits
Single Disease (median)	20
Global Score Maximum	22
Global Score Mean	30
Greedy Max Min Fairness	23
Cluster Score Maximum with KMeans(k=8)	18
Cluster Score Maximum with KMeans(k=3)	20
Union over Disease Rankings (Dr-COVID, [4])	21
Union over Cluster Rankings with KMeans(k=8)	24
Union over Cluster Rankings with KMeans(k=3) [12]	**32**

Table 1. Hits of proposed candidates in actual clinical trials.

while providing significantly more capacity than ordinary DRAM and optionally providing persistence. We chose to use Optane-PM to implement the Global Score Mean and Global Score Maximum aggregation strategies. We chose these strategies for their decent prediction performance (see Table 1) and because they were easy to implement using Optane-PM . We note that we did not select strategies which used clustering due to an incompatibility between scikit-learn [16, 3] (the package clustering was implemented with) and the Optane-PM library. We show that by using Optane-PM , we can process datasets faster than with traditional storage methods such as DRAM + NVMe SSD or memory mapping.

To demonstrate the utility of Optane-PM , we artificially increased the size of the data being operated on. To do so, we extended our ranking matrix of size 2MB by concatenating entries both vertically and horizontally. Using this scheme, we created data matrices of sizes 33, 66, 131, and 261GB. This was necessary to show the performance difference between Optane-PM and other storage methods.

6.1 Interacting with Optane-PM

We use a Python 3 library called PyMM to interface with Optane-PM. PyMM has been developed as part of the Memory Centric Active Storage (MCAS) system[20]. PyMM provides a set of abstractions and framework for managing Python variables in locally-attached Optane-PM. For more details regarding MCAS and PyMM, we direct the reader to [20, 24]. Data that is stored in PyMM is persistent and can be accessed and manipulated in-place, directly on device, without requiring a copy or transfer to DRAM. Using PyMM, we store our large data matrices and create aggregate rankings using the two strategies mentioned in Section 6.

7 Experiments

7.1 Link Prediction

Our experiments are twofold. We first train our link prediction model to generate probability scores to a candidate edge using an encoder-decoder architecture

described in Section 3. We implement the model using PyTorch. We train it using the Adam optimizer [9]. We use 90% of the data for training and the rest for validation. The training is performed on Google Colab utilizing a Nvidia Tesla T4 and it takes ∼2 minutes to prepare the graph dataset. We train our model using 25 epochs with a starting learning rate of 10^{-5} and a weight decay of 10^{-2}. Each training epoch took us 30 seconds, which is a significant improvement over the 610 seconds of the implementation by [4] and can be attributed to the exploitation of data parallelism we added.

Using this model, we generate prediction matrices by sorting drugs for each covid strain using the learned model. Our first experiment concludes with using aggregation strategies to process the prediction matrix to determine the final drug rankings. These drug rankings are then compared against drugs which are being currently tested in clinical trials.

7.2 Scalable Drug Rank Aggregation

Our second experiment compares Optane-PM against other storage methods. We measure the runtime of using different (simple) aggregation strategies implemented on Optane-PM against their implementations using DRAM. We augment the prediction matrices using the procedure mentioned in Section 6 to produce arbitrarily large data. Our experiment compares the following implementations:

1. **PyMM implementation.** Prediction matrices are stored on Optane-PM and are processed on device using the Global Score Mean and Global Score Maximum strategies.
2. **DRAM implementation.** Prediction matrices are stored entirely on NVMe SSD, and then transferred and processed in DRAM. We note that this implementation is only possible if the machine has sufficient DRAM.
3. **MMAP_384 implementation.** Prediction matrices are stored on NVMe SSD. During processing, the required data is loaded from NVMe SSD to DRAM using NumPy's Memory-Mapping functionality. In this implementation we have 384GB of DRAM. This configuration allows the entire dataset to be loaded into DRAM, meaning no evictions will occur. Therefore, this is a best-case scenario for memory mapping performance.
4. **MMAP_64 implementation.** This implementation is a more realistic memory mapping scenario. While this implementation is almost identical to the previous one, the amount of DRAM has been restricted to 64GB. This means that the memory mapping routine will need to evict data from DRAM during processing. From a Cloud/infrastructure perspective, this simulates a low-cost machine memory mapping scenario.

Our experiments were conducted on a Lenovo SR650 2U server equipped with two Intel Xeon Gold 6248 (2.5GHz) processors supporting 80 CPU hardware threads. The server is also equipped with 384GB (12x32GB) of DDR4 DRAM and 1.5TB of Optane-PM (12x128GB) as well as two NVMe SSD disks with 3TB each.

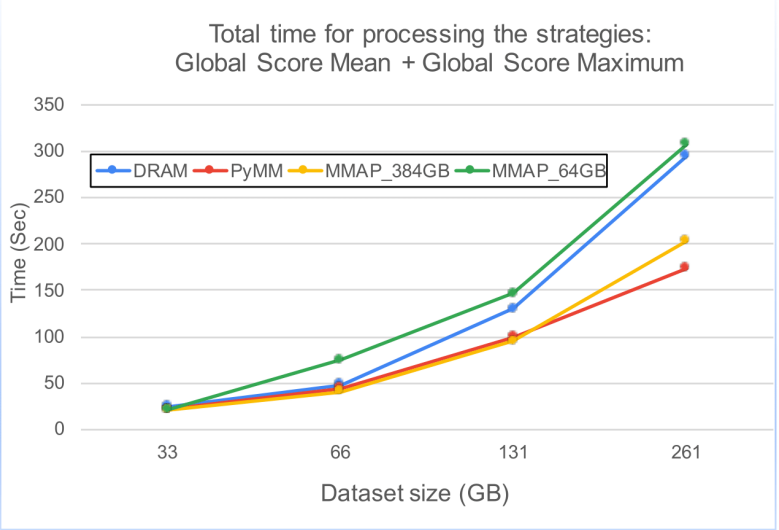
8 Evaluation

8.1 Link Prediction and Drug Aggregation Performance

To test our link prediction model, we compare the top-100 drugs for SARS-CoV2 computed by our learned model to those predicted utilizing the weights of [4]. While their model's top-100 predictions include 22 drugs present in clinical trials, we only reach 15. We suspect the hand-made adjustments to the dataset utilizing undisclosed data sources are responsible for this discrepancy, as this is the sole missing part in our implementation. Consequently, we use their published rankings to measure different aggregation strategies.

To test drug aggregation strategies, we use each strategy to combine rankings of each drug for each covid type to produce a final top-100 ranking. We then compute the number of intersections with the drugs that are currently the subject of clinical trials related to COVID-19 [25]. This information is available on Kaggle as a list of drug names [14].

The results of the the different aggregation strategies can be found in Table 1. We see that our Union over Cluster Rankings with KMeans(k=3) outperforms the other approaches, yielding 32 hits. This is intuitive as using PCA on the prediction scores shows that there are three clusters among the COVID strains. In contrast, Dr-COVID's aggregation method, Union over Disease Rankings, reaches just 21 hits in our evaluation process.



DRAM load time (sec)			
33GB	66GB	131GB	261GB
13.5	26.3	83.6	203.5

Fig. 2. The total time for processing the two strategies (above): Global Score Mean and Global Score Maximum one after another. The time to copy data from NVMe SSD to DRAM as a function of DRAM size (below).

We observe that hits are not evenly distributed along the rankings of the aggregation strategies, with more hits towards rank 60 and higher, suggesting we are unlikely to get better results by predicting more than the top-100 drugs.

8.2 Scalability Results

Our implementation results can be seen in Figure 2. In our experiments, Optane-PM always outperformed DRAM. We note that DRAM performance gets significantly worse after consuming 192GB. This is a result of the dual cpu architecture: DRAM is split between the two sockets in a Non-uniform Memory Access (NUMA) architecture. This means that after 192GB, data must cross to the other socket which induces a latency penalty of around 100ns.

We also note that our experiments include the cost of loading data from disk (as needed). The loading time is non-trivial (see the table in Figure 2). One advantage of Optane-PM is that data is persistent and has no loading time. To measure the cost of compute only, we also tracked the running time after data was loaded. In this case, using Optane-PM is between two and three times slower than DRAM, which is expected since the latency of Optane-PM is known to be approximately three times that of DRAM. We note that the slower latency of Optane-PM is well worth the trade-off for higher capacity as well as persistence.

Surprisingly, the best-case memory mapping implementation performed almost as well as the Optane-PM implementation. This is an artifact of our drug rank aggregation strategies. The two strategies we evaluate are single scan operations, which behaves efficiently using memory mapping. As the memory size increases, we observe the same performance degradation as DRAM due to crossing NUMA node zones.

In a more realistic setting, memory mapping performs the worst. This is due to the eviction policy and DRAM not being able to store the entire dataset. We note that this is also a best-case realistic scenario as once evicted, a row will never be needed again by our strategies. For more advanced strategies, memory mapping will perform significantly worse as multiple passes (sometimes random access) of the data is required.

9 Conclusion

Deep learning can help the development of drugs in the face of a global pandemic. Rather than looking for promising candidates by hand, one can instead rely on graph neural networks. We have been able to clarify the evaluation part of Dr-COVID [4] and proposed an aggregation technique yielding better results. Our own implementation improves both training speed as well as readability. We have also shown that using Optane-PM allows researchers to scale techniques efficiently to large datasets, which benefits the drug repurposing community.

References

1. Ted T. Ashburn and Karl B. Thor. Drug repositioning: identifying and developing new uses for existing drugs. *Nature Reviews Drug Discovery*, 2004.
2. Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NeurIPS)*, pages 2787–2795, 2013.
3. Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.
4. Siddhant Doshi and Sundeep Prabhakar Chepuri. Dr-COVID: Graph neural networks for SARS-CoV-2 drug repurposing. *CoRR*, 2020.
5. Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. SIGN: Scalable inception graph neural networks. *CoRR*, 2020.
6. Deisy Morselli Gysi, Ítalo Do Valle, Marinka Zitnik, Asher Ameli, Xiao Gan, Onur Varol, Helia Sanchez, Rebecca Marlene Baron, Dina Ghiassian, Joseph Loscalzo, and Albert-László Barabási. Network medicine framework for identifying drug repurposing opportunities for COVID-19. *CoRR*, 2020.
7. Vassilis N. Ioannidis, Xiang Song, Saurav Manchanda, Mufei Li, Xiaoqin Pan, Da Zheng, Xia Ning, Xiangxiang Zeng, and George Karypis. DRKG - drug repurposing knowledge graph for covid-19. https://github.com/gnn4dr/DRKG/, 2020.
8. Joseph Izraelevitz, Jian Yang, Lu Zhang, Juno Kim, Xiao Liu, Amirsaman Memaripour, Yun Joon Soh, Zixuan Wang, Yi Xu, Subramanya R. Dulloor, Jishen Zhao, and Steven Swanson. Basic performance measurements of the intel optane DC persistent memory module. *CoRR*, abs/1903.05714, 2019. URL: http://arxiv.org/abs/1903.05714, http://arxiv.org/abs/1903.05714 `arXiv:1903.05714`.
9. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.
10. Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
11. Otto Kißig, Martin Taraz, Sarel Cohen, Vanja Doskoč, and Tobias Friedrich. Drug repurposing for multiple covid strains using collaborative filtering. In *ICLR Workshop on Machine Learning for Preventing and Combating Pandemics (MLPCP@ICLR)*, 2021.
12. Otto Kißig, Martin Taraz, Sarel Cohen, and Tobias Friedrich. Drug repurposing using link prediction on knowledge graphs. In *ICML Workshop on Computational Biology (CompBio@ICML)*, 2021.
13. Maria Nicola, Zaid Alsafi, Catrin Sohrabi, Ahmed Kerwan, Ahmed Al-Jabir, Christos Iosifidis, Maliha Agha, and Riaz Agha. The socio-economic implications of the coronavirus pandemic (COVID-19): A review. *International Journal of Surgery*, 2020.
14. Parul Pandey. Covid19 clinical trials dataset. https://www.kaggle.com/parulpandey/covid19-clinical-trials-dataset, 2021. Retrieved February 19th, 2021.
15. Karl Pearson, F.R.S. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 1901.

16. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
17. Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *International Conference on World Wide Web*, 2001.
18. Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.
19. Bhumi Shah, Palmi Modi, and Sneha R. Sagar. In silico studies on therapeutic agents for COVID-19: Drug repurposing approach. *Life Sciences*, 2020.
20. Daniel Waddington, Clem Dickey, Moshik Hershcovitch, and Sangeetha Seshadri. An architecture for memory centric active storage (mcas). *arXiv preprint arXiv:2103.00007*, 2021.
21. Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.
22. David S. Wishart, Yannick D. Feunang, An Chi Guo, Elvis J. Lo, Ana Marcu, Jason R. Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, Nazanin Assempour, Ithayavani Iynkkaran, Yifeng Liu, Adam Maciejewski, Nicola Gale, Alex Wilson, Lucy Chin, Ryan Cummings, Diana Le, Allison Pon, Craig Knox, and Michael Wilson. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic Acids Res.*, 2018.
23. Andrew Wood, Moshik Hershcovitch, Daniel Waddington, Sarel Cohen, and Peter Chin. Non-volatile memory accelerated posterior. *IEEE High Performance Extreme Computing Conference*, 2021.
24. Andrew Wood, Moshik Hershcovitch, Daniel Waddington, Sarel Cohen, Meredith Wolf, Hongjun Suh, Weiyu Zong, and Peter Chin. Non-volatile memory accelerated geometric multi-scale resolution analysis. *IEEE High Performance Extreme Computing Conference*, 2021.
25. World Health Organization. International clinical trials registry platform (ictrp). https://www.who.int/clinical-trials-registry-platform, 2021. Online; accessed February 19th, 2021.
26. Cheng Ye, Rowan Swiers, Stephen Bonner, and Ian Barrett. Predicting potential drug targets using tensor factorisation and knowledge graph embeddings, 2021. http://arxiv.org/abs/2105.10578 `arXiv:2105.10578`.
27. Da Zheng, Xiang Song, Chao Ma, Zeyuan Tan, Zihao Ye, Jin Dong, Hao Xiong, Zheng Zhang, and George Karypis. DGL-KE: training knowledge graph embeddings at scale. In *SIGIR Conference on Research and Development in Information Retrieval*, 2020.
28. Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinform.*, 2018.