Histogram-Based Gradient Boosting Trees for Efficient Graph Learning with Wasserstein Embeddings

Stanford CS224W

Sharan Ramjee †
sramjee@stanford.edu

Michael Wornow †
mwornow@stanford.edu

Abstract

Molecular property prediction is a task faced in a variety of high-impact medical and biochemical fields, from drug discovery to the development of diagnostic screening tools to the synthesis of biologics. Though molecular property prediction has traditionally been a difficult task for machine learning models, recent advances in graph-based techniques have yielded significant performance increases. This paper contributes to this growing field at the intersection of biology, chemistry, and deep learning by offering an improved pipeline for predicting whether a given molecule is able to inhibit the replication of HIV. We build on previous work that utilizes Wasserstein graph embeddings and optimize an end-to-end pipeline to achieve near state-of-the-art performance on the Open Graph Benchmark's `ogbg-molhiv` dataset using a fraction of the model parameters as the current best model. The source code for this paper is available on GitHub: `https://github.com/sharanramjee/wegl-gbt`

1 Introduction

Machine learning (ML) algorithms have led to breakthrough performance in a number of different fields, from image recognition to natural language processing to speech recognition to drug development to self-driving cars [1]. These traditional ML techniques, however, can be inefficient at learning from graphical data, and thus significant improvement has been achieved on a variety of graph-structured datasets by leveraging graph-specific ML techniques like graph neural networks (GNNs) and kernel-based methods [2, 3, 4].

One such domain is the field of chemical biology. Machine learning on graphs can be leveraged for many biological and chemical applications, including drug development, clinical decision making, and personalized medicine [5, 6, 3]. As opposed to wet-lab methods that require bench work from lab scientists, *in silico* methods like machine learning can help accelerate the development of biologics and therapeutics by reducing the time, equipment, and monetary requirements of screening molecules.

Molecules can be modeled as graphs, where nodes represent atoms and edges represent the bonds between those atoms [7]. Each atom's attributes (e.g. atomic number, charge, etc.) can be represented as a node feature vector, while each bond's attributes can similarly be captured in an associated edge feature vector. The application of advanced graph ML techniques to molecular datasets thus represents a promising and growing area of active research within the GNN community [8].

Though graph ML techniques have led to superior performance on many sets of graphical data, they unfortunately suffer from scalability issues as the size of the input graph increases [9, 10]. In order to address this issue, Kolouri *et al.*proposed a method entitled Linear Wasserstein Embedding for Graph Learning (WEGL) [11].

As primarily a method for generating distance metrics between graph embeddings, however, the final performance of a Linear WEGL model is highly dependent on both the data pre-processing

† Department of Computer Science, Stanford University

and post-WEGL classification steps that are taken. Thus, in this paper we sought to improve the end-to-end performance of the model described in Kolouri *et al.*on the Open Graph Benchmark (OGB) `ogbg-molhiv` dataset [12], in which the task is to predict whether a given molecule will have a desired chemical property (namely, inhibiting the replication of HIV).

Though `ogbg-molhiv` specifically considers the capability of a molecule to inhibit HIV replication, a generalizable method that proves effective at this task could potentially provide insight towards other tasks of therapeutic interest, and help to predict other biological properties based on a molecule's structure.

The architecture of our final pipeline is as follows: given a set of molecules represented as graphs, the molecules are run through Linear WEGL to generate their graph embeddings, PCA is applied on the embeddings to reduce their dimensionality to 20 dimensions, SMOTE is then applied to these embeddings to reduce the class imbalance between positive and negative examples, and, finally, this minority over-sampled dataset is fed through a tuned Histogram-Based Gradient Boosting Tree (HGBT) model which generates a binary prediction on the molecule.

The final average ROC-AUC attained by our model over ten runs is 0.8071 with a standard deviation between runs of 0.0040, which represents an ROC-AUC score gain of 3.14% over the original Linear WEGL model and places our model in second place on the current OGB Leaderboard for `ogbg-molhiv` (as of March 18, 2021).

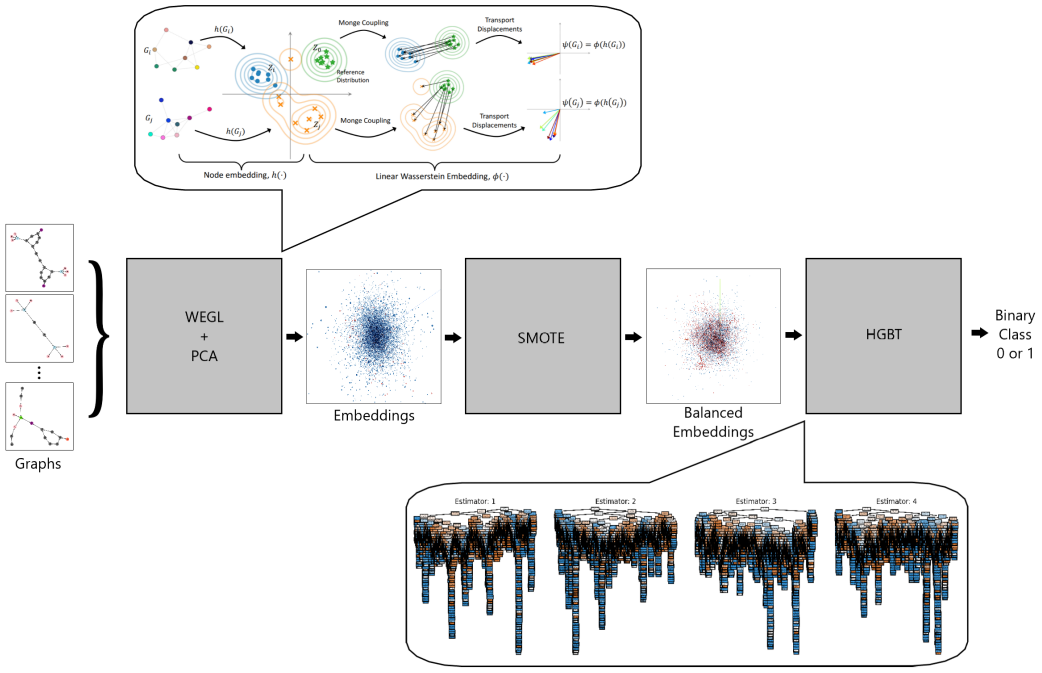

Figure 1: An overview of our HGBT-WEGL pipeline

An overview of our proposed WEGL+HGBT pipeline is given in Fig. 1. Here, molecules represented as graphs are fed into a Linear WEGL model, where the graphs are embedded into a lower dimensional space. The inset diagram of the WEGL process is taken directly from [11]. In the graph of embeddings, blue points represent negatively labeled examples while red points represent positive examples. Note the huge imbalance between blue and red points. Next, these embeddings are fed through SMOTE to over-sample the positive examples, yielding the chart of balanced embeddings with a 1:1 ratio of red to blue points. Finally, this balanced dataset is fed through a tuned HGBT model, which generates a set of decision trees as depicted in the inset diagram. This model then yields a binary prediction on whether or not the input molecule inhibits HIV replication.

2 Existing Method

Linear Wasserstein Embedding for Graph Learning (WEGL), as proposed in Kolouri *et al.*, is a method for efficiently calculating the Wasserstein distance between a set of graph embeddings which can then be utilized for downstream prediction tasks [11]. This method is currently ranked 10th on the OGB Benchmark Leaderboard for `ogbg-molhiv`.

At a high level, Linear WEGL works by embedding an input graph into a Hilbert space where the Euclidean distance between two graphs' embeddings approximates the 2-Wasserstein distance between those two graphs. Unlike previous Wasserstein Embedding-based approaches [9], Kolouri *et al.*'s use of the linear optimal transport (LOT) framework introduced in Wang *et al.*[13] allows them to (approximately) compute Wasserstein distances as a *linear*, rather than quadratic, function of the number of graphs in the dataset [11].

Kolouri *et al.*[11] denote μ_i as a probability measure defined on $Z \subseteq \mathbb{R}^d$, with a corresponding probability density function p_i. Using Brenier's theorem [14], Kolouri *et al.*[11] compute the 2-Wasserstein distance as:

$$\mathcal{W}_2(\mu_i, \mu_j) = (\inf_{f \in MP(\mu_i, \mu_j)} \int_Z ||z - f(z)||^2 d\mu_i(z))^{\frac{1}{2}}$$

where $MP(\mu_i, \mu_j) = \{f : Z \to Z' | f_{\#}\mu_i = \mu_j\}$ and $f_{\#}\mu_i$ represents the pushforward measure of μ_i and f represents a transport map [15].

Next, Kolouri *et al.*define μ_0 to be a reference probability measure with density function p_0 such that f_i is defined as the Monge map that pushes μ_0 into μ_i, thus yielding:

$$f_i = \operatorname*{argmin}_{f \in MP(\mu_0, \mu_i)} \int_Z ||z - f(z)||^2 d\mu_0(z)$$

Kolouri *et al.*[11] define an equidistant azimuthal projection $\phi(\cdot)$ such that $\phi(\mu_i) \triangleq (f_i - id)\sqrt{p_0}$, where $id(z)$ is defined as the identity function. The mapping $\phi(\cdot)$ has several key properties, namely that it provides an isometric embedding for probability measures where the reference is mapped to zero, the mapping preserves distances to μ_0, and the ℓ_2 distance between $\phi(\mu_i)$ and $\phi(\mu_j)$ approximates $\mathcal{W}_2(\mu_i, \mu_j)$. Since $\phi(\cdot)$ provides a linear embedding, it is known as the linear Wasserstein embedding.

WEGL combines node embedding methods with this linear Wasserstein embedding process [11]. More formally, Kolouri *et al.*[11] denote $\{G_i = (V_i, E_i)\}_{i=1}^M$ as a set of M individual graphs, where each node $v \in V_i$ has feature vector x_v and each edge $(u, v) \in E_i$ has feature vector w_{uv}. Here, $h(\cdot)$ is some node embedding process – the three considered in Kolouri *et al.*are **Concat**, where the final embedding is comprised of the concatenated embeddings from each layer of the embedding model, **Average**, where the final embedding is the average of the embeddings generated from each layer of the embedding model, and **Final**, which is simply the final embedding layer's output – where $h(G_i) = Z_i = [z_1, \ldots, z_{|V_i|}]^\top \in \mathbb{R}^{|V_i| \times d}$. A reference node embedding Z_0 is computed using the k-means clustering algorithm on the previously calculated node embeddings $\{Z_i\}_{i=1}^M$. For our model, we used the **Final** aggregation function.

Finally, Kolouri *et al.*[11] compute the linear Wasserstein embedding $\phi(Z_i)$ with respect to Z_0 using the node embeddings $\{Z_i\}_{i=1}^M$ and the reference node embedding Z_0. Thus, the overall embedding $\psi(G_i)$ for a graph G_i is given by $\psi(G_i) = \phi(h(G_i))$, where the function $\psi : \mathcal{G} \to \mathcal{H}$ maps any graph G in the space for all possible graphs $\mathcal{G}$ to an embedding $\psi(G)$ in a Hilbert space $\mathcal{H}$ as shown in the WEGL component diagram in Fig. 1.

3 Dataset

The `ogbg-molhiv` [12] dataset is taken from MoleculeNet [16], a large repository of molecular benchmarks for machine learning, and pre-processed with RDKit [17], a popular cheminformatics library. The `ogbg-molhiv` dataset contains 41,127 graphs, where each graph represents a molecule: the nodes in each graph correspond to the atoms of the molecule, while the edges correspond to

the bonds between atoms. The average number of nodes and edges per graph is 25.5 and 27.5, respectively, while the average node degree is 2.2, and thus the graphs in this dataset are smaller and more tree-like than some of the other more densely clustered datasets in the OGB collection [12].

In terms of features, every node is associated with a 9-dimensional vector that represents biochemical properties of interest such as the atomic number, formal charge, and chirality of that atom. Each edge is associated with a 3-dimensional vector of categorical features that describe the type, stereochemistry, and conjugation of that bond. The dataset comes pre-split in an 80-10-10 train-validation-test ratio, which we preserve for training, validating, and testing our model.

The predictive task is as follows: Given a graphical representation of a molecule, assign it a binary label corresponding to whether or not that molecule inhibits HIV replication. The accuracy of the model is evaluated using the ROC-AUC metric.

4 Methods

4.1 Dimensionality Reduction

The graph embeddings that were generated by our tuned Linear WEGL model are very high dimensional (300 dimensions) and as such, we followed the method described in the original Linear WEGL paper and applied dimensionality reduction before feeding these embeddings to our downstream classification model [11]. We tested three different dimensionality reduction techniques – PCA, ICA, and Sparse Projections – and saw that the best predictive performance was achieved with PCA, which was the same method used in Kolouri *et al.*.

Principal Components Analysis (PCA) One of the most commonly utilized unsupervised dimensionality reduction techniques, PCA was the method that resulted in our classification model achieving its best performance. PCA works by computing a set of n linearly independent "principal components" (where n is less than the dimensionality of the data) which maximize the amount of variance in the original dataset that linear combinations of those n components can explain [18].

Independent Components Analysis (ICA) In addition to PCA, we also considered ICA, which another commonly used unsupervised dimensionality reduction technique that is primarily used in signal processing applications for separating multivariate signals into independent non-gaussian additive signals [19]. More formally, ICA transforms the observed data $\boldsymbol{x} = (x_1, \ldots, x_m)^\top$, using a linear static transformation $\boldsymbol{W}$ as $\boldsymbol{s} = \boldsymbol{W}\boldsymbol{x}$, into a vector of maximally independent components $\boldsymbol{s} = (s_1, \ldots, s_n)^\top$ as measured by some function $F(s_1, \ldots, d_n)$ of independence (usually defined by minimization of mutual information). Unlike PCA, we found that this method did not result in any improvement in performance.

Sparse Random Projections (SRP) Finally, we also tried to project our embeddings into a lower dimensional space using sparse random matrix projections. A faster and less memory intensive alternative to Gaussian random projection matrices, SRP works by projecting its inputs onto a random matrix, as described in Li *et al.*[20]. While more conceptually distinct from ICA and PCA, we found that this method did not result in an improvement in performance.

4.2 Dataset Balancing

The `ogbg-molhiv` dataset is highly imbalanced – only 3.64% of the data belongs to the positive class. Thus, a model could achieve a seemingly impressive accuracy of 98.6% by simply always guessing that a molecule belongs to the negative class, defeating the entire purpose of training a predictive model.

This issue of class imbalance arises in most chemical classification tasks, in that the desired chemical properties (e.g. HIV inhibition) are typically rare [21]. In order to achieve both high sensitivity and specificity on a dataset like `ogbg-molhiv`, we thus looked towards sampling and augmentation methods that could reduce the bias induced by this class imbalance.

Synthetic Minority Oversampling Technique (SMOTE) After surveying the literature, we decided to use SMOTE, first described in Chawla *et al.*, on our dataset to increase the representation

of positive examples [22]. SMOTE has been shown to be effective at remedying class imbalance issues in molecular datasets seeking to predict bioactive properties of interest [23, 21]. When applied to the `ogbg-molhiv` dataset, SMOTE works by over-sampling the positive (minority) class and under-sampling the negative (majority) class. Unlike other methods of over-sampling, SMOTE does not simply repeatedly sample the same positive data points. Rather, it generates synthetic positive examples by interpolating new data points between the existing positive examples. Given the hyperparameter k, the SMOTE algorithm will loop through each positive example p and identify its k nearest positive neighbors. It then randomly selects n of the neighbors of p (where the hyperparameter n depends on how much over-sampling is required) and choose a random point along the line segment which connects that neighbor to p. This leads to larger, more generalizable decision regions being formed around each over-sampled positive example than if a model had been trained directly on the class imbalanced data [22].

We tested several variations on SMOTE, including Borderline SMOTE, SVM-SMOTE, K-Means SMOTE, and ADASYN. However, we found that our end model achieved its best performance with vanilla SMOTE.

Borderline Synthetic Minority Oversampling Technique (B-SMOTE) Originally described in Han *et al.*, B-SMOTE is a popular variant of SMOTE in which only minority class examples that lie near the decision boundary between classes are over-sampled [24]. While this procedure makes more assumptions about the structure of the underlying data, if done properly it can lead to better performance by (a) prioritizing learning on more informative examples that exist at the decision boundary and (b) ignoring the noise of minority class examples whose nearest neighbors are all part of the majority class, especially when most of the end model's prediction errors occur at these decision boundaries [24].

Support Vector Machine Synthetic Minority Oversampling Technique (SVM-SMOTE) SVM-SMOTE, proposed by Nguyen *et al.*, functions similarly to B-SMOTE except that it first applies a standard SVM classifier on the input dataset to get support vectors for the minority examples [25]. Neighborhoods of minority examples around each minority support vector are then identified, and new minority data points are generated through either extrapolation or interpolation along the lines connecting each support vector to the data points in its neighborhood [25].

Adaptive Synthetic Sampling (ADASYN) Developed by He *et al.*, ADASYN [26] follows a similar approach to the aforementioned variants of SMOTE in that it prioritizes over-sampling minority examples that occur in critical decision-making regions. However, ADASYN is more flexible in how it makes this assessment. Rather than a hard boundary like the one used by B-SMOTE, ADASYN weights each minority class example by how many majority class examples are in its neighborhood. The core idea is that the more majority examples surround a minority example, the harder that minority example (and others like it) will be to classify [26]. Thus, minority examples in these types of majority-dominated neighborhoods will be given a higher weighting, and thus over-sampled at a higher frequency than other minority examples.

K-Means Synthetic Minority Oversampling Technique (KM-SMOTE) K-Means SMOTE also aims to reduce noise in the over-sampled resulting dataset by following a three-step procedure first described in [27]. First, the data are clustered into k groups using the k-means algorithm. Next, the ratio of majority to minority class examples within each cluster is computed. The algorithm then proportions the total number of synthetic samples that it needs to generate across these clusters, giving clusters with a higher ratio of majority to minority examples a higher number of minority samples to generate. Finally, SMOTE is applied to each cluster in order to achieve the desired number of synthetic samples. The resulting performance can be highly dependent on the hyperparameter k, but when properly tuned the algorithm has been shown to perform better than SMOTE in certain cases [27].

4.3 Downstream Classifiers

Linear WEGL relies on the end user providing their own downstream classification model to leverage the generated embeddings for predictive tasks. Thus, we considered a variety of classification models to use for our model. We ended up using HGBT for our end model to achieve the highest ROC-AUC,

but we also tested a variety of alternatives including Random Forests, Gradient Boosted Trees, Multilayer Perceptron with ReLU and Dropout, and ADABoost.

Gradient Boosted Trees (GBT) Boosting is a type of ensemble machine learning algorithm in which each iteration of the learning procedure adds an additional component to the model being trained that corrects a prediction error made by the model in its prior state [28]. By learning many "weak" classifiers that lack strong predictive power but have uncorrelated biases, boosting enables one to construct a "strong" classifier by training and then combining these weak classifiers.

GBTs are one of the most popular implementations of such models. A GBT is a specific type of tree ensemble in which a differentiable loss function is specified, and then a decision tree which minimizes the model's overall loss on that function is added to the ensemble at each step of the learning process [28].

Histogram-Based Gradient Boosting Trees (HGBT) For continuous values like the node embeddings considered in this paper, however, classical GBT can be incredibly time consuming, as the model must consider all of a feature's distinct values across all data points before being able to generate a split in its decision trees [28]. By binning the continuous values that a feature can take on into a discrete set of chunks, the computational demands of building each decision tree during the boosting process is greatly reduced, thus allowing us to build larger and deeper models than would have otherwise been possible with vanilla gradient boosting. This procedure of binning features is known as the Histogram-Based Gradient Boosting Tree (HGBT) model [28].

Random Forests (RF) A random forest is one of the most basic ensemble-based learning methods for classification tasks. An RF model combines many different, independently trained decision trees into one central model, aka a "forest" of trees [29]. This combats the natural tendency of decision trees to overfit to their training data by creating an ensemble of trees in which the majority vote, rather than any one tree, decides the end model's output.

Balanced Random Forests (B-RF) As described in Chen *et al.*, B-RF provides a simple extension of the classical Random Forest technique that better addresses class imbalance in the training dataset through down-sampling of the majority class [30]. The B-RF algorithm proceeds as follows: instead of randomly sampling from the entire training dataset at each split in the decision tree as a classical RF algorithm would do, R-BF bootstraps the same number of samples from the minority and majority classes so that each node considers an equal representation of majority and minority class examples.

Dropouts meet Multiple Additive Regression Trees (DART) Developed by Rashmi *et al.*, the DART algorithm combines boosted regression trees with dropout in which a random subset of decision trees in the model are muted during each training iteration [31]. This helps to combat what is known as the "shrinkage" effect of traditional boosted tree models, where each successive tree has less and less influence on the overall model's predictive performance.

Multilayer Perceptron (MLP) Finally, we also tried to construct a multilayer perceptron classification model to serve a a classical deep learning baseline to compare to our graph-based machine learning model. We tested a variety of MLP architectures, including varying the number of layers from 1 to 10, using different activation functions (ReLU, tanh, sigmoid), including/not including dropout, and using different optimizers like SGD and ADAM. As expected, none of these models yielded performance that would qualify for the OGB Leaderboard.

5 Quantitative Analysis

For the pipeline, we used Kolouri *et al.*'s [11] implementation of Linear WEGL with extensive hyperparameter tuning that was was done using an exhaustive grid search [32]. However, we found that the default set of hyperparameters that were empirically determined by Kolouri *et al.*[11] for Linear WEGL worked best, and thus these were the sets of hyperparameters that were used for generating the graph embeddings for our models.

The graph embedding generation, dimensionality reduction, dataset balancing, and downstream classifier were all performed on a Nvidia Tesla K80 GPU. In order to level the playing field for

testing each component of our pipeline, we fix the methods used for the remaining components of each pipeline for a fair comparison. The evaluation of the dimensionality reduction methods is done with SMOTE and HGBT fixed. The evaluation of the dataset balancing methods is done with PCA and HGBT fixed across all the dataset balancing methods considered. Finally, the evaluation of the downstream classifiers is done with PCA and SMOTE fixed across all the downstream classifiers considered for the dimensionality reduction and dataset balancing components of the pipeline, respectively.

The performances of the dimensionality reduction, dataset balancing, and downstream classifiers are compared amongst each other as a measure of the ROC-AUC score of the downstream classifier on the `ogbg-molhiv` test set. This is evaluated using an OGB model evaluator, where the average and unbiased standard deviation of the test set ROC-AUC is reported across 10 runs with random seeds, which were not fixed.

5.1 Dimensionality Reduction

The implementations of all the dataset balancing methods considered are given in the `scikit-learn` library [33]. Dimensionality reduction is applied here for three reasons - to reduce the computational burden of computing the reference node embedding using k-means clustering, to reduce the computation burden of training the downstream classifier, and to reduce overfitting. The quantitative evaluation of the dimensionality reduction methods, as detailed in Section 4.1, is given in Table 1.

Table 1: Quantitative evaluation of dimensionality reduction methods

Method	Test ROC-AUC	Valid ROC-AUC
None	0.7837 ± 0.0056	0.8106 ± 0.0076
ICA	0.7893 ± 0.0068	0.7658 ± 0.0093
SRP	0.7955 ± 0.0034	0.7798 ± 0.0072
PCA	$\mathbf{0.8071 \pm 0.0040}$	0.8226 ± 0.0068

We observed that PCA, as proposed in the original Linear WEGL paper [11], gave us the best end model performance. It is admittedly a bit opaque to us as to why our graph embeddings were more amenable to PCA than the alternative dimensionality reduction methods considered. What is clear, however, is that reducing the dimensionality of the graph embeddings before applying a classification model resulted in a substantial improvement in performance, likely due to a reduction in overfitting.

5.2 Dataset Balancing

The implementations of all the dataset balancing methods considered are given in the `imbalanced-learn` library [34]. Furthermore, given that the main issue with the `ogbg-molhiv` dataset is that it consists of extremely fewer instances of one class over the other, a minority oversampling strategy was used across all the methods in order to equalize the number of instances of each class in the train set of the dataset. The quantitative evaluation of the dataset balancing methods, as detailed in Section 4.2, is given in Table 2.

Table 2: Quantitative evaluation of dataset balancing methods

Method	Test ROC-AUC	Valid ROC-AUC
KM-SMOTE	0.7382 ± 0.0066	0.8316 ± 0.0095
SVM-SMOTE	0.7704 ± 0.0039	0.8917 ± 0.0043
None	0.7751 ± 0.0080	0.8156 ± 0.0102
B-SMOTE	0.7845 ± 0.0049	0.9004 ± 0.0033
ADASYN	0.7986 ± 0.0084	0.8155 ± 0.0063
SMOTE	$\mathbf{0.8071 \pm 0.0040}$	0.8226 ± 0.0068

Here, we did not consider using a random sampler that over-samples by merely duplicating the original samples from the minority class. This is because it would not bring diversity to the training

set and as such, would not result in a worthwhile improvement in generalizability for improved performance on the test set. Instead, we focused on SMOTE, ADASYN, and its variants as they generate new samples in by interpolation. Furthermore, we notice that the performance of ADASYN and the variants of SMOTE are inferior to that of SMOTE. The inferior performance of ADASYN can be attributed to the fact that ADASYN focuses on generating samples next to the original samples, which, when are closely clustered together in the latent space similar to the scenario with the `ogbg-molhiv`, lead to wrong classifications when clustering using the k-Nearest Neighbors classifier during interpolation as examined by Kurniawati *et al.*[35]. The inferior performance of the variants of SMOTE, while could be artefacts of their unique clustering functions similar to the case of ADASYN, can be attributed to the fact that they do not make any distinction between easy and hard samples to be classified using their clustering rules. i.e. the samples interpolated for the minority class are mixed in with those of the majority class, thus resulting in an inferior performance on the downstream classification task as detailed by Fernández *et al.*[36].

5.3 Downstream Classifiers

The implementations of all the downstream classifiers considered can be found in either the `scikit-learn` [33] or `XGBoost` [37] libraries. The quantitative evaluation of the downstream classifiers, as detailed in Section 4.3, is given in Table 3.

Table 3: Quantitative evaluation of downstream classifiers

Method	Test ROC-AUC	Valid ROC-AUC
MLP	0.7379 ± 0.0355	0.8118 ± 0.0085
RF	0.7695 ± 0.0151	0.7908 ± 0.0252
B-RF	0.7731 ± 0.0097	0.7782 ± 0.0166
DART	0.7610 ± 0.0042	0.8370 ± 0.0034
GBT	0.7586 ± 0.0048	0.8440 ± 0.0083
HGBT	$\mathbf{0.8071 \pm 0.0040}$	0.8226 ± 0.0068

The performance of the MLP model, despite extensive hyperparameter tuning and cost-sensitive learning [38] for taking into account the class imbalance in the dataset, is inferior to those of any of the tree-based ensemble models considered. This can be attributed the fact that while the dataset has been balanced, SMOTE merely creates synthetic samples by interpolating the samples from the minority class, which can still lead to an inferior generalizable performance if these samples from the minority class are not diverse. This is where decision trees have an advantage as they compute conditions at each stage of splitting while taking both classes into consideration, which allows them to have a superior performance on imbalanced datasets. In particular, this ability of the models to generalize better to imbalanced datasets is greater in the case of gradient boosted ensemble based decision trees such as the HGBT model, where gradient boosting, while slow, allows the trees in the ensemble to be linked and take decisions sequentially. This improves the feature splits created in subsequent trees since they take the decisions made by the previous trees in the ensemble into account, thus, allowing the HGBT model to outperform the other downstream classifiers considered.

5.4 State-of-the-Art Graph Learning Methods

As of March 18, 2021, our pipeline ranks 2nd on the `ogbg-molhiv` leaderboard among the other graph learning methods, as shown in Table 4. In particular, we see that our WEGL+HGBT pipeline outperforms the WEGL baseline on the test set ROC-AUC metric by approximately 3.14, with a significant reduction in the number of parameters in the model. Here, WEGL comprises of 210,212 parameters while the HGBT comprises of 101,002 parameters, leading to a total of 311,214 trainable parameters for the pipeline. This reduction in the number of trainable parameters in a HGBT model over a Random Forest model, which was the downstream classifier used in the original WEGL baseline, can be attributed to the discretization achieved by HGBTs. The construction of decision trees in the ensemble is significantly sped up by binning continuous input features into a fixed number of buckets, thus leading to a reduction in the number of unique values for each of the features, which in turn leads to a reduction in the number of trainable parameters in the HGBT.

Table 4: `ogbg-molhiv` Leaderboard (as of March 18, 2021)

Rank	Method	Test ROC-AUC	Validation ROC-AUC	# of Params
1	Neural FingerPrints	0.8232 ± 0.0047	0.8331 ± 0.0054	2,425,102
2	**WEGL + HGBT (Ours)**	**0.8071 ± 0.0040**	**0.8226 ± 0.0068**	**311,214**
2	MorganFP+Rand. Forest	0.8060 ± 0.0010	0.8420 ± 0.0030	230,000
3	DGN	0.7970 ± 0.0097	0.8470 ± 0.0047	114,065
4	DeeperGCN+FLAG	0.7942 ± 0.0120	0.8425 ± 0.0061	531,976
5	PNA	0.7905 ± 0.0132	0.8519 ± 0.0099	326,081
6	GCN+GraphNorm	0.7883 ± 0.0100	0.7904 ± 0.0115	526,201
7	HIMP	0.7880 ± 0.0082	Not Reported	153,029
8	DeeperGCN	0.7858 ± 0.0117	0.8427 ± 0.0063	531,976
9	GSN	0.7799 ± 0.0100	0.8658 ± 0.0084	3,338,701
10	WEGL	0.7757 ± 0.0111	0.8101 ± 0.0097	361,064
11	GIN+virtual node+FLAG	0.7748 ± 0.0096	0.8438 ± 0.0128	3,336,306
12	GIN+virtual node	0.7707 ± 0.0149	0.8479 ± 0.0068	3,336,306
13	GCN+FLAG	0.7683 ± 0.0102	0.8176 ± 0.0087	527,701
14	GIN+FLAG	0.7654 ± 0.0114	0.8225 ± 0.0155	1,885,206
15	GCN	0.7606 ± 0.0097	0.8204 ± 0.0141	527,701
16	GCN+virtual node	0.7599 ± 0.0119	0.8384 ± 0.0091	1,978,801
17	GIN	0.7558 ± 0.0140	0.8232 ± 0.0090	1,885,206

6 Qualitative Analysis

In addition to our quantitative experiments, we wanted to gain a better qualitative understanding of the types of correct and incorrect predictions that are being made by our model. We randomly sampled two examples each from the testing set of true positives (Fig. 2), true negatives (Fig. 3), false positives (Fig. 4), and false negatives (Fig. 5) and have displayed these molecules in their respective Figures. Each node in the graphs in these figures is a color-coded atom labeled with its atomic symbol, while the edges represent bonds between atoms (please note that all bonds in these molecules are visually depicted with the same type of edge, regardless of the actual bond type).

As seen in Fig. 2 depicting true positives, our model is able to effectively embed complex molecules that have the desired property of interest. These molecules all share many PN_3 groups, which could be a feature that our model is picking up. This is further corroborated by Fig. 4, which shows examples of false positives. These false positives also have several PN_3 groups, a feature that is not seen in any of the true/false negatives that we sampled. Thus, it appears that our model is identifying PN_3 as a feature that is highly correlated with the ability to inhibit HIV replication.

The true/false negatives predicted by our model in Fig. 3 and Fig. 5 seem to have additional molecular motifs not found in the true/false positives due to the presence of atoms that are not N, P, B, or C. Thus, in addition to the structure itself of the molecule, this result indicates that our model is also leveraging node features to distinguish between positive and negative examples.

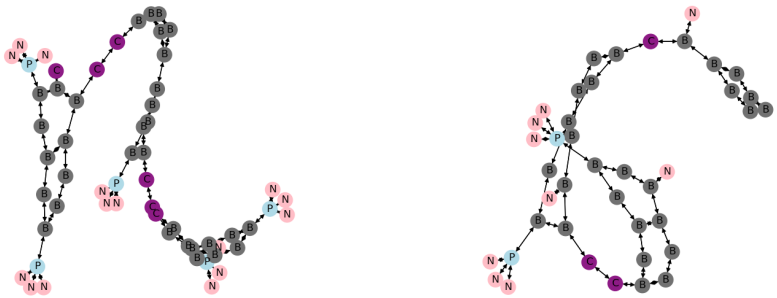

Figure 2: Two randomly selected True Positives predicted by our model, i.e. molecules that are correctly predicted to inhibit HIV replication.

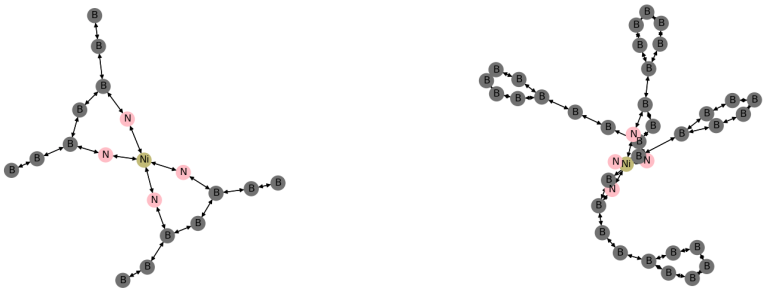

Figure 3: Two randomly selected True Negatives predicted by our model, i.e. molecules that are correctly predicted to not inhibit HIV replication.

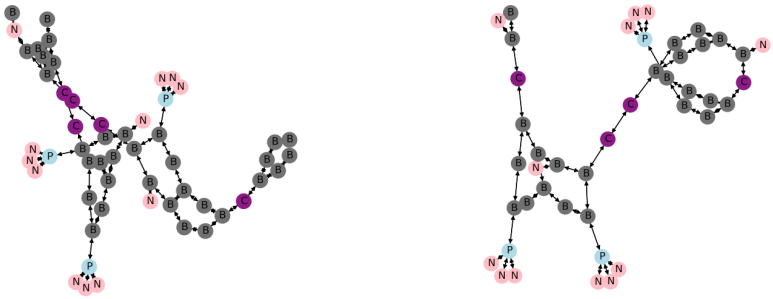

Figure 4: Two randomly selected False Positives predicted by our model, i.e. molecules that are predicted to inhibit HIV replication but actually lack that ability.

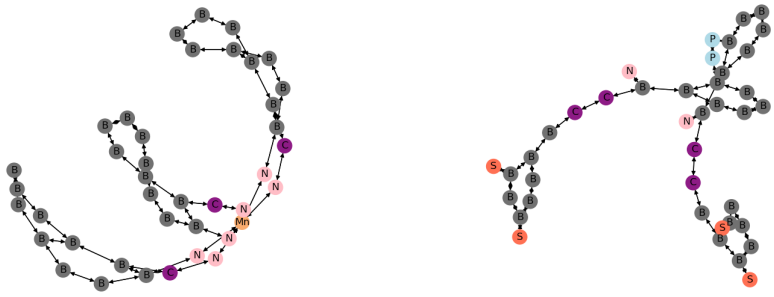

Figure 5: Two randomly selected False Negatives predicted by our model, i.e. molecules that inhibit HIV replication but are predicted to lack that ability.

7 Conclusion

We were able to achieve near state-of-the-art performance on the `ogbg-molhiv` dataset by building an end-to-end prediction pipeline that utilized graph embeddings derived from the computationally efficient Linear WEGL procedure described in Kolouri *et al.*[11]. Compared to the model described in the original Kolouri *et al.*paper, our model is able to achieve an average higher performance of 3.14 ROC-AUC points (0.8071 v. 0.7757) on predicting the HIV inhibitory characteristics of a molecule based on its chemical structure. The further development of *in sitro* methods like ours which leverage the power of both machine learning and graphical data to predict chemical properties of interest can help speed up the screening of molecules and development of small molecule therapeutics for disease.

References

[1] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and SS Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018.

[2] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 635–644, 2011.

[3] Wei Zhang, Jeremy Chien, Jeongsik Yong, and Rui Kuang. Network-based machine learning and graph theory algorithms for precision oncology. *NPJ precision oncology*, 1(1):1–15, 2017.

[4] Xin Li and Hsinchun Chen. Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems*, 54(2):880–890, 2013.

[5] Monica Agrawal, Marinka Zitnik, Jure Leskovec, et al. Large-scale analysis of disease pathways in the human interactome. In *PSB*, pages 111–122. World Scientific, 2018.

[6] Camilo Ruiz, Marinka Zitnik, and Jure Leskovec. Identification of disease treatment mechanisms through the multiscale interactome. *bioRxiv*, 2020.

[7] Hehuan Ma, Yu Rong, Wenbing Huang, Tingyang Xu, Weiyang Xie, Geyan Ye, and Junzhou Huang. Dual message passing neural network for molecular property prediction. *arXiv preprint arXiv:2005.13607*, 2020.

[8] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 2020.

[9] Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. *arXiv preprint arXiv:1906.01277*, 2019.

[10] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Martin Blais, Amol Kapoor, Michal Lukasik, and Stephan Günnemann. Is pagerank all you need for scalable graph neural networks? In *ACM KDD, MLG Workshop*, 2019.

[11] Soheil Kolouri, Navid Naderializadeh, Gustavo K. Rohde, and Heiko Hoffmann. Wasserstein embedding for graph learning. In *International Conference on Learning Representations*, 2021.

[12] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[13] Wei Wang, Dejan Slepčev, Saurav Basu, John A Ozolek, and Gustavo K Rohde. A linear optimal transportation framework for quantifying and visualizing variations in sets of images. *International journal of computer vision*, 101(2):254–269, 2013.

[14] Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

[15] Soheil Kolouri, Se Rim Park, Matthew Thorpe, Dejan Slepcev, and Gustavo K Rohde. Optimal mass transport: Signal processing and machine-learning applications. *IEEE signal processing magazine*, 34(4):43–59, 2017.

[16] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[17] Greg Landrum et al. Rdkit: Open-source cheminformatics. 2006.

[18] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[19] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[20] Ping Li, Trevor J Hastie, and Kenneth W Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296, 2006.

[21] Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Boosted multi-task learning. *Machine learning*, 85(1):149–173, 2011.

[22] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[23] Chetna Kumari, Muhammad Abulaish, and Naidu Subbarao. Using smote to deal with class-imbalance problem in bioactivity data to predict mtor inhibitors. *SN Computer Science*, 1:1–7, 2020.

[24] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[25] Hien M Nguyen, Eric W Cooper, and Katsuari Kamei. Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 3(1):4–21, 2011.

[26] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE, 2008.

[27] Georgios Douzas, Fernando Bacao, and Felix Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465:1–20, 2018.

[28] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[29] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[30] Chao Chen, Andy Liaw, Leo Breiman, et al. Using random forest to learn imbalanced data. *University of California, Berkeley*, 110(1-12):24, 2004.

[31] Rashmi Korlakai Vinayak and Ran Gilad-Bachrach. Dart: Dropouts meet multiple additive regression trees. In *Artificial Intelligence and Statistics*, pages 489–497. PMLR, 2015.

[32] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *25th annual conference on neural information processing systems (NIPS 2011)*, volume 24. Neural Information Processing Systems Foundation, 2011.

[33] 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.

[34] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[35] Yulia Ery Kurniawati, Adhistya Erna Permanasari, and Silmi Fauziati. Adaptive synthetic-nominal (adasyn-n) and adaptive synthetic-knn (adasyn-knn) for multiclass imbalance learning on laboratory test data. In *2018 4th International Conference on Science and Technology (ICST)*, pages 1–6. IEEE, 2018.

[36] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, 61:863–905, 2018.

[37] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[38] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.