Praktikum 1 - report

Self Distillation on Graph Neural Networks for Molecule Property Prediction

Author

Johannes Pascal Urban, B.Sc.

Wien, 2023

Study id:	UA 066 921
Field of study:	Master Computer Science - Data Science
Supervisor:	Assoz. Prof. Dipl.-Inf. Dr. Nils Morten Kriege

Contents

1 Introduction **4**

2 Preliminaries **5**

2.1 Graph Neural Networks 5

2.1.1 General preliminaries to Graph Neural Networks 5

2.1.2 Levels of Graph Neural Network prediction 6

2.1.3 Embedding 7

2.1.4 Dropout 8

2.1.5 Graph Isomorphism Network Convolution 8

2.1.6 Early Stopping 9

2.2 Random Forests 9

2.3 Dataset 10

2.4 Self Distillation 10

2.4.1 Motivation 10

2.4.2 Methods and Definitions 11

3 Methods **13**

3.1 Metrics 13

3.2 GNN choice and setup 15

3.2.1 Embeddings 16

3.2.2 Dropout 16

3.2.3 GINConv Layers 16

3.2.4 Early Stopping 17

3.2.5 Dataset Reduction 17

3.3 Student and Teacher Model 18

3.4 Choice of Self Distillation Items to Add 18

4 Experimental Evaluations **19**

4.1 General 19

4.2 Main Theories 20

4.3 Main Theory: Random Forest 21

4.4 Experiment subset selection 22

4.5 General teacher theory 23

4.6 Side theory: Stochastic Depth in teacher - Concatenation of Layers 24

4.7 Digression: Stochastic depth 24

4.8 Side theory: Initial setup 25

4.9 Unstable Student - Fixing self distillation 26

4.10 Side theory: Self Distillation selection method 28

4.11 Side theory: Label type 30

4.12 Side theory: Class balancing 33

4.13 Side theory: Dropout check 34

4.14 Side theory: Self Distillation set size 35

4.15 Side theory: Enlarge student model 36

4.16 Side theory: Overfitting 38

4.17 Side theory: Dataset completeness 40
4.18 Side theory: Dataset rebuilding 41
4.19 Extra: Advanced accuracy measuring 43
4.20 Extra: Sanity check 43
4.21 Extra: Data replacement 44
4.22 Main theory: Self distillation on graph neural network 46

5 Conclusion **47**
5.1 Final Analysis 47
5.2 Concluding Remarks 48
5.3 Outlook 48

A Experimental results on GitHub **54**
A.1 GitHub repository 54
A.2 Environment setup 54
A.3 Dataset fetching 54
A.4 Notebooks and Result list (sorted by notebook occurrence on GitHub) 55
A.5 Toolbox documentation 57

Abstract

In this project the effect of self distillation was examined on random forests and graph neural networks in respect to prediction performance. This technique is defined as having one model creating artificial labels for unlabeled data which is then input into another model.
Tests showed a significant drop of prediction performance in random forests, which confirms the initial theory of self distillation not being beneficial for random forests using chemical descriptor and morgan fingerprint representations of the molecules as data input.
Based on the successful applications of self distillation on regular neural networks in other papers, the theory that this technique is beneficial on graph neural networks is to be evaluated experimentally. Throughout numerous experiments concerning different settings, methods and concepts it has been shown that self distillation is not applicable on this projects model and data setting as the metric scores were equal or below the reference scores using a 5 layered graph isomorphism network graph neural network. However as a positive effect of this method a slight increase of training stability was observed.
These results in combination with further experiments concerning the benefit of increasing the training set itself, strongly indicate that adding training data is not able to improve the model performance and may thus be a viable cause for the inapplicability of this method in this project. Another possibility is that the used model is too shallow as reference models possessed significantly more layers.
In conclusion it was not possible to prove or disprove a positive effect of self distillation on graph neural networks as only one model was tested extensively.

1 Introduction

In the 2020 and 2021 published papers [16] and [10] a technique called self distillation is being introduced and used to enhance the prediction performance of neural networks. What this technique is doing is creating a model referred to as the 'teacher model' which learns using training data and labeling other elements. These previously unlabeled points in combination with the original training data are used for training another model referred to as the 'student model'. By doing so in combination with noise these publications manage to increase the performance of the original neural network.
The idea of this project is to take this approach and find out whether this approach is also able to enhance the results when using a graph neural network instead of a regular neural network. For this an already established and known to be well performing model is chosen from the Open Graph Benchmark (OGB) top list.
This model will be used in combination with a chemical dataset containing the activity measurements of a molecule in an experimental setting. The molecule will be transformed to its graph representation and trained using graph level prediction in combination with a weighted binary cross entropy loss and the

Adam optimizer.
The goal of this project is to ascertain whether self distillation works for a simpler machine learning model - a Random Forest - and whether it works for a more complicated machine learning model - a graph neural network. Determining whether it works for a random forest is simpler than determining it for a Graph Neural Network (GNN). This is because in the neural network there are much more settings and methods that can be changed in comparison to the Random Forest machine learning classifier where the basic parameters and settings are fixed.
Hence to prove or disprove the main theory that self distillation is working for GNNs, there will be a multitude of smaller theories that aim to predict whether modifications will make self distillation possible. In the end the side theories will be used to make a reasonable declaration on the validity of the main theory by covering multiple variations and modifications in these theories.

2 Preliminaries

Before starting with the theoretical and practical part of this report, for a common understanding some terms and subjects need to be introduced which will be done in the following sections:

2.1 Graph Neural Networks

2.1.1 General preliminaries to Graph Neural Networks

In general a GNN is a neural neural network type learning method that works with graph input instead of array (data) input. A normal neural network would expect an array, matrix or higher dimensional space as input using it to feed into neurons and calculating an output for the neuron. The learning would take place by calculating the error (using a loss function) between the actual and calculated value to make the output of the neurons be close to the expected behavior over time.[11]
The difference for a graph neural network is that instead of a linear or block shaped input we have a graph as an input which is represented by nodes and edges. Technically the difference is not fundamental as nodes and edges would have data on their own, just like the input of a normal neural network. Hence the main difference is that there is more knowledge stored in a graph as data itself may be connected to other data which could prove to be beneficial.
Due to the fact that we have multiple sets of 'normal' data, we can utilize 'normal' neural network operations on these chunks of data; hence the only issue is to connect this blocks of data to each other to exploit the information stored through these connections. What is used for this is what is called a 'Convolution Layer/Network'. This is a component/technique that sets node data in connection so that the updating step of the parameters can take other nodes into account. There are roughly two types of convolution types: spectral and spatial.

- spectral: This method is inspired by the Fourier Transform and uses the spectral representation of the graph meaning it decomposes the graph into subparts like eigenvectors using the adjacency matrix of the graph. An example method that is relatively well known is GCN (Graph Convolution Network)
- spatial: Spatial approaches define convolution is a more direct way, utilizing the graph structure to do so. This means that the neighboring nodes are used to update the node embedding. An example for this is GINConv which is more explained in section 2.1.5 and GraphSAGE which works similarly by sampling and aggregating features from neighboring networks.

After subjecting the input graph to one or more convolution layers the question remains on how to get the graph shaped data into an actual prediction. This transformation depends on the type of problem that we aim to predict - whether the training labels are of nodes, edges or whole graphs as described in the next section (2.1.2). For node level prediction one would simply take the data from each node and decrease its dimension using (graph) neural network modules until the dimension of the training label is met. For edge level prediction it is a bit more complicated as the network did not particularly train the edge data but the node data. Instead of what can be done is by combining the node data for two nodes (which are adjacent to an edge) and transform this to the label dimension and learning this network setting.
What is used in this project is a bit more complicated as it does not solely require one or two node data arrays, but multiple of them. The method used to do this is called 'pooling' and in its simplest form means that the values of several nodes (involved in the graph to predict/learn) are aggregated using maximum, mean, sum or attention operations - in this project a sum pooling will be used. [20, p. 57-66]

2.1.2 Levels of Graph Neural Network prediction

There are three main types of prediction level tasks with graph neural networks: The first and most basic version is having a graph and attempting to predict the label(s) of a specific node according to the connections to other nodes. The second type would relate to predicting the label or the existence of an edge in a graph. The last technique is the supertype of the previous ones in a sense as here not a single node or edge is predicted but a whole graph.[19]
A real world example for a graph type problem would most likely include social relations like a user connectivity graph of e.g. Facebook or a similar service. In this problem space node level prediction could be predicting the age of a person according to their connections to other users. Edge level prediction could include which other users the current user might be interested in, in a manner of recommendations for the user. Finding an example for the third type in this context becomes a little tricky as the input of the neural network is a graph itself and the target for the label is the whole graph in this type.

One possible mapping to an example could be a case where a subgroup of nodes (and the graph these nodes span) to learn specific groups of users. This could be used in trying to identify groups among users, the background setting and usage of this example however is quite vague.
In this projects setting graph level prediction will be applied in the context of chemistry. This subject is ideal for this setting as naturally in chemistry it is always certain to encounter molecules which in essence can be easily represented as graphs: atoms are nodes, bonds are edges and the whole graph is the molecule. However the data and the concrete problem addressed in this paper (and with the GNN prediction) will be more elaborated in section 2.3.

2.1.3 Embedding

Embeddings in neural networks function as representatives of data. If one takes for example node embeddings, then they would represent the state of the node i.e. containing data that has been learned. There are several applications for this, one could for example generate embeddings of a whole network that aim to represent the network in a lower dimensional way, it can be used for node and edge embeddings to store/learn for the nodes or edges for a network or it can be used in general as a representative of a specific feature.[4]
In this project it is used in three meanings:

- Node embedding: When learning on a whole graph learning requires each node to have a data representation of the node that can be used to modify the node embedding of one node using the embeddings of the neighboring nodes
- Edge embedding: Additionally to the node embeddings also data representing the edge can be used (and learned) while training and predicting to derive a better solution, especially if the edges are similarly meaningful as the nodes in the graph.
- Feature embeddings: Makes a property have a data array that can be learned. Useful when working with categorical data as embedding for each category is learned separately (for each category) and uniformly (for all usages of a category the same data is trained).

The usages of node and edge embedding are clear in this project as the data itself is a network/graph(molecule) of nodes(atoms) with edges(bonds) in-between them, hence the representation of these will be utilized and updated through neural network training.
Furthermore feature embeddings can be used inside of node and edge embeddings as the original data assigned to the edges/nodes are partially consisting of categorical values. In general these values are meant to be constant scores with significant importance as they contain properties and not observed behavior. The meaning of using embeddings for these data elements is to potentially give a specific property value a stronger influence and a separate state on its own (which could be learned by the neural network).

To give an example for this one could take an imaginary chemical property into consideration. Let us assume that this property is of $\mathbb{N}$ and in the range $[1, 10]$. For example one could assume that all values beside 7 and 8 have no influence on the prediction target, however if it is 7 then it is beneficial for a positive prediction and 8 if it has a negative influence. With embeddings seperate learning targets are created for all these values so that this influence can be more easily learned and then utilized for enhancing prediction quality.
This is what is done in this project and also in the reference implementation from [9] with `AtomEncoder` and `BondEncoder`. It takes the features of the atoms and bonds from the molecule and transforms them into a different and, depending on the parameter, a higher dimensional representation depending on which value is stored in the respective feature. The transformed data arrays for each feature are then summed up to produce the new representation which is in this implementation then the node or edge embedding. [9]

2.1.4 Dropout

Dropout is a method for both neural network and graph neural network training that acts as a regularizer and ensemble method. According to [15] there are two types of dropout in GNNs which is dropout on feature maps and dropout on graph structure elements. The first one drops a part of the feature/data while the other option drops a whole part of the data like the node or the edge.[15, p. 1128-1131]
In this project the chosen default model from [9] (as can be read in section 3.2) uses the feature dropout type. In the used implementation this means that elements will be randomly set to zero with the probability being sampled from the Bernoulli distribution. Afterwards the remaining values are scaled using the formula $\frac{1}{1-p}$ with p being the set probability of the dropout.[12][1]
This is not a technique exclusive to graph neural networks as this function works by randomly zeroing values in an array which is well applicable in neural networks. It is applied in GNNs after a layer propagation which can be applied as a mask using matrix element wise multiplication or selection techniques. As an alternative to Bernoulli distribution also Gaussian distribution could be used. [15, p. 1131]

2.1.5 Graph Isomorphism Network Convolution

Graph Isomorphism Network Convolution (layer) (GINConv) is a graph neural network component which acts as a convolution layer meaning that it interacts with the representation of other nodes to update the current node embedding(s) in a manner of aggregation.
For the realization of this component Multi-Layer Perceptrons (MLPs) are needed to realize the functions f and ϕ described in Collary 6 of [17]. For this it is possible to only use one MLP in the manner described in the following

[1]As described in their documentation: `https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html`, last accessed 24.05.2023

function with ϵ being a learnable parameter and h representing node embedding(s):

$$h_v^{(k)} = \text{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right) * h_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right)$$

In essence this formula means that a factor/fraction of the original node embedding of the node will be combined with the node embedding of its neighbors to get the new node embedding, after inputting it into a MLP which can be done by using a normal neural network layer like a linear layer (in combination with other neural network modules).[17]

2.1.6 Early Stopping

Neural networks tend to overfit the model after learning for a certain amount of epochs. To be more specific this means that the training algorithm becomes to focused on predicting the train data set perfectly so that the test sets accuracy is neglected. While the loss for the train set continues to drop the test set accuracy decreases (and hence the test loss increases). To prevent this, one could make optimizations to the model such as using a more overfitting-resistant model, by using a penalty for overfitting if possible or to stop the training process early before the test set accuracy decreases. [14]
For this the easiest technique is to split off a validation set from the test set and to use this to observe the loss or accuracy scores of the validation set. By observing this sets scores it becomes possible to roughly predict the scores for the real training set helping to find out if overfitting occurs.
In simplified terms the algorithm should stop if the training is successful (i.e. if the train loss decreases) but the performance for the validation set (and hence theoretically also for the test set) deteriorates. It is not straight forward which technique to use for this early stopping, the paper [14] discusses a few of them. The implementation used in this project will be explained in section 3.2.4. [14]
However please note that due to the fact that in this project more focus lies on theoretical capabilities and not practical results it will only be implemented for an experiment and not throughout the project. The reason for that is that it is undesirable to make it stop early but rather to see what the best result in the number of training epochs for the test dataset is rather than one momentary snapshot of the model.

2.2 Random Forests

As the term "forest" suggests, this machine learning component consists of multiple to many trees which are randomized (during their creation process). The mentioned trees in this case refer to decision trees/regression trees which themselves depend on multiple random variables. Random Forests as well as decision trees are methods in supervised machine learning which means that on the start of the algorithm some labels are known to be true in advance while unsupervised machine learning techniques would have no labels at all. Random Forests are mainly used for classification (also multidimensional) but can also be applied to regression tasks. Read more about Random Forests in the cited literature for this section and in any introductory paper on this topic. [5, p. 1-8][2, p. 197-204]

2.3 Dataset

The dataset used in this project contains measurements on molecules in experiments, hence the dataset stores experiment-molecule pairs together with the information whether the molecule was active in this experiment or not. Each entry of the dataset consists of the following values/parameters:

- aid - experiment identifier
- cid - molecule identifier
- smiles string - compact representation of the molecule structure as a string. can be converted back to the molecule atom graph and is used to extract information on the graph (done by the `pytorch-geometric`[7] function called `from_smiles(smiles)`[2])
- activity - indicates activeness, values: 'active' and 'inactive'

The dataset contains 41620091 data entries of which 41007834(98.5%) are inactive and 612257(1.5%) are active. It contains 2481 experiments and 455079 molecules. In this project in the first step when using Random Forests (RFs) on each experiment, a subset of experiments will be selected from the dataset for the GNN part of this project. The reason for this is to keep the experiments for determining the effect of self distillation in a manageable timeframe in terms of execution time and to work on data that is likely to be machine learnable. See more about this in section 3.2.5.

2.4 Self Distillation

2.4.1 Motivation

The motivation for this project and hence for applying self distillation on a GNN was initially a program called AlphaFold. This is a software that aims to create the 3D structure of a protein based on their amino acid sequences, which is an incredibly hard problem with only a tiny fraction of all possible data combinations known (and tested in experiments). To tackle this task a complex neural network is needed - as can be read more about in [10].
The aforementioned paper illustrates that using self distillation of the kind shown in [16] (which is the self distillation approach taken in this project) is

[2]see documentation `https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/utils/smiles.html`, last accessed 24.05.2023

beneficial, meaning that the authors managed to achieve better results by utilizing this technique. In their algorithm they apply self distillation on unlabeled protein sequences and estimate the related accuracy to integrate it into their learning phase. In their process they create a second dataset of self predicted sequences with high confidence and re-start the learning process using the original data mixed with this artificial data.
In their approaches the authors also apply noise in the form of mutations and random data masks which corresponds with what is later on extracted from [16] in the next section. [10, p. 583-587]
The motivation for this project hence is to use similar approaches to [10] and [16] on a different and simpler dataset (in comparison to this high dimensional input data and complex prediction target) to try to increase the performance. The idea is to stay within the same field of study in the problem setting (within chemical context) but instead use a GNN as a model for this project. The goal is to see if self distillation can be applied to this problem and model setting.

2.4.2 Methods and Definitions

There are multiple definitions of self distillation, but generally they share a main principle - that is that something is to be refined, hence enhanced.

Knowledge Distillation & Self Distillation One definition from [1] defines self distillation as a means of making a model learn the output of another model (of different seed) to potentially improve prediction performance or runtime performance (depending on the goal). A real world example for this definition would be learning a teachers textbook, rewriting and publishing a revised version for the next student generation. In the course of doing so the correctness of the contents are potentially improved.

Multistage Self Distillation Another example for a definition would be from [18] which is similar to the previous definition as it uses the results of one model (teacher) to update multiple student models which are trained simultaneously. This can be seen clearly by looking at Figure 2 of [18] in which the setup contains a pipeline of neural network modules leading to the teacher output prediction at the. From this pipeline there are intermediate branch-offs to student output predictions, hence not traversing the whole pipeline of neural network modules. The purpose of this branching is to get multiple feedback points to learn specific parts of the model (at different stages) to increase prediction accuracy and runtime performance. The model is quite complex and also uses special loss calculation and learning methods to make it work. To give an explaining real world example to this setup it would be: having a teacher teach three students - one from elementary school, one from high school and one from university. The teacher would himself learn a topic using input from e.g. books and teach it to the students. The teachers understanding is not too great and the goal is to refine the knowledge on different levels from simple to complex matters through teaching.

Noisy Student - Self Distillation There are potentially more different interpretations of self distillations, but let us focus on the definition applied to this project. The interpretation of self distillation in this project is based on [16] which is similar to the definition discussed before from [1]. Both techniques have in common that there is an iterative process of a student learning from the teacher and a student becoming the teacher afterwards.
But instead of taking the output of the teacher as the learning goal of the student there is a change to use the output of the teacher for data which label is unknown. In essence this means that the data on which the original teacher learns stays the same, however the training dataset increases a set of elements which labels were generated by the teacher. As an imaginary example one could imagine having an unquestionable input data source akin to unmistakably proven laws and facts and having a teacher learn them. The teacher would use his acquired knowledge and make inferences on similar topics or the same topic with elements that have not been discussed. This knowledge would be passed to a student to learn it and make his own inferences. However in reality the problem is that wrong inferences and maybe also the basics might be wrong - thus posing a potential problem to this method.
The basic working of this method/algorithm is the following:

1. Train teacher with labeled data
2. Infer new labels for unlabeled data
3. Train student with equal or enhanced model on union of old and new data - (potentially noisy and additionally noised on purpose)
4. Repeat the cycle by making the student the new teacher and resume from step 2

[16] There are some further properties discussed in [16] which will be part of the experiments in section 4:

Noise The noise explained in [16] comes in three forms: data augmentation, dropout and stochastic depth. Dropout has already been explained separately in section 2.1.4 as it is a major concept of this project. Data augmentation would mean that the input data is altered to make the input similar but different from the input, this however is difficult in this problem setting as altering molecules is dangerous as a single change in a molecule could change the molecule from being active in this scenario to being inactive and potentially have dangerous side effects, hence it is not applied in this project.
Stochastic depth is an interesting technique which randomly skips layers while learning and may be incorporated in pre-existing layers. In this project there is a small experiment with this and a related technique in section 4 however it is not used in the other parts of this project.

Label type It is also distinguished between soft and hard labels which means if the output of the teacher model for the student data is an explicit integer label/class or the score before transferring it to the real class labels. In this case there are 2 classes: active(1) and inactive(0), so hard labels would either be zero or one while soft labels would e.g be 0.2 or in general a value in the range $[0, 1]$.

3 Methods

This section introduces methods that will be relevant in the latter experiments and in analyzing the produced results. The knowledge supplied in this section is aimed to be eliminating the need to consult external material to understand the following contents or to reduce the amount of additional readings as much as possible.

3.1 Metrics

For measuring the performance in terms of accuracy in this project five metrics will be employed. Please note that due to the nature of the dataset (described in section 2.3) being quite unbalanced the accuracy scores will use class balancing where possible. More about this balancing will be explained in a paragraph in this chapter.
The computation of the used metrics require the following components:

- True Positive (TP) - correctly positive labeled element.
- True Negative (TN) - correctly negative labeled element.
- False Positive (FP) - falsely positive labeled element.
- False Negative (FN) - falsely negative labeled element.

Accuracy The most standard metric for classification metrics which is the simple accuracy defined as:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FN + FP}$$

[3]

Balanced Accuracy Accuracy metric that takes an unbalanced class distribution into account:

$$\text{balanced accuracy} = \frac{1}{2} * \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right)$$

[3]

Receiver Operation Characteristics Area Under the Curve score The Receiver Operation Characteristics Area Under the Curve (ROC AUC) score or in general the ROC curve are commonly used in machine learning and in its curve representation act as a good way of visualizing the performance of a model. The curve itself has the TP-rate plotted on the y-axis while the FP-rate is plotted on the x-axis. The ROC AUC itself is then, as the name suggests, the area under the curve meaning that the score itself is representing an area. The value itself, similar to other accuracy metrics, is still in the range of $[0, 1]$ as the area of a ROC-diagram is a unit square itself and the ROC AUC is only a portion of it.
In this report however it will not be in detail explained how to compute this score as this is beyond of a simple introduction to this metric, read more in [6] to learn more how to generate this curve and hence the AUC score explicitly.

Precision Accuracy metric that determines how many of the positively labeled data points were actually positive:

$$\text{precision} = \frac{TP}{TP + FP}$$

[3]

Recall Accuracy metric that defines how many of the true labels were identified as positive by the model:

$$\text{recall} = \frac{TP}{TP + FN}$$

[3]

Matthews Correlation Coefficient (MCC) This metric was included midways as a mean of adding another score that may be able to tell more about the performance of the experiment and also if the model is overfitting or not. It is calculated using the following formula:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}}$$

[3]

Balancing of metrics While some metrics are inherently balanced like the balanced accuracy and the MCC, other metrics are not balanced and hence sensitive to uneven label distributions. The precision and recall score and ROC AUC not inherently balanced but can be made such by using a setting in their respective functions in `sklearn`[13].
Surprisingly ROC AUC is not balanced by default (especially not in its multiclass version), however the used variant for binary ROC AUC score calculation

in this project is balanced. To make sure the balanced version is used, the parameter is manually set to balanced in this case as well.
The way balancing is implemented in `sklearn` is by using different weights acquired from the `multilabel_confusion_matrix(...)`, which can be seen in the detailed implementation on their GitHub repository[3] in the folder 'sklearn/metrics/_classification'. To be more precise, the methods mentioned above use a weighted average of multiple values which is calculated via the formula $\frac{\sum_{i=1}^{n} value_i * w_i}{\sum_{i=1}^{n} w_i}$. In this formula the values amount to the number of classes in the dataset - in this case there are two such classes (positive and negative). The weights are calculated using a Multiclass Confusion Matrix (MCM) of which the class wise sum of the TP and FN scores result in the computed weights. Essentially this means that the weight for each label corresponds to its true class occurrence count. The values are of course the metric scores for each class. [13]

3.2 GNN choice and setup

For this project the main aim is to find out whether Self Distillation (SD) works on an already established GNN model that is known to be well-performing. The main requirement of the model to choose was to be quite simple for easier implementation and to be containing well known components. To be precise, the goal is to have a well documented model/component and hence is also implemented in the Python library for graph neural network processing in this project which is `pytorch-geometric`.
For this purpose the Open Graph Benchmark[9][8][4] was used. This is an open benchmark on graph related problems; the goal of this site is to enable fair comparisons between models on specific graph related problems using predefined metrics and scores.
On this website/benchmark there is also a differentiation between node-, edge- and graph-level labeling in terms of benchmark leaderboards (see section 2.1.2). From this leaderboard[5] the model to be used in this project was chosen. In this list there are 2 main candidates for the model in this project: GINConv and GCN. As another work of the author already mainly focused on GCN this project uses GINConv as its model. In the hierarchy on the OGB 'graphprop' leaderboard it is ranked place 32[6] which is meant to show that the model is quite simple although well performing. The related paper and code are linked as [17] and [7] in the overview.

[3]`https://github.com/scikit-learn/scikit-learn`, last accessed on the 16th of May 2023
[4]`https://ogb.stanford.edu/`
[5]`https://ogb.stanford.edu/docs/leader_graphprop/`
[6]state of the 16th of May 2023
[7]`https://github.com/snap-stanford/ogb/tree/master/examples/graphproppred/mol`

3.2.1 Embeddings

As described in section 2.1.3 this project uses embeddings for the atom features, bond features and of course for representing nodes inside the training process. The input data is chemical data transformed to graph notation by `pytorch geometric` with `from_smiles(...)`. Through the course of the prediction process it is then put through the embedding generation for the nodes and the edges to map these features to feature dimension 300 (parameter choice, although only this default value will be used throughout the project). This then is the node and edge embedding that is used when training and testing using convolutional layer components (described in section 2.1.5 and 3.2.3).
As described in section 2.1.3 for each chemical feature a seperate embedding exists which then is used by computing the sum over all features to convert the input data into an output feature that is of higher dimension than the input in this case. The reason for this, as already previously hinted, is likely due to the assumption that chemical features are much more powerful than real world measurements and thus can be of great help when trying to fix specific (experimental) behavior/activity to a specific (chemical) feature.

3.2.2 Dropout

As briefly introduced in the preliminary section, and set into context in the subsection on self distillation, the major noise used in this project is dropout. Dropout in this context means the use of a dropout layer applied on the node embedding data. For that the dropout module from `pytorch` of the neural network functional package is used. As mentioned in the preliminaries it works by randomly zeroing entries in the node embedding with a Bernoulli distributed probability for each data entry. This method is a mean of simulating that knowledge may be incomplete and further enhances the learning of the non-zeroed values.
In the project different values for the dropout will be chosen, most commonly 50% or 80%, in some cases also no dropout at all.

3.2.3 GINConv Layers

As introduced in the preliminaries this project will use multiple convolution layers of the GINConv type with dropout layers in between. The number of this convolution layers is mainly determined by the default parameters provided by the original OGB[9] setup in their code repository on GitHub and as transferred to the implementation of this module in the code base of this project. By default the number of layers is set to 5, however in some experiments it will be discussed whether this count is producing overfitting and will be attempted to be reduced to 1.

3.2.4 Early Stopping

As briefly stated in the corresponding part of the preliminary section, early stopping will not play a major role in this project as the goal is not to evaluate realistic capabilities and practical usage of the model but rather determining whether a theoretical benefit between normal training and training using self distillation techniques exists or not.
Hence the implementation of early stopping is only in the context of a single experiment concerning the application of using early stopping to prevent potential overfitting which could worsen the effect of self distillation improvements.
The chosen implementation in this project is straight forward: while training the accuracy scores of the validation set (specifically split off from the training set for this experiment) is used and the best metric score is tracked. The algorithm will train until the point where there are j epochs in which the metric score does not improve beyond the currently stored best score. There is also a parameter controlling to what degree the score must increase to consider it an improvement and to reset the counter, although this parameter is never actually used while testing.
In the experiment the parameters concerning early stopping are set to the following values:

- `min_improvement` = 0 (any improvement counts as such)
- `measurement` = "roc_score" (the ROC AUC score will be used as the accuracy criteria)
- `num_epoch_wait` = 7 (if the accuracy scores do not increase for 7 epochs, stop training)

3.2.5 Dataset Reduction

As shown in the section 2.3 the dataset used in this project is quite large having 4M data entries and 2481 experiments. For the sake of convenience and being able to get faster results in combination to being able to test more it was decided to reduce the dataset to a fraction using only some selected experiments. For this purpose when attempting to apply self distillation on the dataset using RFs to figure out whether self distillation also works for this machine learning model, the results will be used to determine which experiments are machine learnable or not. This is done by choosing experiments above a specific ROC AUC score with a minimum amount of elements in general and positive elements in detail. Chosen experiments need to have a ROC AUC ≥ 0.7, must have ≥ 1000 total elements of which 100 elements must at least be present of each class (0 or 1). This reduces the experiment count for the later tests to 27 experiments out of the initial 2481 ones. Later on it will be decreased to 26 because one particular experiment with ID 686978 is too large (and takes to long to run) - as it has over 10 times more molecules as the other chosen experiments.

3.3 Student and Teacher Model

In the following document, especially in the experiments section (section 4), the terms teacher and student model will be used at multiple points. These terms originate from [16] and refer to the setting where the model, referred to as the 'teacher', will be trained and used to label unlabelled elements which will be referred to as 'self distillation' elements (read more in the prelimiary section 2.4).
The model called 'student' will be taking the input that the teacher used to train in addition to the previously unlabeled elements (which have been labeled by the teacher) to train and, hopefully, get better predictions by utilizing this technique.
In general the setup for the student and teacher could be arbitrary, however these models in the project will follow the instructions from [16]. These instructions include the following simple rules: (some of the stated rules have been inferred from the context of the paper)

1. The student model complexity is equal or larger to the teacher model.
2. The dropout of the student model is to be higher or equal to the teacher dropout if it has any.
3. The teacher model learns on less or equal the amount of data used for training the student model

[16, p. 1-3]
Another point to mention is that for the loss function a weighted binary cross entropy loss is used to counter the imbalance in class distribution of the dataset.

3.4 Choice of Self Distillation Items to Add

When reading the previous sections including the sections on self distillation, the attentive reader might have already come to the question which artificially labeled elements to pick to include to the learning process. In general the available elements are plainly the molecules from other experiments which are not in the database for the current experiment. The simplest solution to this problem is to include all missing elements to the training set. However, when keeping in mind that an experiment has 1000 to 10000 elements adding over 400000 further data entries, it may seem excessive to do this as the original data gets extremely diluted in the process.
So after deciding to take a subset of the possible data to include, the question would remain which elements to pick. For this there are different approaches which will be subject to experimentation in a later section:

1 - Selecting most secure elements The output of the model is a scalar for each input graph, optimally containing a value in the range $[0, 1]$. Normally this score would be interpreted in the following way to derive a hard label output: 1 if $val >= 0.5$ else 0. The original score can be potentially interpreted as a

confidence score or (under the assumption that the previous interval condition is fulfilled) as a probability score.
Hence the most secure elements are the entries with values the furthest away from the decision boundary, 0.5. However, as the experiment section will point out, this method has a vulnerability as, even though the loss is weighted and the class imbalance should be somewhat restored, most of the elements to be added are of the 'inactive' class hence shifting the class balance further off.

2 - Selecting most secure elements with balancing This method is aimed to tackle the issue discussed in the previous section concerning the balance issues of the self distillation data that is to be added. A rather simple solution to this would be by enforcing the same balance by selecting the most secure predictions for each class according to the balance of the training data. This way the most confident elements are chosen but the balance is maintained.
Another slightly different variant of this encompasses the possibility of correcting the data inherent imbalance of the classes by adding elements such that afterwards the training set would be class wise split into 'active' and 'inactive' in a ratio 1:1.

3 - Pick random elements In many software architectures and in specific in software architectures for machine learning purposes randomization appears to be an almighty tool that is applied in almost every scenario from data initialization to algorithmic decision. Hence it is also logical to also try this universal tool on this problem, namely by adding random elements from the set of all self distillation data elements.

4 Experimental Evaluations

4.1 General

In this section the later explained theoretical approaches will be tested to see if, using the contents of this theories, it is possible to generate benefit from using SD techniques with the previously described problem setting in combination with the mentioned dataset (see section 2.3). In the content of this chapter multiple theories will be created, explained and checked that make a reasonable assumptions on the truth of the main theories created in this section. Checking in this context means that the described (sub-)theory will be implemented and experiments will be conducted to test if the initial theory is correct or not.
The overall goal/desire, of course, is to make self distillation work using GNNs on this dataset. At this point in time it remains to be seen if the overall theory that self distillation is beneficial to GNNs is true or not. However during the course of the experiments conducted in this section, it will be attempted to find or at least narrow down the reason why or why not this technique works. This is done to make final theories on whether this implies that self distillation is or is not applicable to GNNs in general using other datasets.

Full experimental results Due to reasons of better readability and compactness of this report, the following sections show the results of the experiments only for (a) sample experiment(s). The full results, especially the diagrams displayed in the results can be acquired from the GitHub repository in the results folder. In section A more information on how to access and navigate the repository will be provided.

4.2 Main Theories

There are 2 main theories concerning if self distillation is applicable to RFs and GNNs in this project:

Random Forest - Self Distillation Theory As introduced, a Random Forest is a machine learning technique using trees to generate the class labels. However, compared to a full fledged neural network, there are far too less trainable parameters in addition to not training the existing parameters with Stochastic Gradient Descend (SGD) or a related technique. This motivates the theory that self distillation will not work using Random Forests, meaning that when using this technique on this machine learning component it should produce equal or worse results than without applying this technique. Hence the theory is that self distillation will not work Random Forests.

Graph Neural Network - Self Distillation Theory As the papers [16] and [10] prove, the self distillation implementation used in their respective projects produce a qualitatively better result than without using these techniques when employing neural networks. Although GNNs are different to 'normal' neural networks, as already pointed out in section 2.1.1, they share the same core and thus should generally be even more powerful.
In theory, using convolution modules which aggregate data from neighboring nodes or using a graph decomposition technique in addition to regular neural network modules, it should be possible to employ self distillation techniques to enhance the performance even in graph neural networks. Hence the theory is that self distillation will work on Graph Neural Networks.

Sub-Theories As one could imagine proving or disproving that self distillation works for GNNs is difficult and sometimes not even possible because of two major reasons. The first and simpler reason is that even if it works or does not work it does not automatically prove that this behavior also applies to all other datasets and is hence not problem specific. The other reason is that, like in other machine learning techniques, there are multiple configurations and variants to consider before being able to state that self distillation in general does not work.
While the first problem is not addressable within this project, as one would need to consider a selection of datasets (and GNN models) which exceeds the capacity for this project, the second problem can be handled relatively well.

In the course of attempting to apply self distillation successfully, multiple possible branching options become evident. The term branching options in this case is referring to having the choice to go with method a or method b (The same also applies when different parameter options are available). In the first place one branch is chosen to get a first result on the initial question (which at its root is the main theory described in section 4.2 concerning GNNs). From this base branch another theory is created concerning the question whether another setting/method(and thus a branch) would have been successful or not.
This results in the structure seen in the further sections where theories are created on whether a method/setting will be beneficial (and more beneficial than the alternative). After getting an answer on a reasonable amount of 'sub-theories' on different parameter settings for GNN self distillation a more confident proof or disproof of the main theory can and will be given.

4.3 Main Theory: Random Forest

Theory As already explained in section 4.2 the theory is that self distillation will not work on RFs. The reason for that is that there are no trainable parameters that allow for training and that the Random Forest structure decides based on input data field decision boundaries. The latter argument should signify that adding more elements to the training would at best allow the algorithm to be more stable through having similar data in all contained tree structures and clarifying the boundaries. Although the algorithm might become more stable it also introduces more noise into the prediction generation which may as well warp previously better decision rules.

Experimental Setup In the experimental setup in notebook 'Random_Forest.ipynb' the base method is using either chemical descriptor data of the molecules or the molecule's Morgan Fingerprint as base data as this machine learning component does not accept graph structured data as input. This step is necessary to get a range of values as input to make the RF train on more data than just the id of the molecule.
The teacher and student model are two separate Random Forests with the same parameter setting and the self distillation elements to add are chosen randomly, selecting 20% of the original dataset size - using 5-fold cross validation.
In the course of experimentation the metrics mentioned in 3.1 (except MCC) are collected and then later used for analysis. In the analysis it will be looked at the number of times where self distillation was beneficial(using averaged ROC AUC and balanced accuracy) while the scores from this experiment will then be also used for analyzing which subset of chemical experiments to pick for the GNN experiments to reduce time overhead.

Experimental Results The results are extremely clear, no experiment profits from self distillation with both the fingerprint and the chemical descriptor data generation method. Note that for this 'voting' procedure to reduce runtime, a selection of 10 experiments were randomly selected from the set of large

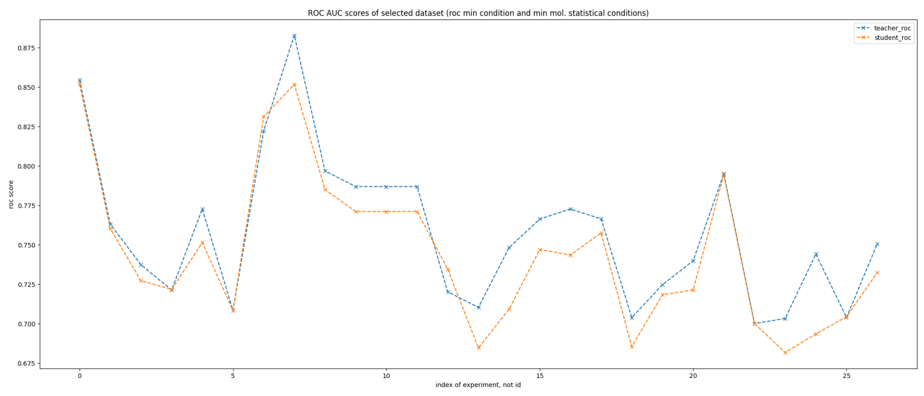


Figure 1: ROC AUC scores of Random Forest for a selection of experiments (see section 4.4)

experiments with $\geq$ 10000 molecules.
In a full version of the experiment, using all the experiments for the experiment subset selection, further results are produced to confirm the previous statement. When looking into the detailed scores in notebook 'RF_exp-analysis.ipynb' (responsible for selecting the chemical experiment subset) it can be seen quite clearly in figure 1 that nearly all chosen experiments have worse ROC AUC scores in the student model than in the teacher model. For this test all experiments were subjected to testing with Random Forests, however the graph only shows a subset of the results. Recalling that this graph contains the better performing chemical experiments and only adds 20% of artificially labeled elements it becomes evident that this model will most likely be not enhanced through self distillation which is why the focus for the further research was shifted entirely to graph neural networks. (Note: for these experiments random molecules not contained in the experiment were chosen for artificial labeling)

4.4 Experiment subset selection

To narrow down the numerous experiment sub-datasets to a reasonable count, a set of rules are applied. The rules are:

1. The teacher model needs to have a ROC AUC of ≥ 0.7
2. The experiment needs to have $\geq$ 1000 molecules
3. The experiments needs to have $\geq$ 100 active and inactive molecules

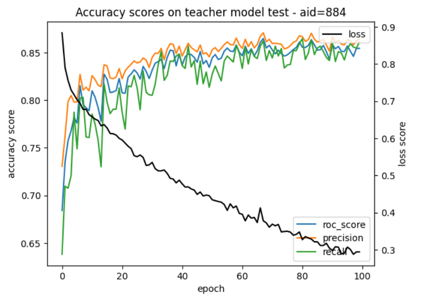
The reasons for these rules are quite simple: First of all the dataset should be machine learnable meaning that a machine learning module can produce a reasonable prediction - in this case a Random Forest Classifier is used as a reference. Secondly the dataset should not be to small or else the predictions

may be unsuited for larger Graph Neural Network Models. Lastly there should be at least 100 molecules of each class or else the experiments may encounter accuracy errors and undefined behavior.
These selection criteria reduce the number of experiments from 1455 to 27. Note that the initial experiment count does not equal the real experiment count 2481. The reason the initial count is not identical is that experimental runs that produced a 'ValueError' while computation could not be recorded. The error occurs if the data put into the RandomForest `sklearn` classifier contains ill-valued data like infinity or 'not a number' or if the training fails because of an accuracy metric error (e.g. that a class is not represented in the testset) or that while training some parameters become invalid.
While this behavior was not initially intended, it helped the selection of experiments as it reduced the number of experiments and introduced a notion of 'fail-safety'. This is meant in the sense that only the most stable experiments are used further on in the test concerning Graph Neural Networks - although the removed experiments might not have been unstable for GNNs.

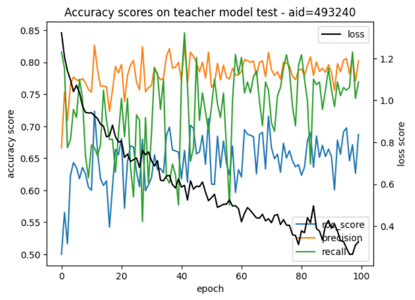
4.5 General teacher theory

Expectations & Theory As mentioned in the preliminaries and introduction, the model for the GNN originates from the OGB benchmark and is one of the best models tested in this benchmark for the datasets used to compare the models (which is also related to chemistry and molecular activity).
Hence the theory is that this model will be able to perform well on this data as well for our selection of experiments.

Results In reality this theory was proven to be (generally) correct. For most of the experiments the ROC AUC is in the range $[0.8, 0.9]$ while some experiments are not working as well and are above 0.7 while only 3 experiments are worse with scores of around 0.6. Figure 2 displays one good example and one bad example.



(a) Example for experiment result with good accuracy and good behavior



(b) Example for experiment result with bad accuracy and good behavior

Figure 2: Exemplary metric history for basic teacher experiment

4.6 Side theory: Stochastic Depth in teacher - Concatenation of Layers

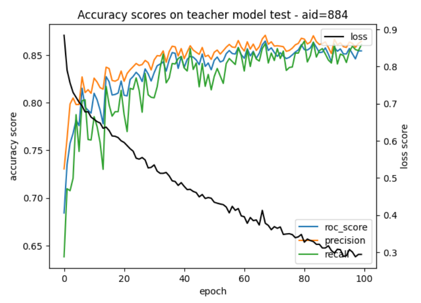
Topic Introduction and Experimental Setup As introduced in the preliminary sections/methods the teacher model uses global mean pooling for generating graph level predictions using the last node embedding. Another variation is using concatenation of all node embeddings after each convolution layer, which is aimed to simulate the principle of stochastic depth. Stochastic depth in this context would mean that a layer can be completely skipped(randomly) while training.
While stochastic depth is not generally used in this project - the technique used attempted here is quite similar. Instead of randomly skipping layers the results after each layer are concatenated to each other and put through a linear network layer. In essence this means that the algorithm may use the information found after multiple stages in training to derive an output.
This can be seen as beneficial in two ways: For once it theoretically should be able to choose the optimal number of layers for the best results and secondly it should be able to train different convolution layers to different degrees. In the course of training the connection to a lower layer may become more influential and thus these layers are trained while it could then make the influence of advanced layers stronger; hence training the whole system equally and individually.
Of course this does not exactly replace stochastic depth as there layers are skipped and the output of a non-predecessor is input into the layer. However it can be seen as a different variant of this method as stochastic depth, as in the course of training the depth of the neural network is changing (i.e. their weighting in the training) which is random in the sense that it is not predictable but not necessarily stochastic.

Theory The theory is that, although this technique might work in general, this method is not expected to bring any benefit. The reason for this assumption is, that there are only few layers and such methods will not be beneficial as stochastic depth is commonly used as a method for deep learning.

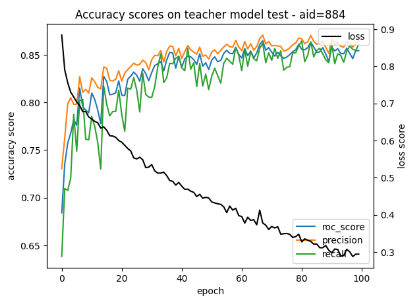
Experimental Results The theory was confirmed by the experiments in which the results were almost the same as the reference metric scores. This can be seen quite clearly in the commonly showcased first experiment in figure 3.

4.7 Digression: Stochastic depth

As hinted in the previous section on concatenation, which was used as a mean to simulate stochastic depth, this technique is not explicitly used in this project although it is mentioned in the category 'noise' in [16]. The main reason for this is, that the chosen model from OGB does not use/support this technique and it was not the goal of this project to majorly change the chosen model.



(a) Teacher accuracy history with using last node embedding



(b) Teacher accuracy history with using concat of all node embeddings

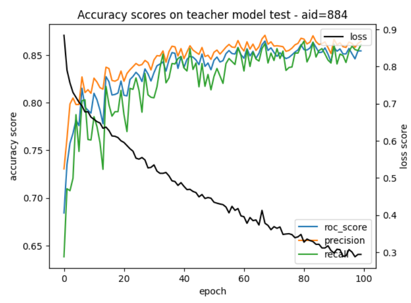
Figure 3: Comparison between concat approach and normal teacher

Furthermore the model setup is not ideal for this method as in most setups there are only 5 layers which are arguably too few to meet the condition of a deep neural network technique such as this. Instead this model extensively uses the notion of dropout which has been introduced and mentioned in multiple previous sections.
Based on a small experiment on one chemical experiment in a seperate branch in the GitHub repository[8], to not disturb the main implementation, it was revealed that using the teacher model (and its standard parameters) it is significantly worse than without using stochastic depth.
The probability for dropping a layer was set to 0.2. The results in figure 4 show the comparison between the stochastic depth results compared to the original teacher result. From this figure it can be safely assumed that stochastic depth would not be beneficial for integrating it into the student model as it is not even capable in its default version to produce a similar accuracy score as the original teacher.

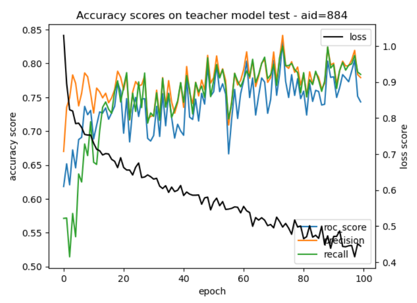
4.8 Side theory: Initial setup

Theory In the first setup the model parameters and settings are set to their default values. The main question addressed in this theory is whether the initial approach is able to bring benefits in terms of prediction accuracy. For this there are different variants available: The first one is, when choosing the most confident self distillation elements, whether the balance of classes should be kept or not(see section 3.4). The second one is whether the self distillation data input for the student model should be soft or hard labels (meaning whether the output should refer to the class labels - hard, or the output before deriving the class predictions - soft).
In this theory there will not be any divisions between the variants as it concerns the general potential of the initial model setup; the variants are of concern in

[8]see section A



(a) Teacher accuracy history with using last node embedding



(b) Teacher accuracy history with stochastic depth applied ($p = 0.2$)

Figure 4: Comparison between concat approach and normal teacher

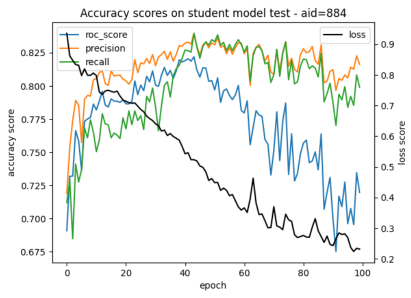
further theories.
For this theory of course it is assumed that it will work as this is the initial setup of the GNN and as the whole project assumes that it will work. However as it is with most projects, the initial approach is very likely to fail. Hence the theory will only contain that self distillation applied to this setups will produce better or equal results as the teacher model.

Experimental Setup For the experimental setup the models' (embedding) size is equal and the dropout in both models is 0.5. The number of self distillation elements to add is set to the same size as the dataset. Both models were run for 100 epochs.
In the self distillation preparation, the self distillation elements are appended to the training data set and not shuffled. As in all later experiments a 20% test set if split off the original data using `StratifiedShuffleSplit` which essentially splits the data so that each group in the data is present in both training and test set.

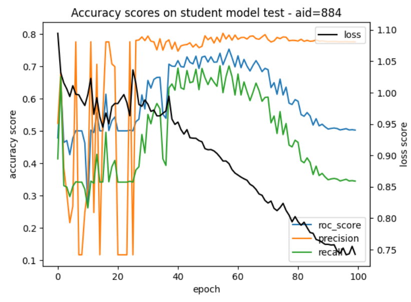
Experimental Results The results from the tests was very unexpected because, although it works and is able got get results close to but below the teacher model accuracies, the learning behavior is not stable and after reaching a peak close to the final epoch in the teacher model the accuracy scores plummet. This can be seen in the figures 5 which show the metric history of a single experiment (for all variants).
This very clearly shows that there might be an issue in the implementation of the student model setup.

4.9 Unstable Student - Fixing self distillation

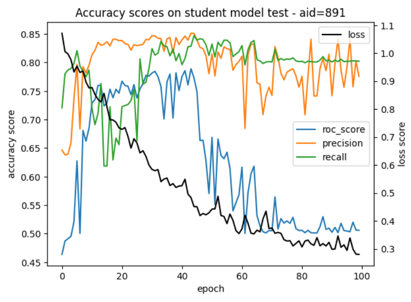
Theory As can be read in the previous section, the student model with the initial setup produced unexpected results. Although they were temporarily quite



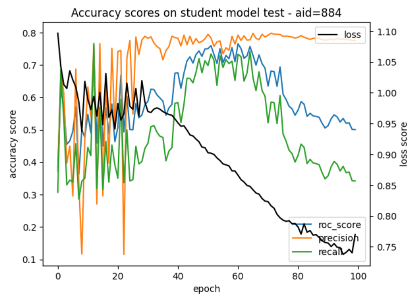
(a) Unstable metric behavior in experiment with soft labels and self distillation method 1



(b) Unstable metric behavior in experiment with soft labels and self distillation method 2



(c) Unstable metric behavior in experiment with hard labels and self distillation method 1



(d) Unstable metric behavior in experiment with hard labels and self distillation method 2

Figure 5: Metric histories for initial theory experiments.

high, the accuracy scores decreased significantly afterwards. After evaluating the results and analyzing the source code again, it became clear that a procedural setting is likely the cause of this unstable learning behavior. The theory is that the appending of the self distillation items to the training set produces this behavior(which was previously not an issue in RFs as the order of training data is insignificant there) and hence shuffling the training data with the self distillation data should resolve this instability.

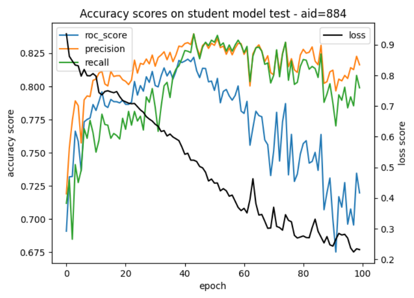
Experimental Setup The change compared to the initial setup is relatively small with only including a shuffling mechanism after the merging of training data and self distillation data. This will be evaluated and compared with the setting of soft labels without keeping the class ratios - other side theories will branch off this setting here and explore variations. The dropout for this setup is set to 80%. For dropout theories and the reason behind using this dropout, see section 4.13ff.

Experimental Results In figure 6 the comparison between the unshuffled and shuffled version for one experiment can be seen. Furthermore the accuracy history of the corresponding teacher experiment is displayed to compare to the non-self-distillation method to see if the previous sections theory can now be confirmed or not.

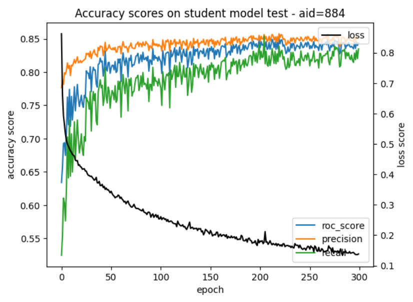
As can be seen these results are quite clear in two instances: First the results certainly show the theory of the unstable behavior fixing theory to be correct as the scores of this setup behave nicely and continuously increase throughout the tested 300 epochs. Secondly the theory that the self distillation results are equal or higher to the teacher model scores from the previous section has been proven to only partially apply. The scores and the histories in general are quite similar although the scores are continuously lower than the teacher model highest scores. This means that the theory is partially approved in the sense that it is roughly equal, however the student model was not able to outperform the teacher model.

4.10 Side theory: Self Distillation selection method

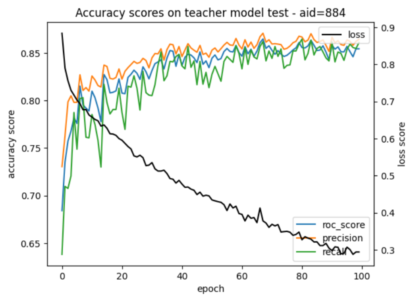
Theory Now that a similarly/equally well working student version has been established, the question is whether other self distillation selection methods yield a better result. These variants have been already introduced in section 3.4. As described, there are 2 different methods besides the baseline method of choosing the most secure self distillation elements, namely picking the most secure elements but keeping the class balance and randomly choosing elements. In theory these methods should produce better and more stable results. The reason for this is that if too many labels of one class are added, it could happen that firstly only the prediction performance of one class is enhanced. The second reason is that the prediction performance of the minority class could be worsened due to the machine learning module learning more and more elements of one



(a) Unshuffled model metric history



(b) Shuffled model metric history



(c) Teacher model accuracy history

Figure 6: Metric histories for unstable student fixing theory.

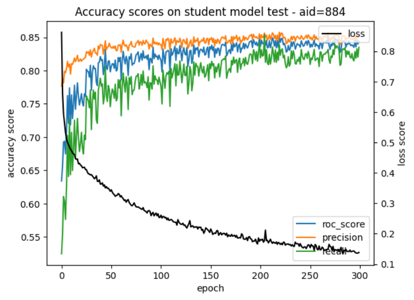
class - even if the class labels are balanced through a balanced loss function. Hence the theory is that both of these variants will produce either similar or superior results compared to the baseline element section method.

Experimental Setup The setup for testing this is the same setup from the last theory with dropout 80% and embedding dimension 300.

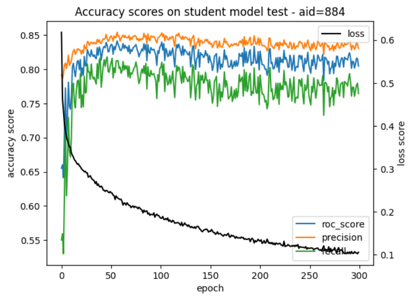
Experimental Results The results themselves are mixed. In some cases (for keeping the ratios) it performs as expected with better values or better training behavior, in other cases not. For randomly selecting elements the results show this method to perform quite well and that it has a more stable behavior than the other variants in some cases. An example for both previously mentioned cases for the ratio keeping approach can be seen in figure 7 and 8 where the three methods are compared for 2 experiments.
In this figures it can be seen that in the first case the first variant performs best while the second variant is still stable but slowly decreasing after a temporary peak until reaching a slightly suboptimal solution. The random approach in this case is stable but not as well performing.
In the second case the second method is best performing and most stable while the first and third method are working well and stable but below a ROC AUC of 0.8.
In conclusion for this theory it is hard to ascertain if method 2 and 3 are more beneficial than the others, as this appears to be quite case specific which is a common occurrence when dealing with machine learning components/methods. Technically the random selection method is the most stable, determining which version is better in terms of metric score results is difficult to tell. The initial theory was partly proven to be correct, with the random selection method as a viable alternative method, as it is in most cases more stable than the other two versions, however all methods do not have (significant) better performance than the initial student and teacher model.
In most of the further experiments the base for experimenting will be the second method as the class balance maintenance method is simpler to use and is more straight forward in theoretical predictions because of the constant class balance. (For the sake of stability it would have also been an option to use random selection, however this experiment addition is one of the chronologically latest experiments but for the sake of better readability the experiments were grouped into logical clusters.)

4.11 Side theory: Label type

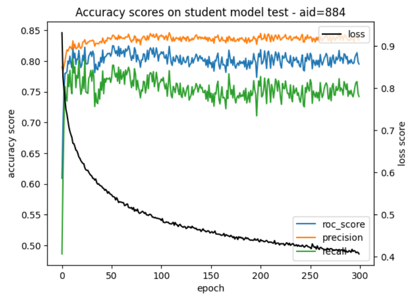
Theory As mentioned in the preliminaries, a label for self distillation elements can either be soft or hard - meaning if only classes are allowed(hard) or if continuous output is possible(soft). In this model setup it was defined as whether the output of the model is either 0. or 1. for each molecule or if the output is in the range of $[0, 1]$. The potential of using soft labels it to capture the learning



(a) First method - selecting most confident

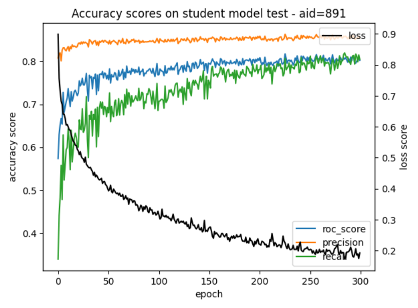


(b) Second method - selecting most confident but keeping ratio

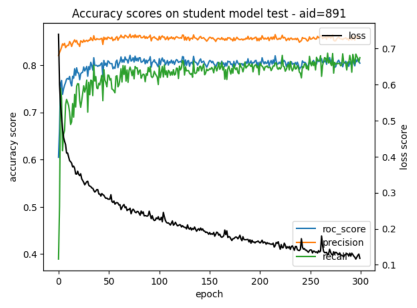


(c) Third method - random selection

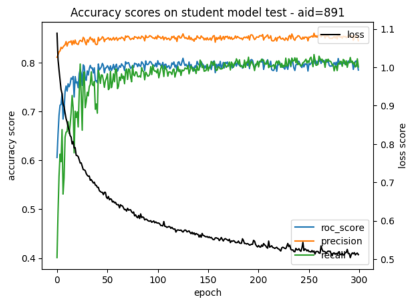
Figure 7: Metric histories for self distillation element selection theory - example 1.



(a) First method - selecting most confident



(b) Second method - selecting most confident but keeping ratio



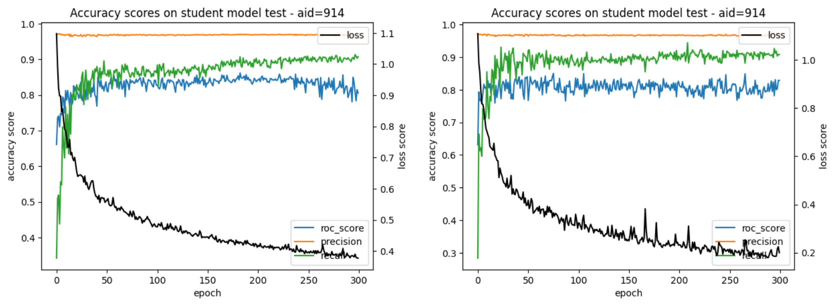
(c) Third method - random selection

Figure 8: Metric histories for self distillation element selection theory - example 2.

state of the teacher and directly train on it. The possibility when correcting this to an actual class label is that uncertainties are corrected and the student will interpret a vague guess of the teacher as an absolute fact.
The theory before experimenting is, that using hard labels will be worse than using soft labels as information will be lost.

Experimental Setup For this the random selection method was chosen to get the largest possible difference between the soft and hard labels. I.e. with the other selection strategies it would have been a small difference as per the criteria selecting the elements with the score highest difference (absolute) from the decision boundary 0.5 which naturally indicates the effect would be rather small. With the random selection strategy also values close to the boundary can be chosen which has the most 'corrective influence' on the labels.

Experimental Results The results are neither proving or disproving the initial guess as the results are almost the same. In some instances the results for one variant are slightly better than the other, but it is not constant which setting is better or worse. For one example this can be seen in figure 9.
The internal conclusion of this section however is that, although the results are almost equal, soft labels are to be preferred due to their initially assumed potential. As will be elaborated more in the outlook section of this report, it may also be possible to alter the definition of soft labels in future models/projects to include more values than the initial label output of the model.



(a) Soft labels - random selection strategy

(b) Hard labels - random selection strategy

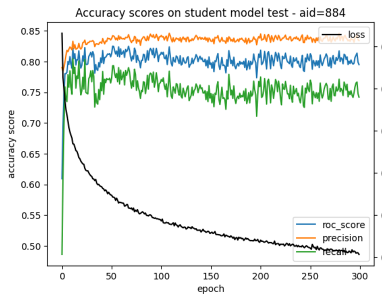
Figure 9: Metric histories for label type theory.

4.12 Side theory: Class balancing

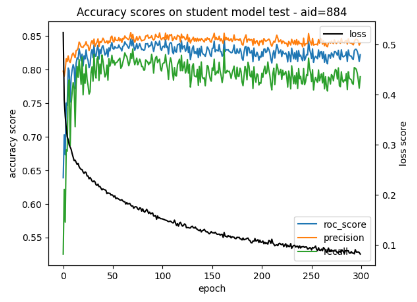
Theory In a previous theory section on the selection method of self distillation of elements to add, the balance between classes has already been discussed. Inherently the class distributions in the experiments are not equally balanced (i.e. a 1-1 split) but in reality the class label distribution can be anything from 2.9%(aid: 914) to 0.44% (aid: 1418) and possibly even below this score. The main idea in this theory is that the class distribution can be shifted to a 50-50 class label distribution using the self distillation elements as a counter-balance. Although this approach sounds interesting and potentially beneficial as possible unwanted behavior due to the class imbalance could be solved, however in reality the assumption (and hence theory) is that it will not be beneficial to the prediction performance as to much noise and incorrectness in introduced onesidedly for one class label.

Experimental Setup The experimental setup is the same as in previous sections with soft labels chosen to keep ratio in one case and correct ratio in the other case with same number of self distillation elements to add.

Experimental Results The results turned out to be roughly as expected, there is no actual benefit for the ROC AUC when using balanced labels and the learning behavior interestingly is a bit more unstable as the accuracy metric histories tend to fall slightly at the end. Surprisingly, in a few cases this modification was able to boost the metric scores, although only in roughly 3-5 experiments. One of this cases can be seen in figure 10. Overall, this theory was confirmed to be true.



(a) Keeping the class balance



(b) Correcting class balance to 50% each.

Figure 10: Metric histories for class balance theory.

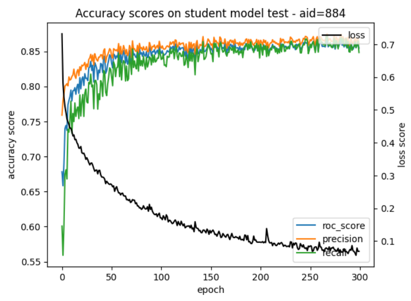
4.13 Side theory: Dropout check

Theory As mentioned in one of the first theories, the dropout there was set to 80% while the teacher originally works on 50% dropout. The reason to use a higher dropout originated from the descriptions in [16] where noise was one of the key components to achieve a performance increase with the student model. There they suggest the dropout to be equal or greater to the teacher model, however, as they have notions of other noise (like the augmentation and

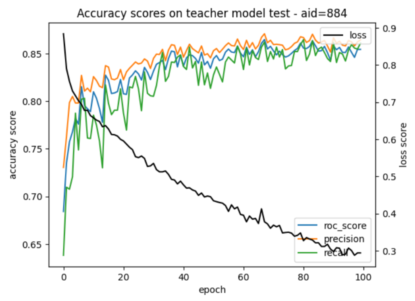
stochastic depth) which are not present in this model, it was decided to go with a higher dropout rate from the start.
The theory now is that, after seeing so many experiments that are relatively equal to the maximum performance of the teacher model, that the previously seen (small) metric difference between the student and the teacher originates from this change in dropout.

Experimental Setup To check this theory two experiments are conducted, one being a reduction of the dropout of the initial first working model with 80% dropout and reducing it to 50%. This initial model was the soft label model using safest element selection.
The second experiment is to test which accuracy results the teacher model generates when using 80% dropout and compare it to the student results.

Experimental Results The results clearly show that the theory is true. The metric scores of the teacher and student in the described cases above are almost identical, showing that the student model results so far have been almost exactly like the teacher model results. The results for both cases can be seen in figure 11 and 12.
However what is also clear with this, is that the model technically works slightly better with a dropout of 50% than with 80%. Never the less because of stated reasons in the theory explanation, the student model will continue to use the higher dropout as a mean of noising the student more than the teacher (for with the 50% dropout was a standard parameter in the chosen model from OGB.)



(a) Student model with lower dropout (50%)

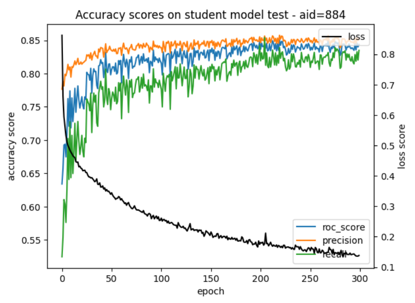


(b) Teacher model with standard dropout (50%)

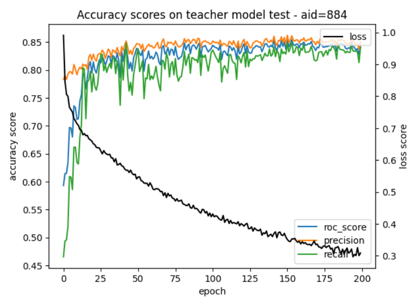
Figure 11: Metric histories for dropout check theory - 1.

4.14 Side theory: Self Distillation set size

Theory One thing that was not explicitly specified in [16] is the number or ratio of the self distillation elements to add in an iteration. Initially in the previous experiments the idea was to add (approximately) as many items as



(a) Student model with standard dropout (80%)



(b) Teacher model with increased dropout (80%)

Figure 12: Metric histories for dropout check theory - 2.

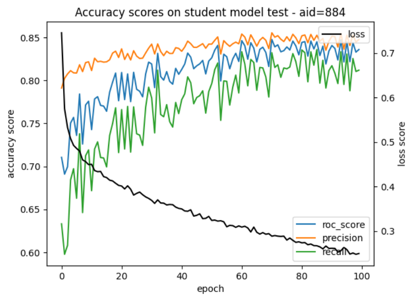
there are in the training set. The question now is what happens if this count is increased or decreased significantly. In the case of increasing, a good effect could be that the last state of the teacher is transferred to the student 'more completely'. However increasing the element count would mean that the true data points get buried in artificial data which would be counter-beneficial. In the case of decreasing on the hand it might prove to be beneficial because the true labels are not buried.
However as shown in the theory section on dropout currently the problem is not reaching the same accuracy as the teacher but surpassing it which is why the theory is that the self distillation data increase should be beneficial while the decrease should not change the results significantly.

Experimental Setup In the experimental setup the soft label type in combination with the keep class balance selection method is chosen while the number of self distillation elements is either half, double or quadruple of the training set size.

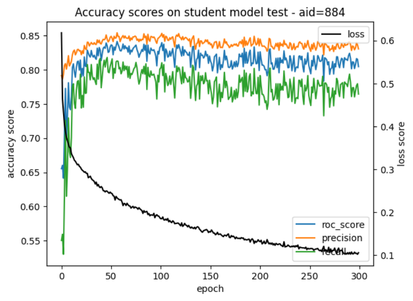
Experimental Results After testing, the theory could not be confirmed or denied as the results from each of the three versions are nearly identical indicating that the number of self distillation elements is most certainly not the problem except if a very large number of self distillation is required so see effects. A comparison of the metric histories for all three versions can be seen in figure 13.

4.15 Side theory: Enlarge student model

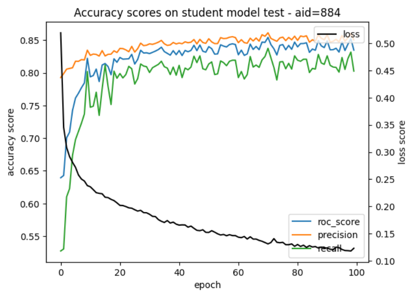
Theory Another proposed method from [16] contains the enlargement of the student model in the meaning of increasing the model parameter count. In most cases this could be beneficial as this would mean more trainable parameters and in combination with self distillation techniques it could mean fitting more



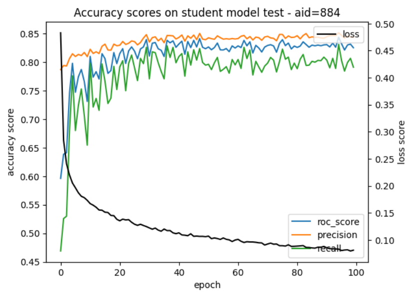
(a) Ratio: 0.5



(b) Ratio: 1.0



(c) Ratio: 2.0



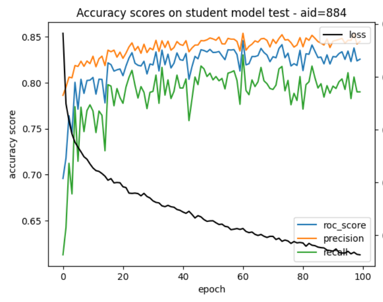
(d) Ratio: 4.0

Figure 13: Metric histories for self distillation set size theory.

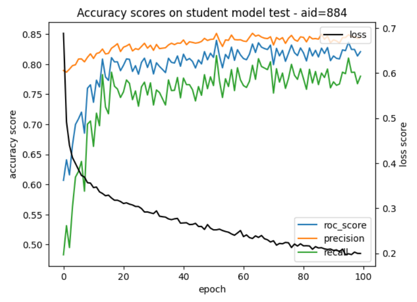
of 'ground truth' behind the addition of self distillation elements, however the current model parameters are quite high so in theory it should not produce any better results.

Experimental Setup To test this theory, two main model parameters can be enlarged: the number of GINConv layers and the size of the node and edge embedding. In the case of the embedding increase test the value is doubled (300 $\rightarrow$ 600) and in the case of the layer count it is changed from 5 to 7 layer.

Experimental Results The results are as expected, this change was not able to break through the teacher max accuracy top score as well which can be seen in figure 14.



(a) Embedding increase



(b) Layer increase

Figure 14: Metric histories for enlarge student model theory.

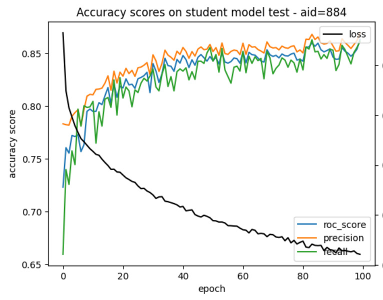
4.16 Side theory: Overfitting

Theory One question one might ask himself is if overfitting may prevent the effects of self distillation to take place in this project. Generally it appears to be hard to overfit a GNN in the first place and the results of the test set are good and stable which is why the general theory is that overfitting does not occur in this setup and if it occurs, it does not change the results.

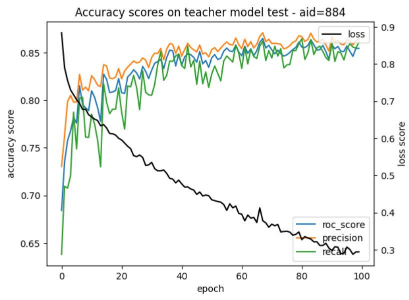
Experimental Setup To check this three tests are carried out:

1. Set dropout rate in teacher model to 0 and test whether the student perform better on this self distillation element labels or not. (technically not overfitting itself but related as a high dropout might learn individual features too intensely)
2. Reduce the number of GINConv layers in teacher to 1 and test if students perform better
3. Test if overfitting in the teacher model happens within 3000 epochs - e.g. if the metric scores drop after some time.

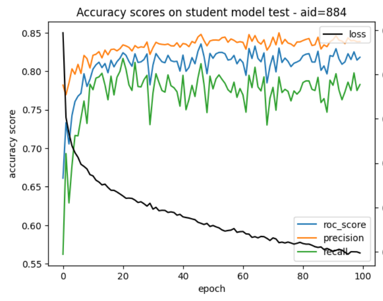
Experimental Results Overall the results show and indicate that overfitting is not happening in the chosen GNN model and that it does not (positively) influence the student's performance.
For the first test the results are quite clear if compared to the respective teacher result with dropout. This can be seen in the example depicted in figure 15.
For the second test the number of layers was reduced to 1 layer, however the



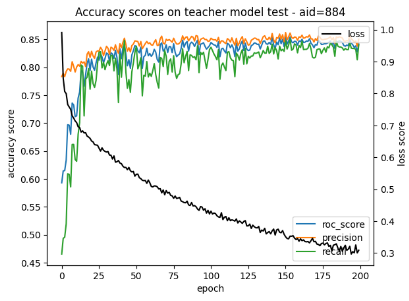
(a) Student - 50% dropout



(b) Original teacher - 50% dropout



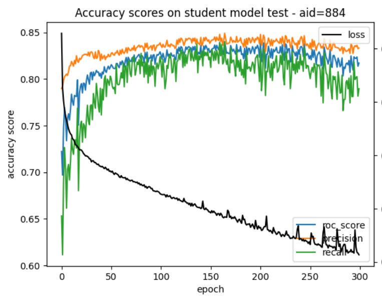
(c) Student - 80% dropout



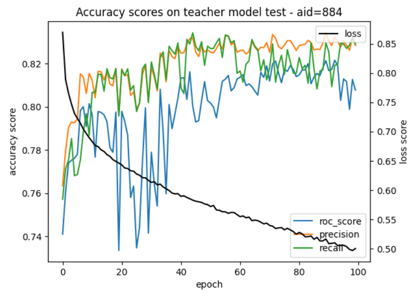
(d) Original teacher - 80% dropout

Figure 15: Metric histories for overfitting theory - test 1.

results of the student with normal layers using the self distillation data produced by a possibly underfitted teacher model are not higher than the accuracy scores seen in previous depictions. The metric history of the underfitted teacher and the described student model can be seen in figure 16 for an exemplary experiment(aid). As can be seen in this figure, the metric history for the teacher model is not as smooth and does not reach accuracy scores close to the original teachers scores, however it should nicely show that this model is hence not subject to overfitting.
As for the last test, the result from this is crystal clear - even after 3000 epochs for the one experiment for which is was run (experiment with id 884 was chosen for this due to its continuous good behavior in the previous tests). This can be seen in figure 17 together with the performance of a student training on the generated labels of this teacher showing no signs of worsening although this model should have been heavily overfitted at this point. Hence the theory that



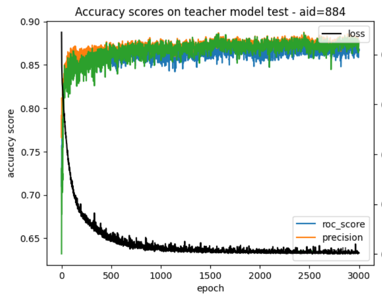
(a) Student - 50% dropout



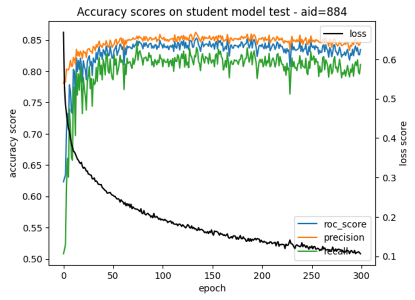
(b) Original teacher - 50% dropout

Figure 16: Metric histories for overfitting theory - test 2.

GNNs in this case are hard to overfit was shown to be correct.



(a) Teacher history - dropout 50%



(b) Student history - dropout 80%

Figure 17: Metric histories for overfitting theory - test 2.

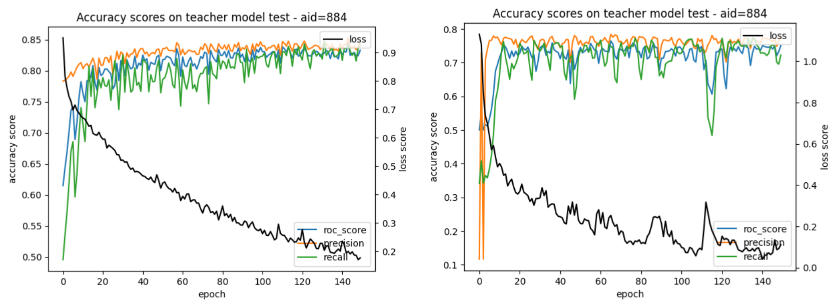
In conclusion the theory that overfitting either does not take place or does not affect the student's training has been shown to be correct through the three conducted tests.

4.17 Side theory: Dataset completeness

Theory Another possibility why self distillation would not bring any benefit in this projects setting may be due to something that could be labeled as a too 'complete' dataset. What is meant with this is that one reason why self distillation is working so well in other projects like [16] and [10] may be because the training benefits from having more datapoints. These datapoints are then artificially constructed with self distillation and thus increase the performance. The theory in this section hence is that the reason self distillation is having no effect in this project is due to the fact that the setup does not benefit from adding further data points.

Experimental Setup An equivalent formulation to the theory in the previous section is taking the statement that adding elements to the dataset must be beneficial and turns it into the counterpart: If adding elements should increase the prediction performance then removing elements should worsen the measured metric scores.
Hence it will be attempted to decrease the number of elements in the training set to see if the performance decreases.

Experimental Results In the first test the number of elements in the training set are reduced to 50% while for the second test it was decreased to 5%. The results from these two tests show that the theory is correct as with halving the element set the performance could only be decreased by 3(for aid 884) percentage points which is significantly less than what would be expected of removing half of the training elements.
When looking at the results of test 2 it becomes even more clearer, in this test only one twentieth of the training data remained and it still managed to produce a ROC AUC score of 0.75 (for aid 884) which is only 10 percentage points less than the teacher result although such a tremendous amount of data has been removed.
In conclusion the theory has been proven to be correct, an increase/decrease of data points is likely to have almost no effect on the results of the data, only removing a significant proportion managed to notably decrease the metric scores. The results for one experiments can be seen in figure 18.



(a) Teacher history - decreased training set to 50%

(b) Teacher history - decreased training set to 5%

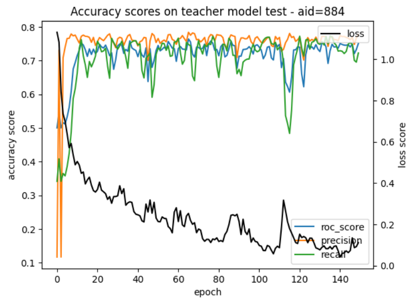
Figure 18: Metric histories for dataset completeness theory.

4.18 Side theory: Dataset rebuilding

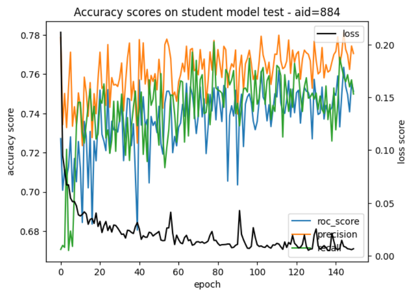
Theory Now as a complementary theory to the previous theory on dataset completeness, the theory is that now that we have a reduced dataset version that self distillation could work on when rebuilding the reduced dataset with self distillation data. The theory is that it should work in accordance to the previous theory although there might be issues with adding 20 fold of the (reduced) training data.

Experimental Setup For this setup the self distillation element prediction from the reduced version of 5% is taken and the number of elements to choose is the number of previously cut off data points. The dropout for this test is 50% in both teacher and student to get an accurate comparison without the influence of dropout. The selection method for self distillation elements is set to selecting the most confident points but keeping the class balance.

Experimental Results The experimental results are not exactly clear in this test. When comparing the results like in figure 19 it can be seen that the results are very similar to each other (neglecting the difference in y-axis scaling, which is done automatically by the Python library `matplotlib`). The results are almost identical with the self distillation method learning faster than the teacher, which is no surprise as the student model has significantly more data to train on and hence more loss correction calls in an epoch. Although it seems that the metric scores fluctuate more in this graph, this is simply due to the automatic axis scaling. In fact the student is more stable in this regard as when comparing the number of times the teacher and student produce metric scores below 0.7% it is clear that the student is more stable in learning behavior.
While this is only a small success because there is no notable improvement of the metric scores at the end it still signifies that self distillation in this case is actually beneficial. Hence the theory is not proven but also not completely proven to be false either. There are still multiple reasons as to why the results may not have increased in this setup containing the previous theories on the full dataset and the fact that using 20 times the elements for self distillation may be too much in this case.



(a) Teacher history - decreased training set to 5%



(b) Student history - rebuilt 5% dataset

Figure 19: Metric histories for dataset rebuilding theory.

4.19 Extra: Advanced accuracy measuring

General The idea of this test is to capture more accuracy scores of the teacher(and student) model to determine whether the prediction process is valid or not. While the results and description on this test are at this point in the report, this test was conducted relatively early to ascertain that the GNN is working properly.
For this, several scores are recorded for both train and test set:

- ROC AUC
- precision
- recall
- MCC
- loss (only for training set)

Results The results, as mentioned in the introductory section of this extra, show that the learning process appears to be valid for the models. The models exhibit overfitting to some degree as the training and test accuracy scores of the train and test set deviate although it never reaches the point where the test accuracy would fall because of overfitting.
Besides this, what is interesting to see is that some models, although exhibiting great 'normal' accuracy scores, the MCC score is quite low (below 0.4) indicating that training for this specific experiment the correlation between observed and true labels is quite low and that the results may not be meaningful while other results above 0.6 can be considered strongly correlated showing the results are meaningful and useful classification takes place.
A few results can be seen in figure 20; please note that these diagrams contain all recorded information and are not trivial to read. To see the data in more detail or to create an alternative graph on the recorded data, read more in section A on how to acquire the measured data/notebooks and create a modified version.

4.20 Extra: Sanity check

General With so many results of the student model using self distillation that are equal to the teachers performance naturally the question arises whether or not self distillation works or if something is wrong with the implementation. To check whether this really is the case, a test is created in which the labels of the self distillation data are inverted, in the sense that a 0 becomes a 1 and vice versa.

Results The results depicted in figure 21 clearly show a drop of metric scores meaning that in this case there is a negative influence meaning that the initial

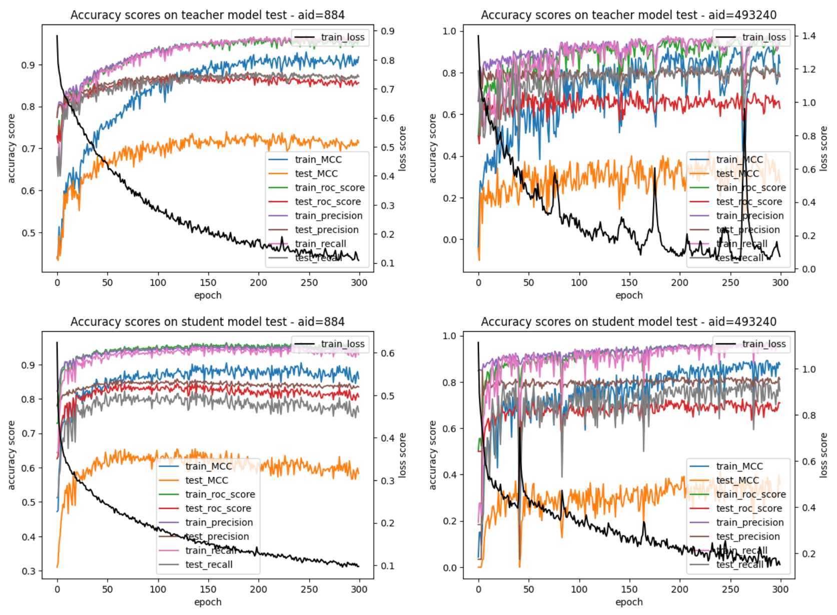


Figure 20: Advanced metric history for extra on accuracy measuring

doubt on the correct implementation(of self distillation) can be safely disregarded. What is interesting is that in some cases although the ROC AUC has significantly dropped, the recall and precision values remain high.

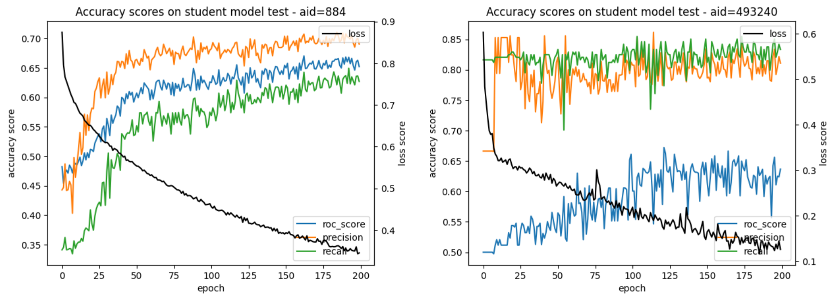


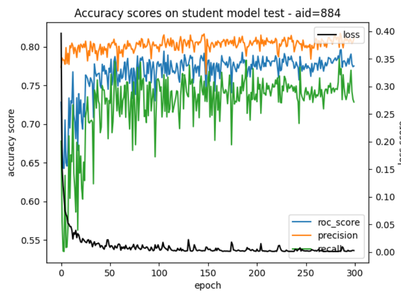
Figure 21: Metric history of extra - sanity check, inverting the self distillation element's labels

4.21 Extra: Data replacement

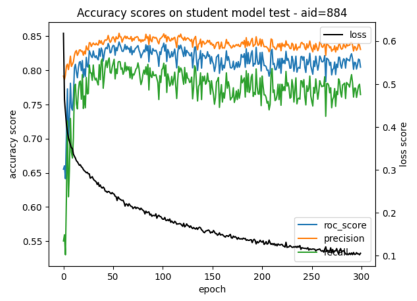
General Self distillation, as described in the previous theory sections, has been shown to work as intended, however the question arises whether generated self distillation data could be useful to generate artificial data in general in the

sense of if it can replace the original training set and produce similar results as the original setup.
For this setup the 50% dropout teacher self distillation data is used with an 80% dropout student model with keep ratio selection method and smooth label input.

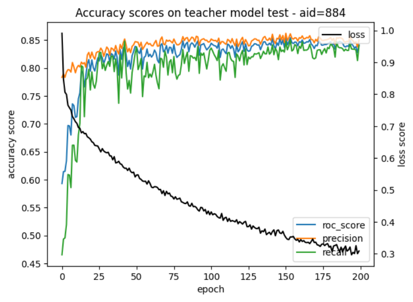
Results The results were quite good, although the scores did not reach the comparison scores of the student model/teacher model, the results were quite close with an observed difference of 0.05 = 5 percentage to 0.1 = 10 points. Considering the not perfect test accuracy this result is quite satisfactory. An example can be seen in figure 22. Of course this has not been the case for all chemical experiments, in some other cases the substitution does not work at all, yielding significantly worse results with unstable learning behavior.



(a) Student with exchanged test set (to self distillation data)



(b) Student model with regular test set consisting of original data and self distillation in proportion 1-1



(c) Teacher model with 80% dropout

Figure 22: Metric history of extra - exchange/replacement of training set

4.22 Main theory: Self distillation on graph neural network

Theory As mentioned in the beginning of the experimental evaluations section the major goal and theory is to ascertain if self distillation works on GNNs (in this particular problem setting). The initial assumption contained that self distillation would enhance the performance of the graph neural network based on the OGB model for the chosen data setting (chemical activity of molecules in several experimental context).

Experimental Setup To conduct the tests several smaller tests and side theories have been created, as can be seen in the previous sections. The reason for this is that it is not sufficient to execute one test and afterwards state that this machine learning method is not applicable to this model and setting simply because of one failed test. Hence in the course of the previous 'experimental evaluation' sections different methods, settings and approaches have been tested in the name of finding a setting where self distillation may use its hidden potential - or to find out why it does not work in this setting.

Experimental Results The experiments in the previous sections clearly show that it has not been possible to verify a positive effect of self distillation on this dataset and model. In fact it took some serious effort to get the model using self distillation to work as good as the model without self distillation.
After getting it to work different selection methods and label types for the self distillation elements were attempted in addition to attempts to use class rebalancing to make the self distillation aid the learning process from a different angle. The displayed efforts in this direction were not rewarded, even after attempting methods to explore the effects of the dropout ratio, the self distillation set size, the student model 'size' and overfitting, the results remained unchanged with self distillation being not beneficial in most way other than increasing the stability of the learning process slightly.
After ascertaining the previously described facts, it was clear that self distillation with the approaches chosen for this model would not be of success, which is why a reason for this behavior became the focus of research. After determining that, as mentioned, overfitting and the self distillation set size (besides the other attempted 'environmental settings') were most probably not the reason the direction of analysis shifted towards the model and data part of the problem.
However, as determining whether self distillation is applicable or not on GNNs in general is not possible in this project, the main focus fell on the dataset. Please note that it is not determinable if it is not working for graph neural networks in general because to do so it would be required to test for different datasets, models and settings which easily exceeds the capacity of this project.
When analyzing the data, the suspicion arose whether the dataset is maybe already complete or satisfied in the context that adding more elements to the training would not add any benefit. While this theory was confirmed, testing

self distillation on a subset of the dataset (for which the performance is significantly below the full dataset) it was still not possible to produce an effect besides improving learning stability.
In conclusion the initial theory was proven to be incorrect. It has not been possible to apply self distillation in a metric score improving manner for this GNN model with the chosen data set. The seen result also heavily suggests that it is not only not improvable with the attempted methods of this project, but also that it is not possible with the model and data combination at all - the data seems to be unable to benefit from this technique. Whether or not it is not possible with this or all GNN models in general was not determinable.
It can also be speculated whether the data is really the problem in this setting. Of course it might be the case that the data is saturated as it is and cannot be improved by adding data or that the target label is too simple, however it is also a viable option that self distillation itself may not work on GNNs that are not deep at all. Of course, as already mentioned, this cannot be confirmed by a single experiment, however it would make perfect sense as self distillation was used in deep neural networks in [16] and [10].

5 Conclusion

5.1 Final Analysis

In the previous experiments is has been determined that self distillation is in general not possible in Random Forests as it yields significantly worse results than using no self distillation at all which was expected because of the lack of trainable parameters using a technique related to error correction (loss).
It has also been determined that the mentioned machine learning technique is not successfully applicable to an established and well working graph neural network model for a selected dataset although multiple different methods, techniques and settings have been attempted. Further tests and evaluations have conducted that the dataset chosen for the evaluation is likely to be the problem causing self distillation to be without significant effect. Such problems related to the dataset include completeness - which is due to the fact that the model is able to make reasonable predictions with a fraction of the data without incurring a heavy metric score drop -, lack of complexity - which is due to the nature of the target label being binary and the real rules behind the correlation between label and data being too simple for this learning approach - or incompatibility of model and dataset, although the latter is unlikely due to the high metric scores measured throughout the experiment.
It is certainly also possible that self distillation is not able to perform well on any graph neural network in general, although this is not assumed to be the case. This is because this technique was successful in neural networks[9] and graph neural networks are 'simply' an extension that uses graph structure in a neural network learning context to make more information usable for training.

[9]See [16] and [10]

Of course, this mentioned extension can certainly be the issue preventing the success of the aforementioned method, however this is doubtful except if self distillation requires a specific byproduct/behavior of neural networks to be successful.
Another theory concerning the model used in this project is that, in order to get a benefit with this method, it requires a deep neural network in order to function properly. This is supported by the fact that previous successful implementations like [16], as 'ResNet', the used model there, has a large amount of layers, or Alpha Fold[10] which can also be regarded as a deep neural network. Hence it is a possibility that a deep graph neural network would be required to make self distillation applicable - although deep graph neural networks as of now are seemingly not yet firmly established and still an ongoing point of research.

5.2 Concluding Remarks

Graph Neural Networks are relatively new and a rapidly growing field with still undiscovered potential. Using graph level prediction on molecular structures is interesting and in this case beneficial with satisfying metric scores and are worth to be investigated on its own. In combination with self distillation, if working successfully, it would have been even more rewarding, although it was nice to achieve good results with graph level prediction on this dataset in the first place.
As the one writing this having also executed the various tests and experiments, it is likely that there are certain tests, settings or features that have not been explored in this project and that there may also be better configurations and ideas on how to apply this technique. This is why an interested reader is welcome to use the code, results and ideas from this project (for access see section A) in other projects to run separate experiments and potentially improve the results shown in this report.
While using said method on graph neural networks was not beneficial in this case, it might prove to be useful in another problem setting. Even if this is not the case there are certainly other methods for enhancing prediction performance for graph neural networks which have yet to be uncovered, applied and analyzed.

5.3 Outlook

As said, there are many settings and methods not tested and experimented with in this project due to time and complexity constraints. Some further research topics include using a larger soft label representation to transfer the teacher training state more accurately to the student model, using other graph neural network components/structure and using other datasets to determine the superior theory whether GNNs are possible to enhance using self distillation, maybe even using a deep graph neural network.
Another certainly interesting research topic could be to use other definitions of self distillation like introduced in [18] and section 2.4.2 on graph neural networks,

maybe even in combination with this projects definition of self distillation to build a powerful machine learning module based on graph structured data.

References

[1] Allen-Zhu, Z., and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning.

[2] Breiman, L. Random forests. *Machine learning 45* (2001), 5–32.

[3] Cao, C., Chicco, D., and Hoffman, M. M. The mcc-f1 curve: a performance evaluation technique for binary classification.

[4] Chen, H., Perozzi, B., Al-Rfou, R., and Skiena, S. A tutorial on network embeddings.

[5] Cutler, A., Cutler, D. R., and Stevens, J. R. Random forests. *Ensemble machine learning: Methods and applications* (2012), 157–175.

[6] Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters 27*, 8 (jun 2006), 861–874.

[7] Fey, M., and Lenssen, J. E. Fast Graph Representation Learning with PyTorch Geometric, May 2019.

[8] Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., and Leskovec, J. Ogb-lsc: A large-scale challenge for machine learning on graphs.

[9] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs.

[10] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature 596*, 7873 (2021), 583–589.

[11] Lawrence, J. *Introduction to neural networks.* California Scientific Software, 1993.

[12] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32* (2019), H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., pp. 8024–8035.

[13] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine

learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[14] Prechelt, L. Early stopping-but when? In *Neural Networks: Tricks of the trade*. Springer, 2002, pp. 55–69.

[15] Shu, J., Xi, B., Li, Y., Wu, F., Kamhoua, C., and Ma, J. Understanding dropout for graph neural networks. In *Companion Proceedings of the Web Conference 2022* (apr 2022), ACM.

[16] Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. Self-training with noisy student improves imagenet classification, 2019.

[17] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks?

[18] Zhang, L., Song, J., Gao, A., Chen, J., Bao, C., and Ma, K. Be your own teacher: Improve the performance of convolutional neural networks via self distillation.

[19] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications.

[20] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *AI open 1* (2020), 57–81.

List of Figures

1 ROC AUC scores of Random Forest for a selection of experiments (see section 4.4) 22
2 Exemplary metric history for basic teacher experiment 23
3 Comparison between concat approach and normal teacher 25
4 Comparison between concat approach and normal teacher 26
5 Metric histories for initial theory experiments. 27
6 Metric histories for unstable student fixing theory. 29
7 Metric histories for self distillation element selection theory - example 1. 31
8 Metric histories for self distillation element selection theory - example 2. 32
9 Metric histories for label type theory. 33
10 Metric histories for class balance theory. 34
11 Metric histories for dropout check theory - 1. 35
12 Metric histories for dropout check theory - 2. 36
13 Metric histories for self distillation set size theory. 37
14 Metric histories for enlarge student model theory. 38
15 Metric histories for overfitting theory - test 1. 39
16 Metric histories for overfitting theory - test 2. 40
17 Metric histories for overfitting theory - test 2. 40
18 Metric histories for dataset completeness theory. 41
19 Metric histories for dataset rebuilding theory. 42
20 Advanced metric history for extra on accuracy measuring 44
21 Metric history of extra - sanity check, inverting the self distillation element's labels 44
22 Metric history of extra - exchange/replacement of training set . . 45

Acronyms

FN False Negative. 13–15

FP False Positive. 13, 14

GINConv Graph Isomorphism Network Convolution (layer). 8, 16, 38

GNN Graph Neural Network. 5, 7, 8, 10, 11, 15, 19–21, 23, 38–40, 43, 46–48

MCC Matthews Correlation Coefficient. 14, 21, 43

MLP Multi-Layer Perceptron. 8, 9

OGB Open Graph Benchmark. 4, 15, 16, 23, 24, 35, 46

RF Random Forest. 10, 17, 20, 21, 28

ROC AUC Receiver Operation Characteristics Area Under the Curve. 14, 17, 21–23, 30, 34, 41, 43, 44

SD Self Distillation. 15, 19

SGD Stochastic Gradient Descend. 20

TN True Negative. 13, 14

TP True Positive. 13–15

A Experimental results on GitHub

A.1 GitHub repository

The code for this project and the results can be found on a GitHub repository available via [10].

In this repository the majority of the notebooks can be found in the folder 'notebooks' except for the first few notebooks of GNN related notebooks which contain the experiments until the fist well working version was established and the Random Forest experiments notebook(s). The folder 'results' contains all the saved measurements, self distillation elements (teacher labeled molecules) and the graphs shown in this report. The folder 'selfdist_toolkit' contains the code containing functionality aiding the notebooks in their experiments.

A.2 Environment setup

It is advised to use a conda environment to set up the python libraries as the installation of pytorch using cuda can be troublesome. Regretfully the 'requirements.txt' and the 'environment.yml' are not able to replicate the environment successfully which is why there is an installation description in the ReadMe and in a text file called 'installation-procedures.txt'.

A.3 Dataset fetching

The original dataset is too large for GitHub which is why the dataset needs to be fetched from `https://ucloud.univie.ac.at/index.php/s/XcnZ8q13sqQgraT`. Place the csv inside of a folder called 'data'. Theoretically it would be better to download the zip data file and extract it so that a folder called 'data' with 6 files and one folder 'experiment-wise' are present. These files are the pre-modified and split data entries so that the splits for each experiment do not need to be generated by the user.

If you only have access to the csv file, place it as mentioned and then use the function `experiment_whole_preprocess()` from the folder-path 'selfdist_toolkit/data_tools/preprocessing.py' to split the notebook and to create the chemical descriptor data and morgan fingerprint for the random forest tests. If you only wish to preprocess the data for the gnn related tests execute the function `experiments_preprocess()` instead.

The total preprocessing can also be found in the notebook for the random forests tests 'Random_Forest.ipynb', where the preprocessing is done for the whole project, this step is not repeated in the other notebooks. Note that this may take some time and the generation of the chemical descriptor data and morgan fingerprint is parallelized with the setting to use all available cores to speed up the computation.

In case not the folder is downloaded it is also required to run the 'Random_Forest.ipynb' and 'RF_exp-analysis.ipynb' notebook to generate the

[10] `https://github.com/JellyJoe13/Self-distillation_P1.git`

list of good notebooks to use in the gnn tests.
The dataset fetching procedure is also described in the ReadMe in detail.

A.4 Notebooks and Result list (sorted by notebook occurrence on GitHub)

In the following list the names of available notebooks are listed and the content of the notebook briefly described. In most cases the naming convention of the result folder is equal or similar, if this is not the case either the folder name is written or in the worst case the folder name can be seen in the corresponding notebook in a cell in the section 'Setting up storage location'.

1. development_GNN.ipynb - Notebook in which development and intermediary training steps can be seen
2. Random_Forest.ipynb - Notebook containing the random forest experiments
3. RF_exp-analysis.ipynb - Notebook containing the analysis which determines which experiments to consider for the elaborate gnn tests
4. SD_analysis.ipynb - some analysis on self distillation (if it works and how the class balance looks like)
5. GNN_test_teacher...
   (a) .ipynb - default teacher implementation with 50% dropout.
   (b) -concat.ipynb - teacher implementation with layer-wise node embedding concatenation as a mean of stochastic depth
   (c) _ACC.ipynb - teacher run with advanced accuracy metric collection
   (d) _conf56_neg.ipynb - teacher run with dropout 0.8
   (e) _data-reduced.ipynb - 50% of training set used
   (f) _data-reduced2.ipynb - 5% of training set used
   (g) _early-stop.ipynb - early stopping attempt - discarded as not quite useful
   (h) _ep-300.ipynb - run teacher for 300 epochs
   (i) _inferior.ipynb - run teacher with only 1 convolution layer
   (j) _no-dropout - run teacher with 0 dropout
   (k) _ulong - run teacher for 3000 epochs
6. GNN_test_self-dist_...
   (a) first.ipynb - student, soft label, confidence first - unshuffled dataset, dropout 0.5

(b) second.ipynb - student, soft label, keep class balance - unshuffled dataset, dropout 0.5

(c) third.ipynb - student, hard label, confidence first - unshuffled dataset, dropout 0.5

(d) fourth.ipynb - student, hard label, keep class balance - unshuffled dataset, dropout 0.5

(e) conf5.ipynb - student, soft label, confidence first - shuffled dataset, dropout 0.8

(f) conf6.ipynb - student, soft label, keep class balance - shuffled dataset, dropout 0.8

(g) conf7.ipynb - student, soft label, correct ratio - shuffled dataset, dropout 0.8

(h) conf8.ipynb - student, soft label, keep class balance - shuffled dataset, dropout 0.8, self distillation ratio set to 2

(i) conf9.ipynb - student, soft label, keep class balance - shuffled dataset, dropout 0.8, self distillation ratio set to 4

(j) conf10.ipynb - student, soft label, keep class balance - shuffled dataset, dropout 0.8, embedding 300-¿600

(k) conf11.ipynb - student, soft label, keep class balance - shuffled dataset, dropout 0.8, layer count 5-¿7

(l) conf12.ipynb - student, soft label, keep class balance - shuffled dataset, dropout 0.8, self distillation ratio set to 0.5

(m) conf-1.ipynb - student, soft label, random selection - shuffled dataset, dropout 0.8

(n) conf-2.ipynb - student, hard label, random selection - shuffled dataset, dropout 0.8

(o) conf-5.ipynb - student, soft label, confidence first - shuffled dataset, dropout 0.5

(p) conf-inferior-1.ipynb - student, soft label, keep class balance - shuffled dataset, dropout 0.8 - one convolution layer

(q) conf_reduced2-rebuild.ipynb - student, soft label, keep class balance - shuffled dataset, dropout 0.8 - 5% of train data-95% self distillation data

(r) confACC.ipynb - student, hard label, keep class balance - shuffled dataset, dropout 0.8, advanced accuracy measurement

(s) exchange.ipynb - student, soft label, keep class balance - shuffled dataset, dropout 0.8 - use self distillation data instead of train set

(t) negative.ipynb - student, soft label, keep class balance - shuffled dataset, dropout 0.8 - inverted self distillation labels

(u) teacher-no-dropout_student-dropout-50 - student, soft label, keep class balance - shuffled dataset, dropout 0.5 - test influence teacher dropout 0

(v) teacher-no-dropout_student-dropout-80 - student, soft label, keep class balance - shuffled dataset, dropout 0.8 - test influence teacher dropout 0

(w) ulong.ipynb - student, soft label, keep class balance - shuffled dataset, dropout 0.8 - self distillation from 3000 epochs teacher

A.5 Toolbox documentation

The code in the folder 'selfdist_toolkit' has been documented using docstring in NumPy format which was used incombination with the `sphinx`[11] library to generate a documentation for the auxiliary routines for the experiments. This can be seen in the folder 'docs' via the 'index.html' or directly via `https://jellyjoe13.github.io/Self-distillation_P1/` where this documentation is hosted by GitHub pages.

[11]`https://www.sphinx-doc.org/en/master/`