Distilling Influences to Mitigate Prediction Churn in Graph Neural Networks

Andreas Roth   ANDREAS.ROTH@TU-DORTMUND.DE
TU Dortmund University, Dortmund, Germany

Thomas Liebig   THOMAS.LIEBIG@TU-DORTMUND.DE
TU Dortmund University, Dortmund, Germany
Lamarr Institute for Machine Learning and Artificial Intelligence, Germany

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Models with similar performances exhibit significant disagreement in the predictions of individual samples, referred to as prediction churn. Our work explores this phenomenon in graph neural networks by investigating differences between models differing only in their initializations in their utilized features for predictions. We propose a novel metric called Influence Difference (ID) to quantify the variation in reasons used by nodes across models by comparing their influence distribution. Additionally, we consider the differences between nodes with a stable and an unstable prediction, positing that both equally utilize different reasons and thus provide a meaningful gradient signal to closely match two models even when the predictions for nodes are similar. Based on our analysis, we propose to minimize this ID in Knowledge Distillation, a domain where a new model should closely match an established one. As an efficient approximation, we introduce DropDistillation (DD) that matches the output for a graph perturbed by edge deletions. Our empirical evaluation of six benchmark datasets for node classification validates the differences in utilized features. DD outperforms previous methods regarding prediction stability and overall performance in all considered Knowledge Distillation experiments.

Keywords: Graph Neural Networks; Knowledge Distillation; Churn; Instability

1. Introduction

Neural networks have achieved remarkable success across various domains (Vaswani et al., 2017; Jumper et al., 2021; Roth and Liebig, 2022a), but their predictions often lack reliability and satisfactory explanations, leading to low trust (Samek et al., 2021). One observed issue contributing to this problem is prediction churn, where models with similar performance exhibit significant variability in their predictions (Summers and Dinneen, 2021; Klabunde and Lemmerich, 2022). This churn occurs even among models with identical hyperparameters differing only in their random initializations (Bhojanapalli et al., 2021; Zhuang et al., 2022). When large portions of correctly classified data are misclassified upon model retraining, the models' reliability, trustworthiness, and explainability are reduced. Churn has been observed across various domains and data structures, but understanding its underlying cause remains challenging. This work focuses on node classification with graph neural networks (GNNs), a domain where churn is particularly prevalent (Schumacher et al., 2022; Klabunde and Lemmerich, 2022). Our work investigates the reason behind prediction churn

in GNNs by comparing the influence of context nodes on predictions. We propose a novel metric, the Influence Difference (ID), which allows us to compare the exploited features for a given prediction between a pair of models and empirically verify this prevalence. Contrary to previous investigations, we hypothesize that stable and unstable predictions exhibit similar differences in their utilized features. We further hypothesize that stable nodes possess redundant features, allowing their stable prediction even when the features utilized for the prediction change. To empirically validate our hypotheses, we introduce additional metrics based on ID.

Knowledge Distillation (KD) is a promising technique to transfer knowledge from a well-performing teacher model to a newly trained student (Buciluă et al., 2006; Li et al., 2014; Hinton et al., 2015) This enables model compression for a computationally expensive teacher or regular model updates using new data (Gou et al., 2021). Users expect consistent behavior after each update, so closely matching the teacher's predictions is crucial. Previous work formulated the goal of KD as directly minimizing churn (Jiang et al., 2022). Based on our findings, we propose to extend KD by also matching the influences of predictions as a regularization. As the exact formulation is computationally prohibitive, we introduce DropDistillation (DD), an efficient approximation that mimics the influence of adjacent nodes by removing random edges equally from both the teacher and the student model. Our empirical analysis validates our hypotheses for several benchmark datasets, further motivating the need for transferring the reasons. Comparing DropDistillation with several state-of-the-art methods, our approach improves prediction churn between teachers and students and overall performance. We summarize our key contributions:

- We investigate the reason behind prediction churn in GNNs by comparing the influence of context nodes on predictions using a novel metric. We also connect differences in prediction stability with the availability of redundant features (Section 3).

- Based on our findings, we extend knowledge distillation to minimize churn directly and match the features exploited for predictions as a regularization technique and propose an efficient approximation, namely DropDistillation (DD) (Section 4).

- Our empirical evaluation validates our claims and confirms the effectiveness of DropDistillation for Knowledge Distillation for various benchmark datasets (Section 5).

These contributions aim to enhance our understanding of prediction churn in GNNs, and propose a novel direction to address churn in knowledge distillation, resulting in increased reliability and overall performance.

2. Preliminaries

We start by introducing basic notations and concepts we use throughout this work. Let $G = (V, E)$ be a graph containing a set of n nodes $V = \{v_1, \ldots, v_n\}$ and a set of edges E indicating the connectivity between pairs of nodes. We also express the set of edges as an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ of pairwise connections that may additionally weigh nodes differently. We consider the task of node classification where training data comes either from other graphs (inductive case) or labels available for a subset of the nodes $V_t \subset V$ (transductive case). Our work investigates graph neural networks (GNNs) (Kipf and Welling, 2016),

though only a superficial understanding is needed. GNNs represent a learnable function $f : \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times c}$, mapping a graph signal and an adjacency structure to output logits $f(\mathbf{X}, \mathbf{A}) = \mathbf{C}$ for c classes and each node. $\mathbf{Y} \in \mathbb{R}^{n \times c}$ denotes the true label matrix.

2.1. Prediction Churn

Prediction churn (Goh et al., 2016; Milani Fard et al., 2016) describes a phenomenon in which models make different predictions on the same data points. Formally, we define churn for any two functions f, g for node classification mapping the d-dimensional graph signal $\mathbf{X} \in \mathbb{R}^{n \times d}$ and the adjacency matrix $\mathbf{A}$ to class probabilities for every node. We define the set of unstable nodes by

$$\mathcal{U}_{f,g} = \left\{ v_i \in V_{\text{test}} \mid \arg\max_{c \in C} f(\mathbf{X}, \mathbf{A})_{ic} \neq \arg\max_{c \in C} g(\mathbf{X}, \mathbf{A})_{ic} \right\}. \tag{1}$$

We further define $\mathbf{s}_{f,g} \in [0,1]^n$ to be the binary vector indicating with a one whether each node was predicted stable between f and g. The pairwise churn is then defined as the ratio of unstable nodes over the total number of test nodes, as given by

$$C(f, g) = \frac{|\mathcal{U}_{f,g}|}{|\mathcal{V}_{\text{test}}|} = \frac{1}{n} \sum_{v_i \in \mathcal{V}_{\text{test}}} \mathbb{1}_{\{v_i \in \mathcal{U}_{f,g}\}}. \tag{2}$$

Here, $\mathbb{1}$ denotes the indicator function, that is 1 if the condition is satisfied and 0 otherwise. Churn is undesirable for many reasons, including the reproducibility of scientific results, reliability, and trust in machine learning models (Jiang et al., 2022; Liu et al., 2022). Especially when continuously delivering updated models, the experience should be stable and consistent (Goh et al., 2016).

Churn occurs in various scenarios, such as using different model architectures or hyperparameters. But even when these are the same, churn occurs even when the initial parameters are slightly altered (Bhojanapalli et al., 2021; Zhuang et al., 2022). Non-determinism of GPU operations also produces churn even when all initial parameters are the same (Summers and Dinneen, 2021). For graph-structured data, several studies observed an instability of node embeddings that holds for stochastic operations like dropout (Wang et al., 2020; Klabunde and Lemmerich, 2022). However, the reason behind churn and the inability of models optimized on the same data to develop similar decision rules remains unclear.

2.2. Knowledge Distillation

Knowledge Distillation (KD) (Buciluă et al., 2006; Li et al., 2014; Hinton et al., 2015) is one particular domain of interest for reducing churn. Here, the goal is to distill the knowledge of a pre-trained large model, called the teacher T, into a smaller model, called the student S. In our case, both S and T are functions that perform node classification. This is typically used to compress the knowledge for resource efficiency during inference (Cheng et al., 2018) but also to optimize iteratively updated models for continuous deployment (Jiang et al., 2022). In general, a distillation loss $\mathcal{L}_{\text{distill}}(S, T)$ is used to match the output or intermediate representations of T and S (Li et al., 2014; Hinton et al., 2015) in addition to the original classification loss. Prediction churn and Knowledge Distillation are inherently connected, as they share a common goal. Pairs of models should produce similar predictions, so in a perfect scenario, the churn between student and teacher would be zero.
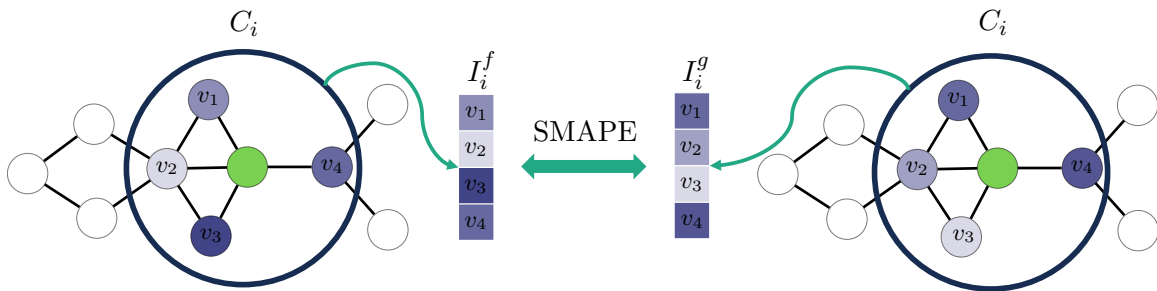
Figure 1: Influence Difference for the green root node between two models f (left) and g (right). Context nodes C_i are encircled, and the different shades of blue represent its nodes' value in the influence distribution. Symmetric Mean Absolute Percentage Error (SMAPE) between the two influence vectors is calculated.

2.3. Related Work

Few investigations connected Knowledge Distillation and prediction churn. Jiang et al. (2022) continuously optimize new models and introduce a churn constraint between consecutive instantiations. They showed that matching the outputs is equivalent to directly minimizing the prediction churn between S and T, though they require strong assumptions on the generalization bounds. Bhojanapalli et al. (2021) propose a co-distillation approach that optimizes two models simultaneously while matching their outputs. Other methods similarly propose to reduce churn solely based on the outputs (Milani Fard et al., 2016; Summers and Dinneen, 2021). Our investigation also builds on ideas from quantifying the similarity of neural networks (Lenc and Vedaldi, 2015; Klabunde et al., 2023). In particular, Jones et al. (2022) determine the similarity between two models using vectorized saliency maps. Allen-Zhu and Li (2023) propose a theory in which models would learn different subsets of the available features but not all of them, as they are not required to correctly classify a large part of the data. We extend these approaches to the specific properties of graph-structured data and relate them to prediction churn.

3. Comparing Differences in Influence of Predictions

We now delve into the underlying reason behind prediction churn in the context of node classification with graph neural networks. Given a single node, we want to investigate what makes its prediction stable or unstable across models. We build our investigations on recent progress in determining the similarity between neural networks (Allen-Zhu and Li, 2023). It was found that comparing the outputs or representations of neural networks alone overestimates their similarity due to the correlation between data points (Jones et al., 2022). In our case, churn may be insufficient in determining the actual similarity between pairs of models. A recent theory indicates that models may only learn a subset of the available features from the data depending on their parameter initialization. Confirming these ideas would provide insights into the causes of prediction churn. We incorporate these ideas and propose the following statement that we will empirically investigate later:

Axiom 1 (A1) *Prediction churn is a consequence of models utilizing different features for their predictions.*

Thus, even when two models have low prediction churn, they may contain different knowledge. Depending on which features a model utilizes, the predicted class may change. Consequently, our objective is to quantify the difference between features used by nodes to make specific predictions between a pair of models. We also aim to investigate the distinguishing factors between stable and unstable nodes in terms of their utilized features. Analyzing this behavior would allow us to construct better-informed methods to mitigate the resulting churn and improve the overall reliability of node classification, e.g., in Knowledge Distillation. Building on these insights, we now propose metrics to quantify the difference in reasoning between pairs of models for their node predictions.

3.1. Comparing the Reasons behind Predictions

Instead of comparing the output differences for a pair of models, we propose to compare which features each model utilized for its predictions directly. Our data's graph structure and the node classification task pose a challenge. Given the prediction for a root node, we want to determine its influencing features and compare this between models. As all neighboring nodes can affect the prediction to some degree, we propose to view each neighboring node as one potential feature the root node can utilize. When a root node relies on different neighbors for distinct models, this indicates that models learn disjoint feature subsets and would thus be more meaningful than churn C.

Formally, we want to determine the importance of an initial node signal $\mathbf{x}_j$ at node v_j on the extracted representation $\mathbf{z}_i$ of node v_i. As the gradient $\frac{\partial z_{ia}}{\partial x_{jb}}$ represents the sensitivity of the a-th feature of node i's representations z_{ia} to a variable x_{jb}, our requirements are met by the well-established influence scores (Xu et al., 2018)

$$I(i,j) = \sum_{a=1}^{d'} \sum_{b=1}^{d} \left| \frac{\partial z_{ia}}{\partial x_{jb}} \right| . \tag{3}$$

for node v_j on node v_i. These sum the gradient magnitudes of each logit output z_{ia} to each input feature x_{jb}. Following Xu et al. (2018), the influence distribution is then defined as the normalized influences

$$I_i(j) = \frac{I(i,j)}{\sum_{k \in C_i} I(i,k)} \tag{4}$$

over all context-nodes C_i of a root node v_i. We further denote the influence distribution of a given model f as $I_i^f(j)$. We now propose two novel metrics that compare the differences between models based on the influence distribution.

We propose a generic metric that compares the deviations between each value in the influence distributions $I_i^f(j)$ of model f and $I_i^g(j)$ of model g. Since the magnitude of each normalized score $I_i(j)$ depends on the number of context nodes $|C_i|$, we use a relative metric. In general, f and g are commutable, so the metric should also be symmetric. Thus, we use the symmetric mean absolute percentage error (SMAPE) (Chen and Yang, 2004),

which has a direct interpretation. We define the expected Influence Difference as

$$\text{ID}(f, g) = \mathbb{E}_{v_i \sim V, j \sim C_i} \left[\frac{|I_i^f(j) - I_i^g(j)|}{0.5 \cdot (|I_i^f(j)| + |I_i^g(j)|)} \right] \tag{5}$$

between a pair of functions f and g with shared domain and codomain. This metric should be small when both f and g rely on the same nodes and large when they rely on different nodes for their representations. Figure 1 provides a visualization of this metric for one root node. A key property of ID is that even when the prediction churn is small, it can still provide meaningful information about the differences between the knowledge acquired between the two models:

Proposition 1 *Given f, g. $C(f, g) = 0$ does not imply a low Influence Difference.*

Given these properties, we think ID provides more profound insights about which features models rely on and how stable that is between models.

3.2. Differences between Stable and Unstable Nodes

Our proposed metric, ID, allows us to determine the difference in influence globally over all nodes between a pair of models. Previous work based on the churn metric showed that only some nodes are predicted unstable, while others are predicted stable across many models (Klabunde and Lemmerich, 2022). However, if models learn different feature subsets, this would hold for all nodes, not only those with an unstable prediction. Thus, we make the following claim:

Axiom 2 (A2) *Stable and unstable nodes have a similar Influence Difference.*

To verify this statement empirically, we calculate the correlation between the average influence differences $\mathbf{id} \in \mathbb{R}^n$ for each node and the stability $\mathbf{s} \in \{0, 1\}^n$ of each node's prediction. We utilize Pearson's correlation coefficient

$$\text{corr}(\mathbf{id}, \mathbf{s}) = \frac{\text{cov}(\mathbf{id}, \mathbf{s})}{\sigma_{\mathbf{id}} \sigma_{\mathbf{s}}} \tag{6}$$

based on the covariance cov, and the standard deviations $\sigma_{\mathbf{id}}$ and $\sigma_{\mathbf{s}}$. A high correlation would indicate that stable nodes also have larger differences in influence between models. Thus, we expect them to be uncorrelated.

3.3. Feature Redundancy Stabilizes Predictions

The question remains what leads to the unstable prediction of some of the nodes. As outlined by Allen-Zhu and Li (2023) regarding model similarity, some elements may contain redundant features. For node classification, our interpretation is that stable nodes similarly have access to redundant features. Thus a similar Influence Difference has a smaller effect on the actual prediction. We propose the following statement:

Axiom 3 (A3) *Stable nodes have access to more redundant features.*

For verification, we need to determine the number of redundant features each node has available for their predictions. However, the discriminating features are not observable in the data, so we use an indicator that should closely correlate with the number of available features. We utilize the number of context nodes with the same label, as these should provide redundant signals. We let $D_i(y)$ be the ratio of each label $y \in c$ in the context C_i of node v_i. We calculate the entropy

$$H(i) = -\sum_{y \in c} \log(D_i(y))D_i(y) \tag{7}$$

of these label ratios for each node i. Low entropy corresponds to redundant views as the prediction could rely on features from different nodes. We denote the vector of label entropies for all nodes as $\mathbf{h} = (H(1), \ldots, H(n))^T$. To validate A3, we again calculate the correlation to the node stability vector $\mathbf{s}$ using Pearson's correlation coefficient

$$\text{corr}(\mathbf{h}, \mathbf{s}) = \frac{\text{cov}(\mathbf{h}, \mathbf{s})}{\sigma_{\mathbf{h}}\sigma_{\mathbf{s}}} \tag{8}$$

based on their covariance cov and respective standard deviations $\sigma_{\mathbf{h}}$ and $\sigma_{\mathbf{s}}$. A high correlation indicates stable nodes would have more variance in their neighboring labels. Thus, if A3 holds, we expect an anti-correlation.

4. Aligning the Influence in Knowledge Distillation

We now build on our claims to mitigate churn in Knowledge Distillation (KD). Here, we assume the teacher model T is given and exhibits desirable performance and reasonings for its predictions. Current state-of-the-art for reducing prediction churn in KD only matches the outputs or representations, e.g., by directly minimizing prediction churn (Jiang et al., 2022). Based on our claimed Axioms, this may not transfer the knowledge in T to S as different features can be exploited for the same predictions, hurting generalizability (see Proposition 1).

We propose minimizing ID to directly match the reasonings behind predictions and an improved knowledge transfer If we could achieve the student to mimic the reasoning of a high-capacity teacher, the performance, reliability, and generalization should be improved. Critically, based on Axiom 2, nodes with the same prediction still provide a strong gradient signal when optimizing for ID. Here, optimizing churn would provide a negligible gradient signal. Consequently, optimizing ID allows for transferring more of the utilized features from the teacher. Formally, we extend any given distillation loss $\mathcal{L}_{\text{distill}}$ matching representations by our ID term matching influences, resulting in the regularized optimization problem

$$S^* = \underset{S \in \mathcal{S}}{\arg\min} \, \mathcal{L}_{\text{distill}}(S, T) + \text{ID}(S, T). \tag{9}$$

We note that matching the influence distributions is related to matching Jacobians, which has been explored for grid-structured data (Czarnecki et al., 2017; Zagoruyko and Komodakis, 2017; Srinivas and Fleuret, 2018). The unique properties of the node classification task make it prohibitive to calculate $\text{ID}(S, T)$ exactly as the number of gradients needed is $(n \times d) \cdot (n \times h)$ when considering all pairs of node representations $\mathbf{H}^l \in \mathbb{R}^{n \times h}$ and input features $\mathbf{X} \in \mathbb{R}^{n \times d}$. Instead, we propose to approximate ID, which we will describe in detail next.

4.1. Approximating the Influence Difference using DropDistillation

Existing work on Knowledge Distillation for image classification proposed approximating the Jacobians by applying Gaussian noise to the input features (Srinivas and Fleuret, 2018; Nam et al., 2021). However, given the smoothing properties of graph neural networks, known as over-smoothing (Oono and Suzuki, 2019; Roth and Liebig, 2022b, 2023), these high-frequency signals are unfit for GNNs as they are filtered out quickly and have little effect on the output. This was similarly argued by Nam et al. (2021) for grid-structured data.

Instead, we argue that a broader class of perturbations can be used to approximate the gradients: When the influence distributions of the teacher and the student are close, any perturbations of the input should lead to a similar output. Input perturbations were shown to be beneficial for learning across multiple domains (Rong et al., 2020; Roth et al., 2021). Our approach uses a perturbation that exploits some of the properties specific to graphs by removing edges of the underlying graph, which we call DropDistillation (DD). Our intuition matches our goal: When removing an edge, the influence of the adjacent node is reduced, while the influence of all other edges should increase correspondingly. When repeatedly removing different edges and matching the outputs, the student can learn to mimic the influence of particular neighbors for the teachers' prediction of a node.

4.2. DropDistillation

For each training step, we remove edges uniformly at random with probability p^*. We define the set of edges to drop $\mathcal{E}^{drop} = \{e_{uv} \in \mathcal{E} | p_{uv} < p^*\}$ using uniformly random values $p_{uv} \sim U(0,1)$ for each edge $(u,v) \in \mathcal{E}$. We note that dropping edges were already successfully employed in DropEdge (Rong et al., 2020), though its reason is fundamentally different, as they are trying to mitigate over-smoothing and over-fitting. While their approach aims to map all perturbed graphs to the same output, we explicitly want different outputs to match the influence distribution. We express the perturbation using a matrix $\mathbf{A}^{\text{drop}}$ that offsets the edges to drop, i.e. $\mathbf{A}_{uv}^{drop} = -\mathbf{A}_{uv}$ for all $(u,v) \in \mathcal{E}^{\text{drop}}$. All other edges are scaled up accordingly, e.g., by recalculating the remaining edges' mean or attentional coefficients. In this case, the expected edge strength remains the same, as was demonstrated in DropEdge (Rong et al., 2020).

Our DropDistillation (DD) matches the logit outputs of T and S for the perturbed inputs using the mean squared error

$$\mathcal{L}_{DD}(S,T) = \frac{1}{N \cdot C} \sum_{v,i=1}^{N,C} \left(S(\mathbf{X}, \mathbf{A} + \mathbf{A}^{\text{drop}})_{vi} - T(\mathbf{X}, \mathbf{A} + \mathbf{A}^{\text{drop}})_{vi} \right)^2 \tag{10}$$

across all n nodes and c classes. We use $\mathcal{L}_{DD}$ in combination with a given method for Knowledge Distillation as described in Eq. 9. Since we use an approximation of the influence distributions, the resulting inputs may not be representative of the distribution of the actual gradients, and minimizing $\mathcal{L}_{DD}(S,T)$ may not lead to optimal results. To mitigate this, we start by solely optimizing $\mathcal{L}_{DD}$ until the loss plateaus and most of the teacher's reasoning is transferred to the student. As the student is typically unable to match the reasoning completely, we fine-tune the student using the regular distillation loss $\mathcal{L}_{\text{distill}}$. This also has a positive effect on runtime, as only a single forward pass is needed in each step.

Table 1: Statistics of our considered benchmark datasets for node classification.

Dataset	Graphs	Nodes (avg.)	Edges (avg.)	Features	Classes	Parameters T→S
Citeseer	1	$3,327$	$9,104$	$3,703$	6	9.7M→ 60k
Photo	1	$7,650$	$238,162$	745	8	3.7M→ 12k
Computers	1	$13,752$	$491,722$	767	10	1.3M→ 13k
WikiCS	1	$11,701$	$216,123$	300	10	872k→ 5.3k
Physics	1	$34,493$	$495,924$	$8,415$	5	4.5M→ 135k
PPI	20	$2,245.3$	$61,318.4$	50	121	3.1M→ 39k

4.3. The Theoretical Motivation for DropDistillation

We provided some intuition for dropping edges to approximate the influence distribution, for which we now also want to give a theoretical reason. This also motivates our usage of the squared error function. Srinivas and Fleuret (2018) show that adding random noise can approximately match the Jacobians using the first-order Taylor approximation. This was also adapted by Nam et al. (2021). We formally show that dropping edges similarly match the gradients and thus the influence difference. The critical similarity is that for many aggregation functions, e.g., mean or using attention, we can still assume zero expectation for edge perturbations $\mathbb{E}(\mathbf{A}_{\mathrm{drop}}) = \mathbf{0}$, as the remaining edges are scaled up equally. Our proof closely follows the proof for random noise (Srinivas and Fleuret, 2018):

Proposition 2 *Let* $T, S : \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times c}$ *be functions, and let* $\mathbf{X} \in \mathbb{R}^{n \times d}, \mathbf{A} \in \mathbb{R}^{n \times n}, \mathbf{A}_{\mathrm{drop}} \in \mathbb{R}^{n \times n}$ *be matrices. We further assume* $\mathbb{E}(\mathbf{A}_{vu}^{\mathrm{drop}}) = 0$ *for all* $v, u \in [1, \dots N]$. *Then,*

$$
\mathbb{E}\left[\sum_{v,i=1}^{n,c}(T(\mathbf{X}, \mathbf{A} + \mathbf{A}_{\mathrm{drop}})_{vi} - S(\mathbf{X}, \mathbf{A} + \mathbf{A}_{\mathrm{drop}})_{vi})^2\right] = \sum_{v,i=1}^{n,c}(T(\mathbf{X})_{vi} - S(\mathbf{X})_{vi})^2
$$

$$
+ \mathbb{E}_{\mathbf{A}_{\mathrm{drop}}}\left[\left(vec(\nabla_{\mathbf{A}}T(\mathbf{X}, \mathbf{A})_{vi} - \nabla_{\mathbf{A}}S(\mathbf{X}, \mathbf{A})_{vi})^T vec(\mathbf{A}_{\mathrm{drop}})\right)^2 + \sum_{v,u=1}^{n,n}\mathcal{O}((\mathbf{A}_{\mathrm{drop}})_{vu}^2)\right]
$$

We provide the detailed proof as supplementary material. We note that we are matching the gradient of the edges instead of the signal, though these are closely connected. While there is potential for a closer approximation, our method is sufficient for most use cases.

5. Experiments

We now empirically evaluate our claims using our proposed metrics and the effect of DropDistillation on Knowledge Distillation. We evaluate our method on the six standard benchmark datasets for node classification. Details about these datasets are given in Table 1. Citeseer (Giles et al., 1998), Photo (Shchur et al., 2018), WikiCS (Mernyei and Cangea, 2020), Computers (Shchur et al., 2018), Physics (Shchur et al., 2018) are transductive node classification tasks, so we randomly split the nodes into train, validation,

Table 2: Mean and standard deviation of our proposed metrics over five runs with random parameter initializations.

Dataset	Acc./F1-score (%)	C (%)	ID (%)	corr($\mathbf{id}, \mathbf{s}$)	corr($\mathbf{h}, \mathbf{s}$)
Citeseer	57.8 ± 2.5	32.2 ± 3.9	48.8 ± 2.8	0.01 ± 0.02	-0.06 ± 0.05
Photo	79.0 ± 3.6	26.3 ± 3.3	98.8 ± 3.2	0.03 ± 0.02	-0.26 ± 0.05
WikiCS	77.2 ± 0.8	19.3 ± 0.7	75.7 ± 4.9	-0.02 ± 0.01	-0.22 ± 0.02
Computers	65.2 ± 5.5	46.3 ± 3.7	104.9 ± 4.3	0.05 ± 0.03	-0.17 ± 0.03
Physics	86.9 ± 1.8	13.3 ± 1.4	75.5 ± 7.3	-0.03 ± 0.02	-0.28 ± 0.01
PPI	84.3 ± 0.2	10.5 ± 0.1	76.6 ± 2.1	-	-

and test sets. The same train nodes are used to optimize the target and distillation loss. The best-performing model based on the validation accuracy is chosen, and metrics are reported based on the corresponding test nodes. PPI (Zitnik and Leskovec, 2017) is an inductive multi-class classification task, so we use the public split based on entire graphs, replace the cross-entropy with the binary-cross-entropy loss and report the F1-score based on the test graphs. Each experiment runs for five random parameter initialization, and each metric's average and standard deviation is reported. We optimize our models using Adam (Kingma and Ba, 2014) using a learning rate of 0.005 and perform early stopping when the validation score does not improve for at least 400 steps. Our implementation reuses the general training framework and the existing methods from (Joshi et al., 2022) based on Pytorch-Geometric (Fey and Lenssen, 2019)[1]. We use a 3-layer Graph Attention Network (GAT) (Veličković et al., 2017) with residual connections as our base model for all presented experiments. We also provide results for all experiments using the Graph Convolutional Network (GCN) (Kipf and Welling, 2016) as supplementary material.

5.1. Empirical Validation of our Axioms

We now use our proposed metrics to empirically validate the presented axioms without considering the application of knowledge distillation. All one-hop neighbors are used as context nodes $C_i = N_i$ for our metrics. We provide the results for accuracy or F1-score, churn C, influence difference ID, the correlation corr($\mathbf{id}, \mathbf{s}$), and the correlation corr($\mathbf{h}, \mathbf{s}$) based on pairwise models with the same hyperparameters in Table 2. While the accuracy is rather stable between runs, ID demonstrates substantial differences in the influence between model instantiations of at least 48.8% in relative change for an average node pair. This already shows that each model bases their predictions on different features in the data, confirming Axiom 1. Notably, the influence difference is still significant even when the prediction churn is relatively low. Further, these influence differences show no correlation to the stability of node predictions. This indicates that all nodes are similarly unstable, not just those predicted differently by two models, supporting Axiom 2. The correlation to the entropy of neighboring labels is much higher and always negative, indicating that less

---

1. Our implementation is available at https://github.com/roth-andreas/distilling-influences.

Table 3: A comparison of the performance on the node classification tasks. For PPI, the F1-score is reported, and in all other cases, accuracy is reported. The best results are indicated in bold, the second-best are underlined.

Accuracy/F1-score	Computers	Physics	WikiCS	Photo	Citeseer	PPI
Teacher	80.8	91.2	79.7	85.4	68.8	98.8
Student	65.2 ± 5.5	86.9 ± 1.8	77.2 ± 0.8	79.0 ± 3.6	57.8 ± 2.5	84.5 ± 0.3
Student+DropEdge	72.2 ± 2.5	88.7 ± 1.3	77.9 ± 0.5	82.4 ± 3.3	60.9 ± 2.7	82.4 ± 0.2
KD	67.5 ± 5.6	87.1 ± 1.9	78.8 ± 0.7	78.6 ± 3.6	59.8 ± 2.1	84.7 ± 0.3
KD+DropEdge	$\underline{77.5} \pm 3.4$	$\underline{89.4} \pm 0.7$	$\underline{79.2} \pm 0.6$	82.4 ± 2.9	$\underline{62.9} \pm 3.3$	79.0 ± 0.2
G-CRD	70.6 ± 3.6	87.3 ± 1.2	77.5 ± 0.3	80.8 ± 1.6	58.0 ± 2.9	$\underline{84.7} \pm 0.2$
G-CRD+DropEdge	73.7 ± 2.7	88.5 ± 1.1	78.9 ± 0.2	$\underline{84.7} \pm 3.5$	59.4 ± 3.9	80.9 ± 0.2
DropDistillation	$\mathbf{80.0} \pm 0.9$	$\mathbf{90.8} \pm 0.9$	$\mathbf{79.6} \pm 0.4$	$\mathbf{84.9} \pm 2.7$	$\mathbf{66.3} \pm 0.9$	$\mathbf{85.0} \pm 0.1$

variance in the labels of neighboring nodes helps the stability of a prediction. This supports Axiom 3. We note that this correlation is small in some cases, showing that the entropy of neighboring labels is insufficient to capture the reason for the stability fully.

These findings show that GNNs use different reasonings to make their predictions. This aligns well with theoretical ideas on model similarity that hypothesize that each model learns different subsets of features from the data (Allen-Zhu and Li, 2023). It also validates our presented method, as matching the influence difference provides a meaningful metric for aligning a student to its teacher and finds signals even in the correctly classified nodes.

5.2. Evaluating of DropDistillation for Knowledge Distillation

We now evaluate our proposed DropDistillation (DD) on several benchmark tasks for Knowledge Distillation in node classification. We train one teacher model and five student models for each constellation based on the same teacher. We report the average prediction churn of a student compared to the teacher and the task performance. In our experiments, both the teachers and the students are GATs. To demonstrate the versatility of DropDistillation, we additionally present results for all experiments using a GCN as the student model in the supplementary material.

The GATs for teacher and student models only differ in the multiplier q on the number of hidden channels per layer and the number of attention heads. For the teacher model, we select the best-performing q and the number of heads fitting into our GPU memory of 8 GB. For the student model, we choose a much smaller q and number of heads with a noticeable difference in performance, allowing us to evaluate the impact of different methods. To only evaluate the effect of the distillation methods, we keep these fixed across all methods. Table 1 reports the number of parameters for all models. Compression factors are at least $40\times$ between teachers and students.

Table 4: Average model churn $C(S, T)$ between the teacher and each student. The models are the same as in Table 3. Lower scores are better.

Churn C	Computers	Physics	WikiCS	Photo	Citeseer	PPI
Student	38.3 ± 5.5	10.3 ± 1.3	16.9 ± 0.5	22.3 ± 2.8	29.3 ± 3.1	9.0 ± 0.2
Student+DropEdge	29.7 ± 2.7	8.5 ± 0.8	16.3 ± 0.6	18.9 ± 2.7	27.6 ± 2.9	10.3 ± 0.1
KD	32.1 ± 4.8	9.0 ± 2.1	9.3 ± 0.4	22.9 ± 3.3	30.9 ± 5.3	$\underline{9.0} \pm 0.1$
KD+DropEdge	$\underline{21.6} \pm 2.5$	$\underline{6.4} \pm 1.0$	$\underline{9.2} \pm 0.4$	19.2 ± 2.9	$\underline{24.6} \pm 2.7$	12.0 ± 0.1
G-CRD	30.0 ± 2.5	11.6 ± 1.9	15.5 ± 0.9	19.8 ± 2.5	34.4 ± 2.9	9.1 ± 0.1
G-CRD+DropEdge	27.6 ± 3.8	10.0 ± 1.7	13.6 ± 0.6	$\underline{16.7} \pm 3.5$	33.9 ± 2.5	11.3 ± 0.2
DropDistillation	$\mathbf{19.1} \pm 1.7$	$\mathbf{4.3} \pm 1.0$	$\mathbf{9.0} \pm 0.3$	$\mathbf{15.1} \pm 1.7$	$\mathbf{20.5} \pm 1.7$	$\mathbf{8.8} \pm 0.1$

5.2.1. COMPARABLE METHODS AND HYPERPARAMETERS

We evaluate several state-of-the-art student models as proposed originally as baselines and directly compare each model with its regularized version. We use grid search with the parameters described for each method below. The following methods are considered:

- As a baseline, we optimize a plain Student (**Student**) only having access to the true labels and no signal from the teacher model.

- The current state-of-the-art in terms of minimizing prediction churn directly performs Knowledge Distillation (**KD**) (Hinton et al., 2015), as this provably optimizes churn Jiang et al. (2022). We tune the parameter $\alpha \in \{0.25, 0.5\}$.

- We also compare our method to Graph Contrastive Representation Distillation (**G-CRD**) (Joshi et al., 2022), which we consider to be the current state-of-the-art in terms of distillation regarding downstream performance. G-CRD matches each node's student representation to its representation in the teacher model while separating representations of different nodes. We tune its parameter $\beta \in \{0.03, 0.1, 0.3\}$.

To quantify the advantage of DropDistillation, we additionally present results when combining each of these three methods with DropEdge. We consider the dropout rates in $\{0.2, 0.4\}$ and apply DropEdge before each convolution of the student during training.

For DropDistillation, we combine it with KD and additionally only tune the number of iterations to use $\mathcal{L}_{DD}$ in $\{50, 800, 1500\}$ and use the same dropout rate $p^* = 0.2$ across all experiments, as we find DD to be sufficiently stable. As the goals of DropEdge are complementary to ours, we also include it in our grid search.

5.2.2. RESULTS

We present the test accuracies and F1-scores of the best-performing models based on the validation scores in Table 3. Our method improves the performance in all considered cases by values between 0.2% and 3.4%. Results for the prediction churn of the same students and their teachers are presented in Table 4. Our method achieves even more significant improvements of up to 4.1%. We observed the least significant difference for PPI, for which

we found the student incapable of fitting the training data. The best results for the other methods and DD are frequently achieved when combined with DropEdge, showing that DD does not interfere with existing advancements. We also find DD to be less prone to overfitting, which we trace back to ID being similar for stable and unstable nodes (Axiom 2). The results for students based on the GCN are provided as supplementary material, as they exhibit similar insights.

Our experiments confirm that using our proposed DD to match the influence of the predictions between a teacher and its student is a valuable addition to KD. The predictions are matched more closely in all considered cases, increasing overall performance across all experiments. This indicates that more of the actual knowledge from the teacher can be transferred to the student. Instead of having to come up with reasons for the predictions of the teacher, transferring these influences aids the learning process of the student.

6. Conclusion

This work investigated the reasons behind prediction churn in graph neural networks by quantifying differences in the influence on a prediction between models based on our proposed Influence Difference (ID) metric. These instabilities are not limited to nodes with unstable predictions but are similarly observed for stable predictions across models. Based on these valuable signals, we propose DropDistillation (DD), a fast approximation to minimize ID between a student and its teacher in Knowledge Distillation (KD). Our experiments confirm the importance of aligning influences, as it improves both the stability of the student's predictions and the overall performance.

Our work suggests that future methods on KD can greatly benefit from incorporating similar influence-matching strategies. Beyond the scope of KD, we see great potential in enhancing our understanding of how models can effectively leverage the diverse features inherent in the data.

Acknowledgments

References

Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Srinadh Bhojanapalli, Kimberly Wilber, Andreas Veit, Ankit Singh Rawat, Seungyeon Kim, Aditya Menon, and Sanjiv Kumar. On the reproducibility of neural network predictions. *arXiv preprint arXiv:2102.03349*, 2021.

Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

Zhuo Chen and Yuhong Yang. Assessing forecast accuracy measures. 04 2004.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018.

Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *Advances in neural information processing systems*, 30, 2017.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, page 89–98, 1998. doi: 10.1145/276675.276685.

Gabriel Goh, Andrew Cotter, Maya Gupta, and Michael P Friedlander. Satisfying real-world goals with dataset constraints. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Heinrich Jiang, Harikrishna Narasimhan, Dara Bahri, Andrew Cotter, and Afshin Rostamizadeh. Churn reduction via distillation. In *International Conference on Learning Representations*, 2022.

Haydn Jones, Jacob M. Springer, Garrett T. Kenyon, and Juston Moore. If you've trained one you've trained them all: Inter-architecture similarity increases with robustness. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

Chaitanya K Joshi, Fayao Liu, Xu Xun, Jie Lin, and Chuan Sheng Foo. On representation knowledge distillation for graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Max Klabunde and Florian Lemmerich. On the Prediction Instability of Graph Neural Networks. *arXiv preprint arXiv:2205:10070*, 2022.

Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A survey of functional and representational measures. *arXiv preprint arXiv:2305.06329*, 2023.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong. Learning small-size dnn with output-distribution-based criteria. In *Fifteenth annual conference of the international speech communication association*, 2014.

Huiting Liu, Siddharth Patwardhan, Peter Grasch, Sachin Agarwal, et al. Model stability with continuous data updates. *arXiv preprint arXiv:2201.05692*, 2022.

Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

Mahdi Milani Fard, Quentin Cormier, Kevin Canini, and Maya Gupta. Launch and iterate: Reducing prediction churn. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Giung Nam, Jongmin Yoon, Yoonho Lee, and Juho Lee. Diversity matters when learning from ensembles. In *Advances in Neural Information Processing Systems*, volume 34, pages 8367–8377. Curran Associates, Inc., 2021.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

Andreas Roth and Thomas Liebig. Forecasting unobserved node states with spatio-temporal graph neural networks. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 740–747. IEEE, 2022a.

Andreas Roth and Thomas Liebig. Transforming pagerank into an infinite-depth graph neural network. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 469–484. Springer, 2022b.

Andreas Roth and Thomas Liebig. Rank collapse causes over-smoothing and over-correlation in graph neural networks. *arXiv preprint arXiv:2308.16800*, 2023.

Andreas Roth, Konstantin Wüstefeld, and Frank Weichert. A data-centric augmentation approach for disturbed sensor image segmentation. *Journal of Imaging*, 7(10):206, 2021.

Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278, 2021.

Tobias Schumacher, Hinrikus Wolf, Martin Ritzert, Florian Lemmerich, Martin Grohe, and Markus Strohmaier. The effects of randomness on the stability of node embeddings. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases: International Workshops of ECML PKDD 2021, Virtual Event, September 13-17, 2021, Proceedings, Part I*, pages 197–215. Springer, 2022.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Suraj Srinivas and François Fleuret. Knowledge transfer with jacobian matching. In *International Conference on Machine Learning*, pages 4723–4731. PMLR, 2018.

Cecilia Summers and Michael J Dinneen. Nondeterminism and instability in neural network optimization. In *International Conference on Machine Learning*, pages 9913–9922. PMLR, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Chenxu Wang, Wei Rao, Wenna Guo, Pinghui Wang, Jun Liu, and Xiaohong Guan. Towards understanding the instability of network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 34(2):927–941, 2020.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR, 2018.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017.

Donglin Zhuang, Xingyao Zhang, Shuaiwen Song, and Sara Hooker. Randomness in neural network training: Characterizing the impact of tooling. *Proceedings of Machine Learning and Systems*, 4:316–336, 2022.

Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.