# Explicit Pairwise Factorized Graph Neural Network for Semi-Supervised Node Classification

**Yu Wang**$^{1,2}$ **Yuesong Shen**$^{1}$ **Daniel Cremers**$^{1}$

$^1$Technical University of Munich, Germany
$^2$University of Illinois at Chicago, USA

## Abstract

Node features and structural information of a graph are both crucial for semi-supervised node classification problems. A variety of graph neural network (GNN) based approaches have been proposed to tackle these problems, which typically determine output labels through feature aggregation. This can be problematic, as it implies conditional independence of output nodes given hidden representations, despite their direct connections in the graph. To learn the direct influence among output nodes in a graph, we propose the **E**xplicit **P**airwise **F**actorized **G**raph **N**eural **N**etwork (EPFGNN), which models the whole graph as a partially observed Markov Random Field. It contains explicit pairwise factors to model output–output relations and uses a GNN backbone to model input–output relations. To balance model complexity and expressivity, the pairwise factors have a shared component and a separate scaling coefficient for each edge. We apply the EM algorithm to train our model, and utilize a star-shaped piecewise likelihood for the tractable surrogate objective. We conduct experiments on various datasets, which shows that our model can effectively improve the performance for semi-supervised node classification on graphs.

## 1 INTRODUCTION

We live in a world full of interconnections. For example, publications are connected with citation links; Social media users are connected when they follow each other [Qu et al., 2019]; protein structures are interconnections of amino acids. Modeling relational data is an important topic for machine learning. These relational data can be represented by graphs, which model a set of objects with node features and their relationships with graph edges [Zhou et al., 2018].

A large number of systems, including social networks [Hamilton et al., 2017], citation networks [Kipf and Welling, 2017], physical systems [Sanchez-Gonzalez et al., 2018], protein interactions [Fout et al., 2017], knowledge graphs [Hamaguchi et al., 2017], etc. can be represented as graphs. And graph-based learning is receiving increasing attention from researchers.

In this paper, we focus on the problem of classifying nodes of a graph, such as a graph of a citation network. This problem can be framed as graph-based semi-supervised learning since the label information is available for only a small subset of nodes.

For problems involving data which have grid-like graph structures, e.g., images and videos, Convolutional Neural Networks (CNN) have achieved state-of-the-art results [Veličković et al., 2018]. However, many problems involve data such as 3D meshes and citation networks, which are represented by graphs having a general structure. An increasing number of researchers try to generalize the notion of convolution for general graph structure, motivated by its success in the domain of computer vision, to address the problems involving general graphs. And there are many attempts to extend neural networks to deal with graphs with arbitrary structure: Graph Neural Networks (GNNs), such as Graph Convolutional Networks (GCN) [Kipf and Welling, 2017] and Graph Attention Networks (GAT) [Veličković et al., 2018] have been receiving increasing attention because of their effective node representation learning on general graphs. These methods take into account the graph structure and aggregate features from neighbor nodes.

However, one weakness of these methods is that they neglect direct local dependency among output label nodes, which can be important for node classification. To learn the interdependency of connected node labels, we propose to model the whole graph with a pairwise partially observed Markov Random Field (MRF) and use explicit pairwise factors to model direct dependencies between connected node labels.

*Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence* (UAI 2021), PMLR 161:1979–1987.

To benefit from the representational power of GNNs, we apply a GNN backbone to learn appropriate unary factors that can reflect the influence from the node input features on node output labels. To avoid overfitting, we use a shared compatibility matrix to parametrize the pairwise factors, which is further multiplied by an edgewise coefficient to reflect the difference among the edges in a graph.

The partially observed MRF representation learns node representation and node label interdependency simultaneously, taking into account both observed and unobserved nodes. However, the conventional learning objective of maximizing observed data log-likelihood requires marginalizing unobserved nodes, which has an exponential computational cost and is thus intractable. To address this challenge, we use the EM algorithm [Koller and Friedman, 2009] to learn the MRF and maximize instead the evidence lower bound with alternating E-steps and M-steps: in each E-step we update the candidate distribution on the unobserved nodes, which is then used in the subsequent M-step to maximize the expected complete data log-likelihood.

This alternating training is still challenging because the inference process on general loopy graphs is known to be intractable [Koller and Friedman, 2009]. To remedy this issue and perform training efficiently, we introduce a piecewise likelihood [Sutton and McCallum, 2009] based local training method, which provides a surrogate objective that makes the inference process more manageable. It divides the graph into tractable subgraphs for easy inference, and avoids the intractable global inference procedure.

Our contributions are:

- We propose the explicit pairwise factorized graph neural network (EPFGNN), which uses explicit pairwise output–output factors to augment a graph neural network backbone so that we can model direct dependencies among output label nodes in graphs.
- We use a shared compatibility matrix to parametrize pairwise factors and extend it with an edgewise multiplicative scaling coefficient. This ensures that the model can aggregate the neighbor node features and label information flexibly on both assortative graphs and disassortative graphs, while simultaneously avoid overfitting.
- We propose an EM-based learning procedure, and employ piecewise likelihood as surrogate objective to make the training on MRF tractable while retaining the structural information.
- We conduct a series of experiments [1], which shows that our model, along with the proposed training procedure, can effectively tackle semi-supervised node classification on graphs, outperforming existing baselines.

## 2 RELATED WORK

**Graph neural network** Motivated by the effectiveness of CNNs, an increasing number of researchers aim to design convolution operation on graphs to extract information from connected neighbor node features. The resulting graph neural networks are generally defined by a message-passing scheme as follows:

$$h_i^{k+1} = update(h_i^k, aggregate_{j\in\mathcal{N}(i)}(h_i^k, h_j^k)), \qquad (1)$$

where $h_i^k$ is the hidden representation of a node at k-th layer, $\mathcal{N}(i)$ is the set of neighbor nodes connected with node $i$. The node representations are updated by weighted aggregations of the features from each central node and its neighbors. A variety of aggregation and update functions have been proposed by different GNN variants.

An effective GNN model named Graph Convolution Network (GCN) by Kipf and Welling [2017] reduces the computational complexity of eigen-decomposition involved in spectral graph convolution by introducing several approximations. The GCN layer is defined as follows:

$$H^{k+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^kW^k), \qquad (2)$$

where $\tilde{A} = A + I_N$ is the adjacency matrix with self-connections, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ indicates the degree of node $i$. Since the graph has an arbitrary structure, the number of neighbors varies for different nodes, and can be large in graphs of massive scale. It is then inefficient to aggregate features from all neighbor nodes. To address this issue, GraphSAGE [Huang et al., 2018] applies a layer-wise sampler to control the size of neighbors. While GraphSAGE assumes the contribution of neighbors to the central node has equal importance, the method named Graph Attention Network (GAT) [Veličković et al., 2018] applies an attention mechanism to weigh the different contributions from neighbor node features to the central node.

While the aforementioned GNN models can effectively process graph structured data for semi-supervised node classification, they neglect the direct interdependency among output nodes despite their connections in the graph structures. To address this problem, graph Markov neural network (GMNN) [Qu et al., 2019] proposes a pseudo-likelihood based variational EM framework to model the interdependency among node labels. In the M-step, they optimize the graph neural network by maximizing the pseudo-likelihood. The pseudo-likelihood assumes the conditioning of connected node labels This assumption deviates from typical semi-supervised settings since many of the nodes are unobserved. In E-steps, GMNN uses another graph neural network to aggregate the information from connected node labels. This implicitly assumes that the graph is assortative, as typical feature aggregation operation in GNNs perform smoothing. However, in case of disassortative graphs where connected nodes tend to disagree with each other, the above assumption no longer holds.

[1] https://github.com/YuWang-1024/EPFGNN

**Surrogate objective function** Given a graph representation $\mathcal{G} = (V, E)$, where $V$ is the set of nodes, and $E$ is the set of edges. Denote the set of node features as $\mathcal{X}$ and the corresponding node labels as $\mathcal{Y}$. We employ a Markov random field [Lafferty et al., 2001] [Koller et al., 2007] to model the node classification problem. Since the inference on the MRF with loops is known to be intractable [Koller and Friedman, 2009], the common learning strategy of maximizing data likelihood is infeasible [Nowozin and Lampert, 2011]. This issue motivates the finding of a surrogate objective function $p_{approx}(y|x)$ that approximates the original likelihood. There are two kinds of surrogate training objectives named pseudo-likelihood (PL) [Koller and Friedman, 2009] and piecewise training (PW) objective [Sutton and McCallum, 2009].

Pseudo-likelihood ($p_{PL}$) consists of the product of node-wise conditional likelihood given its Markov blanket. The intuition is that we can obtain the probability density of a node if its Markov blanket is observed. However, the pseudo-likelihood is known to have difficulties modeling longer-range dependencies [Koller and Friedman, 2009].

$$p_{PL}(y|x) = \prod_{n \in V} p_{PL}(y_n | y_{\mathcal{N}(n)}, x). \tag{3}$$

Piecewise training objective ($p_{PW}$) [Sutton and McCallum, 2009] is an alternative way to make the learning on MRF tractable. It models the graph using a factor graph and split the whole graph into tractable subgraphs called pieces. It is inspired by the intuition that the global distribution will be reasonable if all the local factors fit the data well. Given the set of subgraphs $P$, the set of the factors $F$ within each subgraph $R \in P$, and the partition function $Z_R(x)$ of each subgraph that normalizes the piecewise likelihood, the piecewise training objective can be formulated as

$$p_{PW}(y|x) = \prod_{R \in P} \frac{1}{Z_R(x)} \prod_{F \in R} \phi_F(y_F, x). \tag{4}$$

Piecewise training objective typically yields better results compared to pseudo-likelihood [Sutton and McCallum, 2009], as the factors in the original likelihood are conserved in the piecewise training objective, and the interdependencies among nodes are better retained.

## 3 METHODOLOGY

We denote a graph as $\mathcal{G} = (V, E)$ , where $V$ is the set of nodes, $E$ is the set of edges. Under the framework of semi-supervised node classification, the full node features $x_V$ and a small subset of node labels $Y_V$ are observed in the graph. Thus, node labels are split into labeled and unlabeled subset $Y_V = (Y_L, Y_U)$. Our goal is to learn a model that can predict these unknown classes $Y_U$ based on the full input features $x_V$, observed output labels $y_L$, and the graph structure.

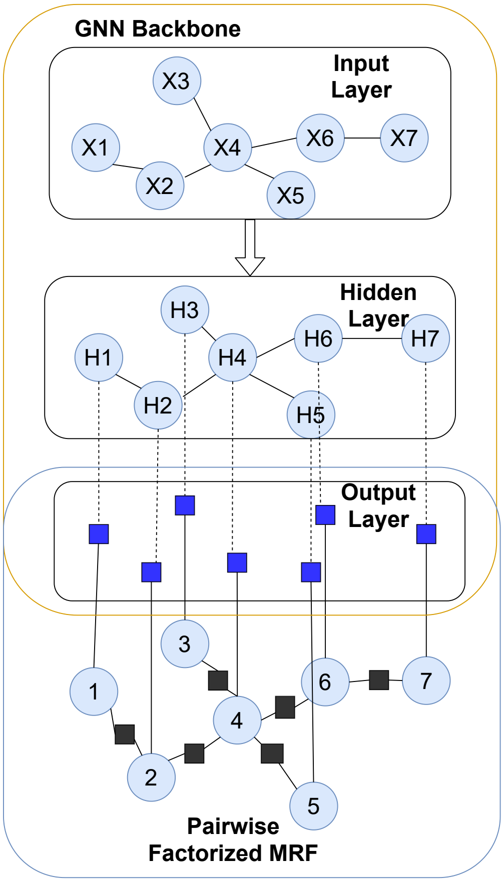

Figure 1: Model representation of EPFGNN. We use the GNN backbone to learn the valid input–output factors that reflect the influence of input node features on the node output labels. The output of GNN backbone is used as the input–output factor value. Furthermore, every connected node pairs are connected with a pairwise factor. We use the shared compatibility matrix of size $c \times c$ to parametrize the pairwise factor, where $c$ is the output dimension.

### 3.1 MODEL REPRESENTATION AND LEARNING OBJECTIVE

To achieve our goal, we propose the explicit pairwise factorized graph neural network (EPFGNN). We represent the overall distribution $P(Y_V|x_V)$ by a MRF and assume that it admits the following factor decomposition:

$$\begin{aligned} &P(Y_V|x_V) \\ &= \frac{1}{Z(x_V)} \prod_{i \in V} \phi_i(Y_i, x_V) \prod_{(j,k) \in E, j<k} \phi_{j,k}(Y_j, Y_k). \end{aligned} \tag{5}$$

In the above expression, all factors are constrained to be positive, and $Z(x_V)$ is the partition function that ensures normalization of the distribution. As shown in Figure 1, this factorization needs two kinds of factors to model the whole distribution:

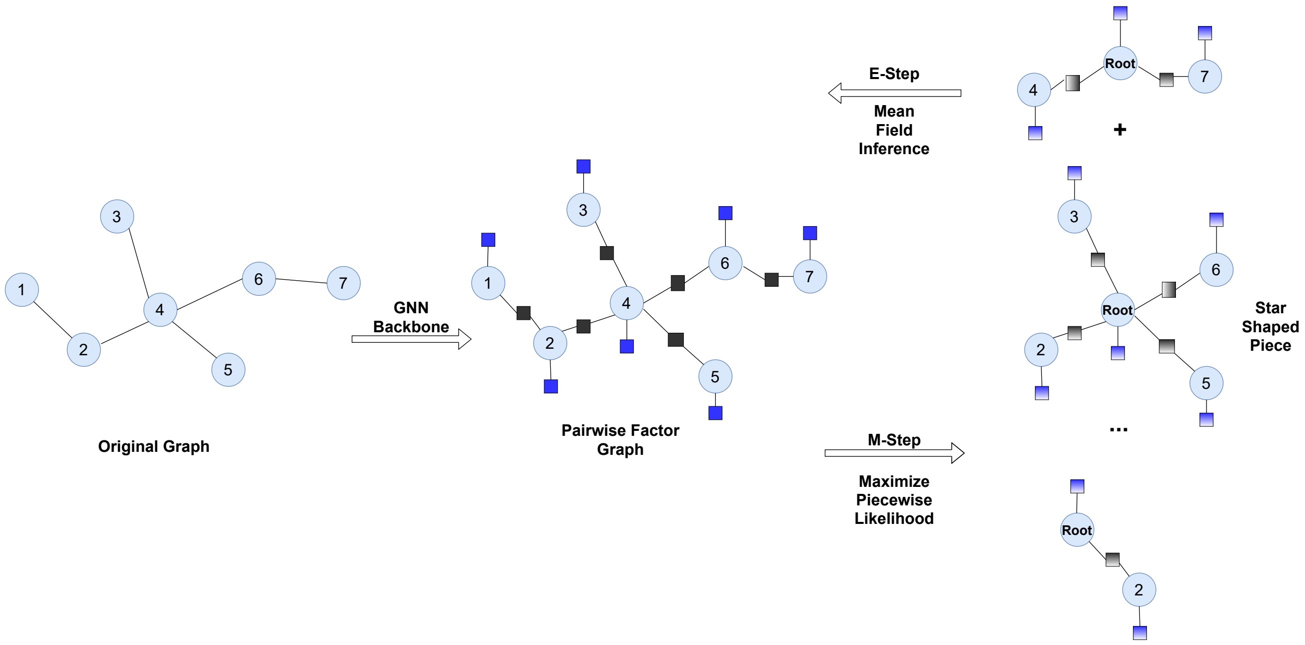

Figure 2: EPFGNN training framework. The GNN backbone output is used as input–output factor value. We additionally parametrized the pairwise factor with a compatibility matrix for every connected node pair. We apply EM alternating training method to tackle the intractable challenge caused by partial observation on MRF. In the M-step, we split the graph into star-shaped pieces and maximize the expected complete log-likelihood for every piece. To avoid overcounting, we redistribute the factor values that appear in more than one piece. In the E-step, we merge the pieces back to the original MRF and estimate the posterior distribution $q$ with fixed factor values using the mean field inference method.

- The input–output factors $\phi_i(Y_i, x_V)$ represent the influence of input node features $x_V$ on output node $Y_i$. These are high order factors since they involve all input features, which make direct factor parametrization intractable. To solve this issue, we use a single graph neural network (GNN) backbone to parametrize all these factors so their values can be obtained via feedforward inference: given an input $x_v$, the GNN backbone should produce a prediction of $\phi_i(y_i, x_v)$ for each output node $i \in V$ and possible output label $y_i$.
- The output–output factors $\phi_{j,k}(Y_j, Y_k)$ represent the influence between connected node labels. They are pairwise factors and model the direct dependency between labels for every connected node pair. To avoid overfitting, we parametrize these factors with a single shared symmetric compatibility matrix $K$ of size $c \times c$, where $c$ is the output dimension.

Thus our EPFGNN model is parametrized by parameters $\theta$ of the GNN backbone and a compatibility matrix $K$. Our learning objective is to find the parameters $\theta^*$ and $K^*$ that maximize the observed data log-likelihood of labeled output $\tilde{y}_L$ given the input features $\tilde{x}_V$:

$$\theta^*, K^* = \operatorname*{argmax}_{\theta, K} \log P(Y_L = \tilde{y}_L | \tilde{x}_V; \theta, K). \tag{6}$$

### 3.2 PIECEWISE EM TRAINING FRAMEWORK

Direct MRF learning under semi-supervised setting is intractable because of the following two challenges:

**Partial observation** Since the output subset $Y_U$ has no observation, we are in the partial observation case where $Y_U$ are unobserved variables. There are two approaches [Koller and Friedman, 2009] to tackle this problem in general: We can either perform gradient ascent to maximize the observed data likelihood directly or use the expectation-maximization (EM) algorithm. In our case, computing the observed log-likelihood directly is computationally infeasible, since we need to marginalize the unobserved random variables $Y_U$, and complex graph structure makes the marginalization computation grows exponentially. Therefore, we apply the EM algorithm, which maximizes instead the expected complete data likelihood w.r.t. estimated posterior distribution. We will discuss the details in Section 3.2.1.

**Intractable inference** It is well known that the inference process on a general MRF is intractable [Koller and Friedman, 2009] because of the complex graph structure. This makes the learning process impractical beyond simplistic settings. One approach to tackle this problem is to use approximate probabilistic inference methods. However, they are commonly iterative procedures, which are unfriendly for auto-differentiation and hard to ensure convergence. We apply an alternative approach to replace the log-likelihood objective with a more manageable surrogate objective, which makes it easy to do inference and learning. The surrogate objective we use is the piecewise training objective [Sutton and McCallum, 2009]. We will elaborate the details in Section 3.2.2.

### 3.2.1 EM framework

In the semi-supervised node classification setting, many nodes are unlabeled, which means factor values of the complete data likelihood (Equation (5)) are undetermined. As the direct maximization over the observed log-likelihood $\log P(y_L|x_V)$ involves intractable marginalization of unobserved output nodes $Y_U$, we apply the EM framework to leverage the information from unlabeled nodes. EM algorithm maximizes a lower bound of the observed log-likelihood: it finds a variational distribution denoted as $q(Y_U|x_V)$ to approximate the posterior distribution of unlabeled nodes $p(Y_U|y_L, x_V; \theta, K)$, and maximizes the expected complete log-likelihood over all nodes w.r.t. the distribution of $q(Y_U|x_V)$. This lower bound comes from our wish to fill in the missing values $y_U$ with a proposal distribution $q(Y_U|x_V)$ and employ the (expected) complete log-likelihood for optimization instead. So the objective function is changed as follows, which is referred to as the evidence of lower bound (ELBO):

$$\begin{aligned}\mathcal{L}(q;\theta,K) =&\mathbb{E}_{q(Y_U|x_V)}[\log P(Y_U, y_L|x_V;\theta,K)]\\ &+\sum_{Y_U} -q(Y_U|x_V)\log q(Y_U|x_V).\end{aligned} \tag{7}$$

where the first term is the expected complete log-likelihood and the second term is the entropy of the proposal distribution $q$. It is guaranteed to be a lower bound of the observed log-likelihood $\log P(y_L|x_V)$ since their difference is the KL-divergence $D_{KL}(q(Y_U|x_v)\|p(Y_U|y_L, x_V))$ which is always positive.

$$\begin{aligned}&\log P(y_L|x_V) - \mathcal{L}(q)\\ &= D_{KL}\big(q(Y_U|x_v)\|p(Y_U|y_L, x_V)\big) \geq 0.\end{aligned} \tag{8}$$

The EM algorithm is none other than an alternating maximization of ELBO with E-step and M-step. During E-steps, we fix the learned parameters $(\theta, K)$ of posterior distribution $p(Y_U|y_L, x_V; \theta, K)$ and update variational distribution $q(Y_U|x_V)$. During M-steps, we fix the proposal variational distribution $q(Y_U|x_V)$ and learn parameters $(\theta, K)$ that maximize the expected complete log-likelihood over all nodes w.r.t. the distribution of $q(Y_U|x_V)$.

**E-Step** In an E-step we fix parameters $(\theta, K)$ and update $q$ to maximize the ELBO $\mathcal{L}(q;\theta,K)$, which according to Equation (8) is equivalent to minimizing the KL-divergence $D_{KL}(q(Y_U|x_v)\|p(Y_U|y_L, x_V))$ since $\log P(y_L|x_V)$ does not depend on $q$. So the update function during E-Step is

$$q^*(Y_U|x_V) = \underset{q}{\operatorname{argmin}}\, D_{KL}(q(Y_U|x_V)\|p(Y_U|y_L, x_V)). \tag{9}$$

The proposal distribution $q$ has two desiderata: it should closely approximate the true likelihood distribution of missing values $p(Y_U|y_L, x_V)$, and it should have a simple form such that the ELBO can easily be maximized. Here we restrict the proposal distribution $q$ to have a fully factorized form

$$q(Y_U|x_V) = \prod_{i\in U} q_i(Y_i|x_V), \tag{10}$$

so that the proposal distribution $q(Y_U|x_V)$ is a mean field approximation [Koller and Friedman, 2009] of $p(Y_U|y_L, x_V)$.

With the mean field approximation, there is a closed-form expression for updating the proposal distribution q when fixing the parameters $\theta$ and K: $\forall i \in U$, we have

$$\begin{aligned}q_i(Y_i|x_V) \propto \exp\big(&\log\phi_i(Y_i, x_v)\\ &+ \sum_{j\in NB(i)\cap L} \log\phi_{i,j}(Y_i, y_j)\\ &+ \sum_{k\in NB(i)\cap U}\sum_{Y_k}\log\phi_{i,k}(Y_i, Y_k)q_k(Y_k|x_V)\big).\end{aligned} \tag{11}$$

Since the mean field update is essentially a message-passing procedure, we could easily design a message passing layer using the above function to update the proposal distribution $q$.

**M-Step** In an M-step we fix the proposal distribution $q$ and update parameters $(\theta, K)$ to maximize the ELBO $\mathcal{L}(q;\theta,K)$, which is equivalent to maximizing the expected complete log-likelihood $\mathbb{E}_q[\log P(Y_U, y_L|x_V;\theta,K)]$ because the entropy term (second part in Equation (7)) only depends on $q$. So the objective function during M-Step is

$$\theta^*, K^* = \underset{\theta,K}{\operatorname{argmax}}\, \mathbb{E}_{q(Y_U|x_V)}[\log P(Y_U, y_L|x_V;\theta,K)]. \tag{12}$$

Computing the (expected) complete log-likelihood is also infeasible since the random variable grows exponentially. To make the learning procedure tractable, we apply the piecewise learning method, which we will discuss in the next section.

### 3.2.2 Piecewise surrogate objective for M-Step

The inference on MRF is intractable, and the computation of partition function $Z(x_V)$ in Equation (5) is infeasible for graph data in practice. This hinders the learning process of our model. To overcome this, we utilize a tractable surrogate objective function to approximate the original likelihood.

**Star-shaped piecewise surrogate objective** The piecewise training method provides an alternative local training approach [Sutton and McCallum, 2009]. It breaks the whole graphical model down into tractable subgraphs and performs inference separately on each piece. The subgraphs can have overlaps among them and should have tractable structures for inference, e.g., trees. The factors contained in each subgraph are used to define its distribution. In our case, we want to model the interdependency between the central node and its neighbors. Thus, we star-wisely split the entire pairwise factor graph of the MRF: As shown in Figure 2, for every node $i$ in the graph, we create a piece which groups this node and its neighbors as a subgraph. So the objective function approximates the original log-likelihood as

$$\begin{aligned}\log P(Y_V|x_V;\theta,K) &\approx \ell_{pw}(Y_V|x_V)\\ &= \sum_{i\in V}\log P_i^*(Y_i, Y_{\mathcal{N}(i)}|x_V),\end{aligned} \tag{13}$$

where $\mathcal{N}(i)$ represents the neighbor nodes of central node $i$. This star-shaped piece is a tree structure rooted at the central node, and this piece is parametrized by the factors from the original MRF and pairwise factorized following Equation (5). Thus, the partition functions of the subgraphs can be easily computed using belief propagation [Koller and Friedman, 2009]. With this approximation, we can design a message passing layer to estimate the surrogate objective function.

**Factor redistribution** For every node, we group this node and its connected neighborhood as a piece, which generates overlaps among different pieces as shown in Figure 2. This means the factors that are contained in intersecting regions of different pieces will be overcounted. To avoid this issue, we redistribute the overlapping factor values in every subgraph. An even redistribution of factors is formulated as follows:

$$\begin{aligned}\psi_i(Y_i, x_V) &= \big(\phi_i(Y_i, x_V)\big)^{\frac{1}{d(i)+1}}\\ \psi_{j,k}(Y_j, Y_k)) &= \big(\phi_{j,k}(Y_j, Y_k))\big)^{\frac{1}{2}}.\end{aligned} \tag{14}$$

With this redistribution, the distribution of each piece $\bar{P}_i^*(Y_i, Y_{\mathcal{N}(i)}|x_V)$ becomes

$$\begin{aligned}&\bar{P}_i^*(Y_i, Y_{\mathcal{N}(i)}|x_V)\\ &= \frac{1}{\bar{Z}_i^*(x_V)}\psi_i(Y_i, x_V)\prod_{j\in\mathcal{N}(i)}\psi_j(Y_j, x_V)\psi_{i,j}(Y_i, Y_j),\end{aligned} \tag{15}$$

so that the redistributed piecewise log likelihood $\bar{\ell}_{pw}(Y_V|x_V)$ of the whole model has the following form:

$$\begin{aligned}\bar{\ell}_{pw}(Y_V|x_V) = &\sum_{i\in V}\log\phi_i(Y_i, x_V)\\ &+ \sum_{(j,k)\in E, j<k}\log\phi_{j,k}(Y_j, Y_k)\\ &- \sum_{i\in V}\log\bar{Z}_i^*(x_V).\end{aligned} \tag{16}$$

Here we see that the factor overcounting issue is fixed. This piecewise likelihood reasonably approximates the original distribution $P(Y_V|x_V)$ since the only difference between Equation 5 is the partition function.

**Edgewise scalar coefficient** For our discussion so far, we only considered a single $c \times c$ compatibility matrix $K$ to parametrize pairwise factors $\phi_{j,k}(y_j, y_k)$ of every connected node pair, meaning that all connected node pairs share a common parametrization (denote $K(l,m) = K_{l,m}$):

$$\log\phi_{j,k}(y_j, y_k) = K(y_j, y_k). \tag{17}$$

This representation can be restricting since it does not differentiate the edges in the graph. It is thus necessary to relax this constraint in order to equip our model with more representational power. However, if we parametrize every edge with a different $c \times c$ matrix, the model will easily suffer from overfitting. In order to balance these two aspects, we introduce an additional scalar parameter $\alpha_{j,k}$ for each edge $(j,k)$ that scales the shared compatibility matrix. Thus, the parametrization becomes

$$\log\phi_{j,k}(y_j, y_k) = \alpha_{j,k}\ K(y_j, y_k) \tag{18}$$

and the star-shaped piecewise log-likelihood becomes

$$\begin{aligned}\bar{\ell}_{pw}(Y_V|x_V) = &\sum_{i\in V}\log\phi_i(Y_i, x_V)\\ &+ \sum_{(j,k)\in E, j<k}\alpha_{j,k}\ K(Y_j, Y_k)\\ &- \sum_{i\in V}\log\bar{Z}_i^*(x_V).\end{aligned} \tag{19}$$

## 4 EXPERIMENT AND RESULT

In this section, we compare our model with state-of-the-art methods such as GCN [Kipf and Welling, 2017], GAT [Veličković et al., 2018], and GMNN [Qu et al., 2019] for semi-supervised node classification problems. We evaluate our model on a wide variety of datasets, including both assortative and disassortative graphs. We also conduct an ablation study to analyze different components of our model. We report average accuracy on 50 runs of experiments for analysis.

### 4.1 DATASET

In this section, we introduce the characteristics of graph datasets including both assortative and disassortative graphs.

**Citation networks** The citation network is a graph where nodes represent papers, edges depict the citation relationship between two papers, and node features are the bag of words of that paper. The node label is the topic of the corresponding paper. Cora, Citeseer and PubMed are common benchmark datasets [Namata et al., 2012] [Sen et al., 2008] of this type. We employ the same configuration as in Yang et al. [2016], which randomly picks 20 samples out of every class as training nodes, and randomly selects 500 and 1000 samples for validation and test, respectively.

**Wikipedia networks** Wikipedia networks are graphs where nodes stand for websites, and edges represent the mutual links of two pages. Node features are representative nouns of that page, and nodes are classified according to their average monthly traffic. Chameleon and Squirrel [Rozemberczki et al., 2021] are both examples of such networks. We split the graph to the train, validation, and test according to the following ratio: 20%, 20%, and 60%.

**Actor co-occurrence networks** Actor co-occurrence network is a graph extracted from film-director-actor-writer network [Tang et al., 2009]. The Actor dataset is one of the graph where nodes represent actors and edges indicate co-occurrence relationship in the same website between two actors. The node features are representative keywords of an actor. We split the graph in the same manner as for Wikipedia networks.

Table 1: Statistics of common datasets. $\beta$ is defined in Section 4.3 and indicates the graph homophily. Higher $\beta$ means the connected nodes in the graph tend to agree with each other and have the same labels.

| Dataset | Nodes/Edges | Features/Classes | $\beta$ |
|---|---|---|---|
| Cora | 2708 / 5429 | 1433 / 7 | 0.83 |
| Citeseer | 3327 / 4732 | 3703 / 6 | 0.71 |
| Pubmed | 19717 / 44338 | 500 / 3 | 0.79 |
| Chameleon | 2277 / 36101 | 2325 / 5 | 0.25 |
| Squirrel | 5201 / 217073 | 2089 / 5 | 0.22 |
| Actor | 7600 / 33544 | 931 / 5 | 0.24 |

### 4.2 CLASSIFICATION ON GRAPHS

To evaluate the effectiveness of our method, we compare our model with several baseline methods on semi-supervised node classification tasks. We use the Cora, Citeseer, and Pubmed datasets for comparison. For our EPFGNN model, we utilize GCN as the graph neural network backbone to learn the representation of input–output factors given input features. We use the same hyperparameter setting as described in [Kipf and Welling, 2017]. We employ a compatibility matrix to model the interdependency among node labels and extend the pairwise factors with a scalar coefficient for every edge. The test accuracies of different models on different datasets are collected in Table 2.

Table 2: Mean classification accuracy of the proposed EPFGNN model compared with baseline methods on Cora, Citeseer and Pubmed datasets.

| Model | Cora | Citeseer | Pubmed |
|---|---|---|---|
| GCN | 81.56 | 70.37 | 78.69 |
| GAT | 82.08 | 71.44 | 77.52 |
| GMNN | 82.05 | 70.53 | 79.38 |
| EPFGNN | **83.54** | **73.13** | **80.15** |

As shown in Table 2, our method outperforms other state-of-the-art baselines, especially on the Citeseer dataset, where we observe a performance improvement for over 2%. These results support our analysis that modeling the direct interdependency among node labels will improve performance. Also, a more significant performance increase is observed on the Citeseer dataset, which has the lowest graph homophily $\beta$ among three datasets. This shows the advantage of the EPFGNN representation: contrary to other baselines methods, it goes beyond feature smoothing.

### 4.3 ANALYSIS OF GRAPH HOMOPHILY

The citation networks are assortative graphs since the connected nodes tend to have the same label. In contrast, the Wikipedia networks and actor co-occurrence networks are disassortative graphs where the assumption of graph homophily no longer holds. To quantify the degree of graph homophily, we use the measure $\beta$ from Pei et al. [2020] defined as follows:

$$\beta = \frac{1}{V} \sum_{v \in V} \frac{\textit{Number of v's neighbors with same label}}{\textit{Number of v's neighbors}}. \tag{20}$$

The measure $\beta$ ranges between 0 and 1, where assortative graphs like Cora have high $\beta$ values above 0.5 while disassortative ones like Chameleon have low $\beta$ values below 0.5. $\beta$ value for various graph datasets are provided in Table 1.

We further experiment with disassortative graph datasets Chameleon, Squirrel, and Actor. We report the test accuracy results in Table 3.

As shown in Table 3, to a certain degree, our model managed to adapt to disassortative graphs. Since we have a shared compatibility matrix and every edge has one additional degree of freedom from the edgewise scalar coefficient, the explicit pairwise modeling in EPFGNN enables the model

Table 3: Mean classification accuracy on disassortative graphs. We compare our model with GCN and GMNN on Chameleon, Squirrel and Actor datasets which have small $\beta$ values.

| Model | Chameleon | Squirrel | Actor |
|---|---|---|---|
| GCN | 34.66 | 24.56 | 26.85 |
| GMNN | 34.69 | 24.77 | **27.04** |
| EPFGNN | **35.31** | **25.12** | 26.66 |

to flexibly aggregate the neighbor node label information when the neighboring nodes tend to disagree.

### 4.4 ABLATION STUDY

The proposed EPFGNN model learns the node representation and aggregates estimated label information from connected neighbor nodes. The key components of this model are the GNN backbone and the parametrization of pairwise factors. In this section, we compare different model variants to examine the contribution of each component.

**GNN-Backbone** The role of the GNN backbone in the EPFGNN model is to model the influence of input features on output labeling. It parametrizes the input–output factors in the MRF. To find an appropriate choice, we consider two commonly used GNN models, GCN and GAT, as backbones for node representation learning, and evaluate their effectiveness by conducting experiments on three standard datasets Cora, Citeseer, and Pubmed.

Table 4: Comparison of classification accuracy with EPFGNN using GCN and GAT backbones.

| GNN backbone | Cora | Citeseer | Pubmed |
|---|---|---|---|
| GCN | **83.24** | **72.35** | **79.61** |
| GAT | 82.32 | 71.99 | 79.23 |

In Table 4, we observe that GCN outperforms GAT on all three datasets. These results indicate that GAT, which has more parameters compared to GCN, is likely to suffer from overfitting as GNN backbone for EPFGNN.

**Redistribution** The star-shaped split of the whole graph will cause overlaps among pieces and result in the overcounting of factors when estimating the piecewise likelihood. To address this problem, we redistribute the factor values as described in Equation (14). We refer to this distribution method as average redistribution. To verify whether this redistribution is reasonable, we compare it with a different redistribution method referred to as center redistribution. For center redistribution, we assign each input–output factor $\phi_i(Y_i, x_V)$ entirely to the subgraph where $i$ is the central node. This yields a different estimation of the piecewise likelihood. We compare these two redistribution methods on the three standard datasets Cora, Citeseer, and Pubmed and collect the classification accuracies in Table 5.

Table 5: Mean classification of EPFGNN with different redistribution methods. For all experiments, we use GCN as GNN backbone and do not use the additional edgewise coefficient.

| Factor redistribution | Cora | Citeseer | Pubmed |
|---|---|---|---|
| average | **83.24** | 72.35 | **79.61** |
| center | 82.42 | **72.46** | 78.67 |

The results in Table 5 show that average redistribution tends to yield better results, possibly owing to the fact that the input–output factors provided by the GNN backbone is accessible in all related subgraphs. It remains an interesting future work to analyze other possible redistribution schemes.

**Edgewise scaling coefficient** To understand the effect of introducing the edgewise scaling coefficient, we compare it with two alternative settings. We name the setting that only has shared parametrization for all output–output factors as EPFGNN w/o coefficient and name the setting with both shared compatibility matrix and shared scalar coefficient for output–output factors on every edge as EPFGNN using layer coefficient. We refer to the original setting, which has a shared compatibility matrix and edgewise scalar coefficients, as EPFGNN using edge coefficient. Their comparison is summarized in Table 6.

Table 6: Mean accuracy of EPFGNN with different output–output factor parametrization settings. For all experiments, we use GCN backbone and average redistribution.

| Scaling factor | Cora | Citeseer | Pubmed |
|---|---|---|---|
| w/o coefficient | 83.24 | 72.35 | 79.61 |
| layer coefficient | 83.03 | 72.33 | 78.95 |
| edge coefficient | **83.54** | **73.13** | **80.15** |

The results in Table 6 show that the edgewise coefficient setting outperforms other alternative settings. Thus we can conclude that the additional flexibility provided by the edgewise scaling coefficient is beneficial and can effectively improve the representational power of the EPFGNN model.

## 5 CONCLUSION

This paper proposes the novel EPFGNN framework in which explicit pairwise factors are defined to model direct dependency between connected node pairs. In this way, the model can simultaneously aggregate node input features and label information from neighbor nodes. By introducing a shared compatibility matrix and edgewise scaling coefficients, we are able to effectively provide a flexible representation while avoiding overfitting. With the application of the EM algorithm and the surrogate piecewise objective in the M-step, we manage to leverage information from unobserved nodes and make the learning procedure tractable. We validate our analysis and model design with a series of experiments, which shows that the EPFGNN framework can effectively handle semi-supervised node classification problems for various graph datasets.

## Acknowledgements

This work was supported by the Munich Center for Machine Learning. We would like to thank Stephan Günnemann for helpful discussions.

## References

Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Advances in neural information processing systems*, pages 6530–6539, 2017.

Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 1802–1808, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. In *Advances in neural information processing systems*, pages 4558–4567, 2018.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*, 2017.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

Daphne Koller, Nir Friedman, Sašo Džeroski, Charles Sutton, Andrew McCallum, Avi Pfeffer, Pieter Abbeel, Ming-Fai Wong, David Heckerman, Chris Meek, et al. *Introduction to statistical relational learning*. MIT press, 2007.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, page 282–289, 2001.

Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, volume 8, 2012.

Sebastian Nowozin and Christoph H Lampert. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4): 185–365, 2011.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *8th International Conference on Learning Representations*, 2020.

Meng Qu, Yoshua Bengio, and Jian Tang. GMNN: graph markov neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *J. Complex Networks*, 9(2), 2021.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Charles Sutton and Andrew McCallum. Piecewise training for structured prediction. *Machine learning*, 77(2-3):165, 2009.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816, 2009.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations*, 2018.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, page 40–48, 2016.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.