Injecting Hierarchical Biological Priors into Graph Neural Networks for Flow Cytometry Prediction

Fatemeh Nassajian Mojarrad [*1] **Lorenzo Bini** [*1] **Thomas Matthes** [2] **Stéphane Marchand-Maillet** [1]

Abstract

In the complex landscape of hematologic samples such as peripheral blood or bone marrow derived from flow cytometry (FC) data, cell-level prediction presents profound challenges. This work explores injecting hierarchical prior knowledge into graph neural networks (GNNs) for single-cell multi-class classification of tabular cellular data. By representing the data as graphs and encoding hierarchical relationships between classes, we propose our hierarchical plug-in method to be applied to several GNN models, namely, FCHC-GNN, and effectively designed to capture neighborhood information crucial for single-cell FC domain. Extensive experiments on our cohort of 19 distinct patients, demonstrate that incorporating hierarchical biological constraints boosts performance significantly across multiple metrics compared to baseline GNNs without such priors. The proposed approach highlights the importance of structured inductive biases for gaining improved generalization in complex biological prediction tasks.

1. Introduction

Flow cytometry (FC) is a technology that provides rapid multi-parametric analysis of single cells in solution. The expression of specific cell surface and intracellular molecules is thereby detected by antibodies coupled to fluorochromes. Exposing labeled cells to a laser leads to the emission of fluorescence which is then collected by a detector. By analyzing more than millions of cells in a short time, information can thereby be collected about the presence or absence of up to fifty different cellular molecules per cell. As a result, histologists obtain tabular data where each row corresponds to a single cell and every column to a marker or feature (cell surface molecule – Table 4, in Appendix A).

This technique is employed in medicine to assess the cellular composition of complex body fluids like blood or bone marrow and to facilitate the diagnosis of hematologic diseases such as leukemia. In collaboration with our University's Hospital, we have built a dataset obtained by the analysis of bone marrow samples from 19 healthy/recovered patients with information obtained up to $1'512'610$ cells/sample about the presence or absence of twelve different surface molecules. Cellular processes are governed by complex hierarchical relationships and neighborhood interactions that are challenging to capture using traditional flat data representations. This strongly motivates our decision to apply GNNs, which have emerged as a powerful class of models capable of leveraging graph-structured data to reason about relationships and dependencies. By representing biological entities and their associations as nodes and edges in a graph, GNNs can effectively encode neighborhood information and propagate signals across the graph topology during training. However, even with this graph-based inductive bias, GNN models may still struggle to generalize well without additional guided constraints from biological domain knowledge.

In this work, we propose a novel approach to inject hierarchically-structured priors (biological in our case) into the GNN framework for multi-class prediction tasks on tabular data. Specifically, we encode the known hierarchical relationships between different cell types or functional classes as a tree-structured hierarchy imposed on the GNN output space. This hierarchical constraint serves as an inductive bias that encourages the GNN to respect the hierarchical dependencies inherent to the biological domain. During training, we use a custom hierarchical loss function that accounts for the hierarchical similarities between classes, in addition to the traditional cross-entropy loss. This approach ensures that the model's predictions not only accurately classify instances into their respective leaf nodes (specific cell types), but also respect the hierarchical groupings at higher levels

[*]Equal contribution [1]Department of Computer Science, University of Geneva, Switzerland [2]Hematology Service, Department of Oncology and Clinical Pathology Service, Geneva University Hospital, Switzerland. Correspondence to: Fatemeh Nassajian Mojarrad <Fatemeh.Nassajian@unige.ch>, Lorenzo Bini <Lorenzo.Bini@unige.ch>, Thomas Matthes <Thomas.Matthes@hcuge.ch>, Stéphane Marchand-Maillet <Stéphane.Marchand-Maillet@unige.ch>.

Proceedings of the ICML 2024 Workshop on Accessible and Efficient Foundation Models for Biological Discovery, Vienna, Austria. 2024. Copyright 2024 by the author(s).

of the taxonomy (broader cell lineages or functional categories). Through rigorous experiments on our real-world datasets, we demonstrate that incorporating hierarchical biological priors significantly boosts the performance of GNN models across a range of metrics. Our hierarchical GNN approach achieves consistent gains over strong baselines that do not leverage such structured domain knowledge. We provide in-depth analysis and empirical evidence highlighting the importance of encoding inductive biases aligned with the underlying data domains, especially for domains with rich hierarchical relationships like cell biology.

2. Related Work

The primary research focus in machine learning has predominantly been centered on developing models for conventional classification problems, wherein an object is assigned to a single class from a set of disjoint classes. However, a distinct subset of tasks involves scenarios where classes are not disjoint but rather organized hierarchically, giving rise to hierarchical classification (HC). The hierarchical structure formalizing the interrelation among classes can manifest as either a tree or a directed acyclic graph (DAG) (Silla & Freitas, 2011). Hierarchical classification problems manifest across a broad spectrum, spanning from musical genre classification (Ariyaratne & Zhang, 2012; Iloga et al., 2018) to the identification of COVID-19 in chest X-ray images (Pereira et al., 2020), the taxonomic classification of viral sequences in metagenomic data (Shang & Sun, 2021) and text categorization (Javed et al., 2021; Ma et al., 2022). Graph-based models can leverage both the global and local characteristics inherent in networks relevant to cell biology. GNNs have been recently adapted in various tasks, e.g. link prediction (Zhang & Chen, 2018), graph classification (Duvenaud et al., 2015; Lee et al., 2019), and node classification (Kipf & Welling, 2017; Velickovic et al., 2017; Yang et al., 2016).For FC data FlowCyt (Bini et al., 2024) represents the first publicly available benchmark, designed to test deep learning models for single-cell multi-class classification, highlighting the benefits of modeling the problem as a graph-structured challenge. In the context of HC problems, DiffPool (Ying et al., 2018) propose a differentiable graph pooling module to generate hierarchical representations of graphs using various GNNs in an end-to-end fashion. Moreover, in the biological domains, HC-GNNs have been widely applied for histological image classification (Hou et al., 2022), protein-protein interaction (Gao et al., 2023), molecules' properties prediction (Han et al., 2023) and for cancer diagnosis and prognosis in digital pathology (Pati et al., 2022).

3. Problem Description

In this research, we began by collecting raw data from the bone marrow of 19 patients who underwent a flow cytometric analysis for diagnostic purposes. All samples were processed by the Diagnostics laboratory of the University Hospital and no malignant disease was detected at this stage. Each patient's sample is primarily tabular data (see supplementary material for the futher details) composed by a $N \times D$ matrix of values, where $D = 12$ dimensions (see Table 4 in Appendix A for details). It is therefore equivalent to a D-dimensional point cloud that cytometrists visualize via 2D projections. The result of such operations is the definition and quantification of different cell populations according to their phenotype. Those populations of interest were defined according to the markers expression, and then grouped together into different categories as shown in Figure 1, serving as our main biological hierarchical prior in this study.

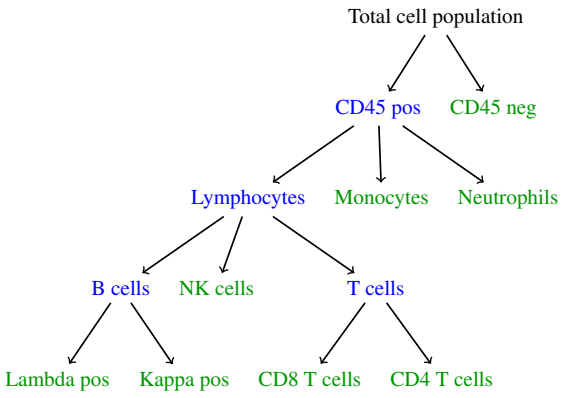

Figure 1. Depiction of our HC set up.

The main focus is to classify different cell types present in the heterogeneous samples into 12 classes, where 8 of them belong to leaf nodes (green ones), while 4 of them are parental nodes (blue ones).

4. Architecture of FCHC-GNN

Given a tabular dataset containing n data samples and m feature fields, denoted by $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n]^T$, the i-th sample in the table associated with m feature values $\mathbf{x}_i = [x_i^{(1)}, \cdots, x_i^{(m)}]$ and a discrete label y_i, part of a label hierarchy. Our learning goal is to find a mapping function f for which given $\mathbf{x}_i$, returns the predicted label $\hat{y}_i$. In the rest of this section, we will describe our plug-in method FCHC-GNN, and the architecture of the network for the underlying classification problem.

We consider an HC problem with a given set $\mathcal{C}$ of C classes,

which are hierarchically organized as a directed tree (Figure 1 for example). If there exists a path of non-negative length from class A_1 to class A_2 in the tree, we call A_2 a subclass of A_1. We assume to have a mapping $f_A : \mathbb{R}^m \to [0,1]$ for every class A such that $\mathbf{x} \in \mathbb{R}^m$ is predicted to belong to A whenever $f_A(\mathbf{x})$ is greater than or equal to some threshold. When A_2 is a subclass of A_1, the model should guarantee that $f_{A_1}(\mathbf{x}) \geq f_{A_2}(\mathbf{x})$, for all $\mathbf{x} \in \mathbb{R}^m$, to guarantee that the hierarchy constraint is always satisfied independently from the threshold. Note that in this model, every class is considered a subclass of itself. The case where $f_{A_1}(\mathbf{x}) < f_{A_2}(\mathbf{x})$ for some $\mathbf{x} \in \mathbb{R}^m$ and A_2 is a subclass of A_1, is called a *hierarchy violation* (Wehrmann et al., 2018). To simplify the notations, we will write $f_A(\mathbf{x})$ as f_A, thus removing the functional dependency of this term concerning $\mathbf{x}$.

Overall structure Given a set $\mathcal{C} = \{A_1, \cdots, A_C\}$ of C hierarchically structured classes A_c, we first build a neural network composed of two modules: a *Early Module* $\mathcal{H}$ with one output for each class in $\mathcal{C}$ and an upper module, referred to as the *Max Constraint Module* (MCM), consisting of a single layer that takes the output of the early module as input and imposes the hierarchy constraint.

If y_A is the ground truth label for class $A \in \mathcal{C}$ and the prediction value $\mathcal{H}_A \in [0,1]$ is obtained by $\mathcal{H}$ for class A, the MCM output for a class A is given by:

$$\text{MCM}_A = \max_{B \in \mathcal{S}_A} \mathcal{H}_B. \tag{1}$$

where $\mathcal{S}_A$ is the set of subclasses of A in $\mathcal{C}$. Note that since the number of operations performed by MCM_A is independent of the depth of the hierarchy, for each class $A \in \mathcal{C}$, the resulting model is scalable.

By construction, the following theorem therefore holds.

Theorem 4.1. *Let $\mathbf{x} \in \mathbb{R}^m$ be a data point. Let $\mathcal{C} = \{A_1, \cdots, A_C\}$ be a set of hierarchically structured classes and let $\mathcal{H}$ be a early module with outputs $\mathcal{H}_{A_1}, \cdots, \mathcal{H}_{A_C}$ ($\mathcal{H}_{A_c} \in [0,1]\ \forall c$) given the input $\mathbf{x}$, and $MCM_{A_1}, \cdots, MCM_{A_C}$ be defined as in equation (1). Then FCHC-GNNs do not admit hierarchy violations.*

Second, to exploit the hierarchy constraint during training, FCHC-GNNs are trained with *max constraint loss* (MCLoss), defined as follows

$$\text{MCLoss}_A = -y_A \ln(\max_{B \in \mathcal{S}_A} y_B \mathcal{H}_B) - (1 - y_A) \ln(1 - \text{MCM}_A),$$

for each class $A \in \mathcal{C}$. The global MCLoss is written as

$$\text{MCLoss} = \sum_{A \in \mathcal{C}} \text{MCLoss}_A. \tag{2}$$

The advantage of defining MCLoss instead of using the standard binary cross-entropy loss is that the more ancestors a class has, the more likely it is that FCHC-GNN trained with the standard binary cross-entropy loss would remain stuck in spurious local optima. The MCLoss prevents this from happening.

As a result, FCHC-GNN can delegate the prediction for a class A_i to one of its subclasses A_j, thanks to MCM and the overall hierarchical constrained loss function MCLoss. In fact, MCM_A can be viewed as the maximum score among subclasses of A being used within the MCLoss. Therefore, $\mathcal{H}_B$ are the prediction scores for subclasses used in computing both MCM_A and MCLoss. More formally, the following proposition holds by construction.

Proposition 4.2. *Under the same assumptions as in Theorem 4.1, considering a class $A_i \in \mathcal{C}$ and $A_j \in \mathcal{S}_{A_i}$ with $i \neq j$, FCHC-GNN delegates the prediction on A_i to A_j for $\mathbf{x}$, if $MCM_{A_i} = \mathcal{H}_{A_j}$ and $\mathcal{H}_{A_i} < \mathcal{H}_{A_j}$.*

Classification layer We have designed the FCHC module to be compatible with GAT, SAGE, and GCN, as highlighted in our code publicly accessible at https://github.com/VIPER-GENEVA/FCHC-GNN-Hierarchical[1], but due to its general nature it can be easily extended to broader GNNs. For the sake of simplicity, in the following discussion, we explain how we built the Early Module $\mathcal{H}$ for the classification as a Graph Attention Network (GAT), following (Velickovic et al., 2017). Given $\{\mathbf{x}_i\}_{i=1}^n$ as nodes, and $\mathbf{x}_i \in \mathbb{R}^m$, the node features are the input of the graph attention layer. The output of the layer is a new set of node features $\{\mathcal{H}_i\}_{i=1}^n$, where $\mathcal{H}_i \in \mathbb{R}^{\tilde{m}}$, for which $\tilde{m}$ might be different from m.

A GAT defines *attention coefficients* as importance scores of node $\mathbf{x}_j$'s features to that of node $\mathbf{x}_i$. W is the weight matrix used to build attention coefficients. *Masked attention* normalizes *attention coefficients* within every graph neighborhood $\mathcal{N}(\mathbf{x}_i)$. In our case, $\mathcal{N}(\mathbf{x}_i)$ represents the Euclidean k-nearest neighbours of node $\mathbf{x}_i$. The attention mechanism is a single-layer feedforward neural network, parameterized by a weight vector $\vec{a} \in \mathbb{R}^{2\tilde{m}}$, over which we apply the LeakyReLU non-linearity (see Fig. 3(a) in Appendix C).

The normalized attention coefficients γ_{ij} are then used to compute a linear combination of the corresponding features, after applying a nonlinearity σ, to serve as the final output features for each node.

We also move from single-head attention to multi-head attention (with L heads) by concatenating the output of all heads at every layer i (but the final layer). The expression for the final layer replaces concatenation by averaging and

[1]Check out VIPER official page for latest work https://viper-geneva.github.io/.

can be written in our notation as

$$\mathcal{H}_i = \sigma(\frac{1}{L}\sum_{l=1}^{L}\sum_{j\in\mathcal{N}(\mathbf{x}_i)}\gamma_{ij}^{l}W^{l}\mathbf{x}_j). \quad (3)$$

5. Experiments

In this section, we present the experimental results to illustrate the behaviour of ours FCHC-GNNs for the multi-class classification problem on our FC dataset. We also compare the proposed method with the respective flat versions (meaning without the FCHC module attached), to highlight how the injection of hierarchical prior knowledge really boost the performance for FC classification with respect to flat structures. Moreover, since they are known to be highly performant on tabular data, we also incuded Deep Neural Network both with (FCHC-DNN) and without (DNN) hierarchical module. We have also obtained competitive performance of our FCHC-GNN module on the public Imagenet1000-(Mini) dataset, where we used the subcategory of "animals"; these additional results are presented in Appendix D due to space constraints.

Furthermore, in Appendix E as ablation studies, we further validate the generality of our FCHC-GNN module by conducting experiments on other 30 different FC patients with a shallower depth of hierarchy. Across this cohort, we observe that injecting the hierarchical priors becomes necessary as the hierarchy grows deeper and more complex. For shallow hierarchies, the performance gains are modest, but they become substantially larger when dealing with intricate, multi-level hierarchies where using prior knowledge becomes fundamental.

Due to space constraints, see Appendix A for experimental setup, metrics definition and graph-construction details. Please see also Appendix E for model's ablation studies on different hierarchy scenario and hyperparameters details. Our hierarchical approach achieves consistent gains over same architectures that do not leverage such structured domain knowledge, as shown in Table 1, 2 and 3, where we compute hierarchical-metrics and the correct predicted classes across all the patients in the dataset. All the results are within the std $= \pm 0.1$, and experiments have been averages across four different seeds.

Table 1. Average metrics across all patients using different models for FCHC module.

Metrics	FCHC-GAT	FCHC-SAGE	FCHC-GCN	FCHC-DNN
HP	0.88	**0.92**	0.83	0.74
HR	**0.85**	0.83	0.81	0.83
HF	0.87	**0.88**	0.82	0.78

Table 2. Average ratios of correct predicted classes across all patients using different models for FCHC module.

Label	FCHC-GAT	FCHC-SAGE	FCHC-GCN	FCHC-DNN
CD45 pos	100.00	100.00	100.00	81.12
CD45 neg	100.00	100.00	100.00	83.43
Lymphocytes	85.21	**97.91**	82.14	81.06
Monocytes	90.31	93.42	**95.52**	90.07
Neutrophils	80.15	**93.38**	75.26	80.84
B Cells	94.86	97.49	**98.52**	85.83
NK Cells	97.48	97.50	**97.98**	94.01
T Cells	83.00	**97.84**	86.74	92.07
Lambda pos	97.77	**98.88**	96.52	86.45
Kappa Pos	96.56	98.61	**99.21**	98.00
CD8 T Cells	90.15	**95.26**	94.25	91.99
CD4 T Cells	89.81	91.63	**92.07**	90.06

5.1. Discussion of Our Results

The results presented in this paper underscore the novelty and effectiveness of our newly introduced FCHC-GNN framework, in performing node classification tasks on FC dataset. However, the presented approach is rather general and can be easily extend to broader applications of GNN, in case there is the need to leverages the inherent hierarchical structure of the data, a feature that frequently appears in real-word dataset. Indeed, as shown in Table 3 where the FCHC model has not been applied, injecting hierarchical biological priors allows the models to not overlook any of the leaf nodes during the prediction. As an example, in Table 2 and 3 we clearly see how the performances drops from 91.63% with hierarchical priors to a 67.80% without, on CD4 T cells for GraphSAGE model (Hamilton et al., 2017). Similarly, for NK cells the correct predicted labels drammatically goes down from 97.48% to 0.15% for the GAT, highlighting once again the need of including the FCHC module to caputure the biology complexity of these cells population.

Table 3. Average ratios of corrected predicted classes across all patients without using the FCHC module.

Label	GAT	SAGE	GCN	DNN
CD45 neg	34.94	98.25	97.10	**98.44**
Monocytes	0.75	**1.46**	0.50	-
Neutrophils	**100.00**	98.96	99.73	98.77
NK Cells	0.15	**1.57**	-	-
Lambda pos	-	0.88	0.07	**1.40**
Kappa Pos	0.24	-	0.10	**0.70**
CD8 T Cells	-	**20.41**	13.68	1.01
CD4 T Cells	0.20	**67.80**	40.08	31.95

Given the classification power of FCHC module, we took a deeper look at each model's performance, and to allow better visualization of the predicted cell patterns, we picked one random patient and we plotted its t-SNE embeddings (Van der Maaten & Hinton, 2008), as shown in Figure 5 of Appendix G. Moreover, we further investigated the interpretability of our module through the expression of feature importance, as resulted in Appendix B.

6. Conclusion and Future Work

We have presented a novel hierarchical GNN framework, namely FCHC-GNN, that injects structured biological priors to enhance multi-class prediction on FC cellular data. By encoding known hierarchical relationships between cell types and functional classes, our model effectively captures the rich dependencies innate to biological domains while operating on a tabular input representation. Extensive experiments across our FC dataset demonstrate substantially improved performance over strong baselines that fail to leverage such hierarchical constraints. Moreover, as demonstrated over well-known ImageNet data, our FCHC-GNN module is easily extendible to other domains where classes may be organized within a hierarchy.

The consistent gains validate our hypothesis that tailored inductive biases aligned with the underlying tabular data distributions are crucial for achieving better generalization. FCHC-GNN architectures introduce a novel method to enhance learning on tabular inputs with prior domain knowledge expressed as hierarchies. Looking ahead, this general approach holds promise for biomedical learning tasks beyond single-cell classification.

Acknowledgements

This work is partly funded by the Swiss National Science Foundation under grant number 207509 "Structural Intrinsic Dimensionality".

References

Ariyaratne, H. B. and Zhang, D. A novel automatic hierachical approach to music genre classification. In *Proceedings of the IEEE International Conference on Multimedia and Expo Workshops*, pp. 564–569. IEEE, July 2012.

Bini, L., Nassajian Mojarrad, F., Liarou, M., Matthes, T., and Marchand-Maillet, S. FlowCyt: A comparative study of deep learning approaches for multi-class classification in flow cytometry benchmarking. In *Conference on Health, Inference, and Learning (CHIL'24)*, 2024. URL https://arxiv.org/abs/2403.00024.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

Gao, Z., Jiang, C., Zhang, J., Jiang, X., Li, L., Zhao, P., Yang, H., Huang, Y., and Li, J. Hierarchical graph learning for protein–protein interaction. *Nature Communications*, 14(1):1093, 2023.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Han, S., Fu, H., Wu, Y., Zhao, G., Song, Z., Huang, F., Zhang, Z., Liu, S., and Zhang, W. Himgnn: a novel hierarchical molecular graph representation learning framework for property prediction. *Briefings in Bioinformatics*, 24(5):bbad305, 2023.

Hou, W., Huang, H., Peng, Q., Yu, R., Yu, L., and Wang, L. Spatial-hierarchical graph neural network with dynamic structure learning for histological image classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 181–191. Springer, 2022.

Iloga, S., Romain, O., and Tchuenté, M. A sequential pattern mining approach to design taxonomies for hierarchical music genre recognition. *Pattern Analysis and Applications*, 21(2):363–380, May 2018.

Javed, T. A., Shahzad, W., and Arshad, U. Hierarchical text classification of urdu news using deep neural network. https://arxiv.org/abs/2107.03141, 2021.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. https://openreview.net/pdf?id=SJU4ayYgl, 2017.

Kiritchenko, S., Matwin, S., Nock, R., and Famili, A. F. Learning and evaluation in the presence of class hierarchies: Application to text categorization. In *Proceedings of the Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 395–406, Québec, Canada, June 2006. Springer Berlin Heidelberg.

Lee, J., Lee, I., and Kang, J. Self-attention graph pooling. In *International conference on machine learning*, pp. 3734–3743. PMLR, 2019.

Ma, Y., Liu, X., Zhao, L., Liang, Y., Zhang, P., and Jin, B. Hybrid embedding-based text representation for hierarchical multi-label text classification. *Expert Systems with Applications*, 187:115905, May 2022.

Pati, P., Jaume, G., Foncubierta-Rodriguez, A., Feroce, F., Anniciello, A. M., Scognamiglio, G., Brancati, N., Fiche, M., Dubruc, E., Riccio, D., et al. Hierarchical graph representations in digital pathology. *Medical image analysis*, 75:102264, 2022.

Pereira, R. M., Bertolini, D., Teixeira, L. O., Jr, C. N. S., and Costa, Y. M. Covid-19 identification in chest x-ray images on at and hierarchical classification scenarios. *Computer Methods and Programs in Biomedicine*, 194: 105532, October 2020.

Shang, J. and Sun, Y. Cheer: hierarchical taxonomic classification for viral metagenomic data via deep learning. *Methods*, 189:95–103, May 2021.

Silla, C. N. and Freitas, A. A. A survey of hierarchical classifcation across different application domains. *Data Mining and Knowledge Discovery*, 22(1):31–72, 2011.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *stat*, 1050 (20):10–48550, 2017.

Wehrmann, J., Cerri, R., and Barros, R. Hierarchical multi-label classification networks. In *International conference on machine learning*, pp. 5075–5084. PMLR, July 2018.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, June 2016.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

A. Description of Our Dataset

The raw data obtained from the cytometer are saved in Flow Cytometry Standard (FCS) files. Each FCS file comprises multidimensional data that corresponds to more than one million of cells, with each cell characterized by various parameters such as size, granularity, and fluorescence intensity, giving them the structure of tabular data (see Table 4 for a complete list of markers in our data).

Table 4. Flow cytometry data markers.

MARKER #	MARKER	EXPLANATION
0	FS INT	FORWARD SCATTER (FSC) - CELL'S SIZE
1	SS INT	SIDE SCATTER (SSC) - CELL'S GRANULARITY
2	CD14-FITC	CLUSTER OF DIFFERENTIATION 14 - ANTIGEN
3	CD19-PE	CLUSTER OF DIFFERENTIATION 19 - ANTIGEN
4	CD13-ECD	CLUSTER OF DIFFERENTIATION 13 - ANTIGEN
5	CD33-PC5.5	CLUSTER OF DIFFERENTIATION 33 - ANTIGEN
6	CD34-PC7	CLUSTER OF DIFFERENTIATION 34 - ANTIGEN
7	CD117-APC	CLUSTER OF DIFFERENTIATION 117 - ANTIGEN
8	CD7-APC700	CLUSTER OF DIFFERENTIATION 7 - ANTIGEN
9	CD16-APC750	CLUSTER OF DIFFERENTIATION 16 - ANTIGEN
10	HLA-PB	HUMAN LEUKOCYTE ANTIGEN
11	CD45-KO	CLUSTER OF DIFFERENTIATION 45 - ANTIGEN

The synthesis of machine learning and biological domain knowledge holds potential for advancing scientific understanding and biomedical applications. While machine learning models excel at finding intricate patterns in data, their performance can be limited without the inductive biases, in particular for tabular data which typically lacks of it, and constraints derived from prior biological principles.

A.1. Hierarchical Metrics Definition and Graph-Construction

In our experiments, we construct the k-nearest neighbors graphs with $k = 7$ neighbors. We perform a 7-fold test procedure where 1/7th of the data (2 or 3 patients out of 19) is held out as a test set and the rest (6/7th of data) as a training set. We then perform a 4-fold cross validation procedure within the training set to set the hyperparameters.

The metrics are reported using the extensions of the renowned metrics of precision, recall and F-score, but tailored to the HC setup. We implement the metrics of hierarchical precision (hp), hierarchical recall (hr) and hierarchical F-score (hf) defined by (Kiritchenko et al., 2006):

$$hp = \frac{\sum_i |\alpha_i \cap \beta_i|}{\sum_i |\alpha_i|},\ hr = \frac{\sum_i |\alpha_i \cap \beta_i|}{\sum_i |\beta_i|},\ hf = \frac{2 \times hp \times hr}{hp + hr}. \tag{4}$$

where α_i is the set consisting of the most specific classes predicted for test sample i and all their ancestor classes and β_i is the set containing the true most specific classes of test sample i and all their ancestors.

B. Interpretability and Feature Importance Analysis

An important aspect of our FCHC plug-in module is its interpretability, as demonstrated in Figure 2 for the FCHC-GAT version.

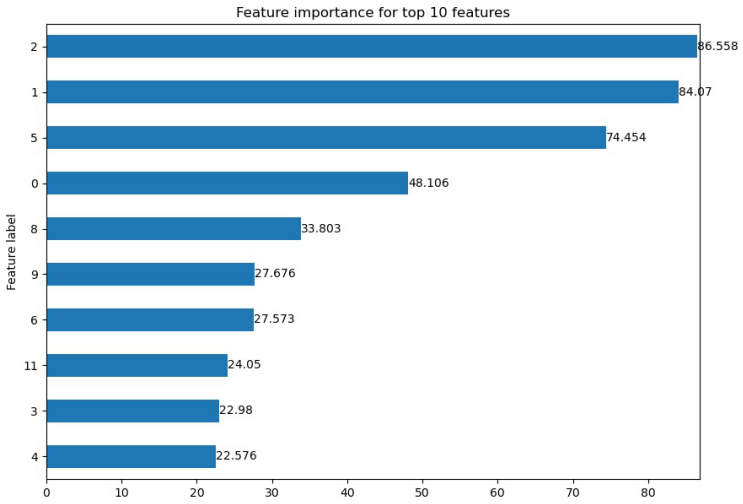

Figure 2. Feature importance for FCHC-GAT attributed by $\mathcal{H}$. Feature labels correspond to that of Table 4.

This feature importance analysis not only provides insights into the decision-making process of our model but also paves the way for potential improvements and adaptations of our model for other applications. As we can see, the features that are given the most importance (at least above 40%) are 2,1,5 and 0, therefore we broke down the choice of the model for each one. In order of importance, we had:

- CD14-FITC: This feature reflects the level of CD14 expression, a receptor found on the cell surface that interacts with lipopolysaccharide (LPS), a component of bacterial cell walls. Predominantly found on monocytes, macrophages, and activated granulocytes, CD14 plays a pivotal role in mediating the immune response to bacterial infections. The high weightage given to this feature suggests that our model is adept at distinguishing between cells involved in innate immunity and those that aren't, thereby potentially detecting the presence of bacterial infection in the sample.
- Side Scatter (SSC)-Cell's granularity: This feature measures the level of light scatter at a 90-degree angle to the laser beam, reflecting the internal complexity or granularity of the cell. This could include the presence of granules, nuclei, or other organelles. The high weightage of this feature implies that our model can differentiate between cells of varying complexity, such as lymphocytes, monocytes, and granulocytes, or identify cells with abnormal granularity, such as blast cells or malignant cells.
- CD33-PC5.5: This feature indicates the expression level of CD33, a cell surface receptor of the sialic acid-binding immunoglobulin-like lectin (Siglec) family. CD33 is expressed by myeloid cells like monocytes, macrophages, granulocytes, and mast cells, and it modulates the immune response by inhibiting the activation of these cells. The high weightage of this feature suggests that our model can distinguish between myeloid and non-myeloid cells, or detect the expression of CD33 as a marker for certain types of leukemia, such as acute myeloid leukemia (AML) or chronic myelomonocytic leukemia (CMML).
- Forward scatter (FSC)-Cell's size: This feature measures the level of light scatter along the path of the laser beam, which is proportional to the diameter or surface area of the cell. This can be used to discriminate cells by size. The importance placed on this feature suggests that our model can differentiate between cells of different sizes, such as small lymphocytes and large monocytes.

C. Graphical Representation of Multi-Head Attention

Figure 3 works as recap of the structural representation for the applied multi-head attention.

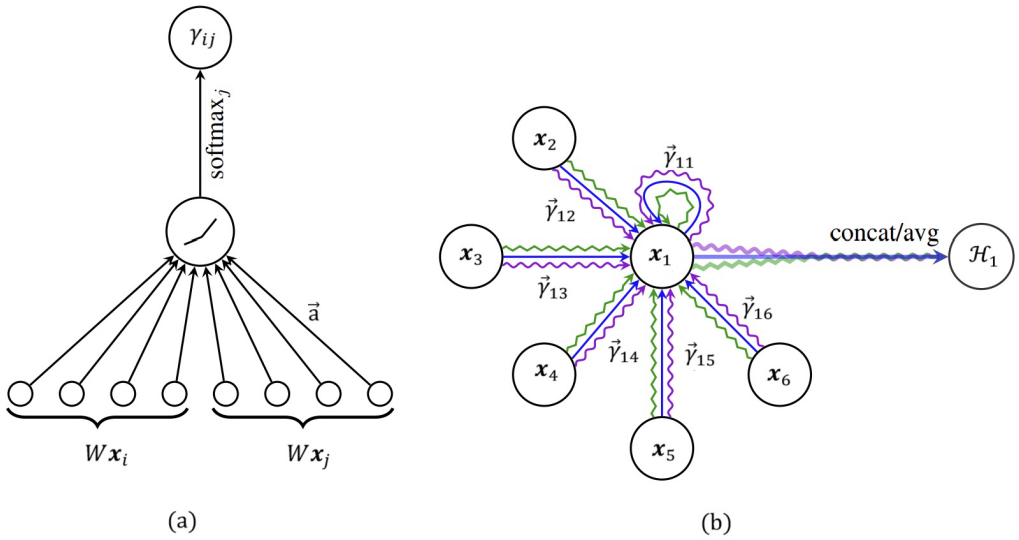

Figure 3. (a): Computing the normalized attention coefficients γ_{ij} (b): Multi-head attention of node 1 on its neighborhood. Arrows show concatenation or averaging of attention (adapted from (Velickovic et al., 2017)).

D. ImageNet1000-Mini Results

Table 5 shows a competitive performance of our FCHC-GNN module on hierarchical image classification task as well. For the sake of simplicity, we only showed our plug-in module being applied to a GAT, against its normal version and the average state-of-art performances (according to what was written on the benchmark `https://paperswithcode.com/sota/image-classification-on-imagenet`).

Table 5. Average accuracy percentage of SOTA methods on Imagenet1000-(Mini). Results are averaged with a std $= \pm 0.01$ across four different seeds.

MODEL	IMAGENET1000-MINI
FCHC-GAT (OURS)	91.88
GAT	3.91
AVERAGE SOTA	88.50 - **92.40**

E. Ablation Studies on a Shallow Experiments Setup

To compare the results given in Table 1 - 3, we present the results on the second cohort of FC patients, provided by our Univerisity Hospitaly, where 30 different bone-marrows samples have been collected and anylized in the same way as the previous nineteens. The depth of hierarchy is shallower than that of the main one, where only one step of parental nodes have been considered, as depicted in Figure 4. As shown in Table 7 and Table 9 on shallow hierarchies, the problem more closely resembles a flat classification task, and thus baseline GNN performance remains relatively good. This is especially true for datasets with large populations of certain cell types, where the inherent class imbalance acts as an inductive bias, as depicted from the hierarchical F-Score results on Table 8. However, when the hierarchies grow deeper and more complex, the use of our FCHC-GNN module becomes strictly necessary to obtain better interpretability and generalization. For example, Table 6 already shows a consistant improvement on the hF-Score metrics with respect to standard one without the hierarchical module, mainly due to the correct prediction of Myeloid HSPC and other minor HSPC cells.

Deeper hierarchies exacerbate the issue of conflicting gradients between nodes at different hierarchy levels during training. Without explicit hierarchical constraints, as in the MCLoss, GNNs struggles to simultaneously optimize for accurate

predictions at the leaf node level while respecting parental groupings (higher-level). Our hierarchical module resolves this tension without suffering of extra time-consumation problem as explained in Appendix F.3, allowing attached models to leverage hierarchical priors as a strong inductive bias for learning more interpretable representations aligned with the known biological taxonomy.

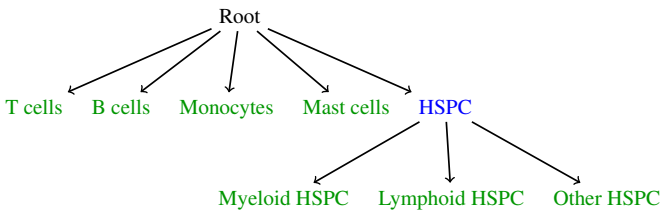

Figure 4. Depiction of the shallow FC setup.

Table 6. Average metrics across 30 patients using different models for FCHC module.

METRICS	FCHC-GAT	FCHC-SAGE	FCHC-GCN	FCHC-DNN
HP	0.94	**0.97**	0.95	0.87
HR	0.93	**0.97**	0.94	0.83
HF	0.94	**0.97**	0.94	0.85

Table 7. Average ratios of correct predicted classes across 30 patients using different models for FCHC module.

LABEL	FCHC-GAT	FCHC-SAGE	FCHC-GCN	FCHC-DNN
T CELLS	93.87	**98.85**	94.49	75.04
B CELLS	95.45	**98.28**	96.03	86.82
MONOCYTES	98.12	**99.21**	98.57	95.24
MAST CELLS	99.79	**99.93**	99.79	99.79
HSPC	95.03	**97.85**	95.73	91.03
MYELOID HSPC	94.54	**96.51**	94.79	93.40
LYMPHOID HSPC	97.40	**98.17**	97.40	97.40
OTHER HSPC	99.46	99.46	99.46	99.46

Table 8. Average metrics across 30 patients without using the FCHC module.

METRICS	GAT	SAGE	GCN	DNN
PRECISION	0.92	0.84	0.92	0.84
RECALL	0.76	0.74	**0.81**	0.71
F1 SCORE	0.75	0.64	**0.79**	0.61

Table 9. Average ratios of corrected predicted classes across 30 patients without using the FCHC module.

LABEL	GAT	SAGE	GCN	DNN
T CELLS	81.83	**98.72**	90.14	96.91
B CELLS	51.50	10.96	**57.22**	9.87
MONOCYTES	**82.89**	69.26	80.89	56.19
MAST CELLS	**98.22**	86.51	52.62	53.33
MYELOID HSPC	**26.16**	-	12.37	-
LYMPHOID HSPC	**90.75**	12.96	84.50	26.43
OTHER HSPC	23.11	-	**32.48**	-

F. FCHC Plug-In Module Implementation

This section provides additional details on how the hierarchical constraint module was implemented in all the tested models. For sake of simplicity, we choose to illustrate how we designed the FCHC-GAT module, while for other FCHC-GNNs implementations refer to our GithHub repository https://github.com/VIPER-GENEVA/FCHC-GNN-Hierarchical. Here are the key steps:

- It first defines the hierarchical structure by creating a directed graph g from the class hierarchy string *ATTRIBUTE_class*, as shown in Listing 1 below. This maps out the parent-child relationships between classes.
- It converts this graph to a matrix R that encodes the ancestor-descendant relationships, where $R[i, j] = 1$ if class i is descendant of class j, 0 otherwise.
- Then the model includes GAT layers, where each of them does message passing and computes attention coefficients between neighboring nodes.
- During training, the raw GAT predictions are returned.
- During inference, the predictions are passed through a constraint function *get_constr_out* that takes the max over descendants to enforce the hierarchy, as depicted in Listing 2 below.
- The loss function is a modified cross-entropy loss MCLoss that also lets subclasses inform about their parent classes predictions.
- The model takes a graph and node features as input, does message passing and attention in the GAT layers, applies the hierarchical constraint, and outputs a prediction vector for each node.

In summary, it implements a hierarchical classifier using a GAT augmented with constraints and loss terms to incorporate the class hierarchy information. The hierarchy structure is predefined and encoded in the constraint matrix R.

F.1. Constraint Layer

The hierarchy consistency is imposed through a constraint layer added on top of the base classifier network. This constraint layer, referred to as the Max Constraint Module (MCM), takes the output predictions from the base network and ensures the hierarchical constraint is satisfied. The MCM output for any class A is computed as:

$$\text{MCM}_A = \max(\mathcal{H}_B \text{ where } B \text{ is subclass of } A), \tag{5}$$

where $\mathcal{H}_B$ is the prediction value from the base network for class B. This takes the maximum over all subclasses of A, guaranteeing that the prediction for class A will be greater than or equal to the prediction for any of its subclasses. By constructing the output this way, the hierarchy constraint is inherently satisfied regardless of the threshold used downstream.

```
# List all the classes
to_skip = ['root']
ATTRIBUTE_class = "1,1_1,1_1_1,1_1_1_1,1_1_1_1,1_1_1_2,1_1_2,1_1_3,1_1_3_1,1_1_3_2,1_2,1_3,2"

# Store nodes and direct connections
g = nx.DiGraph()
for branch in ATTRIBUTE_class.split(','):
    term = branch.split('_')
    if len(term)==1:
        g.add_edge(term[0], 'root')
    else:
        for i in range(2, len(term) + 1):
            g.add_edge('.'.join(term[:i]), '.'.join(term[:i-1]))

nodes = sorted(g.nodes(), key=lambda x: (len(x.split('.')),x))

AA = np.array(nx.to_numpy_array(g, nodelist=nodes))

# Compute matrix of ancestors R
# Given C classes, R is an (C x C) matrix
# where R_ij = 1 if class i is descendant of class j
R = np.zeros(AA.shape)
np.fill_diagonal(R, 1)
gg = nx.DiGraph(AA) # AA is the matrix where the direct connections are stored
for i in range(len(AA)):
    ancestors = list(nx.descendants(gg, i)) # Need to use the function nx.descendants()
    #because in the directed graph the edges have source
    #from the descendant and point towards the ancestor
    if ancestors:
        R[i, ancestors] = 1
R = torch.tensor(R)
#Transpose to get the descendants for each node
R = R.transpose(1, 0)
R = R.unsqueeze(0).to(device)
```

Listing 1. Python code snippet generating a matrix of ancestors R based on the class relationships in the directed graph. Each class is represented as a node, and edges define parent-child connections.

F.2. Constrained Loss Function

The loss function used during training is the max constraint loss (MCLoss), defined as:

$$\text{MCLoss}_A = -y_A \log\left(\max_{B \text{ in subclasses_of_A}} (y_B \cdot \mathcal{H}_B)\right) - (1 - y_A)\log(1 - \text{MCM}_A), \tag{6}$$

where y_A is the ground truth label. This loss allows the model to leverage the hierarchical information by letting predictions from subclasses inform the training for parent classes.

F.3. Time Complexity

The time complexity of the constraint layer is $O(C)$ where C is the number of classes, since it just takes a maximum over the subclasses for each class. So it scales linearly with the size of the hierarchy and does not depend on the depth. The overall model maintains the same asymptotic time complexity as the base classification network used, with just an additional linear factor for the constraint layer. Hence, the hierarchical structure does not add significant overhead during training or inference.

```python
def get_constr_out(x, R):
    ''' Given the output of the graph neural network x returns the output of
    MCM given the hierarchy constraint expressed in the matrix R '''
    c_out = x.double()
    c_out = c_out.unsqueeze(1)
    c_out = c_out.expand(len(x),R.shape[1], R.shape[1])
    R_batch = R.expand(len(x),R.shape[1], R.shape[1])
    final_out, _ = torch.max(R_batch*c_out.double(), dim = 2)
    return final_out

# Define the FCHC plug-in module
class FCHC_GAT(nn.Module):
    ''' During training it returns the not-constrained output that is then passed to MCLoss'''
    def __init__(self,R):
        super(FCHC_GAT, self).__init__()
        self.R = R
        self.nb_layers = 2
        self.num_heads = 2
        self.out_head = 2
        self.hidden_dim = 32
        self.input_dim = 12
        self.output_dim= len(set(ATTRIBUTE_class.split(',')))+1  # We do not evaluate
        #the performance of the model on the 'root' node
        gat_layers = []
        for i in range(self.nb_layers):

            if i == 0:
                gat_layers.append(GATLayer(self.input_dim, self.hidden_dim,
                self.num_heads,True,0.4))
            elif i == self.nb_layers - 1:
                gat_layers.append(GATLayer(self.hidden_dim*self.num_heads,
                self.output_dim, self.out_head, False,0.2))
            else:
                gat_layers.append(GATLayer(self.hidden_dim*self.num_heads,
                self.hidden_dim, self.num_heads, True,0.2))
        self.gat_layers = nn.ModuleList(gat_layers)

        self.sigmoid = nn.Sigmoid()
        self.f = nn.ReLU()
        self.reset_parameters()
        self.drop = nn.Dropout(0.2)

    def reset_parameters(self):
        for gat_layer in self.gat_layers:
            gat_layer.reset_parameters()

    def forward(self, data):
        x, edge_index= data.x, data.edge_index
        for i in range(self.nb_layers):
            x = self.gat_layers[i](x, edge_index)
            if i != self.nb_layers - 1:
                x = self.f(x)
                x = self.drop(x)
            else:
                x= self.sigmoid(x)
        if self.training:
            constrained_out = x
        else:
            constrained_out = get_constr_out(x, self.R )
        return constrained_out
```

Listing 2. FCHC-GAT plug-in module definition with hierarchical constraints. The model incorporates the hierarchy matrix R to enforce class relationships during training. After performing message passing and attention computations, the model's output is constrained during inference using the *get_constr_out* function.

G. t-SNE Embedding Visualization of FCHC-GAT Module

An important aspect of our FCHC-GAT model that warrants special attention is its ability to accurately capture and classify even small populations of certain cell classes. This is a significant achievement, as demonstrates the recall and precision of our model. The ability of FCHC-GAT is further illustrated through visualization of a particular sub-population of cells, particularly interested for physicians. As shown in Figure 5, the t-SNE scatter plots provides a visual testament to the model's ability to accurately classify different cell types of interested, in this case NK Cells, CD4 T Cells, Kappa Pos, Lambda Pos, Monocytes, Neutrophils and CD8 T cells.

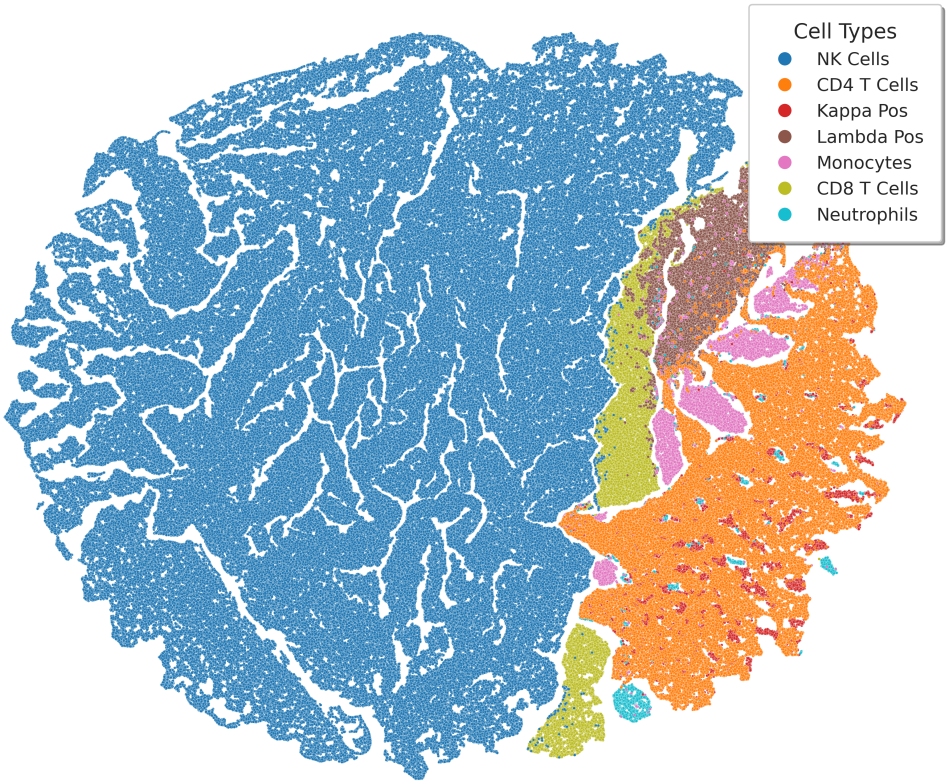

Figure 5. Embeddings t-SNE projection for patient 7, for the FCHC-GAT module.