2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

Classification of Intestinal Gland Cell-Graphs Using Graph Neural Networks

Linda Studer[*†‡§], **Jannis Wallau**[*§], **Heather Dawson**[§], **Inti Zlobec**[§], **and Andreas Fischer**[†‡]

[†]*Document Image and Video Analysis (DIVA) Research Group*
University of Fribourg, Switzerland
{firstname}.{lastname}@unifr.ch

[‡]*Institute of Complex Systems (iCoSys)*
University of Applied Sciences and Arts Western Switzerland
{firstname}.{lastname}@hefr.ch

[§]*Institute of Pathology*
University of Bern
{firstname}.{lastname}@pathology.unibe.ch

2020 25th International Conference on Pattern Recognition (ICPR) | 978-1-7281-8808-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPR48806.2021.9412535

***Abstract*—We propose to classify intestinal glands as normal or dysplastic using cell-graphs and graph-based deep learning methods. Dysplastic intestinal glands can lead to colorectal cancer, which is one of the three most common cancer types in the world. In order to assess the cancer stage and thus the treatment of a patient, pathologists analyse tissue samples of affected patients. Among other factors, they look at the changes in morphology of different tissues, such as the intestinal glands. Cell-graphs have a high representational power and can describe topological and geometrical properties of intestinal glands. However, classical graph-based methods have a high computational complexity and there is only a limited range of machine learning methods available. In this paper, we propose Graph Neural Networks (GNNs) as an efficient learning-based approach to classify cell-graphs. We investigate different variants of so-called Message Passing Neural Networks and compare them with a classical graph-based approach based on approximated Graph Edit Distance and k-nearest neighbours classifier. A promising classification accuracy of 94.8% is achieved by the proposed method on the pT1 Gland Graph dataset, which is an increase of 11.5% over the baseline result.**

I. Introduction

Pathologists consider many different aspects of tissue, such as density of certain cells, morphological changes, and the spatial relationship between cell (sub-)types. For example, dysplasia is a morphological correlate of progression towards cancer defined by certain cytological and architectural features, such as nuclear enlargement, crowding, stratification, mucin depletion and complex architecture. Dysplasia of intestinal glands is especially important in pT1 colorectal cancer, which refers to the earliest stage of invasive colorectal cancer. The invasive cancer arises from a precursor termed adenoma, which by definition is dysplastic (either low- or high-grade) [1].

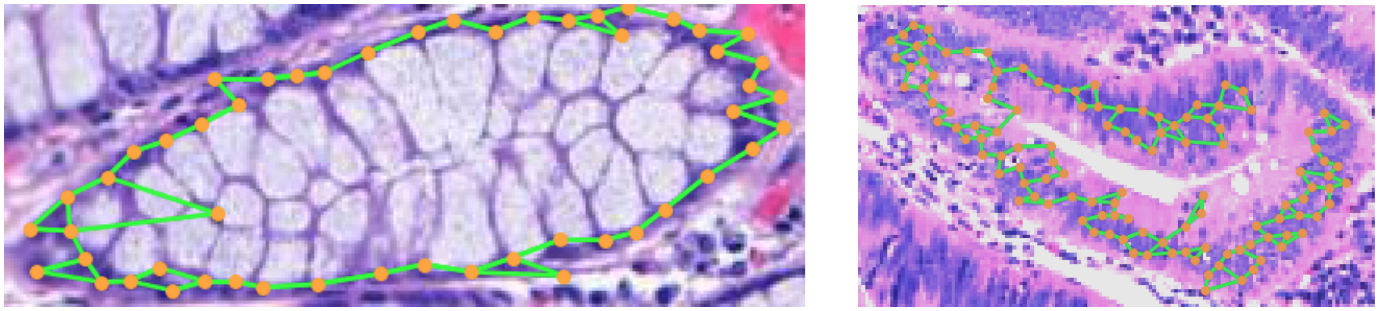
In diagnostics, pathologists look at thin cuts of tissue under the microscope, which are stained to highlight the morphological features. These tissue slides can be scanned as high-resolution images and used for digital pathology and computer-aided diagnosis. Unlike photographs, these images are rotational equivariant since the tissue they depict does not have an orientation. This can be challenging for image-based methods [2], [3].

Graph-based methods however do not face this challenge, as graphs are not oriented. Graph-based representations have the ability to capture the geometrical and topological properties of the tissue morphology orientation-independent and in a more abstract way. They have become popular in the field of digital pathology and have been used to solve a range of different tasks, such as classification, segmentation and Content-Based Image Retrieval (CBIR) [4]. In mathematics, a graph G is defined as a tuple of (N, E, α, β), where N is the finite set of nodes (or vertices), E is the set of edges, α is the node labelling function and β is the edge labelling function. The labelling functions define the node and edge features, which are usually real-valued numbers.

In our previous work [5], we published the pT1 Gland Graph (pT1-GG) dataset, which consists of cell-graphs based on histopathological images of normal and dysplastic intestinal glands. Cell-graphs are a popular graph type in histopathology [6]–[8]. In these type of graphs, each node represents a single cell, or nucleus. To establish a baseline, we used an approximation of the upper bound of the Graph Edit Distance (GED), i.e. an improved version of the bipartite graph-matching method (BP2) [9], coupled with a k-nearest neighbours (k-NN) classifier to perform the classification, as well as forward search feature selection. Inexact graph matching methods are a long-standing and big part of structural pattern recognition, as they provide an error-tolerant similarity measure between two graphs [10]. This approach is the current state-of-the-art (SOTA), and achieves a performance of 83.3%.

Deep learning has become state-of-the art in many computer vision tasks. However, until recently, it has been limited to data from the Euclidean domain [11], which is data sampled on a grid, e.g. images. Scarselli et al. [12] were the first to

[*] These authors contributed equally to this work.



(a) Normal gland (b) Dysplastic gland

Fig. 1: Examples of cell-graphs in the pT1 Gland Graph dataset overlaid on the H&E image. Cells are represented as nodes in the graph (in orange) and are connected with edges (in green) based on the physical distance between them.

introduce what they called a Graph Neural Network (GNN), which extended the application of deep learning to graphs. Since then, many more methods for graph convolution have been developed [13]. This new research domain is sometimes also referred to as geometric deep learning, which is an umbrella term for deep neural models that can learn from non-Euclidean data, such as graphs. GNNs have also been applied to a number of tasks in digital pathology, such as region-of-interest classification [14], detection of malignancy and invasiveness [7], as well as survival-analysis [15].

However, only little research has been done so far to understand the properties and limitations of GNNs [16]. In this paper, we investigate eight different GNN architectures for classifying cell-graphs, which encompass a range of recently introduced graph convolutional layers. We provide an introduction to the general Message Passing Neural Network method, as well as a detailed description of the layers and parameters used. The performance and behaviour of the GNNs are studied on the pT1-GG benchmark dataset.

We first describe the dataset in section II, the network architectures in section III, and the experimental evaluation in section IV, before we draw some conclusions in section V and give an outlook to future lines of research.

II. The pT1 Gland Graph Dataset

The pT1-GG dataset [5] is publicly available[1] and consist of 520 cell-graphs based on images of intestinal glands from 20 pT1 colorectal cancer patients. Tissue samples of pT1 colorectal cancer patients always contain healthy gland tissue as well as dysplastic glandular areas. In a cell-graph [6], each cell is represented as a node in the graph.

The images used to create the graphs are scanned tissue slides that were stained with Hematoxylin and Eosin (H&E). The H&E staining [17] is a routine staining in histopathology. It is used to distinguish different cell and tissue structures. The cell nuclei are stained blue by hematoxylin and the extracellular matrix and cytoplasm is stained pink by eosin. Other structures have a varying degree of staining in between. Based on this staining, individual cells were detected and

[1]https://github.com/LindaSt/pT1-Gland-Graph-Dataset

TABLE I: List of the available node features in the pT1-GG dataset based on the segmented cells of each intestinal gland (total of 33). They are either based on the whole cell, just the cytoplasm, just the nucleus or other factors. The features used by the GED-baseline [5] are denoted with a star (*).

Based on	Available Features
Cell	- Eosin stain mean, std, min and max - Circularity - Eccentricity - Perimeter - Area - Caliper min and max
Cytoplasm	- Eosin mean, std, min* and max
Nucleus	- Circularity - Eosin stain sum, mean, std, min, - max and rage - Hematoxylin stain sum, mean*, std, min*, max*, and range - Area - Caliper min and max - Perimeter - Eccentricity
Misc	- Nucleus/cell area ratio

for each cell 33 features (see Table I) are available to be used as node features. The features are normalised using Z-normalisation, i.e. each node feature value is normalised such that $z = \frac{x-\mu}{\sigma}$ where x is the node features value, and μ and σ are the mean and standard deviation of the respective feature based on the training set. Edges are inserted between nodes based on the proximity, each node is connected to its two spatially closest neighbours. For those networks that can use edge features, the distance between the nodes is added as an edge feature, which are also z-normalised.

The dataset is annotated for a binary classification task, glands are either classified as normal or dysplastic. Figure 1 shows an example for each class, where the cell-graph is over-layed onto the original H&E image. The dataset contains graphs with a number of nodes in the range of 16 and 639, the median graph size is 65 (also see Table II).

TABLE II: Overview of the graphs size distribution and number of patients of the four cross-validation splits of the pT1 Gland Graph dataset. Each split contains 65 normal and dysplastic gland cell-graphs (total of 130 graphs per split).

Dataset split	# Patients	# Nodes per graph		
		min	max	median
1	5	17	386	67
2	5	16	535	61
3	5	19	639	61
4	5	16	434	75
Overall	20	16	639	65

TABLE III: Overview of the mathematical notations used in this paper.

Notation	Description
A	the adjacency matrix A
e_{ij}	feature vector of edge between n_i and n_j
F	feed-forward neural network
G	graph
I	identity matrix
i, j	node index
k	message passing iteration / layer index
K	total number of message passing layers
m_i^k	output of a message passing iteration
M	message passing function
$N(i)$	neighbours of node i
$\|N(i)\|$	degree of node i
R	readout function
σ	non-linearity function (e.g. ReLU)
v_G	feature vector of the whole graph
W^k	learnable weight matrix of the k^{th} layer
x_i^k	feature vector / hidden state of a node

III. Graph Neural Networks

In this section, we introduce the concept of Message Passing Neural Networks (MPNNs), followed by a description of the different GNN architectures considered for intestinal gland classification. Table III provides an overview of the mathematical notations and figure 2 gives a schematic overview of the network architectures.

A. Message Passing Neural Networks

As in CNN, where the local neighbourhood of a pixel is aggregated during convolution, graph convolution creates a hidden representation for each node based on its adjacent nodes.

MPNNs aggregate the information of the neighbourhood of every node in a graph G using so-called messages. A message m_i^{k+1} for a given node i is constructed using a message function M^k and an aggregation function β which together aggregate information from the local neighbourhood of node i. The message is then used as input for an update function U^k which updates the node's representation x_i^k. The first graph convolution layer aggregates the information of the 1-hop neighbourhood. By adding multiple such layers, the aggregation is extended to the k-hop neighbourhood, where k is the number of graph convolutional layers.

TABLE IV: Overview of the number of parameters in each of the Graph Neural Network architectures, as well as the Convolutional Neural Network (CNN) baseline.

	Number of Parameters	
	# Node Features	
	4 (Baseline)	33 (All)
1-GNN	$29'570$	$33'282$
GAT	$21'506$	$23'362$
GCN	$21'122$	$22'978$
GCN-JK	$45'698$	$47'552$
GIN	$33'602$	$35'458$
GraphSAGE	$21'122$	$22'978$
GraphSAGE-JK	$45'698$	$47'554$
ENN	$199'910$	$240'742$
VGG-16	$138'365'992$	

These K graph convolution layers are considered as the message passing phase. One single message passing layer can mathematically be formulated as

$$m_i^{k+1} = \beta_{j \in N(i)}(M^k(x_i^k, x_j^k, e_{ij})) \tag{1}$$

$$x_i^{k+1} = U^k(x_i^k, m_i^{k+1}), \tag{2}$$

where β is the permutation invariant aggregation scheme (e.g. sum, mean, max) and $N(i)$ denotes the neighbouring nodes of node i. For the first iteration, the hidden node state is simply the node feature vector. For example, a very simple way of doing this is taking the mean of the node feature vectors of the node itself and all the neighbouring nodes. Edge features can be included in this process as well, however, not many GNN architectures consider them.

After k iterations of message passing, in order to get a graph classification, the information over the whole graph is collected, summarised into a vector and forwarded to a linear classification layer. In this so-called read-out phase, a readout function R returns a feature vector v_G that is representative for the whole graph:

$$v_G = R(x_i^K | i \in G). \tag{3}$$

There are different options to achieve this, a very simple one is to take the average over all the feature vectors. Often, multiple methods are used and their output is concatenated.

Generally, the read-out function only considers the hidden node representations after the last message passing iteration. The so-called Jumping Knowledge (JK) [18] connections extend the read-out phase to include the hidden representations of the previous layers. All the intermediate representations with different k-hop neighbourhoods can be used and combined for the computation of the last hidden state for each node. This allows the model to have a different neighbourhood size for each node. JK is sometimes also referred to as skip connections.

B. Graph Neural Network Architectures

In the following, we describe in more detail the GNN architectures considered for cell-graph classification. They only differ in the way they construct the messages, update the node's hidden representation and in how they obtain the graph-level feature vector, while still following the concept of a message passing and readout phase [19]. Figure 2 gives a schematic overview of the architectures, and table IV gives an overview of the number of parameters in each architecture. In GNNs, the number of parameters is dependent on the number of node and edge features, as their aggregation is learned.

1) Graph Convolutional Network: Kipf and Wellings Graph Convolutional Network (GCN) [20] is a spectral-based GNN. Spectral-based models define the graph convolution in the Fourier space, where the Fourier transform of a graph signal is multiplied with a spectral filter. To avoid the computationally expensive eigen-decomposition needed for the graph Fourier transform, GCN simplifies the spectral graph convolution by using a first-order approximation of Chebyshev polynomials. This results in spatially localized filters and places the GCN right at the boarder between spectral- and spatial-based approaches. The updated node feature matrix $X \in \mathbb{R}^{N \times C}$ (where C is the number of node features, and N is the number of nodes) after one layer is

$$X^{(k+1)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}X^k W), \tag{4}$$

where $\hat{A} = A + I$ is the adjacency matrix with inserted self-loops, $\hat{D}_{ii} = \sum_{j=0} \hat{A}_{ij}$ denotes its diagonal degree matrix, $W \in \mathbb{R}^{C \times F}$ is the weight matrix (where F is the number of neurons), and σ denotes a non-linearity function.

Gilmer et al. [19] have reformulated this equation as a the message passing layer, which makes it easier to compare to the other message passing functions described in this section. The MPNN formulation is as follows:

$$m_i^{k+1} = \sum_{j \in N(i) \cup i} \frac{x_j^k}{\sqrt{|N(j)||N(i)|}} \tag{5}$$

$$x_i^{k+1} = \sigma(W^k m_i^{k+1}), \tag{6}$$

where $|N(j)|$ and $|N(i)|$ denote the node degree of node j and i respectively, W^k denotes a layer-specific trainable weight matrix and σ is a non-linearity function. The node degrees of the neighbouring nodes is used as a per-neighbour normalisation and results in a weighted average aggregation scheme [19], [20].

2) GraphSAGE: GraphSAGE [21] is a spatial-based method introduced by Hamilton et al. They propose three different aggregation functions: a mean aggregator, a long short-term memory (LSTM) aggregator and a pooling aggregator. For our experiments, we use the mean operator, hence the node representations are updated according to

$$m_i^{k+1} = MEAN_{j \in N(i) \cup i}(x_j^k) \tag{7}$$

$$x_i^{k+1} = \sigma(W^k m_i^{k+1}). \tag{8}$$

In contrast to the aggregator of GCN, which assigns neighbour-specific, predefined weights based on the node degree, the mean operator of GraphSAGE assigns the same weights to all neighbours of a given node.

3) Graph Attention Network: The Graph Attention Network (GAT) [22] is a attention-based GNN. In contrast to GraphSAGE and GCN, which use predefined weights when aggregating the information from neighbouring nodes, GAT uses a neural network architecture to learn neighbour-specific weights. More particularly, an attention coefficient a_{ij}^k for two neighbouring nodes i and j is computed by passing the concatenated linear transformations of x_i^k and x_j^k through a single-layer perceptron with LeakyReLU activation. The linear transformations are obtained by using a shared weight matrix W^k. Normalising the attention coefficients across all choices of j with a softmax function finally results in the following formula for computing the attention coefficient:

$$a_{ij}^k = softmax_j(F(W^k x_i^k, W^k x_j^k)), \tag{9}$$

where $\cdot,\cdot$ denotes the concatenation and F is the single-layer perceptron with LeakyReLU activation. The attention coefficients are used to update the node representations according to

$$m_i^{k+1} = \sum_{j \in N(i)} a_{ij}^k W^k x_j^k \tag{10}$$

$$x_i^{k+1} = \sigma(a_{ii}^k W^k x_i^k + m_i^{k+1}), \tag{11}$$

where σ is a non-linearity.

4) Edge Network: The GNNs discussed so far are only able to learn from node features, but do not use the edge features. Edge Network (enn) [19] is a network that also takes into account the edge features. A layer-specific neural network F^k maps the edge feature vector e_{ij} to a matrix that can be multiplied with the node feature vector x_j^k. The message passing layer is defined as:

$$m_i^{k+1} = \sum_{j \in N(i)} x_j^k \cdot F^k(e_{ij}) \tag{12}$$

$$x_i^{k+1} = \sigma(W^k x_i^k + m_i^{k+1}), \tag{13}$$

where e_{ij} denotes the edge feature vector of the edge between node i and j. For F^k we use a Multilayer Perceptron (MLP) with two hidden layers.

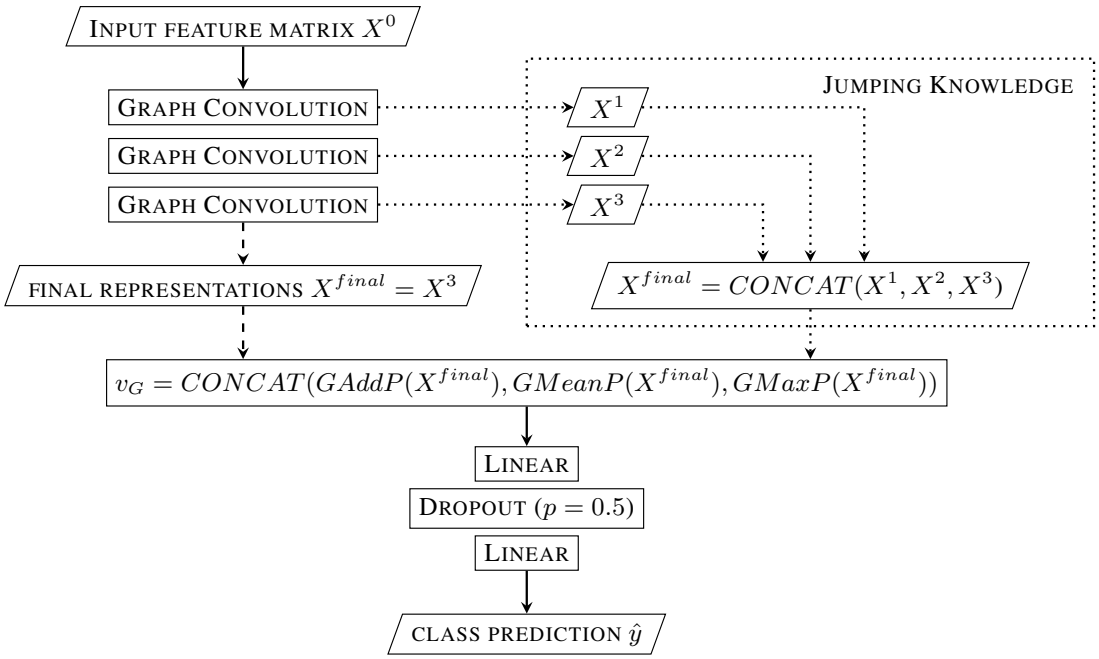
5) Graph Isomorphism Network: The Graph Isomorphism Network (GIN) [16] is a spatial-based GNN that aggregates neighbourhood information by summing the representations of neighbouring nodes. The representation of node i itself is then updated using an MLP:

$$m_i^{k+1} = \sum_{j \in N(i)} x_j^k \tag{14}$$

$$x_i^{k+1} = F((1+\epsilon) \cdot x_i^k + m_i^{k+1}), \tag{15}$$

where F is the MLP and ϵ is either a learnable parameter or fixed. We use an MLP with one hidden layer and ReLU

Fig. 2: Graph Neural Network Architecture. All models were evaluated using three message passing layers (graph convolution), two linear layers (linear) and a dropout layer (dropout). The dotted and dashed arrows correspond to the setup with and without Jumping Knowledge [18] respectively. The three graph convolution layers make up the message passing phase, whose final output is then used by the read-out phase to compute a vector representation of the whole graph v_G. The classification task is binary (normal or dysplastic).



activation to keep the model simple and set ϵ to zero since in [16] the variant with $\epsilon = 0$ slightly outperformed the alternative with a learnable ϵ. GIN's aggregation and read-out functions are injective, and is thus supposed to achieve maximum discriminative power [16].

6) 1-dimensional GNN: The 1-dimensional GNN (1-GNN) corresponds to one of the GNN baselines used in [23].

$$m_i^{k+1} = \sum_{j \in N(i)} W_1^k x_j^k \tag{16}$$

$$x_i^{k+1} = \sigma(W_2^k x_i^k + m_i^{k+1}). \tag{17}$$

The network learns neighbour-specific weights to aggregate local information by using two distinct weight matrices W_1^k and W_2^k. W_1^k is used during message generation to linearly transform the representations of the neighbours of node i. To update the node representation x_i^k the representation of node i itself is linearly transformed using W_2^k, added up with the message m_i^{k+1} and passed through a non-linearity σ.

IV. Experimental Evaluation

In this section, we present the experimental evaluation of the different GNN architectures for the task of intestinal gland classification on the pT1-GG benchmark dataset. They are compared with the graph edit distance baseline system.

A. Experimental Setup

All models described in section III-B are trained using three graph convolution layers with 64 neurons. The ReLU activation function is used as the non-linearity σ in all the models when updating the hidden node representations.

To get the graph-level output, the concatenation of global add-pooling ($GAddP$), global mean-pooling ($GMeanP$) and global max-pooling ($GMaxP$) is used as the readout function

$$v_G = (GAddP(X^K), GMeanP(X^K), GMaxP(X^K)), \tag{18}$$

where $\cdot, \cdot$ denotes the concatenation and X^K is the hidden node representation matrix after the K^{th} graph convolution layer (in our case $K = 3$). $GAddP$ sums up the feature vectors aross the node dimension, $GMeanP$ averages the node features across the node dimension and $GMaxP$ takes the feature-wise maximum across the node dimension of the graph. To classify an input graph, the obtained feature vector v_G is passed through a 2-layer MLP with ReLU activation and a dropout layer ($p = 0.5$) in between.

Additionally, GraphSAGE and GCN are also trained in combination with Jumping Knowledge (JK) [18]. JK allows for an adaptive neighbourhood range by aggregating representations across different layers. To get the final hidden representation of a node, the hidden representations of this node at the end of each message-passing layer is concatenated such that $CONCAT(x_i^1, ..., x_i^K)$, where $\cdot, \cdot$ denotes the concatenation. This allows the following linear layers to consider the read-out across all the graph convolution layers.

We use 4-fold CV to evaluate the performance of the

TABLE V: Mean test accuracy, precision and recall along with their standard deviation (in %) of the 4-fold cross-validation (CV) for 8 different GNN models using 4 and 33 node features (100 runs per fold). Models using Jumping Knowledge are labelled with "-JK". For comparison, we also establish a CNN baseline, we report the performance with and without rotation data augmentation (R). The GED-baseline is the current SOTA result using graph-based methods.

	ACCURACY		PRECISION		RECALL	
	# NODE FEATURES		# NODE FEATURES		# NODE FEATURES	
	4 (BASELINE)	33 (ALL)	4 (BASELINE)	33 (ALL)	4 (BASELINE)	33 (ALL)
1-GNN	**89.2** $\pm$ 3.8	94.6 $\pm$ 2.3	89.7 $\pm$ 5.4	94.4 $\pm$ 3.4	**89.2** $\pm$ 8.3	**95.1** $\pm$ 2.2
GAT	85.5 $\pm$ 5.4	94.3 $\pm$ 2.4	86.5 $\pm$ 9.2	94.7 $\pm$ 2.9	86.2 $\pm$ 13.8	93.8 $\pm$ 3.0
GCN	85.5 $\pm$ 4.9	94.5 $\pm$ 2.6	86.8 $\pm$ 8.3	94.4 $\pm$ 3.3	85.3 $\pm$ 13.6	94.7 $\pm$ 2.7
GCN-JK	85.4 $\pm$ 4.5	**94.8** $\pm$ 2.4	86.5 $\pm$ 8.1	94.7 $\pm$ 3.4	85.5 $\pm$ 13.2	94.8 $\pm$ 2.5
GIN	89.0 $\pm$ 4.1	94.5 $\pm$ 2.6	**90.0** $\pm$ 5.5	94.1 $\pm$ 3.4	88.4 $\pm$ 10.5	**95.1** $\pm$ 2.5
GRAPHSAGE	85.4 $\pm$ 4.5	**94.8** $\pm$ 2.4	86.5 $\pm$ 7.9	94.7 $\pm$ 3.4	84.7 $\pm$ 14.3	**95.1** $\pm$ 2.2
GRAPHSAGE-JK	85.1 $\pm$ 5.2	94.7 $\pm$ 2.4	86.4 $\pm$ 7.6	**95.1** $\pm$ 3.1	84.9 $\pm$ 13.9	94.7 $\pm$ 2.8
ENN	89.1 $\pm$ 3.7	93.7 $\pm$ 3.0	89.7 $\pm$ 5.6	93.2 $\pm$ 4.2	88.8 $\pm$ 8.1	94.5 $\pm$ 2.7
GED-BASELINE [5]	83.3 $\pm$ 1.7	N/A	N/A	N/A	N/A	N/A
CNN (VGG-16)	91.8$\pm$5.5		91.2$\pm$9.5		93.7$\pm$7.0	
CNN (VGG-16-R)	92.0$\pm$5.1		92.0$\pm$9.4		93.3$\pm$6.1	

models on the pT1-GG dataset. The published [5], and online[2] available dataset splits are used. The dataset is split into four equal parts (130 graphs each) on a patient and whole slide level (as we only have one slide per patient). Each subset contains the same number of samples for both classes. Two parts are combined to form the training set, the other two parts are used as a validation set and test set. Table II gives an overview of the graph sizes and number of patients in each split.

The hyper-parameters are optimised on the validation set of each fold using the Tree-structured Parzen Estimator (TPE) [24] from the open-source python library Hyperopt [25]. The TPE algorithm proposes 50 different combinations for the following hyper-parameters: learning rate, learning rate decay, step size for the learning rate decay, and L2-regularisation (weight decay). The step size determines the number of epochs after which the learning rate is decreased by the learning rate decay. The specific hyper-paramter values can be found in our GitHub repository[3].

Using the optimised hyper-parameters, each GNN model is trained 100 times until convergence (80 epochs) using the Adam optimiser [26], Cross-Entropy Loss and a batch-size of 64. We multi-run the experiments in order to take into account the stochastic nature of the experiments. The average accuracy, precision and recall on the test set of each fold is reported over the 100 runs, along with the standard deviation.

We explore two different node feature sets. Firstly, in order to compare our results with the current SOTA, we use the same node features used for the previously published results [5] (referred to as the GED-baseline). These features are the cytoplasm eosin stain minimum, and the nucleus hematoxylin stain mean, minimum and maximum. They were selected using forward search feature selection. Secondly, we also train each network using the full feature set, in order to see if this increases the performance.

All of the GNN experiments are run using the PyTorch Geometric library [27] and the implementation details as well as the hyper-parameters are available open-source[3].

We also establish a CNN baseline, since the pT1-GG dataset does not only contain the gland-graphs, but also the images from which the graphs were extracted. We train a VGG-16 network with batch normalisation [28] with an input image size of $128x128$ pixels and a batch size of 64, with and without rotation data augmentation (rotation by a random degree between 0 and 360 degree). For the hyper-parameter optimisation and model evaluation, the same setup is used as for the GNNs.

B. Results

Table V gives an overview of the results. Overall, we outperform the current SOTA by 11.5%. Using the same four node features as the GED-baseline, 1-GNN shows the best performance with 89.2%, outperforming the SOTA by 5.9%. Using all available node features further increases the accuracy, with GCN with JK and Graph-SAGE being the best-performing models with a classification accuracy of 94.8%. Taking the precision and recall into account, Graph-SAGE seems to perform slightly better. These two models even outperform the CNN baselines. It has also to be noted, that many of the models achieve a similarly high performance to the two best performing models, especially when using the full node feature set.

C. Discussion

We find that all the networks can make use of the full node feature set. Using 33 features leads to a better performance for all tested GNN architectures. This leads to believe, that using more information is beneficial. However, Edge Network, the network also using edge features, which thus has even more

[2]https://github.com/LindaSt/pT1-Gland-Graph-Dataset
[3]https://github.com/waljan/GNNpT1

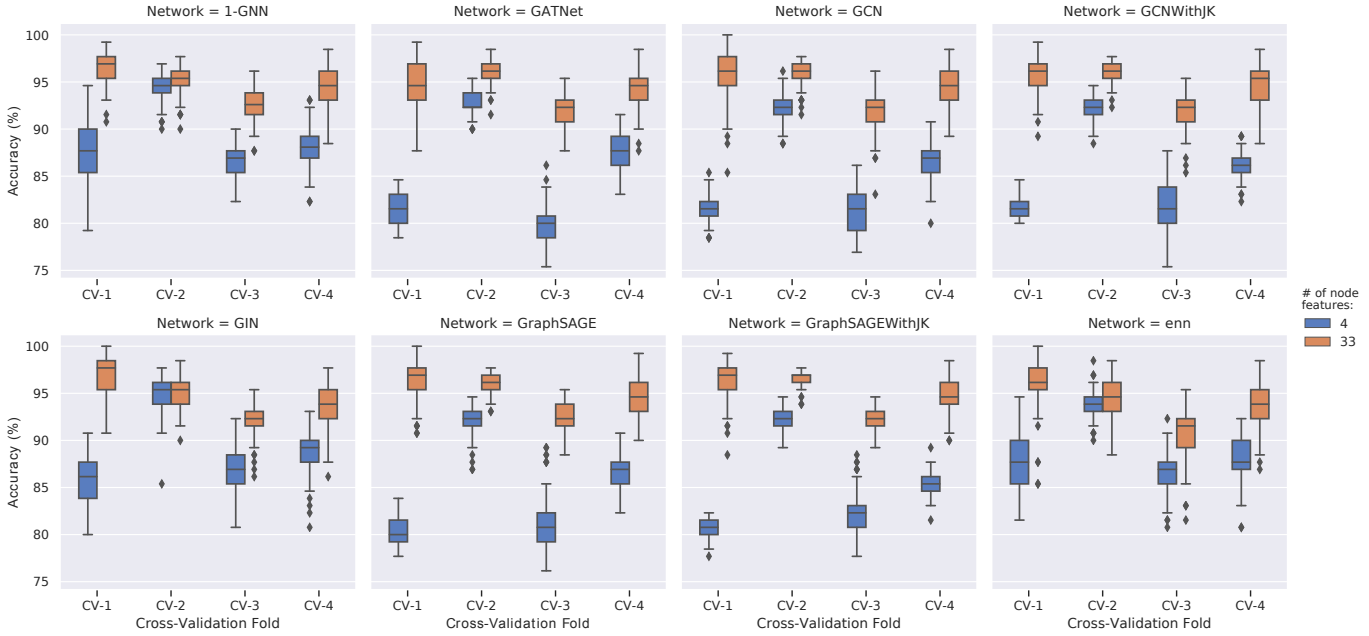
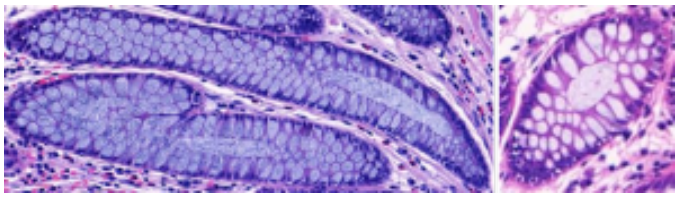
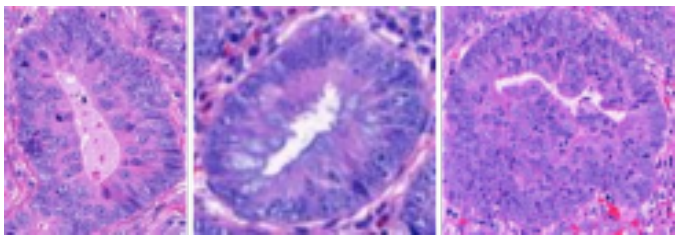


Fig. 3: Visualisation of the variance of the 100 runs within each cross-validation fold for all networks. The box-plots are grouped by the dataset, i.e. the number of node features that are used for the graph (33 is the full feature set, 4 is the subset used by the state-of-the-art method). Models using Jumping Knowledge are labelled with "-JK".

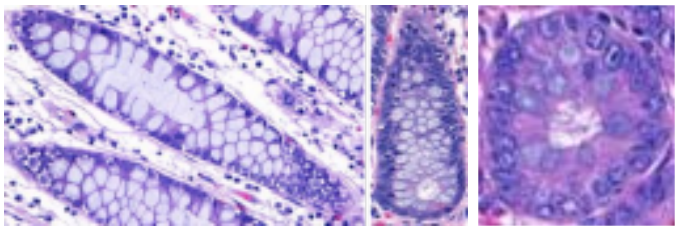


(a) Normal glands classified as dysplastic.



(b) Dysplastic glands classified as normal.

Fig. 4: Example images that are misclassified by the majority of the GNNs using the 4 baseline node features.



(a) Normal glands classified as dysplastic.

(b) Dysplastic glands classified as normal.

Fig. 5: Example images that are misclassified by the majority of the GNNs using the full 33 node features.

available information, does not perform better than the others. But it is noteworthy, that it performs almost equal to 1-GNN on the 4-feature dataset, which performed the best on that dataset. Figure 3 gives an overview of the different networks' performance for the individual cross-validation folds. Fold two shows the smallest difference in performance between the 4-features and 33-features dataset. When looking at the confusion matrices, we find that both normal and dysplastic glands are equally classified correctly, which implies that the GNNs are not biased towards one of the two classes.

The authors who introduced GIN [16] note that under-fitting the data can be a problem of less powerful GNNs. On the 4-feature baseline dataset, we find that this is indeed the case for all the architectures. All of them plateau at approximately 95% on the training data. Using the full available feature set, all models reach 100% training accuracy. We also find that adding Jumping Knowledge connections does not have a big impact on the performance of the network. This could be due to the fact that we use shallow models with only three graph convolution layers.

Figure 4 and 5 show examples of glands, which are misclassified in over 95% times in all of the runs of all the networks. Looking at the misclassified glands, they tend to show an uncommon shape and or staining intensity for the class they belong to. For example, sometimes normal glands are cut diagonally, which then results in an elongated shape. Another example are over-stained nuclei, as nuclei in dysplasic glands are hyperchromatic.

For the CNN-baseline, the VGG-16 model outperforms the GED-baseline and achieves an accuracy of 92.0%. However, it does not outperform the best-performing GNN. A possible

reason for this could be that the pT1 dataset created with a focus on graph creation and not CNN-based learning and the images were extracted at different magnification levels. Additionally, the rotational equivariance of the data, which is the case in digital pathology, has been shown to be challenging for CNNs [2], [3]. We can see that adding rotation data augmentation helps to counteract this effect, but it still performs around 3% worse than the best-performing GNNs.

V. Conclusion

In this paper, we show that using Graph Neural Networks instead of classical graph-based methods increases the classification accuracy on the pT1 Gland Graph dataset. We achieve a performance of 89.2% using the same node feature set as the current SOTA, which is an improvement of 5.9%. Furthermore, we show that GNNs benefit from using the whole node feature set, as using all the 33 features leads to the best performance. Graph Convolutional Network with JKs and GraphSAGE both achieve an accuracy of 94.8%. For future work, it will also be interesting to explore different graph representations. Currently each node is only connected to its two spatially closest neighbours, which leads to very limited information exchange during the message passing phase. The pathologists' feedback can also be helpful in order to improve the graph representation in terms of how to best mirror the biological relationship between the cells. Additionally, evaluating the expert's performance, especially the inter-observer variability, on this task will also be an interesting baseline. Furthermore, adding more edge and node features could also help improve the performance. Instead of hand-crafted node features, a CNN, e.g. an auto-encoder [29], can be used to learn the feature vectors. Using an ensemble of different classifiers can also help further improve the performance.

Acknowledgment

The work presented here has been partially supported by the Rising Tide foundation with the grant number CCR-18-130.

References

[1] I. D. Nagtegaal, R. D. Odze, D. Klimstra, V. Paradis, M. Rugge, P. Schirmacher, K. M. Washington, F. Carneiro, I. A. Cree, and W. C. of Tumours Editorial Board, "The 2019 who classification of tumours of the digestive system," *Histopathology*, vol. 76, no. 2, pp. 182–188, 2020.

[2] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling, "Rotation equivariant cnns for digital pathology," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2018, pp. 210–218.

[3] M. Weiler, F. A. Hamprecht, and M. Storath, "Learning steerable filters for rotation equivariant cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 849–858.

[4] H. Sharma, N. Zerbe, S. Lohmann, K. Kayser, O. Hellwich, and P. Hufnagl, "A review of graph-based methods for image analysis in digital histopathology," *Diagnostic pathology*, 2015.

[5] L. Studer, S. Toneyan, I. Zlobec, H. Dawson, and A. Fischer, "Graph-based classification of intestinal glands in colorectal cancer tissue images," in *2nd COMPAY Workshop at the 22nd MICCAI Conference*, 2019.

[6] C. Gunduz, B. Yener, and S. H. Gultekin, "The cell graphs of cancer," *Bioinformatics*, vol. 20, no. suppl_1, pp. i145–i151, 2004.

[7] D. Anand, S. Gadiya, and A. Sethi, "Histographs: graphs in histopathology," in *Medical Imaging 2020: Digital Pathology*, vol. 11320. International Society for Optics and Photonics, 2020, p. 113200O.

[8] C. Bilgin, C. Demir, C. Nagi, and B. Yener, "Cell-graph mining for breast tissue modeling and classification," in *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2007, pp. 5311–5314.

[9] A. Fischer, K. Riesen, and H. Bunke, "Improved quadratic time approximation of graph edit distance by combining hausdorff matching and greedy assignment," *Pattern Recognition Letters*, vol. 87, pp. 55–62, 2017.

[10] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *International journal of pattern recognition and artificial intelligence*, vol. 18, no. 03, pp. 265–298, 2004.

[11] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[12] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[14] B. Aygüneş, S. Aksoy, R. G. Cinbiş, K. Kösemehmetoğlu, S. Önder, and A. Üner, "Graph convolutional networks for region of interest classification in breast histopathology," in *Medical Imaging 2020: Digital Pathology*, vol. 11320. International Society for Optics and Photonics, 2020, p. 113200K.

[15] R. Li, J. Yao, X. Zhu, Y. Li, and J. Huang, "Graph cnn for survival analysis on whole slide pathological images," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 174–182.

[16] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[17] M. Loda, L. A. Mucci, M. L. Mittelstadt, M. Van Hemelrijck, and M. B. Cotter, *Pathology and epidemiology of cancer*. Springer, 2016.

[18] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," *arXiv preprint arXiv:1806.03536*, 2018.

[19] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1263–1272.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[21] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.

[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[23] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4602–4609.

[24] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[25] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in science conference*. Citeseer, 2013, pp. 13–20.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[29] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.