Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Graph Neural Networks for Identification of Robust Biomarkers from Multi-Omics Data

Jan Lubojacký

Supervisor: Doc. Jiří Kléma
Field of study: Medical Electronics and Bioinformatics
Subfield: Image processing
January 2025

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Lubojacký Jan**	Personal ID number:	**492102**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Medical Electronics and Bioinformatics**		
Specialisation:	**Image processing**		

II. Master's thesis details

Master's thesis title in English:

Graph Neural Networks for Identification of Robust Biomarkers from Multi-Omics Data

Master's thesis title in Czech:

Grafové neuronové sítě pro identifikaci robustních biomarkerů z multi-omických dat

Guidelines:

Recent sequencing progress has led to the generation of extensive omics datasets encompassing various molecular layers, including RNA expression (mRNA, miRNA, circRNA, piRNA), DNA methylation, and copy number variation. However, the development of algorithms capable of effectively integrating omics data for addressing tasks such as cancer subtype classification, survival analysis, or detection of important biomarkers, remains an active and pivotal area of research.
Leveraging graph neural networks (GNNs) to utilize the existing omics interaction knowledge as well as to learn these interactions from data can yield significantly improved performance over classical classification algorithms as well as over other deep learning architectures.
1. Get acquainted with the existing applications of graph (convolutional) neural networks (G(C)NNs) to multi-omics data.
2. Learn about G(C)NNs themselves.
3. Carry out a literature search for methods ad 1 and 2.
4. Propose your own G(C)NN architecture suitable for myelodysplastic syndromes (MDS) data provided by your supervisor. Integrate multiple expression data, learn interactions among features and samples from these data, utilize the existing knowledge about RNA interactions too.
5. Find robust graph markers for class prediction in the above mentioned data.
a. Evaluate their performance (e.g., in terms of AUROC) and compare with their competitors.
b. Validate the found biomarkers in the literature, or discuss them with a domain expert.

Bibliography / sources:

[1] Wang, Tongxin, et al. "MOGONET integrates multi-omics data using graph convolutional networks allowing patient classification and biomarker identification." Nature Communications 12.1 (2021): 1-13.
[2] Zhao, Lianhe, et al. "DeepOmix: A scalable and interpretable multi-omics deep learning framework and application in cancer survival analysis." Computational and structural biotechnology journal 19 (2021): 2719-2725.
[3] Ramirez, Ricardo, et al. "Classification of cancer types using graph convolutional neural networks." Frontiers in physics 8 (2020): 203.
[4] Xiao, Shunxin, et al. "Graph Neural Networks with Multiple Prior Knowledge for Multi-Omics Data Analysis." IEEE Journal of Biomedical and Health Informatics (2023).

Name and workplace of master’s thesis supervisor:

doc. Ing. Jiří Kléma, Ph.D. Intelligent Data Analysis FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **06.02.2024** Deadline for master's thesis submission: **07.01.2025**

Assignment valid until: **21.09.2025**

doc. Ing. Jiří Kléma, Ph.D.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt

Student’s signature

Acknowledgements

I would like to thank my supervisor, doc. Ing. Jiří Kléma, Ph.D., for his time and valuable support and guidance. Furthermore I would like to thank to my family and to S. for their support.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ....

....
signature

Abstract

Modern sequencing methods generate vast amounts of biological data, enabling the acquisition of complex molecular profiles for various cancers and diseases. This thesis explores the applicability of Graph Neural Networks (GNNs) for tasks such as cancer subtype classification from multi-omic data. We systematically described, evaluated, and categorized existing methods available in the literature, proposing and evaluating improvements and extensions. Building upon an existing method, we introduced a novel architecture capable of leveraging a priori biological knowledge about the interactions between different genes and non-coding RNAs. Additionally, we focused on explaining and utilizing these models to identify and extract robust biological markers that provided insights into the underlying biological mechanisms.

Keywords: graph neural networks, multi-omic data, biomarkers

Supervisor: Doc. Jiří Kléma
Karlovo náměstí 13,
121 35 Prague

Abstrakt

Moderní sekvenační metody generují obrovské množství biologických dat, což umožňuje získávání komplexních molekulárních profilů pro různé druhy rakoviny a onemocnění. Tato práce zkoumá aplikovatelnost grafových neuronových sítí (GNN) pro úlohy, jako je klasifikace podtypů rakoviny z multi-omických dat. Systematicky jsme popsali, vyhodnotili a kategorizovali existující metody dostupné v literatuře. Navrhli a vyhodnotili jsme zlepšení a rozšíření. Na základě existující metody jsme představili novou architekturu schopnou využít apriorní biologické znalosti o interakcích mezi různými geny a nekódujícími RNA. Dále jsme se zaměřili na vysvětlitelnost těchto modelů a jejich využití pro identifikaci a extrakci robustních biologických markerů, které poskytují přehled o biologických mechanismech.

Klíčová slova: grafové neuronové sítě, multi-omická data, biomarkery

Překlad názvu: Grafové neuronové sítě pro identifikaci robustních biomarkerů z multi-omických dat

Contents

Figures

Tables

Chapter 1

Introduction

Recent advancements in sequencing technologies have led to the generation of extensive omics datasets encompassing various molecular layers, including RNA expression (mRNA, miRNA, piRNA), DNA methylation, and copy number variation. Despite these advancements, the development of algorithms capable of effectively integrating multiple omics data types for addressing biomedical challenges, such as cancer subtype classification or survival analysis and the detection of important biomarkers, remains an active and pivotal area of research.

In this thesis, we explore the application of Graph Neural Networks (GNNs) to this task. GNNs are suitable for this task because many biological data are often characterized by interaction networks such as gene-gene interactions, protein-protein, miRNA-gene regulations, and similar. Additionally, we can construct sample-sample similarity graphs. GNNs can explore this data in its natural graph form to make meaningful predictions. Recent research suggests that GNNs-based approaches beat classical ML algorithms in this domain [2, 3, 4, 5, 6].

All of this research is very recent, so there are not many comparative studies. This thesis aims to survey the recent methods, characterize and compare the different approaches that these methods use, and finally, it aims to propose a new GNN architecture for classification tasks from Multi-omic data and biomarker discovery. Additionally, we explore the efficacy of the surveyed models in identifying biomarkers within a myelodysplastic syndromes dataset.

In the biological domain, it is also essential that a successful model is also explainable so that we can understand the underlying biological mechanisms. Identifying which biological mechanisms play a role in a disease is an important step in drug discovery because then we can design drugs that target these. That is why, in this thesis, we focus first on a successful model and second on how to extract feature importance from such a model. Consequently, we can report the important raw biomarkers as well as the structured biomarkers that take into account feature interactions. They both help us to differentiate between different cancer types or healthy and cancerous tissues.

Notes on notation

An unified notation is kept throughout the work. For clarity we provide an overview of the notation here.

- Scalars are denoted with lowercase letters in regular font e.g. $a \in \mathbb{R}$.
- Vectors are denoted with lowercase letters in bold e.g. $\mathbf{x} \in \mathbb{R}^n$.
- Matrices & tensors are denoted with uppercase letters in bold e.g. $\mathbf{X} \in \mathbb{R}^{n \times n}$.
- $\sigma(\cdot)$ denotes a non-specific non-linearity e.g. $\mathrm{ReLU}(\cdot)$, $\mathrm{ELU}(\cdot)$, etc..
- For a graph $G = (V, E)$ node feature of vertex $u \in V$ is denoted as $\mathbf{h}_u$.
- Neighborhood (set of adjacent vertices) of a node u in graph G is denoted as $\mathcal{N}(u)$.
- For heterogeneous graphs we denote the set of all relations with $\mathcal{R}$.
- For a matrix of weights $\mathbf{W}^{(l,r,k)}$ or an embedding $\mathbf{h}_u^{(l)}$ superscripts are used to denote the layer (l), where $l \in \{1, \ldots, L\}$, relation (r), where $r \in \mathcal{R}$ and attention head (k), where $k \in \{1, \ldots, K\}$.
- $[\mathbf{x} \| \mathbf{y}]$ denotes the concatenation of vectors $\mathbf{x}$ and $\mathbf{y}$.
- $\mathbf{I}$ is identity matrix.

Code availability

The code for this thesis is available either as attachment with thesis or at github https://github.com/JanLubojacky/DiplomaThesis.

Chapter 2

Biological background

In this chapter we aim to give an overview of the biological background for this thesis. The chapter contains a brief description of the different molecular data used in this thesis and the methods used to obtain it.

2.1 Omic data

Omic data refers to molecular data generated from studies with the "-omics" suffix. The main ones are genomics, transcriptomics, proteomics, metabolomics, and epigenomics. Each of these provides a different but related view, and together, these aim to describe the complex processes in an organism. Omics data is often used in systems biology and functional genomics to study the relationships between different molecules, tissues, and organisms. It can be complex, high-dimensional, and noisy.

As the volume of omic data has increased, researchers have become interested in integrating these different types of data to inform hypotheses and biological questions. Multi-omic data integration is a challenging task due to the heterogeneity of the data, which can have different numbers of variables and mismatched scaling and be generated using different formats and platforms.

Integrating these diverse data types into a single prediction model is challenging, and various methods ranging from simpler ones like KNNs and tree-based methods like random forests to more complex ones like neural networks and, more recently, graph neural networks are used to tackle this task.

2.2 Messenger RNA

Messenger RNA (mRNA) is a fundamental component of gene expression. Inside the nucleus, DNA is transcribed into mRNA, which is then transported outside the nucleus. On ribosomes, mRNA is translated into proteins, which are then involved in most cellular processes. The process of mRNA transcription and translation is central to the cell's ability to carry out its functions, from growth and development to response to environmental stimuli and disease.

In recent years, advances in RNA sequencing and bioinformatics have allowed for identifying and characterizing thousands of mRNA molecules in various organisms, providing valuable insights into gene expression and cellular processes.

Di erential expression of various mRNAs is known to be a good discriminator between cancerous and non-cancerous tissue, as well as between di erent cancer types [7, 8]. This di erential expression indicates how speci c gene expression patterns vary under di erent pathological states.

2.3 DNA methylation

DNA methylation is a crucial epigenetic modi cation that plays a vital role in regulating gene expression and maintaining genomic stability. It involves the transfer of a methyl group (CH_3) to the C_5 position of cytosine bases in DNA, producing 5-methylcytosine (5mC). Methylation regulates DNA transcription by recruiting proteins involved in gene repression or by inhibiting the binding of transcription factor(s) to DNA [9].

This process is primarily carried out by DNA methyltransferase enzymes (DNMTs), which can function as de novo DNMTs (establishing the initial pattern of methyl groups on a DNA molecule) or maintenance DNMTs (copying the methylation pattern from an existing DNA strand to its new partner after replication) [9].

DNA methylation is essential for various cellular processes, including gene regulation, cell di erentiation, and the silencing of repetitive chromosomic regions. DNA methylation has been linked with tumor onset and progression [8].

2.4 Copy number alterations/variation

Copy number variation (CNV) is a signi cant aspect of genetic variation that has gained increasing attention in recent years. CNVs are unbalanced structural variants characterized by the presence of extra or missing copies of DNA segments, which can have a profound impact on gene expression and phenotype. The length of these segments can vary greatly, ranging from a few base pairs to several megabases. These variants play a crucial role in evolution, population diversity, and the development of various diseases. Besides their biological functions, they have been associated with various pathological states such as cardiovascular, autoimmune, neurodegenerative disorders, and cancer [10]. CNVs are detected through a combination of cytogenetic, hybridization, PCR-based, and molecular hybridization techniques [11].

Copy number alterations (CNAs) are conceptually the same thing as variations. However, they usually refer to the variance of copy numbers in somatic cells (i.e., the changes in copy numbers in a tumor vs in a healthy tissue). In contrast, CNVs refer to the variance in germline cells (i.e., between di erent organisms). Though slightly di erent nomenclature might be preferred in di erent sub elds [12].

Copy number variations/alterations may span small or large sections of the genome [10], but for the tasks de ned in this work we are mostly interested in the alterations at the level of individual genes.

2.5 Micro RNAs

MicroRNAs (miRNAs) are a class of small non-coding RNA molecules, approximately 22 nucleotides long, that play crucial roles in the regulation of gene expression. They are known to bind to the 3^0 untranslated regions (3^0 UTRs) of target mRNAs, leading to their degradation or translational repression, thereby modulating gene function at the post-transcriptional level. The discovery and characterization of miRNAs have revolutionized our understanding of RNA biology, highlighting the complexity and diversity of RNA-mediated regulation in biological systems [13, 14].

MiRNAs have been identi ed as signi cant biomarkers for several types of cancer [15, 14]. Furthermore, the study of miRNA targeting has advanced signi cantly, with methods such as CLIP-seq [16] and bioinformatics tools like miRBase [17, 18, 19, 20, 21, 22] playing pivotal roles in identifying and validating miRNA target sites. These advancements have not only deepened our understanding of miRNA function but also opened new avenues for therapeutic interventions targeting miRNA pathways. The ongoing research in this area promises to uncover the full potential of miRNAs as regulators of complex biological processes, thereby expanding our toolkit for disease diagnosis and treatment.

2.6 PIWI-interacting RNAs (piRNAs)

PIWI-interacting RNAs (piRNAs) are small RNAs (21-35 nucleotides long) that are responsible for transposon silencing, regulating gene expression, and ghting viral infections. PiRNAs are involved in various biological processes, including transposon mobilization inhibition, transcript degradation, and chromatin formation regulation. They form complexes with PIWI proteins, members of the PIWI branch of the Argonaute protein, which are crucial for maintaining genomic integrity in germline cells. This has led to a focus on piRNAs in cancer research, as their abnormal expression has been observed in various types of cancers, suggesting their potential as biomarkers and therapeutic targets. Studies have shown that piRNAs play a role in the development and maintenance of germ lines by silencing transposons and protecting genome integrity. In somatic tissues, piRNAs are also involved in gene regulation and have been implicated in processes such as cell proliferation, invasion, metastasis, and apoptosis. Their deregulation in cancer cells can lead to imbalances in cell machinery, contributing to tumorigenesis. Speci c piRNAs have been identi ed as potential therapeutic targets in gastric cancer, and their expression levels in circulating tumor cells have been found to be more stable and easily detectable compared to other biomarkers, indicating their potential as sensitive diagnostic and prognostic markers [23, 24].

2.7 Circular RNAs (circRNAs)

Circular RNAs (circRNAs) are a class of endogenous RNAs that form a covalently closed ring-like molecular structure and are extensively found in mammalian cells. They are conserved across species and often show tissue or developmental stage-speci c expression. While the majority of them are non-coding circRNAs, they have been found to possess regulatory functions as microRNA (miRNA) sponges, regulating splicing and transcription. Recent research has highlighted their association with human diseases, particularly cancers, suggesting they could serve as biomarkers.

CircRNAs have been implicated in various types of cancer, including bladder carcinoma, where overexpression of a speci c circRNA was linked to the downregulation of miRNAs and the promotion of cell proliferation and migration. This has led to the exploration of circRNAs as potential diagnostic and prognostic markers, as well as therapeutic targets[25]. They have also been implicated in Acute myeloid leukemia and myelodysplastic neoplasms [26, 27].

2.8 Interaction data

Biological systems are complex. Often, certain traits (phenotypes) arise not under the in uence of a single factor but rather are determined by interactions between di erent biomolecules. One of the hypotheses of this work is that by including prior biological knowledge about these interactions in our models, we can improve their performance.

Gene-gene interactions

Gene-gene interactions (epistasis) describe the way that di erent combinations of genes lead to di erent phenotypes. Quite early on, it has been discovered that the classical Mendelian inheritance model, where one gene is responsible for one trait, is not the complete picture and that, in reality, many traits depend on the interactions between multiple genes. Interactions between genes are discovered with classical statistical models such as linear regression (quantitative traits) or logistic regression (discrete traits). More recently, deep learning models have also been used for this task [28, 29].

Protein-protein interactions

Proteins control nearly all cellular processes. This includes regulation of gene expression, catalyzation of chemical reactions, transport of small molecules across membranes, and the transmission of signals across membranes. Proteins interact with each other by physically binding with each other, forming larger structures composed of multiple proteins. Protein-protein interaction (PPI) networks are used to study the molecular basis of certain diseases and to nd potential therapeutic targets. PPIs are detected with experimental methods such as yeast two-hybrid (Y2H) and tandem a nity puri cation-mass spectrometry (MS) or with computational methods like In silico two-hybrid (I2H) and Ortholog-based approach [30].

Other interactions

As already mentioned in Section 2.5, miRNAs are known to regulate the expression of certain genes by binding their mRNA transcripts. These pairs form miRNA-mRNA interactions. Computational methods, including machine learning, as well as experimental methods like High-Throughput sequencing technologies, can be used to detect them [31].

CircRNAs can bind to miRNAs, acting as sponges for those miRNAs, thus promoting the activity of its target genes. CircRNA-miRNA interactions have been implicated to play a role in di erent diseases such as lung adenocarcinoma or Parkinson's disease [32].

Chapter 3

Graph Neural Networks

This chapter aims to provide an overview of Graph Neural Networks so that the reader can understand the methods discussed later. Methods and specific graph neural layers that are used and compared in this work are explained in more detail.

First let us lay out some basic notation. We have a graph $G = (V, E)$ with vertices (nodes) $v \in V$ and edges $\{v_1, v_2\} \in E$. Each node v has a node feature vector $h_v \in \mathbb{R}^d$. Nodes that share an edge with v are said to be the neighborhood of v and are denoted as $\mathcal{N}(v)$.

3.1 Neural message passing: Aggregation & Transformation

Graph neural networks have different flavors, such as Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), or Relational Graph Convolutional Networks (RGCNs), but they all utilize the same two steps.

In the first step, the information gets passed around the network, and each node $u \in V$ aggregates the information from its neighbors $v \in \mathcal{N}(u)$. After that some non-linear transformation is used to transform the node feature h_u together with the aggregated information from its neighbourhood $\{h_v, \forall v \in \mathcal{N}(u)\}$, these two steps can be described as

$$h_u^{(l)} = \mathrm{UPDATE}\left(h_u^{(l-1)}, \mathrm{AGGREGATE}^{l-1}\left(\{h_v^{(l-1)}, \forall v \in \mathcal{N}(u)\}\right)\right), \quad (3.1)$$

where $l \in \{1, \ldots, L\}$ denotes the layer.

Then, the different GNN flavors differ in how these UPDATE and AGGREGATE functions are defined. We want UPDATE to be a non-linear transformation, like a linear layer followed by a non-linearity, and AGGREGATE to be a permutation invariant aggregator like mean or sum. For a basic GNN, we define the update and aggregate functions like this

$$h_u^{(l)} = \sigma\left(W_{self}^{(l)} h_u^{(l-1)} + W_{neigh}^{(l)} \sum_{v \in N(u)} h_v^{(l-1)} + b^l\right). \tag{3.2}$$

Where W are weight matrices used to transform the embedding, we usually use a different matrix for transforming the self-embedding and the aggregated neighbor embedding. However, sharing a single matrix for both is also possible. Bias is shared in both scenarios [33, 34].

3.2 Graph Convolutional Networks

The basis of Graph Convolutional Networks (GCNs) [35, 33] is the message-passing algorithm. This algorithm provides an invariant permutation method to propagate information in the network, which can be seen as a generalization of convolutions for non-Euclidian data. After aggregation, the features are transformed with a non-linear transformation.

We describe this using matrix notation, which allows us to easily describe message passing with matrix multiplication. If we have a matrix $H^{(l-1)}$ with node features as columns message passing can be performed simply by multiplying the current node features $H^{(l-1)}$ with a modified adjacency matrix $\hat{A}$ as

$$H^{(l)} = \hat{A} H^{(l-1)}. \tag{3.3}$$

The modifications of the adjacency matrix involve adding self loops

$$\tilde{A} = A + I, \tag{3.4}$$

which allows us to share a weight matrix for the node feature transformations. Additionally, we normalize the matrix as

$$\hat{A} = D^{-\frac{1}{2}} \tilde{A} D^{-\frac{1}{2}}, \tag{3.5}$$

where D is the degree matrix with the degree of each node on the diagonal. This normalization corresponds to adjusting each edge by the square root of the degree sizes of the two nodes it connects, which plays a crucial role in mitigating issues such as vanishing or exploding gradients, contributing to the stability and effectiveness of the node aggregation process.

With this step, each node in the network can aggregate information from its neighbors. After the aggregation, we apply a non-linear transformation to the updated feature vector in the form of a linear layer with a weight matrix W, bias b and a non-linearity $\sigma(\cdot)$:

$$H^{(l)} = \sigma(\hat{A}H^{(l-1)}W^{(l)} + b^{(l)}) \quad \forall l \in \{1, \ldots, L\}. \tag{3.6}$$

Oversmoothing

With each layer, the receptive field of a node increases by one hop. For L layers, each node aggregates information from its L-hop neighborhood. It is important to note that having the network deeper than the length of the longest shortest path in the network is not beneficial, as it leads to oversmoothing. As we continue to aggregate messages in the network, the messages will start to average out, increasing the entropy of the system. Eventually, the information in each node will be identical. For this reason, in practice, most Graph Neural Networks are only a couple of layers deep [36].

3.3 Graph Attention Networks (GATs)

In graph attention networks (GATs) [1], we modify the aggregation step. In GCNs, we use static edge weights given by the product of reciprocal degrees of the two nodes u, v, and then aggregate the messages from neighbors with a permutation-invariant aggregator such as mean, sum, or max [34].

Figure 3.1: Schema for GAT with 3 attention heads (purple, blue, green) taken from [1]

Here, instead of the static weights, we use learned attention weights $\alpha(v,u)$ to define the importance of each neighbor $v \in \mathcal{N}(u)$ relative to the node u (note that the graph involves self-loops so $u \in \mathcal{N}(u)$), and then aggregate the messages via a weighted sum.

The node update equation 3.1 for Graph Attention Neural Networks is then expressed as

$$h_u^{(l)} = \sigma\left(\sum_{v \in \mathcal{N}(u)} \alpha(v,u) W^{(l)} h_v^{(l-1)}\right), \tag{3.7}$$

where $l \in \{1, \dots, L\}$ is the layer, $W^{(l)}$ is a weight matrix and $\alpha(v,u)$ is softmax of the attention scores $a(h_v^{(l-1)}, h_u^{(l-1)})$

$$\alpha(v,u) = \mathrm{softmax}(a(h_v^{(l-1)}, h_u^{(l-1)})). \tag{3.8}$$

This framework allows the use of different attention mechanisms to compute the attention scores $a(h_v^{(l-1)}, h_u^{(l-1)})$. In the original GAT paper [1], they used a single linear layer with the LeakyReLU activation function, which gives us the attention scores as

$$a(h_v, h_u) = \mathrm{LeakyReLU}(a^T [W^{(l)} h_v^{(l-1)} \| W^{(l)} h_u^{(l-1)}]), \tag{3.9}$$

where the $[\cdot \| \cdot]$ operator denotes the concatenation of the two transformed vectors and $W^{(l)}$ is reused from (3.7). The final attention weights equation is expressed as

$$\alpha(v,u) = \frac{\exp\left(\mathrm{LeakyReLU}(a^T[W^{(l)}h_u^{(l-1)} \| W^{(l)}h_v^{(l-1)}])\right)}{\sum_{k \in N(u)} \exp\left(\mathrm{LeakyReLU}(a^T[W^{(l)}h_u \| W^{(l)}h_k^{(l-1)}])\right)}. \tag{3.10}$$

We can also have multiple attention heads, meaning that in each layer l, we have a set of weights $a^{(l,k)}, W^{(l,k)}$ for $k \in \{1, \ldots, n\}$ for n attention heads. We can concatenate the output from each head into a new vector

$$h_u^{(l)} = \Big\|_{k=1}^{K} \left(\sum_{v \in N(u)} \alpha(u,v)^{(l,k)} W^{(l,k)} h_v^{(l-1)} \right). \tag{3.11}$$

Or as an average it if this is the last layer and we want to obtain a single prediction

$$h_u^{(l)} = \left(\frac{1}{K} \sum_{k=1}^{K} \sum_{v \in N(u)} \alpha(u,v)^{(l,k)} W^{(l,k)} h_v^{(l-1)} \right). \tag{3.12}$$

Later, it was shown that the attention mechanism as defined in equations (3.9) and (3.10) computes a weaker, less expressive version of attention called static attention. This boils down to how the attention scores are computed, in (3.9) a and W are applied directly after each other, thus collapsing into a single linear layer. This can be fixed by changing the order of operations and pulling the attention scores outside the non-linearity:

$$a(h_v, h_u) = a^T \mathrm{LeakyReLU}(W[h_v \| h_u]) \tag{3.13}$$

This allows the layer to compute dynamic attention, which increases its expressiveness. This improved version is referred to as GATv2 [37].

Note that in these equations, as taken from the original papers, a shared weight matrix W is used to transform all the messages. However, it is simple to modify this and use two different matrices W_{self} and $W_{neighbors}$ instead. This is the default in the torch geometric library [38].

3.4 Relational Graph Convolutional Networks (RGCNs)

Relation is a triplet of $(V_x; E_{(x \to y)}; V_y)$, where x and y denote a different type of the node. In our context, this could be an interaction between two different features, for example, a miRNA-mRNA interaction.

In RGCNs [39, 33], we have a different weight matrix for each of these triplets. The update equation looks like this:

$$h_u^{(l)} = \left(W_s^{(l)} h_u^{(l-1)} + \sum_{r \in R} \sum_{j \in N_r(u)} \frac{1}{c_{ur}} W_r^{(l)} h_v^{(l-1)} \right) : \tag{3.14}$$

Where R is the set of all relations on graph G and c_{ur} is a normalization constant that can be learned or chosen in advance (it is set to $c_{u;r} = |N_r(u)|$ in [39]).

If the number of relations is large, it is beneficial to introduce some form of regularization to prevent overfitting.

Block diagonal matrix

We can constraint the matrices W_r to be block diagonal to reduce the number of parameters and allow only positions close in the embedding to interact. This ensures that W_r remains sparse [39].

Basis learning

Or we can put a cap on how many base weight matrices V_b we want per layer and compose the necessary matrices W_r out of them via a linear combination with learned coefficients. This is useful in case we have a heterogeneous graph with many relations [39]:

$$W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} V_b^{(l)} : \tag{3.15}$$

3.5 Relational Graph Attention Networks (RGATs)

In relational graph attention networks, attention coefficients $a_{u,v}^{(r)}$ for a relation $r \in \mathcal{R}$ between nodes u and v are obtained in the following manner. First, the query and key vectors are computed using the query $Q^{(r)} \in \mathbb{R}^{F' \times D}$ and key kernels $K^{(r)} \in \mathbb{R}^{F' \times D}$ as:

$$q_u^{(r)} = W_1^{(r)} h_u \cdot Q^{(r)} \in \mathbb{R}^D; \tag{3.16}$$

and

$$k_u^{(r)} = W_1^{(r)} h_u \cdot K^{(r)} \in \mathbb{R}^D. \tag{3.17}$$

Two schemes are proposed for computing the attention logits. Additive where $D = 1$ and the logit is computed as:

$$a_{u,v}^{(r)} = \text{LeakyReLu}\left(q_u^{(r)} + k_v^{(r)}\right); \tag{3.18}$$

or multiplicative where D can be any positive integer and:

$$a_{u,v}^{(r)} = q_u^{(r)} \cdot k_v^{(r)}. \tag{3.19}$$

In case the graph contains edge features $e_{u,v}^{(r)}$ this expands to

$$a_{u,v}^{(r)} = \text{LeakyReLu}\left(q_u^{(r)} + k_v^{(r)} + W_2^{(r)} e_{u,v}^{(r)}\right); \tag{3.20}$$

and

$$a_{u,v}^{(r)} = q_u^{(r)} \cdot k_v^{(r)} \cdot W_2^{(r)} e_{u,v}^{(r)}. \tag{3.21}$$

Next, attention logits are normalized to be comparable for different nodes. Two normalization schemas are proposed, within-relation

$$\alpha_{u,v}^{(r)} = \frac{\exp\left(a_{u,v}^{(r)}\right)}{\sum_{k \in \mathcal{N}_r(u)} \exp\left(a_{u,k}^{(r)}\right)}; \tag{3.22}$$

which normalizes attention logits in a neighborhood with respect to relations and consequently encodes the prior that relation importance is a global property, and across-relation

$$\alpha^{(r)}_{u,v} = \frac{\exp\left(a^{(r)}_{u,v}\right)}{\sum_{r' \in R} \sum_{k \in N_{r'}(u)} \exp\left(a^{(r')}_{u,k}\right)}, \tag{3.23}$$

which normalizes attention logits in a neighborhood irrespective of relations and thus encodes the prior that relation importance is a local property.

Finally, we can express the node update for the current layer $l \in \{1, \ldots, L\}$ as

$$h^{(l)}_u = \sigma\left(\sum_{r \in R} \sum_{v \in N_r(u)} \alpha^{(l,r)}_{u,v} W^{(l,r,k)} h^{(l-1)}_i\right) \in \mathbb{R}^{N \times F'}, \tag{3.24}$$

or for multi-headed attention as

$$h^{(l)}_u = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{r \in R} \sum_{v \in N_r(u)} \alpha^{(l,r,k)}_{u,v} W^{(l,r,k)} h^{(l-1)}_u\right) \in \mathbb{R}^{N \times KF'}, \tag{3.25}$$

where $\Vert$ denotes concatenation [40].

3.5.1 Pooling on Graphs

Pooling operations have proven to be very useful in classical convolutional neural networks [41]. Pooling helps to reduce the size of the feature map, which reduces noise and redundant information and forces the model to focus on the important information in the data. Traditional pooling can and is used in graph neural networks. For example, it is common to use global max or mean pooling to obtain an embedding from a graph. However, we can also use pooling to reduce the size of the graph in between layers.

Since graphs are non-Euclidian, we have to adapt the pooling operators to them, as per [41] pooling operators on graphs can be seen as a combination of three functions: *selection*, *reduction* and *connection*. With selection we form clusters of nodes called supernodes, then with the reduction function we reduce the nodes in each supernode into a output node and in the connection function we connect the new output nodes thus obtaining a new graph.

3.5.2 Dropout in GNNs

Dropout is well established as a regularization method for Convolutional (CNNs) and Fully connected neural networks (FCNs), where it is applied during training after a layer to omit a part of units (neurons). Afterward, all units are used during testing. Using dropout is known to reduce over tting and improve performance [42].

We can also successfully apply dropout in graph neural networks. Two places where dropout is utilized involve the layer outputs, which are exactly the same as those used in CNNs and FCNs. The second place to apply dropout is utilized in GATs, where it is applied to the attention coe cients. This e ectively exposes each node during training to a stochastically sampled neighbourhood [1].

3.5.3 Tasks on graphs

We can de ne di erent tasks on graphs. The three most common categories would be node classi cation, graph classi cation, and relation prediction [33]. In node classi cation, we try to assign a label to each node in the graph (e.g., a graph where each node is a patient). In graph classi cation, we are trying to predict a label for an entire graph (e.g., toxicity prediction for drugs, where a graph of its molecular structure represents each drug), and in relation prediction, we are trying to predict new edges based on the existing ones (knowledge graph completion). Based on how we formulate our task, in this work, we will make use of mainly node classi cation.

Chapter 4

Classification from omic data

In this chapter, we deal with the problems that arise when using multi-omic datasets; this usually means that we have large numbers of features and different feature channels corresponding to the different omics. Thus, we have to be able to select useful features from the many available features and integrate features from different channels.

Multi-omic datasets typically contain many features m and few samples n, formally:

$$m >> n \tag{4.1}$$

This results from the fact that for biological datasets, it can be difficult to obtain samples (e.g., there are only that many tissue samples from patients with a specific type of cancer) but easy to obtain features as once we have a tissue sample, we can measure the expression of many different mRNAs, miRNAs or other sequencing data from it.

There are several rules of thumb for the ratio of samples to features, such as the one presented in [43], which recommends having

$$m = n - 1 \tag{4.2}$$

for uncorrelated features and

$$m = \sqrt{n} \tag{4.3}$$

for correlated features. Or the recommendation from [44], which for classi-cation tasks recommends

$$n \quad 10 \;\; C \;\; m \tag{4.4}$$

where C is the number of classes.

To provide an example with one of our datasets, for the BRCA dataset 7.3, we have $m = 43982$, $n = 483$, and $C = 4$. This would make the models slow to learn, and high-dimensional features are usually redundant and noisy [45]. So, we need to select features prior to building any model. Or utilize an algorithm that can select features during the learning process, such as XGBoost [46], if we hope to obtain a good model. Now, let us look at di erent approaches that are commonly used for reducing the number of features when $m >> n$.

4.1 Feature selection & extraction

We make the distinction between feature selection and feature extraction [47].

Given a feature vector $x\ 2\ R^F$ and a large pool of features F, the goal of feature selection is to select a subset of features f F and obtain a new vector $x\ 2\ R^f$ where usually $jf\,j << jF\,j$.

Given a feature vector $x\ 2\ R^F$ and a large pool of features F, the goal of feature extraction is to project $x\ 2\ R^F$ onto a lower dimensional subspace d as $x0\ 2\ R^d$, where $d << F$. The important distinction to make is that $x0\ 2\ R^d$ is not a subset of the original features but a di erent space.

4.1.1 Feature selection methods

Feature selection can be further divided into three di erent methods [43, 45, 47].

Filter methods happen prior to the model tting and can be super-vised like minimum redundancy maximum relevance [48], semi-supervised or unsupervised such as variance thresholding.

Embedded methods , where feature selection is part of the model and is utilized when tting the model. Examples of this would be L1 regularization, which is traditionally well utilized in generalized linear models like linear or logistic regression [49], but is also very well applicable to neural networks [50]. Another example is XGBoost [46].

And wrapper methods , which measure feature importance during model evaluation in a model-agnostic way. Examples of this class of methods would be recursive feature elimination (RFE) or permutation importance.

4.2 Feature integration

As we already mentioned, modern methods can generate vast amounts of multi-omic data. One of the assumptions of this work is that we can take heterogeneous sets of features in the form of di erent omic channels, integrate them, and use them inside machine-learning models to obtain better predictions than we would if we took any of those individually. Di erent papers de ne di erent categories, but a sensible consensus seems to be to separate the integration methods into early , mixed , and late integration [51, 52].

Early integration

We can either use early integration, where for each sample, we take each of its feature vectors $x_1 \in \mathbb{R}^{n_1}; x_2 \in \mathbb{R}^{n_2}; \ldots; x_n \in \mathbb{R}^{n_n}$ coming from N di erent omic channels and fuse them into a single feature via some transformation. A simple and often-used example of this is to concatenate them into one feature vector

$$x = [x_1 || x_2 || \ldots || x_n] \in \mathbb{R}^{n_1 + \cdots + n_n}; \tag{4.5}$$

where $||$ denotes concatenation. Then, we use this new feature vector x as an input to our model.

Other examples of this approach would be fusing all modalities into a heterogenous graph as done in [3][4] prior to feeding it to a GNN.

Middle integration

Middle integration uses a model that is able to learn a joint representation of the di erent omic channels. This approach is the most common in recent research [53].

An example of this approach would be creating sample-sample similarity graphs for different modalities and then fusing them via similarity network fusion algorithms and fusing the different omic feature vectors into a joint embedding via an autoencoder as done in [54].

Late integration

Another approach is late integration, where we build a model $f_i(\cdot)$ for each $x_i, \forall i \in \{1, ..., N\}$, where N is the number of channels, to produce an embedding for each sample:

$$z_i = f_i(x_i) \tag{4.6}$$

Then, given an embedding z_i for a sample x_i we use some integration model $g(\cdot)$ to integrate these into a final prediction

$$y_i = g(z_i) \tag{4.7}$$

to obtain a final prediction y_i for the sample.

Several methods have been proposed for the integration of predictions or embeddings. A simple but often used approach [2][55] is concatenating these vectors and using them as an input to a shallow (often single-layer) neural net. More methods are discussed later in 6.1.

Chapter 5

State of the art GNN approaches

Here, we provide an overview of several notable papers that use GNNs for multi-omic data. The tasks concerning us are mostly cancer type/subtype classification and disease classification. We are also interested in how the methods are able to score features for biomarker discovery.

This section is thus split into two main parts. The first describes recent methods and categorizes them by their choices for building the input graph. The second discusses the approaches used to score features and discover biomarkers and their similarities and differences across different methods.

5.1 Building input graph

Based on available literature, we can divide existing GNN methods into three categories based on how they choose to build the input graph [56].

A common choice is to build the graph based on sample-sample similarity [2, 5, 54, 55]. This can be as simple as thresholding the cosine similarity between samples. In these types of methods, a node represents a patient, and the goal is to perform node classification on the graph.

Other methods choose to include feature-feature similarity [3, 4, 57]. Which for omics datasets means to explore the existing databases with gene-gene interactions [58], protein-protein interactions [59] or miRNA-mRNA interactions [17, 18, 19, 20, 21, 22, 60]. Here, the graph will be the same for all samples, but for each sample, different feature vectors will be placed in each node, which here represents features. The goal is then to produce a graph-level prediction for the sample; this is graph classification.

The last approach is to combine these methods and build a heterogenous graph that contains both nodes that represent samples and nodes that represent features. Such a graph can then contain different relations between the nodes [61].

5.1.1 Sample based graphs

The simplest method for building these kinds of graphs is to compute some similarity measure between two samples and then connect them if they are similar enough. In this work, we will reference this approach as *thresholded similarity sample-sample graph* .

A common choice is to use the cosine similarity

$$s(x_i; x_j) = \frac{x_i \cdot x_j}{||x_i|| \, ||x_j||}; \tag{5.1}$$

and then threshold it based on some hyperparameter epsilon. Based on this, we can construct the adjacency matrix like this [2, 55, 62]

$$A_{ij} = \begin{cases} s(x_i; x_j); & \text{if } i \neq j \text{ and } s(x_i; x_j) \geq \epsilon \\ 0; & \text{otherwise} \end{cases} : \tag{5.2}$$

A similar approach is used in the MOGLAM paper [5], where in addition to the base graph another graph is contructed via a learned similarity measure

$$s(W_s x_i; W_s x_j) = \frac{W_s x_i \cdot W_s x_j}{\|W_s x_i\|_2 \, \|W_s x_j\|_2}; \tag{5.3}$$

and then from this, a learned graph is constructed

$$S_{ij} = \begin{cases} s(W_s x_i; W_s x_j); & \text{if } i \neq j \quad \text{and } \cos(W_s x_i; W_s x_j) \geq \epsilon \\ 0; & \text{otherwise} \end{cases} ; \tag{5.4}$$

this learned graph is then merged with the base graph with a hyperparameter $\alpha \in [0; 1]$

$$\hat{A} = \alpha A + (1 - \alpha) S: \tag{5.5}$$

This graph is constructed for each omic channel. Afterward, a modified GCN model named FSDGCNs 5.3.2 is applied to each channel as an encoder, producing an embedding for each omic. Afterward, a model that is similar to the attention integrator model described in this work 6.1 (though with minor differences) is applied to combine these embeddings into a final prediction.

In MoGCN [54], a thresholded similarity sample-sample graph is constructed for each omic channel, though here they use scaled exponential similarity

$$W(i,j) = \exp\left(-\frac{\|v_i - v_j\|^2}{\mu\,\varepsilon_{i,j}}\right); \tag{5.6}$$

where μ is a hyperparameter and $\varepsilon_{i,j}$ is a scaling factor. These graphs are then merged together into a single graph utilizing similarity network fusion algorithms. In parallel, a joint embedding is constructed from all the omic features using an autoencoder. This joint embedding is then placed into the fused graph, and a GCN is used to predict node labels.

5.1.2 Feature based graphs

A different way to build the input graph is to rely only on interaction networks between different biomolecules such as mRNA, miRNA, and proteins. Here, usually, this information is used to construct a complex interaction graph that is then used as an input to a graph neural network.

Li and Nabavi [3] used this approach to construct a feature-based graph from gene-gene interactions, protein-protein interactions, and co-expression networks from RNA expression and copy number variations (CNV). An autoencoder consisting of a GCN applied to the aforementioned graph as an encoder and a linear layer as a decoder is applied to obtain a joint embedding of the input. The GCN-encoded embedding is then concatenated with another embedding vector obtained with a shallow linear neural network into a final embedding vector that is then used as an input into a final classification linear layer.

Later Li and Nabavi [4] improved upon their method. This time, they construct the input graph from three omic channels, mRNA, CNV, and miRNA, and add edges for gene-gene and miRNA-gene interactions. The heterogeneous graph now contains three types of nodes: mRNA, CNV node, and miRNA nodes. Since mRNA, CNV nodes have an embedding of $x_i \in \mathbb{R}^2$ and miRNA nodes have $x_i \in \mathbb{R}^1$ the embeddings are lifted to a higher, but equal dimension with a linear layer. The rest of the method is analogous to their previous one. They use an autoencoder, which employs a GAT layer encoder and a linear decoder, to obtain a joint embedding. The embedding is then concatenated with another joint embedding obtained from a shallow linear neural network, and the final concatenated vector is then used as input to a final linear classification layer.

Xiao et al. [57] used a similar approach; they constructed a feature graph from gene-gene and protein-protein interactions, placing m dimensional feature vectors in each node, where m is the number of omic channels. Afterward, a GCN encoder is used to produce an embedding from this graph. A sample-level module in the form of a linear neural network is used in parallel to produce another embedding. Both are concatenated and passed through a final classification linear neural network. An interesting specificity of the method is that they make use of supervised contrastive loss to learn the graph embeddings.

One disadvantage of [57] is that their method requires all the omic channels to be of the same dimension and contain the same features (genes). This is not practical as we commonly have sequencing data available for thousands of genes but only for hundreds of miRNAs and maybe even less for other types of RNAs.

In Ramirez et al. [63], two kinds of input graphs are considered. First is a co-expression graph computed from Spearman Correlations between gene expressions in different tumors (classes). Second uses the String DB [59] and connects the genes in a graph based on whether the proteins produced by these genes interact. The findings indicate that the model performs similarly on both input graphs. The inputs are 1D expression vectors placed in the corresponding gene nodes. The task here is thus graph classification, for which Graph Convolutional layers with graph pooling were used.

5.1.3 Sample-feature graphs

A different approach is taken in [61]. For each different omic channel (gene expression, copy number variations) they construct a bipartite graph, where nodes represent either a sample (patient) or a feature (gene). Then, samples are connected to features based on defined relationships, differential expression for mRNA expression, and thresholded CNA numbers for CNA.

A different GCN is then applied to each input graph to produce an omic-specific embedding. These embeddings are then fused together with a linear layer neural network to produce a final prediction.

5.2 GNN layers

The methods described earlier in this chapter utilize either GCN layers or GAT layers. Existing research suggests that approaches employing GATs [4, 5, 57] exhibit superior performance compared to those that do not [2, 3, 55]. This superiority can be attributed to the theoretically greater modeling capacity offered by GATs [1, 34, 37]. We were unable to identify any methods utilizing relational layers [39, 40], prompting us to explore their efficacy in our study.

5.3 Feature importance in GNN approaches

The end goal of the tasks outlined in this work, such as cancer subtype classification, is to identify important biomarkers that are the main differentiating factors between the different classes. If these biomarkers are identified, then additional research can be carried out to explore their functions and help us better understand what causes different pathological states. Identifying these markers in silico is several orders of magnitude faster and cheaper than trying to do the same in a lab.

As we already mentioned, if a model is able to achieve good classification performance, it is also important that we obtain some feature importance scores that would suggest which features the model considers important. In this section, we provide an overview of the different approaches that can be used to compute these feature importance scores from Neural Networks with a focus on GNNs.

5.3.1 Wrapper methods

One approach is to use feature permutation or ablation . We remove the signal of one feature from the model in and measure the decrease in performance. This can be done in principle by either randomly permutating the feature or by setting it to zero if the inputs are scaled to $[0;1]$ with min-max scaling. The decrease in performance is then assumed to be directly proportional to feature importance. This is repeated for every feature. In MOGONET and MORONET [2, 55], the performance is measured on the test set, and for robustness, 5-fold cross-validation is utilized, and the feature scores are accumulated across all folds. The feature scores are further scaled by the amount of features selected from each channel.

A similar approach is taken in [63] with the difference that the drop in performance is computed with a) setting the feature to 0 for all samples and

b) setting the feature to 1 for all samples, with the assumption that the input is scaled with MinMax scaling between 0 and 1. Afterward, the bigger drop in performance is taken as the score for that feature.

5.3.2 Embedded methods

In GCFANet [6], a feature con dence learning module (FCLM) is presented. They use an encoder $f(\ )$ in the form of a linear layer to produce a feature weight vector.

$$w = \ (f(x)): \tag{5.7}$$

The weight vector is then used to update the importance of each feature with an element-wise product to produce an updated vector.

$$\hat{x} = w \quad x; \tag{5.8}$$

which is then used further in the model. A sigmoid activation $(\ )$ is used to constrain the weight values between 0 and 1. L1 regularization is imposed onto the weight vector to keep it sparse. The motivation here is that the model will learn to assign high values to important features and silence the redundant, uninformative ones.

In MOGLAM [5], a dynamic graph neural network with feature selection (FSDGCN) is proposed. Given the rst layer in a GCN

$$\hat{H}^1 = \quad \hat{A}H^0W_1 \quad ; \tag{5.9}$$

with weight matrix W_1 they apply inner product regularization [64] to the weight matrix which is formulated as

$$(W) = \|W_1W_1^T\|_1 \quad \|W_1\|_2^2; \tag{5.10}$$

and added to the loss function with a corresponding hyperparameter that determines the strength of the regularization. Then, we can take the col-wise sum of W to compute how much each feature from H^0 contributes to positions in the embedding H^1. This gives us a score for each feature. After this initial layer, several normal GCN layers are added.

In principle, di erent regularization schemes could be applied, such as L1, but the inner matrix regularization paper [64] claims that inner matrix regularization has some advantages over those. Notably, it is designed to introduce sparsity in the feature scores.

Chapter 6

Methods

This section reviews the methods and architectures used and proposed in this work. First, let us look at the building blocks. Following the earlier definitions, we present several architectures that can be used to integrate feature vectors from different channels.

6.1 Integrators

Here, we describe three different integration modules to integrate embeddings for different omic channels into a single prediction.

Linear integrator

The first and simplest is linear integrator, where we concatenate the vectors across different channels $\{x_1, \ldots, x_m\}$ and pass them through a linear layer to obtain a prediction $\hat{y}$

$$\hat{y} = W \bigoplus_{i=1}^{M} x_i + b \; . \tag{6.1}$$

View Correlation Discovery Network

Second is View Correlation Discovery Network (VCDN) [65], which was initially proposed in a computer vision task a two-feature setup and then generalized to 3 or more dimensions in the MOGONET paper [2]. In the context of this work, let us assume we use a different model as an encoder for each feature channel. Given that we have m omic channels, we will have m encoders. The i-th encoder $f_i(\cdot)$ then takes the the feature vector of the j-th sample $x_j \in \mathbb{R}^{o_i}$ and produces either a label distribution or an embedding $\hat{y}_j^{(i)} = f_i(x_j)$. Then, given that we have an embedding for each omic channel $\hat{y}_j^{(i)}, i \in \{1, \ldots, m\}$ we construct a cross-feature discovery tensor $C_j \in \mathbb{R}^{o_1 \times \cdots \times o_n}$ where we calculate each entry in the cross-feature discovery tensor as

$$C_{j,a_1,\ldots,a_n} = \prod_{i=1}^{n} \hat{y}_{j,a_i}^{(i)}. \tag{6.2}$$

This tensor is then used as an input to a fully connected network to obtain a final prediction $\hat{y}_j$. The loss function for the VCDN can be expressed as

$$L_{VCDN} = \sum_{j=1}^{n} L_{CE}(VCDN(C_j), y_j), \tag{6.3}$$

where L_{CE} is the cross entropy loss and y_j is the true label for the j-th sample. The theoretical advantage of this approach over the simple concatenation is that this tensor captures the correlations in the label/embedding space, and the neural net should discover these correlations to produce more accurate outputs. Note that in the MOGONET paper [2] where this approach is presented, they integrate these outputs only in the label space. In this work, we further extend this approach by exploring whether increasing the dimension of the integrated vectors can be beneficial.

Attention integrator

Third is Attention integrator, which can be seen as an extension to the Linear integrator where we add scaled-dot product attention [66].

Figure 6.1: Attention integrator diagram

Given $\{x_1; \ldots; x_m\}$ omic embeddings for a sample s, we first apply the attention mechanism between the different omic embeddings. Then, we pass the results through a linear layer before concatenating them and passing them through another linear layer, which outputs classification logits. 6.1

6.2 Bipartite GNN

The proposed method constructs a bipartite graph between the samples and genes based on differential gene expression and uses a relational graph attention network (RGAT). This architecture is inspired by previous work [61], but we further extend it with several different ideas.

Firstly, the previous work [61] uses only the bipartite graphs. Since we represent features as nodes in the network, we find it natural to add interaction edges as shown in figure 6.2. The second improvement lies in using a different integration scheme. The original work uses a linear integrator. We find using a different integrator, such as VCDN or attention integrator, beneficial.

6.2.1 Graph construction

The heterogeneous input graph $G = (V; E)$ is constructed as follows. It contains both nodes, which represent the samples (patients) $v_{s_{mi}}$, and features (genes, miRNAs) $v_{f_{mi}}$. Each node $v_{j_{mi}} \in V$ contains a feature vector $h_{j_{mi}}$, where 6.2 $h_{s_{mi}}$ denotes the feature vector u for the i-th sample of the m-th omic channel and $h_{f_{mi}}$ denotes the the feature vector u for the i-th feature of the m-th omic channel.

Figure 6.2: Example heterogeneous graph for three samples and two omics with two features each, intra-omic interaction in blue connects $h_{f_{11}}$ and $h_{f_{12}}$, inter-omic interaction in green connects connects $h_{f_{12}}$ and $h_{f_{22}}$

The heterogeneous graph contains multiple types of relations, illustrated with di erent colors in Figure 6.2. The sample-feature edges $E(V_{s_{mi}}; V_{f_{mi}})$ (shown in red and orange) are constructed through di erential expression analysis. The edge creation process follows these steps:

1. A distribution is tted to the gene counts to model the expected expression levels.

2. For each feature-sample pair $(f_i; s_i)$, we compute a z-score measuring the deviation of the observed feature value from the distribution's expected value.

3. A hyperparameter serves as a z-score threshold. An edge is created between sample s_i and feature f_i if

$$|z_{f_i;s_i}| > \; ; \tag{6.4}$$

where $z_{f_i;s_i}$ represents the standardized distance between the observed feature value and the expected value under the tted distribution.

Gene expression data is traditionally modeled using a negative binomial distribution [67], e ectively capturing the overdispersion characteristic of RNA sequencing counts. In our approach, we t a single distribution across all samples despite these samples originating from di erent experimental conditions. This e ectively means we are tting a distribution to what is technically a mixture of di erent underlying distributions. We experimented with using both normal and negative binomial distributions for this purpose. Our results indicate that the choice between these distributions has negligible impact on the model's performance, suggesting that the precise form of the distribution is less critical than the ability to capture signi cant deviations from the expected expression values.

Gene-gene interaction database [58], protein-protein interaction database [59] and miRNA-gene interaction databases [17, 68] are used to construct feature interaction edges $E(V_{f_{mi}}; V_{f_{ni}})$ for both inter $m = n$ (blue edge) and intra $m \neq n$ (green edge) interactions between the corresponding features. The addition of both inter and intra-omic edges is, to the best of our knowledge, novel to our method and not explored in previous works utilizing similar methods such as [61].

6.3 Model architecture

Here, we present the model architecture.

Figure 6.3: Bipartite graph architecture

Di erent omics can contain di erent numbers of features. Since our input graph can take advantage of both intra and inter-omic interactions, we require all feature vectors x_i in the graph to be of the same dimension d so that message passing is possible between any two feature nodes in the graph. We propose two approaches that aim to deal with this. The rst is to project the features onto a common dimension, and the second is to select the same number of features from each channel. This is further discussed in the results chapter.

$$\forall u_{j\,mi} \in V(G) \quad u_{j\,mi} \in \mathbb{R}^d \tag{6.5}$$

This is accomplished by first projecting each sample feature vector with an omic-specific projection module $f_m()$ onto a common embedding dimension d

$$\hat{u}_{s_{mi}} = f_m(u_{s_{mi}}) \quad u_{s_{mi}} \in \mathbb{R}^n; \ \hat{u}_{s_{mi}} \in \mathbb{R}^d; \tag{6.6}$$

and initializing the feature nodes with vectors of ones of that same dimension d

$$h_{f_{mi}} = 1^d \quad \forall m \in M; i \in U: \tag{6.7}$$

Where M is the set of all omics and U is the set of all features from omic M.

Once the graph is initialized, several graph neural network layers (we experiment with 2 and 3) propagate information through the graph. Afterward, the sample embeddings are collected into a tensor, which gives us m feature vectors x_i for each sample i. These embeddings are integrated into a single prediction using an integration module.

6.3.1 RGAT layers

Since we have constructed a heterogenous graph that contains different relations such as (mrna sample; mrna gene) or (miRNA sample; mrna gene) we can increase the expressiveness of the model by modeling each relation with a different weight matrix, this approach is described earlier as relational graph attention networks.

Neural network training

Datasets with unbalanced class counts can lead to poor performance on less common classes. To amend this, we use a weighted loss function and place larger weights on less common classes and smaller weights on more common classes.

Chapter 7

Data

Here, we briefly review the datasets used in this work and the preprocessing applied to them.

7.1 Preprocessing sequencing data

It is common to apply preprocessing to raw sequencing counts before using them for any analysis. Two such commonly used preprocessing are TMP (transcripts per million) [69]:

$$\mathrm{TPM}_i = \frac{q_i / l_i}{\sum_j (q_j / l_j)} \cdot 10^6, \tag{7.1}$$

where q_i denotes the number of reads mapped to a sample for a gene i and l_i is the transcript length for that gene. The second one is RPKM/FPKM (reads / fragments per kilobase million) [69]:

$$\mathrm{RPKM}_i \text{ or } \mathrm{FPKM}_i = \frac{q_i}{l_i \sum_j q_j} \cdot 10^9. \tag{7.2}$$

The difference between reads and fragments is that fragments come up in paired-end sequencing, where we have to account for two different reads mapping to the same location in the genome. Other than that, these two are the same.

Afterwards, it is common to log-transform the normalized counts via

$$\log(x + 1)\,; \tag{7.3}$$

logarithm is used as a variance stabilizing transformation since gene expressions are usually modeled with a negative binomial distribution, and a pseudocount of 1 is added to handle undetected genes (zeroes) [70].

7.2 Myelodysplastic syndrome

Myelodysplastic syndromes (MDS) are a group of clonal bone marrow disorders characterized by ine ective hematopoiesis, leading to peripheral blood cytopenias and a risk of progression to acute myeloid leukemia (AML) [71]. The data has been obtained from Ústav Hematologie a Krevní Transf·ze (UHKT) [72, 73].

The dataset contains molecular pro les of 98 patients. These include mRNA, miRNA, piRNA, and circRNA sequencing data, as well as the Transposable Elements counts (TE counts). Using samples for which all data is available, we get 74 fully annotated samples if we use mRNA, miRNA, circRNA, and TE counts. PiRNAs are missing the most annotations, so with them added, we get only 66 usable samples; for this reason, we exclude them from the analysis; with such low sample counts, we decide that having more samples is more important than having one more omic channel.

mRNA	miRNA	circRNA	piRNA	TE counts
19874	1880	70	556	688

Table 7.1: Available features per channel.

On this dataset, three binary classi cation tasks are de ned. The rst one is aimed at distinguishing between healthy controls and patients with MDS. There are already well-established biomarkers based on which this classi cation can be made, so this task is expected to be quite easy [74].

classes	healthy control	diseased
sample counts	13	61

Table 7.2: Disease.

In the second one, we di erentiate between low- and high-risk patients. MDS is a chronic, progressive disease. Based on the International Prognosis Scoring System (IPSS), the patients are split into two categories for low and high risk [75].

classes	low risk	high risk
sample counts	29	24

Table 7.3: Risk.

In the third task, we are trying to di erentiate between patients with a mutation on the SF3B1 splicing factor and patients without that mutation. Since the fth edition of the World Health Organization (WHO) classi cation of haematolymphoid tumors, MDS with the mutated SF3B1 gene is recognized as an independent subtype. The mutation of the SF3B1 gene is associated with good prognosis [76].

classes	not mutated (wild type)	mutated
sample counts	14	12

Table 7.4: Mutation.

7.3 Breast cancer (TCGA-BRCA)

The TCGA-BRCA dataset for breast invasive carcinoma, de ned in [77] classi es breast cancer into ve molecular subtypes de ned by a 50-gene signature called the PAM50 model. The classes are Basal-like (98 samples), HER2-enriched (58 samples), Luminal A (231 samples), Luminal B (127 samples), and Normal-like (8 samples). The Normal-like class contains only eight samples, so it was excluded from the dataset for further analysis.

The dataset contains omic data for copy number variations (CNV), DNA methylation, gene expression (RNASeq), micro RNA (miRNA mature strand expression RNAseq), pathway activity, protein expression, and somatic mutations. Of those, we used copy number variations, DNA methylation, gene expression, and micro RNA expression. The data is available preprocessed (normalized & log-transformed). Furthermore, we kept only samples for which data was available for all of the used omic channels.

The dataset was acquired trough the USCS Xena portal [78].

Luminal A	Luminal B	HER2 enriched	Basal-like	Total samples
220	121	55	87	483

Table 7.5: Sample counts for used TCGA-BRCA classes after preprocessing.

We applied further preprocessing to the data. mRNA and copy number variations (CNV) are measured at the level of individual genes. Methylation data is measured per probe. Each probe measures the methylation of the DNA in a small region, and the dataset also contains a mapping of the probes to genes. Since multiple probes can map to one gene, we take the average of those. If they do, this gives us methylation values at the gene level. Furthermore, we only keep those genes for which mRNA expression, CNV, and methylation values are available. The resulting feature counts are available in the table below.

mRNA	miRNA	CNV	DNA methylation
10736	237	10736	10736

Table 7.6: Available features per channel after preprocessing.

Chapter 8

Results

In this chapter we present the results, this involves benchmarking and comparison of our methods across several diverse datasets. We also present and validate the ability of our method to identify signi cant biomarkers.

For implementing the KNN and SVM baseline models the scikit-learn [79] library was used. XGBoost is implemented using the dmcl xgboost package [80]. For deep learning and graph neural networks we used the PyTorch [81] and the PyTorch Geometric [38] libraries.

8.1 Evaluation & Hyperparameter tuning

As baselines for classical methods we choose three models, k-nearest neighbours, linear support vector machines with recursive feature elimination and XGBoost [46].

All these algorithms usually require dataset-speci c tuning of di erent parameters to t the speci c dataset well and using default parameters may add a pesimistic bias to their results thus making our proposed algorithms look better in comparision [82].

For this reason we apply rigourous model selection to the baseline models using the hyperoptimization framework Optuna [83]. Optuna allows us to de ne a custom objective function which evaluates the model and returns a single number measuring the performance of that model. Inside the objective function we sample the parameters and afterwards evaluate the performance of the model with 5-fold cross validation, and use the mean of some robust model performance measure across the folds such as weighted f1 to measure the performance. Afterwards we select the hyperparameters which maximise this metric.

Internally optuna implements various black-box optmization algorithms which can be as simple as grid-search or random search, or more advanced like Tree-structured Parzen Estimator [84] (default) or NSGAII [85]. It also implements pruners, which allow us to terminate the trial early in case the model fitting is significantly worse than the best obtained result. These techniques combined allow us to find a good set of parameters for a model reasonably fast, reducing the pessimistic bias of estimates for the baseline model in the process.

We can utilize the same approach for the deep learning models as well, where we can use it to tune the many hyperparameters deep learning models usually have including the number of layers, size of layers, learning rate or dropout. We can also use it to tune categorical parameters, like which type of integration module should our model use.

Overall we evaluate all models using an approach that is described using the following pseudocode. Generally the more trials we do the higher the probability that we get a good model. Practically 20-30 trials seems to be a good trade off between speed and performance.

```
init num_trials = N
init best_performance

for each trial in 1 to N:

    1. sample hyperparameters (optuna)

    for each cross validation split:
        2. split data into training and testing subset
        3. feature selection based on the train set
        4. normalize features based on the train set
        5. fit model
        6. test model
        7. record performance for this split

    8. Average performance across all folds
    9. If the current cross validated performance
    is better than the best_performance
    record new best_performance

return best_performance
```

8.1.1 Feature selection

As a first step we use the following pipeline for preprocessing RNA sequencing data analysis. We employ multiple filtering criteria to filter out genes which are not at all or only minimally expressed in our dataset, this is a common step that has backing in biology, not all genes are expressed in every tissue.

First we filter the features by detection rate, i.e. the fraction of samples where the gene is expressed, we find 20 % to be a reasonable value for this filtration. Additionally we filter the features by minimum median expression thresholding to exclude genes with overall low counts, the parameter used for this ranges from 10-100 and depends on the type of sequencing data. If more than the desired number of features remain after the filtering steps we order the features by the coefficient of variance, which has the nice property that it is scale free. This property is very useful considering that read counts for different genes can differ by up to 4 orders of magnitude. Then we select the top n features by this ranking, where n is usually in the order of thousands. Then we normalize the features with counts per million (CPM) normalization. And finally we apply log2 transformation to stabilize the variance and normalize the data. These steps can be summarized as follows.

1. Detection rate filtering
2. Minimum median expression filtering
3. Optionally sort features by Coefficient Of Variance and select top n most varied ones
4. Normalize with CPM (counts per million)
5. Apply log transformation

For additional feature selection we apply Maximum relevance, minimum redundance (mRMR) [48]. This method is theoretically much more powerful as it selects features using mutual information which means that it makes no assumptions about the underlying distribution. It is also known to work very well in practice [86]. The disadvantage of this method is that computing mutual information is inherently expensive and the methods scales unfavorably with the number of features. For practical purposes it is useful to first select a reasonably informative subset of features with a weaker (but much faster) method such as the variance ranking and then apply mRMR. We find this sequential feature selection to be reasonably fast, while also producing good features.

8.1.2 Neural network baseline and the effects of regularization

A simple shallow neural network baseline, here we use two or three layers, will perform noticeably worse than if we use an algorithm like XGBoost which is overall a very good method to use on tabular data and often beats even much more complex deep learning methods [87].

Here we show that for omic data this mostly comes down to the fact that omic data contains many redundant or unimportant features and that we can largely close this gap by imposing L1 regularization onto the initial layer in the network.

These techniques have been applied before for example for using step-wise l1 l2 linear regression to select variables from noisy datasets [49]. L1 regularization in shallow linear neural networks also has theoretical advantages in tighter statistical risk bounds [50].

For our classification tasks we are using the cross entropy loss function. Given a linear neural network (MLP) with parameters $\{W_1, \ldots, W_n\}$ and $\{b_1, \ldots, b_n\}$ we can impose L1 regularization onto the first layer in the network by adding the L1 norm of the first weight matrix to the loss function

$$L = -\sum_{n=1}^{N} \sum_{c=1}^{C} \omega_c \log(\hat{y}_{n,c}) y_{n,c} + \lambda \|W_1\|_1. \tag{8.1}$$

where λ is a hyperparameter, that defines the strength of the regularization.

Note the use of class weights ω_c in the cross entropy loss, this is useful for imbalanced datasets generally and especially useful later for training the GNN based architectures on imbalanced datasets. In cases where we use sample based graphs all samples are present in the input graph and we train the model by masking nodes and training on the training nodes only. With this approach using techniques like batch balancing for ensuring equal prior class probabilities is not possible so we use weighted loss instead.

Besides increasing performance this provides the advantage of performing feature selection. Pytorch represents a linear layer like [88]

$$y = x \cdot A^T + b, \tag{8.2}$$

so if we then take the col-wise sum of W_1 we can measure the relative importance of each input feature to the rest of the network and since we have used L1 regularization less informative features should get lower scores as their coe cients will get more often set to zero. [54].

Model	Accuracy	F1 macro	F1 weighted
MLP, no regularization	0:84 0:05	0:82 0:06	0:84 0:05
MLP, l1 regularization	0:85 0:05	0:84 0:06	0:85 0:05
MLP, inner matrix	0:85 0:03	0:84 0:03	0:85 0:03

Table 8.1: E ects of regularization on MLP, classi cation from 10000 genes on the BRCA dataset

When using large feature counts regularization is helpful, and this e ect will only get stronger if we have a noisier dataset. And at the very least regularization does not make performance of the model any worse, while it is able to provide information about which features the model considers important.

8.1.3 Bipartite GNN

Here we present the experiments and reason the architectural decisions for the Bipartite GNN model.

Adding interactions

Here we show the in uence of adding intra and inter omic interactions to the model.

The performance metrics show remarkably consistent accuracy across all model con gurations, with all variants achieving 96% accuracy except for the three-layer model without interactions, which performed slightly lower at 94.7%. Interestingly, the addition of interaction terms appears to reduce the variance in the metrics, as evidenced by the lower standard deviations in the interaction-based models.

Model Con guration	Accuracy	F1 Macro	F1 Weighted
Two layers - no interactions	0:96 0:05	0:94 0:07	0:96 0:05
Three layers - no interactions	0:95 0:05	0:91 0:08	0:95 0:05
Two layers - interactions	0:96 0:03	0:93 0:06	0:96 0:04
Three layers - interactions	0:96 0:03	0:93 0:06	0:96 0:03

Table 8.2: Performance metrics of various model con gurations. MDS dataset disease task.

We also repeat the same evaluation for the BRCA dataset. In this experiment we are using mRNA and miRNA, with 500 mRNA features and 277 miRNA features. We observe similar results, adding omic interactions provides a small increase in performance of about 1 % in all metrics, though this is well within the standard deviation of the estimates. Same as in the previous experiment on the MDS dataset adding the omic interactions approximately halves the variance in the metric estimates.

Model Con guration	Accuracy	F1 Macro	F1 Weighted
Two layers - no interactions	0:917 0:024	0:913 0:024	0:917 0:023
Three layers - no interactions	0:867 0:033	0:849 0:027	0:865 0:034
Two layers - interactions	0:923 0:010	0:926 0:011	0:923 0:010
Three layers - interactions	0:875 0:016	0:869 0:017	0:873 0:017

Table 8.3: Performance metrics of various model con gurations. BRCA dataset.

We also compare how the inclusion of omic interactions a ects the network if we use either two or three layers. If we use two layers than the longest path in the graph is (sample node -> feature node -> sample node) and if three other paths that include feature interactions are possible (sample node -> feature node -> feature node -> sample node). So if the passing of information between sample nodes trough feature interactions was important for the model the three layer con guration should perform better than the two layer con guration. Based on the previous results this does not seem to be the case in our experiments.

Two or Three layers?

In the previous section we also experiment with using two vs three layers. For MDS dataset there does not seem to be a di erence between using either two or three layers. For BRCA dataset using three layers performs slightly worse than using two. It is possible that the performance of three layers could increase if we trained the network for more epochs, however since for the MDS Disease task the performance is equal and training the two layer network is faster in all other experiments we work with two layers.

Projection module vs feature selection

One problem we have to deal with is that di erent omic channels will have di erent numbers of features. Typically, there are many more genes than miRNAs, circRNAs, and piRNAs. We explore two approaches that can deal with this.

The first approach uses a feature selection method to select the same number of features from each channel. We choose to use maximum relevance minimum redundancy (mRMR) [48], this method is particularly suitable for this task as its aim is to select small feature sets that carry as much information as possible. One disadvantage of this method is that the number of features we can select is upper-bounded by the number of features in the smallest feature set (omic channel), which may be disadvantageous.

Another approach we can use is to employ the regularized projection layers. Using a different encoder for each channel, we project all features onto the same dimensions and then use those as input features for the input graph. The advantage of this approach is that we can select a different number of input features for each channel. An indirect disadvantage of this approach is that the number of input features still cannot be very large as the number of features directly impacts the structure of the input graph due to the choice of representing each feature with a node, and if we have too many redundant input features the graph will become noisy which will degrade the performance of the model and large which will negatively impact training times.

We evaluate both approaches on the BRCA dataset. All model configurations use mRNA and miRNA as model inputs with the interactions included. The projection module is a single layer linear neural network with L1 regularization. We compare the performance of a model that uses 200 mRNA and 200 miRNA features and thus doesn't need the projection layers, a model that uses a projection from 500 mRMR selected mRNA features and all 277 miRNA features, a model that uses a projection from a 1000 variance filtering selected mRNA features and all 277 miRNA features and a model that uses a projection from 2000 mRMR selected mRNA features and all 277 miRNA features. This is summarized in the table below.

Model	mRNA Features	miRNA Features	Proj. module
Projection: 2000 to 200	2000 (mRMR)	2777 (All)	Yes
Projection: 1000 to 200	1000 (Variance)	277 (All)	Yes
Projection: 500 to 200	500 (mRMR)	277 (All)	Yes
Selected Features	200 (mRMR)	200 (mRMR)	Yes
Selected Features	200 (mRMR)	200 (mRMR)	No

Table 8.4: Feature selection and projection approaches for different models

Table 8.5 summarizes the resulting performances.

We find that the projection module provides a substantial improvement in performance and that the amount of input features is also important. Projections that use more features do not work nearly as well, probably due to the graph becoming too large and noisy. Nonetheless these results are promising as using the projection modules also provides a way to assign

Model	Accuracy	F1 macro	F1 weighted
Projection 2000 to 200	0:865 0:026	0:860 0:023	0:866 0:025
Projection 1000 to 200	0:867 0:028	0:867 0:022	0:866 0:030
Projection 500 to 200	0:917 0:009	0:920 0:008	0:916 0:010
200 w projection	0:915 0:010	0:917 0:017	0:915 0:010
200 w/o projection	0:763 0:049	0:754 0:058	0:761 0:052

Table 8.5: Comparison of using projection vs selection on the BRCA dataset

feature scores if a regularization such as L1 or inner matrix product is used. A nal note is that the projection module might be better used with a model that is less sensitive to the number of features used than Bipartite GNN is.

8.1.4 Improving MOGONET

We believe that the MOGONET approach can be improved, methods with superior performance [4, 5] use GAT layers which o er increased modeling capabilities compared to the simpler GCN layers. So the rst proposal is to swap the layers in the omic encoder modules from GCNs to GATs. Speci cally we use the improved GATv2 layer as described earlier. This is evaluated on the BRCA dataset. We can see that this results in a substantial improvement.

Model	Accuracy	F1 macro	F1 weighted
MOGONET GCN	0:86 0:04	0:86 0:07	0:86 0:04
MOGONET GAT	0:91 0:02	0:91 0:03	0:90 0:02

Table 8.6: Performance gain from using GAT layers on the BRCA dataset

Second proposal is to experiment with a di erent graph building approach, instead of keeping edges with similarity above a certain threshold we propose connecting each node with k most similar neighbours, where k is a hyperparameter. We nd that this method is inferior to the thresholding method. This is evaluated across varying degrees an example of the di erence is shown in the table for mean degree of 10.

Model	Accuracy	F1 macro	F1 weighted
MOGONET knn	0:85 0:05	0:84 0:04	0:85 0:05
MOGONET threshold	0:91 0:02	0:91 0:03	0:90 0:02

Table 8.7: Comparison between using a thresholding based graph and k-nearest neighbours based graph, mean degree is 10 for both graphs

Finally we experiment whether the attention integrator provides any improvements over the previously used VCDN. For completeness we also evaluate this against the simple concatenation based integrator. We nd, that utilizing the attention integrator yields a small improvement in performance. However the more important result is that using attention for integrating the omic embeddings over VCDN does not degrade the performance and scales more favorably with the number of omic channels. The dimension of the correlation tensor constructed with VCDN is $R^{n_1 \quad n_2 \qquad n_m}$ where n is the dimension of the embedding vectors and m is the number of omics, which means it grows exponentially with the number of omics. Whereas for attention the matrix of attention scores is $R^{m \quad m}$ which gives it a much more favorable quadratic scaling with respect to the number of omics being integrated. This makes the training of the network with the attention integrator noticeably faster.

Model	Accuracy	F1 macro	F1 weighted
MOGONET linear	0:87 0:02	0:86 0:03	0:87 0:02
MOGONET VCDN	0:88 0:01	0:89 0:01	0:88 0:01
MOGONET attention	0:91 0:02	0:91 0:03	0:90 0:02

Table 8.8: Comparision between using di erent integrators

So in summary we found that using GATs over GCNs as encoders for each omic channel and swapping the integration model we are able to improve the MOGONET [2] model. This improved version of the model is what is used in the classi cation and biomarker mining experiments.

8.2 Classi cation performance

In this section we compare the performance of di erent algortithms on our datasets.

8.2.1 Myelodysplasic syndrome (MDS)

This dataset has very small sample counts, 74 for the Disease task, 53 for the Risk task and 26 for the Mutation task. As such the all models su er from large variances.

Disease

Here we present the results for the Disease task. First we t each model using only a single omic channel. This gives us an idea about how useful each omic channel is to each model. Next we explore wheter combining multiple omics leads to increased performance. We always include mRNA as that is the most powerful predictor and it does not make biological sense to exclude it. From gure 8.1 we observe that mRNA provides the best predictions if a single channel is used. Combinations of channels lead to increased performance for all models with the best performance being achieved with the mRNA, miRNA and circRNA or the mRNA, miRNA and TE counts combinations. The best overall performance is achieved by the BiRGAT network using a combination of mRNA, miRNA and TE counts.

Risk

Now we repeat the same comparison for the Risk task 8.2. Here we observe that mRNA is again the best single-channel predictor and that combinations of di erent channels lead to increased performance, particularly for the GNN based models. The best performance is achieved by the improved MOGONET model using all the available modalities

Mutation

Here we present the results for the Mutation task. We draw similar conclusions as before with mRNA being the best single-channel predictor and combining multiple omics leading to increased performance. There is even less data available for this task, just 26 samples. This leads to large variances in the estimates. The best performance is reached by MOGONET using all omics but given the sample size and the other results, this is likely to be an artifact of the data. Excluding that the second best performance is reached by MOGONET using the mRNA, circRNA and mRNA, TE counts combinations and by XGBoost on the mRNA, circRNA and mRNA, TE counts combinations and also when using only mRNA.

Figure 8.1: Model metrics on the MDS dataset for the Disease task

Figure 8.2: Model metrics on the MDS dataset for the Risk task

Figure 8.3: Model metrics on the MDS dataset for the Mutation task

8.2.2 Breast cancer subtype classification (BRCA)

Here, we present results on the breast cancer dataset. We evaluate the models on two variants of it. One uses mRMR selection to select the 200 most important features from each channel. The second uses 1000 features from each channel (except miRNA, for which there are less than 1000 features available) that are selected with variance filtering only. A similar setup with variance filtering is commonly used in other works, so we also have it here for comparison.

In the first evaluation 8.4, we can see that all models perform very well using only mRNA and not very well when using only other channels. Combining mRNA with either methylation data or copy number variations (CNV) yields a small improvement (albeit well within the standard deviation of the estimates) in performance for SVMs, XGBoost, and Bipartite GNN.

Bipartite GNN performs much worse on the methylation data than the other models. A possible reason for this is that methylation data commonly follows a bimodal distribution [89]. So, our assumption of modeling the features with either a normal or a negative binomial distribution when building the graph does not hold, explaining the poor performance.

There are a few notable differences for the 1000 feature experiment 8.5. First, methylation and copy number variations seem much more useful. Secondly, combining omic channels provides higher yields in performance than it did when using only 200 features. That holds both for the other models and for the GNN models, where this is more significant, probably due to the more powerful integration modules. However, the overall performance is still lower than in the 200-feature experiment.

Figure 8.4: BRCA results, 200 features from mRMR

Figure 8.5: BRCA results, 1000 features from variance lering

8.3 Biomarker discovery

In this section, we present the biomarkers discovered by our models, discuss their significance, and validate our findings by searching the literature and databases of known biomarkers for MDS. When searching for validation, we look for first direct links to MDS, then links to acute myeloid leukemia (AML); that is because about 30 % of MDS patients develop AML, and these diseases are genetically related and a disease continuum rather than being completely separate states [90]. If we can not find either, we can also look for links to other types of cancer.

For extracting biomarkers, we used and compared two models. XGBoost was the best non-GNN model in our experiments. For the GNN model, we chose to use MOGONET with our improvements as it is the best-performing model on the MDS datasets.

Obtaining the importance of weight with XGBoost is straightforward since, in parameter tuning, we got the best results using the linear booster, which assigns a weight to each feature. We then directly use the absolute value of each weight to measure the importance of each feature to the model.

For MOGONET, we use the more general permutation feature importance algorithm.

Let $D = \{(x_i; y_i)\}_{i=1}^{n}$ be a dataset with n samples, where x_i represents the feature vector and y_i the corresponding target value.

Let f be a trained model and L be a loss function.

1: Calculate baseline performance: $L_{base} = L(y; f(X))$

2: For each feature j in $\{1; \ldots; p\}$:

a. Generate K (where $K = 5$ in our experiments) permuted versions of feature j: $X_j^{(k)} = \text{permute}(X_j)$

b. Calculate performance drop for each permutation:

$$L_j^{(k)} = L(y; f(X_j^{(k)})) - L_{base}$$

c. Aggregate the importance scores:

$$FI_j = \sum_{k=1}^{K} L_j^{(k)}$$

To obtain a more robust estimate, we run the feature importance estimation for both models over the cross-validation folds used to train them and aggregate the feature importance across folds.

We made one more modification to the feature permutation algorithm used to obtain feature importance from the GNN. We found out that when using the algorithm and varying the random seed used to train the model, it outputs a different set of features with a relatively small overlap for each run, whilst the performance on the classification task remains similar. For this reason, we run the algorithm N times, where $N = 10$ in our experiments, and aggregate the feature importances across all these runs, hoping to suppress the noise and obtain a better estimate of the feature importances.

One possible reason for this behavior could be the low sample counts in the Myelodysplastic Syndrome dataset. Another possible (and not independent) reason could be that cancer has many molecular drivers [91], so different combinations of features can be used to obtain a good prediction performance.

8.3.1 Myelodysplastic Syndrome Disease task biomarkers

Figure 8.6 shows the importance the XGBoost model assigns to the top 50 features.

Searching for validation of the top scoring biomarkers, we found that hsa_circ_0005325 has been reported to be upregulated in AML [92], in cervical cancer [93], colorectal cancer [94]. MIR1205 has been implicated to serve a role in many types of cancer such as Hepatocellular Carcinoma [95], Glioma [96], Breast Cancer [97] and Colorectal Neoplasms [98]. PYGM encodes an enzyme involved in glycogenolysis. Enzymes of glycogen metabolism are dysregulated in a wide variety of malignancies, including cancers of the kidney, ovary, lung, bladder, liver, blood, and breast [99], PYGM is also identified by both XGBoost 8.8 and by MOGONET 8.9. CREB5 has been identified as a biomarker in multiple types of cancer including AML [100] and glioma cancer [101].

Figure 8.7 shows the feature importances recovered using MOGONET and the feature permutation algorithm. The expression of ACTA2 (actin alpha 2) is influenced by the Serum Response Factor (SRF) [102], and the activation of the SRF pathway has been shown to play a role in MDS [103]. NPIPA8 is significantly enhanced in gallbladder cancer and slightly enhanced in leukemia according to protein atlas [104, 105]. However, we found no other evidence that it plays a significant role in cancer.

CTC-260F20.3 or E7ENQ6 (newer name) produces a protein that is an accessory subunit of the mitochondrial membrane respiratory chain NADH dehydrogenase. While no direct evidence links it to myelodysplastic syndromes, some links can be inferred based on other evidence. Firstly it interacts with

Figure 8.6: MDS Disease Feature Importance's XGBoost

other subunits of NADH dehydrogenase [106, 59] which have been used as biomarkers in various other cancers such as NDUFA4 in gastrointestinal cancer [107], NDUFA 2, 4 and 10 in head and neck cancer [108], or NDUFA 2 in lung cancer [109]. A more direct link is that E7ENQ6 prevents the trans-activation of STAT3 [110, 111], STAT3 has been identi ed as a promising treatment target for both MDS and AML [112].

The next group of genes that scores fairly high are zinc ngers. In 8.7, these are ZDBF2, ZNF133, and ZNF467. Zinc ngers are transcriptional factors that regulate the expression of other genes. The expression of ZNF133 is reportedly elevated in chronic myeloid leukemia [113], and ZNF467 has been linked to MDS and leukemia [114]. An experimentally validated target of ZDBF2 is SENP6 [59, 115], SENP6 and other genes from the SENP family (SENP-1, -2, -3, -5, -6, -7) are a part of the SUMO pathway, which has been reported as a possible promising treatment target in AML [116].

Three other high-scoring genes present are TAS2R3, TAS2R4, and TAS2R5. These have been shown to play a signi cant role in AML and the TAS2Rs expression levels show signi cant associations with poor-prognosis AML groups [117].

Figure 8.7: MDS Disease Feature Importance's Mogonet

Figures 8.8 and 8.9 show a multi-omic network constructed out of the top 60 biomarkers from each omic. We use the following interactions between them: gene-gene (gene-gene from biogrid [58], protein-protein from stringdb [59], transcriptional factors via TFLink [118]), miRNA-gene from miRDB [68, 60] and circRNA-miRNA from miRBase [17, 18, 19, 20, 21, 22] to construct the resulting network. The graphs are drawn using the python networkx package [119].

We search the literature for some of the promising structured biomarkers that appear in the graph. Genes HBA1, HBA2, HBB, HBG1 and HBG2 form a small connected subgraph that appears both in the network recovered using XGBoost 8.8 and MOGONET 8.9, these genes all deal with hemoglobin, and for HBA1 and HBB it has been reported that they may function as tumor suppressors in AML patients [120]. MIR1205 has been identified as the most important miRNA by the model for this task, and we can see that it is regulated by several of the top 60 circRNAs. ZMAT3 has been identified as an important regulator in the p53 tumor suppression program [121]. The p53 protein is an important protein that protects the DNA integrity of the cell, and the p53 pathway is one of the most studied pathways in oncology [122]. Another gene related to this pathway that appears in the network and has a relatively high score assigned to it is TP53INP2 [123]. Another group of genes with presence in both graphs is the Tumor Necrosis Factor Receptor Superfamily (TNFRSF) which is intricately associated with governing critical functions including hematopoiesis, activation and migration of leukocytes, inflammation, cell death, and development [124] the genes of this family are known to play a role in MDS, being more associated with low-risk MDS [125, 126]. Genes from this family in our graphs are RELL1, TNFSF10, and TNSRSF10B. RELL1 has been identified as an oncogene in glioma cancer [127]. Furthermore, RELL1, RELL2, and RELT are genes that together form the RELT family. The proteins of this family are expressed in hematopoetic tissues and also in bone marrow and peripheral blood leukocytes (PBLs). A smaller-than-expected size of RELT was observed in PBLs, suggesting a proteolytically cleaved form of RELT [127]. These genes have been reported to play a role in different types of cancer, including blood cancer [124]. RELL1 also appears in both multi-omic networks 8.8 and 8.9.

Figure 8.8: MDS Disease Multi-Omic Network