Bocconi

Bachelor of Science in Economics Management and Computer Science

Integrating Single-Cell RNAseq and Graph Neural Networks to Predict TP53 Mutation Status

Advisor:
Prof. Francesca Buffa

Bachelor of Science thesis by:
TOMMASO RAVASIO
Student ID no. 3192281

Academic Year 2024-2025

Bocconi
Università Commerciale
Luigi Bocconi

Desidero ringraziare la Professoressa Buffa e il Professor Tangherloni per il tempo, l'attenzione e la disponibilità che hanno dedicato a questo progetto. Il loro supporto è stato fondamentale per la qualità del lavoro, e sono grato per tutto ciò che ho avuto modo di imparare sotto la loro guida.

Ringrazio l'Università Bocconi per avermi regalato i tre anni più belli e impegnativi della mia vita. Ringrazio tutte le persone che sono state al mio fianco in questo percorso. Mi auguro di avere davanti a me altrettante sfide e di avere la fortuna di poterle affrontare con persone che mi vogliono bene, proprio come in questi anni. Ringrazio Mattia, per essere stato spesso la mia guida quando mi sentivo perso e disorientato. Ringrazio Corinne, che mi ha sempre guardato con gli occhi con cui non sono mai riuscito a guardarmi. Un ringraziamento speciale va ai miei genitori, che hanno creduto in me quando nemmeno io ero in grado. Infine ringrazio l'Italia per questi bellissimi 22 anni, che sia un arrivederci e non un addio.

I would like to thank Professor Buffa and Professor Tangherloni for the time, attention, and dedication they devoted to this project. Their support was crucial to the quality of the work, and I am grateful for all that I was able to learn under their guidance.

I thank Bocconi University for giving me the best and most challenging three years of my life. I thank all the people who have been by my side in this journey. I hope to have as many challenges ahead of me and to be lucky enough to be able to face them with people who love me, just like in these years. I thank Mattia, for often being my guide when I felt lost and disoriented. I thank Corinne, who has always looked at me with the eyes with which I have never been able to look at myself. Special thanks go to my parents, who believed in me when even I was unable to. Finally, I thank Italy for these beautiful 22 years, may it be goodbye and not farewell.

Abstract

The TP53 gene is often referred to as the "guardian of the genome" due to its key role in maintaining genomic stability and preventing the propagation of damaged DNA. It is among the most commonly mutated genes in human cancers. Detecting its mutational status is crucial for cancer diagnostics. In this thesis, we explore whether the TP53 mutation status can be predicted from single-cell transcriptomic data using graph-based deep learning methods.

For our research, we used single-cell RNA sequencing data from 32 breast cancer cell lines, for a total of 35,276 single-cell observations, merged with the mutation status data of each cell line indicating whether the cell has undergone a mutation in TP53 or is a wild type. Using these data, we construct gene co-expression networks based on Spearman correlation, transforming each single-cell observation into a gene-gene graph with nodes representing genes and edges representing statistically significant co-expression between two genes. We then apply Graph Neural Network (GNNs), more specifically Graph Convolutional Network (GCNs) and Graph Attention Network (GATs), to perform graph classification and predict the TP53 mutation status of each single-cell. To better understand the impact of feature selection, throughout the full experiment we follow two parallel preprocessing strategies: one statistically driven, selecting Highly Variable Genes (HVGs), and the other biologically inspired, retaining only the genes known to be the TP53 target genes.

Before applying graph-based models, we first trained XGBoost classifiers on bulk and single-cell expression data to verify the presence of significative signal for predicting the mutation of TP53. These models achieved, respectively, F1 scores of 0.88 and 0.99, confirming the feasibility of the task. Once verified the presence of the signal, we verified whether a graph approach could lead to interesting results. We tested various GNN architectures and configurations exploring the effects of different design choices such as batch correction, and regularization techniques. On the graphs built with the HVGs, the GAT model with GraphNorm obtained the best performance, achieving an F1 score of 0.89. For the TP53 target genes graphs, the best model was the GAT combined with ComBat batch correction. A final hyperparameter tuning using Optuna identified the optimal values for different parameters such as number of hidden channels in the model,

dropout rate, learning rate, weight decay for the L2-Regularization, number of attention heads in the GAT, whether to apply loss weighting, and whether to include additional layers. After this step, the best model was a GAT applied to ComBat-corrected TP53 target genes, achieving an F1 score of 0.998. This results is marginally better than the XGBoost baseline (F1 = 0.995). While the performance gain is modest, the graph based approach is able to capture gene–gene interactions, which can improve generalization to more heterogeneous datasets and allow for more biologically interpretable insight.

Contents

1 Introduction

The TP53 gene, encoding the tumor suppressor protein p53, is mutated in more than 50% of human cancers. Loss of p53 function undermines the cell's ability to repair DNA or halt the cell cycle, making TP53 a critical biomarker in oncology. For this reason, the p53 protein is often called the "guardian of the genome", and the loss of its function due to mutation often lead to cancer.

Traditional approaches to assess the TP53 status rely on bulk sequencing data which collapse the expression of millions of cells into a single aggregate measurement. However, tumors are notoriously heterogeneous, and even within a single cell line, individual cell can differ dramatically in their transcriptomes. Single-cell RNA sequencing, also know as scRNA-seq, has emerged in the last decade, offering a higher resolution for the expression data. This allows not only to resolve the mentioned heterogeneity but also to profile tens of thousands of individual transcriptomes in a single experiment. At the same time, scRNA-seq data come with their own challenges: extreme sparsity (usually above 90%), high dimensionality and pronounced batch effects. For this reason, a more careful and rigorous preprocessing and downstream analysis is needed to treat them.

In parallel, graph based deep learning approaches, leveraging the so called Graph Neural Networks (GNNs), have obtained success in many domains by allowing to explicitly model relationships among entities. In the biology domain, GNNs are able to exploit not only each expression level per gene but also their interactions and co-expression.

Recent studies have already explored the use of Graph Neural Networks in genomics. For instance, Algabri et al., 2022 proposed scGENA, a framework to build co-expression networks from single cell data to uncover biological mechanism. Li et al., 2025 offers an in depth review of the current landscape of GNNs applied to single-cell data, highlighting the potential of graph-based approaches to improve interpretability and performances in classification tasks. However, few works have focused specifically on predicting the mutation status of TP53 using single-cell data structured as graphs. Recent efforts have explored the possibility of predicting TP53 mutation status from gene expression data using machine learning techniques. More precisely, the work by Triantafyllidis et al., 2023 show that the bulk transcriptomic data contain sufficient signal to uncover patterns

associated with TP53 mutations. While their results are promising, they are based on bulk RNA-seq. In this research, we build upon these results and ask whether such predictive signal is present at the single-cell resolution and whether a graph-based approach can be effective to detect it. To do that, we will apply Graph Neural Networks to gene co-expression graphs constructed from single-cell RNA-seq data. This allows us to model both the expression of individual genes and their relationships with each other. Our expectation is that this approach will provide more accurate and biologically interpretable predictions of the TP53 mutational status.

2 Background

Since this research lies at the intersection of biology and machine learning, we now introduce the necessary background to understand our methodology. The background is structured into two parts: the first introduces the biological foundations necessary to understand the context and significance of TP53 mutations. The second outlines the machine learning methods used, with a focus on graph-based approaches. The aim of this section is to provide the reader with the fundamental biological and computational tools to follow the experiment pipeline; not to provide a comprehensive, nor exhaustive, review of the topics involved.

2.1 Biological Background

2.1.1 Quick Introduction to Biology

The central dogma of molecular biology explain how the flow of information works in a self replicating organism. Genetic information is encoded in the DNA. The genetic information is copied into RNA molecules in a process called transcription. Then, a process called translation, convert them into proteins. Proteins are the biological entity which perform all the function of the cell; including translation, transcription and DNA replication. The real mechanism of life is much more complex but this model is good enough to give an idea of how living organism operate. DNA is a polymer, which means that it is a molecule composed of four different molecules called nucleotides (or bases). These four bases (A: adenine, G:guanine, C:cytosine, T:thymine) can be chained together in an arbitrary order along a sugar backbone, allowing the encoding of information. Usually, the chromosomes of an organism consist of two chains that are twisted around each other in the well know double-helix structure. The two strands are held together by chemical bonds between the couple of complementary bases. Specifically, the four bases falls under two categories: purines (A and G) and pyrimidines (C and T). The difference is due to their chemical composition and is beyond the scope of this introduction, for us it is enough to know that their chemical structure allows the pairing of purine A with pyrimidine T and of purine G with pyrimidine C, and these bonds hold together the double-helix. RNA is a molecule that is very similar to DNA with thymine (T) nucleotide changed to uracil (U). But the

biggest difference is in its structure, which is single stranded and can fold onto itself. This property is the key to many biological and structural functions of the RNA.

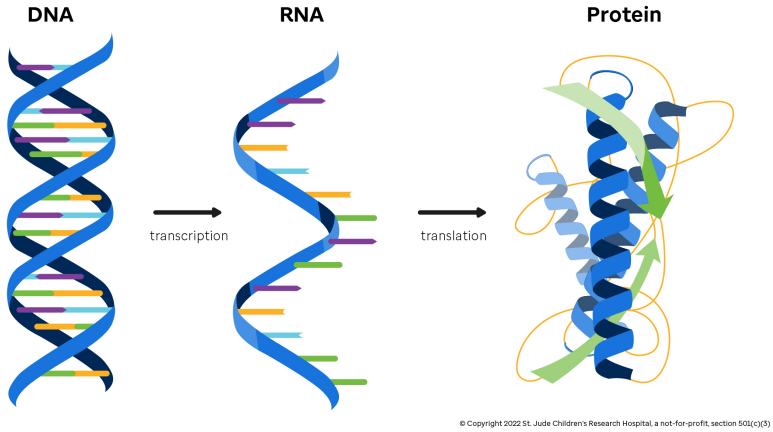


Figure 1: Illustration of the central dogma of molecular biology. Adapted from Learn Genomics (2024).

2.1.2 Transcriptomics

Transcriptomics is the study of RNA transcripts produced by the genome under specific conditions – collectively referred to as the transcriptome. It is key to highlight the difference between the transcriptome and the more well known genome. While the genome is usually more stable across different cells of the same organism, the transcriptome usually is more dynamic and reflects cellular responses. In transcriptomics, we look at the gene expression at the transcript level to get insight into the regulatory mechanism that control cellular function. Transcriptomics is now a cornerstone of biomedical research, with application ranging from cancer biology to drug discovery and personalized medicine. In this research, transcriptomic data are used as a foundational layer to build co-expression networks and predict the associated TP53 mutation status. In particular, the integration of single-cell RNA sequencing data offers higher resolution into transcriptional variability at the cellular level, enabling more precise modelling approaches.

2.1.3 Single-Cell RNA Sequencing

Modern sequencing technologies have revolutionized the way we analyse biological system. Traditional DNA sequencing techniques, such as Sanger sequencing, were limited in scalability. The introduction of the so called NGS (next generation sequencing) plat-

forms enabled massively parallel sequencing, allowing researchers to read millions of reads. Among the applications of NGS, RNA sequencing is the standard for transcriptome-wide analysis. RNA sequencing was initially applied to bulk RNA extracted from cell populations. This provided an average expression profile, but was missing to represent cell-to-cell heterogeneity. The development of single cell RNA sequencing (scRNAseq) enabled transcriptomic profiling at the resolution of individual cells. This technique capture transcriptional variability that would otherwise be undetectable in bulk measurements. In this analysis, we leverage scRNAseq data to investigate the relationship between gene expression patterns and TP53 mutational status at single cell level.

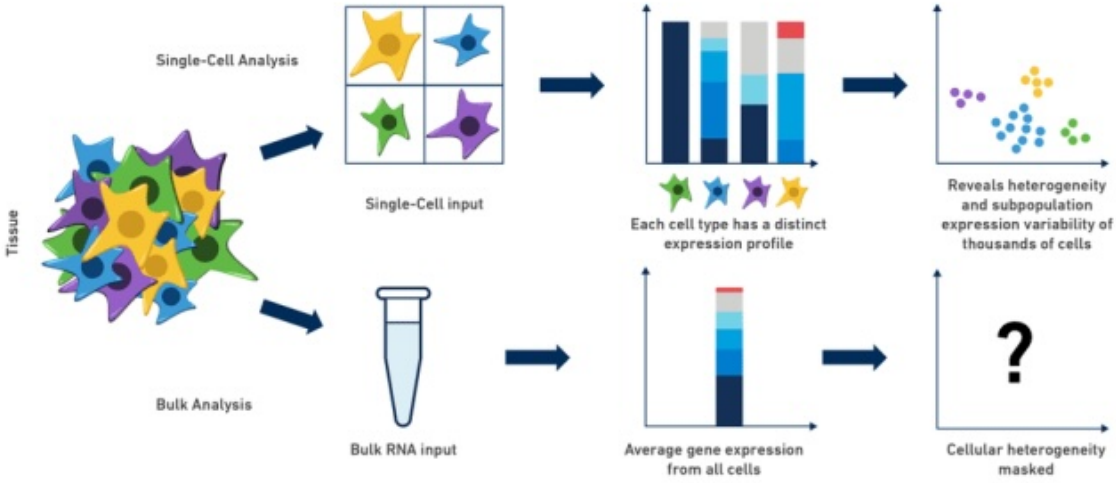


Figure 2: Single cell RNA-seq reveals cellular heterogeneity that is masked by bulk RNA-seq methods. 10x Genomics (2024)

2.1.4 The TP53 Gene And Its Role

The TP53 gene encodes the protein p53, a tumor suppressor protein that plays a central role in maintaining genomic stability. The p53 is often referred to as the "guardian of the genome" due to its activation in response to cellular stresses such as DNA damage, hypoxia or oncogene activation. Once activated, p53 induce cell cycle arrest preventing the propagation of damaged cells. Mutations in TP53 are among the most common alterations observed in human cancers, found in over 50% of human primary tumors (Chen et al., 2022). These mutations often results in loss of p53 function, allowing cells with genomic instability to avoid cell death and continue to proliferate. Given its role in tumor suppression, tp53 has been extensively studied as a biomarker and therapeutic target. In this research are investigated how transcriptomic profiles represented in gene

co-expression network derived from single cell RNA sequencing can be used to predict TP53 mutational status of individual cells.

2.1.5 Gene Co-Expression Networks

Gene co-expression network are mathematical graph representations of relationships between genes based on their expression profiles across a set of samples. In these networks, nodes represent genes, and weighted undirected edges represent the level of correlation between the expression level of the two genes. The underlying idea is that genes with similar expression patterns are likely to be related, co-regulated, or part of the same biological pathways. Constructing co-expression networks usually involves computing pairwise Spearman or Pearson correlation coefficients for each pair of gene. A threshold is then applied based on both correlation and p-values to retain only significant correlation while controlling for false positives. In the context of single-cell data, constructing such networks is usually more challenging due to increased sparsity and noise, but it is still a great structure to reveal biological signal (Algabri et al., 2022). In this thesis, co-expression graphs serve as the basis for constructing input graphs used by the graph neural networks models to predict TP53 mutation status.

2.2 Machine Learning Background

2.2.1 Graph Theory

Graph theory is a branch of discrete mathematics that studies the properties of graphs. A graph can be defined as an abstract representation of pairwise relationship between different objects. Graph theory has a wide range of applications and it often provides a useful framework for modelling interconnected systems in many fields such as computer science, physics and biology. Mathematically, a graph $G = (V, E)$ is defined as a structure composed of a set of nodes (or vertices) V and a set of edges $E \subseteq V \times V$ which defines pairwise relationships between nodes.

In the context of biological science, graphs are often used to model gene regulatory interactions or co-expression relationships. Given a gene expression matrix $X \in \mathbb{R}^{n \times d}$ where n is the number of samples and d is the number of genes, one can construct a graph either at the level of samples (cell-cell graphs and the ML problem is formulated as a node

classification problem) or at the level of features (gene-gene graphs and the ML problem is now formulated as a graph classification problem). In this work, we focus on gene-gene co-expression graphs, performing therefore a graph classification with our model. More precisely, in our graphs each node $v_i \in V$ represent a gene, each edge $e_i \in E$ represent, if present, the significant correlation between the two genes obtained by their expression profiles across the cells.

Each node $v_i \in V$ can be associated with a feature vector $x_i \in \mathbb{R}^f$, where f denotes the number of features per node. These features are allocated in a feature matrix $X \in \mathbb{R}^{|V| \times f}$, where each row corresponds to a node's features. Edges in the graph can be represented using an adjacency matrix $A \in \mathbb{R}^{d \times d}$, where A_{ij} represents the weight of the edge between gene i and j. For unweighted graphs, $A_{ij} \in \{0, 1\}$, while for weighted graphs, A_{ij} is a continuous value, usually the Pearson or Spearman correlation coefficient.

Graph based representations are particularly useful in machine learning models, especially in those problems in which standard vectorial approaches fail to fully capture relationships between features. The graphs above defined serve as a foundation for applying Graph Neural Networks, which learn from both node features and graph topology to perform tasks such as classification and link prediction.

2.2.2 Graph representation for single-cell transcriptomic data

To apply GNNs to transcriptomic data, we first have to define how graphs are constructed from raw gene expression matrices. From this data, we build gene-gene co-expression graphs. To quantify the correlation between genes, the Spearman correlation coefficient is usually used. This is done because Spearman is more robust to non-normality and outliers than Pearson correlation. Briefly, it is computed by ranking the expression values of each gene across cells and then applying Pearson to these ranks. A threshold is then applied to both the correlation coefficients and the corresponding p-values to retain only statistically significant edges. For the p-value threshold, it is common to apply the Bonferroni correction to control the family-wise error rate by dividing the desired significance level α by the number of tests performed. In our case becomes

$$p = \frac{\alpha}{\binom{n}{2}}$$

where p is the adjusted significance threshold obtained for the p-values, α is the significance level and $\binom{n}{2}$ corresponds to the number of pairwise tests performed with n genes.

One challenge in graph construction is the presence of batch effect. Batch effect can be defined as a source of non-biological variation introduced when different samples in the dataset are processed in different experimental groups, this lead to systematic differences in the data not related to biological variables. If not corrected, this can distort correlation patterns and introduce biases in the graphs. Many algorithms can be applied to correct for this behavior. Two of the most widely used in transcriptomics are ComBat (Johnson et al., 2006) and Harmony (Korsunsky et al., 2019) with their implementation in Scanpy, both known for their robustness and computational efficiency.

The final output is a weighted adjacency matrix $A \in \mathbb{R}^{d \times d}$, where each entry A_{ij} reflects the strength of association between gene i and gene j. This matrix defines the edge structure for each single-cell graph used as input to the GNN. Together with the node feature matrix $X_i \in \mathbb{R}^{d \times f}$, where X_i corresponds to the expression values of the i-th cell, this defines the full input graphs $G_i = (X_i, A)$. Each graph G_i can be seen as an individual observation in the graph classification task. Note that this construction strategy is just one of the many, different approaches may apply different strategies. For a more complete review of GNN approaches for single-cell omics see Li et al. (2025).

2.2.3 Graph structure modeling in biology

Modeling single-cell transcriptimic data as graphs present challenges that are less common in other domains. Let's consider a graph $G_i = (X_i, A)$ associated with a single cell i, where $X_i \in \mathbb{R}^{d \times f}$ is the feature (or gene expression) matrix. and $A \in \mathbb{R}^{d \times d}$ is the adjacency matrix.

A first challenge is handling the sparsity of scRNA-seq data. Sparsity refers to the number of zero entries in the gene expression matrix X, which arises because most genes are not expressed in most single cells. Sparsity levels exceeding 90% are common, while such values would be considered high in other contexts, they are expected here. In practice, high sparsity can lead to unreliable edge weights in the adjacency matrix and disconnected components in the graphs, resulting in a worse model performance.

Moreover, the high dimensionality of gene expression profiles can be a problem. Feature

selection is often performed to retain only a subset of genes. However, defining the optimal subset of features is a non-trivial optimization problem. Feature selection strategies can be either algorithmic or based on prior biological knowledge. An example of an algorithmic approach is to identify the Highly Variable Genes (HVGs), defined as the genes with the greatest variance across cells. Alternatively, one may leverage known biological pathways, for example by selecting only the TP53 target genes identified in Fischer (2017).

2.2.4 Introduction to GNNs

Graph Neural Networks (GNNs) are a family of deep learning models designed to operate directly on graph-structured data. Unlike traditional Neural Networks which assume input data to lie in the Euclidean space (for example 2D grids like images or 1D sequences like texts), GNNs apply neural networks operations to the graphs domain, incorporating both node-level features and the underlying topology of the graph into the learning process.

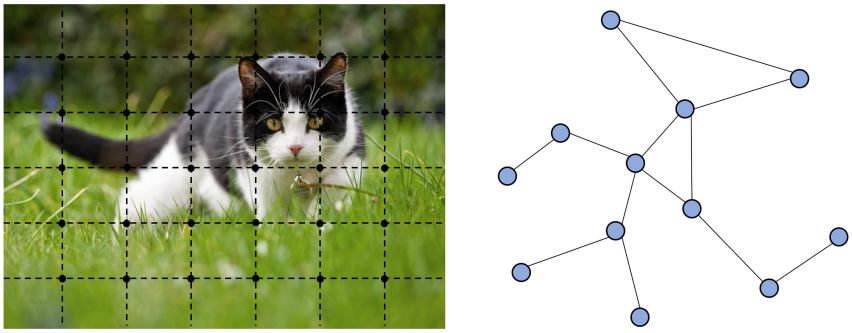


Figure 3: Left: image in Euclidean space. Right: graph in non-Euclidean space. Zhou et al., 2020

To give the reader a high level glimpse of how GNNs operates, we can describe them as iterative message passing architectures: at each layer, every node updates its representation by aggregating information from its neighbors. The generic update rule in a GNN is the following:

$$h_i^{(l)} = f^{(l)}(h_i^{(l-1)}, h_j^{(l-1)} : j \in \mathcal{N}(i))$$

where $h_i^{(l)}$ denotes the representation of node i at layer l, and $N(i)$ indicates the set of neighbors of node i. The function $f^{(l)}$ aggregates the features of the neighbors and combines them with the current node representation.

2.2.5 GCNs and GATs

Different type of GNNs architecture implements different $f^{(l)}$. Graph Convolutional Networks (GCNs) generalize Convolutional Neural Networks (CNNs) to work in the graph domain. GCNs apply convolutional operations to nodes and their neighbors, aggregating features from neighboring nodes to update a node's representation. The forward propagation rule in a GCN is:

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)})$$

Here, $\tilde{A} = A + I$ denotes the adjacency matrix of the graph after adding self loops through the addition of the Identity matrix I. This ensures that each nodes includes its own features during message passing. $\tilde{D}$ is the degree matrix, in which each diagonal entry $\tilde{D}_{ii}$ indicates the degree of the node i (that is the sum of edge weights connected to node i). $H^{(l)}$ is the matrix containing the features embedding of all nodes at layer l, where each row corresponds to a node. $W^{(l)}$ is the weight matrix that transforms the input features, and σ is the non linear activation function applied at the end of the aggregation step.

Another popular variant is the Graph Attention Network (GATs), which leverages the attention mechanism to improve how GNNs aggregate information from neighbors. Instead of treating all neighbors equally, GATs assign different attention weights to each neighbor based on its importance. This allows GATs to focus on the most relevant parts of the graphs. In a GAT layer, the propagation rule is

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l)} W^{(l)} h_j^{(l)}\right)$$

where $\alpha_{ij}^{(l)}$ is the attention coefficient between node i and node j, computed as:

$$\alpha_{ij}^{(l)} = \text{softmax}_j\left(\text{LeakyReLU}\left(a^\top\left[W^{(l)}h_i^{(l)} \,\|\, W^{(l)}h_j^{(l)}\right]\right)\right)$$

Here, a is the learnable weight vector, $\|$ denotes vector concatenation and $W^{(l)}$ is a shared linear transformation.

By stacking multiple GNN layers, each node can incorporate information from increasingly distant neighbors; improving as a consequence context-awareness of the model. This

information can be used for tasks such as node classification or, as in our case, graph classification. (Zhou et al., 2020)

2.2.6 Typical GNNs architecture

A common GNNs architecture pipeline begins with a series of GNN layers responsible for message passing and local aggregation. To improve generalization and reduce overfitting, dropout and normalization layers are inserted between message passing layers. Dropout layers simply deactivates a fraction of node features during training to prevent the model from over-relying on specific input patterns. Normalization layers can have different nature. Two popular choices are BatchNorm and GraphNorm. BatchNorm normalizes node embeddings across a mini-batch of graphs, while GraphNorm normalizes at the individual graph level, making it more suitable for graph-level tasks. Batch normalization operates by standardazing features across all nodes in a batch. For a feature dimension k, it computes

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)2} + \epsilon}}$$

where $\mu_B^{(k)}$ and $\sigma_B^{(k)2}$ are the mean and variance computed across the entire batch for feature k. Graph Normalization, instead, computes statistics at the individual graphs level. Given a graph $G = (V, E)$ and node feature $x_v^{(k)}$, it computes:

$$\hat{x}_v^{(k)} = \frac{x_v^{(k)} - \mu_G^{(k)}}{\sqrt{\sigma_G^{(k)2} + \epsilon}}$$

After each normalization step, we can find a non-linear activation function, usually ReLU, followed by dropout. Convolution, normalization, activation and dropout together forms the typical GNN layer which is repeated multiple times in the full architecture. At the end of the last message passing layer, a pooling layer is applied. A common method is to use global mean pooling, which computes the average of node embeddings across the entire graph. This graph representation is then passed to one or more fully connected layers to perform the final classification task on the final embeddings. These embeddings integrate both the expression profile of each gene and its topological role in the co-expression graph.

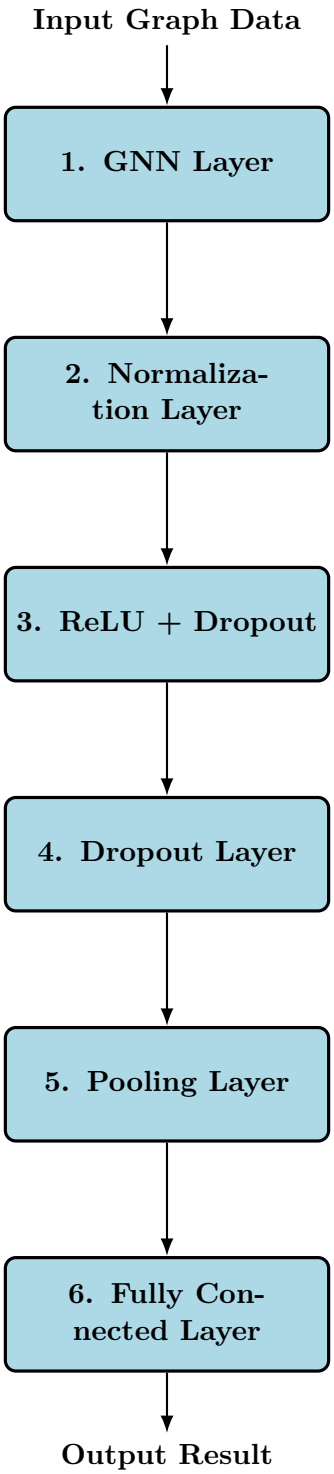


Figure 4: Schematic example of a minimal Graph Neural Network (GNN) architecture showing the core layers of a basic graph-based model.

3 Experiment

3.1 Data

Before starting to talk about the experiment, it is important to spend some time on the data used. To build our final CSV file used in the analysis, we had to merge the information coming from two different datasets: one for the expression data and one for the mutation data.

The first is the scRNA-seq publicly available dataset *Single Cell Breast Cancer Cell-line Atlas* (Gambardella, 2022), published on Figshare and available at https://doi.org/10.6084/m9.figshare.15022698.v2. It consists of raw UMI counts of 35,276 single cells derived from 32 human breast cancer cell lines, generated using the 10x Genomics Chromium platform. Each matrix provides gene-level expression counts per individual cell, enabling high-resolution transcriptional profiling. The data are unfiltered and unnormalized, preserving the original unique molecular identifier (UMI) counts for each gene-cell pair. This format is optimal to conduct preprocessing, quality control, and analytical workflows using pipelines such as Scanpy. We will go deeper into this in the next pages. The mutation data are obtained from the publicly available dataset *The TP53 Database* (R21, Jan 2025): https://tp53.cancer.gov. For a full description of the database structure and transition, see de Andrade et al. (2022). This dataset provides comprehensive information on the TP53 mutation status across a wide array of human cancer cell lines. Each entry includes detailed annotations such as cell line identifiers and genetic alterations, including TP53 mutation specifics. We then merged the information from the two datasets based on the cell line IDs. More precisely, for almost the totality of the single cells (to be more precise, 90.24%) in the expression data, we were able to find the respective cell lines in the mutation data and obtain the information on whether that specific single cell was mutated or not. At this stage, our dataset consisted of 31,833 single-cell observations and 47,096 genes. The bulk RNA-seq expression profiles are obtained from the Cancer Cell Line Encyclopedia (CCLE), specifically the DepMap 22q2 release (DepMap, 2022). The list of TP53 target genes used for biologically driven feature selection was obtained from Fisher's curated list of p53 targets (Fischer, 2017), which can be found at the following link https://tp53.cancer.gov/target_genes.

3.2 Preliminary Checks

Before conducting an in-depth analysis in a graph-based approach, we performed some preliminary experiments to verify whether gene expression data contains detectable signal predictive of TP53 mutation status.

The first experiment consisted in training a basic XGBoost classifier on the bulk RNA-seq gene expression data, using TP53 mutation status as the target variable. The model achieves an accuracy of 0.85 and a F1-score of 0.88. With these results we can conclude that the bulk data contains a strong predictive singal for TP53 mutation status. These findings align with the one presented in Triantafyllidis et al. (2023).

We then checked if this signal is also present at a single-resolution. We trained another XGBoost classifier, this time on the single-cell expression data, achieving an accuracy of 0.99 and an F1-score of 0.99. These results confirms the preservation of the mutational signal in the single-cell data.

These findings support the hypothesis that expression profiles carry relevant information about TP53 mutation status, both at bulk and single-cell resolution. This motivates the use of more advanced models which can incorporate both individual gene expression and the relational structure among these genes.

In the next pages, we will evaluate the performance of a model using mainly the F1-score, more suitable for slighly imbalanced datasets like ours. This metric score is computed as follows:

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

where precision and recall are computed as follows:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

with TP being the number of True Positives, FP the number of False Positives and FN the number of False Negatives.

3.3 Preprocessing and EDA

The first step in the analysis was to perform some exploratory data analysis to get to know the data better and to prepare them to be fed to the model. The majority of the following operations are performed leveraging the Python library scanpy presented in Wolf et al. (2018).

3.3.1 Gene Symbols mapping

The 10x Genomics expression matrix retrieved from the data used Ensembl ID to identify the genes; to ensure interpretability and consistency with external annotations, a gene symbol mapping step was required. To achieve this, we query the MyGene database using the mygene python library (Xin et al., 2016). Although most of the Ensembl IDs were successfully converted, 928 genes had no corresponding symbol and 22 appeared more than once due to duplicates or annotation inconsistencies. This step enabled a clearer biological interpretation of downstream analysis.

3.3.2 Single-Cell Mutation Mapping

To associate each single cell with its corresponding TP53 mutation status, we performed a matching operation between the expression dataset and the mutation one. To do that, we first ensured the consistency between the two datasets by cleaning and standardizing cell line identifiers. The mutation status was then added to the expression data. Cells with missing matching information were excluded from the dataset. This led to removing 4,562 cells, leaving 30,714 single cells which corresponds to the 87% of our original dataset. We report here the cell lines found in the expression data with a missing match in the mutation data: 'HS578T' 'MCF12A' 'MX1' 'MDAMB453'.

3.3.3 Sparsity

The gene expression matrix presented a high level of sparsity. The number of non-zero entries was only 100,804,217 out of 1,446,506,544. The corresponding sparsity is 93.03%. As said before, although sparse, this distribution is expected for scRNAseq data and does not indicate technical errors. We took into account this level of sparsity when designing our downstream analyses.

3.3.4 Quality Control Plots

To assess the initial quality of the single-cell dataset, we created some visualizations plotting some key metrics across all cells using violin plots (Figure 5) and scatter plots (Figure 6). In particular, we focused on two quality control metrics: the number of genes detected per cell and the total number of transcript counts, that is the total UMI count per cell. The violin plots reveal the distribution and helps in the visual detection of outliers. Cell with extremely low or extremely high total counts may indicate respectively damaged cells or doublets. To sum up we plotted a scatter plot of the two dimensions versus each other, that is "n_genes_by_count" and "total_counts" to support the detection of cells with abnormal transcription profiles. These plot were used as a diagnostic tool to confirm overall quality of the data before normalization and feature selection. As a further step of quality assessment we applied the Scrublet algorithm (Wolock et al., 2019) directly on the raw expression data. In our dataset, no doublets were detected suggesting a clear single cell capture. As a consequence no filtering based on the doublet prediction was applied.

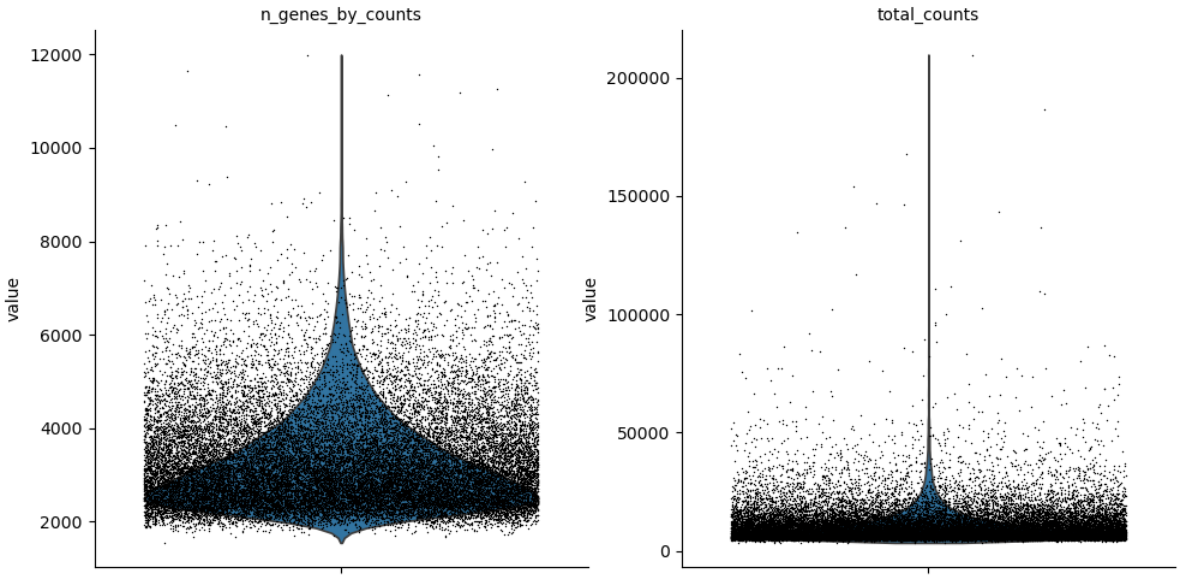


Figure 5: Left: Violin plots of the total number of genes detected per cell Right: total number of transcript counts

3.3.5 Normalization

Before downstream analysis, the raw count matrix was normalized to account for variations in sequencing depth between samples. First a Counts Per Million (CPM) normal-

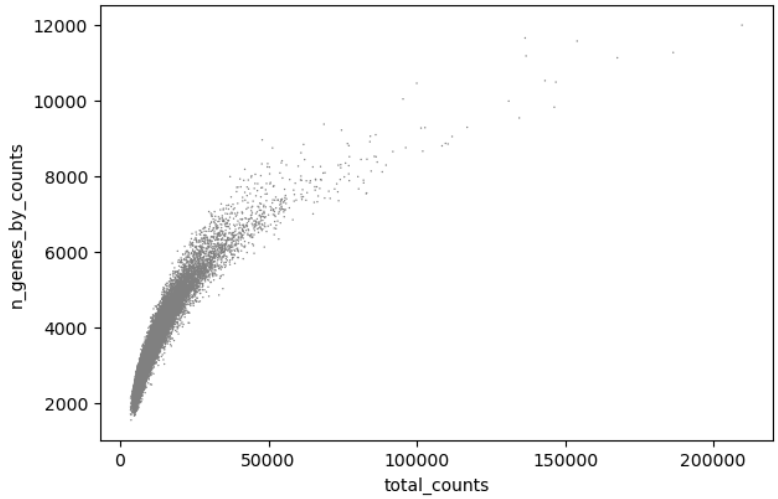


Figure 6: Scatter Plot of total number of genes detected per cell vs total number of transcript counts

ization was applied. The CPM normalization scales the raw read counts for a gene by the total number of reads in a sample, multiplied by a million. Subsequently, a logarithmic transformation was applied to stabilize variance and approximate a normal distribution. This normalization step is key to reduce the impact of highly expressed genes and to allow for more robust dimensionality reduction.

3.3.6 Highly Variable Genes

Feature selection was performed to retain only the most informative genes. We identified the highly variable genes (HVGs) across the data features. This step reduces the noise introduced by non-informative genes, which can obscure biological signal. Gene were filtered based on their mean expression and dispersion, applying a threshold of a minimum mean of 0.0125, a maximal mean of 3 and a minimal dispersion of 0.5. This procedure allowed us to reduce our number of considered genes from 47,096 to 2,071. The selected HVG were then visualized in a mean dispersion plot (Figure 7), showing the HGVs with respect to the others.

3.3.7 Principal Component Analysis

We then applied Principal Component Analysis (PCA) to the subset of HVGs we previously identified. This is done to reduce the dataset dimensionality while preserving its most significant sources of variation. To investigate how many principal components were needed, some visualizations were plotted. The first graph illustrates how much variance

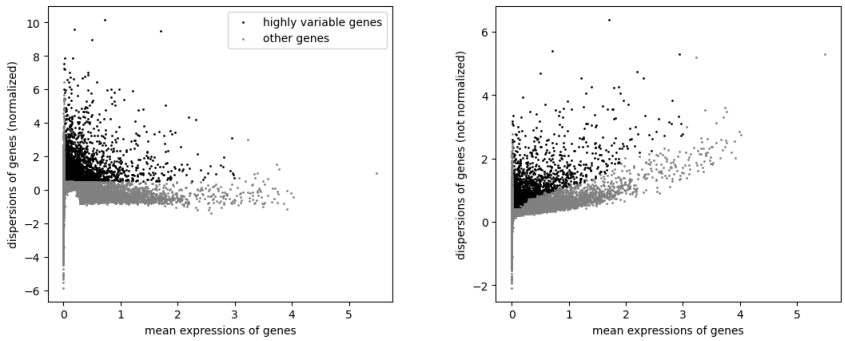


Figure 7: Mean Variance Plot for HVGs

is explained by each principal component (ranked in decreasing order of the variance explained). A good rule of thumb in these graphs is to identify an elbow point, that is when the slope of the variance explained changes significantly. In Figure 8 there is a significant change of slope around the 50th component. However we can see from our Cumulative Explained Variance plot (Figure 9) that this would explain less than the 50% of the variance of our data which is too little. This pushed us to increase the number of principal component to 700, reaching a total explained variance of 80%.

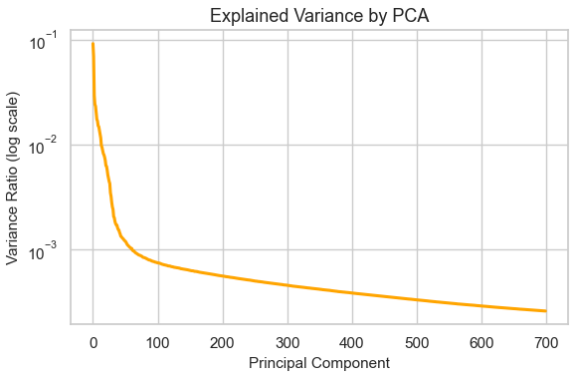


Figure 8: PCA variance ratio plot

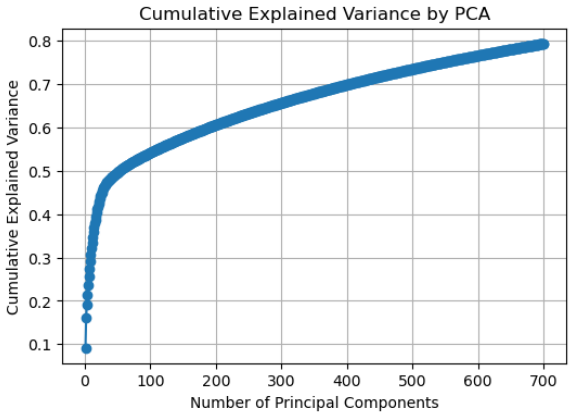


Figure 9: PCA cumulative explained variance

3.3.8 Uniform Manifold Approximation and Projection

To visualize the global structure of the dataset, we computed a neighbourhood graph on the PCA-transformed space and applied a Uniform Manifold Approximation and Projection (UMAP). The graph you can see below (Figure 10) are colored using two different criteria. The first one assign different colors to different cell lines. The second graphs assign different colors to different mutation status. In blue are the mutated cell lines,

and in orange the non-mutated or wild type. From the left plot the clusters are separated and correspond to individual cell lines, indicating a strong batch effect due to cell line. In our further analysis, we will be aware of this as it may bias the model to learn cell line differences rather than mutation status. In the right plot, we observe that mutated and wild-type cells are well-separated within individual cluster. This suggest that TP53 mutation has a detectable impact on gene expression, although it does not override the dominant effect of cell line identity. We will take this into account in our modeling strategy.

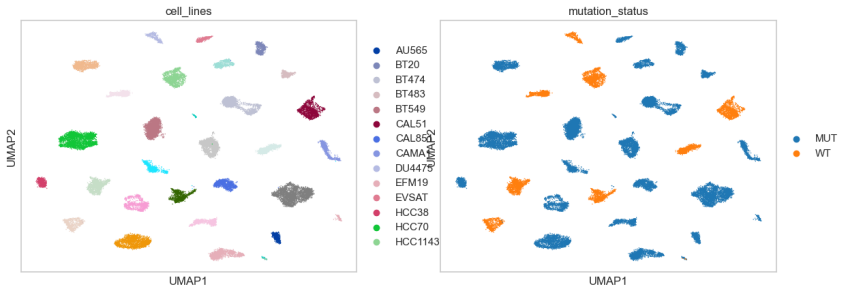


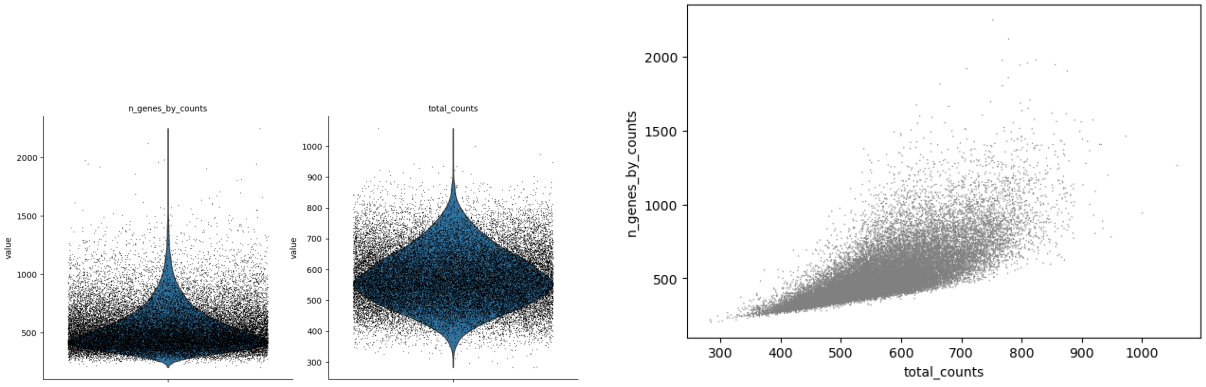
Figure 10: Left: UMAP colored by cell lines Right: UMAP colored by cell status

3.3.9 Final check on Quality Control Plots

As a final step, we compared the quality control plots before and after the filtering and transformation steps. By looking at the violin plots in Figure 11a we observe significantly more symmetric distributions. Similarly, the scatter plot in Figure 11b appeared more compact, indicating an improvement in the uniformity of the dataset. This confirms the effectiveness of our preprocessing pipeline to contribute to a cleaner and more biologically significant representation of our data. Consequently, for our subsequent analysis we are going to use the reduced and transformed data produced by this pipeline.

3.4 Analysis of TP53 Target Genes

In addition to our preprocessing pipeline based on Higly Variable Genes, we investigated whether a biologically driven feature selection could be a useful alternative. To be more precise, we explored the use of TP53 target genes derived from the list published by Fischer (2017). From the expression data, we derived a pairwise correlation matrix using only this set of target genes and compared that to the one built using the HVGs set, both at single-cell and bulk resolution. We analyzed the sparsity and plotted the histogram of



(a) Violin plots of genes per cell and total counts after preprocessing.

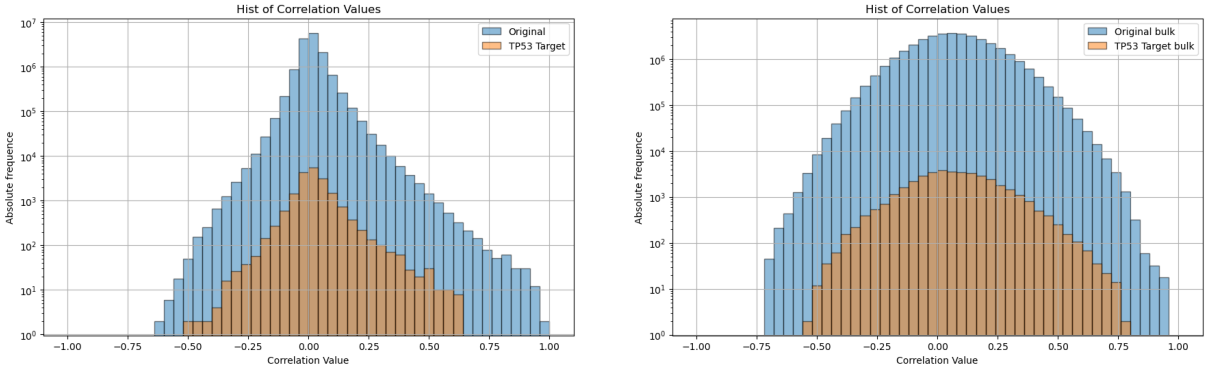
(b) Scatter plot: genes per cell vs. total counts after preprocessing.

Figure 11: Quality-control metrics after preprocessing of single-cell data.

the correlation values in the two matrices to compare the distributions (Figure 12).

Our initial hypothesis was that the TP53 target genes would exhibit a different correlation profile, with a distribution shifted to the right due to stronger positive co-expression among genes regulated by common transcription factor. However, as shown by the histograms, the two distribution of correlation values does not substantially differ. A Komolgorov-Smirnov test confirms that the two distributions are significantly different, yet the difference is much less than expected.

Despite this smaller-than-expected shift, we will further investigate both feature selection strategies as two parallel pipelines throughout the entire model construction and comparison phases. This will allow us to assess the impacts of each feature selection method on the model performance.



(a) Single-cell data

(b) Bulk data

Figure 12: Distribution of gene–gene correlation values for HVG (blue) and TP53 targets (orange), computed on single-cell and bulk data.

3.5 Network Construction

In this section we present the pipeline to transform the preprocessed single-cell expression data into input graphs for the GNN.

The starting point is therefore the tabular structured expression matrix, consisting of 30,714 cells per 2,071 genes for the HVG data and 30,714 cells per only 139 genes for the TP53 target data. Each row represents the expression profile of a single cell across all selected genes.

3.5.1 Train and Test split

As a first step, the expression matrix was randomly split into training and test subsets, using the common rule of 80% of the cells assigned to the train set and 20% to the test.

3.5.2 Correlation matrix

After the split, we computed a gene-gene correlation matrix using the training set only. By doing this, we prevent any information from the test set from leaking into the input data. This is important to avoid data leakage problem. The matrix is built such that it captures pairwise gene relationships by calculating the Spearman correlation between each pair of genes across all training cells. To retain only meaningful connection and filter out noise, we applied a correlation threshold of 0.2 and a p-value threshold of 0.05 Bonferroni corrected. Values below this threshold were set to 0.

3.5.3 Graph Construction

Each individual cell was represented as a graph $G_i = (X_i, A)$ where $X_i \in \mathbb{R}^{d\times 1}$ is the node feature matrix containing the expression value of the d genes for cell i, and $A \in \mathbb{R}^{d\times d}$ is the adjacency matrix. It is important to note that the adjacency matrix is pre-computed and shared across all observations, therefore all graphs share the same topology but differ in the node features based on their expression level.

Graphs were stored in PyTorch Geometric objects and written to disk in batches of 500 graphs to ensure scalability (Fey & Lenssen, 2019). The final train and test sets contain 24,571 and 6,143 graphs, respectively. Each graph has 2,071 nodes and 157,158 edges when using HVGs and 138 nodes and 846 when using TP53 target genes.

3.6 Model Architectures

Two distinct Graph Neural Network architectures were implemented: A Graph Convolutional Network (GCN) and a Graph Attention Network (GAT). Each model received the same input graphs described in the previous sections.

The general architecture of both GCN and GAT consists of:

- Two graph convolutional or attention layers depending on the model type
- A normalization layer after each message-passing step (either BatchNorm or GraphNorm)
- ReLU activation and dropout to improve generalization
- A global mean pooling layer to compress node-level information into a single graph-level embedding
- A final fully connected (linear) layer for binary classification

Each of these component has a specific role. The message passing layers take care of updating the node embeddings based on the neighbors information. Normalization and dropout are useful in preventing overfitting. Finally, the pooling layer aggregates the graph representation into a vector, which will then be passed to the fully connected layer to perform classification. Both models were implemented in Pytorch Geometric in a modular design to allow for comparison of different configurations. Additional information about the different architectural and training modifications explored will be discussed in the next section.

3.7 Model Comparison

To evaluate the performance of the different models, for each GNN architecture, we started with a baseline two-layer implementation, followed by various modifications to explore how different design choices impact classification performances. These variations included:

- **L2 Regularization (AdamW)**: applied weight decay regularization using the AdamW optimizer to mitigate overfitting.
- **Batch correction with ComBat**: used data preprocessed with the scanpy imple-

mentation of ComBat for batch effect correction (Johnson et al., 2006).

- **Batch correction with Harmony**: used data preprocessed with the scanpy implementation of Harmony for batch effect correction (Korsunsky et al., 2019).
- **GraphNorm instead of BatchNorm**: replaced Batch Normalization layers with GraphNorm, a normalization method tailored for graph data (Cai et al., 2020).
- **Weighted cross-entropy loss**: introduced class weights in the loss function to address class imbalance in the mutation labels.

To compare the performances of the various model configurations, we summarize them in the table below. Individual performances plots for each model configuration can be found in the Appendix A (4).

Using Highly Variable Genes			**Using TP53 Target Genes**		
Model Variant	**Accuracy**	**F1-score**	**Model Variant**	**Accuracy**	**F1-score**
GCN Baseline	0.76	0.86	GCN Baseline	0.76	0.86
GCN + L2 Reg.	0.76	0.86	GCN + L2 Reg.	0.76	0.86
GCN + ComBat	0.67	0.79	GCN + ComBat	0.57	0.65
GCN + Harmony	0.76	0.86	GCN + Harmony	0.67	0.78
GCN + GraphNorm	0.77	0.87	GCN + GraphNorm	0.76	0.86
GCN + Weighted Loss	0.76	0.86	GCN + Weighted Loss	0.74	0.85
GAT Baseline	0.79	0.87	GAT Baseline	0.76	0.86
GAT + L2 Reg.	0.79	0.87	GAT + L2 Reg.	0.76	0.86
GAT + ComBat	0.76	0.86	GAT + ComBat	0.86	0.91
GAT + Harmony	0.78	0.87	GAT + Harmony	0.80	0.88
GAT + GraphNorm	0.81	0.89	GAT + GraphNorm	0.77	0.86
GAT + Weighted Loss	0.74	0.81	GAT + Weighted Loss	0.61	0.69

Table 1: Side-by-side comparison of model performances using HVG-based and TP53 target gene-based feature selection.

Based on the results presented in the table and on a visual inspection of the different performance plot (see Appendix 4), we observed that the Graph Attention Networks consistently outperformed the Graph Convolutional Networks. In particular, for the model

with the HVGs, the GAT model implementing GraphNorm was the model that showed the most promising results and, for this reason, it was selected as the basis configuration for additional hyperparameter tuning. For the model leveraging the TP53 target genes, both the GraphNorm and the combat-batch-corrected implementation showed promising results, therefore we explored a combination of the two methods. However, this model yielded an F1 score of 0.88, lower than the one obtained with the simple combat-batch-corrected model, we decided therefore to proceed with the latter version. Summing up, for the hyperparameter tuning step we will use the GAT + GraphNorm model for the HVGs genes and the GAT + Combat for the TP53 target genes.

3.8 Hyperparameter Tuning

We now proceed to the hyperparameter tuning phase to optimize the hyperparameters of the best-performing model. The tuning process is conducted leveraging the Optuna optimization framework and its Python implementation (Akiba et al., 2019). Our aim is to maximize the F1 score in the test set. The following hyperparameters were included in the optimization: number of hidden channels in the model, dropout rate, learning rate, weight decay for the L2-Regularization, number of attention heads in the GAT, whether to apply loss weighting and whether to include a third GAT layer.

During the search, two separate hyperparameter tuning procedures were performed: one using training graphs built from Highly Variable Genes (HVGs) and one using graphs constructed from target genes. The optimization was run for 20 trials and each trial was run for 50 epochs. The final selected configuration was the one that maximized the F1 score. The results of the tuning phase are presented in the table below.

Model Configuration	**Accuracy**	**F1-score**
GAT + GraphNorm (HVG)	0.918	0.948
GAT + ComBat (TP53 targets)	0.997	0.998

Table 2: Performance comparison between models using HVGs and TP53 target genes.

4 Conclusion

In this work, we explored the use of Graph Neural Networks (GNNs) for predicting TP53 mutations from scRNA-seq data. We compard two strategies for graph construction: one based on the Highly Variable Genes (HVGs), and another one limited to the TP53 target genes. Although we used much fewer genes, 139 compared to 2,071 , the model based on the TP53 target genes achieved better performances, confirming that these genes carry strong predictive signal for the mutation of TP53 and may offer better interpretability of the results.

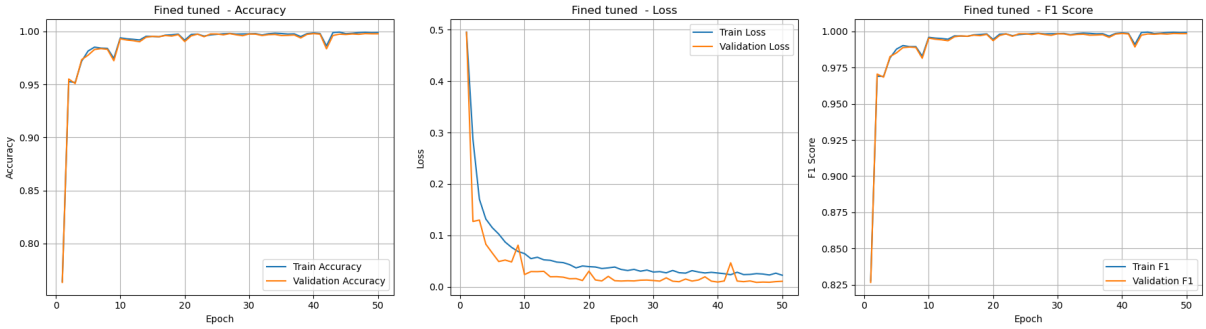


Figure 13: Training and validation curves (accuracy, loss, and F1) for the fine-tuned GAT model using TP53 target genes.

Compared to the initial XGBoost baseline trained directly on expression data, our final GNN model reaches comparable and slightly better results, with a final F1 score of 0.998 compared to the XGBoost's F1 of 0.995. While the performance improve is limited, the graph approach introduces structural advantages, such as gene-gene interactions, that may results in a better generalization on more heterogeneous datasets and offer deeper biological insight.

References

10x Genomics. (2024). Single cell rna-seq: An introductory overview and tools for getting started [Accessed: 2025-06-17].

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Algabri, Y. A., Li, L., & Liu, Z.-P. (2022). Scgena: A single-cell gene coexpression network analysis framework for clustering cell types and revealing biological mechanisms. *Bioengineering*, *9*(8), 353. https://doi.org/10.3390/bioengineering9080353

Cai, T., Luo, S., Xu, K., He, D., Liu, T.-Y., & Wang, L. (2020). Graphnorm: A principled approach to accelerating graph neural network training. https://doi.org/10.48550/ARXIV.2009.03294

Chen, X., Zhang, T., Su, W., Dou, Z., Zhao, D., Jin, X., Lei, H., Wang, J., Xie, X., Cheng, B., Li, Q., Zhang, H., & Di, C. (2022). Mutant p53 in cancer: From molecular mechanism to therapeutic modulation. *Cell Death and amp; Disease*, *13*(11). https://doi.org/10.1038/s41419-022-05408-1

de Andrade, K., Lee, E., Tookmanian, E., & et al. (2022). The tp53 database: Transition from the international agency for research on cancer to the us national cancer institute. *Cell Death and Differentiation*, *29*, 1071–1073. https://doi.org/10.1038/s41418-022-00976-3

DepMap, B. (2022). Depmap 22q2 public. https://doi.org/10.6084/M9.FIGSHARE.19700056.V2

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric. https://doi.org/10.48550/ARXIV.1903.02428

Fischer, M. (2017). Census and evaluation of p53 target genes. *Oncogene*, *36*(28), 3943–3956. https://doi.org/10.1038/onc.2016.502

Gambardella, G. (2022, January). Single Cell Breast Cancer cell-line Atlas. https://doi.org/10.6084/m9.figshare.15022698.v2

Johnson, W. E., Li, C., & Rabinovic, A. (2006). Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, *8*(1), 118–127. https://doi.org/10.1093/biostatistics/kxj037

Korsunsky, I., Millard, N., Fan, J., Slowikowski, K., Zhang, F., Wei, K., Baglaenko, Y., Brenner, M., Loh, P.-r., & Raychaudhuri, S. (2019). Fast, sensitive and accurate integration of single-cell data with harmony. *Nature Methods*, *16*(12), 1289–1296. https://doi.org/10.1038/s41592-019-0619-0

Learn Genomics. (2024). *The central dogma* [Accessed: 2025-06-17]. https://learngenomics.dev/docs/biological-foundations/the-central-dogma/

Li, S., Hua, H., & Chen, S. (2025). Graph neural networks for single-cell omics data: A review of approaches and applications. *Briefings in Bioinformatics*, *26*(2). https://doi.org/10.1093/bib/bbaf109

Triantafyllidis, C. P., Barberis, A., Hartley, F., Cuervo, A. M., Gjerga, E., Charlton, P., van Bijsterveldt, L., Rodriguez, J. S., & Buffa, F. M. (2023). A machine learning and directed network optimization approach to uncover tp53 regulatory patterns. *iScience*, *26*(12), 108291. https://doi.org/10.1016/j.isci.2023.108291

Wolf, F. A., Angerer, P., & Theis, F. J. (2018). Scanpy: Large-scale single-cell gene expression data analysis. *Genome Biology*, *19*(1). https://doi.org/10.1186/s13059-017-1382-0

Wolock, S. L., Lopez, R., & Klein, A. M. (2019). Scrublet: Computational identification of cell doublets in single-cell transcriptomic data. *Cell Systems*, *8*(4), 281–291.e9. https://doi.org/10.1016/j.cels.2018.11.005

Xin, J., Mark, A., Afrasiabi, C., Tsueng, G., Juchler, M., Gopal, N., Stupp, G. S., Putman, T. E., Ainscough, B. J., Griffith, O. L., Torkamani, A., Whetzel, P. L., Mungall, C. J., Mooney, S. D., Su, A. I., & Wu, C. (2016). High-performance web services for querying gene and variant annotation. *Genome Biology*, *17*(1). https://doi.org/10.1186/s13059-016-0953-9

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, *1*, 57–81. https://doi.org/10.1016/j.aiopen.2021.01.001

Appendix A - Complete Model Results

We leave here the full training and validation curves for all tested GCN and GAT architecture. Each plot shows the evolution of accuracy, loss and F1 score on both training and validation sets over 50 epochs.

Highly Variable Genes (HVG)

Graph Convolutional Networks (GCN)

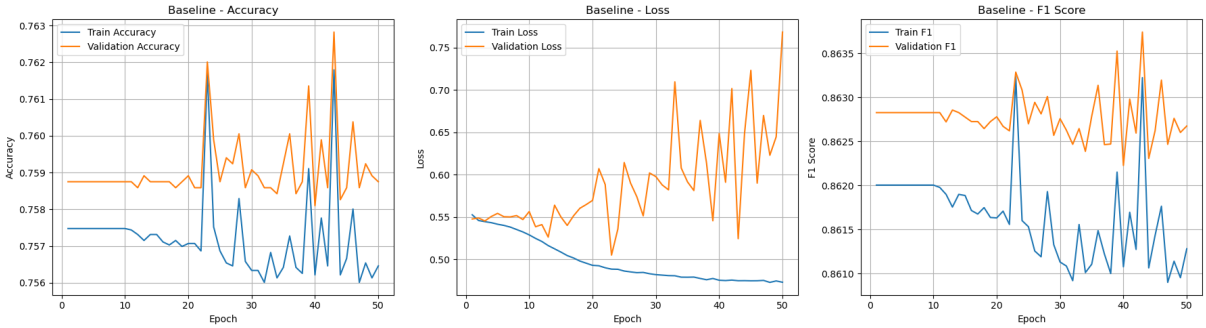


Figure 14: Training and validation curves for the GCN baseline on HVG data.

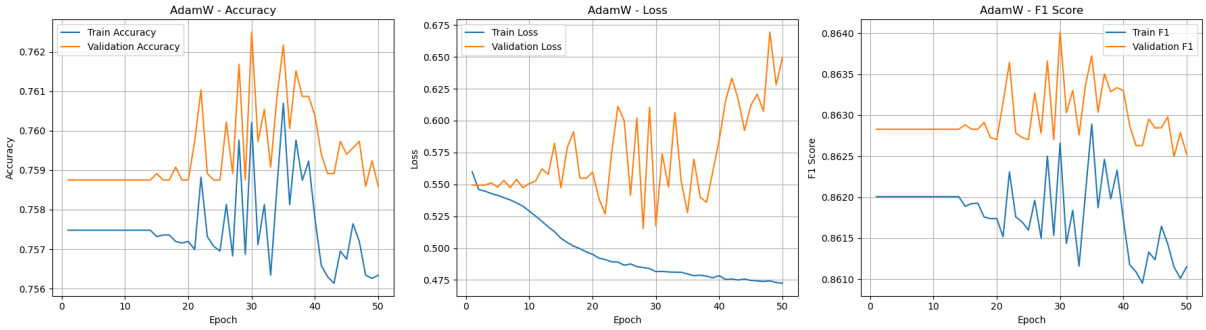


Figure 15: Training and validation curves for the GCN + L2 Reg. on HVG data.

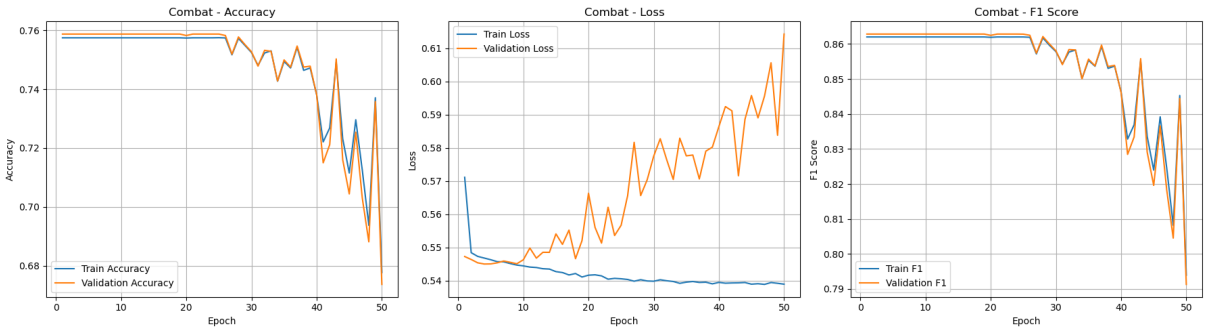


Figure 16: Training and validation curves for the GCN + ComBat on HVG data.

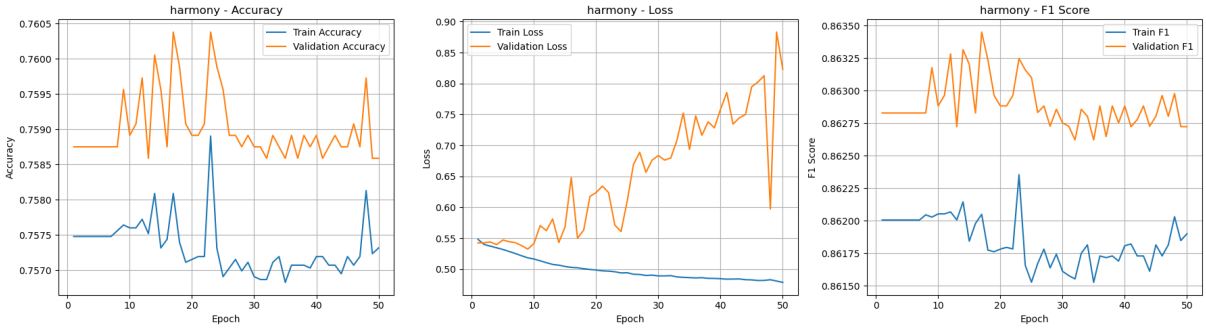


Figure 17: Training and validation curves for the GCN + Harmony on HVG data.

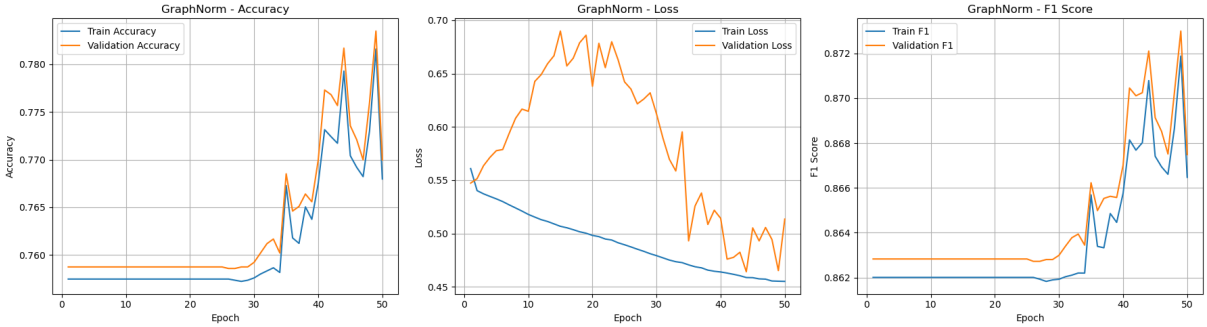


Figure 18: Training and validation curves for the GCN + GraphNorm on HVG data.

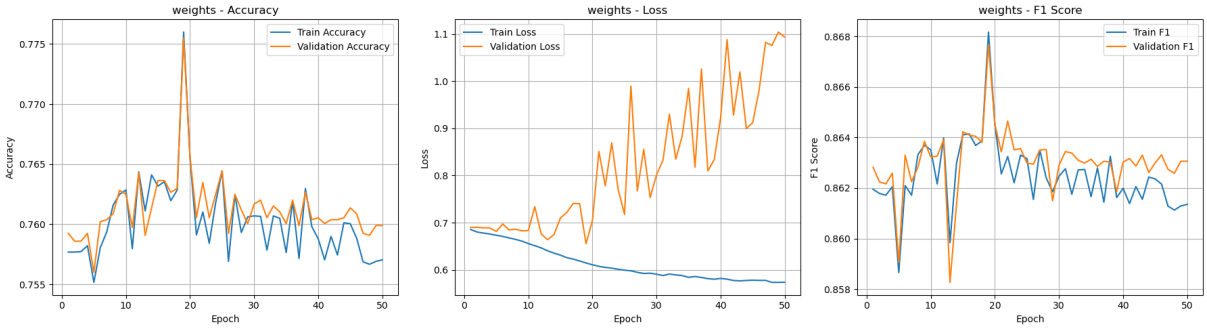


Figure 19: Training and validation curves for the GCN + Weighted Loss on HVG data.

Graph Attention Networks (GAT)

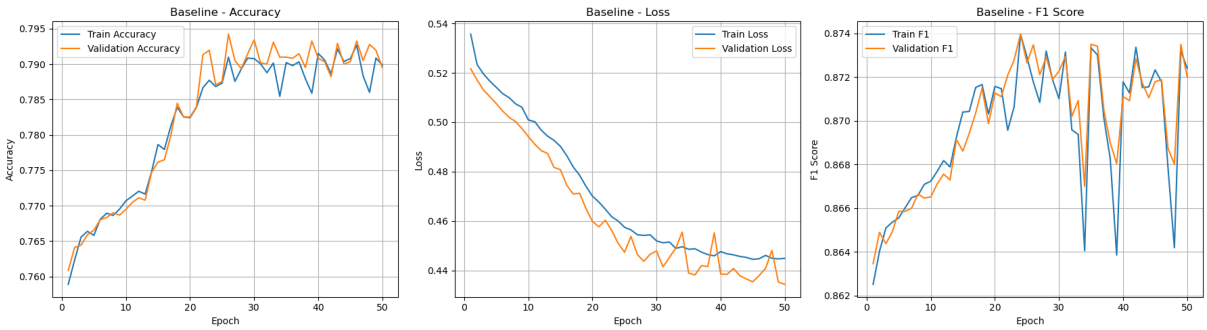


Figure 20: Training and validation curves for the GAT Baseline on HVG data.

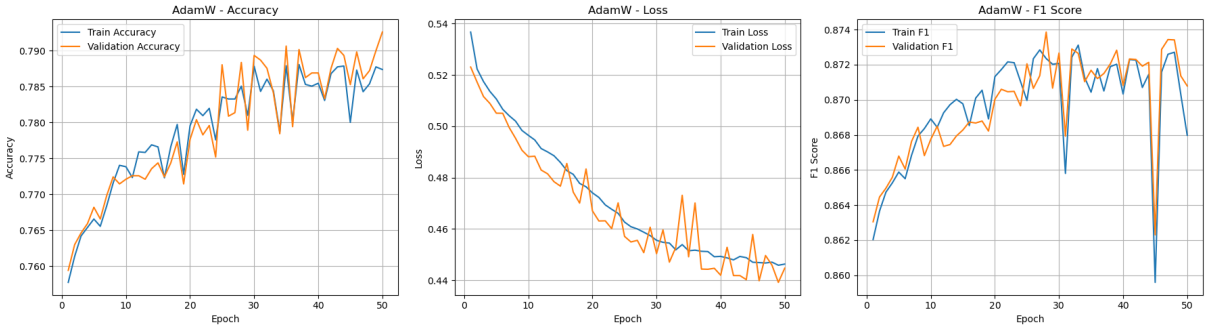


Figure 21: Training and validation curves for the GAT + L2 Reg. on HVG data.

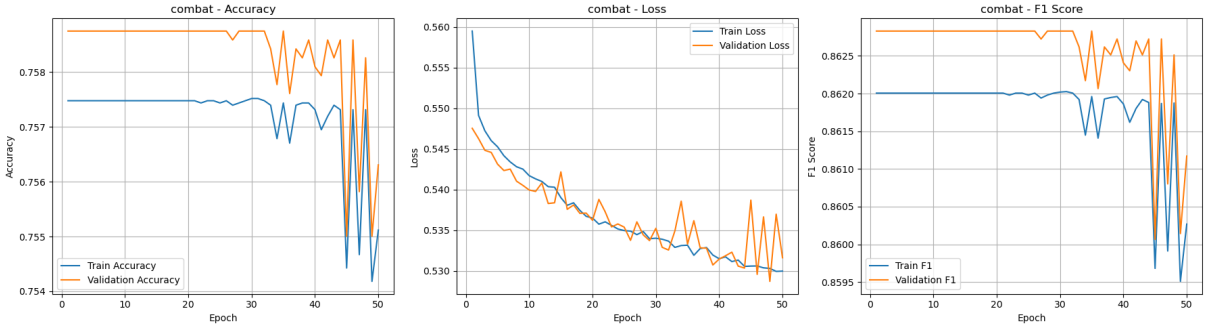


Figure 22: Training and validation curves for the GAT + ComBat on HVG data.

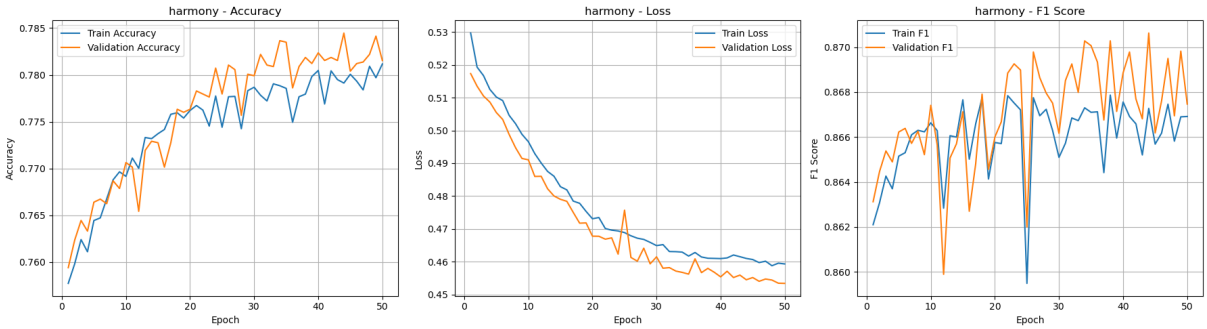


Figure 23: Training and validation curves for the GAT + Harmony on HVG data.

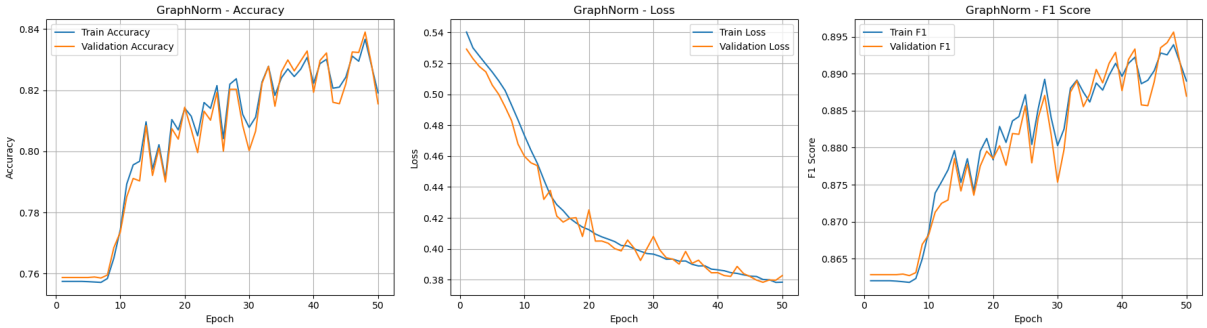


Figure 24: Training and validation curves for the GAT + GraphNorm on HVG data.

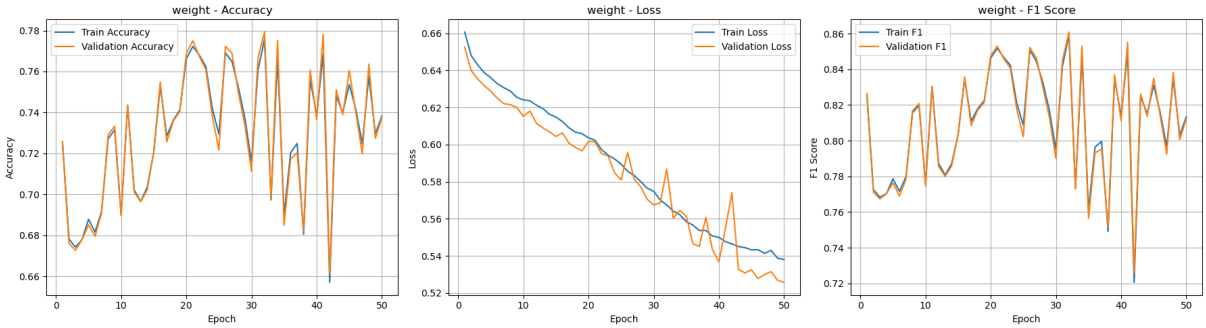


Figure 25: Training and validation curves for the GAT + Weighted Loss on HVG data.

Hyperparameter Tuned Model

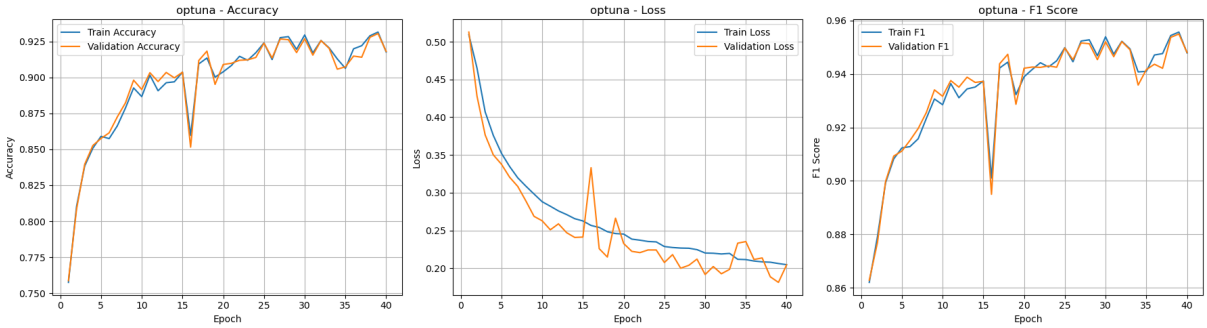


Figure 26: Training and validation curves for the fine tuned GAT + GraphNorm on HVG data.

TP53 Target Genes

Graph Convolutional Networks (GCN)

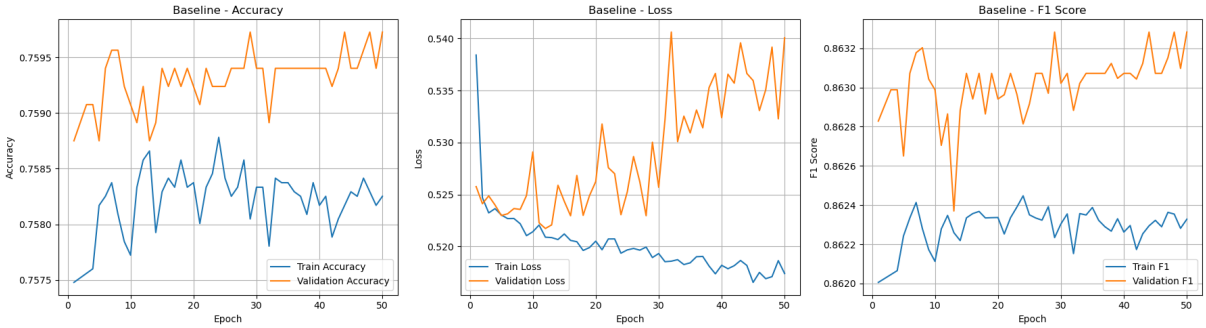


Figure 27: Training and validation curves for the GCN baseline on TP53 Target data.

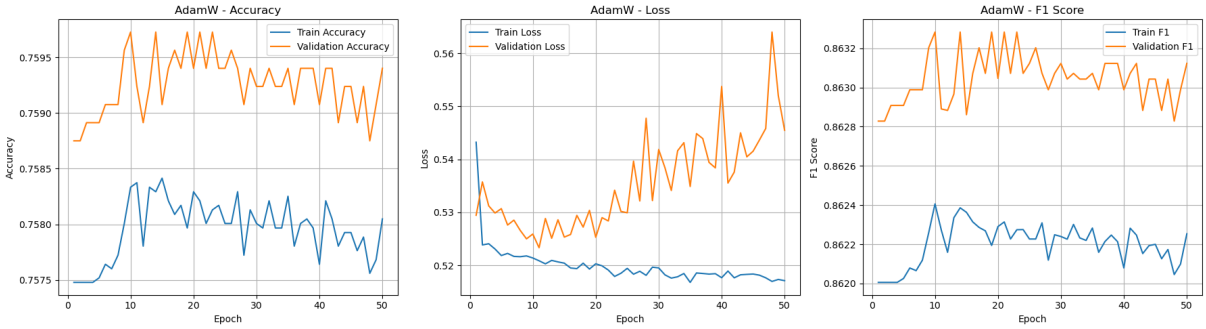


Figure 28: Training and validation curves for the GCN + L2 Reg. on TP53 Target data.

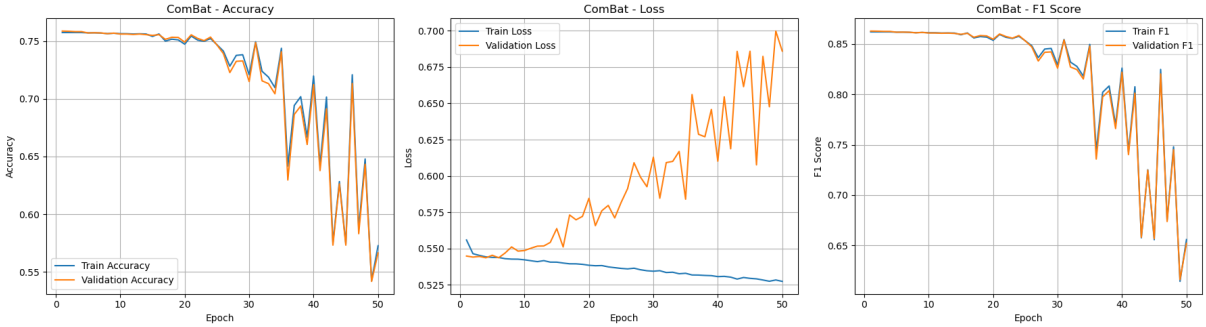


Figure 29: Training and validation curves for the GCN + ComBat on TP53 Target data.

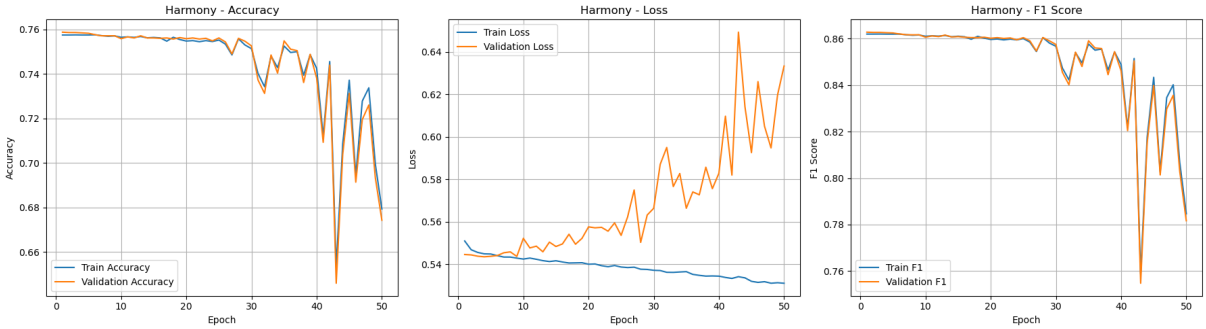


Figure 30: Training and validation curves for the GCN + Harmony on TP53 Target data.

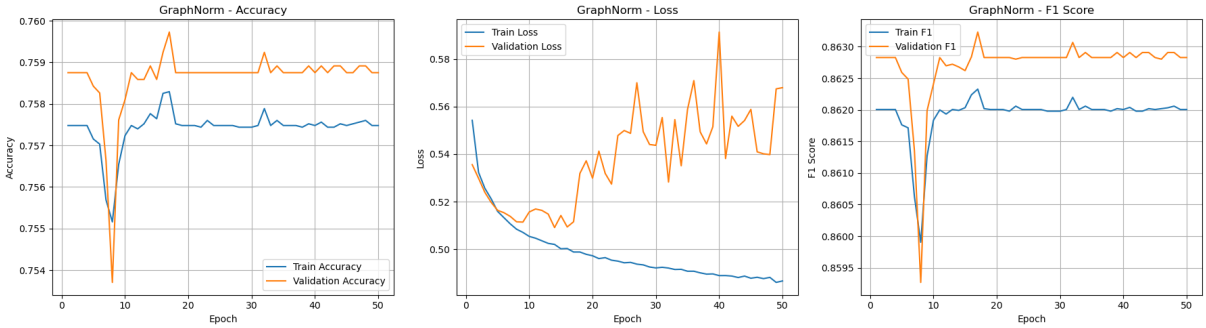


Figure 31: Training and validation curves for the GCN + GraphNorm on TP53 Target data.

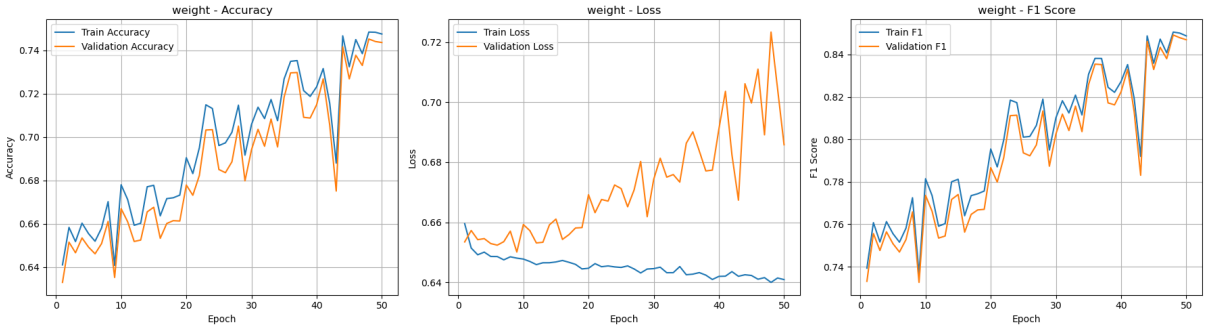


Figure 32: Training and validation curves for the GCN + Weighted Loss on TP53 Target data.

Graph Attention Networks (GAT)

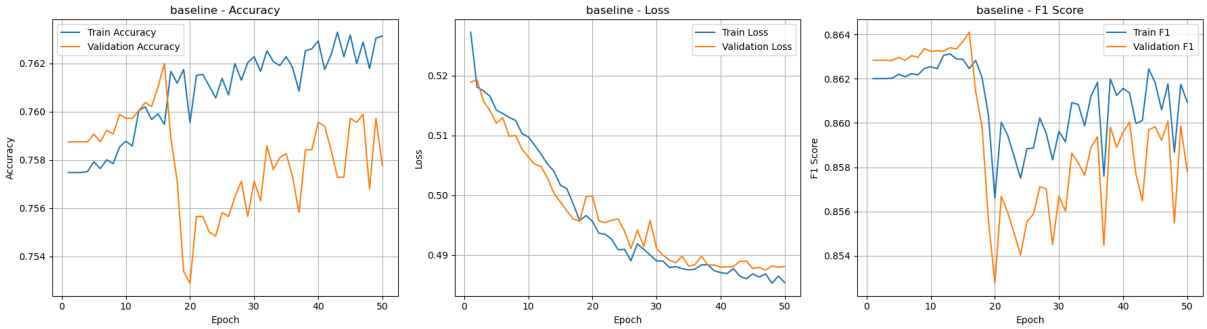


Figure 33: Training and validation curves for the GAT baseline on TP53 Target data.

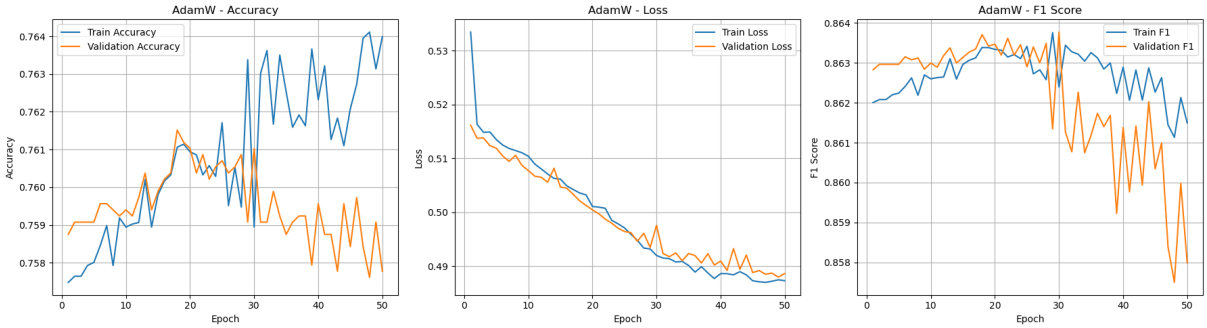


Figure 34: Training and validation curves for the GAT + L2 Reg. on TP53 Target data.

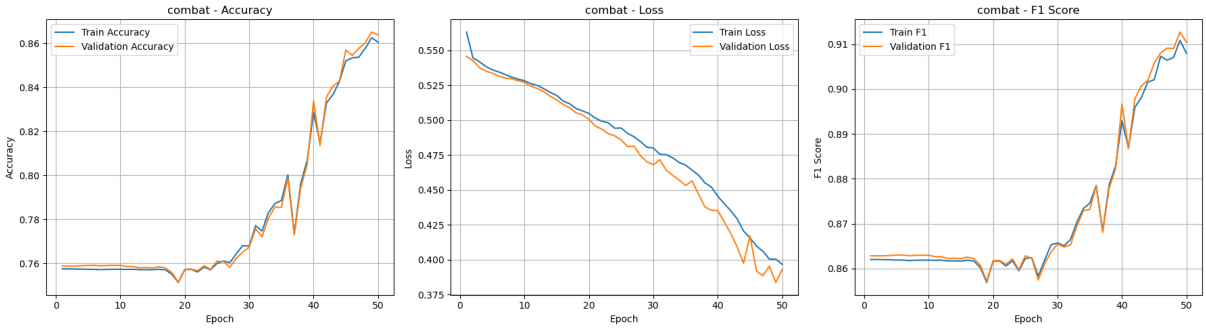


Figure 35: Training and validation curves for the GAT + ComBat on TP53 Target data.

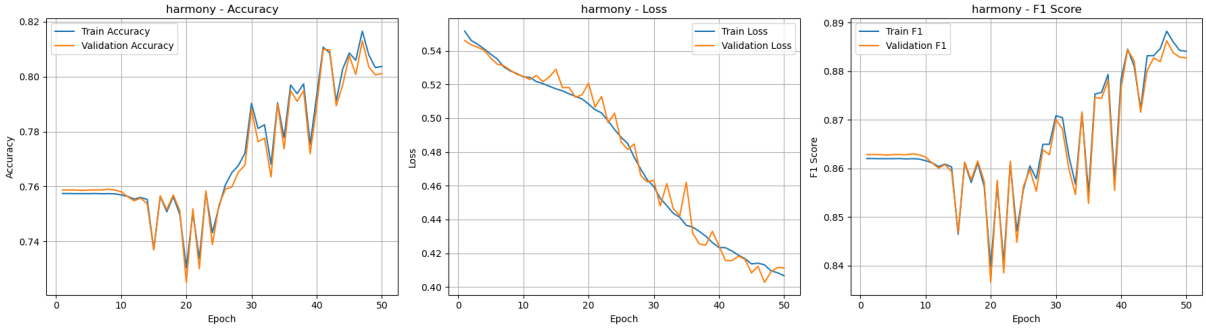


Figure 36: Training and validation curves for the GAT + Harmony on TP53 Target data.

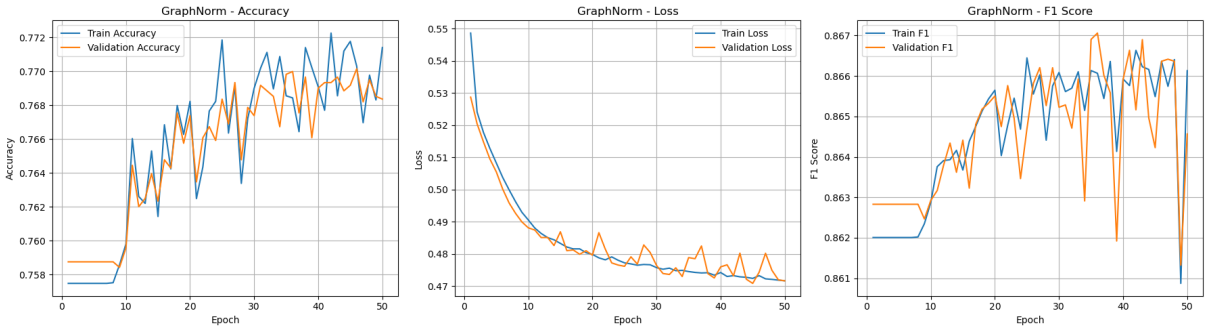


Figure 37: Training and validation curves for the GAT + GraphNorm on TP53 Target data.

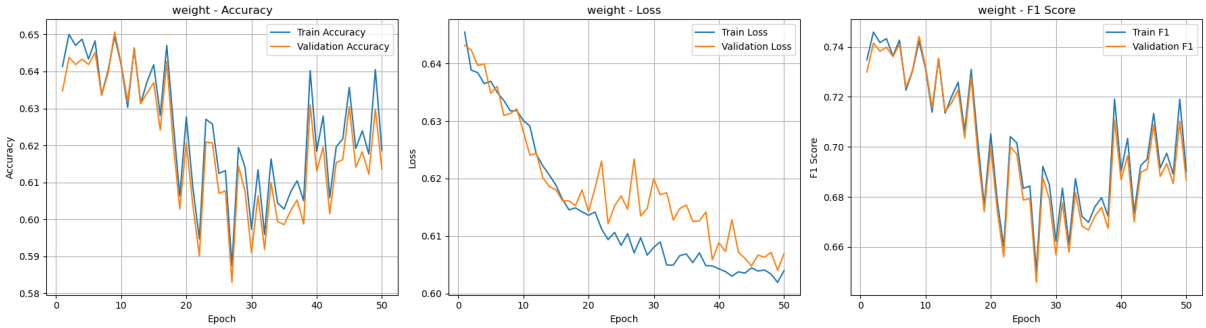


Figure 38: Training and validation curves for the GAT + Weighted Loss on TP53 Target data.

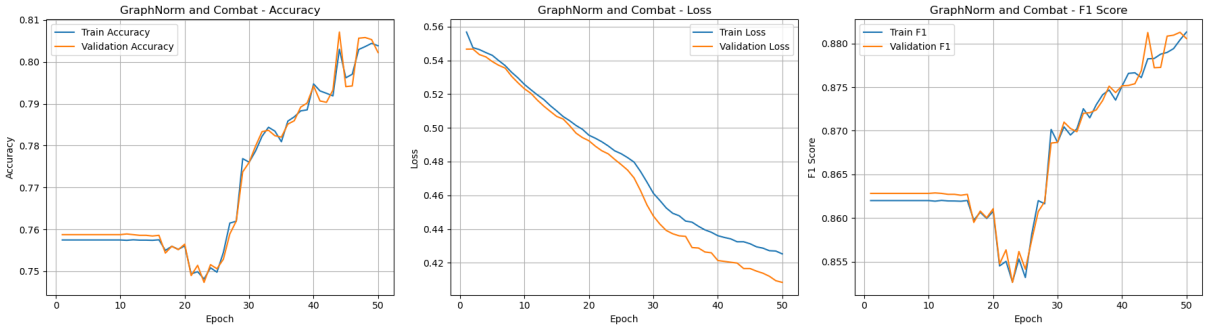


Figure 39: Training and validation curves for the GAT + Weighted Loss + ComBat on TP53 Target data.

Hyperparameter Tuned Model

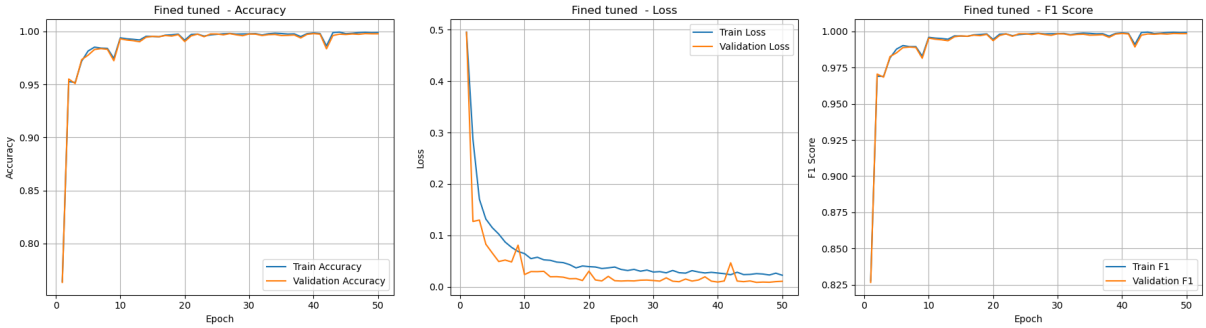


Figure 40: Training and validation curves for the fine tuned GAT + ComBat on TP53 Target data.

Appendix B - UMAP Visualizations After Batch Corrections

To visualize the effects of the different batch correction algorithms, we computed UMAP projections before and after applying them. Each plot shows the distribution of cells colored by cell line on the left and by TP53 mutation status on the right.

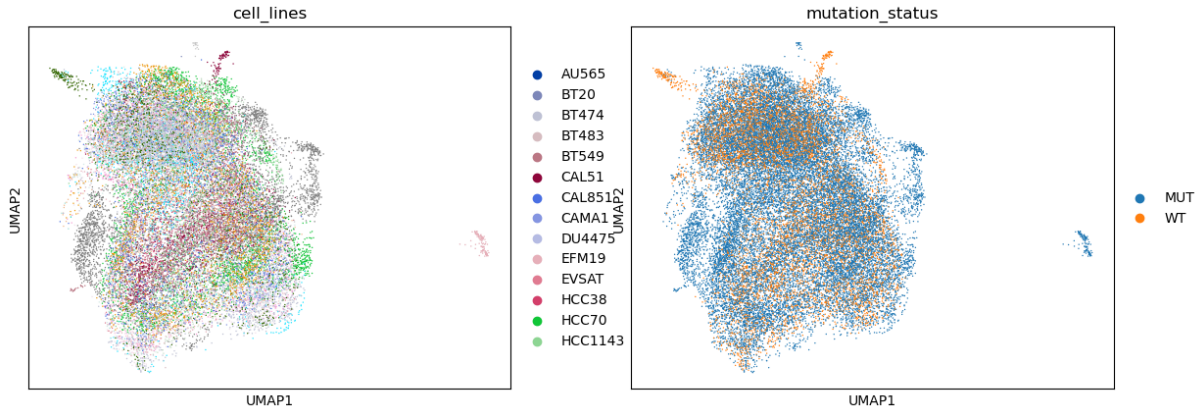


Figure 41: UMAP after ComBat correction applied on HVG features.

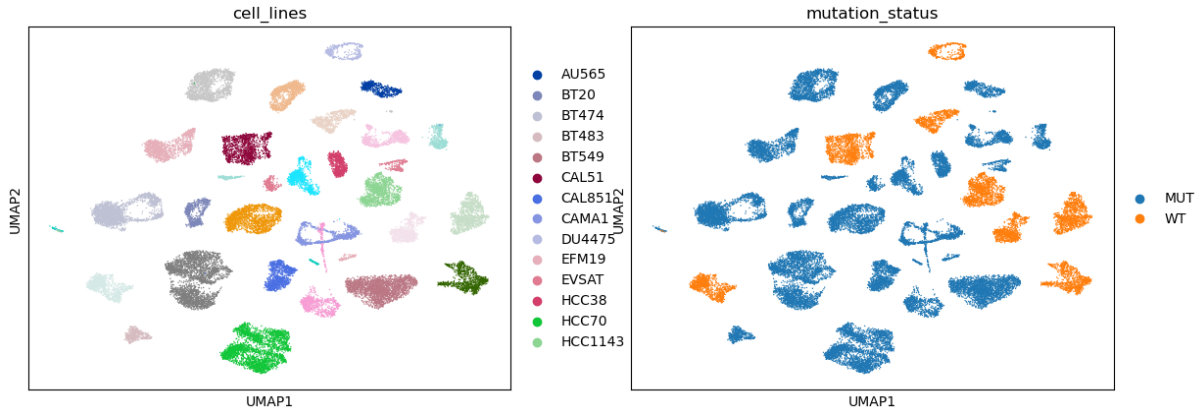


Figure 42: UMAP after Harmony correction applied on HVG features.

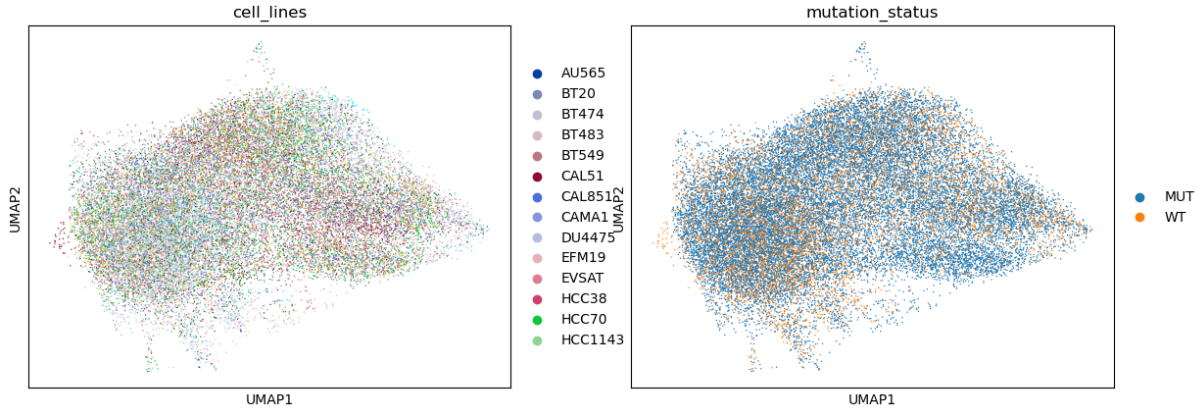


Figure 43: UMAP after ComBat correction applied on TP53 target genes.

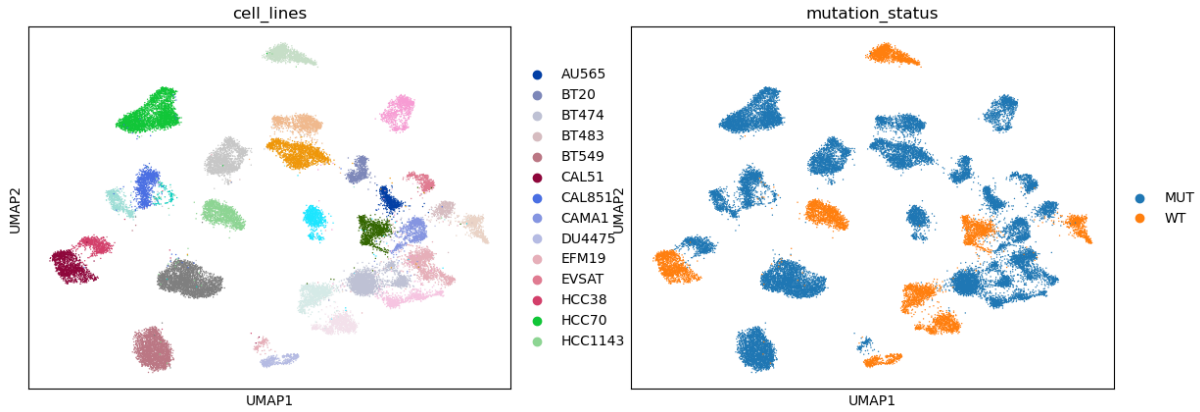


Figure 44: UMAP after Harmony correction applied on TP53 target genes.

Appendix C - Computing Resources

All computational tasks, including graphs construction, model training, and hyperparameter optimization, were executed on a high-performance computing (HPC) cluster managed via SLURM. The system was made of multiple compute nodes equipped with dual Intel Xeon processors, large memory capacities (up to 1TB RAM), and NVIDIA A100GPUs (up to 4 per node, 80GB each).

GPU acceleration was essential for an efficient training of Graph Neural Networks and large-scale hyperparameter tuning. The high-memory nodes, toghether with the multiple GPUs, allowed parallel processing of thousands of graph-based single-cell representations.