Graph-based deep learning approaches for phenotype prediction

Céline Brouard,
with Nathalie Vialaneix and Raphaël Mourad

23/11/2023

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité
INRAE

Outline

Introduction

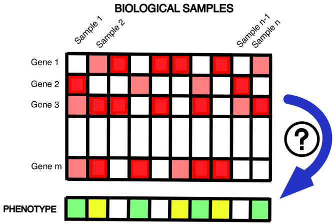
Graph Neural Network

Benchmark

Experiments

Objective

Prediction of phenotype from gene expressions

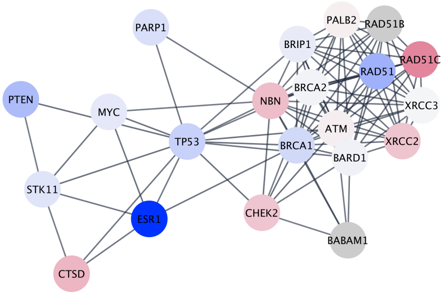


- Machine learning is increasingly used for transcriptomic-based predictions
  - Example: prediction of cancer type or the likelihood of a patient responding to a specific treatment

- Challenging due to the high dimensionality and small-to-moderate sample size

Objective

- Genes are organized into regulatory networks in cells
  → some works have used the **gene network information** to improve phenotype predictions



- Gene network: gene regulatory network, protein-protein interaction (PPI) network, co-expression network etc.

Earlier work in this field

Problem: predict y (numerical) from X (multivariate, dimension p) with a linear model:

$$y = X \times \beta + \epsilon$$

Examples:

- [Rapaport et al., 2007]: y is irradiated/not irradiated sample and X is gene expression. A network is given on the p genes based on KEGG metabolic pathways
- [Li and Li, 2008]: y is time to death (Glioblastoma) and X is gene expression. A network is given on the p genes based on KEGG metabolic pathways

Background and notations

We have a network (graph) $\mathcal{G}$, with p nodes $v_1, \ldots, v_p$ and edges between these nodes

An important matrix: the Laplacian

$$L_{ij}^{\mathcal{G}} = \begin{cases} -1 & \text{if } i \neq j \text{ and } v_i \text{ and } v_j \text{ are linked by an edge} \\ 0 & \text{if } i \neq j \text{ and } v_i \text{ and } v_j \text{ are not linked by an edge} \\ d_i & \text{if } i = j \end{cases}$$

with d_i the degree of nodes v_i

Eigendecomposition of the Laplacian

L is symmetric and positive so it can be decomposed into:

$$L = \sum_{i=1}^{p} \lambda_i e_i e_i^T$$

with λ_i the eigenvalues (in increasing order) and e_i the orthonormal eigenvectors in $\mathbb{R}^p$

To extract the most relevant information from the network, use the eigenvectors associated to the smallest eigenvalues:

- **low pass filter**: $F^{\mathcal{G}} = \sum_{i=1}^{r} \lambda_i e_i e_i^T$ for $r < p$
- **regularization**: $F^{\mathcal{G}} = \sum_{i=1}^{p} \phi(\lambda_i) e_i e_i^T$ with $\phi(\lambda_i) = e^{-\beta\lambda_i}$ or $\frac{1}{\lambda_i}$ for instance

[Rapaport et al., 2007]

- **Transformation of expression profiles**: spectral decomposition of gene expression profiles with respect to the eigenfunctions of the Laplacian

$$S_\phi(x_j) = \sum_{i=1}^{p} x_{ji}\phi(\lambda_i)e_i$$

- Optimization problem:

$$\min_{\beta\in\mathbb{R}^p} \sum_{i=1}^{n} \ell\left(\beta^t S_\phi(x_i), y_i\right) + C\|\beta\|^2$$

How to use L in prediction models ? [Li and Li, 2008]

Incorporate information on the gene network by using a **network constrained regularization**:

$$\arg\min_{\beta\in\mathbb{R}^p} \sum_{i=1}^{n} \left(\beta^t x_i - y_i\right)^2 + \lambda_1 \beta^T L \beta + \lambda_2 \|\beta\|_1$$

Motivation: genes that are linked on the network are expected to have similar functions and therefore smoothed regression coefficients

Implemented in R package glmgraph (not maintained, archived on CRAN)

Outline

Introduction

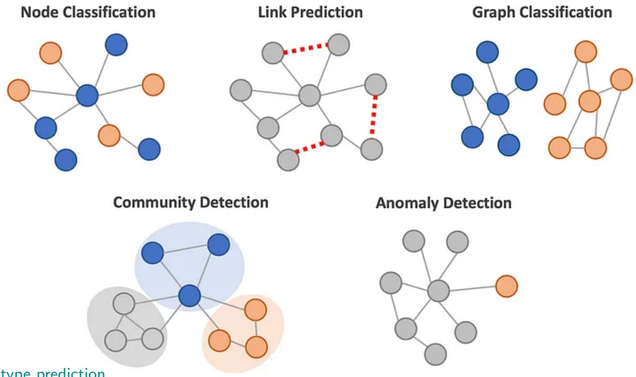
Graph Neural Network

Benchmark

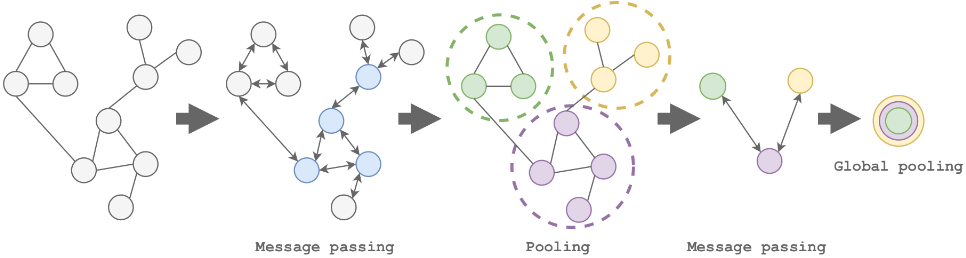
Experiments

Graph Neural Networks

- Recently graph neural networks (GNN) were proposed for phenotype prediction
- Particular type of convolutional neural network:
  - a graph representing pairwise relationships between nodes is used to drive the convolution
- GNN can be used to solve different problems:



Overview of GNN



General idea: the representation of a node is computed from the representations of nodes in the neighborhood

The last layer is fed to a standard MLP for prediction

Message passing layers

Generalization of convolutional layers to graph data

The representation of node v_i is learned iteratively with:

$$
\begin{aligned}
h_{v_i}^{0} &= x_i \\
h_{v_i}^{t+1} &= F\left(h_{v_i}^{t}, \square_{v_j \in \mathcal{N}(v_i)} \phi_t(h_{v_i}^{t}, h_{v_j}^{t})\right)
\end{aligned}
$$

- $\square$: differential permutation invariant function (mean, sum)
- F and ϕ_t: parameterized functions which parameters are learned during the training

Example of message passing layer

$$h_{v_i}^{t+1} = F\left(W_t \frac{1}{N(v_i)} \sum_{v_j \in \mathcal{N}(v_i)} h_{v_j}^t + B_t h_{v_i}^t\right)$$

W_t, B_t: trainable weight matrices

Matrix formulation:

$$H^{t+1} = F\left(D^{-1} A H^t W_t^T + H^t B_t^T\right)$$

GNN in practice

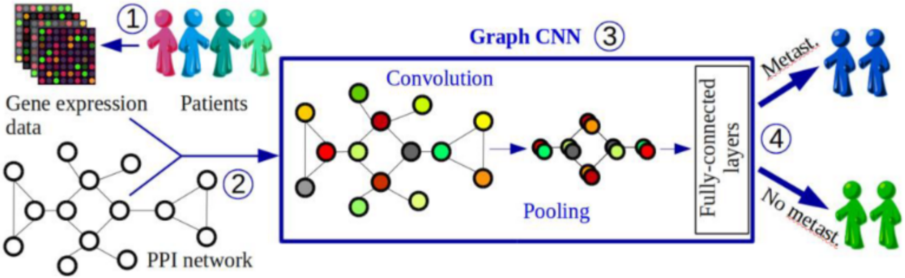
GNN libraries:

- **Spektral** [Grattarola and Alippi, 2020]
  - based on tensorflow
- **PyTorch Geometric** [Fey and Lenssen, 2019]
  - based on PyTorch
- also Graph Nets, Deep Graph Library

Graph Neural Networks for phenotype prediction

- Some authors have used GNNs for phenotype prediction
  - Example: metastatic event prediction
- They used biological knowledge on gene regulatory networks:
  - PPI networks or co-expression networks

Graph Neural Networks for phenotype prediction



Each patient is represented as a **graph signal**:

- the molecular network structures the genes and is the same for every patient
- patient's gene-expressions are assigned to the vertex of the network

Phenotype prediction is addressed as a **graph classification task**

Graph Neural Networks for phenotype prediction

- In other fields of applications, recent works tend to show that GNNs are frequently over-complex for the task [Errica et al., 2020, Böther et al., 2022, Santana et al., 2023]
- [Smith et al., 2020] even showed that classical ML methods often outperform deep learning for phenotype prediction
- $\Rightarrow$ simpler models can obtain comparable results
- Ratio between benefits and costs (in particular computational) of these methods ?

Outline

Introduction

Graph Neural Network

Benchmark

Experiments

Benchmark

Comprehensive and reproducible benchmark comparing GNN to other ML methods for transcriptomic-based phenotype prediction

- We used previously published datasets and models
- Systematic comparison using a common ground methodology

Published works

- **BreastCancer** [Chereda et al., 2021]:
  - prediction of metastasis within the first 5 years in breast cancer
  - PPI network (HPRD)
- **CancerType** [Ramirez et al., 2020]:
  - classification of different tumor and non-tumor samples into 33 cancer types or as normal (data from TCGA)
  - PPI network and co-expression network

Published works

- **F1000** [McDermott et al., 2020]:
  - gene expression profiles over 76 cell lines, that are treated with bioactive small molecules or genetic perturbations (LINCS)
  - 3 classification tasks : prediction of primary site (tissue type), subtype, drug mechanism of action
  - network of transcription-factor and micro-RNA regulatory relationships from several external datasets (RegNetwork)

These 3 works used the model and the implementation of [Defferrard et al., 2016]. This model uses **Chebnets** as convolutional layer and **graph coarsening** as pooling.

Chebnets [Defferrard et al., 2016]

It is based on a **spectral decomposition of the graph**

$$y = g_\theta(L)x = \sum_{k=0}^{K} \theta_k T_k(\tilde{L})x$$

- $\tilde{L}$: scaled Laplacian
- T_k: Chebyshev polynomial of order k
- θ_k: layer's trainable parameters

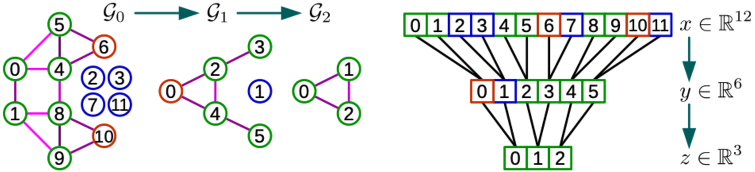
It can capture information from a node's wider neighborhood by including higher-degree polynomials

GCN [Kipf and Welling, 2017]

- More scalable approach by using a first-order approximation of spectral graph convolution
- A linear model w.r.t. L is considered by limiting K to 1.
- Using this model and a single parameter θ, the equation simplifies to:

$$y = \theta \left(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x$$

Graph coarsening [Defferrard et al., 2016]



1. **Multilevel clustering algorithm**: each level produces a coarser graph (the size of the graph is reduced by a factor 2)
    - At each level, a vertex i is matched to the neighbor j that maximizes $A_{ij}(\frac{1}{d_i} + \frac{1}{d_j})$
2. **Fast pooling**
    - Vertices are arranged such that a graph pooling operation becomes as efficient as a 1D pooling
    - Creation of a balanced binary tree: fake (disconnected) nodes are added to pair with singletons

Outline

Introduction

Graph Neural Network

Benchmark

Experiments

Datasets

Dataset	# nodes	# observations	Prediction type (# classes)
BreastCancer [Chereda et al., 2021]	6,888	969	Classification (2)
CancerType [Ramirez et al., 2020]	4,444	11,070	Classification (34)
F1000 prostate [McDermott et al., 2020]	978	25,565	Classification (9)
F1000 full [McDermott et al., 2020]	978	156,461	Classification (12, 14, 49)
Simulated (new)	21	100	Regression

Simulated dataset

- Using the simulation tool sismonr
- Dataset generated from 20 genes
- 200 times steps were simulated for 100 independent individuals

Comparison

Comparison with different approaches:

- Standard machine learning methods: **random forest**, **multilayer perceptron**, **SVM**
- **glmgraph**: graph-constrained regression model
- **GNN**
- **GNNo**: GNN based on convolution between observations rather than between features

We systematically used cross-validation

Implementations

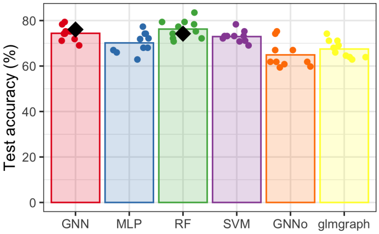
- **GNN**:
  - We kept the coarsening approach
  - We implemented the convolutional layer using the Spektral library and the neural network model in tensorflow/keras
- **GNNo**: modification of the implementation of GNN from keras

Implementations

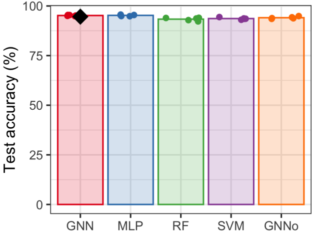
We also run the same methods with different implementations:

- **multilayer perceptron**: functions from the Python libraries scikit-learn and keras/tensorflow 2
- **SVM**: Python library scikit-learn and the R package **e1071**
- **random forests**: Python library scikit-learn and the R package **randomForest**

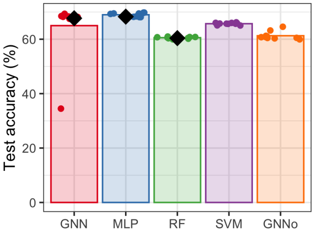
Results: test accuracy



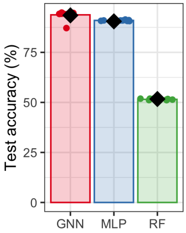
BreastCancer



CancerType



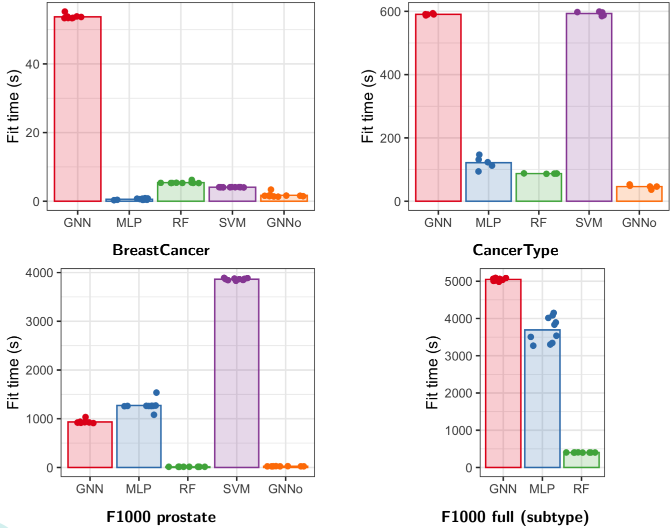
F1000 prostate



F1000 full (subtype)

- Good reproducibility of published results
- Except in F1000 full, GNN is not the best method
- Unlike GNN, other methods (MLP, RF, SVM) were used with default hyperparameters
- No clear winner stands out
- GNN performs better than GNNo

Results: computational time



BreastCancer

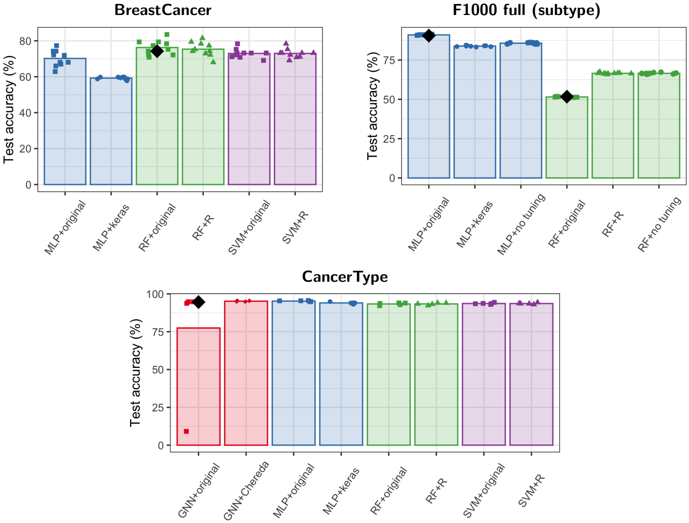
CancerType

F1000 prostate

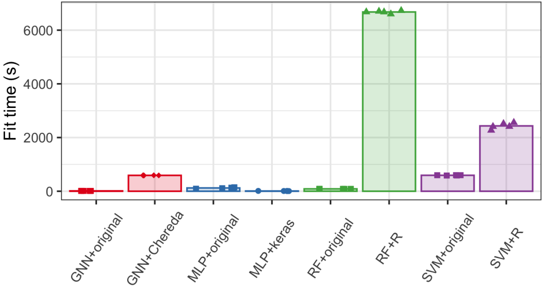
F1000 full (subtype)

- glmgraph is the most computationally demanding method for BreastCancer (not represented for the sake of readability)
- SVM is strongly influenced by the number of samples and the number of classes
- GNN computational time is increased when both the number of samples and the number of genes are large

Impact of the implementations: accuracy



Impact of the implementations: computational time



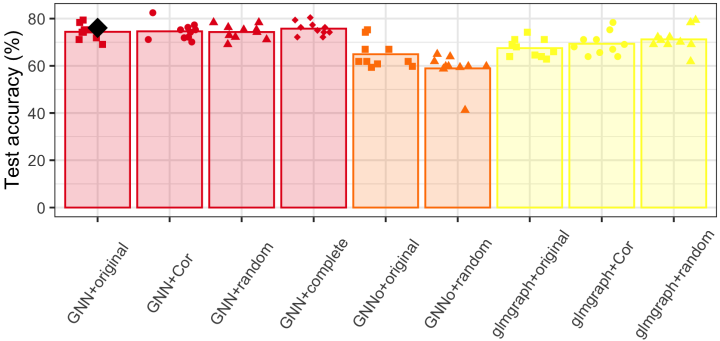
However, the improvements came sometimes at the cost of a larger computational time.

Impact of the input graph

In order to see the usefulness of the added information in graph based models, we also used these methods with naive graphs for the BreastCancer dataset:

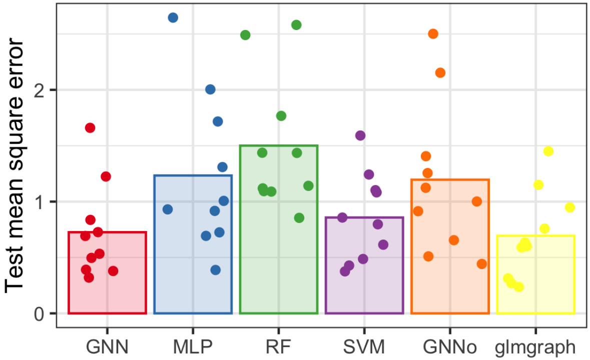
- **Cor**: simple thresholding of the Pearson correlation matrix between genes
- **random**: random permutation between gene edges (to obtain random graph with same degree distribution)
- **complete**: complete graph

Impact of the input graph

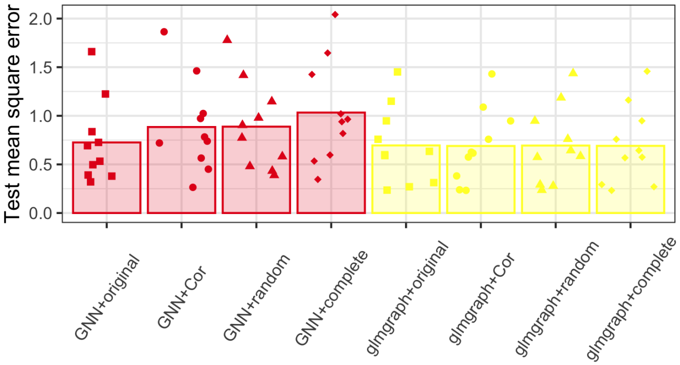


- The impact of the input network is not visible
- For GNN and glmgraph, the random and complete networks achieve better performance that networks based on biological knowledge

Results: simulated data



Results: simulated data

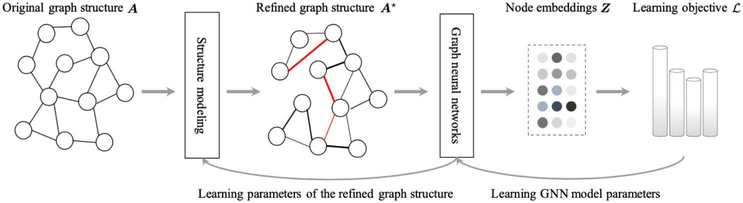


Discussion

- Standard ML methods, not explicitly accounting for the dependency structure between genes, frequently obtain better or comparable performance on the prediction task
- In addition, benchmarking with real expression datasets and irrelevant networks do not show decrease in performance compared to using a biologically relevant gene network
- When the network is perfectly known, better performances are obtained with GNN and glmgraph
- The lack of improvement for GNN with real data might be due to the low accuracy of available gene networks

Perspectives

Graph structure learning: learn simultaneously the relevant graph for the prediction task and the GNN's parameters


Original graph structure A
Structure modeling
Refined graph structure A^*
Graph neural networks
Node embeddings Z
Learning objective $\mathcal{L}$
Learning parameters of the refined graph structure
Learning GNN model parameters

- Few existing hybrid approaches, and not always relevant for omics data
- Difficulty: learning a discrete structure while descent gradient is used for learning GNN's parameters

References

(unofficial) Beamer template made with the help of Thomas Schiex and Andreea Dreau: https://forgemia.inra.fr/nathalie.villa-vialaneix/bainrae

Böther, M., Kißig, O., Taraz, M., Cohen, S., Seidel, K., and Friedrich, T. (2022).
What's wrong with deep learning in tree search for combinatorial optimization?
In *Proceedings of the 10th International Conference on Learning Representations (ICLR 2022)*.

Chereda, H., Bleckmann, A., Menck, K., Perera-Bel, J., Stegmaier, P., Auer, F., Kramer, F., Leha, A., and Beißbarth, T. (2021).
Explaining decisions of graph convolutional neural networks: patient-specific molecular subnetworks responsible for metastasis prediction in breast cancer.
Genome Medicine, 13:42.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016).
Convolutional neural networks on graphs with fast localized spectral filtering.
In Lee, D. D., von Luxburg, U., Garnett, R., Sugiyama, M., and Guyon, I., editors, *Advances in Neural Information Processing Systems (NIPS 2016)*, volume 29, pages 3844–3852, Red Hook, NY, USA. Curran Associates Inc.

Errica, F., Podda, M., Bacciu, D., and Micheli, A. (2020).
A fair comparison of graph neural networks for graph classification.
In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*.

Fey, M. and Lenssen, J. E. (2019).
Fast graph representation learning with pytorch geometric.
In *Proceedings of RLGM Workshop at ICLR 2019*.

Grattarola, D. and Alippi, C. (2020).
Graph neural networks in TensorFlow and Keras with Spektral.
In *Proceedings of the Graph Representation Learning and Beyond – ICML 2020 Workshop*.

Kipf, T. N. and Welling, M. (2017).
Semi-supervised classification with graph convolutional networks.
In *In Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, Toulon, France.

Li, C. and Li, H. (2008).
Network-constrained regularization and variable selection for analysis of genomic data.
Bioinformatics, 24(9):1175–1182.

McDermott, M. B., Wang, J., Zhao, W.-N., Sheridan, S. D., Szolovits, P., Kohane, I., Haggarty, S. J., and Perlis, R. H. (2020).
Deep learning benchmarks on L1000 gene expression data.
IEEE/ACM Transations on Computational Biology and Bioinformics, 17(6):1846–1857.

Ramirez, R., Chiu, Y.-C., Hererra, A., Mostavi, M., Ramirez, J., Chen, Y., Huang, Y., and Jin, Y.-F. (2020).
Classification of cancer types using graph convolutional neural networks.
Frontiers in Physics, 8.

Rapaport, F., Zinovyev, A., Dutreix, M., Barillot, E., and Vert, J.-P. (2007).
Classification of microarray data using gene networks.
BMC Bioinformatics, 8:35.

Santana, I., Lodi, A., and Vidal, T. (2023).
Neural networks for local search and crossover in vehicle routing: a possible overkill?
In Goos, G. and Hartmanis, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Operations Research (CPAIOR 2023)*, volume 13884 of *Lecture Notes in Computer Science*, pages 184–199, Nice, France. Springer Nature, Switzerland.

Smith, A. M., Walsh, J. R., Long, J., Davis, C. B., Henstock, P., Hodge, M. R., Maciejewski, M., Mu, X. J., Ra, S., Zhao, S., Ziemek, D., and Fisher, C. K. (2020).
Standard machine learning approaches outperform deep representation learning on phenotype prediction from transcriptomics data.
BMC Bioinformatics, 21:119.