# Report: A Geometric Deep Learning Framework for Single-cell Trancriptomics Data Analysis

***Elyas Heidari*[*1] *and Laleh Haghverdi*[†2]**

[1]Department of Biological Systems Sciences and Engineering, ETH Zürich, Switzerland
[2]EMBL Heidelberg, Germany

*eheidari@student.ethz.ch †laleh.haghverdi@embl.de

**July 27, 2020**

**Abstract**

abstract.

## Contents

# 1 Background and motivation

# 2 General setting, objectives, and challenges

As emphasized previously, current single-cell classification methods take genes as independent features to assign cell types, disregarding their interactome which has been shown to be of great importance in cellular processes. Considering the local and modular architecture of gene regulatory networks (GRNs), we propose to further classify and use this information for cell type identification in new data sets.

Recent advancements of deep learning models to capture local (modular) structures in other applications such as image, audio and speech analysis as well as emergence of high-throughput scRNA-seq technologies which provide the large amounts of data required for such models motivated us to employ deep learning models applicable to non-regular network data. Graph Convolutional Networks (GCNs) amongst other deap learning models, have significant power in extracting informative feature maps from signals defined over nodes of a graph while considering meaningful local structures and connectivities among them, making them extremely suitable for analyzing non-regular gene regulatory networks (Zhou et al. 2018).

Our framework consists of three steps; 1) adaptation of a geometric deep learning model applicable to (non-regular) gene regulatory networks (GRNs), 2) construction of a modular GRN, which serves as the input graph structure for the geometric deep learning model, and 3) validation of the model on available single-cell RNA sequencing (scRNA-seq) data sets. For each of these steps, there are multitude of methods proposed in the relevant literature. Our main challenge however, was to find the best amongst the set of possible solutions. This immediately divides into 1) how to integrate all steps into one pipeline and later 2) how to optimize the whole pipeline to achieve the best solution. Other challenge, which specifically rises in the field of biology is that of interpretability. This objective is difficult to meet since we use deep learning, a well-known black box solution(Samek, Wiegand, and Müller 2017). Furthermore, one should note that the best solutions are not necessarily the most interpretable ones. We further elaborate on the matter of interpretability later on.

## 2.1 The scGCN

To address the challenges mentioned, we introduced single-cell Graph Convolutional Network (scGCN) which takes gene-gene interactions into account by constructing a co-expression network of a subset of genes and uses a graph deep learning approach to classify single-cells. The pipeline is an integrated data flow, with tunable options to achieve the most accurate cell type annotation. Moreover, multiple visualization tools are provided at different stages to facilitate interpretation (Fig. 1[1]).

[1] **HVG:** highly variable gene, **TF:** transcription factor, **PGM:** probabilistic graphical models

# 3 The solution space

We sought to introduce a solution to 1) present gene-gene interactions as a network (a GRN) and then feed it into a GCN and 2) use a GCN which accurately annotates cells given a GRN and is generalizable to new datasets. For either of the steps quite a few solutions have been developed in the literature. Nevertheless, our aim is to find the best combination of methods

with respect to our main goal. This raises particular challenges, e.g., due to computational limitations, feeding the complete GRN to a GCN is not possible, and one has to feed a subset of the GRN to the GCN. Another challenge is that most of the methods for GRN reconstruction, including knowledge-based or data-based, produce neither modular nor sparse networks and hence are not suitable for GCNs, as they are inherently designed to exploit local (modular) structures of the input graphs to perform the learning task.

## 3.1 The pipeline

In order to formulate the problem, we designed a pipeline of modules each of which represent a particular functionality. Each module is parametrized to ensure flexibility in trying different methods available for each step. This translates the whole solution space into a parameter grid, where parameters are methods and their arguments. Such an approach enables us to examine different solutions as paths of parameters on the parameter grid. The pipeline has three main modules each of which consist of multiple submodules. In the following we describe each of the modules in detail.

### Module 1: prepare GCN's input

After selection of training and test datasets, one has to preprocess them and prepare them to feed them to the GCN module. Preprocessing includes three steps, first, cell type selection and cell-wise data preprocessing, second, gene subset selection and lastly, GRN reconstruction. For each of these steps, we encountered some challenges and define our solution space as follows.

- **Cell type selection & preprocessing:** This stat is straightforward, in that one should select a subset of cell type in the intersection of cell types present in both training and test datasets. Afterwards, data for each cell should be preprocessed to rule out major technical noises. We used a very general method to perform preprocessing, that is cell-wise log scaling, with no tunable parameter.
- **Gene subset selection:** The main objective for gene subset selection is to select genes which are most relevant to cell type identification. There are multiple ideas discussed in the literature but we restricted our solution space to transcription factors (TFs) and highly variable genes (HVGs) which are both known to play a significant role in cell type processes. Once one decides which of the gene sets they are going to use, they should decide how many genes they are going to pick to construct the GRN upon.
- **GRN reconstruction:** There are numerous methods for GRN reconstruction proposed, some of which rely on knowledge bases, such as regNetwork [ref], and some simply use the data itself to estimate the underlying GRNs for the genes in hand, probabilistic graphical models and coexpression networks embody such an approach. With the aim of choosing the best method for our pipeline, we examined both approaches. The most important features of the GRN we were seeking were sparsity and modularity of the GRN.

Overall, while seeking the best way to prepare input data for the GCN among multitude of solutions, we restricted our solution space to the ones which are computationally feasible and potentially enhance the learning efficiency of the GCN the most.

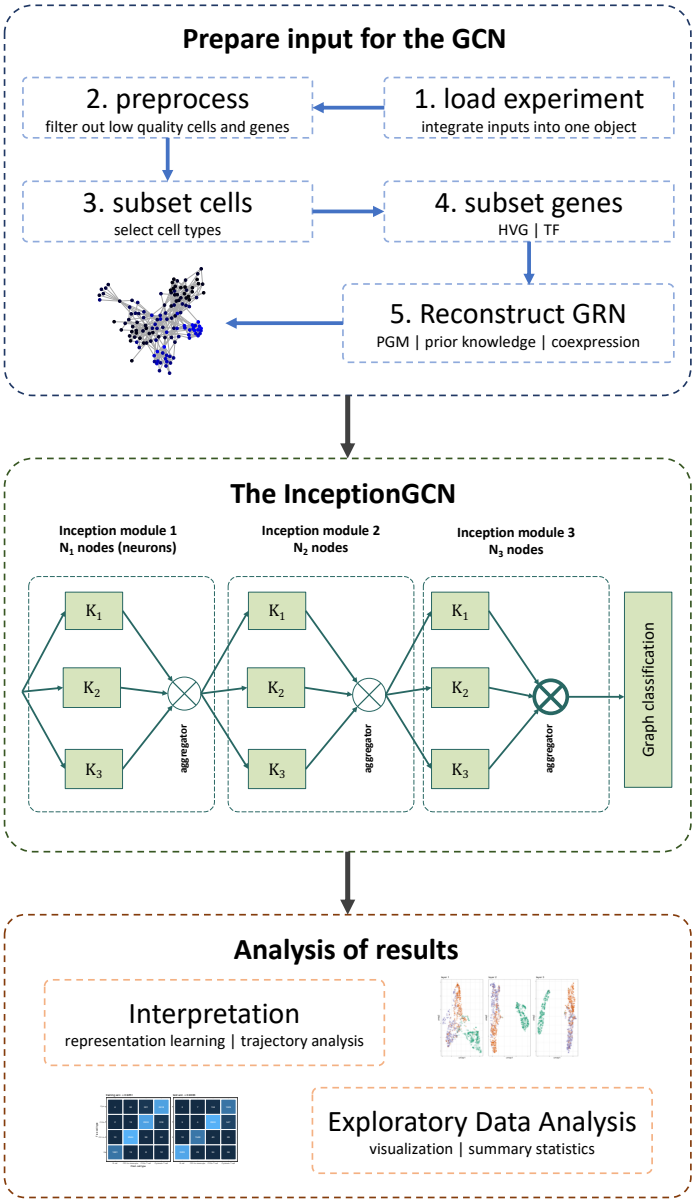


**Figure 1:** The graphical abstract of the scGCN

## Module 2: the GCN

Among geometric deep learning strategies we decided to use the implementation of the InceptionGCN (Kazi et al. 2019), a Graph Convolutional Network (GCN) approach that captures a great range of distinct locality sizes on the network, hence suitable for the context of learning on modular gene networks. Moreover, ([cite] blah blah shows that) as an instance of a regular GCN, InceptionGCN has a mathematical interpretation, in contrast to most of

other geometric deep learning models. The (hyper)parameters of InceptionGCN, e.g., number of layers, number of hidden nodes, and locality sizes should be tuned to find the optimal solution.

### Module 3: validation and interpretation of results

The last module of the pipeline precesses the output of the GCN and tries to interpret them. This is utilized by proper visualizations and computations to evaluate specific measures, such as annotation accuracy, technical noise removal, and computation dynamics.

## 4 `scGCNUtils` package

Once we parametrize our solution space, we are able to implement it. We implement the pipeline as a software package in R and python. The package `scGCNUtils` [ref] provides the user with the building blocks of the pipeline as well as performing pre-/post- exploratory data analysis (EDA). The user is able to run an end-to-end pipeline by defining their desired parameters, and decide on the ultimate parameter path using EDA tools provided for each module. The package also facilitates benchmarking, in that each step of the pipeline is implemented as an independent module and can be tweaked or substituted by another module with the same functionality. In the end, the performance and efficiency of the path can be validated by the package. Logging and caching are added to the package for the same purpose.

Ever since the beginning of the project, we reformatted the implementation from scratch quite a few times. For each dataset, we had to implement an ad-hoc pipeline, as different labs publish their studies in different formats, besides using different scRNA-seq platforms. We had to integrate new datasets into specific data objects in our programming environment, and pass them through a pipeline, specifically tailored for the dataset in hand. Eventually, we decided to develop a package to automate everything for the user. We integrated all of the tools we had exploited throughout the project into a single end-to-end pipeline, in which one can set specific parameters to validate the parameter path optimality. Also, each input experiment is stored in a specific data object in the environment with certain features and attributes which are modified in the pipeline data flow step by step. The whole pipeline is optimized in memory and performance by using appropriate data structures and computational resource optimization. Integrating everything into one single package also facilitates benchmarking, as now carrying out new experiments are made as convenient as reparametrization of the pipeline.

For the sake of reproducibility and meeting the criteria of open science, the package is designed to meet the standards of software engineering such as coherence, integrity, documentation, readability, and testability.

## 5 Experiments and results

We examined the functionality of our framework on several datasets, e.g., (Paul et al. 2015), Nestorowa et al. (2016)], and (Cao et al. 2019). The following path turned out to be the optimal for all most all of our experiments.

1. **Cell subset selection:** Select a balanced subset of cells from the training dataset, with respect to the cell types (at least 500 cells from each cell type). Select cells from the same cell types, from the test dataset.
2. **Gene subset selection:** Subset both datasets to the genes which are highly variable in the training dataset (at most 1000 genes).
3. **GRN reconstruction:** Use the glasso algorithm[2] and tune its parameters to reconstruct the GRN on the training dataset.
4. **Cell type classification:** Employ InceptionGCN and tune its hyper parameters (dependent on the input dataset), to achieve the optimal classification.

[2] A PGM to construct sparse Graphical Models (Friedman, Hastie, and Tibshirani 2008).

However, the nature of each dataset essentially determines the best path. Therefore, for a new dataset multiple paths should be explored.

## 5.1 The PBMC datasets

The pivotal application of our method is transferring (generalizing) a learned model to new datasets in which overcoming intra-dataset variations, e.g., batch effects is of great importance. Thus, we sought to find two dataset with the same cell type profile, but acquired in separate experiments to inspect how well our method is able to overcome unwanted variations. Here, we present an application of the scGCN on two human peripheral blood mononuclear cell (PBMC) scRNA-seq experiments (Ding et al. 2019). The dataset contains two separate experiments, hereafter called PBMC 1 and PBMC 2, generated in a single center, but employing seven different scRNA-seq methods. This suits our goal to verify our framework on a simulated task of real transfer learning. As in potential applications we expect our method to learn annotation priorly on a complete dataset and transfer the learned model to annotate an unseen one.

### 5.1.1 Input

We take PBMC 1 as our training dataset and PBMC 2 as our test dataset. The task is to learn the model on PBMC 1, and validate it on PBMC 2, by predicting cell type annotations, and comparing them to true labels. We select a subset of size 4 of cell types, to carry out our computational experiment. Afterwards, a subset of 300 most highly variable genes is selected from PBMC 1 and PBMC 2 is subset to the same set of genes. For GRN reconstruction, we use the glasso algorithm with tuned parameters, which gives us a modular as well as sparse GRN. The GRN is presented in Fig. 2.

Fig. 2(A) and Fig. 2(B) indicate that although our GRN reconstruction method is based on data and no prior knowledge, it is robust across datasets. As one finds in the figures, the network communities from PBMC 1, perfectly transfer to PBMC 2. Fig. 2(C) illustrates the same observation, as the density of expressions, i.e., activated modules are the same for both datasets, for each cell type. In fact, such an observation motivated us for the whole idea of exploiting a network topology to classify cell types.

### 5.1.2 Classification with InceptionGCN

We implement an InceptionGCN with three hidden layers with 36, 18, and 9 hidden nodes, respectively. At each layer, a complex, non-linear transformation (Chebnet filters) are applied to the inputs (see Fig. 1). One can describe such operations as diffusing information along the input GRN which further facilitates an accurate classification. Our InceptionGCN gives an

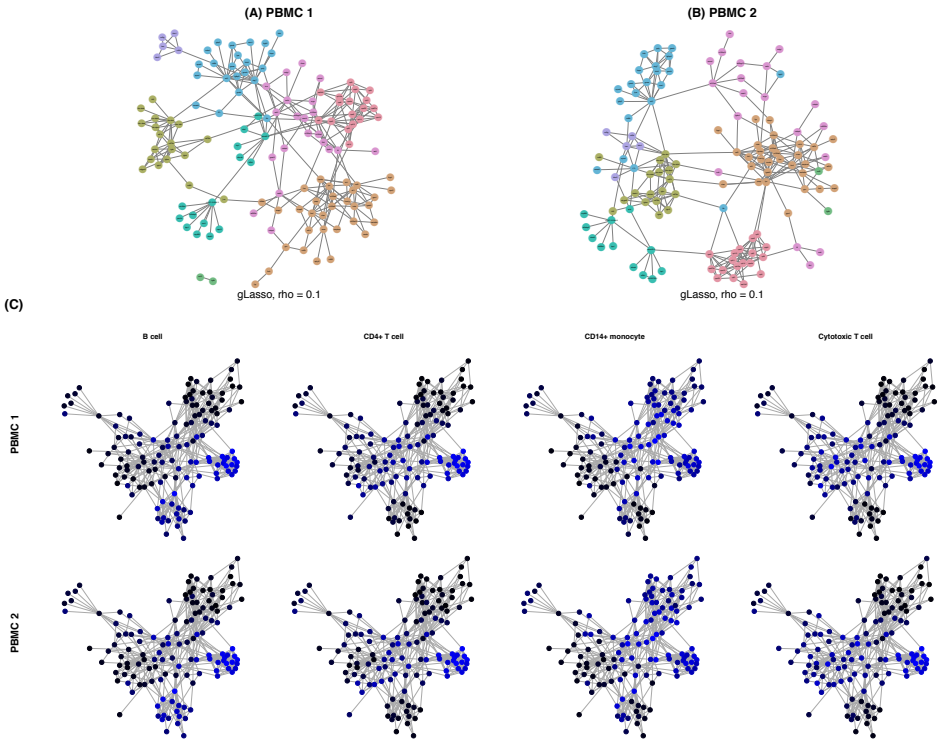


**Figure 2: The learned GRNs**
(A) and (B) GRNs learned on PBMC 1 and PBMC 2 respectively. Colors indicate network communities based on PBMC 1, with the louvain [ref] algorithm. (C) Density of expression for each class on the GRN learned from PBMC 1. Visualized for PBMC 1 and PBMC 2.

accuracy of 90% on test dataset, which is rather a very high accuracy in a transfer learning task, specifically, when the classes are quite similar to each other (including two classes of T cells). Fig. 3 indicates the results of the classification on training and test datasets.

### 5.1.3 Analyze the results

As mentioned before, one our main objectives was interpretability. Our framework enables us to explore the learning procedure in our deep neural network. Our design facilitates representation learning,[3] in that one can visualize the data at the hidden layers and observe the trajectory of data passing through the complex transformations, done by the InceptionGCN. This gives an overview of the learning procedure, inspecting batch-effect removal, and heterogeneous cell populations being separated. One can also use the same information to study the trajectory of expression density along the GRN, for each class. This gives one an idea of how information diffuses on the network and utilizes the classification task. Fig. 4 represents an embedding trajectory analysis, visualized as UMAPs[4]. As indicated, we observe gradual batch-effect removal and cluster separation along the layers.

[3] **Representation learning** is learning representations of the data by transforming or projections into repetitively simpler spaces (Bengio, Courville, and Vincent 2013).

[4] **U**niform **M**anifold **A**pproximation and **P**rojection is a non-linear dimension reduction method.

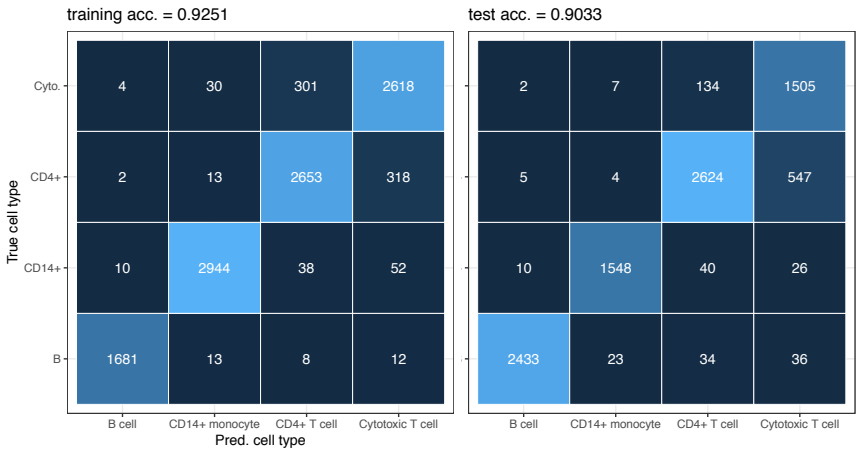


**Figure 3:** **The confusion matrices of the classification**
Each (table) cell indicates the number of respective (biological) cells.

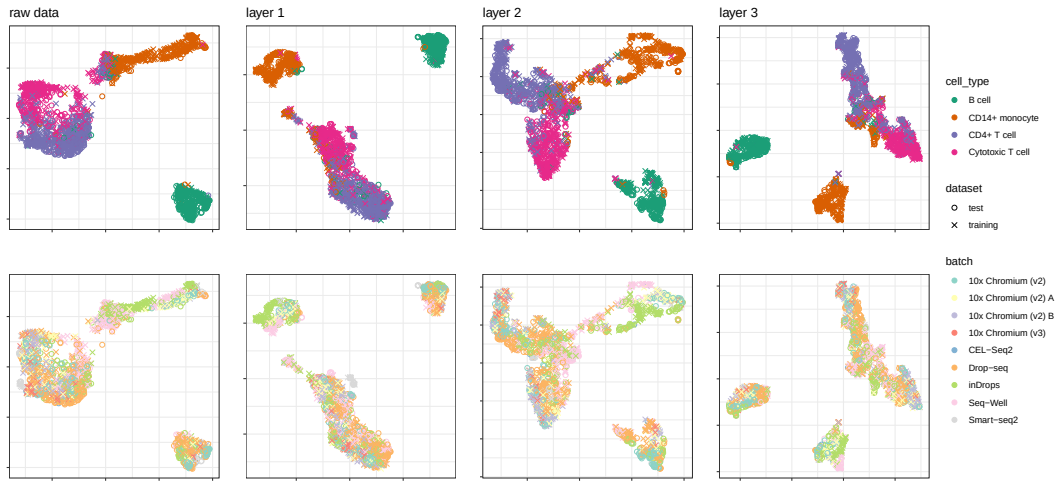


**Figure 4:** **UMAPs of the raw and hidden embeddings**

# 6 Discussion

# 7 Future steps

# Code and data availability

- Code: GitHub repository
- PBMC dataset: link

# References


Bengio, Yoshua, Aaron Courville, and Pascal Vincent. 2013. "Representation Learning: A Review and New Perspectives." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (8). IEEE: 1798–1828.

Cao, Junyue, Malte Spielmann, Xiaojie Qiu, Xingfan Huang, Daniel M Ibrahim, Andrew J Hill, Fan Zhang, et al. 2019. "The Single-Cell Transcriptional Landscape of Mammalian Organogenesis." *Nature* 566 (7745). Nature Publishing Group: 496–502.

Ding, Jiarui, Xian Adiconis, Sean K Simmons, Monika S Kowalczyk, Cynthia C Hession, Nemanja D Marjanovic, Travis K Hughes, et al. 2019. "Systematic Comparative Analysis of Single Cell Rna-Sequencing Methods." *BioRxiv*. Cold Spring Harbor Laboratory, 632216.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 2008. "Sparse Inverse Covariance Estimation with the Graphical Lasso." *Biostatistics* 9 (3). Oxford University Press: 432–41.

Kazi, Anees, Shayan Shekarforoush, S Arvind Krishna, Hendrik Burwinkel, Gerome Vivar, Benedict Wiestler, Karsten Kortüm, Seyed-Ahmad Ahmadi, Shadi Albarqouni, and Nassir Navab. 2019. "Graph Convolution Based Attention Model for Personalized Disease Prediction." In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 122–30. Springer.

Nestorowa, Sonia, Fiona K Hamey, Blanca Pijuan Sala, Evangelia Diamanti, Mairi Shepherd, Elisa Laurenti, Nicola K Wilson, David G Kent, and Berthold Göttgens. 2016. "A Single-Cell Resolution Map of Mouse Hematopoietic Stem and Progenitor Cell Differentiation." *Blood, the Journal of the American Society of Hematology* 128 (8). American Society of Hematology Washington, DC: e20–e31.

Paul, Franziska, Ya'ara Arkin, Amir Giladi, Diego Adhemar Jaitin, Ephraim Kenigsberg, Hadas Keren-Shaul, Deborah Winter, et al. 2015. "Transcriptional Heterogeneity and Lineage Commitment in Myeloid Progenitors." *Cell* 163 (7). Elsevier: 1663–77.

Samek, Wojciech, Thomas Wiegand, and Klaus-Robert Müller. 2017. "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models." *arXiv Preprint arXiv:1708.08296*.

Zhou, Jie, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2018. "Graph Neural Networks: A Review of Methods and Applications." *arXiv Preprint arXiv:1812.08434*.