POLITECNICO DI TORINO
MASTER DEGREE COURSE IN BIOMEDICAL ENGINEERING

MASTER DEGREE THESIS

A PIPELINE FOR GENE EXPRESSION PROFILES ANALYSIS TO PREDICT PHYSICAL CONNECTIONS THROUGH THE BRAIN REGIONS

Author:
ILARIA ROBERTI

Supervisors:
PROF. ELISA FICARRA
PHD. GIANVITO URGESE
ENG. MARTA LOVINO

A thesis submitted in fulfillment of the requirements for the degree of Master Doctor of Biomedical Engineering

in the

DAUIN
DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING

DECEMBER 2018

Declaration of Authorship

I, Ilaria ROBERTI, declare that this thesis titled *'A pipeline for gene expression profiles analysis to predict physical connections through the brain regions '* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"The starry sky above me, the moral law within me"

[I. Kant]

Abstract

Ilaria ROBERTI

A pipeline for gene expression profiles analysis to predict physical connections through the brain regions

The aim of this thesis is to perform an integrative analysis of gene expression and connectivity data using machine learning techniques. The implemented algorithm is able to detect the connection's degree between brain areas exploiting patterns of gene expression profiles. Brain is a complex machine made up of more than 100 billion neurons grouped in many functional regions that communicate to each other through electro-chemical signals. When referring to brain, physical connectivity is meant as a pattern of anatomical links constituted by the neuron's axons and connected to the dendrites of post-synaptic neurons. It is shown in literature that functional properties of neurons and neuronal systems depend on neural connectivity patterns. Due to the crucial role played by brain anatomical connectivity, scientists created and made available a rising number of maps modeling axonal connections between brain regions. Due to ethical issues, available human data are not detailed enough to enable a significant and complete analysis, therefore whole mouse connectome is studied and a lot of models have been designed. These connectivity models have enabled scientific community to build investigation methods to detect the existence of anatomical neural connections, features and correlations between intrinsic properties in the brain tissue. In this regard, the leading trend is the hypothesis that gene expression influences physical brain connectivity patterns at anatomical level. Gene expression is the process by which information from a gene is used in the synthesis of a function gene product such as proteins. Gene regulation gives the cell control over structure and function, and is the basis for cellular differentiation, morphogenesis and the adaptability of any organism. Gene expression can be evaluated for each gene in a sample and represents an index of the gene's activity. Many genes have shown spatial patterns of expression in restricted brain areas. During this thesis, I implemented a pipeline based on a machine learning technique able to learn from gene expression data unexplored connectivity's existence and intensity.
Materials:
In this study, I used gene expression and connectivity data available on the Allen Mouse Brain Atlas (AMBA) resources combined with the connection's intensity reported on Brain Architecture Management System (BAMS) database.

Allen Institute Mouse gene expression data consist of whole-brain in situ hybridization (ISH) data that have been obtained from 24 m sections of 56-day old C57BL/6J mice. For each gene, expression data are provided as grid data, a 3D matrix representing mouse brain volume. Each item of the matrix is a voxel, that stores gene expression quantified as energy level. Gene expression data are provided at 200m resolution. The Allen Mouse Brain Connectivity Atlas comprises axonal projections labeled by rAAV that is a viral tracer injected in a specific site and detected through two-photon tomography for more than 200 mouse brains' regions in coronal section. When the viral tracer is injected in a brain region, called source region, it produces axonal projections in several target regions. In Allen Mouse Brain Connectivity database, more than one injection site can be found for a region. Likely gene expression, connectivity grid data is available for each injection site. Each element of the 3D matrix is a voxel that contains a value representing the projection energy level detected. Connectivity grid data are provided at 100m resolution. In the reference space created by AMBA to model the mouse brain, each cerebral region is composed by several voxels. With the purpose to link grid data voxels to a region, AMBA provides a structural annotation file at different resolutions. Other information about neural circuitry were collected from BAMS. To date, BAMS contains on the order of 45,000 reports of connections between different gray matter regions in the rat. Even if AMBA's data were retrieved from mouse, some studies have demonstrated that mouse and rat's brains share anatomical features but at different dimension scales. BAMS provides an interactive matrix that reports for each couple of brains' regions the existence of a connection and the intensity of this connection.

Method:

The first phase of the work was preliminary data downloading step. I have obtained grid data for 3318 genes and 2333 injection sites for coronal section from Allen Brain Atlas. I stored the whole data in a SQlite database processed through a pipeline designed in Knime dataflow framework, an open-source platform that enables to perform elaborations and to organize data in an easy accessible model. Projection data were reconducted to an unique value representing the connectivity's intensity between the source and the target region. This to overcome the impossibility to match the volumes provided by ABA at different resolutions without introducing a significant error. The connectivity value was obtained calculating the median of all the projection energy values for each source-target couple. When multiple injections were provided for a region, the connection's intensity between couples was evaluated as the mean or maximum of the median values obtained for each injection situated in a given region. As declared early in this document, the main goal of this thesis is the creation of a model capable to recognizing the level of connection of two areas by looking at the gene expression profiles. In order to achieve this

purpose, N source-target regions are selected in accordance with their connectivity intensity and the type of analysis to perform. These data undergo the following pipeline:

i) For each source-target pair, I selected on the gene data annotation M voxels belonging to source region and M voxels to the target region.

ii) For each selected voxel, I created a vector composed of 3318 elements where each element corresponds to a specific gene provided by AMBA. In particular, each array's cell contains the gene expression energy detected for that voxel. At the end of this step, I obtained M vectors representing source's gene expression profile and M vectors representing target's gene expression profile.

iii) I created the dataset selecting P combinations among all possible combinations of source and target voxels. In particular, for each combination I concatenated the gene expression vector corresponding to the source voxel with the gene expression vector corresponding to the target voxel. Therefore, the dataset will be made of P vectors.

iv) In the end, I assigned a label representing the source-target connectivity value to each of previously created combinations.

I repeated these steps for all the N source-target regions selected at the beginning. Sequentially, the whole vectors representing gene expression profiles for the selected source-target regions and their labels are used as dataset to feed a neural network. Evaluated the nature and complexity of the performed analysis, I implemented a Multilayer Perceptron (MPL) that is a class of feedforward artificial neural network used for data classification and regression. Thus, the dataset composed by source-target vectors and their labels is divided in training, validation and test set to feed the MPL. More in detail, training set accounts for the 70% of the whole data and test set the remaining 30%. The validation set is made up selecting 10% of pairs in the training set.

Results:

MLP was requested to accomplish different tasks in order to test two main hypotheses. Initial experiments consisted in a binary classification between connected and unconnected regions. In order to achieve this result, seven source-target pairs have been selected. In particular, four pairs were chosen among the ones that on BAMS are reported with maximum connection's intensity and other 3 pairs were chosen with minimum connection's intensity (corresponding to unconnected condition on BAMS). I obtained the dataset processing the original data through the before mentioned pipeline. To perform the binary classification, I assigned label "1" (codifying connected regions) to the source-target vectors obtained by pairs with maximum intensity on BAMS, otherwise I assigned label "0" (codifying unconnected condition).

The trained model has shown accurate performances suggesting that a strong correlation between gene expression and axonal connectivity exists. More in detail, accuracy, recall and F1_score reached 1.0 value on the test set after few epochs. Verified this first hypothesis, I designed a second set of experiments to test if gene expression profiles can contain enough information to predict the intensity of connections between regions. To achieve this result, I built a MLP to perform the regression task. For this purpose I created another dataset following the above mentioned pipeline. The dataset was created selecting only one source region and some of its multiple targets with different connection's intensity. In this case, I assigned the connectivity value obtained from AMBA as the mean(medians) to the label representing the connection's intensity between source and target. As before, I trained MLP model on the 70% of the dataset and tested the 30%. The regression produced real output values representing connection's intensity that have been revealed to be accurate. In fact, encouraging results have been observed reporting a mean squared error $(1,47 \pm 0,000187) * 10^{-5}$ and mean absolute percentage error of $(5,72\% \pm 0,0040\%)$ on the test set. Since the model has proved to be promising, further experiments focused to the generalization to a wider dataset composed of cortex and cerebellum's data (approximately 58 regions). The analysis performed on the wider dataset was composed of two phases and these differentiate in the choice of the connectivity value assigned to each Source-Target combination. In fact, mean and maximum of the medians of the projection energies of each connectivity experiment were assigned respectively in phase 1 and phase 2. The first phase has outlined the limitations of the regression model on a dataset composed of gene expression profiles measured for many different regions. Instead, the predictions obtained by the multiclass classification highlighted a low precision on the class of the regions connected through a strong intensity. In light of this results, I have hypothesized that the mean of the medians was not representative of the connectivity status between two regions. Therefore, in the second phase I have assigned as connectivity value to each Source-Target vector the maximum of the medians. A multiclass task was performed subdividing the dataset in three classes: unconnected, weakly and strongly connected classes. The choice of the connectivity value, calculated as maximum(medians), has proved to be crucial in the improvement on the precision and recall on the strongly connected regions with F1_score 0.74. This value is affected by a 0.67 recall on the strongly connected class. Despite of the recall on strongly connected regions, almost the totality of misclassified vectors were attributed to the class of weakly connected regions. In light of this, a last classification experiment with two classes, econding connected and unconnected conditions, have been designed. This has proved on the available data that analyzing gene expression profile of Source-Target region combination is possible to recognize the connected pairs with a 0.84 F1_score. The purpose of this thesis is to predict connection's intensity for the couples of brain

regions for which BAMS matrix does not report complete connectivity data. The complete matrix is obtained analyzing gene expression and connectivity's energy data provided by AMBA. The trained model can be used to individuate the gene patterns affecting the connectivity networks. This consists in a feature selection executed through wrapper-based approaches. The presence of many structural similarities and genetic homologies between mouse and human brain can allow scientists to use the obtained table as a connectivity reference. This reference table constructed on the information retrieved from energy levels represents a promising means to perform analogues analysis exploiting gene expression data coming from RNA-seq experiments instead of the energy data provided by Allen Brain Atlas.

Acknowledgements

Gli anni trascorsi a Torino sono stati i più intensi e duri della mia vita. Sento di essere partita senza idea di quello che avrei voluto fare e diventare. Per me, la laurea è sempre stata solo una tappa di un lungo percorso di cui mi era ignota la destinazione. Ora, grazie a quello che ho appreso in questi due anni, mi è più chiaro qual è il percorso che voglio intraprendere. Non credo di esagerare nello scrivere che oltre alle esperienze vissute, soprattutto le persone a me vicine hanno contribuito alla mia formazione, seppur in maniera diversa.

In primis, vorrei ringraziare la professoressa Ficarra non solo per avermi dato la possibilità di lavorare a questa tesi ma per avermi mostrato il lato umano dell'università. Al mio primo arrivo qui a Torino, le sue lezioni di bioinformatica mi hanno confermato di aver scelto la specialistica giusta per i miei interessi.

Non meno importante è stato il contributo di Gianvito che mi ha proposto questo lavoro e mi ha seguito fino alla sua conclusione. Mi ha sempre spronato a non demoralizzarmi di fronte agli imprevisti e lo ringrazio per questo.

Non potrei non ringraziare Marta che si è sempre resa disponibile all'ascolto. Le sue spiegazioni concise hanno avuto la capacità di concretizzare molti concetti, per me astratti, appresi durante il mio percorso universitario. Grazie per la pazienza dimostratami.

Ringrazio tutti i ragazzi del laboratorio 4 dove ho lavorato a questa tesi per diversi mesi. Forse sarebbe più appropriato scusarsi per aver più volte intasato il server.

La mia crescita personale è stata segnata dalla presenza e, nell'ultimo anno e mezzo, dalla mancanza della mia sostenitrice più accanita, mia madre. So che non leggerai mai queste righe ma meriti i miei ringraziamenti. Se sono riuscita ad affrontare questo percorso prima e dopo la tua perdita è solo grazie al coraggio che mi hai trasmesso con il tuo modo di affrontare qualsiasi difficoltà. Hai sempre messo al primo posto la mia serenità e i miei studi, sacrificando silenziosamente te stessa. La forza che hai dimostrato fino all'ultimo istante mi ha motivato, non volevo deluderti. Sei il modello che tutti dovrebbero avere nella propria vita.

Essenziale è stata la presenza di Michele che mi ha sorretto nei momenti difficili e mi ha sempre dimostrato stima nei momenti di sconforto. E' la persona che ha più sofferto a causa della mia scelta ma anche quella che ha gioito di più dei miei successi. Lo ringrazio.

Ringrazio mio padre che non mi ha mai negato la possibilità di realizzarmi, nonostante le incomprensioni. Se non fosse per lui, non avrei potuto intraprendere questo percorso.

In questi anni, mio fratello Paolo mi ha sempre aiutato a sdrammatizzare momenti altrimenti pesanti. Nonostante le continue polemiche, ho sempre saputo di poter contare su di lui e questa sicurezza mi ha dato forza.
La mia vita scolastica, universitaria e personale è legata a Milena la cui presenza è stata fondamentale in questi anni soprattutto nei momenti difficili. Mi sono sempre sentita a casa con lei, la considero come una sorella. Grazie.
In questi due anni, la mia famiglia fuorisede ha accolto anche Irene, la mia amica e coinquilina. Non la ringrazierò mai abbastanza per le parole di conforto e di incoraggiamento che hanno saputo spesso riportarmi alla realtà quando mi ero persa nelle mie elucubrazioni.
Ricorderò per sempre le risate con Irene e Milena in quella cucina di Via Montevecchio 9.
L'Università ha incrociato il mio percorso con quello di altre due ragazze, Marilena e Giorgia, con le quali ho portato avanti i progetti più significativi del mio percorso. Le voglio ringraziare per avermi sempre mostrato fiducia. Spero che la conclusione di questa parentesi non ci faccia perdere.
Voglio ringraziare la mia amica di una vita Silvia e la mia amata cugina ed amica Paola per avermi sempre accolto con affetto quando tornavo a casa. Grazie per avermi supportato nei momenti difficili.
Infine, vorrei ringraziare tutta la mia famiglia, in particolare le mie Zie Liliana e Pina, per avermi aiutato in modo diverso nei momenti tristi e difficili che ho dovuto affrontare quest'ultimo anno.

Contents

List of Figures

List of Abbreviations

ABA	**A**llen **B**rain **A**tlas
AMBA	**A**llen **M**ouse **B**rain **A**tlas
API	**A**pplication **P**rogramming **I**nterface
BAMS	**B**rain **A**rchitecture **M**anagement
CA	**C**onnectivity **A**nnotation
FISH	**F**luorescent **i**n **s**itu **H**ybridization
GEA	**G**ene **E**xpression **A**annotation
MLP	**M**ulti **L**ayer **P**erceptron
NN	**N**eural **N**etwork **S**ystem

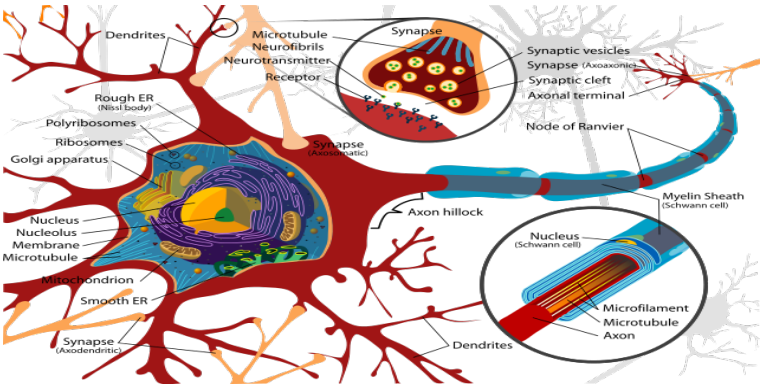
Dedicated to my mother…

Chapter 1

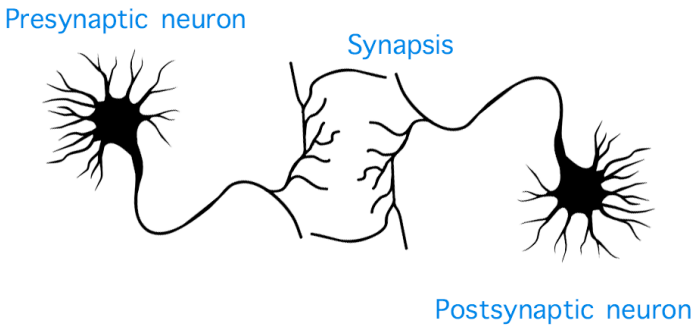
Introduction

1.1 Brain Connectivity

The brain is an organ that serves as the center of the nervous system in all vertebrate and most invertebrate animals. It is encased in the head with the advantage of direct short connections with the sense organs localized in or on top of the head (olfaction, taste, vision, audition, vestibular sense). The intimate relation of the brain to the sense organs points to the brain's essential role as an information handling device.
The brains of all species are composed primarily of two type of cells: neurons and glial cells. Glial cells (also known as glia or neuroglia) can be of different typed, and perform several essential functions, including structural support, metabolic support, insulation, and guidance of development. However, neurons are usually considered the most important cells in the brain. The property that makes neurons unique is their capability to send signals to specific target cells over long distances.
Neuron (fig:1.1.A) is star shaped cell composed of a cell body from which two type of nerve fibers emanate. The neuronal body is called *soma* and it is seat of the cellular nucleus and organelles. Here, the DNA is transcripted in RNA whose genetic sequence is used to produce different type of proteins. Neuronal soma receives and sends elettro-chimical signals to other neurons through nerve fibers. Dendrites carry the neural signal toward the neuronal body. Instead, axons carry trains of signal pulses called action potentials to distant target cell of the brain or body [11][27]. Axons transmit signals to other neurons by means of specialized junctions called synapses (fig:1.1.B). A single axon may be connected to thousand other cells' axons through synaptic connections. [12].
The physical connections that link numerous group of neurons constitute a network. This represents the anatomical brain connectivity which is a pattern of physical links that enable interactions between neurons. Thus, connectivity plays crucial roles in determining the functional properties of neurons and



(A) Complete neuron cell diagram[14]



(B) Pre/Postsynaptic neuronal connections [3]

FIGURE 1.1: Neurons

neuronal systems. This has long attracted the attention of neuroanatomists that dedicated their studies to connectomics. This is the field of science dealing with the assembly, mapping and analysis of connectome [24].

Despite the intense effort that has gone into elucidating the structure of neural systems, many connections results still unexplored for all the species. Many studies focused in discovering factors that may affect physical connectivity. In this regard, connectivity is constituted by fibers that propagate from the neuronal bodies. These,in turn, are the seat of the nucleus and all the nuclear component that contribute with their activity to the cellular differentiation and morphogenesis. According to this evidence, many scientists have assumed that the main factors influencing connectivity's patterns have to be researched at cellular scale. Thus, the leading trend is the hypothesis that the cellular activity influences physical brain connectivity patterns at anatomical level. On top of that, the analysis of neuronal gene expression profiles may represent a means to understand connectome more in deep.

1.2 Gene expression profiling

Gene expression is the process by which information from a gene is used in the synthesis of a function gene product such as proteins. Gene regulation gives the cell control over structure and function, and is the basis for cellular differentiation, morphogenesis and the adaptability of any organism. The complete set of transcripts in a cell is called transcriptome. Understanding the transcriptome is essential for interpreting the functional elements of the genome and revealing the molecular constituents of cells and tissues [28]. Gene expression can be evaluated for each gene in a sample and represents an index of the gene's activity.
Gene expression profiling is the measurement of the activity (the expression) of genes. Many experiments of this sort measure an entire genome simultaneously, that is, every gene present in a particular cell. Several transcriptomics technologies can be used to generate the necessary data to analyze through DNA amplification techniques. Sequence based techniques, such as RNA-Seq, provide information on the sequences of genes in addition to their expression level. However, these techniques extract gene expression profiling through bulk analysis of tissue samples. This way, the gene expression information obtained is only an averaged profile among the cell's population in a sample. Thus, it is not possible to detect the variation among the members of the population through this techniques.
Technological advances allowing the precise measurement of single-cell transcriptional states overcome the limitation imposed by bulk analysis. These techniques are called Single-cell RNA-seq (scRNA-seq)[17] and rely on separation of single cells from tissue by enzymatic or mechanical dissociation resulting in loss of the spatial information. Single cells are then subjected to downstream analysis, providing analyzed results that can then be linked to the spatial localization in the original tissue. However, the contextual information is limited because only the target cells are analyzed but not their surrounding neighbor cells that form the micro environment of the target cells [22].
These limitations pushed to develop new techniques able to save the spatial information. In *in situ* techniques, the species of interest is detected in its location in individual cells. In fact, through these techniques it is possible to sequence nucleic acids directly in cells and tissue. [22]. An effective *in situ* technique is the fluorescence *in situ* hybridization (FISH or ISH). This approach allows to detect a specific RNA and DNA sequence in a tissue. In fact, it uses RNA or DNA complementary hybridization probes labeled to fluorescent molecules. Once the probes has hybridized the target in the fixed tissue, the transcript can be localized and quantified through fluorescence microscopy images. It is possible to detect multiple targets at once using different fluorophores (fig:1.2). In particular, fluorescence *in situ* hybridization permits to save both spatial and morphological information. Futhermore, FISH produces good-quality images

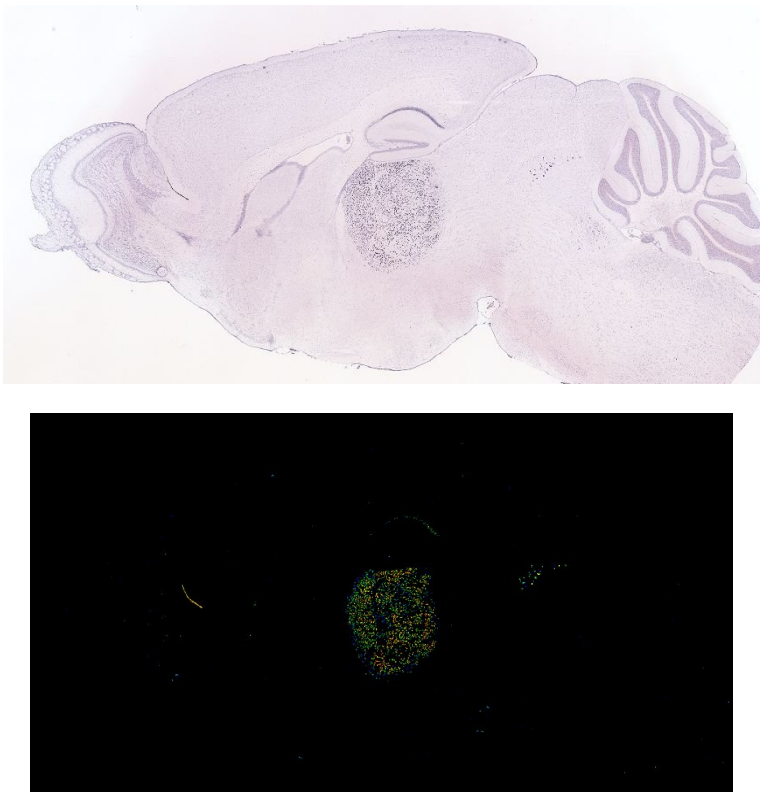


FIGURE 1.2: ISH image result for Plekhg1 gene in mouse's Thalamus from Allen Mouse Brain Atlas.

that are easier to interpret than the ones produced with others *in situ* techniques [17].

1.3 Thesis layout

This thesis aims to implement a pipeline able to predict axonal connections between brain regions. This is obtained analyzing the gene expression profiles detected in those brain's areas.
First of all, in Chapter 2 the computational resources used in this work are presented. The first part of the chapter is dedicated to machine learning techniques focusing on the fundamental notions to implement a Multilayer Perceptron. Subsequentially, some knowledge about Sqlite database architecture and the potentials of an open-source data processing tool, are given.
Chapter 3 offers an overview about the main data resource, Allen Brain Atlas. After a brief historical excursus, the informatic pipeline designed by Allen Brain Science Institute to retrieve, elaborate and organize connectivity and gene expression data will be presented.

Chapter 4 is dedicated to the detailed presentation of the method designed in this work. The first part of the chapter illustrates the steps executed in the data downloading procedure. In this section, gene expression and connectivity data nature will be clarified. Subsequently, the *p*ipeline implemented ad hoc for the construction of the datasets, is described focusing on the steps executed. In the last part of the chapter, the datasets, used to perform different experiments in this work, will be presented focusing on the description of the brain areas selected and computational means used.
The result produced in the will be discuss in Chapter 5.

Chapter 2

Technical Background

2.1 Machine Learning

Machine learning algorithms analyzes data in the same way humans do: learning from experience. These techniques figure out natural patterns in examples and use them to make previsions on unknown data. During the training, machine learning algorithms extract information from raw data and represent it in some type of model. This is used, in the testing, to deduct information about unknown data that have not been analyzed yet.
When the algorithm is presented with inputs and their desired outputs for training, the machine learning method is called supervised. Supervised learning uses classification and regression techniques to develop predictive models. Classification's goal is to build a model that is able to assign a given input to a class (label). Otherwise, the output of the model is a continue value when a regression is performed. Deep learning is a particular supervised machine learning's category and refers to the learning mode characterizing Neural Networks (NN). These are computational models inspired by animals' brain behavior and its attitude to retrieve knowledge from experience. Similar to how animal brains work, in Neural Networks many simple units work in parallel with no centralized control unit. Each network's node is connected to another one through weighted links. These connections are the primary means of learning and information's storage process in Neural Networks. In fact, NN learn how to analyze data updating connections' weights. A specific category of NN is represented by Multilayer Perceptron that is a powerful means to accomplish classification and regression tasks.

2.1.1 Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) is a class of feedforward artificial neural network used for data analysis. The main architecture is composed by many

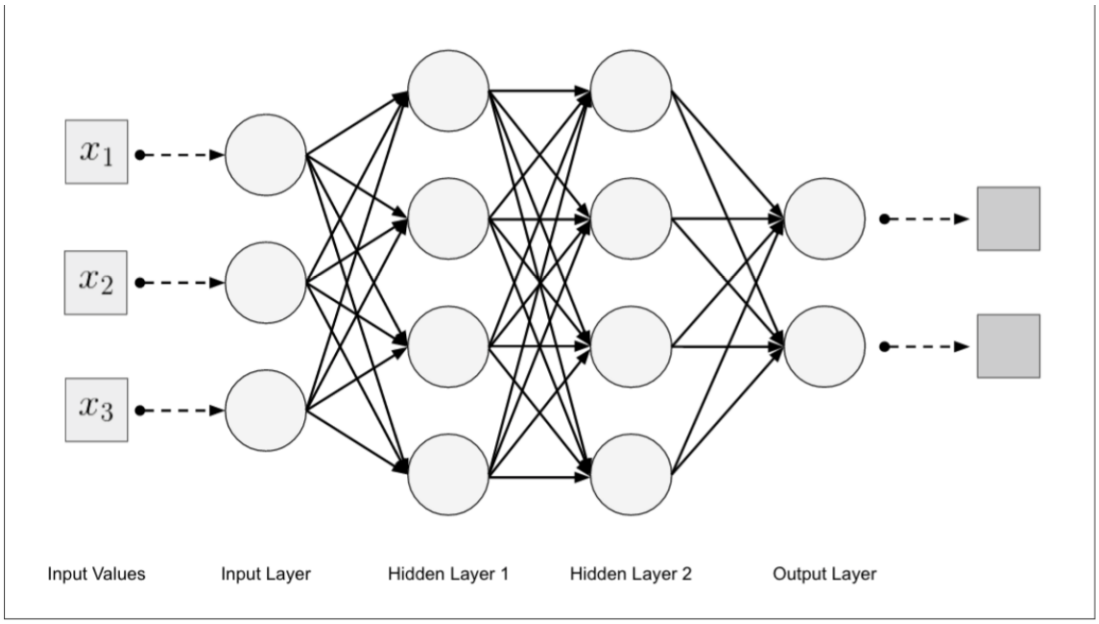


FIGURE 2.1: Fully connected multilayer feed-forward neural network topology

layers, at least three: an input layer, one or more hidden layers and an output layer. Each layer comprises a certain number of nodes depending on the network's complexity and it is fully connected to the adjacent one. The connections between the neurons in the layers form an acyclic graph (fig:2.1). The nodes of Neural Networks model brain's neurons and, except of the first layer, each layer includes nodes characterized by a non-linear activation function that maps the weighted inputs to the output of each neuron. MLP utilizes a supervised learning technique called backpropagation in order to adjust the weights of the connection between nodes. The artificial neuron takes an input that, based on the weights on the connections, can be ignored (by a 0.0 weight on an input connection) or passed on to the activation function. The activation function also has the ability to filter out data if it does not provide a non-zero activation value as output. One full training cycle on the training dataset is called epoch.

2.1.2 Backpropagation

The process by which Neural Networks learn to recognize data patterns is called backpropagation. This uses an optimization method to adjust the weights of the connections in a neural network and minimize the error on the output of the network. At the start of the training process, weights are initialized randomly. During a forward-propagation, the network propagates the input

pattern from layer to layer until the output pattern is generated by the output layer. All the computational steps performed by each neuron of each layer in order to transform the input x_i into the output y_i can be summarized by the equation:

$$y_i = F(\sum_i w_{ij} x_i + b_j) \tag{2.1}$$

Where w_{ij} is the weight that links the i-th input to the j-th neuron, b_j is the bias added by that neuron to the input, and F() is the activation function computed by the j-th neuron.
Once the output of each hidden layer is produced, the output layer is reached. In order to evaluate the error, the overall output o and the desired output, called target t, are provided to a loss function[1.2].

$$loss = l(o, t) \tag{2.2}$$

Thus, the gradient of the output o, with respect to the input, is computed through the chain rule of differentiation in order to determine the contribution of each parameter to the resulting loss. The partial derivatives of the loss with respect to the weights are computed, obtaining each component as

$$\frac{\partial l}{\partial w_{ij}} = \frac{\partial l}{\partial o} \frac{\partial o}{\partial y_i} \frac{\partial y_i}{\partial w_{ij}} \tag{2.3}$$

The network weights are modified in order to decrease the loss at the next iteration. The parameter values are updated following the opposite direction of the gradient and accordingly to a learning step ∎ which defines how big the changes to make must be. In general, it is preferred proceeding with small changes a time. This approach of making changes iteratively depending on the direction of a negative gradient is called gradient descent. Parameter update is made for all the weights at the same time.

2.1.3 Activation Functions

Activation functions are scalar-to-scalar functions, yielding the neuron's activation. There are different type of activation functions that can be more suitable to accomplish some task than others. It is important to choose the activation function based on the type of data and task required to the NN. In fact, specific activation functions allow to perform regression or classification.

Linear

A linear transform is basically the identity function where the dependent variable has a direct, proportional relationship with the independent variable (fig

2.2.a):

$$y_i = w_i * x_i \tag{2.4}$$

w_i is the neuronal connection's weight, x_i is the input, y_i is the output that was producted by the i-th node.

Sigmoid

Sigmoids can reduce extreme values or outliers in data without removing them. A sigmoid function is a machine that converts independent variables of near infinite range into simple probabilities between 0 and 1, and most of its output will be very close to 0 or 1 (fig 2.2.b).

Softmax

Softmax is a generalization of logistic regression in as much as it can be applied to continuous data (rather than classifying binary) and can contain multiple decision boundaries. It handles multinominal labeling systems.

Rectified Linear

Rectified linear activates a node only if the input is above a certain quantity (fig 2.2.c). While the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship with the dependent variable:

$$f(x) = max(0, x) \tag{2.5}$$

Tanh

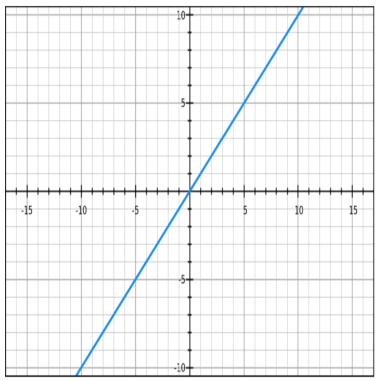
Tanh is a hyperbolic trigonometric function. Just as the tangent represents a ratio between the opposite and adjacent sides of a right triangle, tanh represents the ratio of the hyperbolic sine to the hyperbolic cosine:

$$tanh(x) = sinh(x)/cosh(x) \tag{2.6}$$

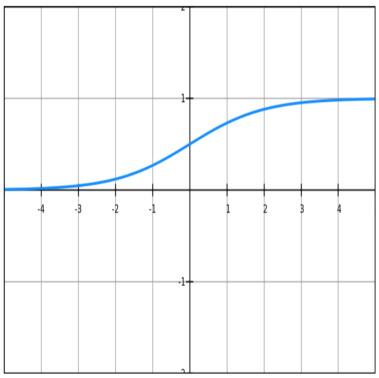
Unlike the sigmoid function, the normalized range of tanh is –1 to 1. The advantage of tanh is that it can deal more easily with negative numbers (fig 2.2.d).

2.1.4 Loss Function

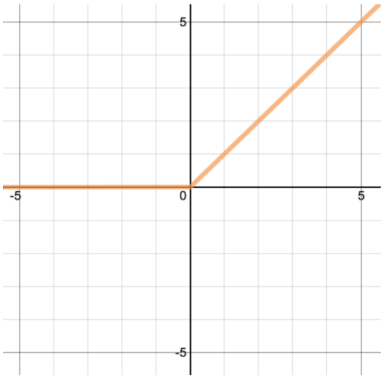
Loss functions quantify how close is the output predicted by the NN to the desired output. To this purpose, a metric based on the observed error in the network's prediction is calculated. These errors are aggregated over the entire



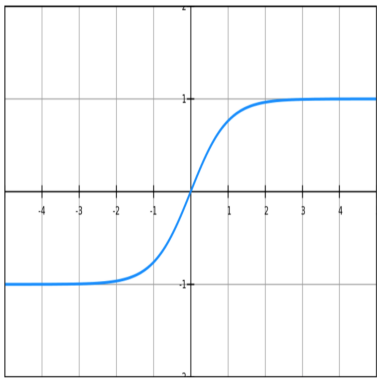
(A) Linear



(B)



(C) Rectified Linear



(D) Tanh

FIGURE 2.2: Activation functions

dataset and averaged. The single number obtained represents how close the neural network is to the desired output. Then, it is necessary to find the set of parameters that minimize the error. This way, loss functions help reframe training neural networks as an optimization problem.

Loss functions for regression

When dealing with regression problems, a widely used loss function is mean square error (MSE). The error in a prediction is squared and is averaged over the number of data points. However MSE is particulary sensitive to outliers. Thus, in some cases it is suggested to use mean absolute error loss that simply averages the absolute error over the entire dataset.

Loss function for classification

Cross-entropy loss is commonly-used in binary classification (labels are assumed to take values 0 or 1) as a loss function, which is computed by

$$Cross - entropy = -\frac{1}{m}\sum_{i=1}^{m} y_i * log p_{model}(y_i|x_i;\Theta) \tag{2.7}$$

where p_{model} is the deep neural network model to estimate the p_{data} distribution, x_i is the example drown from the p_{data}, is the distribution of network parameters and y_i the label of the x_i example. A particular case of cross-entropy is called binary cross-entropy and it is used in case of two classes classification.

2.1.5 Optimization

The process of adjusting weights to produce more accurate predictions about the data is known as parameter optimization. This aims to reduce the error calculated by the loss function. One of the most used optimization method is Gradient descent. This is a way to minimize the loss function F(θ) dependent by a set of model's parameters θ. Gradient descent minimize the error updating the parameters in the opposite direction of the gradient of the function $\nabla_\theta(F(\theta))$. The steps taken to reach a local minimum is determinated by the leaning rate η. Gradient descent has three main variants that differ each other in how much data is used to calculate the gradient of F(θ).

Batch gradient descent

Batch gradient descent (BGD), also known as Vanilla gradient descent, calculate the cost function for the entire training dataset. This way only one update is performed. This method is slow and can not be indicated for big dataset that would not fit in memory.

Stochastic gradient descent

Stochastic gradient descent (SGD) updates parameters for each training example. It is very fast optimization method even if it performs many updates resulting in significant fluctuations of the objective function. These fluctuations enables the SGD to jump to new better local minimum.

Mini-batch gradient descent

Mini-batch gradient descent (MBGD) takes the best of both method metioned before. In fact, MBGD updates parameters for every mini-batch of n training examples. This results in a reduction of fluctuations that characterize SGD. Common mini-batch sizes range between 50 and 256. However, MBGD does not provide an automatic learning rate adjustment during the training process. For this reason adaptive optimization methods have been implemented in order to overcome the learning rate choice's issue.

Adagrad

It adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters. In order to minimize the error, Adagrad uses a different learning rate for every parameter θ_i.

RMSprop

RMSprop is an unpublished, adaptive learning rate method proposed by Geoff Hinton. RMSprop, as Adagrad, uses a different learning rate for each update. At each iteration, the learning rate is divided by an exponentially decaying average of squared gradients.[23]

2.1.6 Hyperparameters

The performance of Neural Networks are affected by parameters that can be freely chosen. Obviously, those parameters deal with controlling optimization function and have been selected in order to minimize the error calculated by the loss function. They fall into several categories and characterize the model.

Network Size

The size of the model depends on the number of layers that constitute the NN and the number of nodes comprised in each layer. The number of nodes for hidden layer is affected by the problem's complexity and can be chosen arbitrarily; whereas the number of input and output layer's nodes depend on the feature of the input vector and the output class desired, respectively.

Learning Rate

The learning rate influences how much fast the vector of parameters have to be changed during the training in order to minimize the error. An high learning date permits to reach the best set of parameters rapidly but the pass might be so large to miss the best parameters set. On the contrary, a small learning rate might make the algorithm too slow in solving the required task.

Dropout

Dropout is a computationally inexpensive way of regularization during model training by removing units from the network. In hidden layers, dropout is done with a probability of 0.5. By randomly omitting neurons co-adaptation can be prevented among detectors, which helps drive better generalization in models on held-out data.

Momentum

Momentum helps the learning algorithm get out of spots in the search space where it would otherwise become stuck[11]. .

2.1.7 Typical Dataset

In general, the goal of machine learning technique is to build a mathematical model able to make predictions on unknown data. To achieve this purpose, the algorithms are trained to recognize pattern on data learning from them. A typical dataset is composed by raw data, such as gene expression and connectivity levels. Usually, these data undergo a normalization process in order to obtain values that belong to the the activation functions' domain. In deep learning supervised techniques, data used to build the final model is splitted in three datasets:

- **Training dataset**: examples used to fit the model in order to adjust the connections' weights of the neural networks. It consist of input data and their desired output (labels). It counts an higher number of elements than test and validation sets.
- **Validation dataset**: this dataset is provided to the model already fitted on the training set. The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the model's hyperparameters[2]. Usually it derives from the training set and it is only a little fraction of it.
- **Test set**: is used to test the final model's performance. Test dataset is provided without his labels and consist of data that the network has never seen during the training phase. It is smaller than training dataset.

All the datasets have the same distribution and are independent each other. Notice that is important to avoid the overfitting phenomenon that occurs when the classifier begins to memorize data pattern that are specific on the training dataset and loses its capability to generalize. In this case, the performance on the test set are much lower than on training set. To avoid overfitting, a validation set is used to perform a regularization technique by *early stopping*: stop training when the error calculated on the validation dataset increases, as this is a sign of the overfitting to the training dataset[21].

2.2 SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. It reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform, it is possible to freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. SQLite strives to provide local data storage for individual applications and devices. SQLite emphasizes economy, efficiency, reliability, independence, and simplicity. Raw data can be imported from CSV files, then that data can be sliced and diced to generate a myriad of summary reports. More complex analysis can be done using simple scripts written in Tcl or Python (both of which come with SQLite built-in) or through the use of tools such as Knime. Many bioinformatics researchers use SQLite in this way[25].

2.2.1 SQlite Architecture

The architecture of SQLite is composed by four main units (fig:2.3):

- **Core**: contains user interface, the SQL command processor, and the virtual machine. The user interface consists of a library of C functions and structures to handle operations such as initializing databases, executing queries, and looking at results. Function calls that execute SQL queries use the SQL command processor. The command processor functions exactly like a compiler. When executing a program, the virtual machine directs control flow through a large switch statement, which jumps to a block of code based on the current opcode.

- **SQL compiler** contains a tokenizer, a parser, and a code generator. When a string containing SQL statements is to be executed, the interface passes

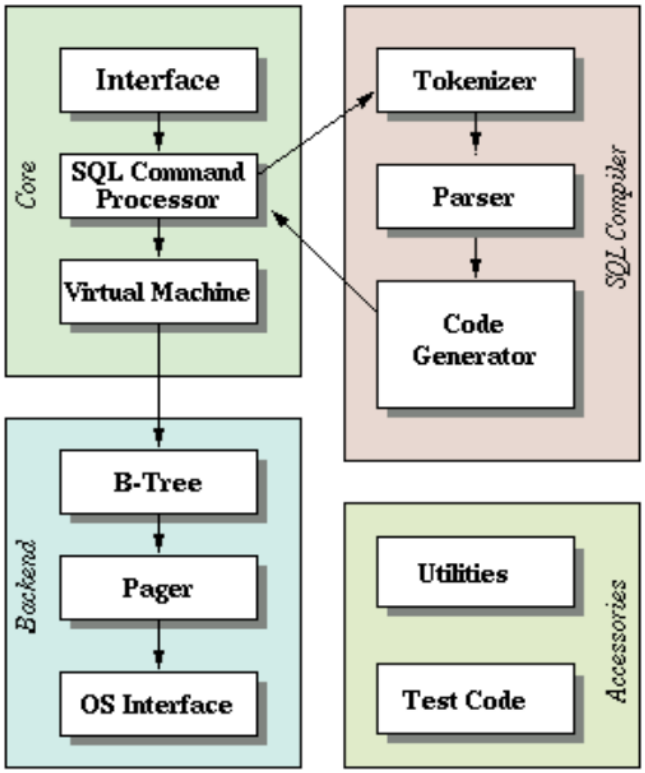


FIGURE 2.3: SQlite's architecture

that string to the tokenizer. The job of the tokenizer is to break the original string up into tokens and pass those tokens one by one to the parser. The parser is the piece that assigns meaning to tokens based on their context. The parser for SQLite is generated using the Lemon LALR3 parser generator. After the parser assembles tokens into complete SQL statements, it calls the code generator to produce virtual machine code that will do the work that the SQL statements request.

- **Backend** contains B-Tree, Page Cache, OS Interface: An SQLite database is maintained on disk using a B-tree implementation found in the btree.c source file. A separate B-tree is used for each table and index in the database. All B-trees are stored in the same disk file. The B-tree module requests information from the disk in fixed-size chunks. The page cache is responsible for reading, writing, and caching these chunks. In order to provide portability between POSIX and Win32 operating systems, SQLite uses an abstraction layer to interface with the operating system.

- **Accessories** contains Utilities and Test code: SQLite provides some utility related functionality such as memory allocation and case less string comparison routines are located in util.c more than half the total code base of SQLite is devoted to testing[26].

2.2.2 Knime

KNIME allows users to visually create data flows (or pipelines), selectively execute some or all analysis steps, and later inspect the results, models, and interactive views. KNIME is written in Java and based on Eclipse and makes use of its extension mechanism to add plugins providing additional functionality. The core version already includes hundreds of modules for data integration (file I/O, database nodes supporting all common database management systems through JDBC or native connectors: SQLite, SQL Server, MySQL, PostgreSQL, Vertica and H2, data transformation (filter, converter, splitter, combiner, joiner) as well as the commonly used methods of statistics, data mining, analysis and text analytics. Visualization supports with the free Report Designer extension. KNIME workflows can be used as data sets to create report templates that can be exported to document formats like doc, ppt, xls, pdf or stored in databases [1].
KNIME boasts a rich catalogue of processing nodes that have been used in this thesis work in order to process and organize data in an easy accessible SQlite database. In the following, the nodes used mainly will be presented.

I/O nodes

- **Excel Reader (XLS)**: this node reads a spread sheet and provides it at its output port. It reads only data from one sheet at the moment. It can read only numeric, date, boolean and string data but, of course, no diagrams, pictures, or other items.It reads in the data from the sheet and sets a type for all columns that is compatible with the data in that column (in the worst case "String" covers all).
- **CSV Reader**: Reads CSV files. Use this node if the workflow is used in a server or batch environment and the input files structure change between different invocations. In particular, this includes a variable number of input columns. Upon executing the node will scan the input file to determine number and types of the columns and output a table with the auto-guessed structure.

Manipulation nodes

- **Join**: This node joins two tables in a database-like way. The join is based on the joining columns of both tables.

Filter nodes

- **Column/Rows filter:** This node allows columns/rows to be filtered from the input table while only the remaining columns/rows are passed to the output table. Within the dialog, columns can be moved between the Include and Exclude list.
- **GroupBy:** Groups the rows of a table by the unique values in the selected group columns. A row is created for each unique set of values of the selected group column. The remaining columns are aggregated based on the specified aggregation settings. The output table contains one row for each unique value combination of the selected group columns.

Database nodes

- **Database Reader** Establishes and opens a database access connection to read data from.
- **Database Writer:** Establishes and opens a database access connection to which the entire input table is written to.
- **SQlite Connector:** This node creates a connection to a SQlite database file via its JDBC driver. You need to provide the path to the file [13].

Chapter 3

Allen Brain Atlas

"The Allen mouse brain Atlas" and "The Allen mouse Connectivity Atlas" are projects within the Allen Institute for Brain Science which aims to generate a gene expression and connectivity mouse brain's 3D maps through the combination between neuroanatomy and genomics. The institute's founder was Paul G. Allen who devoted 100 million dollars to this purpose in September 2003. An initial release of The Allen Mouse Brain was the first project to be made publish in 2004. Subsequently, more data has been added to the first project's version and many other brain atlases have been published. From its birth, Allen institute has released seven brain atlases: Mouse Brain Atlas, Human Brain Atlas, Developing Mouse Brain Atlas, Developing Human Brain Atlas, Mouse Connectivity Atlas, Non-Human Primate Atlas and Mouse Spinal Cord Atlas. Each brain atlas focuses on its own project and it is built by a selected team of researchers. These groups have collected and produced brain scans, medical data, genetic information and psychological data with the purpose to construct a mouse brain 3D map. In order to obtain data, several different techniques have been used. One technique involves the use of postmortem brain in brain scanning technology to discover where in the brain genes are turned on and off. Then *in situ hybridization* is used to individuate gene expression patterns as *in situ hybridization* images.
Even though the majority of research has been done in mice, 90 per cent of genes have a counterpart in humans. Thus, the atlas based on mouse is a powerful means to model and study human diseases.

The main goal and motto for Allen Brain Institute is "fueling discovery". As matter of the fact, their atlases have the purpose to provide to the scientific community the means needed to study mouse brain's functional and anatomic connections. In order to archive this purpose, the atlases are open source and data are freely available to everyone for consulting and downloading. Thus, any scientist is able to read all the information stored in Allen Brain Atlas's site and retrieves knowledge from them while designing their experiments. The Allen Brain Atlas lets researchers view the areas of differing expression in the brain which enables the viewing of neural connections throughout the brain

[19][15].

In this thesis, gene expression and connectivity data were collected from Allen mouse Brain Atlas and Allen mouse Brain Connectivity Atlas respectively.

3.1 Allen Mouse Brain Atlas

The Allen Mouse Brain Atlas represents an integration between genome and neuroanatomic mouse brain data. It is a complete genome-wide, high-resolution atlas of gene expression throughout the adult mouse brain. It is composed by different sections and tools that enable an easy data's navigation and analysis. Gene expression patterns are available as images obtained by in situ hybridization technique. The Atlas contains ISH data for approximately 20,000 distinct mouse genes in 56 day old male "black" mice (P56).

The informatic pipeline designed by Allen Brain Science Institute to retrieve informations from ISH images implements the following units [Fig.3.1]:

- **Processing Module**: this unit balances white and intensity normalize the image to get an higher quality visualization. This step is followed by a global adaptive thresholding method to obtain a separation of the background and foreground and then a morphological filtering and connected component analysis steps are applied to the ISH image in order to remove noise and connect broken segments.

- **Detection**: is applied to each ISH image to create a grayscale mask identifying pixels in the high-resolution image that correspond with gene expression. The grayscale intensity represents the average ISH signal within a connected area. For Web presentation, the intensity is color-coded to range from blue (low expression intensity), through green (medium intensity) to red (high intensity).

- **Expression Gridding Module**:creates a low-resolution 3-D summary of the gene expression and project the data to the common coordinate space of the 3-D reference model to enable spatial comparison between data from different specimens.

- **Structure unionizer module**:computes expression statistics for each structure delineated in the reference atlas by "unionizing" grid voxels with the same 3-D structural label. Then, the results of this operation are used in the web application to display expression summary bar graphs for each structure and for each image.

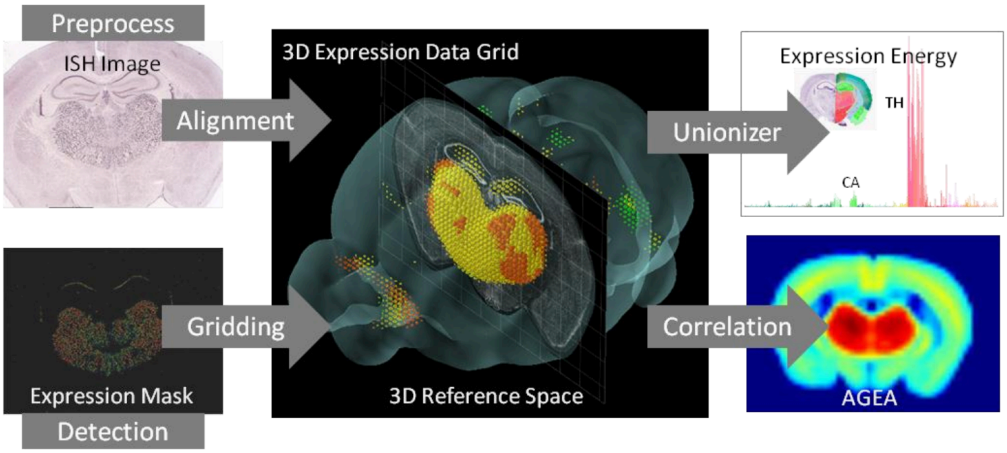


FIGURE 3.1: Mouse Brain Atlas Pipeline[4]

- **Alignment module**: operates on a per-specimen basis where all image series from a specimen are combined as one series. Based on maximization of image correlation, the module interleaves reconstructing the specimen as a consistent 3-D volume with co-registration to the 3-D reference model. Once registration is achieved, information from the 3-Dreference model can be transferred to the reconstructed specimen and vice versa. The resulting transform information (a 2-D affine transform per image and 3-D affine transform per image-series) is saved in the database to support the image synchronization feature in the Zap viewer and generation of grid-level gene expression summaries.

- **Expression Grid Search Service**: it is a grid search service that has been implemented to allow users to instantly search over the 25,000 image series to find genes with specific expression patterns.

- **Anatomic gene expression Atlas**: AGEA is a relational atlas that allows users to explore spatial relationships in the adult mouse brain based on the expression patterns of 4,000 genes, which comprise the set of coronal image-series in the Allen Mouse Brain Atlas[4].

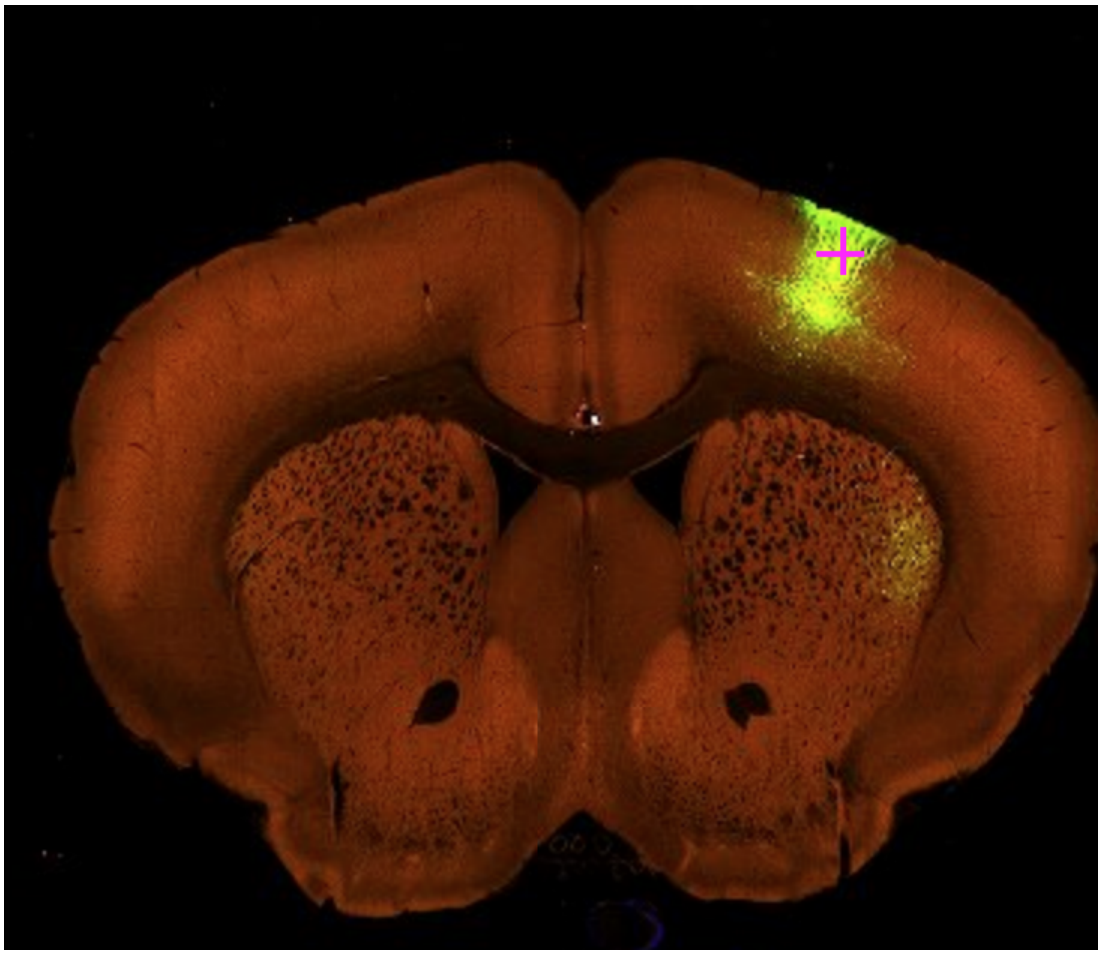


FIGURE 3.2: Projection's signal detected for an injection situated in Primary Visual Area

The output of the pipeline is quantified expression values at a grid voxel level and at a structure level according to the integrated reference atlas ontology.

3.2 Mouse Brain Connectivity Atlas

Mouse Brain Connectivity Atlas is a 3D map of neural connections in the mouse brain, built on an array of transgenic mice genetically engineered to target specific cell types. It consists of numerous 2D images corresponding to a mouse brain's slice that can be visualized side-by-side in a 3D reference space. Allen Brain Institute provides powerful tools that enable the 3D visualization and spatial search of connectivity data.

In order to obtain neural projections [Fig.3.2], a specific region per mouse brain

was chosen to be injected with a green fluorescent protein (EGFP) as an anterograde viral tracer (AAV). EGFP labeled axonal projections and images were obtained by using two-photon microscopy[16]. More than one injection was used for larger regions. Subsequently, intrinsic signal imaging data (ISM) have been added to the two-photon images to permit a more accurate individuation of visual areas. Allen Brain Institute has implemented an informatic data processing pipeline [Fig.3.3], similar to the one designed for gene expression data, that allows an integration of the connectome's information derived by two-photon images and ISM data. This algorithm consists of different processing steps:

- **Preprocessing**: scanned image tiles are stitched to form a single high-resolution image in a three step process that consists of tile positioning using a calibration matrix, intensity correction, and tile transition blending.
- **the Alignment module**: registers each projection image to the common coordinates of a 3D reference model
- **Projection Detection Module**: separates signal from background through three main steps (intensity rescaling and noise reduction, tissue region segmentation, projection signal segmentation).
- **The Gridding module**: summarize the detected signal for each data set and report it in a common coordinate space.
- **Grid Search Service**: it is a services that implement an intuitive search of the desired experiment. User can use different types of search' functions that allows to find specific projection depending on the information needed. For instance, it is possible to visualize all the exeperiments for a couple of source region (where the injection is placed) and target region (where a projection's signal is detected)[10].

3.3 The Allen Common Coordinate Framework

The Allen common coordinate framework (CCF) allows connectivity and gene expression data mapping on the mouse brain, quantification, visual presentation and investigation [fig.3.4]. Allen brain institute has released many CCF's versions since the first one was made public in 2005. The latest version (v3) was uploaded on the site in 2017 to support the new products available in the connectivity's section.
To create the CFF, an anatomical template of the mouse brain was obtained by the shape and background signal intensity average of 1675 specimens from the Allen Mouse Brain Connectivity Atlas. Averaging many mouses' brains,

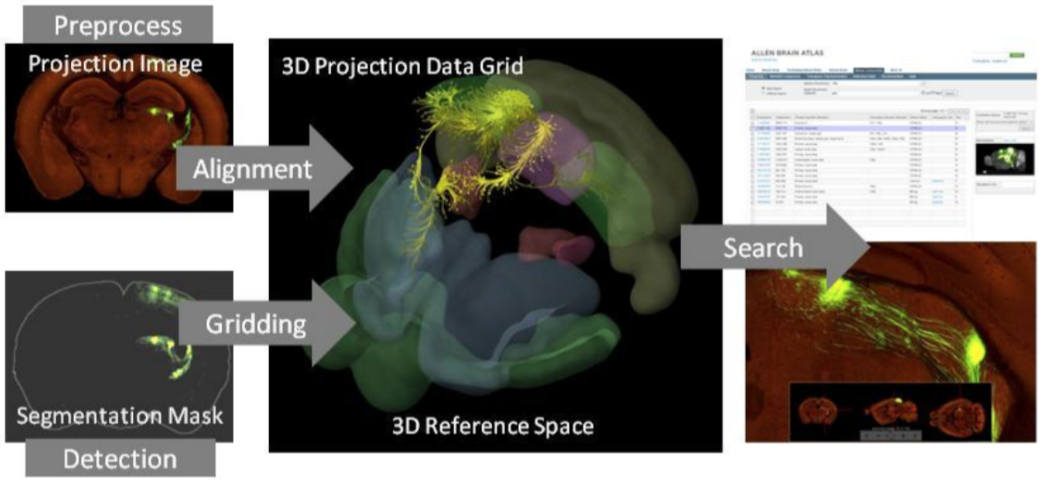


FIGURE 3.3: Connectivity informatic pipeline [10]

Allen experts generated a population average. Thus, the anatomical template is the average shape and average appearance of the population of 1675 specimens and shows remarkably clear anatomic features and boundaries for many brain structures. Many reference data sets were used to individuate anatomical structures in the created template. For the 3D annotation, manual delineation of the anatomical template was a combined process of structure discovery and 3D illustration carried out at various levels: as individual structures, group of local structures and interface between groups. Using anatomical template contrast features from select supporting (connectivity data), structures were land marked, filled in serially, and validated. In certain cases the process was modified to include previously drawn structures. Once a critical mass of content was reached, a global merging of all individual and local structure groups was performed. The process was completed by a final evaluation of structures in the component 2D plates as well as the rendered 3D composition. The final CCF product consists of 662 annotated structure volumes, including gray matter, white matter and ventricles. Overall, 242 cortical and 330 subcortical gray matter, 82 fiber tracts,and 8 ventricle and associated structure volumes were delineated natively in 3D[9].

3.4 Grid Data

3.4.1 Gene expression Grid Data

The aim of the Gridding module, is to create a low-resolution 3-D summary of the gene expression and project the data to the common coordinate space of

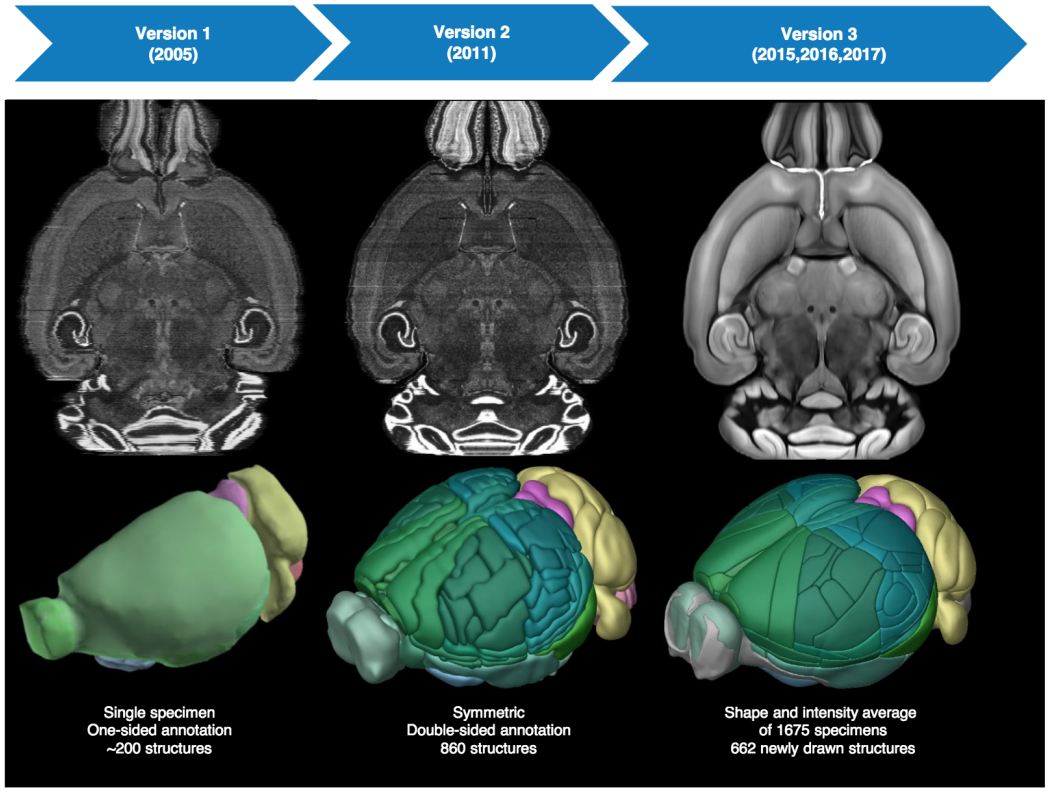


FIGURE 3.4: CCF versions [9]

the 3-D reference model. This enables spatial comparison between data from different specimens.
The Gridding module operates on a per image-series basis. Each image is divided into a 200µm x 200µm grid. For each division, pixel-based statistics of the sum of the number of expressing pixels and sum of expressing pixel intensity were collected. From these statistics measures were obtained for:

- **expression density** = sum of expressing pixels/sum of all pixels in division
- **expression intensity** = sum of expressing pixel / sum of expressing pixels
- **expression energy** = sum of expressing pixels intensity / sum of all pixels in division

In the previous step, the Alignment module computes the transforms that rotates each 2-Dimage to form a consistent 3-D volume per specimen. Each per-image 2-D expression grid is smoothed and rotated to form a 3-D grid. Finally, z-direction smoothing is applied to the 3-D grid which is then transformed into the standard reference space[4].

3.4.2 Projection Grid Data

The aim of the Gridding module, which is the last unit of connectivity informatic pipeline, is to create an isotropic 3D summary of projection data and resample the data to a common coordinate space to enable spatial comparison between data from different specimens. The Gridding module operates on a per specimen basis. Images are divided into 10µm x 10µm grids. In each division, the sum of the number of detected pixels and the sum of detected pixel intensity were collected. A second set of summations was computed for regions identified as belonging to the injection site for injection site quantification. The 2D per-image grids were combined to form 3D grids of resolution 10m x 10m x 100m. Using the transform parameters computed by the Alignment module, each 3D grid was transformed to the standard 10m isotropic reference space using linear interpolation to generate sub-grid values [10]. From these statistics measures were obtained for:

- **projection density** = sum of detected pixels/sum of all pixels in division
- **projection energy** = sum of detected pixel intensity / sum of expressing pixels
- **injection fraction** = fraction of pixels belonging to manually annotated injection site

- **injection density**= density of detected pixels within the manually annotated injection site

The data grids were used for downstream search and analysis, and were further processed as computational paths for viewing in the Brain Explorer 2 3D viewer, alongside the Allen Mouse Common Coordinate Framework.Grids at 10, 25, 50 and 100 m isotropic resolution are also generated and are available for download through the Allen Brain Atlas Application Programming Interface (API)[5].

3.4.3 SectionDataSet

In Allen Brain Atlas data portal, experiments' information about conditions and results are organized in records called SectionDataSet. Each SectionDataSet is identified by a unique id and stores the grid-data. For each experiment, the gene expression and projection grid-data can be viewed directly as 3-D volumes in the Brain Explorer 2 3-D viewer, alongside the 3-D version of the Allen Reference Atlas. Furthermore, SectionDataSets are used for downstream search and analysis. In fact, they are freely downloadable through API service for both data type.
ISH's experiments measure only one gene's expression per time. Sub sequentially, each SectionDataSet contains the expression profile of a single gene as grid-data and information about the gene and experiment's conditions.
For connectivity data, instead, each experiment is associated to the tracer's injection site. In the record, along with the projection data detected, the injection's region is reported. Through API, it is possible to search for specific SectionDataSet IDs using experiment's conditions as filters. The unique ID that identifies a SectionDataSet is required to download the grid-data stored in it.

Chapter 4

Method

This thesis aims to implement a complete pipeline to perform enabling the collection, organization and processing of gene expression and connectivity data from mouse brain. In this chapter, the procedures designed and executed to achieve this purpose will be illustrated in detail. An overall view of the main steps composing the pipeline is shown in figure 4.1. This represents only a guide to the lecture of the chapter.

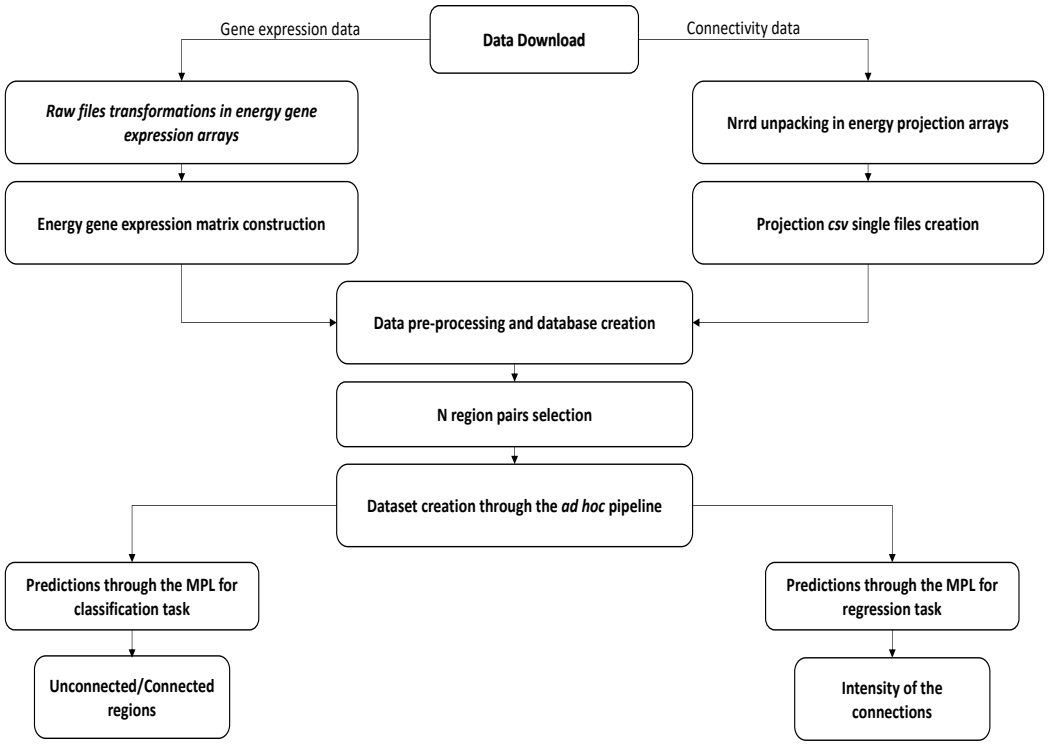


FIGURE 4.1: Flowchart of the implemented pipeline

4.1 Downloading data phase

In this work, two different type of data have been used to perform an integrative analysis. Specifically, gene expression of the adult mouse brain have been retrieved from Allen Mouse Brain Atlas (AMBA); whereas connectivity data have been collected from Allen Mouse Connectivity Brain Atlas (AMCBA) and Brain Architecture Management System (BAMS).
For both type of data, Allen Brain Atlas (ABA) provides data as grid-data at different resolutions. These are the result of the elaborations performed by the Gridding Module implemented by ABA. This unit creates a 3D summary of the gene expression or connectivity data and resamples them to the common coordinate space of the 3D brain model. For each connectivity and gene expression experiment, a grid data is downloadable through API service. All the queries, used to download data from AMBA and AMCBA, were composed through the RMA BUILDER that is made accessible on Allen Brain Atlas API's section[8]. In addition to the gene expression and connectivity grid-data, ABA provides a structural grid-data annotation at the same resolution of the other data. This allows to link mouse brain's voxels to anatomical structures in the Common Coordinate Framework.
In the downloading phase, the two datasets have been organized in data structures depending on the type of data.
In this work, data were retrieved from AMBA and AMCBA in July 2018. In October 2018, connectivity experiments have been added on AMCBA that haven't been used for this work.

4.1.1 Gene expression grid-data

Allen Institute Mouse Gene expression data consist of whole-brain *in situ* hybridization data that have been obtained from 24 µm sections of 56-day old C57BL/6J mice. The detected expression levels have been transformed by a pipeline in energy, density or intensity levels and then reported as grid-data. Data are available for coronal and sagittal sections. Although sagittal section counts more than 20,000 genes, connectivity data are available only for coronal section. Furthermore, coronal section has a better coverage of the mouse brain. On top of that, only coronal experiments have been taken in account.

In AMBA, the expression profile of each gene throughout the mouse's brain is associated to SectionDataSet, a record in which all the experiment's information are stored.
To download the grid-data, a query was built to retrieve the SectionDataSet IDs for gene expression experiments through the API service. This returns an XML containing all gene expression SectionDataSet IDS annotated on AMBA for coronal section. Thus, 3318 SectionDataSet IDS corresponding to 3318 gene

expression grid-data have been obtained.
The SectionDataSet IDS retrieved were annotated in a list to be used in the creation of 3318 queries. Each of them was used to download a gene expression grid-data.
To obtain the gene expression grid-data, the presented steps were performed for each of the 3318 genes:

i) a query declaring SectionDataSet ID and the volume type (energy) desired has been built through the RMA BUILDER.

ii) A request has been sent to the server through the constructed query obtaining an archive (.zip) as response

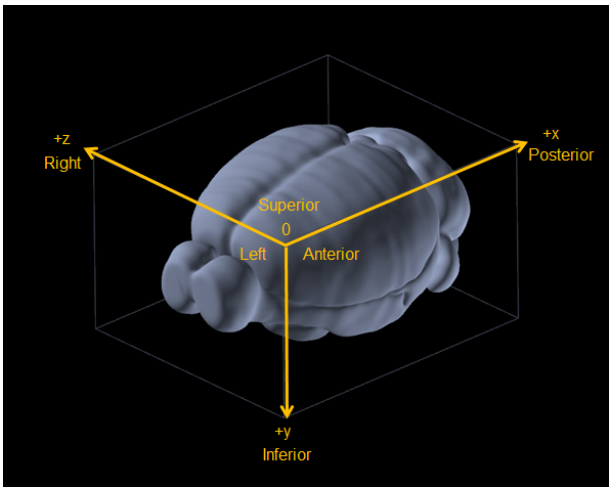
iii) *Energy.mhd* and *energy.raw* files have been extracted from the archive.

These steps produced an *energy.raw* file for each of 3318 gene expression experiments.
The *energy.raw* file contains a vector of 159'326 elements. Each of these is a voxel that corresponds to a 3D element of the mouse brain model. In fact, the grid-data has a shape of a volume, specifically 67x41x58 at 200 µm resolution, but it is provided packed into a 1-D numerical array, stored in the *energy.raw*. However, ABA provides references to reconstruct the volume starting from the array (fig.4.2). The values in the array are the expression energy levels detected for a specific gene.
At end of this procedure, 3318 arrays of 159'326 elements, containing gene expression's energy levels, were obtained. These vectors have been transferred in a 3318x159'326 matrix. Subsequently, this matrix underwent few steps of processing and had been organized in a database created through Knime.

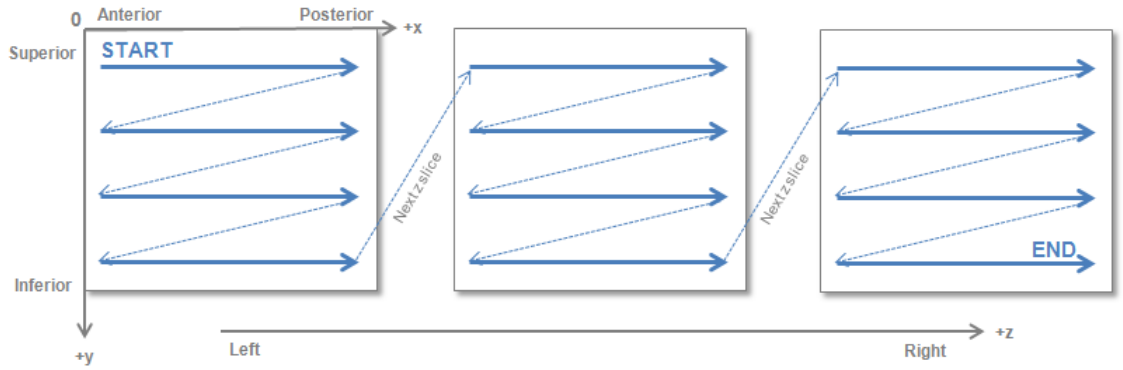
4.1.2 Connectivity data

Allen Connectivity grid-data

The Allen Mouse Brain Connectivity Atlas comprises axonal projections labeled by rAAV and detected through two-photon tomography for more than 200 mouse brain's regions in coronal section. On AMCBA, injection sites refer to the spots where the viral tracer is injected. The region where a certain injection site is placed is referred as source region; whereas the brain structures where the injection produced axonal projections is defined target region. Due to the restricted dimensions of the mouse brain, it may be possible that some injections involved more than one region. To distinguish this two conditions, injection sites situated in a single region are defined primary. Instead, secondary injections involve more regions. AMCBA provides a number of primary injection experiments adequate to build a significant dataset. On top of that, only primary injections were analyzed in this work.



(A) 3D Volume



(B) Packing's criteria

FIGURE 4.2: The common reference space is in PIR orientation where x axis = Anterior-to-Posterior, y axis = Superior-to-Inferior and z axis = Left-to-Right [5]

Connectivity data are available at different resolutions (10, 25, 50, 100 µm). In this work only 100 µm resolution was taken in account as it is the closest to the one provided for gene expression data (200 µm).

Each primary injection site corresponds to a SectionDataSet, which stores the experiment's conditions and the detected projections as grid-data.
To download the grid-data, query was built to retrieve the SectionDataSet IDs for injection experiments through the API service.This returns an XML containing 2333 primary injection SectionDataSet IDS annotated on AMCBA for coronal section. These were annotated in a list and subsequently used to create 2333 queries. Each of them was used to download a connectivity grid-data.
More in detail, the presented steps were performed for each of the 2333 injection sites:

i) a query declaring SectionDataSet ID and the volume type (projection energy) and a integer expliciting the resolution desired has been built through the RMA BUILDER.

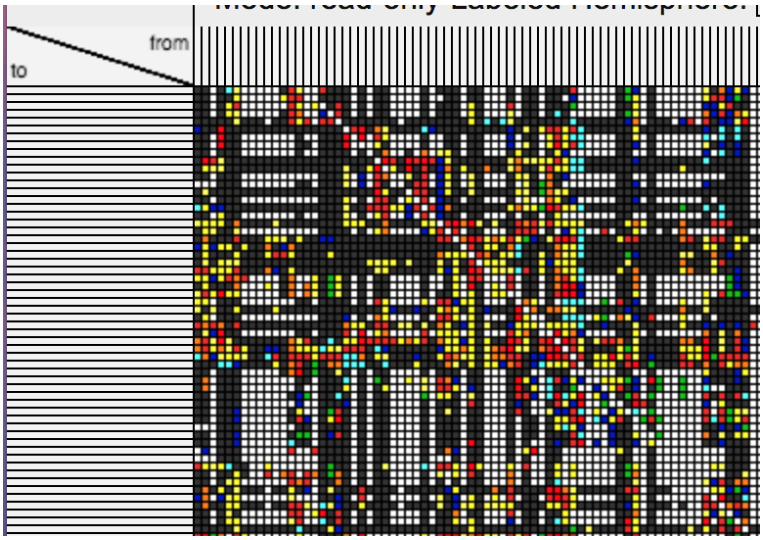
ii) A request has been sent to the server through the constructed query obtaining a single 32-big floating *NRRD* file as response

Requesting data for all the SectionDataset IDs retrieved, 2333 files .Nrrd were obtained. Each of them represents the axonal projections' distribution produced by a specific primary injection site throughout the mouse brain.

For connectivity experiments, the 3-D volumetric grid-level data at 100 µm are provided packed in a 3D numerical array *.nrrd* with shape 132x80x114 (fig: 4.2.a). Each element of this 3D array is a voxel that composes the mouse brain model. It contains projection energy level produced by the injection site, to which the grid data refers, and detected for that point of the mouse brain. Keeping the spatial reference provided by ABA, the 3D array was unpacked in a 1D array of length 1'203'840 (fig: 4.2.b). This way, 2333 1D arrays were create as many injection sites and *nrrd* files. The voxels with projection energy equal to 0 have been filtered out from each array. Due to the vectors dimension, the 2333 arrays containing energy were stored in single *csv* along with the source region indication.

Brain Architecture Management System (BAMS)

Other neural circuitry data were collected from BAMS to be used as connectivity reference. To date, BAMS contains on the order of 45,000 reports of connections between different gray matter regions in the rat. Even if AMBA's data were retrieved from mouse, some studies have demonstrated that mouse and rat's brains share many anatomical features but at different dimension scales. BAMS provides an interactive matrix (fig.4.3) that reports for each couple of brains' regions the existence of a connection and its intensity for the rat's brain



(A) Submatrix extracted from the 496x496 matrix of BAMS

(B) Detail of BAMS interactive matrix

FIGURE 4.3: Each element of the matrix define a connection between the two regions reported in column and row. Different colours have been used to codify connection's intensity. In particular, white colour represents unknown connections[20].

regions. This matrix was built collecting from the literature already known and demonstrated connections. In this matrix, the strengthen of the connection is codified by a value belonging to the range [1,9]. Therefore, the intensity of the connection grows with the increase of the value reported in the matrix. For unknown connections, a 0 was assigned. Furthermore, regions are reported with a universal acronyms enabling the matching with ABA's region annotation. The matrix can be freely consolidated and downloaded as an *xlsx*

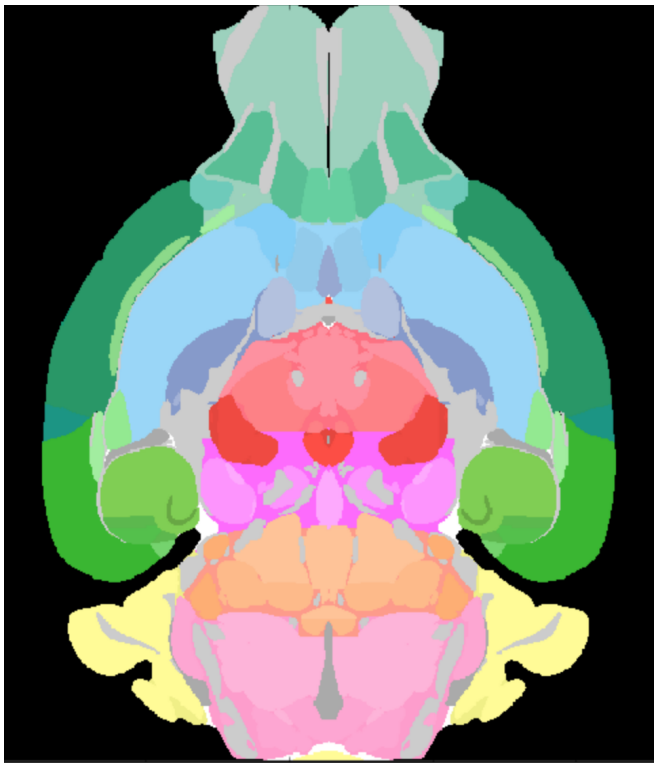


FIGURE 4.4: Annotation's slice [6]

from the site of BAMS [20].

4.1.3 Structural annotation file

An annotation volume is a 3D raster image that segments the reference space into structures. These are composed of a number of voxels depending on the dimensions and the model's resolution taken in account. In the annotation volume, each voxel is assigned to an integer value that describes the structure to which it belongs [6]. Brain structures in Allen reference spaces are arranged in trees. The leaf nodes of the tree describe the very fine anatomical divisions of the space, while nodes closer to the root correspond to gross divisions [6]. The annotation file reports regions' IDs with the finest anatomical division's detail (fig.4.4). Gene expression and connectivity data are registered to one of several common reference spaces. With the purpose to link each data's voxel to the membership brain region, ABA provides a structural annotation file at different resolutions. In fact, the i-th annotation element allows the matching between the i-th voxel in the data array and its membership brain structure.

As for the other gene expression data, the gene expression annotation (GEA) is provided at 200 µm resolution. It is a 1D array that counts the same number of

voxels that compose array containing the gene expression data (about 159k). Similarly, the connectivity annotation (CA) is provided at 100 µm. As for the connectivity data, it has been reshaped in a 1D array of length 1'203'840.

However, primary injection structures annotation is not at the finest annotation's level. Then, a procedure to trace back both annotation to the same detail's level was implemented.
The purpose is to regroup all the finest regions nested in the belonging rougher region. In order to achieve this, ABA provides a structure graph, a list of dictionaries documenting brain structures and their containment relationships. This reports the depth at which each region can be found. In the graph, finest region annotations correspond to higher depths. The structure id path enables to determinate the route to follow to trace back finest regions to rougher structures. Combining this two information, it was possible to construct a dataframe reporting for each region the belonging structure at different depths. This was used to group and trace back voxels, assigned to finest regions in the grid data annotation, to coarser regions.

4.2 Implementation choices

Connectivity and gene expression grid-data are provided at different resolutions. Connectivity data are available at 100 µm unpacked in a array of 1'204'840 voxels. Gene expression data, instead, are provided at 200 µm corresponding to an array of 159'326 voxels.
This means that the data volume is composed of different numbers of 3D fundamental elements that correspond to the voxels. In fact, a voxel in the gene expression data volume should match to 8 voxels in the connectivity one. Thus, the information in the connectivity data volume that is subdivided in a major number of voxels, is finer. On top of this, an alignment of the two volumes was required to analyze the connectivity and gene expression information detected in specific point of the mouse brain.
However, due to the data inconsistency, it was not possible to overlap the two volumes through a resampling process (fig. 4.5).
First of all, gene expression volume counts approximately 9'000 spare voxels that do not correspond to any voxel of the connectivity data.
Furthermore, another possible issue was the assignation of the group of voxels located across two or more regions to a single region of the resampled volume. Thus, in order to not introduce significant error, the connectivity data was aggregate per region. Hence, the connection between Source-Target regions is associated to a unique value that represents the connectivity intensity. The aggregation methods used to obtain a unique value is the median of all the energies detected for a Source-Target pair in a specific experiment.

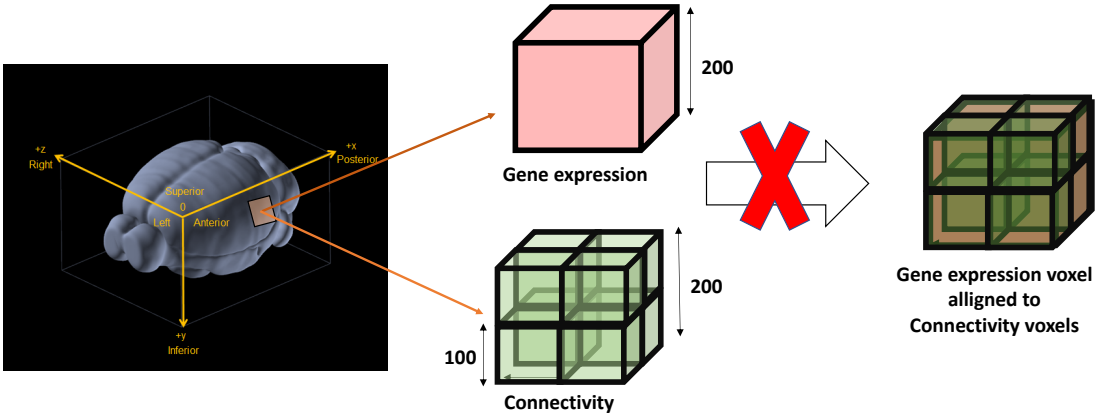
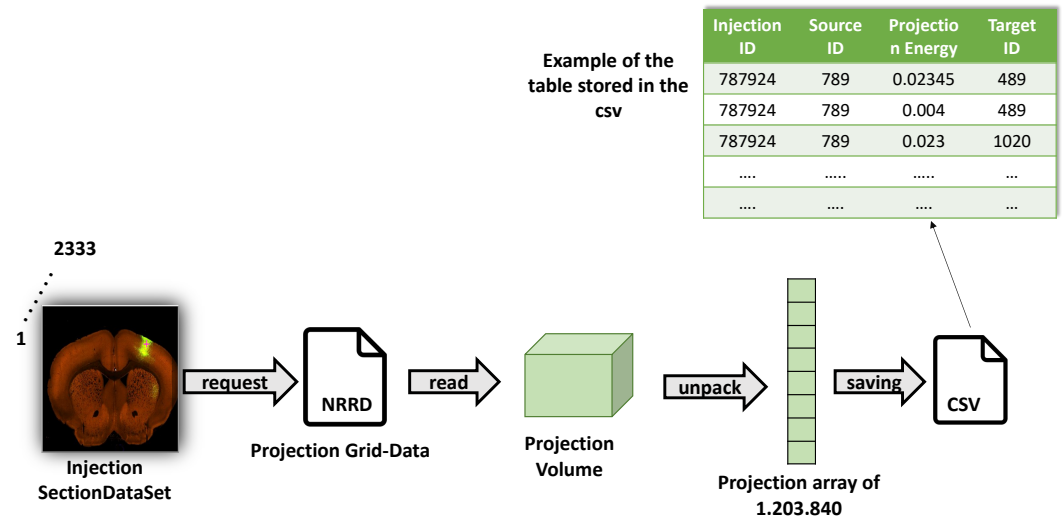


FIGURE 4.5: Representation of the two data volumes

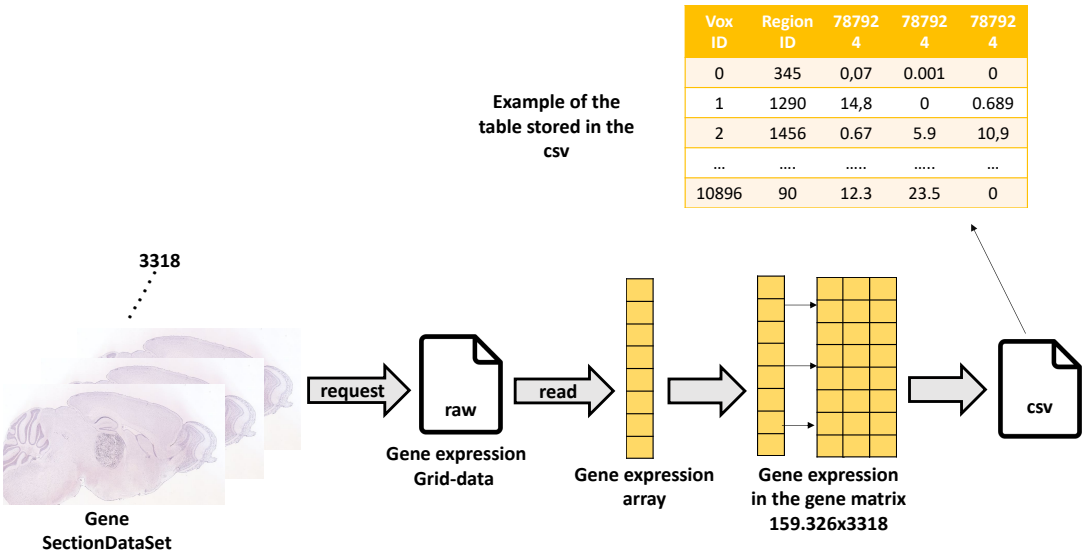
4.3 Database creation

At the end of the collecting data phase, grid data for 3318 genes and 2333 injection sites have been obtained for coronal sections.
Gene expression data have been organized in a single matrix 3319x159'326 that have been stored in a *csv* file. In particular, 3318 columns correspond to genes, whereas rows are associated to a specific voxel that composes the 3D volume. An additional column reports the region annotation for each voxel (fig. 4.6.A).
Connectivity data, instead, have been saved in 2333 singles *csv*. Each file is associated to an injection site and stores the array obtained at the end of connectivity's downloading phase. This array is composed by 1'203'840 elements filled with projection energy detected for the specific voxel. Along with connectivity data, two region annotations have been inserted. A column reports the **Source region** annotation which is the structure of the injection site. Since each file stores data for only one injection site, a unique id is reported in the Source column. Another column, instead, reports the membership region annotated for the voxel. This corresponds to the **Target region** produced by a specific injection site (fig. 4.6.B)

Both connectivity and gene expression data have been stored in a database to allow an easy and faster access to the data. The database created has an SQlite architecture and has been filled with data processed through Knime.



(A) Elaborations of connectivity data data



(B) Elaborations of gene expression data data

FIGURE 4.6: Data elaboration flows

This is an open-source platform that enables elaborations of data through computational nodes. A workflow has been designed to process gene expression and connectivity data obtained in the previous phase. The processed data have been organized in tables designed with the purpose to access data more quickly and easily.

4.4 Knime workflow

The workflow designed in Knime to process connectivity and gene expression data is composed by three main sections (fig. 4.7). In turn, each section is composed by several nodes that apply different elaborations on data. First of all, the loading block read connectivity and gene expression data from the *csv* files. After this, data is sent to a processing block where undergo several elaborations. In the end, the output tables, produced in the previous section, are saved in the SQlite database by the writing block.
Each section of the workflow will be presented focusing on the elaborations executed.

FIGURE 4.7: Blocks in Knime's workflow

Data Loading

The nodes in figure 4.8 compose the reading section that loads the data to be processed.
On the bottom, gene expression matrix has been red using a **CSV Reader** node. This has produced a 3318x159'326 table whose columns correspond to voxels and rows to genes.
The series of nodes, used to load connectivity data, are shown on the top of the workflow. Starting from left:

1. the **List Files node** creates a list with the locations of the injections files contained in a specific folder. Therefore, it has produced a table whose rows are the 2333 paths to as many *csv*. This table is sent in input to a **Variable Loop Start** node.

2. This uses each row of a data table in input to define new variable values for each loop iteration. Therefore, it sends a single path per time to the next nodes. The elaborations executed by these nodes are repeated for all the paths in the input table.

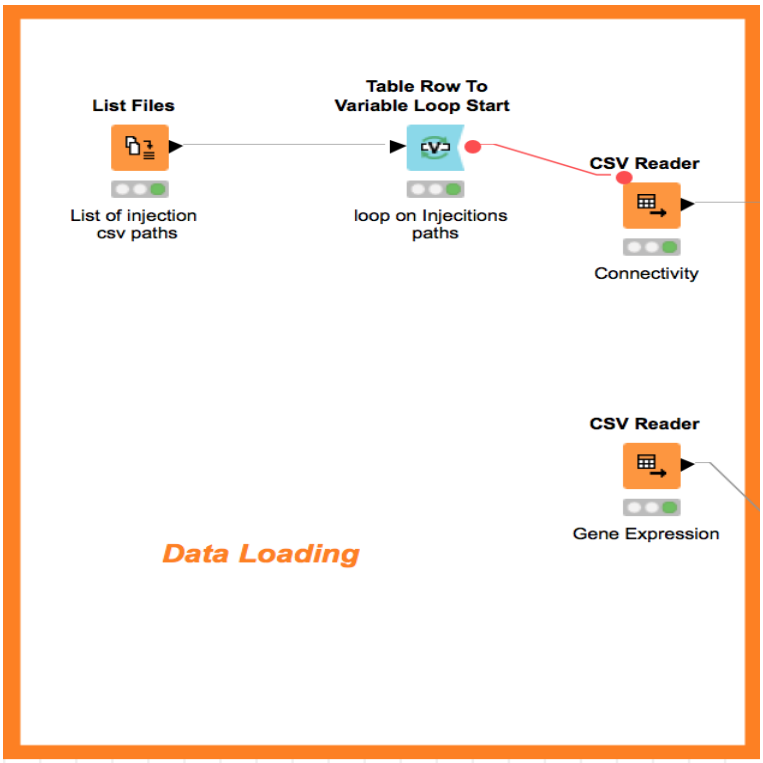


FIGURE 4.8: Data Loading block in Knime's workflow

3. In the loop, the current path is sent to a **CSV Reader** enabling the reading of the correspondent connectivity *csv*.

Data Processing

The nodes in figure 4.9 have been used to process data and organize them in tables.
On the top, the connectivity matrix at the current iteration is processed in order to obtain two tables.
On the bottom, the gene expression matrix containing the energy values for each gene is sent in input to a processing pipeline. As for connectivity data, these data is organized in 2 tables that will be wrote in the SQlite database.
The connectivity matrix, at current iteration, stores injection id, energy projection values of each voxel, Source and Target region IDS. This is sent in input to the following pipeline:

1. the **Row Splitter** node divides rows in two group according to a filter criteria. The table at the upper port (with index 0) includes the rows whose Target Region id is not equal to 0. Voxel annotated for region 0 composes the background of the 3D volume modeling the mouse brain.

2. After this, the filtered table is sent in input to two nodes:

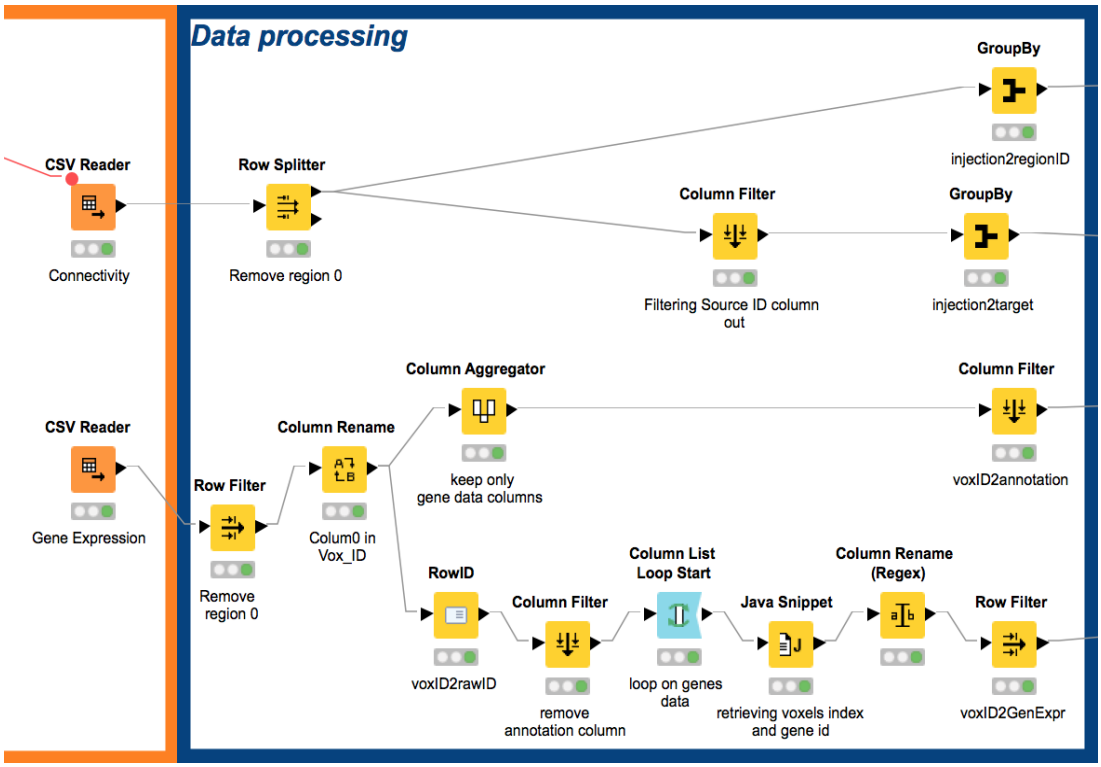


FIGURE 4.9: Data processing block in Knime's workflow

2.a **GroupBy** node that groups the rows of a table by the unique values in the columns reporting Source and injection id. This reduces the table (*injection2regionID*)(fig. 4.10) in a row of two elements (injection id, Source id).

2.b on the underlying branch, **Column Filter** node allows Source id column to be filtered out from the input table. After this, the filtered table is sent in input to a **GroupBy** node that groups the voxel reported on the rows of the table by the Target id. This node aggregate all the voxels belonging to the same Target region by the median of the energy levels associated to each of these voxels. Hence, the output table (*injection2target*) (fig. 4.10) is composed by three columns: injection id, median of the energy obtained for a Target id and its ID annotation.

The gene matrix, instead, stores expression energy values for all the 3318 genes, the gene ids and the region annotation for the single voxel. This table is sent in input to a **Row Filter** node. This filters all the rows containing voxels annotated to region 0 (background). After this, the column containing indexes of voxels is renamed through **Column Rename** node. Then, the output table is processed by two pipelines.
In the series of nodes on the top, the output table is sent in input to:

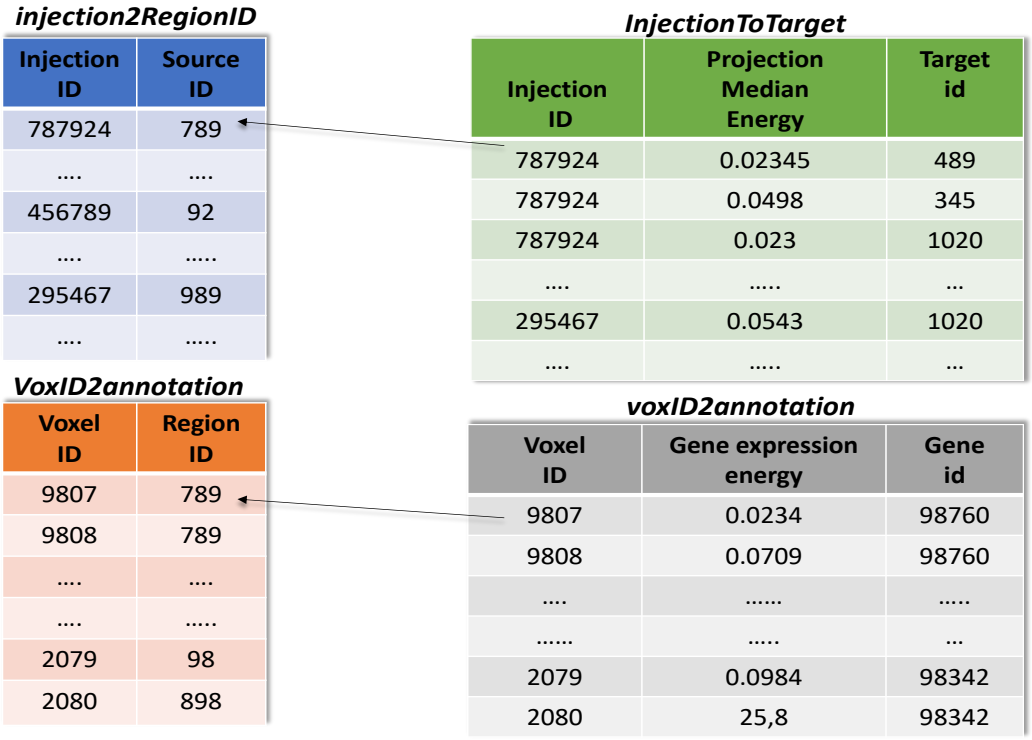


FIGURE 4.10: Tables created through Knime and stored in SQlite database

1. **Column Aggregator** node which groups the selected columns per row and aggregates their cells using the selected aggregation method. This was used to calculate median, mean and sum statistic.

2. The final table (*voxID2Annotation*) (fig. 4.10) was created keeping the voxel id and their annotation through **Column Filter** .

In the series of nodes on the bottom, the output table is sent in input to:

1. **RowId** node that replaces the RowID of the input data with the values of the voxel column. This permits to use the voxel index to indentify the rows. After this, the annotation column is filtered out by a **Column Filter** node.

2. The filter table is sent in input to a **Column List Loop Start** node that iterates on the list of columns that correspond to the arrays of expression energies for each gene. The next following steps are then repeated for each gene expression array:

   2.a **Java Snippet** executes arbitrary java code that generates a table composed by rows reporting for each gene id the voxel id and energy value detected for that voxel.

2.b After this, the voxel with energy value equal to 0 are filtered out through a **Row Filter** node. The output table (*voxID2GenExpr*) (fig. 4.10) is composed by columns reporting gene expression energy value, voxel ID and gene ID.

Database Writing

The database writing block is composed by four **Database Writer** nodes linked to the **SQlite Connector** node. In turn, the connector node creates a connection to a SQLite database located in a specific path.
At each iteration, the four tables, obtained in the previous step, are sent in input to **Database Writer** nodes. These are connected to **Variable loop end** nodes that end the loop started on injection files and genes array in the previous steps. At the end of the first iteration, writing nodes initialize the current tables in the database. In the next iterations, the already existent tables are filled in with the current data.

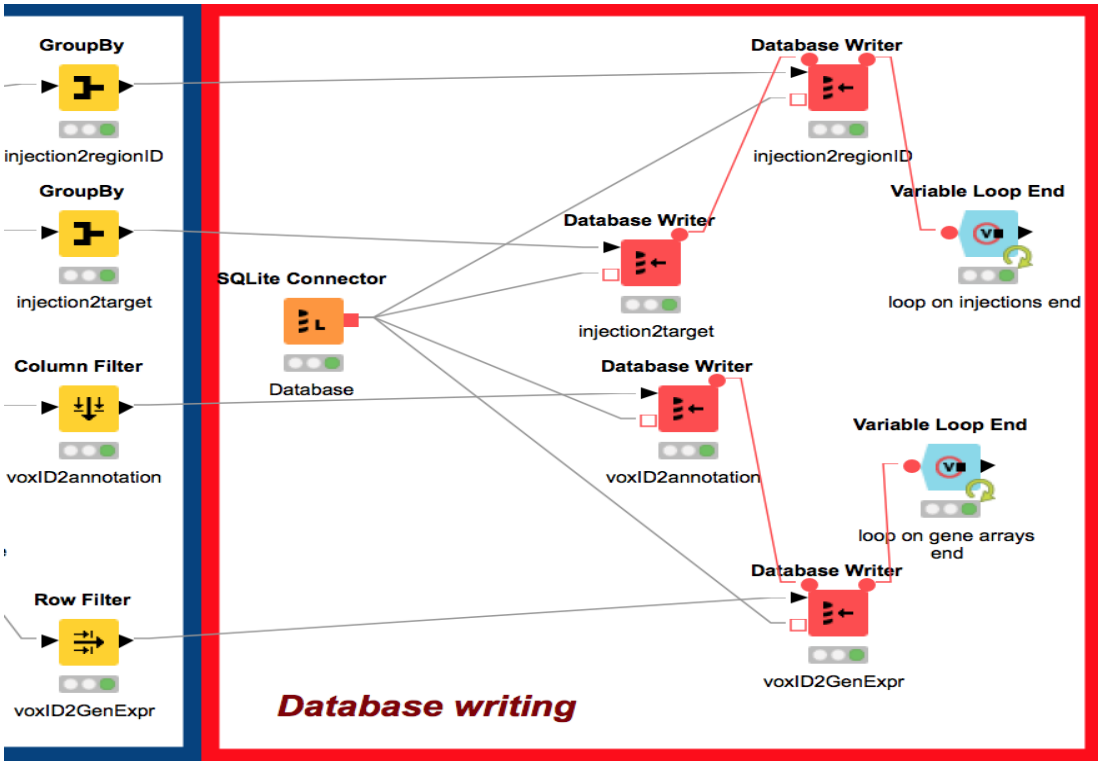


FIGURE 4.11: Database Writing block in Knime's workflow

4.5 Dataset creation

In this section, a detailed description of the steps followed to create the datasets to be analyzed in different experiments, will be given. An ad *ad hoc* pipeline have been implemented to extract gene expression and connectivity data for selected Source-Target regions. The pipeline provides as output vectors containing gene expression profiles detected for two random voxels belonging to the source and target. After its creation, each vector is accompanied by a connectivity value according to specific assigning criteria.
Three dataset with different sizes and features have been created through the implemented pipeline. Each of this have been used to test the informative content of gene expression profiles enabling the exploration of connectivity networks.

4.5.1 *Ad hoc* pipeline for dataset creation

While implementing a machine learning model, the construction of the dataset is a crucial step. A significant dataset has to be created in accordance to the task required to the network. Input examples should be characterized by features through which the NN can learn to predict on unknown data. On top of this, an *ad hoc* pipeline was implemented to combine gene expression and connectivity data to feed a Multilayer Perceptron.

Gene expression vectors construction

First of all, N source-target regions are selected in accordance with their connectivity intensity and the specific analysis to perform.
Gene expression and connectivity data of the selected pairs undergo the following pipeline:

i) For each source-target pair, M voxels belonging to the source region and M voxels to the target regions are selected on the expression gene annotation[1].

ii) For each selected voxel, a vector composed of 3318 elements where each element corresponds to a specific gene provided by AMBA, is created. In particular, each array's cell contains the gene expression energy detected for that voxel. At the end of this step, M vectors representing source's gene expression profile and M vectors representing target's gene expression profile, are obtained.

[1] the M voxels are selected among all the ones belonging to the regions nested in the source and target regions using the dictionary constructed in the previous step.

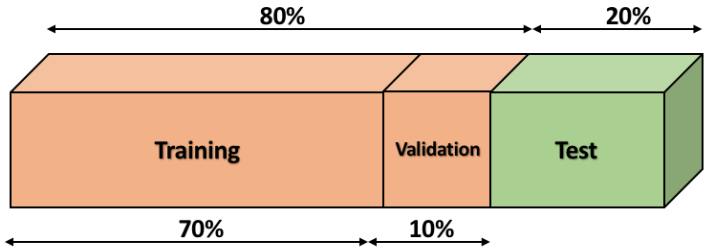


FIGURE 4.12: Dataset suddivision in training, validation and test sets

iii) The dataset is created selecting P combinations among all possible combinations of source and target voxels. In particular, for each combination the gene expression vector corresponding to the source voxel is concatenated with the gene expression vector corresponding to the target voxel. Therefore, the dataset will be made of P vectors.

iv) In the end, a label representing the source-target connectivity value to each of previously created combinations, is assigned[2].

These steps are repeated for all the N source-target regions selected at the beginning.
In the final step of the pipeline, the obtained dataset undergo a normalization process. This scales input vectors individually to unit norm. The normalized dataset is then divided in training, validation and test sets (Fig.4.12). Training set accounts for the 70% of the whole data and test set the remaining 30%. The validation set is made up selecting 10% of pairs in the training set. The N source-target regions are selected following different criteria dependent on the analysis to perform. Therefore, starting from different source-target regions three datasets have been created through the presented pipeline. These have been used to feed a Multilayer Perceptron in three experiments.

Connectivity assigning criteria

The last step of the dataset creation is constituted by the assignation of the connectivity information to the gene expression vectors. The label assigned is value obtained processing the median connectivity values stored in database. When connectivity data have been stored in the database, all the energy values reported for a specific injection ID (experiment) and for a specific Source-Target combination have been aggregated through the median value. However, a region may be site of injection in more experiments. For each of this

[2]the assigned connectivity label is retrieved from Allen or BAMS connectivity data depending on the analysis to be performed

experiment, the axonal projections, produced in the target regions, are stored as energy in a SectionDataSet. Then, if a certain source has targeted the same region in different experiments, more than one energy median will be found in the database for that combination of Source-Target regions.
Therefore, in regression tasks, the following criteria have been applied in order to assign the labels to the vectors obtained for a certain Source-Target combination:

- if any projection energy median > 0.001 in database was annotated for the given Source-Target pair, label "0" was assigned.
- if projection energy median values > 0.001 were annotated for the given source-target combination, the assigned label is the max(medians) or mean(medians) depending on the experiment to perform.

4.6 Predictive Model

The output datasets obtained through the presented pipeline have been used to feed a predictive model. This aims to make prediction about the possible connection between regions recognizing gene expression patterns. A Multilayer Perceptron has been implemented to accomplish classification and regression experiments executed on the datasets created through the *ad hoc* pipeline. The MLP architectures that have provided best performances for multiclass, binary classification and regression, will be illustrated. Details of the datasets and results obtained through the presented architectures will be discuss in the dedicated chapter.

4.6.1 MPL architecture for binary classification

To perform binary and multiclass classification tasks, the MPL architecture (fig.4.13) is composed by an input layer with 64 nodes and two hidden layers with 32 nodes each. The input nodes apply 'sigmoid' activation function on the entries. In the hidden layers, nodes apply 'ReLu' activation function on their inputs. Three Dropout layers are placed after the hidden layers to avoid overfitting phenomenon. This occurs when the MPL specializes on the training set and loses its ability to generalize on the training set. When the error on the validation set starts to increase, the dropout layers "drop out" random neurons. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.
Notably, two options have been given for the activation function of the output layer. In fact, the outputs were calculated by the "sigmoid" function for binary classification and "softmax" for multiclass tasks.

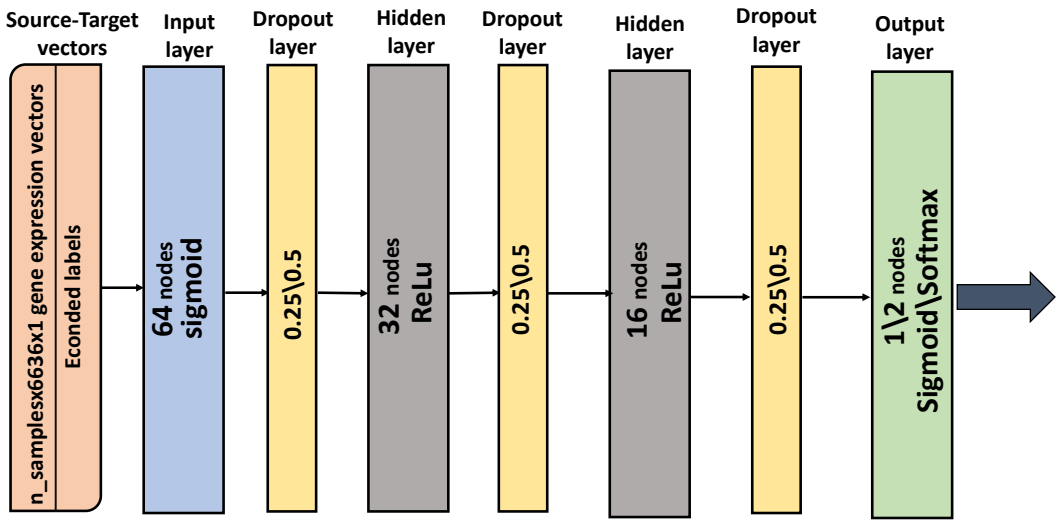


FIGURE 4.13: MLP architecture for classification tasks

4.6.2 MPL architecture for regression tasks

To perform the regression task, the built MPL architecture (fig.4.14) is characterized by nodes whose activation functions return real values ('sigmoid' and 'linear'). The input layer is composed by 128 nodes and it is followed by two hidden layers composed by 64 nodes. Two Dropout layers are placed after the hidden layers to avoid overfitting phenomenon. The output is calculated by the output layer's nodes through 'sigmoid' activation function in order to obtain a real value. In fact, 'sigmoid' returns real values between 0 and 1.0.
When performing regression the error, used to make adjustments, is calculated as mean square error.

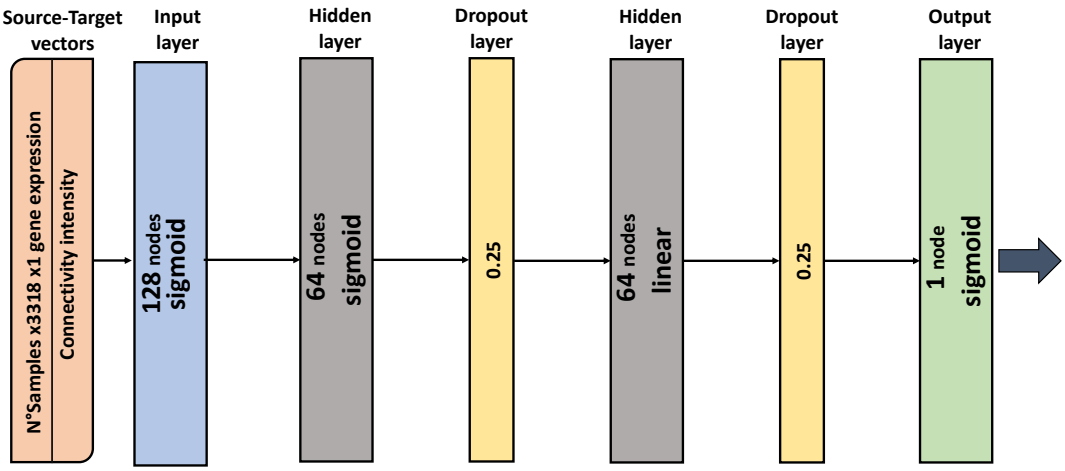


FIGURE 4.14: MLP architecture for regression tasks

Chapter 5

Results

The first part of this chapter is dedicated to the description of metrics to evaluate the performances of the MLP. Subsequently, the three dataset obtained through the *ad hoc* pipeline presented in the previous chapter will be described. The results obtained through the MLP architectures for the experiments, executed on the three dataset, will be illustrated and discussed. In the first part of each discussing section, quantitative details will be given on the dimensions of the dataset used to feed the Multilayer Perceptron. For each experiment, a fine tuning phase was executed in order to select the set of parameters enabling the best performances. The set of the parameters resulted from the tuning and used to train the MLP is reported from each experiment.

5.1 Performance quality metrics

The evaluation of a machine learning model consists in the process of understanding the effectiveness of the algorithm to accomplish a given task. This means to estimate the quality of the predictions provided by the model. Many metrics, highlighting different aspects of the performances, are available to quantify the achievements of a predictive model. Furthermore, due to the different nature of their outputs, different metrics are required for classification and regression.

5.1.1 Classification

A binary classifier predicts all data instances of a test dataset as either positive or negative. Many metrics measures the quality of the performances provided by the classifier. Each of this metrics are applied to the given set of results manipulated. In this evaluation process, the results provided by a classifier are defined and regarded, as follow:

- **true positive (TP)**:
  - Positive prediction

- Label was positive

- **true negative (TN)**:
  - Negative prediction
  - Label was negative

- **false positive (FP)**:
  - Positive prediction
  - Label was negative

- **false negative (FN)**:
  - Negative prediction
  - Label was positive

Using the number of these outcomes, it is possible to achieve a more accurate analysis on the performance of the classifier. In fact different metrics can be evaluated:

- **Accuracy**: is calculated as the number of all correct predictions divided by the total number of the dataset.

$$ACC = \frac{TP + TN}{TP + TN + FN + FP} \tag{5.1}$$

- **Precision** (Positive predictive value): proportion of predicted positives which are actual positive:

$$PREC = \frac{TP}{TP + FP} \tag{5.2}$$

- **Recall** (Sensitivity): proportion of actual positives which are predicted positive :

$$REC = \frac{TP}{TP + FN} \tag{5.3}$$

- **F-score**: F-score is a harmonic mean of precision and recall.

$$F_1 = \frac{2 * PREC * REC}{PREC + REC} \tag{5.4}$$

The described parameters have values in [0, 1.0]. Therefore, the best score is 1.0, whereas the worst is 0.0.
Another powerful means to evaluate a classifier performance is the **confusion**

matrix that reports on columns the predictions provided by the model and on rows the real values.
A graphical quality metric for binary classification is the area under the curve (**AUC**) representing TP rate respect to FP rate (fig.5.1). The resulting curve is called **ROC** and the optimal classifier has AUC=1.0 .

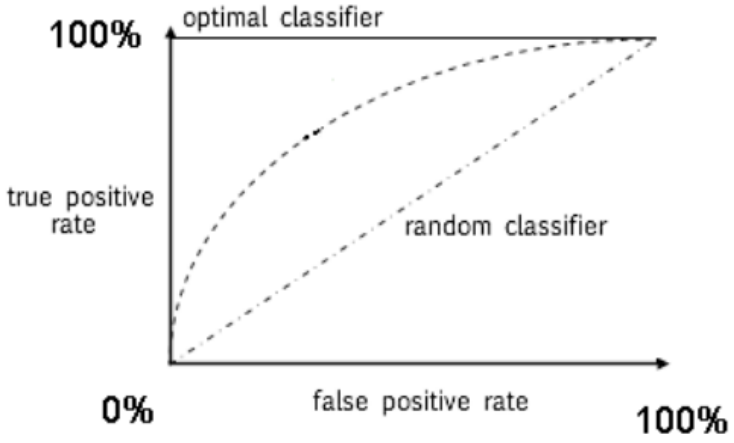


FIGURE 5.1: ROC curve

5.1.2 Regression

Regression refers to functions that attempt to predict a real value output. This type of function estimates the dependent variable by knowing the independent variable. Since the prediction provided by the NN is a real value, it is not possible to evaluate the performances using the metrics presented for binary classification. Therefore, the quality of the predictions provided by the network can be estimated in term of error. In general, an appropriate measure of the error, in regression tasks, is the mean squared error (**MSE**):

$$MSE = \frac{1}{n}\sum_{i}(y_i - y_i^*)^2 \tag{5.5}$$

where: n is the number of the inputs, y_i is the desired output for the i-th input, y_i^* is the prediction provided for the i-th input. Both desired and predicted label are continue values. MSE is usually used to evaluate the performances trend of the model during the training.
To evaluate the performances on test set, a measure of the error normalized in the range of the value to predict is required. The mean absolute percentage error (**MAPE**) provides a the percentage of the error normalized on the range

of the signal:

$$MAPE = \frac{100}{n} \sum_i (\frac{y_i - y_i^*}{y_i}) \tag{5.6}$$

5.2 Datasets description

The following paragraphs are dedicated to the description of the datasets obtained through the *ad hoc* pipeline. Each dataset was obtained selecting different Source-Target combinations to give in input to the pipeline. This way, three datasets have been created and used to perform as many experiments whose results are discussed in the last part of the chapter.

TABLE 5.1: Source-Target pairs for binary classification task

	Source			Target	
Label	**ID**	**Acronym**	**Intensity**	**ID**	**Acronym**
1	362	*MD*	9	44	*ILA*
	385	*Visp*	9	541	*Tea*
	48	*Acav*	8	972	*PL*
	218	*LP*	8	894	*RSPagl*
0	44	*ILA*	1	985	*MOp*
	731	*Orbm*	1	507	*MOB*
	29	*AON*	1	985	*MOp*

5.2.1 Dataset 1: Unconnected/Connected regions

The dataset used to train and test the MLP was created through the *ad hoc* pipeline. The Source-Target pairs have been selected in accordance to the connectivity intensity reported on BAMS. In particular, the created dataset was composed of gene expression and connectivity data from:

- 4 Source-Target region pairs chosen among pairs that on BAMS are reported with high connection's intensity. These belong to the "connected" class encoded with label "1".
- 3 Source-Target region pairs chosen among pairs that on BAMS are reported with minimum connection's intensity. These belong to the "unconnected" class encoded with label "0".

The resulting dataset was composed by approximately 4000 source-target vectors and then subdivided in training, validation and test set.
These were used to feed the MLP in a preliminary experiment. This was performed to verify the hypothesis that gene expression is strongly correlated to connectivity's physical network in the mouse's brain. The first task aims to

verify the existence of gene expression patterns recognizable by the Multilayer Perceptron. Therefore, a binary classification was performed to distinguish between connected and unconnected regions.

5.2.2 Dataset 2: one Source with multiple targets

The dataset was created through the pipeline selecting a source and 3 multiple targets (fig. 5.2). According to BAMS, each target region is connected to the source with different intensities. The pipeline output was a dataset composed by 2200 source-target vectors with their assigned labels. For this task, the connectivity value, obtained averaging the medians of projections data, was assigned as label. This dataset was used to feed a MLP for regression task in a second experiment.
The previous experiment tested the existence of gene expression patterns enabling the discrimination between connected and unconnected regions. This is the foundation on which next tests are based. In the second experiment, the hypothesis that gene expression profiles enable to predict the intensity of connection between regions was evaluated.

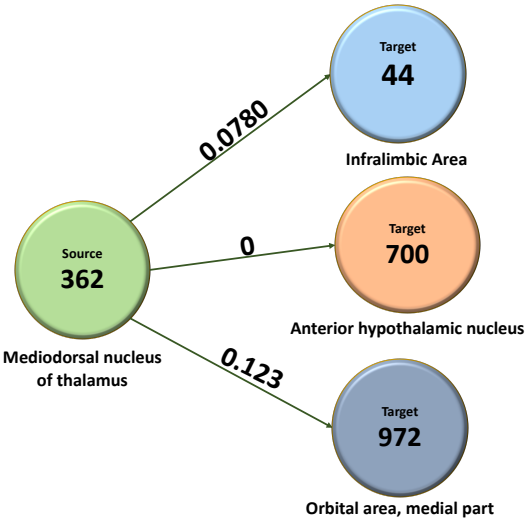


FIGURE 5.2: Source region is paired to three Targets with different connectivity intensity

5.2.3 Dataset 3: Source-Target pairs from macroregions

The dataset was created through the *ad hoc* pipeline selecting the regions, sites of at least one injection, belonging to Cortex and Cerebellum brain areas. These represent important functional regions and then expected to be characterized

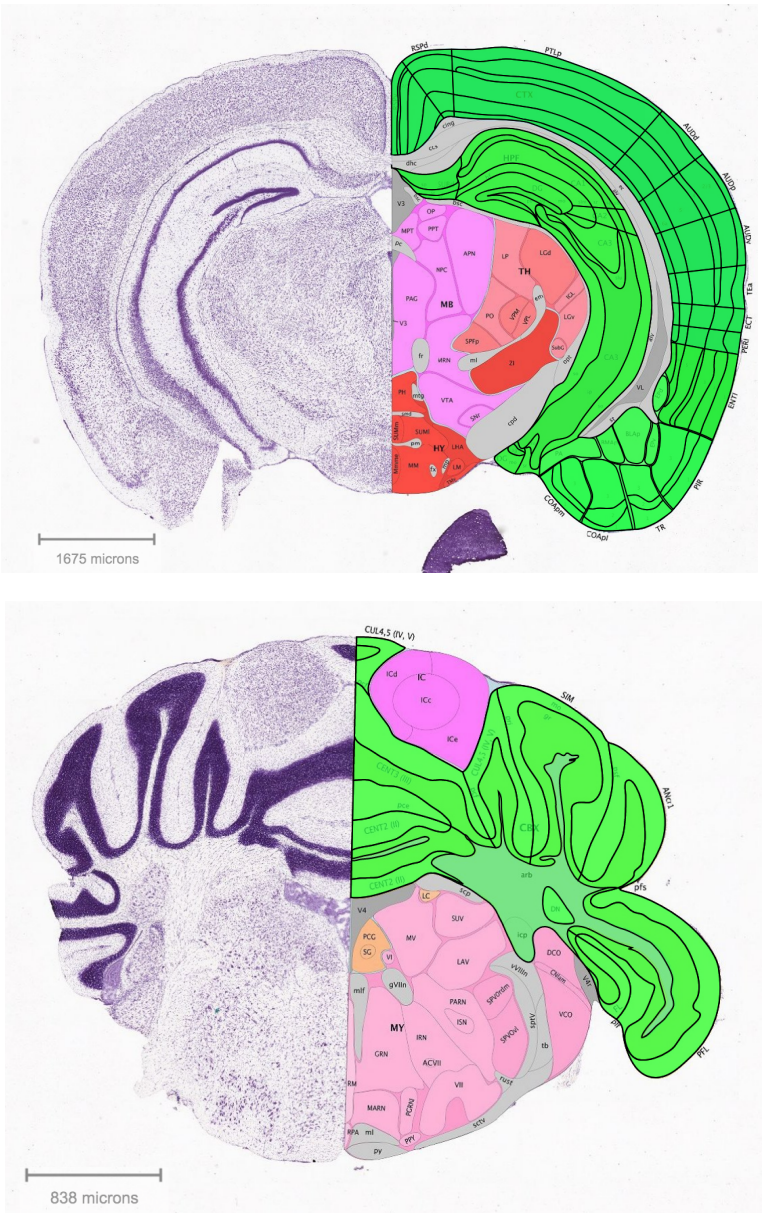


FIGURE 5.3: .
Slices of mouse brain in coronal section. Regions highlighted in green belong to Cerebral Cortex (A) and Cerebellum (B) have been used to build the dataset in phase B [7].

by a dense connectivity network. Thus, the selected regions amount to approximately 58. In particular, eight regions belong to Cerebellum whereas the remaining group composes the Cortex. Hence, the region combinations between Cerebellum and Cortex were more than 3000.
A wild dataset have been created selecting 21 voxels for each possible Source-Target combination. This is composed by source-target vectors created through the implemented pipeline. The total number of the Source-Target vectors was 54'495 with as many connectivity labels associated. These were obtained selecting the mean or maximum value of the connectivity medians reported in the database for the specific Source-Target combination. When any median was found for a region pair, 0 connectivity value was assigned to the corresponding gene expression vector. Depending on the experiment to perform, one between maximum and mean values were chosen.
This dataset has been used to feed the MLP for both classification and regression tasks whose results will be given in the chapter 5.
The experiments of classification and regression executed through dataset 3 aim to consolidate the results on the previous narrow datasets and generalize the performance of the model to a wilder dataset.
Regression task aims to predict the connectivity values assigned to each vector at the output of the *ad hoc* pipeline.
Classification, instead, consists of binary or multiclass tasks in which the Source-Target vectors have been divided in classes in accordance with their connectivity value.
In particular, the connectivity values, calculated as mean or maximum of medians, of the selected Source-Target combinations fall in [0 , 1). Notably, the group that counts most of elements falls in [0, 0.005], whereas only a small fraction has significant values of connectivity. Reasonably, the most accounted group represents the Source-Target combinations that are not connected. Instead, the remaining values constitute the region pairs connected through different intensities.
Therefore, the connectivity distribution of values has led to define connection intensity ranges associated to classes. This way, each Source-Target vector is included in a class in accordance with its connectivity value. In binary classification tasks, Source-Target vectors were divided as follow:

- **Class "0" (unconnected)** : Source-Target vectors with connectivity value equal to 0. This class represents Source-Target regions of the dataset that are not connected.
- **Class "1" (connected)**: Source-Target vectors with connectivity value >0.006. This class is made of Source-Target regions satisfying the condition of connection in spite of the intensity.

In multiclass classification tasks, Source-Target vectors were divided in three classes as follow

- **Class "0" (unconnected)** : Source-Target vectors with connectivity value equal to 0. This class represents Source-Target regions of the dataset that are not connected.
- **Class "1" (connected)**: Source-Target vectors with connectivity value belonging to [0.006,0.05) . This class is made of Source-Target regions of the dataset that are connected.
- **Class "2" (strongly connected)**: Source-Target vectors with connectivity value >0.05. This class is made of Source-Target regions of the dataset that are strongly connected.

5.3 MLP performances on dataset 1

The Multilayer Perceptron has been presented with the gene expression data selected from 7 Source-Target pairs regions. Labels codifying the connected (label "1") and unconnected (label "0") conditions have been assigned to each pair according to the connectivity intensity reported on BAMS. The experiment aims to reveal the existence of gene expression patterns enabling the discrimination between connected and unconnected regions.
The complete dataset, obtained from the selected Source-Target pairs through the *ad hoc* pipeline, was composed of 4027x6636 Source-Target vectors with their labels. Each vector contains the gene expression energy level detected for two random voxels of the source-target pairs regions. The 4027 samples have been divided as follow:

A. **Training phase**: 3.221 (80%) source-target vectors of the complete dataset were divided in training set and validation set. Training set accounts for the 90% of the 3221 source-target vectors of length 6636, resulting in a 2898x6636 matrix. The validation set, instead, was made up of 323 samples, resulting in a 323x6636 matrix.

B. **Testing phase**: the test set was composed by the remaining samples, obtaining a 806x6636 matrix.

The architecture of Multilayer Perceptron implemented to accomplish the binary classification task is illustrated in Chapter 4.

The training phase consisted of *30 epochs* during which the training dataset was propagated in *batches of size 32*. At the end of each propagation, the error between predicted values and desired outputs was calculated by *binary cross_entropy* loss function. In order to minimize the error, *Stochastic gradient descent* (SGD) optimizer updated parameters with a *learning rate* of 0.05 for each training example. The parameters set in the training are reported in Table 5.1.

	Stocastic gradient descent					
epochs	**learning rate (Lr)**	**decay**	**nesterov**	**clip value**	**Loss function**	**batch size**
30	0.05	Lr/epochs	True	0.5	binary cross entropy	32

TABLE 5.2: Training parameters for binary task

After 30 epochs, the implemented MLP reached, in the training phase, 1.0 accuracy with a loss of 0.0160 (fig.5.4). On the test set, the MLP has provided

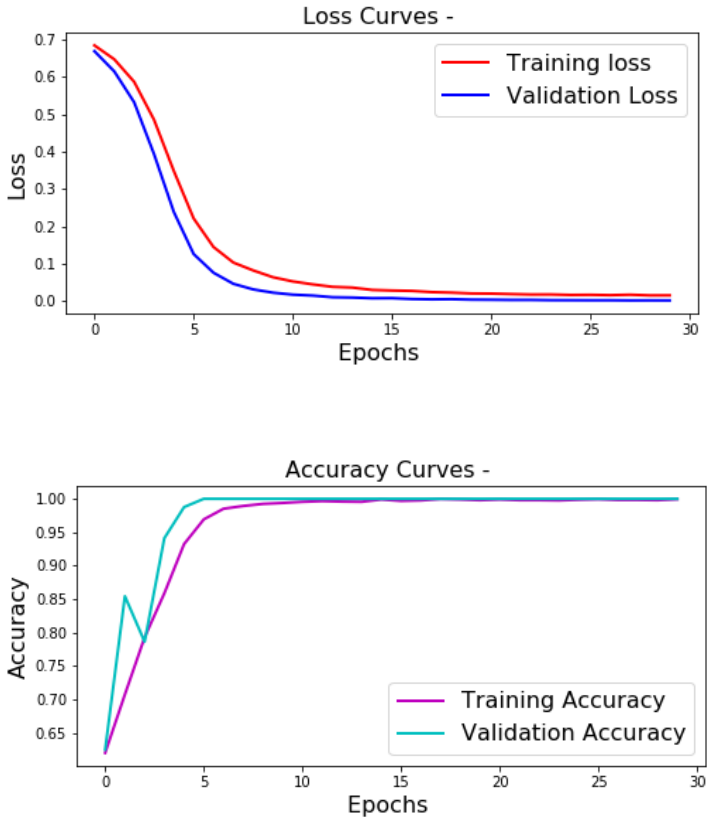


FIGURE 5.4: Training performance curves

accurate results with accuracy, precision, recall and f1-score equal to 1.0 . The obtained results demonstrate that gene expression profiles carry the information enabling the discrimination between connected and unconnected regions. Furthermore, the ROC curve constructed on the predictions obtained on test set has an area of 1.
Notably, the performance of the MLP have been evaluated on a dataset composed of source-target pairs with strong or weak intensity of connection. Hence,

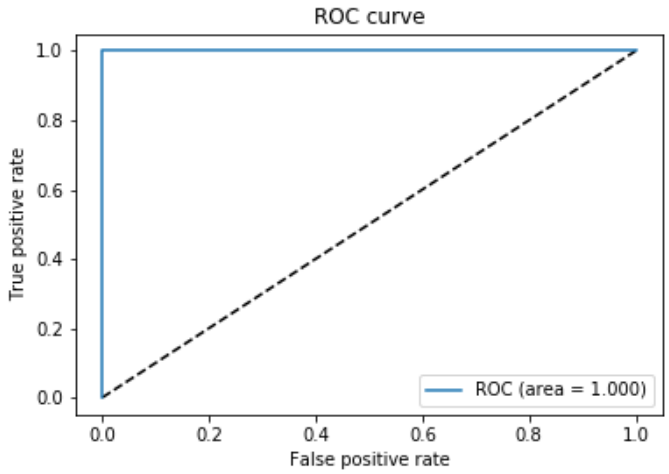


FIGURE 5.5: ROC curve on test set

performances are expected to decrease on a dataset composed of regions with a wild range of connectivity intensities. In light of this, this experiment represents a preliminary analysis on whose results the next experiments have been built.

5.4 MLP performances on dataset 2

In this paragraph, the results of the regression performed through the MLP on the dataset composed of Source and its multiple Targets, will be discussed. As described in the previous chapter, the dataset is composed of the gene expression vectors obtained from the selected voxels of a Source region paired with some of its Targets. The Source region is connected to the Targets with different connectivity intensity.
The label assigned to each Source-Target gene expression vector is the unique value obtained averaging the medians of the projection energies values reported for the specific Source-Target pair. Only median with a value > 0.01 have been taken in account in the average. Hence, labels are real values codifying the intensity of the connection between the Source and its Target. The experiment aims to reveal if gene expression profiles are characterized by meaningful patterns enabling to predict the intensity of connections between regions.
The complete dataset, obtained from the selected Source-Target pairs through the *ad hoc* pipeline, was composed of 2200x6636 Source-Target vectors with the assigned labels. Each vector contains the expression energy levels of the 3318

	RMS prop					
epochs	**learning rate (Lr)**	**decay**	**epsilon**	**rho**	**Loss function**	**batch size**
100	0.01	0.0	None	0.9	mean squared error	32

TABLE 5.3: Training parameters for regression task

genes, detected for two random voxels of the source-target pairs regions.
The 2200 vectors have been divided as follow:

A. **Training phase**: 1584 (80%) source-target vectors of the complete dataset were divided in training set and validation set. Training set accounts for the 90% of the 1584 source-target vectors of length 6636, resulting in a 1584x6636 matrix. The validation set, instead, was made up of 176 samples, resulting in a 176x6636 matrix.

B. **Testing phase**: the test set was composed by the remaining samples, obtaining a 440x6636 matrix.

The architecture of Multilayer Perceptron implemented to accomplish the regression tasks is illustrated in Chapter 4.
The training phase consisted of *25 epochs* during which the dataset has been propagated in *batches of size 32*. At the end of each propagation, the error between predicted values and desired outputs has been calculated by *mean square error* loss function. In order to minimize the error, *RMS prop* optimizer updated parameters with a *learning rate* of 0.005 for each training example.
The parameters set in the training for the regression task are reported in Table 5.2.
In regression tasks, the only metric to check the performance of the MLP during the training is the error calculated by the loss function. After 100 epochs, the error calculated by the mean squared error function reached the convergence.
On test set, the mean squared error (MSE) between the real values (labels) and predicted values has been calculated. The best performance of the trained MLP provided **MSE**=$(1,47 \pm 0,000187) * 10^{-5}$ and **MAPE**=$(5,72\% \pm 0,0040\%)$ on the test set.
However, the MSE and MAPE are a global evaluations of the error committed by the network on all the samples of the test set. Hence, a graphic reporting predicted value against actual value of 20 random test samples is provided in fig 5.7.

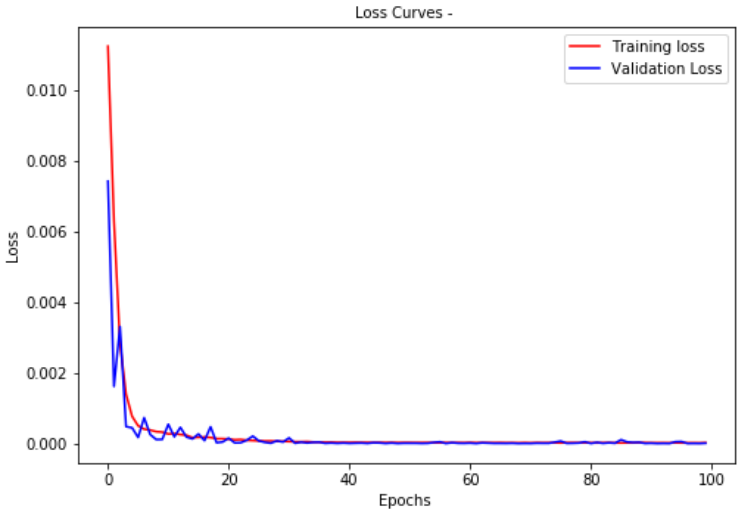


FIGURE 5.6: Training performance

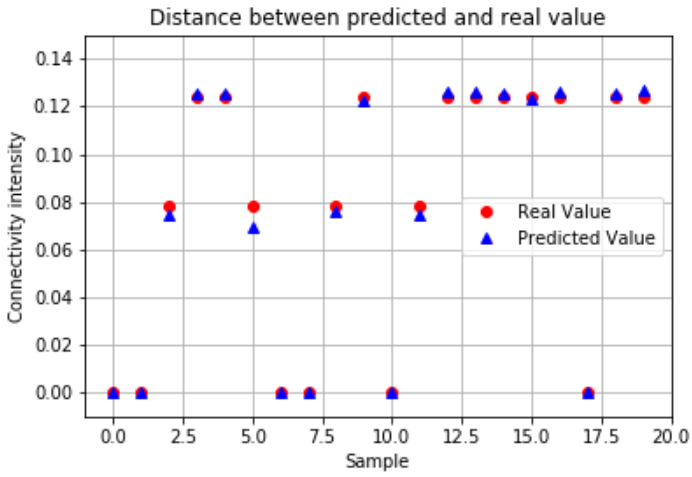


FIGURE 5.7: Distances between real and predicted connectivity intensity for 20 samples of the test set

5.5 MLP performances on dataset 3

In this paragraph, the results obtained through the MLP on the dataset 3 composed of Source-Target pairs from Cortex and Cerebellum, will be discussed. As described in the previous chapter, the dataset 3 is composed of 54'495 gene expression vectors obtained from random voxels selected from the Source-Target combinations of Cortex and Cerebellum.
Two phases composes the experiment:

A) In phase A, regression and multiclass classification were performed through the architectures shown in fig 4.13. The Source-Target vectors are accompanied by the **mean** of the medians reported in the database for the specific Source-Target combination.

B) In phase B, multiclass and binary classification were performed through the architectures shown in fig 4.14. The Source-Target vectors are accompanied by the **maximum** of the medians reported in the database for the specific Source-Target combination.

5.5.1 Phase A

In phase A, the predictive model was trained and tested on dataset 3 using the architectures previously shown. This phase consists in two tasks:

i) Regression task: the MLP was required to predict the connectivity value obtained as mean of the connectivity medians.

ii) Multiclass classification task: the dataset 3 were divided in 3 classes as described in the previous section. The MLP was required to assign each Source-Target vector to a class in accordance with connection intensity.

Regression task

The regression task was characterized by a tuning phase of the parameters shown in Table 5.2. However, it was not possible to select a set of parameters providing predictions with a MAPE lower than 86.7%.

Multiclass classification task

In multiclass task, the dataset 3 was divided in 3 classes in accordance with the connectivity values associated. Only 1090 Source-Target vectors of the whole dataset 3 belong to the class of strongly connected. Therefore, to perform the classification, the three classes were balanced as follow:

- **Class "0"**: 2000 source-target vectors with connectivity values equal to <0.006

- **Class "1"**: 2000 source-target vectors with connectivity values in [0.006,0.1)
- **Class "2"**: 1090 source-target vectors with connectivity >0.1 .

The MLP for multiclass classification was trained on 3665 samples, validated on 408 and tested on 1019.
The parameters set in the training are reported in Table 5.3 . The best performances obtained were 0.73 accuracy on training set and 0.68 accuracy on test set. The precision on the strongly connected class was 0.58.

Discussion of the results

In phase A, the connectivity value associated to each of Source-Target gene expression vectors is calculated as the mean of the medians values associated to each Source-Target combination for each experiment. The results show that averaging the connectivity values between the experiments may not provide a value which is representative of the actual intensity of connection. This has led to chose as connectivity value, the maximum of the medians instead of the mean.
Each connectivity experiment is conducted injecting different sites of the mouse brain. In fact, a single injection is usually not enough to produce the entire projection volume. Indeed, some injections may interest portion characterized by neurons that are strongly deputed to the connectivity respect to others. In addition, it is possible that only some portions of a region are strongly connected to another one. In light of this, the connectivity network detected in each experiment seems to be dependent on the position of the injection. Then, the connectivity status between Source and Target regions may be produced injecting a specific portion of the source region.
On top of this, to take in account that some experiments may be less representative for a specific Source-Target combination, due to the injection condition, maximum value of the medians is used as connectivity value. This means to select the experiment characterized by the best spatial conditions.

5.5.2 Phase B

In light of the results obtained in phase A, classification experiments have been designed in order to improve the precision on the connected classes. Then, phase B is constituted by two classification tasks:

i) Multiclass classification: the dataset was divided in three classes in accordance with the connectivity value calculated as the maximum of the medians. Each class encodes an intensity of connection.

ii) Binary Classification: the dataset was divided in two classes codifying the connected (label "1") and unconnected (label "0") conditions.

TABLE 5.4: Training parameters for multiclass classification

	Nadam					
epochs	**learning rate (Lr)**	**decay**	**beta1**	**beta9**	**Loss function**	**batch size**
200	0.002	0.004	0.9	0.999	categorical cross entropy	6

Multiclass Classification task

In phase B, the connectivity value associated to each Source-Target vector is the maximum of the medians reported for the specific region combination. All the samples have been divided in three classes in accordance with their connectivity value. The three classes are composed as follow:

- **Class "0"**: 5000 source-target vectors with connectivity equal to 0.
- **Class "1"**: 5000 source-target vectors with connectivity values in [0.006,0.1)
- **Class "2"**: 2583 source-target vectors with connectivity >0.1 .

Therefore, the dataset was composed of 14583x6636 gene expression vectors with their labels. In turn, the whole dataset was divided in training, validation, test set. Training set was composed of 10499 samples, validation set 1167 samples, test set 2917 samples.
The architecture of Multilayer Perceptron implemented to accomplish the multiclass tasks is shown in 4.6.2 paragraph. The training phase consisted of *200 epochs* during which the dataset has been propagated in *batches of size 6*. At the end of each propagation, the error between predicted values and desired outputs has been calculated by *categorical cross entropy* loss function. In order to minimize the error, *Nadam* optimizer updated parameters with a *learning rate* of 0.002 for each training example. The parameters set in training phase are summarized in Table 5.3.

After 200 epochs, the implemented MLP reached 0.914 accuracy in the training phase and provided 0.764 accuracy on test set. The confusion matrix (Tab 5.4) shows the distributions of the classifieds and misclassifieds among the three classes. The quality metrics calculated for each class are reported in

TABLE 5.5: Confusion matrix for multiclass classification

		Predicted		
		0	***1***	***2***
Real	***0***	772	238	13
	1	133	791	58
	2	9	166	357

Table 5.4.

TABLE 5.6: Quality metrics for multiclass classification

		Quality metrics			
		Recall	***Precision***	***F1_score***	***Accuracy***
Class	***0***	0.755	0.864	0.806	0.763
	1	0.822	0.662	0.733	
	2	0.671	0.834	0.744	

The implementation phase was focused on building a classifier able to recognize the connected regions with a good precision and recall. Despite of the 0.671 recall on strongly connected regions, almost the totality of misclassified vectors (166) of this class ("2") have been assigned to class "1" against the modest group (9) attributed to class "0". Class "1" still represents the regions that are connected even through weaker connections. Therefore, despite of the misclassifications committed in the predictions on connections intensity, the model implemented has provided accurate performances in the recognition of connected regions.

Binary classification task

In light of the results obtained in predicting connection intensities, binary classification task has been designed to further test the capability to recognize connected regions. Then, the goal is to train an MLP providing precise predictions on the class of connected regions. To perform the binary classification task, the dataset 3 was divided in two classes, balanced as follow:

- **Class "0"**: 20'000 Source-Target vectors with connectivity values equal to 0. This class is composed of gene expression vectors obtained selecting unconnected Source-Target region pairs.
- **Class "1"**: 17'136 Source-Target vectors with connectivity values > 0.006. This class is composed of gene expression vectors obtained selecting the connected Source-Target region pairs.

Therefore, the dataset was composed of 37'136x6636 gene expression vectors with their binary labels. In turn, the whole dataset was divided in training, validation, test set. Training set was composed of 26'737 samples, validation set 2516 samples, test set 7428 samples.
The architecture of Multilayer Perceptron implemented to accomplish the binary tasks is shown in 4.6.2 paragraph.
The training phase consisted of *100 epochs* during which the dataset has been propagated in *batches of size 32*. At the end of each propagation, the error between predicted values and desired outputs has been calculated by *binary cross entropy* loss function. In order to minimize the error, *Nadam* optimizer updated

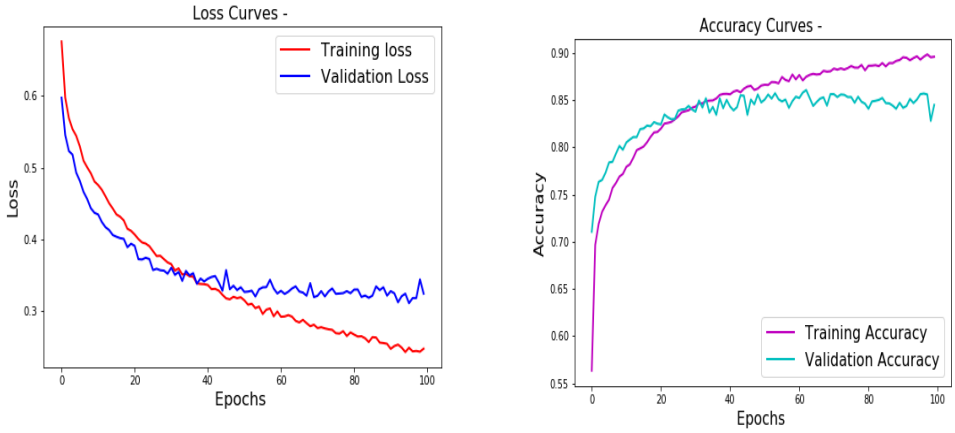


FIGURE 5.8: Training performance curves for binary classification on dataset 3

TABLE 5.7: Training parameters for binary classification

	Nadam					
epochs	**learning rate (Lr)**	**decay**	**beta1**	**beta9**	**Loss function**	**batch size**
100	0.002	0.004	0.9	0.999	binary cross entropy	32

parameters with a *learning rate* of 0.002 for each training example. The parameters set in training phase that allowed to obtain the best performances, are reported in Table 5.6.
After 100 epochs, the implemented MLP reached 0.89 accuracy in the training phase with 0,247 loss (fig. 5.8). The performances and quality metrics on test set are reported in figure 5.9 and Table 5.7.

TABLE 5.8: Quality metrics for binary classification

	Quality metrics			
	Precision	***Recall***	***F1_score***	***Accuracy***
0	0.947	0.755	0.841	0.847
1	0.772	0.952	0.853	

As expected, this model has proved to be precise and accurate on detecting connected regions with a 0.853 F1_score. This last experiment outlines that it is possible to distinguish reliably between connected and unconnected regions, even though improvements have to be made to predict on the intensities of these connections with the same level of accuracy.

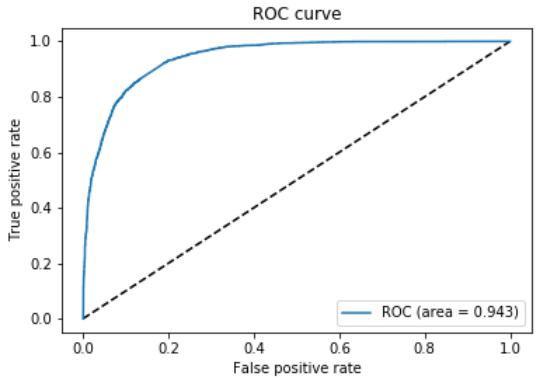


FIGURE 5.9: ROC curve on test set for binary classification on dataset 3

Chapter 6

Conclusions and future works

In this thesis work, the correlation between gene expression profiles of two regions of the mouse brain and their physical connections has been explored through an pipeline designed for this purpose. The pipeline implements the procedures essential to perform this exploration, starting from data retrieving, storing and elaboration to the analysis aiming the investigation of connections between the brain regions. In this regard, the implementing aim of this pipeline is to detect connected pairs with the maximum precision possible.
Selecting different Source-Target combinations, three datasets have been constructed. Dataset 1 and 2, composed of gene expression profiles of selected Source-Target regions, have been used to conduct preliminary analysis to test the potentials of the available data in studying the physical connections. The quality of the predictions obtained through the MLP for regression and classification tasks on these narrow datasets, have proved that gene expression affects the development of connectivity networks. The reliability and accuracy, characterizing the results obtained in this preliminary phase, have encouraged to generalize this assumptions to a wilder dataset.
To achieve this purpose, dataset 3 was created from Cortex and Cerebellum brain areas through the *ad hoc* pipeline. The analysis performed on dataset 3 was composed of two phases and these differentiate in the choice of the connectivity value assigned to each Source-Target combination. In fact, mean and maximum of the medians of the projection energies of each connectivity experiment were assigned respectively in phase 1 and phase 2. The first phase has outlined the limitations of the regression model on a dataset composed of gene expression profiles measured for many different regions. Instead, the predictions obtained by the multiclass classification highlighted a low precision on the class of strongly connected regions.
This has led in the second phase to assign as connectivity value the maximum of the medians since it may be more representative of the connectivity network between two regions. In multiclass task, this hypothesis was revealed to be crucial in the improvement on the precision and recall on the strongly

connected regions with a F1_score of 0.74. This value is affected by a 0.67 recall. Despite of the recall on strongly connected regions, almost the totality of misclassified vectors were attributed to the class of weakly connected regions. In light of this, a last classification experiment with two classes have been designed. This has proved on the available data that analyzing gene expression profile of Source-Target region combination is possible to recognize the connected pairs with a 0.84 F1_score.
This work represents a foundation on which improvements and future works can be build. In fact, gene expression patterns affecting the connectivity networks should be extracted from the trained model. This consists in a feature selection procedure in which the features are the genes of the dataset.
In MLP architectures, the nodes of the layers are fully connected each others. This does not permit to evaluate the weight of the single feature directly on the links of the trained model. However, some techniques, called wrapper-based approach, are able to exploit knowledge of the specific structure of the learning algorithm. These methods consists in training the model excluding a feature from the dataset iteratively. At each iteration, the feature importance is estimated in accordance with the performances obtained excluding that feature. Wrapper methods are more effective than normal filtering feature selection but has higher computational costs [18].
Thus, the individuation of the genes, whose expression are quantified through the FISH, would led to an improvement on the results for both regression and classification.
The model trained using only the gene expression profiles extracted through the wrapper selection, could be used to construct a connectivity reference to explore brain connections in others species.
In this regard, almost the totality of the mouse genes used in this work have homologous in human genes. However, any connectivity data are available to perform an integrative analysis. Thus, the selected genes along with the reference obtained in this work would represent a starting point to the construction of a similar analysis using gene expression profiles obtained through RNA-seq techniques.

Bibliography

[1] M. R. BERTHOLD, N. CEBRON, F. DILL, T. R. GABRIEL, T. KÖTTER, T. MEINL, P. OHL, C. SIEB, K. THIEL, AND B. WISWEDEL, *KNIME: The Konstanz Information Miner*, Springer, 2007.

[2] J. BROWNLEE, *What is the difference between test and validation datasets?*, (2017).

[3] EMANUEL, *An easy-to-read introductory guide to neural networks*, (2018).

[4] A. B. INSTITUTE, Allen Institute for brain science documentation.

[5] ——, *Allen brain atlas api documentation*, Allen Brain Atlas API.

[6] ——, *Allen brain atlas api documentation*, Allen Brain Atlas API.

[7] ——, *Allen reference atlases*, Allen Mouse Brain Atlas.

[8] ——, *Rma query builder*, Allen Brain Atlas API.

[9] ——, *Technical white paper: Allen mouse common coordinate framework and reference atlas*, Allen Institute for brain science documentation, (2017).

[10] ——, *Technical white paper: informatics data processing*, Allen Institute for brain science documentation, (2017).

[11] A. G. JOSH PATTERSON, *Deep Learning*, Oreilly, 2017.

[12] J. H. J. T. M. KANDEL, ERIC R.; SCHWARTZ, *Principles of neural science*, McGraw-Hill, 2000.

[13] KNIME, *Node description*, (2018).

[14] LADYOFHATS, *Complete neuron cell diagram*, (2012).

[15] E. LEIN, *Genome-wide atlas of gene expression in the adult mouse brain*, Nature, (2006).

[16] C. L. LEONARD KUAN, YANG LI, *Neuroinformatics of the allen mouse brain connectivity atlas*, Methods, 73 (2014).

[17] J. M. LEVSKY AND R. H. SINGER, *Gene expression and the myth of the average cell*, TRENDS in Cell Biology, (2003).

[18] J.-B. Y. . K.-Q. S. . C.-J. O. . X.-P. LI, *Feature selection for mlp neural network: The use of random permutation of probabilistic outputs*, IEEExplore, (2010).

[19] A. MEHTA, *First-ever brain atlas completed*, National Geographic News.

[20] T. U. OF SOUT CALIFORNIA, *The bams rat connectome project*, (2013).

[21] L. G. B. O. PRECHELT, *Early stopping- but when?*, Lecture Notes in Computer Science, (2013).

[22] T. H. M. N. RONGQIN KE, MARCO MIGNARDI, *Fourth generation of next-generation sequencing technologies: Promise and consequences*, Human Mutation, (2016).

[23] S. RUDER, *An overview of gradient descent optimization algorithms*, Insight Centre for Data Analytics, (2017).

[24] O. SPORNS, *Connectome*, 2010.

[25] SQLITE, *About sqlite*, SQlite Documentation.

[26] M. P. P. S.T. BOHSALE, MISS. TEJASWINI PATIL, *Sqlite: Light database system*, International Journal of Computer Science and Mobile Computing, (2015).

[27] J.-H. S. VON BARTHELD, CHRISTOPHER S.; BAHNEY, *The search for true numbers of neurons and glial cells in the human brain: A review of 150 years of cell counting*, The Journal of Comparative Neurology, (2007).

[28] M. S. ZHONG WANG, MARK GERSTEIN, *Rna-seq: a revolutionary tool for transcriptomics*, Nat Rev Genet, (2009).