Project Drosophila

Marco Ballarin, Francesco Bianco
marco.ballarin.6@studenti.unipd.it
francesco.bianco.5@studenti.unipd.it

Abstract

In this work we will analyze the hemibrane of a Drosophila, commonly called "small fruit fly". This analysis will be fascinating since we will apply all the techniques proper of the network science field to a neural connectome, thus connecting two fields that can really benefit of each others. One of the most challenging section of this project will be handling the dataset, since it amounts of more than $2 \cdot 10^4$ neurons and $2 \cdot 10^6$ synapses. We will so develop a technique to reduce the network size, monitoring important quantities. We will then analyze the robustness of the network and its evolution along the procedure. Finally, we will pose particular interest in the analysis of the communities, trying to understand if we are able to distinguish the different macro-areas of the brain.

Contents

1 Introduction **3**

2 Coarse graining **4**
2.1 Random aggregation 4
2.2 Clustering aggregation 4
2.3 Metrics evolution 5

3 Robustness **7**
3.1 Random attack 7
3.2 Highest-betweenness attack 7
3.3 Cascading attack 7

4 Communities **9**
4.1 Community detection 10

5 Conclusions **11**

6 Appendix **12**

1 Introduction

In this work we will analyze the structural connectome of a Drosophila, commonly known as "small fruit fly". We will use the hemibrane dataset provided by the FlyEM project[3], which is composed by 21733 neurons connected by 2760725 synapses. The size of the resulting network is overwhelming, and lots of the more advanced analysis are not feasible in this framework. For this reason, borrowing the terminology from statistical physics, we will apply a coarse graining to the network, reducing its size and monitoring important metrics.

The dataset contains a complete set of the neuron cell type, which we will not use, and of the synaptic weights and their brain zones. We will mainly use the python library Networkx[2] for the analysis.

This report will be organized as follows:

1. This brief introduction;
2. A definition of the coarse graining technique that we will use to reduce the size of the network, with the evaluation of important metrics of a graph. We will focus on the results of the coarse graining on the network;
3. The robustness of the network, simulating brain damage.
4. An application of the community detection algorithm, confronting the results with the knowledge that we have in the dataset. In particular, we will try to understand also from a physiological point of view which are the reasons of the results that we will obtain;
5. A conclusion to wrap up the results obtained, presenting some ideas for future works.

2 Coarse graining

It is necessary to define rigorously a way to apply the coarse graining procedure, i.e. our way to combine different neurons in super-neurons. We stress that this procedure is not simple at all, and present different degrees of freedom that can (and will) be chosen with respect to the performances on the data. We start by defining these degrees of freedom:

- the **metric** d_{ij} that we use to define when two different neurons are similar;
- The way in which we aggregate the weights after we combine two nodes.

We will so present two different algorithms that approaches in different ways the second degree of freedom. We stress that the first algorithm is not really working, but it is however instructing analyzing why it is so.

2.1 Random aggregation

We so present the steps of the first algorithms. Calling $n_i^{(\alpha)}$ the i-th neuron at the α-th iteration in the coarse graining algorithm and $w_{ij}^{(\alpha)}$ the links between neurons $n_i^{(\alpha)}$, $n_j^{(\alpha)}$:

1. Pick a neuron $n_i^{(\alpha)}$ at random;
2. Compute the distance $d_i^{(\alpha)} = (d_{i1}^{(\alpha)}, d_{i2}^{(\alpha)}, \dots, d_{iN}^{(\alpha)})$;
3. Combine the two nearest neurons:
$$n_i^{(\alpha+1)} = \left\{ n_i^{(\alpha)}, \min_{d_i} \left[n_k^{(\alpha)} \right] \right\}$$
4. Build the new network connections:
$$w_{ij}^{(\alpha+1)} = \left\{ w_{ij}^{(\alpha)}, \bar{w}_{kj}^{(\alpha)} \right\}$$
where $\bar{w}_{kj}^{(\alpha)}$ are a random subset of the connections $w_{kj}^{(\alpha)}$ such that the density of the network is conserved.
5. Start again from point 1.

This constraint on the density was needed, since it avoids a proliferation of connections. It is, however, a bound that is not easy to fulfill. In particular, it can be implemented by randomly extract the correct number of connections. This does not work in all cases, since not all nodes have the same degree: there are a lot of nodes we very small degree, that have less connections than the ones needed. Another problem of this algorithm is that it is an iterative algorithm and it is so really slow: at each iteration we need to compute all the distances and we can eliminate only a neuron at each iteration.

2.2 Clustering aggregation

The second algorithm is simpler and more effective. We make use of a hierarchical agglomerative clustering technique, i.e. an algorithm that recursively merges the pair of clusters that minimally increases a given linkage distance. This is really similar to the idea developed for the community detection using the dendogram technique. We so perform the following steps:

- Select the final number of neurons N', i.e. the number of clusters for the agglomerative clustering;
- Compute the distance matrix d_{ij};
- Apply the clustering algorithm;

- Combine nodes in the same cluster in a supernode I, which is connected to another supernode J if at least a node in I was connected in a node in J. The weight of the connection is the sum of the weights fo the single nodes, i.e.:

$$W_{IJ} = \sum_{k \in I, l \in J} w_{kl}$$

This algorithm has several advantages over the previous one:

- We do not have to impose the density constraint to avoid the proliferation of the connections;
- We have to compute the distance matrix only once. This can not seem an advantage. In the random algorithm we had to compute the distance from a node $N \cdot (N - N')$, where N is the total number of nodes and N' the final number of nodes after the coarse graining, while in the clustering case we must compute the distance N^2 times. However, if we want to try different N' or run the algorithm multiple times in the first case we should start from scratch for each graining, while in the second case we only need to compute the distance matrix once;
- The clustering algorithm used is optimized for even larger systems, and it is so really fast.

We decided to use two different metrics for this procedure. We only have to remember that we are minimizing a distance in these algorithms, so strongly connected nodes will be nodes considered nearer.

1. The weights, with
$$d_{ij}^{w} = \frac{1}{w_{ij}}$$
This is a really naive distance metric, and we do not expect a really good performance for the algorithm. However, it has the advantage of being already computed;
2. The local page rank, with:
$$d_{ij}^{pr} = \frac{1}{\mathrm{pr}_{ij}}$$
This is a measure often used in network science, and takes into account both the outcoming and incoming links. We expect much better results with this metric.

We present in this report only the results for the clustering algorithm using the local page rank metric. This choice has been done to keep the report readable, focusing only on the interesting results.

2.3 Metrics evolution

We will mainly focus on five different aspects to analyze along the graining procedure, shown in Figure 1:

- The exponent of the degree power-law distribution α, where $p(k) = \beta k^{-\alpha}$. We notice that, starting from a coefficient of 3.6, which is the usual one for this type of networks, we increase this coefficient up to 5.5, where it remains almost constant. This means that the degree distribution become steeper and steeper, showing a really quick change between low-degree nodes and hubs. We are so losing the nodes "in the middle". This is an interesting result, since it confirms what we were discussing in Subsection 2.2: the hubs are not destroyed by the algorithm.
- The density of the network, which is defined as:
$$\rho^{(t)} = \frac{\#\text{ of edges}}{(\#\text{ of nodes})^2} \tag{1}$$
The density of the network increases really slowly initially, but grows quickly when we reach lower number of neurons, after 5000. This behavior is sensible, since we are really zipping the

graph structure at these stages, and when the number of neurons goes to 0 also the number of edges does so, i.e.:

$$\lim_{N' \to 0} \rho = 1$$

- The average degree. In this case the actual average is not the more informative measure. It is really more significant its standard deviation. We can indeed see that the variance of the degree is decreasing along our graining, as we have already understood for the power-law distribution.
- The average degree connectivity, which gives a measure of how much the neighbors of a node with a given degree are connected. We see that, after an initial increase it linearly decreases, with some fluctuation in the variance. This graph is the hardest to interpret, and we will so analyze more in detail the average degree connectivity distribution in the following.

However, in all the measures analyzed we can point out an outlier, with 3000 neurons. We so plot in Figure 2 the degree distribution and the degree connectivity distribution in this case. We can see an anomalous peak in the degree distribution, which fools the automatic computation of the power law degree coefficient, but we can clearly observe that the distribution follows the distribution with coefficient $\alpha = -5$. The degree connectivity distribution follows instead an interesting bimodal distribution.

We then show the evolution of the degree connectivity distribution, in Figure 3. We see that the first hint of the bimodal distribution at 10500 neurons, and it becomes then more visible, as we have also seen in Figure 2. This is again due to the coarse graining, which removes the nodes that are in the middle from the degree point of view, dividing the network in hubs and lowly connected nodes.

We have finally understood, from many metrics, the effect of the coarse graining on the network structure: the elimination of the nodes in the middle.

3 Robustness

There are a number of measures and techniques through which the robustness of a network can be assessed, and such assessment can be more or less meaningful. The main question to answer is: can the coarse graining preserve the graph's properties in such a way that its robustness can be preserved, while being simpler to operate on?

For this project, we tried to assess an answer by empirically experimenting such idea. The tool of choice was TIGER[1] (**T**oolbox for evaluat**I**ng **G**raph vuln**E**rability and **R**obustness), described in [1], which implements several attack and defense techniques.

All the attacking experiments have been conducted consistently. The *Largest Connected Component* (LCC) dimension has been selected as robustness measure: a larger dimension implies a more connected network and thus better robustness. Due to computational time constraints, the attacks were conducted starting from roughly half the original graph size to the coarsest graph, thus from 10500 to 500 neurons, with steps of 1000 aggregated neurons. Every attack removed 30% of the total nodes.

Three attacks were considered: random attack, highest-betweenness removal attack, and cascade attack. We will now review each of them.

3.1 Random attack

At each iteration a random neuron is removed from the graph. As was already known, being the Drosophila connectome a scale-free network with small-world properties, it is quite robust to random node removal. Furthermore, such robustness is kept among the graining, as can be seen in Fig. 4, where the LCC dimension decreases linearly with the number of removed nodes.

The random attack may be used to simulate a normal degrading and failure of neurons that does not involve rejuvenation and creation of new neuronal paths.

3.2 Highest-betweenness attack

The graph global betweenness is computed at the start, and iteratively the neuron with the highest betweenness is removed. This approach leads to the destruction of as many paths as possible and, although the betweenness is computed only once and not at every iteration, it is considered a global-strategy attack, as betweenness is a measure of how much the network is aggregated. Quite surprisingly from Fig. 5, the network is very robust to such attack and the robustness is kept among the graining, with an effect on the LCC dimension indistinguishable from the random attack. This may be due to the fact that the presence of more hubs and super-hubs that are created in the graining procedure is able to cope with catastrophic failures of one third of the main pathways, with neurons "promoted" to an higher betweenness once other bridges are removed.

This attack may be used to simulate the consequences of failure of the main neuronal highways in disrupting the normal neuronal activity.

3.3 Cascading attack

The last attack represents a cascading failure. Let's first consider an example to have a better idea of the mechanism. Consider an electrical grid where a central substation goes offline. In order to maintain the distribution of power, neighboring substations have to increase production in order to meet demand. However, if this is not possible, the neighboring substation fails, which in turn causes additional neighboring substations to fail. The end result is a series of cascading failures, i.e., a blackout.

For this attack, we have made some strong, worst-case assumption: the net has redundancy parameter $r = 0.1$, i.e., 10% of the network is dedicated to redundancy; the (normalized) starting load of each neuron is $l = 0.8$; the maximum capacity of each node before failure is $c = 0.2 \times N$, with N total number of nodes. Furthermore, the attack is conducted like the previous one, but the cascade is made by removing by strictly looking for the highest-betweenness neighbor of the removed node.

[1] available at https://github.com/safreita1/TIGER

Again, as can be seen in Fig. 6 the results are quite consistent across the graining. Furthermore, due to the creation of super-hubs in coarse graining, roughly between 19000 and 2000 nodes and that disappears below the latter number, such procedure is destructive very quickly, as it consistently attacks first the hubs, shattering the LCC within very few iterations. This kind of behavior is nonetheless expected and in line with the scale-free, small-world characteristic of the net.

This attack may be used to simulate the progressive failure of a whole neuronal region, due to heavy hits that cause brain damages, or degenerative illnesses.

Final remarks. Although a more exhaustive examination of the coarse graining procedure should be applied, considering more attacks and measures, we can positively say that a first answer to the question is yes, the coarse graining procedure seems able to preserve the graph's robustness properties.

4 Communities

We analyze here the community detection task. As we already state we have the synapses location is an entry of the dataset. However, certain synapses had different locations. We so decided to select the simplest path possible, and assign to the synapse i the community (location) C_i if the relative entry has the highest weight value. We present now the different zones of the drosophila brain:

1. **AL**: antennal lobe. The antennal lobe is the primary olfactory brain area, and it is a sphere-shaped deutocerebral neuropil in the brain that receives input from the olfactory sensory neurons in the antennae and mouthparts;
2. **CX**: Central complex. It is the center that is assigned to navigation, sleep, learning, memory, nociception (perception of pain);
3. **GNG**:gnathal ganglia, devoted to taste and feeding;
4. **INP**: Inferior neuropils;
5. **LH**: Lateral horn. It is one of the two areas of the insect brain where projection neurons of the antennal lobe send their axons. The other area is the mushroom body. Several morphological classes of neurons in the lateral horn receive olfactory information through the projection neurons lateral horn, axons of pheromone-sensitive projection neurons are segregated from the axons of plant odor-sensitive projection neurons. In addition, the dendrites of lateral horn neurons are restricted to one of these two zones, suggesting that pheromones and plant odors are processed separately in the lateral horn.
6. **LX**: Lateral complex, similar to the central complex;
7. **MB**: Mushroom body. It is known to play a role in olfactory learning and memory. The mushroom bodies and the lateral horn are the two higher brain regions that receive olfactory information from the antennal lobe via projection neurons;
8. **OL**: Optical lobe. It sits behind the eye and is responsible for the processing of the visual information. It is made up of three layers;
9. **PENP**: periesophageal neuropils;
10. **SNP**: Superior neuropils;
11. **VLNP**: Ventrolateral neuropils;
12. **VMNP**: ventromedial neuropils;
13. **Not Primary**: other zones which are not clearly defined.

Neuropil (or "neuropile") is any area in the nervous system composed of mostly unmyelinated axons, dendrites and glial cell processes that forms a synaptically dense region containing a relatively low number of cell bodies.

In most cases what we need are the communities of nodes, not the one of edges. We so define a simple rule to translate the information from the edges to the nodes:

Node i is in the community C_i if most of its synapses are in the community C_i

We show in Figure 7 the non-weighted adjacency matrix of the full system, where we first order the neurons by their community, and then, inside the community, in a decreasing degree order. We notice that, in general, nodes are not connected inside a community but are mostly connected to other zones. This is particularly true for the neuropils.

We want to see how the communities evolves under the coarse graining, and we so define a simple rule to keep track of it. The synapses between the supernode I and the supernode J after an iteration of the clustering coarse graining is in the community C_l if most of the old synapses between nodes in I and J belonged to the community C_l.

In Figure 8 we show the synapses communities at the initial size and after the coarse graining. We can understand, by recalling our method for the community graining, that the zones that lose density are more sparse, i.e. not a lot of the synapses of the same community connects same areas, while the ones that increases are more strongly interconnected. We so understand that the zones that are more strongly interconnected are the Mushroom body, the Central Complex and the lateral horn. The superior neuropils also increases, but its relative increase is minimal, and by looking at the full time evolution we would see that there is an initial decreasing in the density.

4.1 Community detection

We finally want to analyze how the community detection varies along the coarse graining, and to understand if we are able to recover the given community through the network structure using the Louvain algorithm, which optimizes the modularity of the communities. We choose as resolution parameter 0.7. It is linked to the precision in the community detection, i.e. the number of communities founded is inversely proportional to the resolution. The modularity is defined as:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(C_i, C_j) \tag{2}$$

where A_{ij} is the weighted adjacency matrix, $k_{i,j}$ is the weighted degree of node i, j, m is the sum of all the edge weights in the graph, $C_{i,j}$ are the communities i, j and $\delta(x)$ is the Kronecker delta function.

We take as correct the neuronal community defined in the previous section from the synapses, $C_i^{(cor)}$, and compute the error ϵ_i of the community $C_i^{(alg)}$, computed by the algorithm, as:

$$\epsilon_i = 1 - \frac{\left| C_i^{(cor)} \bigcap C_i^{(alg)} \right|}{\max\left(\left| C_i^{(cor)} \right|, \left| C_i^{(alg)} \right| \right)} \tag{3}$$

where with $|\cdot|$ we denote the cardinality, i.e. the number of element, of a set and with $\bigcap$ the intersection operation. Basically we are looking at the number of shared neurons in the two communities, and then we divide by the largest community. In this way we do not incur in the problem of a big community which fully contains a smaller one having a small error. Notice that the error is defined in $[0, 1]$.

We apply the Louvain algorithm both starting from the correct communities and without previous knowledge. The results are shown in Figure 9. We present the results using confusion matrices, where on the x-axis we have the correct labels and on the y-axis the predicted ones. Notice that we do not have a preferred order in the predicted regions, and it is so important to check the error with all the possible regions. We notice that the two regions more easily detected are the central complex and the mushroom body, which are well detected also in the case of no previous knowledge. We notice that it is more difficult to detect the small communities, but it can be a bias of how we have defined the error. We indeed notice a very interesting feature: with the coarse graining we decrease the error with no previous knowledge. For example, in the starting case we have an error on the central complex of 0.15, which becomes 0.14 with 1700 neurons and 0.06 with 12000 neurons! The error increase again if we grain further, with 0.38 with 7000. Using the correct communities as starting communities we see that the MB and the CX are almost perfectly detected. The others communities are instead not detected particularly well. We only have a partial recognition of VLNP.

What is important in this analysis is that we have highlight two regions of the drosophila brain that are easily detected and modular. This result is also interesting for the function of these two regions: they are both involved in learning and memory. This detail can be interesting for further studying.

5 Conclusions

We can finally say that the coarse graining procedure is a very interesting procedure that let us reduce the size of a network, maintaining some important quantity. Indeed, it can even help in evaluating some of the metrics we used, showing new interesting insight for the network. We wrap up our results, dividing them in the different section of the report.

- The coarse graining procedure using the clustering technique is meaningful, producing interesting results even when we reach less than 1/10 of the original size of the network;
- The coarse graining procedure seems able to preserve the graph robustness properties. This is particularly useful, since it is really time consuming to run such simulations. This means that we can perform meaningful analysis on smaller graph and so deduce the effect on the larger ones, sparing a lot of resources;
- The information on the communities are conserved along the coarse graining, and the more connected community become even more distinguishable after the application of the procedure. We also highlighted an interesting fact: the more distinguishable communities are the ones involved in learning and memory.

We can finally conclude that the procedure developed is an interesting tool, which gives new insights on the studied network. It could be interesting to apply it to other networks, to see if this is a global tool that can help also in other analysis.

6 Appendix

The figures related to the report are displayed here.

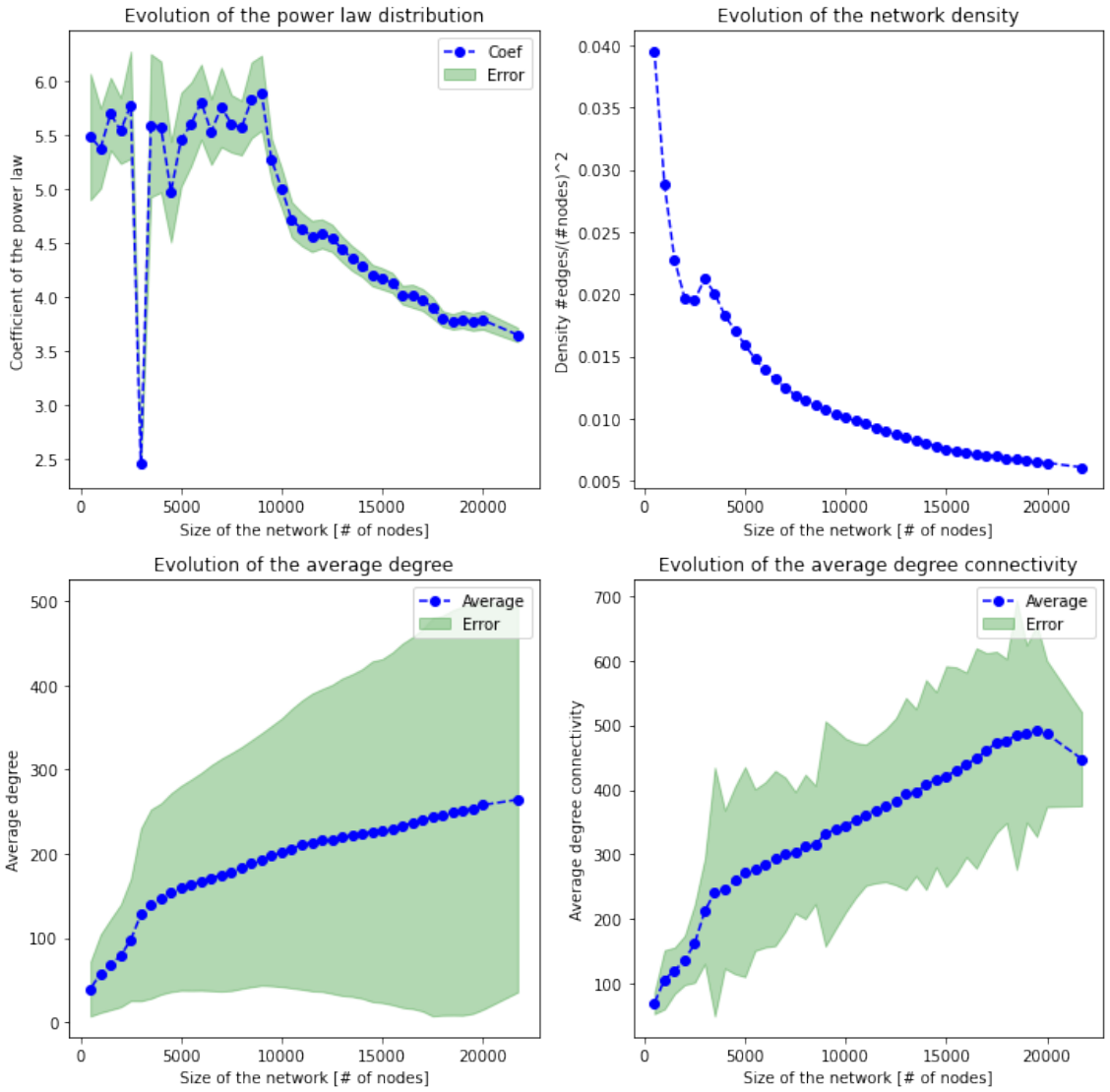


Figure 1: Evolution of different metrics along the coarse graining procedure. We notice an outlier at 3000 neurons, that we will analyze in details. We understand the various metrics does not change too much up to 10000 neurons, which is half of the system.

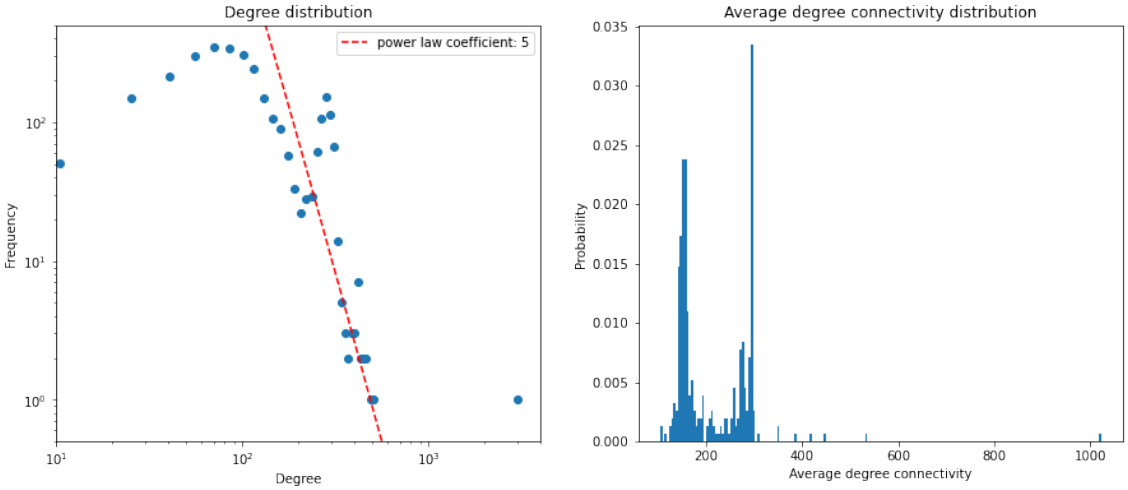


Figure 2: Degree and degree connectivity distribution for the outlier point at $N' = 3000$ neurons.

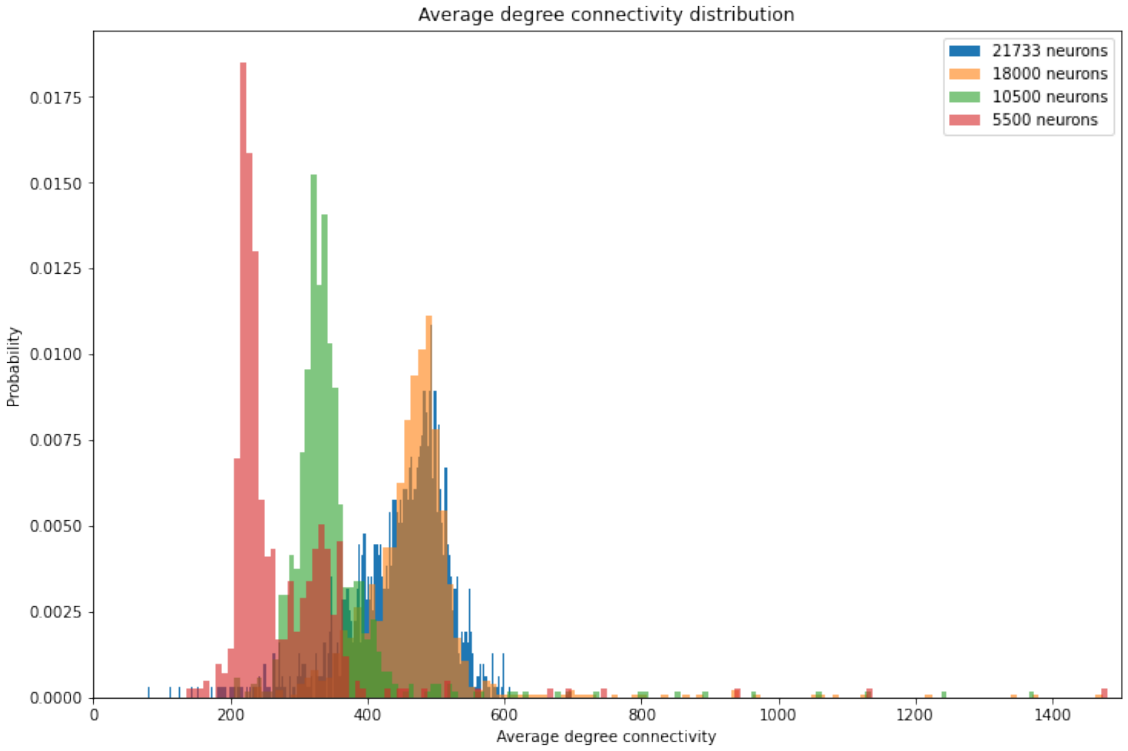
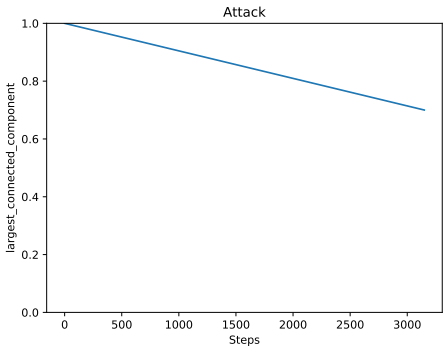
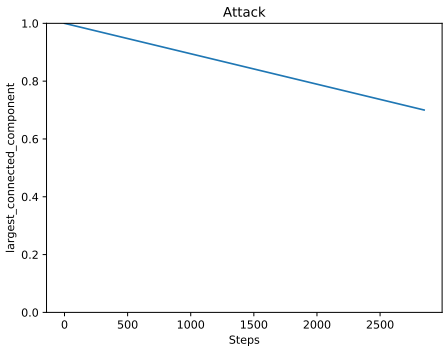


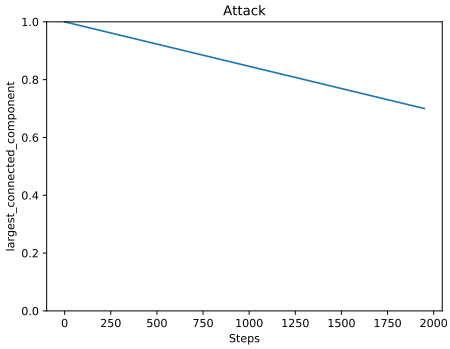
Figure 3: Average degree connectivity distribution for different number of neurons along the clustering coarse graining technique.



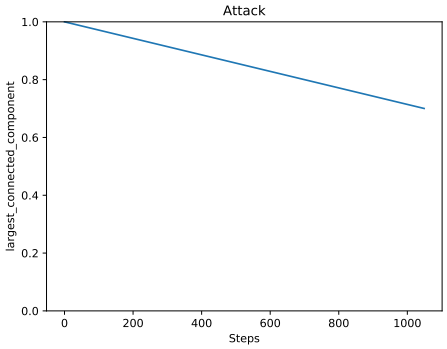
(a) for 10500 neurons



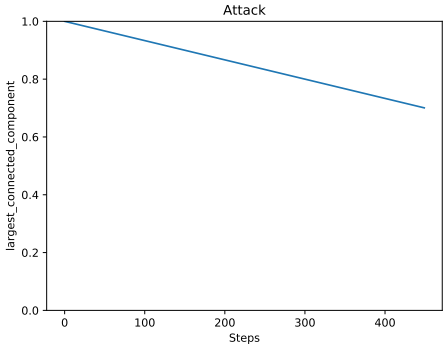
(b) for 9500 neurons



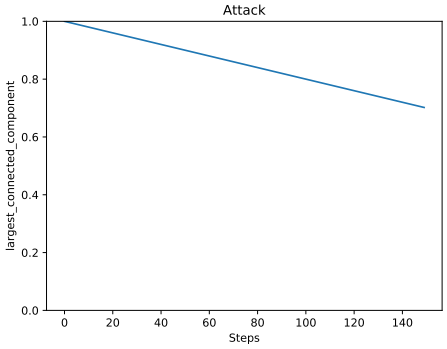
(c) for 6500 neurons



(d) for 3500 neurons

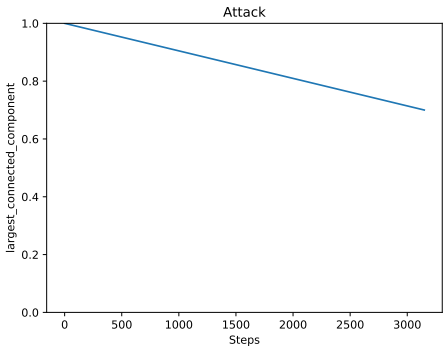


(e) for 1500 neurons

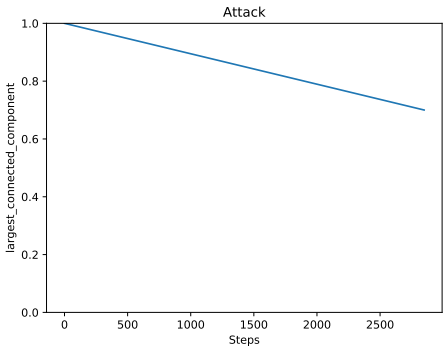


(f) for 500 neurons

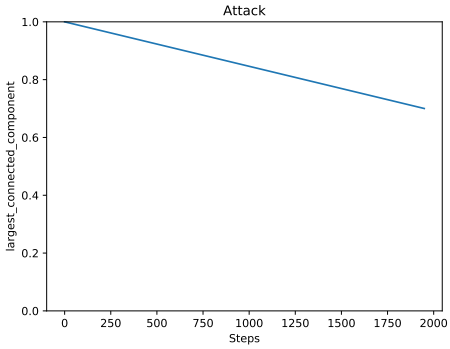
Figure 4: Robustness of the network at various graining scales against random node removal attacks, using the largest connected component as measure.



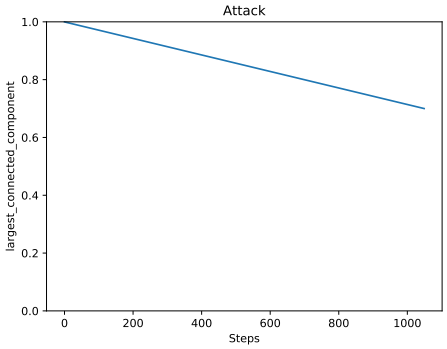
(a) for 10500 neurons



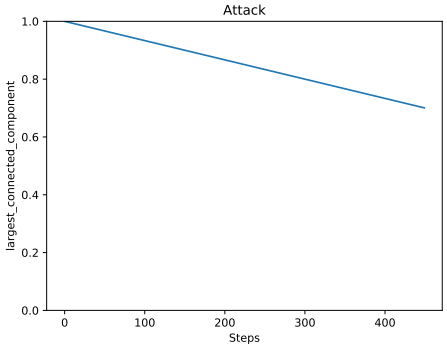
(b) for 9500 neurons



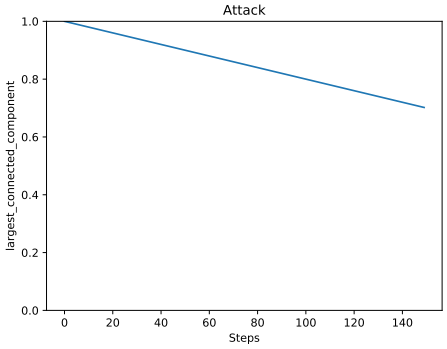
(c) for 6500 neurons



(d) for 3500 neurons

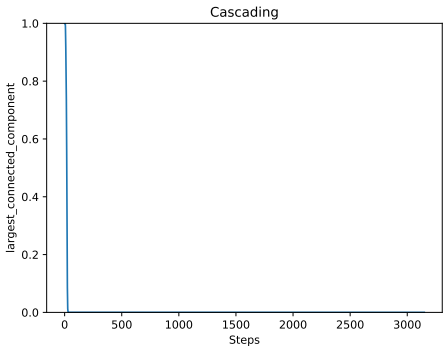


(e) for 1500 neurons

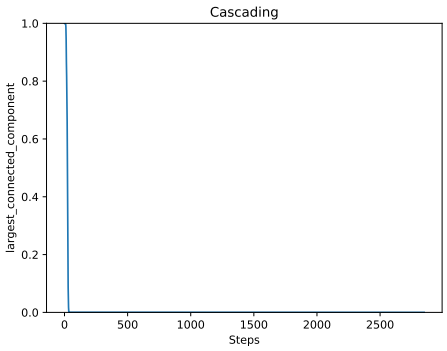


(f) for 500 neurons

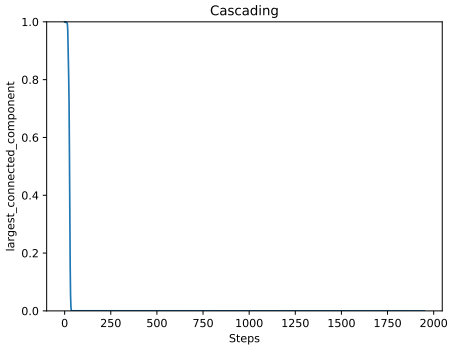
Figure 5: Robustness of the network at various graining scales against removal of highest-betweenness nodes attack, using the largest connected component as measure.



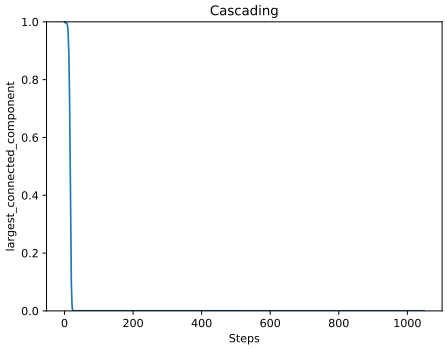
(a) for 10500 neurons



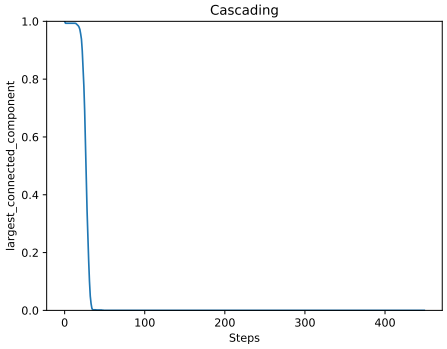
(b) for 9500 neurons



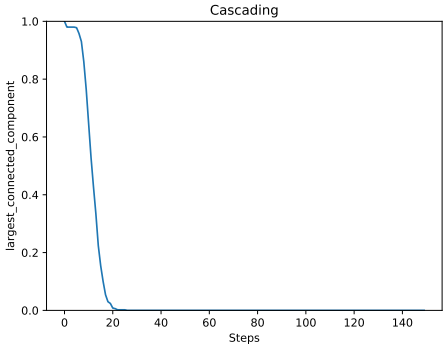
(c) for 6500 neurons



(d) for 3500 neurons



(e) for 1500 neurons



(f) for 500 neurons

Figure 6: Robustness of the network at various graining scales against removal of highest-betweenness nodes cascading attack, using the largest connected component as measure.

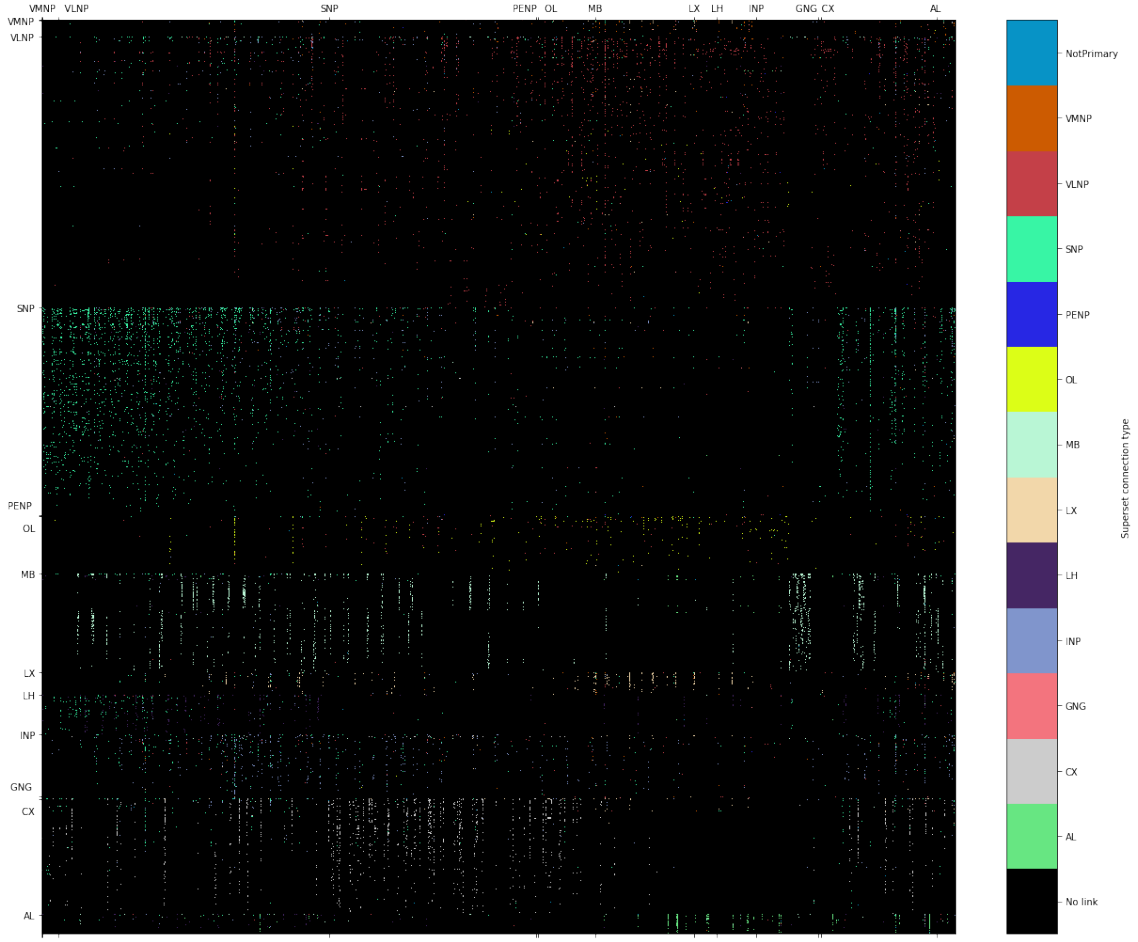
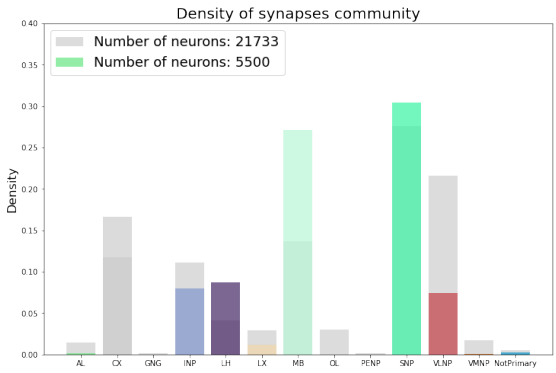
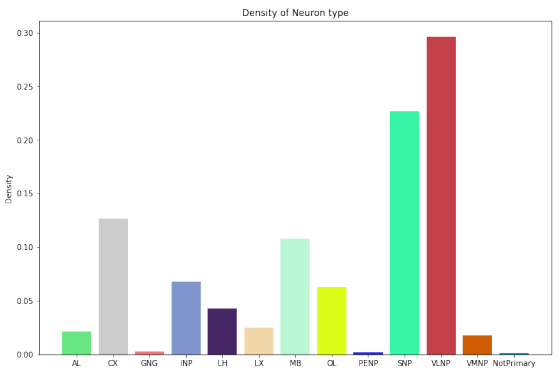


Figure 7: Non weighted adjacency matrix of the full system. The neurons are ordered by community and then displayed in a decreasing degree order. In the y-axis the zone is, starting from its label, up to the lower label. Similarly, for the x-axis the zone starts from the label and finishes to the next label to the right.

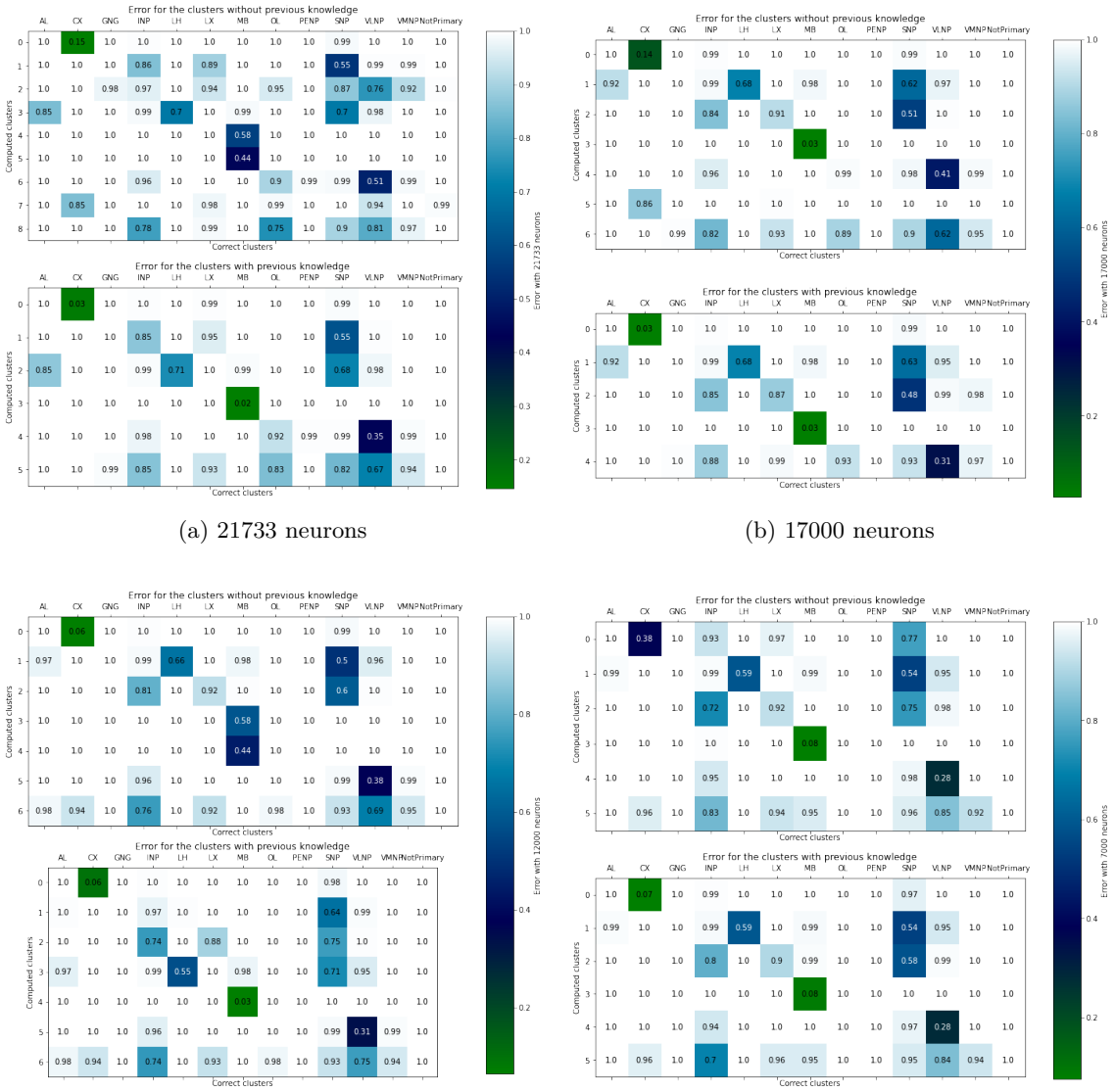


(a) Density of the synapses communities before the coarse graining in gray, and after the coarse graining with colors. We notice that the densities strongly vary in the procedure.



(b) Density of neurons before the graining procedure. The distribution is similar, even if not the same, of the synapses distribution.

Figure 8



(a) 21733 neurons

(b) 17000 neurons

(c) 12000 neurons

(d) 7000 neurons

Figure 9: Confusion matrices for different graining steps, starting forms both without previous knowledge and from the correct communities. On the x-axis we see the correct communities, while on the y-axis the communities detected by the Louvain algorithm. There is no preferred numbering on the communities. The error is defined as in Equation 3.